ELSA-FL: Energy and Load Serving Aware Federated Learning

Anna Lackinger, Pantelis A. Frangoudis, Andrea Morichetta, Schahram Dustdar

Distributed Systems Group, TU Wien, Vienna, Austria
{a.lackinger, p.frangoudis, a.morichetta, dustdar}@dsg.tuwien.ac.at

2026 International Conference on Future and Intelligent Networking (FINE) | 979-8-3195-0065-6/26/$31.00 ©2026 IEEE | DOI: 10.1109/FINE71157.2026.00026

***Abstract*—Federated Learning (FL) enables collaborative model training across distributed devices. A major concern in FL is how to operate it smoothly in resource-constrained environments, where this process must perform under strict operational constraints, such as cost or energy budgets. An often-overlooked aspect is the effect of introducing a new FL process into a running environment, where the additional workload competes for limited, shared resources, thereby disrupting already-running workloads that rely on stable, low-latency performance. This challenge is most pronounced in continuous FL, where inference requests and training overlap, potentially leading to offloading of model serving. Offloading can add costs, raise privacy concerns, and increase latency. Consequently, mitigating the number of inference requests offloaded is beneficial and should be considered when selecting devices for training.**

We propose ELSA-FL, a novel method that autonomously learns an effective client-selection policy under a strict budget constraint. Our approach aims not only to improve FL performance within the given energy budget but also to reduce the latency of parallel inference requests by preferentially selecting clients with lower background load. The objective is to enable a more efficient and practical deployment of FL in real-world, multi-task environments. Our results demonstrate that the proposed method consistently offloads significantly fewer requests than other RL and heuristic methods, while minimally affecting FL performance. Therefore, our method ensures good performance of the FL process, while interrupting ongoing processes as little as possible.

***Index Terms*—Federated Learning, Reinforcement Learning, Edge Computing, Sustainable ML, Green ICT.**

I. Introduction

The rapid proliferation of mobile edge devices has resulted in the massive growth of data generated at the edge [1]. To identify patterns in this ever-growing volume of data, there has been a shift from static, rule-based systems to adaptive, learning-oriented Artificial Intelligence (AI) models [2]. The traditional approach to training a Machine Learning (ML) or Deep Learning (DL) model is to use data collected on a single machine. However, this traditional approach has several drawbacks, such as data often being distributed across various devices, and collecting all data at a central server raises both logistical and privacy concerns [3].

Federated learning (FL) has emerged as a solution to overcome these challenges and has gained significant popularity [4]. Instead of sending their local data to a different server, clients in FL train an ML model on their data and only upload the trained weights to a central server for aggregation, thereby ensuring data locality and privacy. After receiving the new models, the server aggregates them and redistributes a new global model, starting a new training round. This iterative process continues until some convergence criterion is met. While this approach offers several benefits, such as privacy and distributed, parallel training, it also introduces new challenges compared to traditional deep learning. These novel obstacles include the high number of participating nodes, unbalanced data that are often not independent and identically distributed (non-IID), and performance that depends on the communication and computation capabilities of participating entities, which are often heterogeneous, resource-constrained, and sometimes even unreliable [5].

To overcome the challenges introduced by heterogeneous FL devices, *client selection* has become an emerging topic. Client selection has indeed been shown to improve model accuracy, enhance fairness, strengthen robustness, and reduce training overheads [6]. Due to various factors to consider in the selection process, such as data distribution, communication overhead, computational capabilities, dataset size, and other system- and data-level considerations, client selection is a difficult, multi-dimensional problem. While several heuristic approaches have been proposed in recent years, they cannot easily adapt to dynamic changes, and a fixed algorithm may lead to a biased selection and overrepresentation of a few clients [7].

To overcome the limitations of heuristic approaches, *Reinforcement Learning* offers a solution that autonomously learns a selection policy and can self-adapt to dynamic changes. While RL has shown to be a promising solution, current literature often does not consider practical aspects when introducing a new FL process in an environment [8]–[14]. One of these aspects is budget constraints, which have mostly not been considered in related work so far [7]. Considering such constraints is important, since the resource consumption of an FL process cannot be assumed unbounded; instead, this application may has to operate within explicit limits imposed by cost or energy restrictions. Another important aspect is that several applications, such as video analytics for autonomous driving or security monitoring, require continuous FL training to adapt the model to data drift [15]. This leads to training and inference requests overlapping, requiring end devices to perform both simultaneously, creating severe resource pressure [15]. To address this issue, client devices can offload inference tasks to ensure enough capacity for

979-8-3195-0065-6/26/$31.00 ©2026 IEEE
DOI 10.1109/FINE71157.2026.00026

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.

FL training. Offloading tasks adds additional overhead to the network, increasing latency, and may lead to additional costs for running workloads on an additional server [16]. It can also introduce vulnerabilities during task transmission and raise privacy concerns for sensitive data [17]. Therefore, it is also essential to consider whether and how many latency-sensitive tasks the devices are serving when selecting clients for training.

Accounting for clients' additional inference (or other) workloads introduces a new dimension to estimating the energy cost of a single client's participation. In particular, this necessitates incorporating an additional metric into cost estimation and, subsequently, into the selection process to account for offloading-related costs. At the same time, this introduces new objectives associated with offloading optimization that may compete with other goals, such as maximizing FL accuracy.

In this work, we propose an energy-efficient solution that considers a new objective, which aims to minimize the offloading of latency-critical tasks that client devices have to serve. Our novel method is called ELSA-FL (abbreviated as ELSA) because it is both Energy and Load Serving Aware, and it contributes to answering the following: *How can an energy-constrained client selection method be designed that maximizes accuracy while minimizing task offloading?*
In summary, our contributions are the following:

- We present ELSA (§ III-A – § III-C), an *Energy* and *Load Serving Aware* FL client selection approach, which considers both statistical and system heterogeneity of the clients. Compared to existing schemes (§ II), ELSA incorporates the additional workload clients may have to serve into the selection strategy.
- We address scalability by modeling client selection as a cooperative Multi-Agent RL (MARL) problem, where each client device is represented by a small RL model (§ III-E). Since ELSA operates under a fixed energy budget, the number of selected clients depends on the given budget and the energy consumption of the devices. To estimate the latter, ELSA agents use an energy consumption model that not only accounts for the communication and computation energy for FL model training but also includes the additional energy required for task offloading (§ III-D).
- ELSA is designed to optimize multiple objectives simultaneously: maximizing accuracy in FL and minimizing inference offloading. We propose a mixed reward design (§ III-E) with a team reward and a local penalty, which helps ELSA to filter clients that must serve a higher number of inference requests more effectively.
- We experimentally evaluate ELSA (§ IV) in a simultaneous FL training-inference serving setup for an image classification task on the Fashion-MNIST dataset, with heterogeneous devices and non-IID data. We assess different reward designs and show that ELSA outperforms heuristic and RL-based client selection approaches.

II. Related Work

Client selection is an important, multi-dimensional problem in FL, as it can speed up the training process and improve performance. Aspects to consider for client selection include system heterogeneity and statistical heterogeneity [6].

Statistical heterogeneity includes challenges such as unbalanced dataset sizes and non-IID data. Non-IID data show differences in data distribution and characteristics between clients, which can significantly affect FL performance, as the imbalanced class distribution leads to uneven local model updates, prolonging the process to reach satisfactory accuracy [20], [21]. Zhang et al. [22] propose a method that utilizes the weight divergence to recognize the non-IID degrees of clients and chooses clients with a lower degree of non-IID data more frequently. Cho et al. [23] propose a method that biases client selection towards clients with higher local loss to achieve faster error convergence.

System heterogeneity consists of both computation and communication capabilities that can vary between clients. Other aspects that can additionally affect the performance of clients include battery and applications running in the background. Asad et al. [24] propose a hierarchical setting, which achieves both cost savings and converges within fewer communication rounds. Danilenka et al. [25] adjust batch size and learning rate of heterogeneous devices using active inference to ensure that training rounds remain within a given objective. The authors in [26] select clients randomly, but they reduce uplink communication costs using structured and sketched updates, where the first method learns a restricted space using fewer variables, and the second compresses the model update.

RL for FL client selection Reinforcement learning has been shown to be an effective solution by learning a strategy autonomously, so that one does not have to define algorithmic rules explicitly [8]–[14]. Additionally, RL can adapt over time to changing environmental dynamics. Table I compares our work to other approaches, where we focus on the comparison with RL-based methods. The comparison includes the proposed method, whether non-IID data distributions and device heterogeneity are considered. Moreover, we evaluate whether that work also considers heterogeneous resource constraints in the form of a budget that accounts for energy- or cost-related metrics in the selection process. Approaches that select a fixed number of clients without energy or cost constraints are not listed. Additionally, we evaluate whether the proposed approaches account for clients performing other tasks besides FL and incorporate this into the selection process.

From the table, it is visible that there is no clear trend towards favoring any specific approach. Almost all methods consider statistical and device heterogeneity; only a few methods consider budget constraints [1], [7], [19], and none of these works accounts for additional workloads clients may have to serve.

Inference serving while training In the literature, only a few papers consider and emphasize the importance of other load FL clients may need to perform, but current approaches

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.

TABLE I: FL non-IID client selection solutions in the literature.

Ref.	Year	Method	non-IID Data	Device Diversity	Budget	Additional tasks
[18]	2024	DDPG	✓	✓	–	–
[8]	2025	PPO	✓	–	–	–
[9]	2024	REINFORCE	✓	✓	–	–
[10]	2025	DDQN	✓	✓	–	–
[11]	2024	SAC	✓	✓	–	–
[12]	2026	DDQN	✓	∼	∼	–
[13]	2024	DDQN	✓	–	–	–
[14]	2023	iPPO	✓	✓	–	–
[19]	2023	DQN and DDPG	✓	–	✓	–
[7]	2025	iPPO	✓	✓	✓	–
[1]	2022	REINFORCE	✓	✓	✓	–
ELSA	2026	iPPO	✓	✓	✓	✓

do not account for budget constraints, and are not based on reinforcement learning, limiting their ability to adapt automatically [15], [27], [28].

III. System Design

A. Operating environment

Fig. 1 provides an overview of the environment with the Federated Learning process, the service requests sent by external users, and our novel coordination method. At the

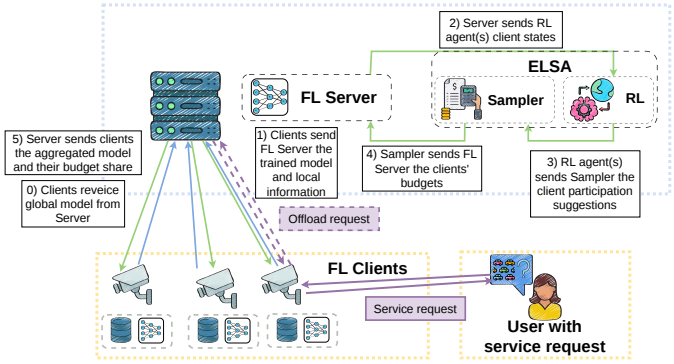


Fig. 1: Overview of our system design.

bottom of this figure, at the Internet of Things (IoT) layer, one can see users sending service requests, which, in our case, are latency-sensitive tasks that the client devices either compute themselves or offload to the server when they lack local capacity. In our resource-constrained environment, we assume that clients participating in FL training do not have the capacity to serve additional tasks. Since these tasks are latency sensitive, they cannot wait until the clients finish their training and require offloading. Since the server has a copy of the global model, the inference requests can be offloaded there. The main application visualized in Fig. 1 is the FL process, which is divided into multiple steps. The overall procedure consists of one FL round, followed by a sampling round, after which the next FL round is started.

B. FL process

To start the FL process, the server sends the global model to the clients (Step 0). Once all participating clients receive the most recent global model parameters $\mathbf{w}^t$, they update their local model parameters $\mathbf{w}_n^t$ with the global model $\mathbf{w}^t$ using gradient descent to minimize their local loss function $L_n(\mathbf{w}_n^t)$. The local loss function is defined as the average of the individual batch losses $l_j(\mathbf{w}_n^t)$ [1]:

$$L_n(\mathbf{w}_n^t) = \frac{1}{|D_n|} \sum_{j \in D_n} l_j(\mathbf{w}_n^t) \tag{1}$$

When a new FL process is introduced (round $t = 0$), all clients train for one epoch, and send the server the trained local model parameters $\mathbf{w}_n^{t+1}$ together with additional information, including an estimation of their energy consumption (Step 1). Once the server receives all trained models and state information, the client selection process starts, which is repeated in each FL round t. The choice of clients is determined by **ELSA**, as detailed in Steps 2-4 and Section III-C. While ELSA evaluates the clients' budgets, the server can aggregate the models received by the clients using the Federated Averaging (FedAvg) algorithm:

$$\mathbf{w}^{t+1} = \frac{\sum_{n=1}^{N} |D_n| \mathbf{w}_n^{t+1}}{\sum_{n=1}^{N} |D_n|} \tag{2}$$

where $|D_n|$ is the number of data samples used by client n for local training. Once ELSA has calculated the energy budget for each client for the next round, the updated global model parameters $\mathbf{w}^{t+1}$ together with the new budget are sent back to all clients for the next training round (Step 5). If the cost of a client training is within the budget received by the server, the client performs local training for several epochs. This completes the first FL round. In the following rounds, clients who received sufficient budget participate in the training and send their results back to the server (Step 1).

C. ELSA client selection procedure

The design of ELSA is based on our previous work [7] and extended with the new dimensionality of inference requests, which not only adds a new component to state and energy consumption but also increases the complexity of the *reward* in our RL-based client selection process. The selection process of ELSA starts with the server sending the *RL agents* the clients' state information (Step 2). Details of the individual parameters in the state can be found in Section III-E. ELSA is based

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.

on Multi-Agent RL (MARL), since the complexity of the model as well as the energy consumption for training are much smaller compared to Single-Agent RL (SARL) [7]. Moreover, this design choice also improves robustness in environments where clients could join or leave [7]. In our case, the most important factor is the energy consumption, where MARL ensures a fixed, small model size [7]. The RL models are non-pretrained, which lowers energy consumption since no training is required before the process starts, and it also ensures that the models are not dependent on a specific training environment and can be used in any setting. The training of the RL models starts with the beginning of the FL process, and our tests show that our small models can indeed learn a suitable selection strategy in a short time. Each agent validates possible actions in the given state using its own model and sends them to the *sampler* (Step 3). The action values produced by the RL agents are called *participation suggestions*, as they strongly influence the participation decision, though the decision is not based solely on them. These participation suggestions have values between 0 and 1, indicating the client's certainty about participating. Based on these suggestions, the clients' energy consumption, and the given budget, the sampler then decides on the final action and sends its decision to the server (Step 4).

ELSA builds on Proximal Policy Optimization (PPO), which does not have a replay buffer. Therefore, the data collected during its training do not need to be stored for long; once they have been used to update the policy, they are discarded. This also makes PPO well-suited to our scenario, where we want to train the RL method on the go without relying on large amounts of stored experience. In addition, PPO is very effective in cooperative multi-agent settings [29]. To increase stability of the sampler, we have decided to use an ϵ-*greedy* sampling mechanism, which has been shown to improve stability and convergence [7]. This sampling mechanism is usually not used for PPO, which typically uses a *stochastic* sampling mechanism. The stochastic sampler uses the RL network's output to randomly select an action, with higher-valued actions more likely to be chosen. The ϵ-greedy sampler chooses the action with the highest value with probability $1-\epsilon$ (exploitation); otherwise, it selects an action uniformly at random (exploration). The $epsilon$ parameter decreases in each round, gradually reducing exploration.

D. Energy Consumption Model

We model the energy cost as a function of communication and computation, which is consistent with previous work [7], [18], [30]–[33]. The energy is calculated only for clients participating in one FL round; it is therefore the variable energy consumption dependent on the selection process and does not consider other aspects of the FL process, such as model aggregation. In addition to the energy needed to train clients, we also consider the energy used for task offloading.

The **communication energy** [32] is measured at each FL round as the energy required for letting the clients that participated in the training round send model updates. To ensure that this work compares to related work, we use the formula from Mao et al. to calculate the communication energy [9]:

$$E_n^{t,\text{comm}} = P_n \frac{M_n}{B \log_2\left(1 + \frac{h_n P_n}{\sigma}\right)} \tag{3}$$

where B is the bandwidth, transmitter power P_n, background noise power σ, the channel gain h_n, and model size M_n.

When a client participates in the FL training, it not only adds the communication energy for its model, but also the communication energy for offloading its tasks, which is defined as:

$$E_n^{t,\text{tasks}} = P_n \frac{\tilde{\ell}_n^{off}}{B \log_2\left(1 + \frac{h_n P_n}{\sigma}\right)} \tag{4}$$

where $\tilde{\ell}_n^{off}$ is the size of the task-related data in bits that node n offloads during a round.

The second part of the energy consumption formula is the **computation energy**. For the computation energy, we use a formula that has been often used in the literature [7], [31]–[33]:

$$E_n^{t,\text{comp}} = \frac{\alpha_n}{2} \eta_n \left|\tilde{D}_n\right| f_n^2 \tag{5}$$

where $|\tilde{D}_n|$ is the data size of client n in bits that it trains on in round t, f_n the CPU-cycle frequency, the number of CPU cycles to execute one data bit η_n, and the effective capacitance of client n's computing chipset $\frac{\alpha_n}{2}$.

Considering both the communication and computation energy terms, the energy consumed by one participating client can be formulated as:

$$E_n^{t,\text{total}} = E_n^{t,\text{comm}} + E_n^{t,\text{tasks}} + epochs \times E_n^{t,\text{comp}} \tag{6}$$

Therefore, the energy for one round consumed by all participating clients is expressed as:

$$E^{t,\text{total}} = \sum_{n \in \mathcal{P} \subseteq \mathcal{N}} \left(E_n^{t,\text{comm}} + E_n^{t,\text{tasks}} + \text{epochs} \times E_n^{t,\text{comp}}\right) \tag{7}$$

where $\mathcal{P}$ is the set of participating clients, i.e., a subset of all clients $\mathcal{N}$. The energy in each round $E^{t,\text{total}}$ has to stay within the given budget constraint.

In our proposed framework, the FL clients utilize this function to estimate their own energy consumption. These estimates are then sent to the server as input for ELSA to know how many clients can be selected to stay within the given budget. In a real-world deployment, these estimates could easily be replaced by energy cost measurements.

E. Multi-Agent Reinforcement Learning

To make our approach scalable and keep the RL network's model size minimal, we have formulated our system design as a cooperative Multi-Agent RL (MARL) problem. The objective in such a setting is to train I agents so that they perform actions that maximize a shared *team* reward [34]. At every time step t, each agent i observes the local state s_n^t of a client n and chooses an action a_n^t based on that state. After all agents perform their actions, they receive a joint reward r^t, and the environment transitions to the next state s_n^{t+1}. In

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.

this work, we have two objectives, which are to maximize accuracy and minimize the tasks that have to be offloaded. To accelerate convergence, we have decided not only to use a shared team reward to maximize performance, but also to add a local penalty for clients who must offload local tasks. This design choice of a mixed reward signal with both a global and local objective can indeed improve performance [35].

State: The state s_n^t represents the state of client n at time t and it is defined as $s_n^t = \{\overline{acc}^t, |\ell_n^t|, |D_n|, acc_n^t, e_n, p_n^t\}$. s_n^t contains context information, which in our case is the average accuracy of all clients $\overline{acc}^t$ achieved after communication round t. Furthermore, the state also consists of the normalized task size $|\ell_n^t|$ at round t, the normalized dataset size of the client $|D_n|$, the accuracy on its local test dataset acc_n^t, the energy consumption for one training round e^n, which includes communication, computation and task offloading energy, and whether client n participated in the last round p_n^t.

Action: In our FL setting, we consider $\mathcal{N}$ clients. At each time step t, the RL agent(s) suggest for each client n whether to participate in the FL training, denoted by a probability value a_n^t. The participation suggestion is forwarded to the sampler, which will decide on the final action $\bar{a}_n^t$. The final action is structured as a list including the energy share for each client in round t.

Reward: The reward function evaluates the effectiveness of the action a^t taken in state s^t. The reward consists of a global objective and a local penalty. We define the reward for agent n as: $r_n(s^t, a^t) = \varphi\lambda^{|\overline{acc}^t - \overline{acc}^{t-1}|} - \psi\lambda^{(\frac{\ell_n^{t,off}}{\sum_{m\in\mathcal{N}}\ell_m^t})}$, where λ is a constant that makes the reward grow with the global test accuracy, $\sum_{m\in\mathcal{N}}\ell_m^t$ represents the total number of tasks received by all nodes during the FL training and $\ell_n^{t,off}$ is the number of tasks agent n offloads at round t. Since the accuracy improvement of the global model tends to slow down as it approaches convergence, the updates become smaller. To ensure that the RL agent can still capture these small improvements, we set λ to 64 in our experiments, which is consistent with related approaches [9], [36]. When the last global accuracy is smaller than the current global accuracy, φ is 1; if the reward has not changed, it is 0; otherwise, it is -1 to decrease the participation probability of the clients that took part in that round. Since accuracy maximization is not our only objective, we additionally include the task percentage. To ensure that the offloaded task percentage is equally important as the reward, we also use lambda. While the accuracy is a shared objective across all agents, the offloading penalty is only used for individual clients, taking into account their offloading tasks. To ensure that clients who did not offload any tasks receive no penalty, ψ is set to 0 when $\ell_n^{t,off} = 0$. In future work, these objectives could be weighted differently to reflect their importance.

The reward does not include the energy cost of the clients, since our goal is not to minimize energy consumption of the FL clients, but to optimize performance in energy-constrained environments.

IV. Evaluation

The baselines for the evaluation include both heuristic and RL methods. The main goal of the analysis is to evaluate the efficiency of the learned policy in selecting the best clients within the budget to improve both model accuracy and reduce task offloading. In the evaluation, we compare and evaluate our proposed mixed reward design with a global reward and a reward without penalty.

A. Experimental Setup

Datasets: For our tests, we chose a classification dataset commonly used in FL prediction tasks, namely Fashion-MNIST. In our non-IID setting, we distinguish among four dataset types that clients receive. One fourth of the clients have a balanced label distribution, one fourth have samples of three labels, another fourth have samples of two labels, and the rest of the clients have samples from only one label.

Client energy consumption and tasks: We differentiate between cheaper and more expensive clients. For each of the four data distribution categories, there are 2 cheap and 3 expensive clients. Moreover, in each category are 3 clients that do not have to serve inference requests and 2 clients with 300 requests per round. These requests are images being sent to the client devices. We include them in the computation of the system state, energy consumption, and reward.

Hyperparameters: The system configuration parameters related to computation and communication energy consumption are taken from [32]. All hyperparameters can be found in [7].

Budget: The budget is minimal to show the performance, when the right client selection is extremely important. Within the budget, three cheaper clients can be chosen, one cheap and one expensive client, or one client with high energy consumption.

Hardware: The tests conducted in this work ran on a server that is equipped with 32 CPUs, specifically the AMD Ryzen 9 5950X 16-Core Processor, each with a maximum clock speed of 5083.3979 MHz. Additionally, the server incorporates two NVIDIA GeForce RTX 3090 GPUs, each featuring 24576 MB of memory.

B. Baselines

Our method is compared to both RL and heuristic baselines.

FedAvg (Heuristic): This approach uses a sampling mechanism that randomly selects clients until the budget is reached [4], [8].

PoC (Heuristic): The *Power-of-Choice* method is a heuristic approach, which selects the clients with the highest training loss [23]. To stay within the budget, the clients are first sorted based on their local loss and then selected until the budget is reached.

FPPO (RL): FPPO is a SARL approach based on PPO. It first gathers participation suggestions for each client via a single network, and then selects the ones with the highest values until it reaches the budget threshold. To train the policy, it

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.

computes the reward and stores samples of the participating clients to train with. This baseline is inspired by the PPO implementation of the recently published work by Zhao et al. called FedPPO [8] and adapted to our use case. FPPO is therefore a SARL approach that compares to ELSA, as it has the same sampling mechanism.

PPO (RL) We implement PPO, pairing it with a traditional, stochastic-policy sampler, which has a weighted random selection approach, within budget boundaries.

IPPO (RL) This baseline has the same sampler as PPO, but instead of the RL method having a global view, each client has its own network.

C. Mixed Reward Design

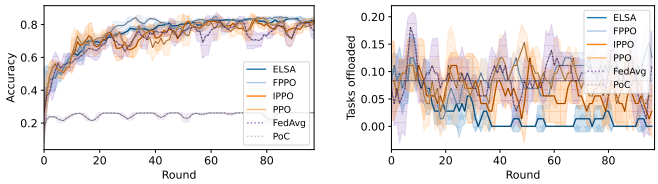
The goal of these tests is to evaluate ELSA's performance when some clients must perform tasks, with the objective of both maximizing accuracy and minimizing task offloading. For these tests, we use a mixed reward design in which clients receive a global accuracy reward, and some may receive a local penalty depending on whether they have to offload tasks. Table II presents the results, including both averages and

TABLE II: Accuracy and offloading results in %.

First half	ELSA	FPPO	IPPO	PPO	FedAvg	PoC
Accuracy mean	67.13	**69.08**	64.91	65.43	63.87	24.22
Accuracy std	**15.97**	17.58	16.62	16.69	16.64	3.39
Offloaded mean	**4.34**	7.99	7.31	8.16	8.42	8.33
Offloaded std	**6.19**	**6.15**	7.39	7.48	7.70	0.00
Second half	ELSA	FPPO	IPPO	PPO	FedAvg	PoC
Accuracy mean	**82.00**	**82.21**	79.11	79.23	76.54	25.60
Accuracy std	**3.04**	4.50	5.99	7.24	7.85	1.46
Offloaded mean	**0.58**	9.67	5.42	8.42	9.42	8.33
Offloaded std	**2.56**	5.39	6.58	7.16	7.09	0.00

standard deviations of accuracy and the percentage of tasks offloaded. In bold, we highlight the highest mean accuracy, the lowest mean number of offloaded requests, and the lowest standard deviations, as they indicate the most stable results. The results are split into two parts. In the first half, the RL method may have to explore more compared to the second half. During the first 50 rounds, the performance of FPPO has a higher average accuracy, followed by ELSA. PPO, IPPO, and FedAvg have lower accuracy, and PoC has the lowest accuracy. The standard deviation is also comparable in the first half; ELSA has a slightly lower one, FPPO a slightly higher one, and PoC has the lowest. When it comes to task offloading, ELSA already has a better strategy than the other methods, offloading only about half of the tasks on average. In the second half, ELSA and FPPO achieve gains in accuracy of almost 3 percentage points over the RL methods and more than 5 percentage points over FedAvg. Compared to PoC, ELSA has much higher accuracy, whereas PoC performs poorly in our tests. A possible reason is that it mostly chooses clients with highly unbalanced datasets, since they often have the highest loss. This, however, leads to the global model being biased toward a few samples, decreasing the overall accuracy. While FPPO has slightly higher mean accuracy, ELSA has a lower standard deviation, indicating more stable performance. When it comes to offloading, ELSA has significantly better performance than the other methods. IPPO has also slightly better performance, showing that the local penalty works well for MARL methods. The SARL approaches and heuristic approaches do not perform as well when it comes to minimizing task offloading.

Overall, these tests demonstrate that the RL methods can indeed improve the objective of maximizing accuracy, and ELSA additionally significantly minimizes task offloading.



(a) Accuracy with FashionMnist. (b) Proportion of tasks offloaded.

Fig. 2: Moving average and standard deviation with a window of 3.

Fig. 2 additionally visualizes the results of the accuracy using a moving average over 3 data points. This helps to illustrate trends of the various methods. Fig. 2a shows that PoC visibly performs worse than the other methods, and Fig. 2b highlights that ELSA offloads much fewer tasks than the other methods.

D. Team Reward Design

In this section, we test how the RL methods perform when they receive only a shared team reward that includes both the global accuracy and the overall number of tasks offloaded. Therefore, the overall reward can be described as $r_n(s^t, a^t) = \varphi\lambda^{|\overline{acc}^t - \overline{acc}^{t-1}|} - \psi\lambda^{\left(\frac{\sum_{p\in\mathcal{P}}(\ell_p^{t,off})}{\sum_{m\in\mathcal{N}}\ell_m^t}\right)}$, where the penalty now includes the overall proportion of tasks offloaded if the number of offloaded tasks is greater than 0.

TABLE III: Relative difference compared to tests with tasks in reward.

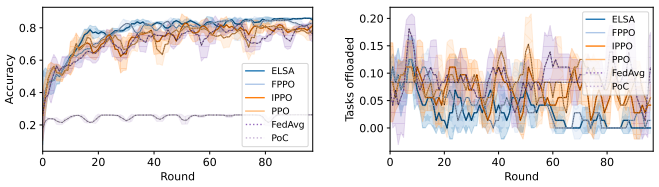
First half	ELSA	FPPO	IPPO	PPO	FedAvg	PoC
Accuracy mean	**3.55**	-0.35	0.32	-0.14	0.00	0.00
Accuracy std	1.43	-0.14	0.09	**-0.49**	0.00	0.00
Offloaded mean	0.68	**-1.87**	0.43	0.09	0.00	0.00
Offloaded std	0.21	0.57	**-0.08**	0.25	0.00	0.00
Second half	ELSA	FPPO	IPPO	PPO	FedAvg	PoC
Accuracy mean	**2.00**	0.36	0.19	-1.45	0.00	0.00
Accuracy std	0.91	**-0.60**	0.09	0.31	0.00	0.00
Offloaded mean	1.00	**-7.92**	0.25	-0.17	0.00	0.00
Offloaded std	1.45	**-1.32**	-0.14	0.48	0.00	0.00

Table III shows the relative results in comparison to the results when the task penalty is designed as a local penalty. A value of 0 indicates no difference; a value above 0 can be interpreted as a higher percentage when we do not consider task offloading. To begin with, PoC and FedAvg show no

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.

difference, which is not surprising, since their selection strategies are unaffected by the reward design. While the overall accuracy of ELSA and IPPO increases, so does the average number of tasks offloaded. Both have an increase in standard deviation, indicating a slightly less stable performance. Therefore, the team reward makes the penalty less effective for MARL methods, compared to the local one. The team reward slightly improves FPPO's accuracy in the second half and decreases PPO's accuracy, but both SARL methods show a decrease in offloaded tasks, indicating that, for SARL methods with a global view of all clients, a team reward is more effective for mitigating task offloading.

Overall, the team reward increases the mean accuracy and the number of offloaded tasks for MARL, showing that a local reward design is more effective for mitigating offloading. For SARL, it decreases task offloading, indicating that a global reward design is more beneficial for these methods.



(a) Accuracy with global reward. (b) Proportion of tasks offloaded.

Fig. 3: Moving average and standard deviation with a window of 3.

Fig. 3 shows the moving average and standard deviation of the various methods with the global reward design. In Fig. 3a, one can see that ELSA indeed performs better than the other methods on average. Figure 3b additionally illustrates the number of tasks offloaded, where ELSA outperforms the other methods. This figure also shows that FPPO has improved its selection strategy towards task offloading significantly.

E. Reward Without Penalty

These tests aim to evaluate the performance when only the accuracy is considered in the reward function, so that $r_n(s^t, a^t) = \varphi\lambda^{|\overline{acc}^t - \overline{acc}^{t-1}|}$. Based on these tests, we can evaluate the trade-off one may have to pay in terms of FL accuracy when additionally aiming to minimize task offloading. Furthermore, these tests will show whether adding the task penalty indeed reduces the number of tasks offloaded.

Table IV shows the results of subtracting the results with the local penalty from the results when only the accuracy is considered. Over the last 50 rounds, ELSA can improve accuracy by more than 2.5 percentage points on average, but it has to offload, on average, 9 percentage points more tasks. While IPPO improves the accuracy only slightly, with less than 1 percentage point on average, the tasks offloaded increase by more than 3 percentage points. FPPO interestingly shows worse performance in terms of accuracy compared to the local reward design, indicating that it either needs more time to learn a stable strategy or that the local reward design does not work well in SARL. PPO shows a slight improvement in accuracy, and interestingly, it offloads, on average, fewer tasks than in the previous tests. FedAvg and PoC both have no changes, since they are not affected by the reward definition.

TABLE IV: Relative difference compared to tests with tasks in reward

First half	ELSA	FPPO	IPPO	PPO	FedAvg	PoC
Accuracy mean	0.92	-0.84	-0.08	**1.10**	0.00	0.00
Accuracy std	1.27	0.53	0.32	**0.28**	0.00	0.00
Offloaded mean	4.76	1.53	1.36	**0.26**	0.00	0.00
Offloaded std	**-0.93**	-0.28	0.06	-0.10	0.00	0.00
Second half	ELSA	FPPO	IPPO	PPO	FedAvg	PoC
Accuracy mean	**2.66**	-2.96	0.72	0.99	0.00	0.00
Accuracy std	0.84	2.69	0.30	**-0.26**	0.00	0.00
Offloaded mean	9.00	-0.67	3.42	**-1.42**	0.00	0.00
Offloaded std	**-0.11**	1.12	0.96	**-0.15**	0.00	0.00

Overall, ELSA may lose almost 3 percentage points of FL performance when accounting for inference requests, while saving, on average, 9 percentage points of offloaded tasks, thereby significantly reducing the need for offloading.

V. Discussion

While our work shows the performance of ELSA with different reward designs using Fashion-MNIST, it would be valuable to examine the method's performance on other learning tasks. In future work, we also plan to evaluate our approach in a dynamic setting where the number of tasks executed by clients varies over time. Furthermore, we intend to perform experiments on real heterogeneous hardware to obtain accurate energy measurements.

VI. Conclusion

This paper introduces ELSA, an Energy and Load Serving Aware RL–based client selection method for federated learning. ELSA selects clients for each training round according to a predefined energy budget, making it well-suited for real-world deployment scenarios where resources are limited. In contrast to existing approaches, ELSA is the first RL-based client selection method that accounts for additional workloads clients may have to serve, which must be offloaded when clients are occupied with training. The objective of our method is not only to maximize accuracy, but more importantly, to stay within given budget constraints and minimize offloading as much as possible. To mitigate task offloading, we design a mixed reward signal that integrates both global and local information. The global component rewards overall model accuracy, while the local component is a penalty for individual clients for the task offloading.

Our evaluation, which includes both heuristic and RL–based methods, shows that ELSA reduces the number of offloaded tasks by up to 9 percentage points, while improving the model's accuracy. The local reward design reduces the tasks offloaded with ELSA to less than 1% on average. When task offloading is not considered, accuracy may improve by roughly 2.5 percentage points, but task offloading increases by almost 9

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.

percentage points, adding network overhead, increasing energy consumption, and potentially incurring additional costs. Overall, our results demonstrate that ELSA significantly reduces offloading while minimally affecting accuracy, all within an energy budget.

ACKNOWLEDGMENTS

This work is funded by the HORIZON Research and Innovation Action 101135576 INTEND "Intent-based data operation in the computing continuum."

REFERENCES

[1] Y. Deng, F. Lyu, J. Ren, H. Wu, Y. Zhou, Y. Zhang, and X. Shen, "Auction: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1996–2009, 2022.

[2] M. Abel, I. Ahmad, C. A. Casado, R. Berner, M. Bettinelli, K. M. Björk, M. Capobianco, J. Gross, T. H. Nguyen, P. Hui *et al.*, "Large language models in the 6g-enabled computing continuum: a white paper," Ph.D. dissertation, University of Oulu, 2025.

[3] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, 2022.

[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*. Pmlr, 2017, pp. 1273–1282.

[5] J. Wang, Y. Hong, J. Wang, J. Xu, Y. Tang, Q.-L. Han, and J. Kurths, "Cooperative and competitive multi-agent systems: From optimization to games," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 5, pp. 763–783, 2022.

[6] L. Fu, H. Zhang, G. Gao, M. Zhang, and X. Liu, "Client selection in federated learning: Principles, challenges, and opportunities," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 21 811–21 819, 2023.

[7] A. Lackinger, A. Morichetta, P. A. Frangoudis, and S. Dustdar, "Bippo: Budget-aware independent ppo for energy-efficient federated learning services," *arXiv preprint arXiv:2511.08142*, 2025.

[8] Z. Zhao, A. Li, R. Li, L. Yang, and X. Xu, "Fedppo: Reinforcement learning-based client selection for federated learning with heterogeneous data," *IEEE Transactions on Cognitive Communications and Networking*, 2025.

[9] W. Mao, X. Lu, Y. Jiang, and H. Zheng, "Joint client selection and bandwidth allocation of wireless federated learning by deep reinforcement learning," *IEEE Transactions on Services Computing*, vol. 17, no. 01, pp. 336–348, jan 2024.

[10] S. Ghazi, S. Farzi, and A. Nikoofard, "Federated learning for all: A reinforcement learning-based approach for ensuring fairness in client selection," *IEEE Access*, 2025.

[11] F. Zheng, Y. Sun, and B. Ni, "Fedaeb: Deep reinforcement learning based joint client selection and resource allocation strategy for heterogeneous federated learning," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 6, pp. 8835–8846, 2024.

[12] S. Khan, M. M. Karim, M. Muzammal, and Q. Qu, "Dynamic client selector: A double deep q-learning framework for efficient federated learning," in *2026 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2026, pp. 1–10.

[13] G.-m. Li, W.-x. Liu, Z.-z. Guo, and D.-x. Chen, "Client selection method for federated learning based on grouping reinforcement learning," in *2024 9th International Conference on Computer and Communication Systems (ICCCS)*, 2024, pp. 327–332.

[14] K. Yan, N. Shu, T. Wu, C. Liu, J. Huang, and J. Yu, "Joint client selection and training optimization for energy-efficient federated learning," in *2023 19th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2023, pp. 849–854.

[15] K. Luo, K. Zhao, T. Ouyang, X. Zhang, Z. Zhou, H. Wang, and X. Chen, "Efficient coordination of federated learning and inference offloading at the edge: A proactive optimization paradigm," *IEEE Transactions on Mobile Computing*, vol. 24, no. 1, pp. 407–421, 2024.

[16] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Cost-minimized computation offloading and user association in hybrid cloud and edge computing," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16 672–16 683, 2024.

[17] H. Mohapatra, "Task offloading and edge computing in iot—gaps, challenges and future directions," *Computers, Materials and Continua*, vol. 87, no. 3, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1546221826003310

[18] X. Chen, Z. Li, W. Ni, X. Wang, S. Zhang, Y. Sun, S. Xu, and Q. Pei, "Towards dynamic resource allocation and client scheduling in hierarchical federated learning: A two-phase deep reinforcement learning approach," *IEEE Transactions on Communications*, 2024.

[19] Z. Cheng, X. Fan, N. Chen, M. Liwang, L. Huang, and X. Wang, "Learning-based client selection for multiple federated learning services with constrained monetary budgets," *ICT Express*, vol. 9, no. 6, pp. 1059–1064, 2023.

[20] Z. Lu, H. Pan, Y. Dai, X. Si, and Y. Zhang, "Federated learning with non-iid data: A survey," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 188–19 209, 2024.

[21] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," 2021. [Online]. Available: https://arxiv.org/abs/2106.06843

[22] W. Zhang, X. Wang, P. Zhou, W. Wu, and X. Zhang, "Client selection for federated learning with non-iid data in mobile edge computing," *IEEE Access*, vol. 9, pp. 24 462–24 474, 2021.

[23] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[24] M. Asad, A. Moustafa, F. A. Rabhi, and M. Aslam, "Thf: 3-way hierarchical framework for efficient client selection and resource management in federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 085–11 097, 2021.

[25] A. Danilenka, A. Furutanpey, V. C. Pujol, B. Sedlak, A. Lackinger, M. Ganzha, M. Paprzycki, and S. Dustdar, "Adaptive active inference agents for heterogeneous and lifelong federated learning," *arXiv preprint arXiv:2410.09099*, 2024.

[26] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[27] P. Han, S. Wang, Y. Jiao, and J. Huang, "Federated learning while providing model as a service: Joint training and inference optimization," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024, pp. 631–640.

[28] A. Lackinger, P. A. Frangoudis, I. Čilić, A. Furutanpey, I. Murturi, I. P. Žarko, and S. Dustdar, "Inference load-aware orchestration for hierarchical federated learning," in *2024 IEEE 49th Conference on Local Computer Networks (LCN)*. IEEE, 2024, pp. 1–9.

[29] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in neural information processing systems*, vol. 35, pp. 24 611–24 624, 2022.

[30] T. Zhang and S. Mao, "Energy-efficient federated learning with intelligent reflecting surface," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 2, pp. 845–858, 2022.

[31] Y. Sai, X. Wu, J. Jiang, Y. Huang, Q. Yan, Z. Li, and H. Huang, "Optimizing hierarchical federated learning: A reinforcement learning approach," *IEEE Transactions on Consumer Electronics*, pp. 1–1, 2024.

[32] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE, 2020, pp. 1–6.

[33] N. H. Tran, W. Bao, A. Y. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. IEEE INFOCOM*, 2019.

[34] S. Q. Zhang, J. Lin, and Q. Zhang, "A multi-agent reinforcement learning approach for efficient client selection in federated learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 8, 2022, pp. 9091–9099.

[35] H. Mao, Z. Gong, and Z. Xiao, "Reward design in cooperative multi-agent reinforcement learning for packet routing," *arXiv preprint arXiv:2003.03433*, 2020.

[36] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE conference on computer communications*. IEEE, 2020, pp. 1698–1707.

Authorized licensed use limited to: TU Wien Bibliothek. Downloaded on November 09,2026 at 15:25:40 UTC from IEEE Xplore. Restrictions apply.