DELEA: Distributed Ensemble Learning for Energy-Efficient IoT Applications

Benjamin Scott
UCSD & SDSU
bmscott@ucsd.edu
San Diego, USA

Kaila Rosing
UCSD & SDSU
karosing@ucsd.edu
San Diego, USA

Yi-Yang Chen
UCSD
yic079@ucsd.edu
San Diego, USA

Flavio Ponzina
SDSU
fponzina@sdsu.edu
San Diego, USA

***Abstract*—Federated Learning (FL) enables collaborative model training across distributed devices while preserving data privacy, making it particularly valuable for IoT applications. However, communication overhead remains a critical bottleneck, especially for resource-constrained edge devices that must transmit large model parameters over long distances and across networks with limited bandwidth. We present *DELEA (Distributed Ensemble Learning for Energy-efficient Applications)*, a novel framework complementary to compression, client selection, and hierarchical FL, that reduces communication cost by exploiting the multi-instance structure of CNN ensembles. Instead of synchronizing a single large CNN model, *DELEA* maintains an ensemble per client and shares only G of N learners each round, cutting per-round payloads to $\rho = \frac{G}{N}$ of a single-instance baseline while keeping the remaining learners local and personalized. *DELEA* implements a constrained ensemble design, trains learners using a bagging approach to increase diversity, and applies bidirectional aggregation that globally aligns shared learners yet preserves local specialization. Across IoT-style embedded Machine Learning (ML) workloads, *DELEA* attains 2.0× improvement in energy efficiency with no loss in accuracy compared to single-instance CNN baselines.**

***Index Terms*—Federated learning; Ensemble learning; Energy-Efficient IoT**

I. Introduction

The widespread deployment of AI-powered and connected devices has resulted in an enormous amount of data being produced by embedded systems at the edge of the network. This development has drawn increasing attention to federated learning (FL), a distributed machine learning (ML) paradigm that allows models to be trained collaboratively across multiple devices without exposing their private data. FL is particularly valuable in domains with stringent privacy requirements or that rely on distributed IoT systems for data collection, such as surveillance applications [1], smart transportation [2], and healthcare [3]. In its original form [4], FL comprises two primary components: a central server and a collection of client devices. Each client performs local training using its own dataset and shares only the learned model parameters with the server, thus preserving data privacy. The server then aggregates these parameters to construct an updated global model and transmits it back to the clients.

Despite the advantage of learning from distributed data, energy efficiency remains a key requirement in embedded ML scenarios, where edge devices often operate under severe computational and energy constraints. Local model updates during training epochs and edge-cloud communications are extremely costly for constrained embedded systems and may limit the applicability of this learning paradigm in the real world. Fig. 1 illustrates the energy breakdown of an FL round on a Raspberry PI 5, indicating that either training or communication can dominate FL energy cost in IoT devices, depending on the model architecture. To address this challenge, different solutions have been explored. One line of work focuses on smart client selection in FL rounds [5], [6], [7], with different approaches being developed to determine which clients to involve at each step based on their contribution to the global model update. Another line of work studies network topologies and designs hierarchical network architectures to reduce communication cost at the very edge [8]. Additional solutions include well-known pruning and quantization methods, as well as an early stopping mechanism to limit communication cost [9], [10]. Finally, previous works also studied how to jointly quantify training rounds and local training epochs to minimize overall energy consumption in the local clients [11].

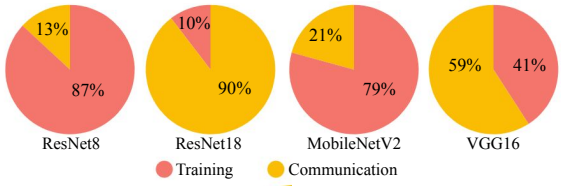


Fig. 1: Energy breakdown of federated learning (FL) on a Raspberry Pi 5, using 10 local training epochs and LoRa-based communication.

This work presents *DELEA*, a novel approach that focuses on reducing the communication cost of FL rounds while improving accuracy and convergence speed. *DELEA* is an approach orthogonal to other frameworks previously proposed in the literature that mainly focus on client selection [12], resource allocation [13], early-stopping [5], or sleep strategies [14]. It targets wide-area IoT systems with geographically dispersed devices operating under limited bandwidth and energy constraints, typically deployed using LPWAN technologies such as LoRa/LoRaWAN.

DELEA leverages ensemble learning, but in a way not yet explored. Instead of leveraging their superior generalization capabilities and hence performing better in non-iid settings [15] and dealing with highly application-specific use cases [16], this work exploits the multi-instance structure of

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

ensembles to limit communication and training cost while maintaining convergence speed and accuracy. Our contributions can be summarized as follows:

- We introduce *DELEA*, a novel framework that exploits the multi-instance structure of ensembles to limit communication overhead between edge and cloud.
- *DELEA* partitions the ensemble instances into local and global, introducing an early stopping training mechanism that optimizes training energy in the edge clients.
- Experiments on different ensemble sizes and benchmarks show that the *DELEA* partitioning, combined with the early stopping training strategy, reduces communication and training energy by up to 28.4% and 21.16%, respectively, leading to a $2.0\times$ efficiency gain compared to single-instance CNNs.
- Thanks to the local learners, *DELEA* also supports personalized federated learning, with client-specific accuracy improvements of 43%.

II. Background and Related works

A. Background

Federated Learning (FL) is a decentralized framework enabling multiple clients to collaboratively train a shared model without exchanging raw data [4]. Clients train locally and send model updates to a central server for aggregation into a global model. This enhances privacy and reduces centralized data requirements, making FL suitable for sensitive domains such as healthcare, finance, and mobile applications. However, FL faces challenges such as non-IID data, client heterogeneity, communication costs, and potential information leakage. To overcome these, research explores communication-efficient IoT methods [14], adaptive aggregation [5], personalization [17], and privacy-preserving techniques [18].

Ensemble Learning combines multiple models to achieve better performance than any single model [19]. By aggregating diverse learners, it reduces variance, mitigates bias, and improves robustness against overfitting [20]. Random Forests [21] are a classic example, while deep learning also benefits from ensembles, such as CNN ensembles in vision tasks [22], trained with varying initializations, architectures, or data subsets to boost accuracy and reliability, albeit with higher computational and memory costs [23].

B. Energy-efficient Federated Learning

Energy efficiency has become a central focus in FL for resource-constrained and large-scale IoT environments. Recent studies tackle this through decentralized frameworks, hierarchical aggregation, compression, device selection, and resource optimization. A decentralized FL framework with Master/Slave clustering and dynamic sleep/wake-up strategies is presented in [14], achieving up to 52% energy savings without a central server. Likewise, a hierarchical FL scheme using intermediate aggregation in a tree topology is proposed in [8], reducing communication distance and training latency by 45.34%. Compression-based methods, such as GWEP [24], combine model pruning and quantization to cut computation and communication costs while maintaining accuracy via error feedback. ETEDS [5] selects devices based on capabilities and applies early stopping once convergence stabilizes, improving both energy and time efficiency. Resource allocation techniques also enhance energy-aware FL. FL-E2WS [13] employs a two-stage optimization framework that maximizes data coverage and schedules communication via mixed integer linear programming, while [12] jointly optimizes client selection and resource allocation, prioritizing clients with more data and better channels. BEFL [25] further considers fairness by balancing energy usage across devices. Using heuristic client selection and reinforcement learning, BEFL improves accuracy while reducing both total and uneven energy consumption.

Compared to these approaches, the proposed *DELEA* framework offers an orthogonal solution that jointly optimizes energy and communication while improving accuracy, and can integrate prior techniques for further enhancement.

C. Federated Ensemble Learning

Ensemble learning strategies have recently gained traction in FL, offering flexible, privacy-preserving, and adaptive solutions for non-IID data, concept drift, and heterogeneous client models. CDF [26] implements a hierarchical FL pipeline leveraging ensemble learning: edge gateways perform ensembling within clusters, while the cloud aggregates parameters globally. Ensemble-GNN [16] applies federated ensemble learning in biomedical domains to improve interpretability and predictive accuracy, decomposing global graphs into subnetworks and aggregating local GNNs via majority voting. ECFL [27] treats the global model as an ensemble of heterogeneous local learners, supporting continual classification under concept drift with flexible client models and drift detection. FedEL [28] enhances transportation mode detection under heterogeneous client conditions, while Fed-Ensemble [15] and FedEL [29] (distinct from [28]) address non-IID challenges by training diverse weak learners locally before ensemble aggregation. Although ensembles are widely used in FL to improve accuracy and handle non-IID data, no prior work has explored leveraging ensemble architectures to optimize energy efficiency in embedded FL systems.

III. *DELEA*

The key intuition behind *DELEA* is that, with an ensemble-based architecture comprising multiple CNN learners, it is possible to limit the number of models shared between edge and cloud, thereby reducing communication and energy overhead. Consider an ensemble of N CNNs: by sharing only $G < N$ models, *DELEA* cuts the data transmitted per FL round to $\frac{G}{N}$ of that in traditional single-instance CNNs. This preserves accuracy while reducing communication, as non-shared learners remain local and enhance ensemble diversity without leaving the device. Moreover, maintaining some local learners offers two advantages. First, they converge faster since they do not require global updates, allowing early stopping to save local compute energy. Second, these local learners specialize in client-specific data, improving model personalization.

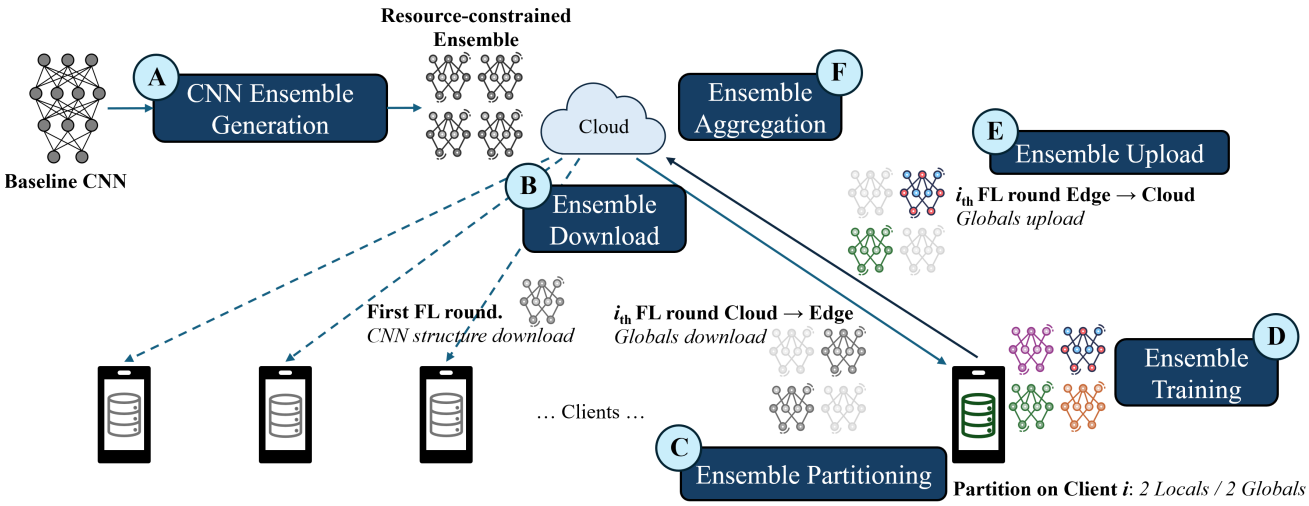


Fig. 2: DELEA overview. The baseline CNN model is first transformed into an equivalent ensemble via [30]. The ensemble is broadcasted to all the clients who implement local/global partitioning to enable early-stop of local learner training and optimize communication by sharing only global learner updates.

Ensemble Design. The discussion above holds only when the size of the generated ensemble does not exceed the size of the baseline single-instance model. Obviously, replicating one or more CNNs to form a typical CNN ensemble will inevitably incur overhead, even when just a subset of the deployed instances are shared between edge and cloud. To avoid communication overhead and save compute and communication power, *DELEA* takes inspiration from the concept of constrained CNN ensembles [31], [30] to combine model pruning and structural replication with an input CNN model to design ensembles that exhibit the same memory and compute requirements as the baseline CNN model (Fig. 2.A). The baseline CNN that would normally be deployed on a device is first pruned by a factor $cmpr$ (i.e., structural filter-level pruning). Then, the obtained pruned structure is replicated $cmpr$ times, creating an ensemble with $N = cmpr$ learners that does not incur memory and compute overhead compared to the initial CNN baseline, hence also preventing communication overhead between edge and cloud. This initial design stage is performed in the cloud and offline, before any FL interaction between cloud and edge clients.

Learner assignment in edge clients. As a first step in FL, the global ensemble model is shared with local clients (Fig. 2.B). Observing that the learners composing the generated ensemble share the same structure and noticing that none of them has been used for training purposes in the local clients yet, only a single learner structure needs to be transferred to the local clients in the first FL round. Clients will then replicate the received structure $cmpr$ times to form the ensemble.

Before local training stages, each client partitions the ensemble into *local (L)* and *global (G)* learners (Fig. 2.C). Only global learners are shared with the clouds during FL rounds, while both global and local learners are trained on the local dataset. Consequently, the partition will affect energy efficiency, which will improve when more learners are kept local. Doing so will also improve model personalization, at the cost of global model training quality, as a lower number of learners will be shared with the cloud.

Let $\mathcal{C} = \{1, 2, \ldots, maxClients\}$ denote the set of FL participating clients, where $maxClients$ is the total number of clients. Each client $c \in \mathcal{C}$ maintains an ensemble of N CNN learners:

$$\mathcal{E}_c = \{\theta_c^{(1)}, \theta_c^{(2)}, \ldots, \theta_c^{(N)}\} \tag{1}$$

where $\theta_c^{(i)}$ represents the parameters of the i-th learner for client c. For each client c, we define a random selection function $S_c : \{1, 2, \ldots, N\} \rightarrow \{0, 1\}$ that partitions the ensemble into two disjoint sets:

$$\mathcal{G}_c = \{i : S_c(i) = 1\} \quad \text{(global learners)} \tag{2}$$

$$\mathcal{L}_c = \{i : S_c(i) = 0\} \quad \text{(local learners)} \tag{3}$$

subject to $|\mathcal{G}_c| = G < N$. This partitioning ensures that only parameters $\mathcal{T}_c = \{\theta_c^{(i)} : i \in \mathcal{G}_c\}$ are transmitted to the central server, achieving a communication reduction factor of $\rho = \frac{G}{N}$ in each client. In *DELEA*, we constrain G to be fixed and common among the $\mathcal{C}$ clients. Although this parameter affects the trade-off between global model quality and client-specific model personalization, this work focuses primarily on improving *efficiency* in FL, and preliminary experiments do not indicate that different partition sizes lead to better accuracy/efficiency trade-off. This happens due to the limited number of learners that are used to compose the *DELEA* ensemble, from two to just four. This number is kept low as increasing the number of learners requires aggressive pruning in the baseline CNN model (i.e., to avoid memory/compute overhead), leading to overall accuracy degradation.

Client Ensemble Training. *DELEA* employs bagging for local training (Fig. 2.D), promoting ensemble diversity by training each learner on a bootstrap-resampled subset of local data. This reduces inter-model correlation and enhances both

generalization and robustness in distributed settings. Formally, let $\mathcal{D}_c = \{(x_j, y_j)\}_{j=1}^{n_c}$ denote client c's dataset. For each learner $i \in \{1, \ldots, N\}$, a bootstrap sample is drawn:

$$\mathcal{B}_c^{(i)} = \{(x_{j_k}, y_{j_k})\}_{k=1}^{n_c}, \tag{4}$$

where each (x_{j_k}, y_{j_k}) is sampled uniformly with replacement from $\mathcal{D}_c$. The learner $\theta_c^{(i)}$ is trained on $\mathcal{B}_c^{(i)}$ by minimizing

$$\theta_c^{(i)} = \arg\min_{\theta} \frac{1}{n_c} \sum_{(x,y)\in\mathcal{B}_c^{(i)}} \ell(f_\theta(x), y), \tag{5}$$

where $\ell(\cdot,\cdot)$ is the loss and $f_\theta(\cdot)$ the CNN model. *DELEA* employs *static bagging*, where $\mathcal{B}_c^{(i)}$ is fixed per round ($\mathcal{B}_c^{(i)}(t) = \mathcal{B}_c^{(i)}(0)$), stabilizing convergence. By fixing each learner's bootstrap subset across rounds, clients optimize stable targets, and the server aggregates updates that point in consistent directions. Unlike the ensemble training approach in [31], where each learner is trained on the entire training set, the adopted method reduces variance, speeds up convergence, and keeps our early-stop signals clean.

Optimizing local training. As previously discussed, each client partitions the learners composing the ensemble into local and global learners. Since the local learners are only trained on the local dataset and do not require FL updates, their training convergence is faster than global learners. *DELEA* leverages this effect to implement an optimization training strategy that monitors the training curve of the local learners and applies an early-stop mechanism to reduce training overhead in the local clients (Fig. 2.D).

Specifically, *DELEA* monitors the validation accuracy of the local learners in each FL rounds and automatically stops training when the accuracy improvement becomes less than 30% of the average gain observed in the first five FL rounds, suggesting that the model converged and that further updates are not significantly improving classification quality. This adaptive approach scales naturally with dataset complexity without requiring manual tuning, as we observed empirically that convergence thresholds should be proportional to initial learning rates across different task difficulties. In this way, only the G global learners require training updates in the latest FL rounds, reducing training cost in the edge clients.

Federated Parameter Sharing. In *DELEA*, the communication protocol leverages the multi-instance architecture of ensemble models and the partitioning described above to reduce communication overhead by restricting parameter exchange to only a subset of the ensemble, termed the global learners $\mathcal{G}_c$ for each client c (Fig. 2.E). This ensures that each client's local learners $\mathcal{L}_c$ remain untouched and specialized to private data distributions, while global learners are aligned through FL collaboration. After each round of local training, each client c transmits only the updated parameters of its global learners $\mathcal{T}_c = \{\theta_c^{(i)} : i \in \mathcal{G}_c\}$ to the central server. The local learners $\{\theta_c^{(j)} : j \in \mathcal{L}_c\}$ are excluded from the parameter sharing process, meaning that their weights are never transmitted to the server. This creates a set of globally consistent ensemble members without requiring full-model communication, preserving diversity across the ensemble. As we will show, this will not significantly affect the accuracy of the global model because of (i) the relatively low number N of learners, which ensures that each client shares a relatively high percentage of its ensemble parameters, and (ii) a sufficiently high number of clients $maxClients$ allows the global model to aggregate enough updates to improve learning. The total communication payload per client is reduced from transmitting all N learners to only $|\mathcal{S}_c| = G$ learners, achieving the communication reduction factor $\rho = \frac{G}{N}$ as previously defined.

A. Aggregation Strategy

The *DELEA* aggregation protocol operates in a bidirectional manner to synchronize only the shared learners across clients (Fig. 2.F). It implements a simple FedAvg aggregation strategy [4], but it integrates this algorithm in the context of ensembles. After each round of local training, each client c transmits only the updated parameters of its shared learners $\mathcal{T}_c = \{\theta_c^{(i)} : i \in \mathcal{S}_c\}$ to the central server, while local-only learners $\{\theta_c^{(j)} : j \in \mathcal{L}_c\}$ remain untransmitted. The server groups updates by learner index, collecting $\{\theta_c^{(i)} : c \in \mathcal{C}\}$ for each global learner index i, and performs federated averaging to produce a global version of each shared learner:

$$\bar{\theta}^{(i)} = \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} \theta_c^{(i)} \quad \forall i \in \mathcal{S}_c \tag{6}$$

To be effective, it is imperative to aggregate learners based on their index i in the ensemble. This ensures consistency and avoids weight updates in non-coherent directions. However, since the partition between local and global learners is random in each client, a sufficiently large number of clients must be selected at each round to ensure each learner in the global model stored in the cloud can be updated by aggregating weights from at least two clients. In practice, the probability of having at least two learners per index for the global ensemble update can be computed as follows:

$$q = 1 - (1-\rho)^{\text{maxClients}} - (\text{maxClients})\,\rho\,(1-\rho)^{\text{maxClients}-1} \tag{7}$$

$$P\left(\forall i \in \{1,\ldots,N\} : X_i \geq 2\right) = \begin{cases} 0, & \text{if } (\text{maxClients})G < 2N \\ q^N, & \text{if } (\text{maxClients})G \geq 2N \end{cases} \tag{8}$$

We formalize the aggregation event by indexing ensemble learners as $i \in \{1, \ldots, N\}$. At initialization (the first FL round), each client $c \in \mathcal{C} = \{1, 2, \ldots, maxClients\}$ draws a subset $\mathcal{G}_c \subseteq \{1, \ldots, N\}$ of size G without replacement and uniformly at random, and keeps this partition fixed across all subsequent rounds. For a fixed index i, define

$$X_i \;=\; \#\{\, c \in \mathcal{C} : i \in \mathcal{G}_c \,\},$$

the number of clients whose fixed global set contains i. Each client independently includes a given index with probability $\rho = G/N$ at initialization, thus $X_i \sim \text{Binomial}(k, \rho)$. The per–index probability that the server can aggregate (receive at least two updates at i) is

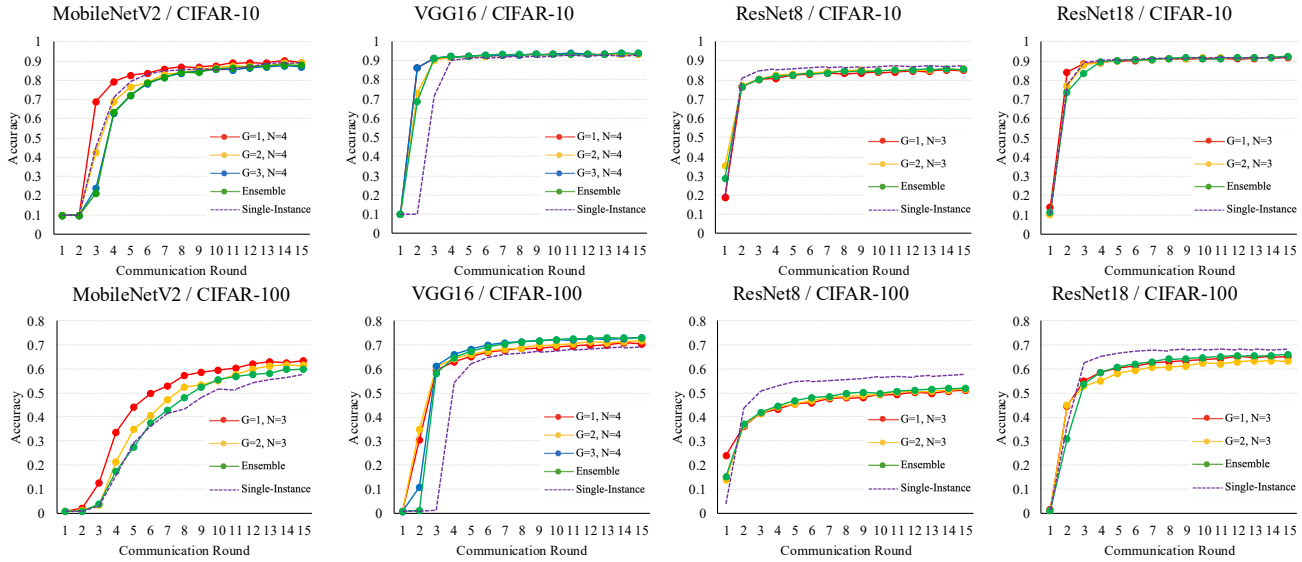


Fig. 3: Convergence analysis of the considered benchmarks, comparing the proposed *DELEA* approach showcasing different local/global partitions, with single-instance CNN baselines.

$$q = \Pr[X_i \geq 2] = 1 - \Pr[X_i = 0] - \Pr[X_i = 1] \tag{9}$$

The probability that *every* index admits aggregation in the same round: $\mathcal{A} = \bigcap_{i=1}^{N} \{X_i \geq 2\}$. A necessary feasibility constraint is $(maxClients)G \geq 2N$. If $(maxClients)G < 2N$, the system cannot place at least two selections on *each* of the N indices, so $\Pr(\mathcal{A}) = 0$. When $(maxClients)G \geq 2N$, events $\{X_i \geq 2\}$ are not independent because each client has a fixed budget of G picks across N indices, so selecting one index leaves fewer slots for the others. Consequently, the probability of the intersection is upper bounded by the product of the marginals:

$$\Pr(\forall i \in \{1, \ldots, N\} : X_i \geq 2) \;\leq\; \prod_{i=1}^{N} \Pr[X_i \geq 2] \;=\; q^N \tag{10}$$

and equals 0 otherwise. Equations (7) and (8) thus summarize, respectively, the per-index aggregation probability and a tight, closed-form upper bound on the round-level probability that all N indices are aggregatable. For $G = 2$ and $N = 4$, the feasibility constraint $(maxClients)G \geq 2N$ gives $maxClients \geq 4$; thus with $maxClients = 4$ clients, the event (all indices receive at least two updates) becomes achievable and strengthens for $maxClients > 4$. Hence, *DELEA* only requires a relatively small pool of selected clients in each FL round to be effective in practice.

B. Sharing global updates back to the clients

Once aggregation is complete, the server then broadcasts the new global aggregated parameters $\{\bar{\theta}^{(i)} : i \in \mathcal{G}_c\}$ back to all clients (Fig. 2.B, after the first FL round). By keeping track of the ensemble partitioning in each client, the server is able to send clients only the updated models for the global learners they shared. Therefore, each client performs a selective update of its local ensemble, updating only the global learners with the new data from the cloud:

$$\theta_c^{(i)} \leftarrow \begin{cases} \bar{\theta}^{(i)} & \text{if } i \in \mathcal{G}_c \\ \theta_c^{(i)} & \text{if } i \in \mathcal{L}_c \end{cases} \tag{11}$$

As previously discussed, this selective broadcast reduces the per-round communication overhead to just $\frac{G}{N}$ (i.e., from updating all N models to updating only G models). When using only a fraction $\frac{G}{N}$ of the ensemble members as global learners, clients benefit from an average reduction of the communication overhead of $\frac{G}{N}$, while suffering a macro-averaged accuracy drop of only $0.57\% \pm 0.04\%$ (at $\frac{G}{N} = \frac{1}{4}$), and $0.08\% \pm 0.06\%$ (at $\frac{G}{N} = \frac{1}{3}$) relative to a *DELEA* design that shares global information from all learners in every round ($G = N$). This design ensures that local-only learners continue to adapt uniquely to the client's data distribution, while shared learners remain synchronized with the global consensus. By restricting communication to shared learners for both uplink and downlink, *DELEA* achieves two goals simultaneously: (i) reduced communication overhead compared to transmitting the full ensemble, and (ii) a balanced trade-off between global generalization (through shared learners) and local specialization (through local-only learners).

IV. Experiments

A. Experimental Setup

We evaluated *DELEA* on the following benchmarks: ResNet8 [32], ResNet18 [32], MobileNetV2 [33], and VGG16 [34] using the CIFAR-10 and CIFAR-100 datasets [35]. In line with [31], we first train *DELEA* configurations with $N = \{1, 2, 3, 4\}$ learners in centralized settings (no FL) and, for each benchmark,

we report the configuration that achieves the highest accuracy. Next, we evaluate all possible combinations of $G = \{1, 2, ..., N-1\}$ in FL experiments. We use the Flower framework [36], partitioning data across $C = \{10, 50\}$ clients, aligned with FL-over-LoRa baseline settings [37] and representative of LoRa/LoRaWAN-scale deployments. Unless stated otherwise, all $|C|$ clients participate in every FL communication round. We apply bagging for multi-learner ensembles ($G > 1$) and standard training for single-instance baselines. IID settings use identical partition indices, while non-IID settings follow a Dirichlet distribution with concentration parameter $\alpha = 0.1$. All experiments use a batch size of 64 and a 0.9 train/test split for consistency.

Energy and performance evaluations are conducted on a Raspberry Pi 5 with a Broadcom BCM2712 quad-core 2.4GHz Arm Cortex-A76 CPU, 8GB LPDDR4X-4267 RAM, and Raspberry Pi OS Bookworm (64-bit). Power data is obtained from the on-board PMIC, sampled once per second via a Python script logging voltage $V_r(t_i)$ and current $I_r(t_i)$ for each rail r at time t_i. Instantaneous power is $P(t_i) = \sum_{r=1}^{NumRails} V_r(t_i) I_r(t_i)$, and the total energy is approximated as $E \approx \sum_{i=1}^{K-1} \frac{P(t_i)+P(t_{i+1})}{2}$, with K samples. Training energy is isolated by subtracting idle consumption: $E_{train} = E - P_{idle} t_{train}$, where P_{idle} is the average idle power and t_{train} is the training duration. Communication energy is estimated using the SX1262 LoRa Connect transceiver [38] under LoRaWAN Data Rate 6 (250 kHz, 235-byte payload), representing a lower-bound energy configuration [39].

B. Experimental Results

Accuracy Analysis. We first validate the accuracy benefits of constrained ensembles in centralized settings and observe an average 2.6% accuracy improvement when compared to single-instance CNNs, in line with what was reported in previous work [31], [30]

Then, we evaluate the accuracy of the proposed *DELEA* framework in comparison to baseline single-instance approaches. The results are presented in Fig. 3. We observe two key advantages. First, the accuracy benefits of ensemble-based implementations extend effectively to FL scenarios, achieving similar global accuracy to single-instance baselines. Second, and more importantly, reducing the number of global learners G participating in cloud-based aggregation does not substantially degrade learning performance, thereby enabling significant reductions in communication overhead while preserving the accuracy. On average, we observe negligible accuracy drops of 0.66% and 0.36% compared to a fully-global *DELEA* design ($G = N$) when only one ($G = 1$) or two ($G = 2$) learners are designated as global, respectively.

Convergence of Local Learners. In alignment with the convergence trends discussed above, we further analyze the convergence speed of both local and global instances. Fig. 4 shows that the local-only learner requires a significantly lower number of epochs to achieve top accuracy. The faster saturation of the local model is expected as it trains on a smaller, client-specific dataset and does not track cross-client updates.

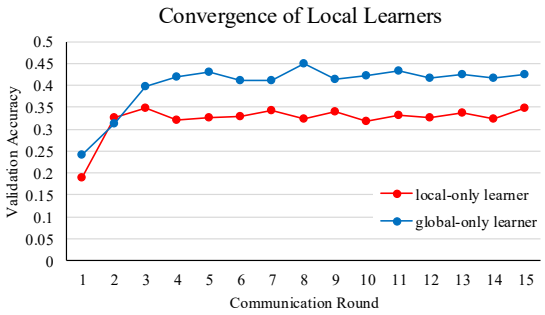


Fig. 4: Validation accuracy of local and global learners in the *DELEA* at different FL rounds.

The global model reaches higher accuracy since it benefits from cross-client aggregation (larger effective dataset). Early-stopping the local learner, therefore, cuts computation without affecting the global accuracy and empirically matches the accuracy of training locals for all rounds.

Energy, Runtime, and Communication Analysis. We now evaluate the energy, performance, and communication efficiency of the proposed *DELEA* framework. Our assessment accounts for both computation and communication energy consumption. For the considered devices and benchmarks, local training and wireless communication consume 0.8-28.4 kJ and 0.1-22.7 kJ per round, respectively, accounting for approximately 54.35% and 45.65% of the total energy expenditure during FL training, on average. Consequently, a solution such as *DELEA* , which optimizes both costs, is particularly advantageous. Fig. 5 illustrates the total energy consumed by local clients to complete the FL training process. We compare the proposed approach against single-instance baselines and, for completeness, include the energy improvements derived from the individual optimizations introduced by *DELEA* (i.e., early stopping of local training and reduced communication load). On average, early termination of local instance training yields up to 21.16% energy savings, while the partitioning of *DELEA* into local and shared instances provides an additional 28.4% reduction in energy consumption, leading to an overall 2.0× efficiency gain[1]. In the case of ResNet8, this reduces the total energy cost of training the single-instance model from 14 kJ to just 7kJ when using *DELEA*, while staying within 1% of the baseline accuracy.

Orthogonality to Prior Work. We integrate *DELEA* with energy-efficient federated learning techniques that operate along axes orthogonal to *DELEA*'s ensemble decomposition. This evaluation allows us to assess whether the gains achieved by *DELEA* compose with existing communication-efficient and client-selection approaches. Results are summarized in Fig. 6.

We first consider and implement FedPAQ [40], a representative communication-efficient FL method that applies stochastic uniform quantization to the model updates exchanged during each communication round, thereby reducing the communica-

[1]We have set G to the smallest value that maintains accuracy within 1% of the fully shared *DELEA* configuration.

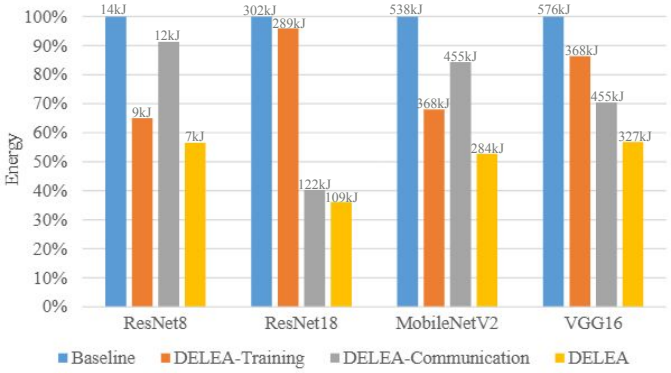


Fig. 5: Total energy consumed per local client to achieve convergence in federated learning. Bar heights are normalized to the single-instance CNN baseline (blue).

tion component of the per-client energy budget while keeping local training at full precision. FedPAQ alone reduces total communication energy by 34.2%, on average, corresponding to a 1.52× efficiency factor relative to a non-optimized baseline. In comparison, *DELEA* alone reduces both training energy, through early stopping of local learners, and communication energy, through local/global learner partitioning, achieving a 49.6% reduction and a 2.00× efficiency factor. When combined, *DELEA* and FedPAQ yield a 62.5% total energy reduction, corresponding to a 2.67× efficiency factor.

We next implement and evaluate FedCS [41], which performs resource-aware partial client selection at each communication round. Because non-selected clients perform neither local training nor communication, FedCS reduces both components of the per-client energy budget in proportion to the participation rate. To isolate the cardinality-reduction effect of selection-based methods, we implement a simplified variant based on uniform random client subsampling at each round. FedCS alone reduces total energy by 50.0%, corresponding to a 2.00× efficiency factor, which is similar to what *DELEA* achieves. When combined, *DELEA* and FedCS achieve a 74.8% total energy reduction and a 3.97× efficiency factor, as client subsampling scales down both the training and communication components already reduced by *DELEA* . These results show that *DELEA* can effectively be used on top of existing FL optimization methods to further increase energy efficiency in IoT clients.

Across all four integrated configurations shown in Fig. 6, combining *DELEA* with either baseline preserves accuracy within 2.5 percentage points of the corresponding *DELEA* reference. In several cases, the integrated configurations match or modestly exceed the accuracy of *DELEA* alone. These results indicate that *DELEA*'s energy savings compose with both communication compression and client selection while maintaining comparable accuracy. Therefore, *DELEA* provides an orthogonal contribution to the two principal families of energy-efficient FL methods considered in the literature.

***DELEA* for Personalized FL.** While the primary considered FL scenario aims to train a high-quality global model, keeping a subset of instances L local on each device, as done

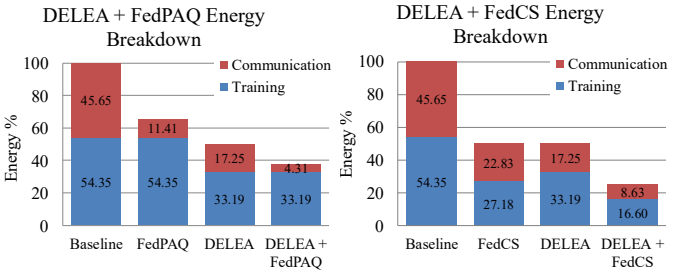


Fig. 6: Energy breakdown for *DELEA* composed with two orthogonal energy-efficient FL methods, FedPAQ [40] and FedCS [41].

TABLE I: Local clients' *DELEA* design vs. global model accuracy on each clients' local dataset in non-IID experiments.

Configuration	*DELEA* Local	Fully-Global
ResNet8 / CIFAR-10	0.866	0.407
ResNet18 / CIFAR-10	0.896	0.565
MobileNetV2 / CIFAR-10	0.839	0.319
VGG16 / CIFAR-10	0.916	0.611
ResNet8 / CIFAR-100	0.575	0.178
ResNet18 / CIFAR-100	0.656	0.228
MobileNetV2 / CIFAR-100	0.573	0.135
VGG16 / CIFAR-100	0.917	0.339

in *DELEA*, allows clients to achieve superior performance on their individual datasets (i.e., personalized FL). This local adaptation does not degrade the global model's quality, which still surpasses state-of-the-art accuracy when trained with *DELEA* (see Fig. 3). We run an accuracy analysis to measure the impact of *DELEA* in the context of personalized FL, and we summarize our results in Table I. It compares the average accuracy of the baseline global model (Fully-Global, in Table I) on each client's test set and that of each client's *DELEA* model (including both local and global learners) in highly non-IID FL settings. Local models share the same global instances but retain locally fine-tuned parameters. Results show that *DELEA* enhances personalization while preserving global performance (see Section IV.B), yielding an average accuracy gain of +43.20% on local datasets compared to the global model. Furthermore, we also compare a fully-local ensemble, trained and evaluated on local data only, against the proposed *DELEA*, and observe that our solution outperforms purely local configurations, improving accuracy by an average of 1.15%. This indicates that FL aggregation offers additional benefits to training quality by leveraging knowledge extracted by a federation of clients. Finally, we observe that training without early-stopping local learners yields negligible accuracy changes also in this study, confirming the robustness of the method.

V. Conclusions

In this work, we introduced *DELEA*, a novel methodology, orthogonal to previous approaches, to improve energy efficiency in low-power IoT devices involved in federated learning. The proposed solution leverages the multi-instance

architecture of ensemble models to implement a local/global learner partitioning that reduces communication overhead. *DELEA* also implements an early-stop training of local instances to further optimize energy consumption in the edge clients. Experimental results show that the proposed method improves energy efficiency by 2.0× compared to the single-instance baselines, with no accuracy loss.

REFERENCES

[1] N. Sugianto, D. Tjondronegoro, and G. Sorwar, "Collaborative federated learning framework to minimize data transmission for ai-enabled video surveillance," *Information Technology & People*, 2025.

[2] R. Zhang, J. Mao, H. Wang, B. Li, X. Cheng, and L. Yang, "A survey on federated learning in intelligent transportation systems," *IEEE Transactions on Intelligent Vehicles*, 2024.

[3] B. Annappa, S. Hegde, C. S. Abhijit, S. Ambesange *et al.*, "Fedcure: A heterogeneity-aware personalized federated learning framework for intelligent healthcare applications in iomt environments," *IEEE Access*, vol. 12, pp. 15 867–15 883, 2024.

[4] 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*, 2017, pp. 1273–1282.

[5] A. Alzailaa, H. R. Chi, A. Radwan, and R. L. Aguiar, "Energy-Efficient Federated Learning-based Device Selection in IoT Environments," in *2023 IEEE 9th World Forum on Internet of Things (WF-IoT)*, 2023.

[6] Q. Li, X. Li, L. Zhou, and X. Yan, "Adafl: Adaptive client selection and dynamic contribution evaluation for efficient federated learning," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024.

[7] R. Tamirisa, C. Xie, W. Bao, A. Zhou, R. Arel, and A. Shamsian, "Fedselect: Personalized federated learning with customized selection of parameters for fine-tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[8] R. R. de Oliveira, K. V. Cardoso, and A. Oliveira-Jr, "Energy-Efficient Hierarchical Federated Learning in Massive Wireless IoT Networks," in *Anais do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2025.

[9] Z. Niu, H. Dong, A. K. Qin, and T. Gu, "Flrce: Resource-efficient federated learning with early-stopping strategy," *IEEE Transactions on Mobile Computing*, 2024.

[10] S. Lee and H. Jang, "Biprunefl:: Computation and communication efficient federated learning with binary quantization and pruning," *IEEE Access*, 2025.

[11] L. Leconte, "Compression and federated learning: an approach to frugal machine learning," Ph.D. dissertation, Sorbonne Université, 2024.

[12] J. Zhao, Y. Ni, and Y. Cheng, "Joint Client and Resource Optimization for Federated Learning in Wireless IoT Networks," *Applied Sciences*, 2024.

[13] R. R. de Oliveira, K. V. Cardoso, and A. Oliveira-Jr, "Improving Energy Efficiency in Federated Learning Through the Optimization of Communication Resources Scheduling of Wireless IoT Networks," *arXiv preprint arXiv:2408.01286*, 2024.

[14] N. T. Javan, E. Zakizadeh Gharyeali, and S. Mostafavi, "An energy-efficient decentralized federated learning framework for mobile-IoT networks," *Computer Networks*, 2025.

[15] N. Shi, F. Lai, R. Al Kontar, and M. Chowdhury, "Fed-ensemble: Ensemble models in federated learning for improved generalization and uncertainty quantification," *IEEE Transactions on Automation Science and Engineering*, 2024.

[16] B. Pfeifer, H. Chereda, R. Martin, A. Saranti, S. Clemens, A.-C. Hauschild, T. Beißbarth, A. Holzinger, and D. Heider, "Ensemble-gnn: federated ensemble learning with graph neural networks for disease module discovery and classification," *Bioinformatics*, 2023.

[17] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE transactions on neural networks and learning systems*, 2022.

[18] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, 2021.

[19] U. Sarmah, P. Borah, and D. K. Bhattacharyya, "Ensemble learning methods: An empirical study," *SN Computer Science*, 2024.

[20] E. Halabaku and E. Bytyçi, "Overfitting in machine learning: A comparative analysis of decision trees and random forests." *Intelligent Automation & Soft Computing*, 2024.

[21] L. Breiman, "Random forests," *Machine learning*, 2001.

[22] L. Nanni, S. Ghidoni, and S. Brahnam, "Ensemble of convolutional neural networks for bioimage classification," *Applied Computing and Informatics*, 2021.

[23] A. Wasay and S. Idreos, "More or less: When and how to build convolutional neural network ensembles," in *International conference on learning representations*, 2021.

[24] P. Prakash, J. Ding, R. Chen, X. Qin, M. Shu, Q. Cui, Y. Guo, and M. Pan, "IoT Device Friendly and Communication-Efficient Federated Learning via Joint Model Pruning and Quantization," *IEEE Internet of Things Journal*, 2022.

[25] Z. Ju, F. Shen, and T. Wei, "BEFL: Balancing Energy Consumption in Federated Learning for Mobile Edge IoT," *arXiv preprint arXiv:2412.03950*, 2024.

[26] Z. Gao, S. Liu, and Y. Zhang, "Cognitive data fusing for internet of things based on ensemble learning and federated learning," *IEEE Internet of Things Journal*, 2024.

[27] F. E. Casado, D. Lema, R. Iglesias, C. V. Regueiro, and S. Barro, "Ensemble and continual federated learning for classification tasks," *Machine Learning*, 2023.

[28] M. M. Alam, T. Ahmed, M. Hossain, M. H. Emo, M. K. I. Bidhan, M. T. Reza, M. G. R. Alam, M. M. Hassan, F. Pupo, and G. Fortino, "Federated ensemble-learning for transport mode detection in vehicular edge network," *Future Generation Computer Systems*, 2023.

[29] X. Wu, J. Pei, X.-H. Han, Y.-W. Chen, J. Yao, Y. Liu, Q. Qian, and Y. Guo, "Fedel: Federated ensemble learning for non-iid data," *Expert Systems with Applications*, 2024.

[30] L. Zhang, O. Gungor, F. Ponzina, and T. Rosing, "E-quartic: Energy efficient edge ensemble of convolutional neural networks for resource-optimized learning," in *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, 2025.

[31] F. Ponzina, M. Peon-Quiros, A. Burg, and D. Atienza, "E 2 cnns: Ensembles of convolutional neural networks to improve robustness against memory errors in edge-computing devices," *IEEE Transactions on Computers*, 2021.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[35] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[36] 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 *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[37] A. Singh and S. S. Borkotoky, "Federated learning over lora networks: Simulator design and performance evaluation," *arXiv preprint arXiv:2508.10574*, 2025.

[38] Semtech, *SX161/2 Data Sheet*, 2nd ed., Semtech, 2024, available at https://www.semtech.com/products/wireless-rf/lora-connect/sx1262.

[39] J. de Carvalho Silva, J. J. Rodrigues, A. M. Alberti, P. Solic, and A. L. Aquino, "Lorawan—a low power wan protocol for internet of things: A review and opportunities," in *2017 2nd International multidisciplinary conference on computer and energy science (SpliTech)*, 2017.

[40] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2021–2031.

[41] Y. Liu, S. Chang, and Y. Liu, "Fedcs: Communication-efficient federated learning with compressive sensing," in *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2023, pp. 17–24.