BIPPO: Budget-Aware Independent PPO for Energy-Efficient Federated Learning Services

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

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

***Abstract*—Federated Learning (FL) is a promising machine learning solution in large-scale IoT systems, guaranteeing load distribution and privacy. However, FL does not natively consider infrastructure efficiency, a critical concern for systems operating in resource-constrained environments. Several Reinforcement Learning (RL) based solutions offer improved client selection for FL; however, they do not consider infrastructure challenges, such as resource limitations and device churn. Furthermore, the training of RL methods is often not designed for practical application, as these approaches frequently do not consider generalizability and are not optimized for energy efficiency. To fill this gap, we propose BIPPO (Budget-aware Independent Proximal Policy Optimization), which is an energy-efficient multi-agent RL solution that improves performance. We evaluate BIPPO on two image classification tasks run in a highly budget-constrained setting, with FL clients training on non-IID data, a challenging context for vanilla FL. The improved sampler of BIPPO enables it to increase the mean accuracy compared to non-RL mechanisms, traditional PPO, and IPPO. In addition, BIPPO only consumes a negligible proportion of the budget, which stays consistent even if the number of clients increases. Overall, BIPPO delivers a performant, stable, scalable, and sustainable solution for client selection in IoT-FL.**

***Index Terms*—Federated Learning, PPO, Edge Computing, AIoT, Sustainable ML, Green ICT**

I. Introduction

Internet of Things (IoT) devices embed increasingly sophisticated sensing, computing, and connectivity capabilities pervade our daily lives and have led to a surge in data created at the edge. To identify hidden patterns in the vast amount of data, big data analytics, machine learning (ML), and deep learning (DL) algorithms are essential [1]. The application of AI for data processing in IoT is also known as *Artificial Internet of Things (AIoT)*, which is a new computing paradigm [2]. A leading solution for AIoT and for processing the vast amount of edge data is *Federated Learning (FL)* [3], due to its ability to distribute ML model training load while guaranteeing privacy and data locality. Despite its advantages, training shared models with FL comes with several challenges [4]. Without care, non-independent, identically distributed (non-IID) training data on client devices, characteristic of many IoT scenarios we target, can affect FL convergence behavior negatively [5], [6]. Furthermore, as resources are finite, it is crucial to properly orchestrate the ML services that use them [7]. In the context of FL in particular, a major orchestration dimension is client selection. Identifying a client selection strategy that autonomously balances the multidimensional problem of non-IID data and heterogeneous device resources is not trivial. Several heuristic approaches have been proposed in recent years, which are practical, but they cannot easily adapt to dynamic changes, and a fixed rule may lead to a biased selection and overrepresentation of a few clients [8].

In such complex scenarios, *Deep Reinforcement Learning (DRL)* offers a novel perspective through its ability to autonomously learn effective strategies [9]. However, existing methods that propose RL solutions for FL client selection are not optimized to perform well in practical AIoT applications, and many challenges that come with them remain unsolved. The main challenges we address in our work can be split into *AI challenges* that address both the model performance and generalizability, and *IoT challenges* addressing the infrastructure and resource management level.

The AI challenges we address in our work include both our FL use case and our RL method. The FL challenge we address in this work is client selection in environments with *non-IID data* and *heterogeneous devices*. Although various methods have been proposed for performance improvement with non-IID data, a single best solution still needs to be identified. This FL challenge can be solved with an RL solution that autonomously learns a good client selection policy. However, many proposed RL-based solutions are not *generalizable* to new environments as they are often trained to fit only one specific environment [10]–[13]. This limits their applicability in real-world scenarios, where information such as data distribution or device capabilities may not be available, making it impossible to reproduce the exact training conditions. Therefore, making our RL-based selection generalizable to new scenarios is an important challenge we address.

To make our method *sustainable*, efficient resource management is needed, which is a relevant IoT challenge. To support practical applications, a proposed method should operate efficiently in a budget-constrained environment, where available resources have a fixed limit. Such budget-constrained orchestration scenarios can arise when coordinating multiple FL and/or other ML workflows in an as-a-service manner [6], [14] over shared edge-compute infrastructure. In most existing RL work, such a budget constraint cannot easily be added, as it heavily influences exploration.

A low budget will lead to clients exploring non-participation more, which biases their action preference. An important resource management aspect that is often not considered, but highly relevant in energy optimization problems, is the energy consumed by the proposed RL method and how to minimize the energy consumption for training the RL method itself [10]–[13]. The importance of energy-efficient solutions is further emphasized by an increasing number of industrial companies that are investing in environmental protection to meet growing environmental awareness [15]. The sustainable use of resources is indeed crucial and has to be taken into account when finding solutions for efficient data processing. *Scalability* is needed since AIoT networks are expected to grow significantly, with an increase in the number of devices, users, applications, and data. It is therefore essential to find methods that can adjust to larger-scale environments to cope with this growth. *Stability* is another important IoT challenge that has not been considered so far in most existing RL-based FL client selection schemes, where clients join or leave during the FL training. Due to structural changes, network availability, and device updates, FL clients may join or leave over time, and it is essential for the client selection method to be able to adapt in these dynamic environments.

To address these limitations, we propose *Budget-aware Independent Proximal Policy Optimization (BIPPO)*, a novel approach for efficient client selection in FL. BIPPO is based on IPPO, but it has an improved sampler that not only manages the budget constraint but also improves the overall performance. BIPPO is built to work in resource-constrained environments as it coordinates a limited energy budget by intelligently selecting a subset of clients, considering multiple client parameters such as energy usage, data size, and local model performance. Due to our energy-optimized training, BIPPO's training cost remains independent of the number of clients and is orders of magnitude lower than SARL's as the number of clients increases. For BIPPO, the energy consumed for training is independent of the number of clients and stays consistent, making our approach more scalable. In summary, we make the following contributions:

- *New energy-aware client selection strategy (§ IV)*: We introduce a novel RL method that coordinates a shared energy budget between various heterogeneous IoT devices in the context of FL (§ III). Contrary to other client selection methods (§ II), *the number of selected clients is not assumed fixed* but depends on the given budget and the energy consumption of the participating clients' devices.
- *Performance considering AI challenges (§ V)*: Our extensive evaluation demonstrates that BIPPO, compared to traditional PPO, IPPO, and heuristic baselines, improves model accuracy and stabilizes performance in non-IID environments. It functions well in resource-constrained, low-budget environments, where identifying the right clients has a high impact on performance and *generalizes* well to new datasets.
- *Performance considering IoT challenges (§ V)*: BIPPO *scales* to more complex environments, while training it only uses up a negligible amount of the energy budget, which notably does not scale with additional clients, further highlighting the *sustainability* of our design. BIPPO also shows consistent performance when clients join or leave, showing its *stability* in dynamic environments. The implementation of our method and of our evaluation framework is available on GitHub [1].

II. Related Work

FL Solutions for Non-IID Data: Non-IID data shows 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. As a result, this increases communication overhead [19], since it takes the model longer to reach a satisfactory accuracy. In recent years, various solutions have been proposed to mitigate the negative effects of non-IID data, which can be grouped into data-based, algorithmic, and system-level approaches [20].

Data-based solutions can be split into data augmentation and data sharing techniques. Both of these approaches enhance learning performance, but most of them require data sharing, increasing the risk of data privacy leakage [20].

A *system-level* approach is to cluster the clients based on their data distribution. In the work of Chen et al. [21], the FL server estimates the statistical heterogeneity of the data of each client by using the client's update of the network's output layer. Based on this information, the clients are then clustered. This can also be extended by not only taking the data distribution into account, but also classifying the clients based on additional factors, such as sample size [22]. However, system-level approaches may need reconfiguration when the number of clients changes, as shown in [14]. Reconfiguration may be expensive and delay FL training.

The selection of FL clients, which falls in both *system-level* and *algorithmic* approaches, is an efficient way to deal with non-IID data, as it can speed up convergence and increase accuracy [23]. To solve the challenge of choosing the right subset, various client selection algorithms have been developed that show promising performance improvements [8]. However, determining the optimal client selection method for a specific Federated Learning process can be challenging due to the wide range of available strategies and influencing factors [24]. One algorithmic approach is to identify non-IID clients by looking at their weight divergence [25]. Clients with a lower degree of non-IID data will more frequently be chosen for the training, which can significantly increase accuracy. Another well-known method is the power-of-choice method, where clients with the highest loss are chosen to participate in the FL training [26]. However, these approaches do not focus on multiple dimensions, such as dataset size and device heterogeneity.

[1]https://github.com/Lacki28/BIPPO/

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

Ref.	Year	RL Method	Device Diversity/ non-IID Data	Generalizability of RL method	Resource Limits (Budget)	Scalability	Stability (Client-churn)
[1]	2022	REIN-FORCE	✓	∼	✓	✓	∼
[16]	2024	DDPG	✓	∼	–	∼	–
[4]	2022	VDN	✓	✓	–	✓	∼
[17]	2025	PPO	✓	✓	–	∼	–
[10]	2024	REIN-FORCE	✓	–	–	∼	–
[11]	2023	IPPO	✓	–	–	∼	∼
[5], [18]	2020,2024	DDQN	–	∼	–	–	–
[12]	2022	PPO	✓	–	–	–	–
[13]	2023	SAC	✓	–	–	–	–
BIPPO	2025	IPPO	✓	✓	✓	✓	✓

Several solutions have improved the overall performance in non-IID environments, but some methods only consider one dimension, such as data distribution, and do not include other factors, such as device heterogeneity and dataset size. Additionally, an algorithmic approach with a fixed selection strategy may not perform well in every environment, and it has no chance of learning or improving its strategy over time. Due to these limitations, we focus on RL methods, since they can autonomously learn and improve their strategy in various environments. With the appropriate reward configuration, which provides feedback for the RL agent to improve its policy, RL is also able to learn a strategy that considers multiple dimensions and autonomously learns a solution that finds a trade-off between multiple defined goals.

RL for FL Client Selection To achieve an intelligent client selection that can learn and improve its strategy, several works have been presented that use Reinforcement Learning to optimize Federated Learning. An overview of existing approaches is given in Table I, where we compare our method to existing RL methods that aim to learn an optimal client selection in heterogeneous environments. A check mark in the table indicates that a feature is addressed in the paper, a minus sign denotes it is absent, and a tilde signifies that it is either not clear whether this feature is in the paper or it means that it could be applicable, but it is not evaluated. When considering the RL methods that have been used, one can observe that numerous approaches have been applied, but no clear preference towards one RL method has emerged. Therefore, there is no specific method that has become a standard in FL client selection settings yet.. Most methods not only consider non-IID data, but also device heterogeneity. Only about half of the RL methods have been trained in one setting and tested in a new environment. Only one work so far considers budget constraints related to cost; other existing work often selects a fixed number of clients, or the selected clients entirely depend on the action of the RL method. Few works are also scalable and tested in different environments. Some of the solutions have the same number of clients and change the number of selected clients. However, to the best of our knowledge, no work has shown the effects of client churn so far.

To the best of our knowledge, our article presents the first RL-based client selection work that shows the effect on the performance of clients joining and leaving during the FL training process, and that optimizes the RL training energy consumption. In addition, our method is the first one that not only considers a shared budget, but it also does not heavily rely on accurate pretraining, as it learns fast and is capable of adjusting to a changing number of clients.

III. System Model

In this section, we provide an overview of the system model, its components, and the information flow.

A. System Model and Federated Learning Process

Figure 1 illustrates the coordinated FL service with its elements and how they communicate with each other.

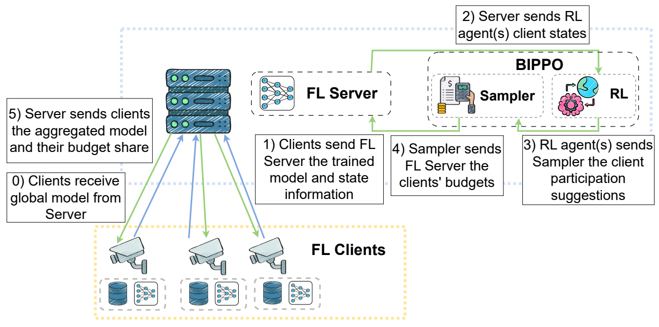

2) Server sends RL agent(s) client states
BIPPO
FL Server
Sampler
RL
5) Server sends clients the aggregated model and their budget share
0) Clients receive global model from Server
1) Clients send FL Server the trained model and state information
4) Sampler sends FL Server the clients' budgets
3) RL agent(s) sends Sampler the client participation suggestions
FL Clients

Fig. 1: Overview of our system design.

At the core of our system is the FL process, which is a distributed model training technique. In each training round t, a shared global model is adjusted by aggregating local updates coming from the participating clients. In detail, the FL process starts with the server sending the global model to the clients (Step 0). After every participating client receives the most recent global model parameters $\mathbf{w}^t$, each client n updates its local model parameters $\mathbf{w}_n^t$ with the global model $\mathbf{w}^t$ using gradient descent to minimize its local loss function $L_n(\mathbf{w}_n^t)$, which is defined as the average of the individual batch losses $l_j(\mathbf{w}_n^t)$ [1]:

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

At the beginning of a new FL workflow (round $t = 0$), the clients train for only one epoch, and the updated local model parameters $\mathbf{w}_n^{t+1}$ are sent to the server together with their state information, including an estimation of their energy consumption (Step 1). During each FL round t, the central server chooses a subset of candidate clients from the whole set to participate in the following training round. The choice of clients is determined by BIPPO, as detailed in Steps 2-4. In our system, we segment this selection phase into smaller steps, as we include an RL layer. First, the server sends the RL agent(s) the state information of the clients (Step 2). The RL agent(s) use their own model to validate possible actions in the given state and send them to the sampler (Step 3). In our model, we call these values, derived from the RL actor network(s), *participation suggestions*. More details on the sampler can be found in Section IV-D. The sampler decides on the final action, based on the participation suggestions and budget, and sends its decision to the server (Step 4). The server aggregates 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. The updated global model parameters $\mathbf{w}^{t+1}$ are then sent back to all clients for the next training round. The server then sends each client its budget share for the following round, together with the aggregated model (Step 5). If the received budget share is equal to or higher than the training cost, the client trains locally for several epochs. This completes the first Federated Learning round. In the following rounds, clients that received enough budget train their model based on the budget that they received and send their results back to the server (Step 1).

B. Energy Consumption Model

We model the energy cost as a function of the energy consumption for communication and the computation of the participating clients in one FL round. Our model is consistent with previous work [16], [27]–[30], and it assumes that this function only takes into account the variable energy consumption, which is the energy consumed by participating clients, but not other aspects of the FL process.

The first of two terms in the energy consumption computation is the **communication energy** [29], which is measured at each FL round as the energy required for letting the clients send model updates. To make our work comparable to related work, for the communication energy, we use the formula proposed by [10]. Specifically, we can express the communication energy as:

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

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

The second term in the energy consumption formula is the **computation energy**. We use the following expression to model it, which is typical in FL literature [28]–[30]:

$$E_n^{t,\text{comp}} = \frac{\alpha_n}{2} \eta_n |\tilde{D}_n| f_n^2 \tag{4}$$

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 $\frac{\alpha_n}{2}$.

Given the communication and computation energy terms, we can express the energy consumed by one participating client as:

$$E_n^{t,\text{total}} = E_n^{t,\text{comm}} + epochs \times E_n^{t,\text{comp}} \tag{5}$$

The total energy consumed by all participating clients in one round, which should stay within the energy budget, can therefore be formulated as:

$$E^{t,\text{total}} = \sum_{n \in P \subseteq N} \left(E_n^{t,\text{comm}} + \text{epochs} \times E_n^{t,\text{comp}}\right) \tag{6}$$

where P is the number of participating clients, i.e., a subset of all clients N.

In our proposed framework, the FL clients use this function to estimate their own energy consumption and send their energy estimate to the server at the beginning of the FL process. In a real-world deployment, these estimates could easily be replaced by energy cost measurements if possible, as discussed in Section IV-E.

IV. BIPPO: Budget-aware Independent PPO

The main premise of BIPPO is that RL-based methods can autonomously learn to find a good strategy for client selection and adapt over time. In RL, an agent learns a policy for selecting actions that lead to favorable subsequent states through interaction with the environment. RL is a popular approach for FL client selection [1], [5], [10], [17], [18], since it can learn complex patterns without having to define rules a priori, and it can adjust its strategy over time. A popular RL method for FL client selection is Double Deep Q-Networks (DDQN) [31], as shown by [5], [18]. However, the effectiveness of DDQN can be inconsistent due to the sensitivity of the hyperparameters chosen, as demonstrated by Sadiki et al. [32], who showed that batch size alone can make a huge performance difference. As an alternative, we investigate Proximal Policy Optimization (PPO) [33]. Compared to DDQN, PPO is an on-policy method that does not have a replay buffer. Therefore, the data collected during its training does not need to be stored for long; once it has been used to update the policy, it is discarded. This makes PPO also well-suited in our scenario, where we want to train the RL method on the go without depending on large amounts of stored experience. In addition, PPO is very effective in cooperative multi-agent settings [34]. Yu et al. [35] also recently showed that PPO and Multi-Agent PPO (MAPPO), in particular, outperform DDQN. PPO's effectiveness and robustness represent key desirable properties for our problem. Compared to MAPPO, *Independent PPO (IPPO)* [36] is a logically decentralized design where each agent has its own value and critic network, and they do not exchange local information. MAPPO uses a critic that takes into account the global state of all agents. When a new client joins, this influences the dimension of the global state space, and MAPPO would therefore need to retrain a new critic network. IPPO does not have this problem, since each agent has its own actor/critic networks, and their dimensions do not depend on the global state space.

A. Proximal Policy Optimization

PPO uses two neural networks, a policy network $\pi_\theta(a|s)$ that outputs a probability distribution over the actions and a value network $V_\phi(s)$ to estimate returns. A key feature of PPO is the clipped surrogate objective function, which prevents large policy updates and stabilizes performance.

After a given batch size, based on data collected in the last rounds, the critic network is used to calculate the generalized advantage estimation (GAE) [11], [33]. For each agent (I), the GAE can be calculated using the following formula:

$$A_t^i = \sum_{l=0}^{T} (\gamma\mu)^l \delta_{t+l}^i \tag{7}$$

where $\delta_t^i = r_t^i + \gamma V_\phi(s_{t+1}^i) - V_\phi(s_t^i)$ is the Temporal Difference error at time step t. $V_\phi(s_t^i)$ and $V_\phi(s_{t+1}^i)$ denote the estimated state-value function at time steps t and $t+1$, using the critic network V_ϕ. γ is the discount factor that controls the weighting of future rewards, and μ denotes the parameter used to govern the bias-variance trade-off in the estimation of the advantage function.

To stabilize training and prevent large, destabilizing updates to the policy, PPO optimizes a clipped surrogate objective function to calculate the policy loss for each agent (i):

$$L^{\text{CLIP}}(\theta_i) = \mathbb{E}_t^i \left[\min\left(r_t^i(\theta_i)\hat{A}_t^i, \text{clip}(r_t^i(\theta_i), 1-\epsilon, 1+\epsilon)\hat{A}_t^i \right) \right] \tag{8}$$

where

- $r_t^i(\theta_i) = \frac{\pi_{\theta_i}(a_t^i|s_t^i)}{\pi_{\theta_i^{\text{old}}}(a_t^i|s_t^i)}$ is the probability ratio between the new and old policies,
- $\hat{A}_t^i$ is the advantage estimate of agent i,
- ϵ is a hyperparameter controlling the trust region.

Additionally, the critic network parameters of each agent (i) ϕ_i are updated by minimizing the squared error between the predicted value and the observed return:

$$L^{\text{VF}}(\phi) = \mathbb{E}_t \left[\left(V_{\phi_i}(s_t^i) - R_t^i \right)^2 \right] \tag{9}$$

Where the return is defined as $R_t^i = A_t^i + V_{\phi_i^{old}}(s_t^i)$

B. Multi-Agent Reinforcement Learning

To make our RL approach adjust to a changing number of clients, we formulate it as a cooperative multi-agent solution. In cooperative MARL, a set of I agents is trained to produce optimal actions that maximize *team* reward [4]. At each time step t, every 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 act, the team receives a joint reward r^t and the environment transitions to the next state s_n^{t+1}. The overall objective is to maximize the total expected reward.

State: The state s_n^t represents the state of client n at time t. It is defined as $s_n^t = \{\overline{acc}^t, |D_n|, acc_n^t, e_n, p_n^t\}$. s_n^t contains context information, which in this case is the average accuracy $\overline{acc}^t$ achieved after communication round t. In addition, the state also consists of the 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, and whether client n participated in the last round p_n^t.

Action: In our FL setting, we consider 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. Details on the sampling strategy can be found in subsection IV-D.

Reward: The reward function is used to evaluate the effectiveness of the action a^t taken in state s^t. We define it as: $r(s^t, a^t) = \varphi\lambda^{(|\overline{acc}^t - \overline{acc}^{t-1}|)}$, where λ is a constant that makes the reward grow with the global test accuracy. Inspired by related work [5], we set λ to 64. When the last global accuracy is smaller than the current global accuracy, φ is 1; otherwise, it is -1 to decrease the participation probability of the clients that took part in that round. Details on why the reward was defined this way and how this definition helps with the energy-aware RL training can be found in Section IV-C. 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. It may be the case that an optimal selection includes only one client, which takes up all the budget, but it has a balanced dataset that helps improve performance. Our policy should learn these trade-offs automatically; neither our reward nor our sampler is optimized to select cheaper client, because they may decrease performance if their datasets are highly imbalanced.

C. Energy-Efficient RL Training

An important aspect of our energy-efficient framework is to guarantee that the RL training only consumes a minimal amount of energy. The energy consumed by the RL training is an aspect that, to the best of our knowledge, is not considered or evaluated in other RL-based FL client selection works. The first step for our energy-efficient training is to make it independent of the number of clients, so that when there is a large number of clients, we are still able to train our RL model on the go with only a minimal amount of energy. One design choice that supports the energy consumption stability is our MARL approach, where the state and action dimensions of each network are independent of the number of clients. However, with an increasing number of FL clients, there will also be more actor/critic networks. To keep energy consumption stable, we only let the clients who participated in an FL round update their policies; the other clients will not train their RL methods. This, however, leads to them only exploring one of the possible actions. To update the probabilities in our stochastic policy accordingly, it is important that we introduce a negative reward of comparable magnitude to the positive reward in order to adequately discourage non-participation and decrease the participation probability. The results of our evaluation V show that BIPPO is indeed able to learn a solid strategy, even if the models are only updated when the client participates. Due to our training design choice, we can save 90% of the energy used for RL training if only 10% of clients are selected. A lower energy budget leads to a decreasing number of participating clients, which makes the energy used for RL training also decrease accordingly. If more clients are included in the FL process, the training cost of BIPPO remains consistent. More details on the energy consumed by our RL model are in the evaluation section V-H.

D. Sampling Strategies

Traditionally, in IPPO, the action decision will be made for each actor network individually, by sampling over the output probability distributions. In our case, this, however, could lead to the global energy consumption in our FL process being too high or inefficiently used if not further controlled. Given our energy budget, we therefore need a global sampler that selects as many clients as possible, without going over the budget. Two different sampling strategies are evaluated for our sampler, both inspired by popular RL methods. The sampling strategies are called **stochastic policy sampling** and **ϵ-greedy sampling**, inspired by how actions are chosen by PPO and DDQN. Given the probabilities of the actor network, the stochastic action sampler of PPO randomly selects an action, where actions with higher probability are more likely to be chosen. However, the deterministic action selection strategy of DDQN chooses the action with the highest value. To guarantee exploration, DDQN needs an epsilon parameter that requires fine-tuning. The idea of epsilon is that whenever a random number is smaller than this epsilon, a random action is chosen. In each round, or every few rounds, epsilon decreases, reducing the chances of choosing a random action, which reduces the exploration. An overview of the client selection steps is provided in Algorithm 1.

As shown in the algorithm, each agent (i) first gets the probability for client i to participate in the next RL round a_i^t, given its policy (π_i) and the current state (s_i^t) of client i. The stochastic policy randomly selects a client considering its probability, while the ϵ-greedy sampler chooses a random client with probability ϵ, otherwise it chooses the client with the highest probability. After a client has been selected and added to the participating clients $\bar{\mathbf{a}}^t$, its probability is set to

Algorithm 1 Budget-Based Client Selection for IPPO with different samplers

```
for each agent $i = 1$ to $I$ do                              ▷ RL
    Get action probability for client $i$: $a_i^t = \pi_i(s_i^t)$
end for
cost $\leftarrow 0$
while budget not reached do                                ▷ Sampler
    if policy = stochastic then                              ▷ IPPO
        Sample client $c \sim \mathbf{a}^t = [a_i^t]_{i=1}^N$
    else if policy = $\epsilon$-greedy then                  ▷ BIPPO
        Sample $e \sim \mathcal{U}(0, 1)$
        if $e < \epsilon$ then
            $c \leftarrow$ random client
        else
            $c \leftarrow \arg\max_i \ a_i^t$
        end if
    end if
    if cost + energy($c$) $<$ budget then
        cost $\leftarrow$ cost + energy($c$)
        Add client $c$ to $\bar{\mathbf{a}}^t$
        Set $a_c^t \leftarrow 0$
    end if
end while
$\epsilon \leftarrow \max(\epsilon * 0.9, 0.05)$
```

0, and the process continues until the budget is reached. At the end of one client selection round, the exploration for the ϵ-greedy method is reduced by lowering ϵ.

Figure 2 illustrates the performance of BIPPO with the different samplers. The data for this plot was collected by running FL on Fashion-MNIST with 20 clients. Further configuration details can be found in the Subsection V-C. From this Figure, it is visible that the ϵ-greedy sampler used for our BIPPO method provides a more stable performance than the IPPO traditional stochastic sampler. Therefore, we have decided not to use the PPO traditional sampling mechanism, but instead the improved and more stable version.

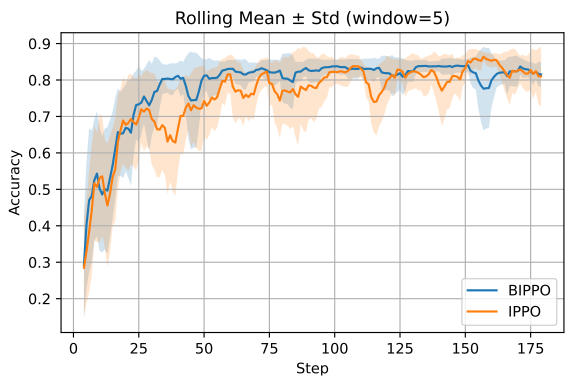


Fig. 2: Comparison of different sampling strategies.

E. Practical Considerations

BIPPO requires that clients report their energy costs. The accuracy of the reported values depends on the measuring capabilities of the client devices. For example, for popular NVIDIA GPUs, the `nvidia-smi` utility or the `NVML` library[2] can be used to retrieve GPU power draw information during training. However, accurate monitoring of training energy costs is challenging [37]. Regarding the energy cost of communicating the computation results of an FL round, this is a function of the time a transmitter is active sending data (itself depending on link quality; the poorer the signal conditions, the longer the time to transmit the results), the size of the ML model, and the transmission power used. These vary depending on the network topology and the connectivity technology. When the latter applies transmit power control (e.g., 4G/5G), a conservative estimate can be derived assuming that devices transmit at the maximum power allowed. In the absence of precise energy measurement capabilities, devices can resort to models for estimating energy cost. This is also the approach we follow in this work, applying models, definitions, and assumptions typical in the FL literature [16], [27], [28]. We briefly elaborate on this approach in Section III-B.

V. Evaluation

We evaluate BIPPO against a number of baselines, including PPO, IPPO, our implementation of the FedPPO method, the FedAvg random selection approach, the Power-of-Choice method, and our new baseline, Power-of-Choice-Accuracy, which selects the lowest performing clients that train on data that may be underrepresented in the global model. The analysis aims at checking the goodness of the learned strategy in picking the best clients, given the budget, to improve the model accuracy.

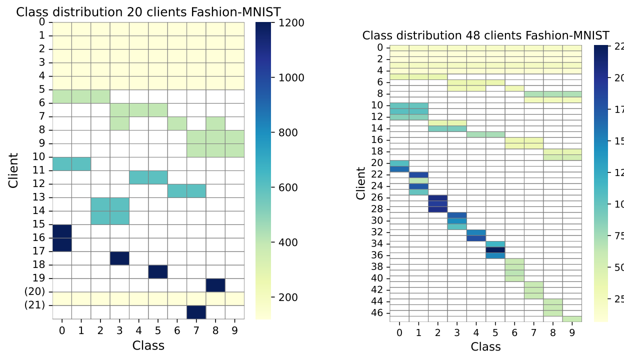
A. Experimental Setup

Datasets: For our tests, we chose two different classification datasets, which are often used in FL prediction tasks, Fashion-MNIST and CIFAR-10. The non-IID label distribution for 20 clients is shown in Figure 3a. The distribution for Fashion-MNIST and CIFAR-10 was the same; the only difference was that clients received fewer samples, as shown in Table II. In this figure, one can additionally see the two clients that join in the stability tests. The class distribution for 48 clients is in Figure 3b. Details on the models and the data size of each client are in Table II.

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.

Hyperparameters: The hyperparameters for the FL process, the RL models, and the ϵ-greedy sampler are listed in Table II.

[2]https://docs.nvidia.com/deploy/nvml-api/index.html



(a) 20 clients class distribution (b) 48 clients class distribution

Fig. 3: Comparison of class distributions for different numbers of clients.

TABLE II: FL, PPO, and ϵ-greedy sampler Hyperparameters

FL Hyperparameter	**Value**
Batch size	32
Local epochs	5
Learning rate	0.1
Fashion-MNIST model	Simple CNN [14]
Fashion-MNIST client training data	1.200
Fashion-MNIST client test data	200
CIFAR-10 model	Resnet-18 [38]
CIFAR-10 client training data	1.000
CIFAR-10 client test data	200
Energy Hyperparameters [29]	**Value**
Bandwidth (B)	1 MHz
Transmitter power (P_n)	0.5 W
Channel gain (h_n)	10^{-8}
Background noise power (σ)	10^{-10}
CPU cycles per bit	20
$\frac{\alpha_i}{2}$	2×10^{-28}
CPU-cycle frequency (f)	700, (1430), 1500 MHz
PPO Hyperparameter	**Value**
Batch size	2
Local epochs	8
Hidden layer size	128
Learning rate	0.001
Discount factor (γ)	0.9
Clip parameter	0.2
Bias-variance tradeoff (μ)	0.8
ϵ-greedy Hyperparameter	**Value**
Epsilon (ϵ)	1.0
Epsilon decay	0.9
Minimum epsilon ($\epsilon_{\min}$)	0.05

B. Baselines

To compare our method, we decided on various heuristic and RL baselines:

FedAvg (Heuristic): The sampler of this approach randomly selects the clients until the budget is reached [3], [17].

PoC (Heuristic): The *Power-of-Choice* method [26] selects the clients with the highest loss until the budget is reached. Using clients with the highest loss often improves model convergence, as shown by [26].

PoCA (Heuristic): The *Power-of-Choice-Accuracy* method is our adaptation and implementation of the Power-of-Choice method; instead of selecting the clients with the highest loss, it selects the clients with the lowest accuracy until the budget is reached. The lowest performing clients have data that may be underrepresented in the current global model.

FPPO (RL): FPPO gathers the participation suggestions for each client through a single network and chooses the ones with the highest values until reaching the budget threshold. To train the policy, it 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 [17] and adapted to our use case.

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. Non-IID and Heterogeneous Device Performance

These tests evaluate the influence of the different client selection methods on the overall FL model's performance. The data distribution is non-IID as described in Figure 3a, the clients have different device types, and the goal is to select clients that optimize performance in an energy-constrained environment. To simulate heterogeneous devices, we included 8 low- and 12 high-energy device types, which are evenly distributed, given by the CPU-cycle frequency in table II. The budget for the participating clients in each round is set to approximately 11% of the total energy that would be consumed if all clients were training for one round. Relative to the selected budget, a cheap client takes about 33% of this budget to train, and thus up to 3 clients of this type could be selected. Only one expensive client can be selected in each round, since each of them needs about 56% of this budget. We evaluate our approach in a setting with 20 clients and using the Fashion-MNIST dataset.

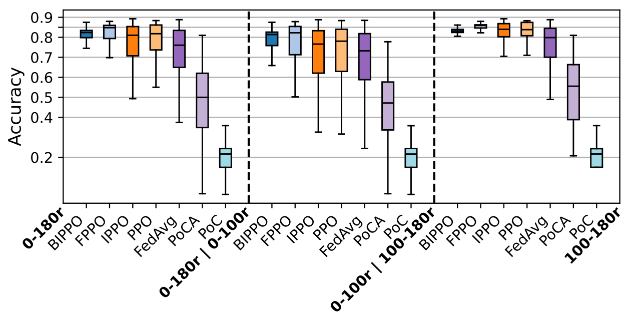


Fig. 4: 11% Energy budget, 20 clients training on Fashion-MNIST.

Looking at the results of Figure 4, one can see that BIPPO and FPPO have the best performance, since they achieve high accuracy and have a low interquartile range (IQR), which indicates stable performance. The RL methods with the traditional stochastic sampler (IPPO, PPO) also achieve high performance, but they have higher IQR ranges, indicating less consistency of the global performance. The heuristic approaches (FedAvg, PoCA, PoC) have lower performance. Especially in the last rounds, one can see a clear difference. PoC performed so badly in these first tests that we introduced PoCA, which we will use for the other evaluation sections

D. Generalizability

To see how well our results generalize to a different FL setting, we test our method on a more challenging FL dataset, which is trained on Resnet-18.

For these tests, we used 20 clients with two heterogeneous device types and the same data distribution as in the Fashion-MNIST tests. For the CIFAR-10 dataset, we set our energy budget at about 10%, so that it has roughly the same percentage as the first tests. The cheaper clients, which take about 32% of the budget, could be chosen 3 times, and the most expensive client with about 54% of the budget can only be selected once. In these tests, we included 8 cheaper and 12 expensive clients, the same distribution as in the first set of tests.

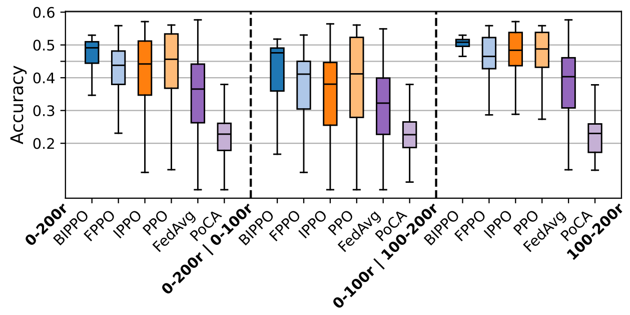


Fig. 5: Accuray distribution on CIFAR-10.

The results of the CIFAR-10 tests, visualized in Figure 5, show that BIPPO outperforms all other methods as it has the highest median and the smallest IQR, indicating a consistently high performance. The other RL methods have comparable performance, with the difference that FPPO has a slightly lower IQR range. The heuristic methods cannot keep up performance-wise with the RL methods, as they both have lower performance.

E. Transfer-Learning

Most existing work heavily relies on accurate pretraining. However, pretraining an RL method for FL is challenging, as to the best of our knowledge, there is no FL dataset that demonstrates the effect on the accuracy of selecting certain clients. This additional step not only costs additional energy for creating the training data, but also for performing the actual training. In real-world scenarios, one may not be able to traditionally train numerous RL episodes in the test environment or get an accurate training dataset that represents the real environment. To make our approach relevant in such settings, we demonstrated that even if accurate training data is not available, our energy-efficient BIPPO still manages to find a good strategy by learning on the go while adding only a minimum amount of additional energy, as demonstrated in Figure 12. To evaluate how much the performance would increase or decrease if a training setup with training data is available, we perform additional tests on pretrained RL. Since one usually does not have access to data for the exact same environment, we use the data collected from the Fashion-MNIST runs, pretrain our RL methods on this data, and use the trained RL methods for an FL process on CIFAR-10. This approach is inspired by transfer learning and may be used in real-world settings. It should be mentioned here that both setups differ in terms of dataset, FL model type, and energy consumption, but the data distribution and the distribution of cheaper and more expensive devices were the same.

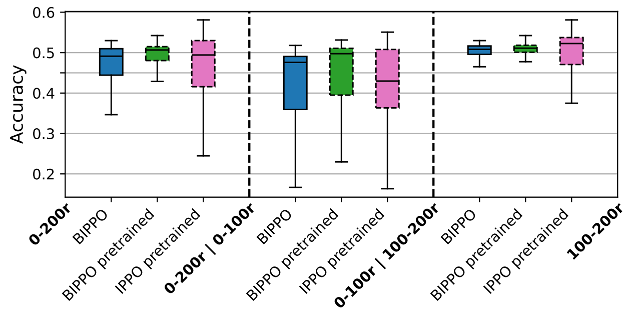


Fig. 6: Accuracy distribution of CIFAR-10 pretrained.

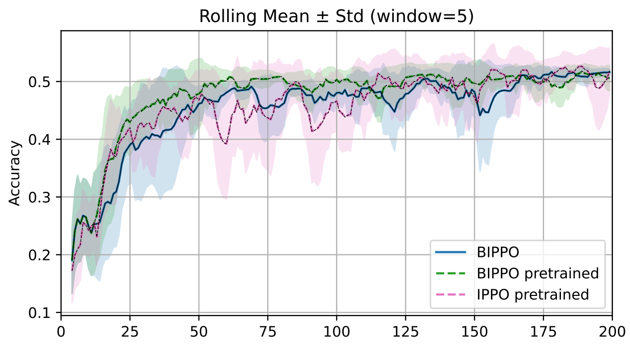


Fig. 7: CIFAR-10 pretrained, rolling mean +- standard deviation.

Figure 6 shows the performance of BIPPO in an FL setting that is training on CIFAR-10. We compare BIPPO, which is untrained and learning a completely new strategy, to a pretrained version that has been trained on the previous experiments with Fashion MNIST and fine-tunes its policy to the new setup. Not surprisingly, the pretrained RL method outperforms a non-pretrained method, especially in the first rounds. For the last 100 rounds, the mean accuracy of BIPPO is 0.485 for the non-pretrained method and 0.500 for the pretrained method. In addition, we compare BIPPO with a pretrained IPPO. IPPO reaches a mean accuracy of 0.495, which is slightly higher than that of BIPPO. Not surprisingly, pretraining the RL model can additionally increase the performance. However, pretraining adds additional energy cost for training and for generating training data. Additionally, in real-world settings, one may not have detailed information about data distribution to generate an accurate training environment. More details on the performance difference can be found in Figure 7, where one can see the rolling mean of the accuracy.

F. Stability

In a practical use case, IoT devices may join or leave the FL process over time. Therefore, it is important for an RL method to be robust to such changes. For the following tests, we do not focus on transferring previously learned knowledge to a new policy. Instead, the purpose of these experiments is to evaluate what each RL type (SARL, MARL) can achieve in terms of stabilizing the performance in dynamic environments. For the MARL approaches BIPPO and IPPO, we either added or removed the actor and critic networks for the clients that join or leave, while for the centralized SARL approaches PPO and FPPO, we initialized a new network to fit the new action and state dimensions.

After 100 rounds, two new clients join the FL process: one cheap client with a balanced dataset and one expensive client with an unbalanced dataset. In addition, after 150 rounds, two clients leave. To see the effect of clients leaving, we remove two (10% of the original) of the first five clients, which have a balanced dataset.

When comparing rounds 100-180, in which some clients join and leave, there is not much difference between the BIPPO and IPPO performance in Figures 4 and 8. However, as expected, the PPO and IPPO models have significantly higher performance drops than the MARL approaches, since they need to train from scratch when the state and action spaces change, while the MARL methods only have to learn new strategies for the new clients. The performance of FedAvg also decreased slightly, compared to Figure 4, while the PoCA remained more consistent. However, both have significantly lower accuracy values compared to our RL methods.

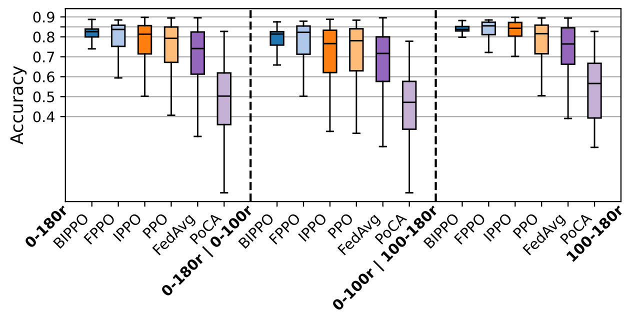


Fig. 8: Accuracy distribution in dynamic scenarios.

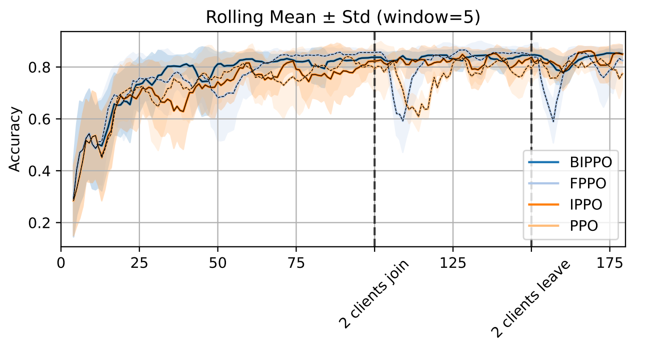


Fig. 9: Rolling mean +- standard deviation of RL methods in dynamic scenarios.

To provide a more detailed overview of the performance of our RL methods over time, we depict, in Figure 9, the rolling mean and standard deviation of BIPPO, IPPO, PPO, and FPPO. For the sake of clarity, we omit the line plots of the heuristic baselines, as their performance is much lower. This figure shows a significant performance degradation of PPO and FPPO when new clients join, while BIPPO and IPPO do not have a visible performance drop in such a case. When clients leave, all methods remain stable, except for the FPPO, where one can see that the accuracy significantly drops.

These tests demonstrate that the design of BIPPO, in fact, makes it robust to maintain its performance in dynamic environments where clients may join or leave, without requiring knowledge transfer optimization.

G. Scalability

To test whether our method will scale to a larger number of clients, we run experiments with 48 clients, which is more than double the number of clients from the previous settings, and it is within the limit of clients our hardware supports. To make the environment even more challenging, clients in this setting have 3 different device types, non-IID data, and their data sizes are also different. We ran the experiments for 300 rounds.

Our larger-scale tests show that BIPPO improves the performance, compared to IPPO, even in larger settings with more clients, as illustrated in Figure 11a. Compared to FPPO, it takes a bit longer to converge, shown in 11b. While FPPO takes about 75 rounds to reach a global accuracy of about 80%, BIPPO needs about 125 rounds. FPPO may find a good selection policy faster, but the energy cost for training FPPO in larger settings increases, compared to BIPPO, which manages to keep energy consumption consistent even in larger settings, as discussed in section V-H. Figure 10 shows that our RL methods again outperform heuristic approaches, with BIPPO and FPPO reaching higher accuracy than PPO and IPPO.

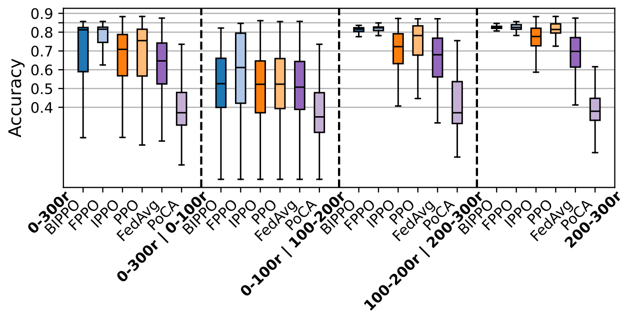
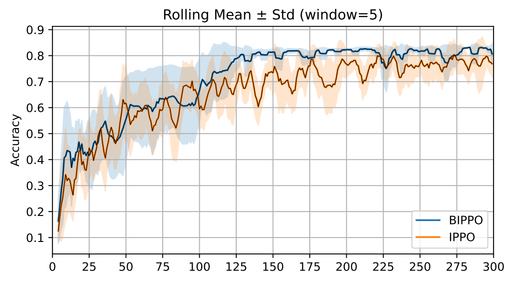
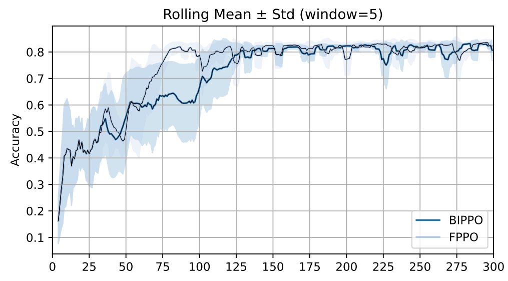


Fig. 10: Performance with 48 clients training on Fashion-MNIST.



(a) Rolling mean +- standard deviation of BIPPO and IPPO.



(b) Rolling mean +- standard deviation of BIPPO and FPPO.

Fig. 11: Comparison of BIPPO scalability results.

H. Sustainability

Since performance is not the only relevant factor of scalability, this section provides a comparison of how much additional computation load (and, in turn, energy cost) the RL methods incur. For the energy consumption comparison, we used the metric MACs (Multiply-Accumulate Operations). An approximate translation from FLOPs to MACs can be achieved by dividing the number of FLOPs by two [39]. Despite not being an exact measure of energy consumption [40], MACs are sufficient to show the relative difference between our RL and FL training.

Figure 12 provides an overview of the MACs it takes to pass all the data through the NN once. To get the energy consumed for training, this would have to be multiplied by the buffer size and by the number of epochs. Both of these parameters are the same for SARL and MARL For

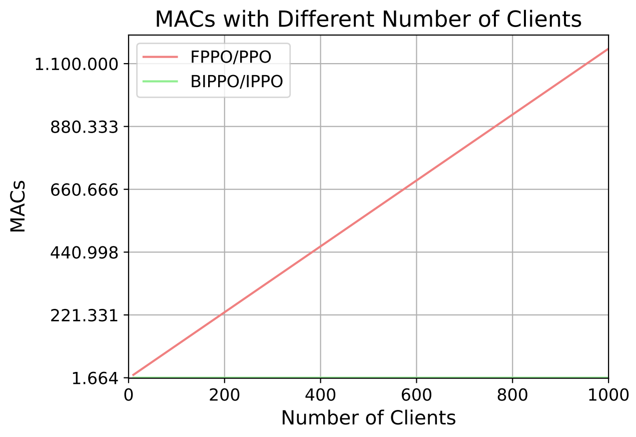


Fig. 12: MACs comparison between SARL and MARL.

these measurements, we used torchprofile.[3] When we have 20 clients, SARL needs 23.424 MACs, which is roughly 14 times the amount PPO needs for one NN pass with 1.664 MACs. In the case of our larger setting with 48 clients, SARL needs 55.680 MAC, which is more than double the MACs needed for 20 clients, while for BIPPO it stays consistent. Figure 12 shows the MACs for up to 1000 clients, where one can see that for SARL it increases linearly, but for MARL it stays consistent. The number of MACs to pass one image through the CIFAR-10 dataset is about 556.037.632, and for Fashion-MNIST it is 2.751.000. To get the number of MACs for one training client, this would have to be multiplied by the number of samples and by the epochs. Therefore, the energy consumed by the RL methods only adds minimal additional cost, which makes them useful in resource-constrained real-world scenarios. One important aspect to mention here is that the energy consumption for one round of training SARL and MARL depends on the budget. With a lower budget, fewer clients will be chosen, which leads to fewer training samples for training the RL methods. This lowers the energy consumption for RL training in low-budget environments.

To summarize, BIPPO has an energy consumption that, compared to SARL, stays consistent with an increasing number of clients, while it costs only a negligible proportion of the total budget. Due to its multi-agent design, both action selection and training can easily be parallelized for BIPPO, which reduces computation time and makes this approach scalable in larger environments.

VI. Discussion

Table III summarizes the main results of the evaluation. The best results are always bold, and the second-best results are underlined. A few rows are highlighted in grey. These rows are emphasized as these are metrics that, to the best of our knowledge, no other RL-based FL client selection work has considered. This table shows the average number of rounds until round x is reached, where round x is the round at which the average of the last 10 rounds is equal to or above the target accuracy [17]. For Fashion-MNIST, the target accuracy is 80%, while for CIFAR-10 it is 40%. For Fashion-MNIST, the target accuracy is inspired and taken from [41]. The average accuracy reached with CIFAR-10, given the budget and the non-IID environment, was between 35% for FedAvg and 45% for BIPPO. We therefore decided on the target accuracy of 40% as a representative mid-point.

The first finding from this table is that heuristic methods may consume less energy, but they cannot keep up performance-wise with the RL methods. The traditional PPO methods that use the stochastic sampler perform worse than the RL methods with the ϵ-greedy sampler. Table III shows that when it comes to the mean accuracy or mean number of rounds until round x is reached, BIPPO always performs best or second best. Our evaluation also demonstrates BIPPO's stable performance in dynamic environments, while exhibiting minimal and constant energy consumption.

In summary, BIPPO demonstrates good performance while it consumes only a minimal amount of additional energy, which is independent of the total number of FL clients. Importantly, it is also robust to clients joining and leaving, managing to maintain good performance in dynamic settings.

To improve convergence in later settings, in future work, we want to optimize transfer learning to improve the current strategy even more. We aim to pretrain a model that will generalize well over various settings to help MARL converge faster. Additionally, we want to consider security aspects, particularly in scenarios where clients may behave maliciously. The energy consumption in this work was estimated using a formula, which is a standard approach in related studies that facilitates comparison to existing work. Furthermore, the hyperparameters to calculate the energy consumed in each FL round were also taken from previous studies. However, in the future, we aim to measure actual energy consumption. The datasets used for this work are commonly used in related FL research, ensuring comparability; however, testing our approach on a dataset for a different learning task is an interesting future direction.

VII. Conclusion

This paper presents BIPPO, an RL-based method for selecting IoT devices to participate in an FL task within a limited energy budget, making it suitable for real-world scenarios. BIPPO quickly and autonomously learns a good strategy that outperforms conventional heuristic client selection methods and traditional RL methods in various environments, achieving higher mean accuracy. Compared to the Single-Agent RL (SARL) approach FPPO, BIPPO had comparable or improved accuracy in smaller settings. Still, BIPPO needed more time in larger settings to find a good selection strategy. However, FPPO's performance comes at a much higher cost, given that it consumes more than 14 times the energy consumed by BIPPO in smaller settings.

[3]https://github.com/zhijian-liu/torchprofile

TABLE III: Summary of the experiments, bold is the best result, and underlined is the second best.

Section	Challenge	Metric	BIPPO	FPPO	IPPO	PPO	FedAvg	PoCA
V-C	non-IID	mean of round x	43.0	**41.75**	66.75	43.0	88.25	-
V-C	non-IID	std of round x	**10.416**	13.663	17.297	12.748	41.342	-
V-C	non-IID	mean accuracy	0.779	**0.787**	0.751	0.759	0.701	0.491
V-C	non-IID	std of accuracy	**0.14**	0.147	0.157	0.157	0.158	0.163
V-D	Generalisability	mean of round x	**36.25**	38.0	42.0	44.5	80.25	-
V-D	Generalisability	std of round x	9.808	**9.67**	13.019	16.286	12.577	-
V-D	Generalisability	mean accuracy	**0.452**	0.415	0.413	0.427	0.352	0.226
V-D	Generalisability	sstd of accuracy	0.096	0.105	0.121	0.121	0.114	**0.053**
V-F	Stability	mean of round x	43.0	**41.75**	66.75	43.0	89.5	-
V-F	Stability	std of round x	**10.416**	13.663	17.297	12.748	43.396	-
V-F	Stability	mean accuracy after round x	**0.784**	0.767	0.754	0.739	0.698	0.495
V-F	Stability	std of accuracy after round x	**0.139**	0.158	0.158	0.158	0.16	0.16
V-G	Scalability	mean of round x	110.333	**72.667**	201.333	197.333	242.5	-
V-G	Scalability	std of round x	32.014	**9.843**	22.455	54.199	57.5	-
V-G	Scalability	mean accuracy	0.709	**0.736**	0.657	0.678	0.621	0.396
V-G	Scalability	std of accuracy	0.178	0.17	0.178	0.183	**0.164**	0.133
V-H	Scalability	RL NN MACs 20 clients	**1.664**	23.424	**1.664**	23.424	-	-
V-H	Scalability	RL NN MACs 48 clients	**1.664**	52.096	**1.664**	52.096	-	-

The energy consumption for SARL scales with the number of clients, making it even higher in larger settings, while for BIPPO it remains consistent. Importantly, compared to SARL, BIPPO also allows seamless integration of new clients without the need to retrain the entire system from scratch. The evaluation showed that BIPPO indeed manages to keep its performance in dynamic conditions where clients join and leave.

VIII. Acknowledgment

This work has been supported by the European Union's Horizon Europe under grant agreements No. 101079214 (AIoTwin) and No. 101135576 (INTEND). Thanks to Axel Brunnbauer for the valuable help and expert feedback on RL and PPO.

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] K. S. Awaisi, Q. Ye, and S. Sampalli, "A survey of industrial aiot: Opportunities, challenges, and directions," *IEEE Access*, 2024.

[3] 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.

[4] 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.

[5] 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.

[6] W. Gao, O. Tavallaie, S. Chen, and A. Zomaya, "Federated learning as a service for hierarchical edge networks with heterogeneous models," in *International Conference on Service-Oriented Computing*. Springer, 2024, pp. 85–99.

[7] A. Heidari, N. J. Navimipour, and M. Unal, "Applications of ml/dl in the management of smart cities and societies based on new trends in information technologies: A systematic literature review," *Sustainable Cities and Society*, vol. 85, p. 104089, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670722004061

[8] 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.

[9] B. Azhati and X. Li, "Optimizing client selection for federated learning: a ppo-based method," in *2024 3rd International conference on big data, information and computer network (BDICN)*. IEEE, 2024, pp. 199–203.

[10] 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.

[11] 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.

[12] T. Yu, X. Wang, J. Yang, and J. Hu, "Proximal policy optimization-based federated client selection for internet of vehicles," in *2022 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2022, pp. 648–653.

[13] 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.

[14] I. Čilić, A. Lackinger, P. Frangoudis, I. P. Žarko, A. Furutanpey, I. Murturi, and S. Dustdar, "Reactive orchestration for hierarchical federated learning under a communication cost budget," in *Proc. IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2025.

[15] T. Koliopoulos and P. Kouloumbis, "Urban computing and smart cities: Web utilities characteristics that support sustainable smart cities," in *Resilient and Responsible Smart Cities: Volume 2*. Springer, 2022, pp. 115–124.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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

[21] H. Chen and H. Vikalo, "Heterogeneity-guided client sampling: Towards fast and efficient non-iid federated learning," *Advances in*

Neural Information Processing Systems, vol. 37, pp. 65 525–65 561, 2024.
[22] S. Rai, A. Kumari, and D. K. Prasad, "Client selection in federated learning under imperfections in environment," *AI*, vol. 3, no. 1, pp. 124–145, 2022.
[23] Y. J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 10 351–10 375.
[24] S. Mayhoub and T. M. Shami, "A review of client selection methods in federated learning," *Archives of Computational Methods in Engineering*, vol. 31, no. 2, pp. 1129–1152, 2024.
[25] 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.
[26] 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.
[27] 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.
[28] 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.
[29] 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.
[30] 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.
[31] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.
[32] A. Sadiki, J. Bentahar, R. Dssouli, A. En-Nouaary, and H. Otrok, "Deep reinforcement learning for the computation offloading in mimo-based edge computing," *Ad Hoc Networks*, vol. 141, p. 103080, 2023.
[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[34] 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.
[35] T. Yu, X. Wang, J. Hu, and J. Yang, "Multi-agent proximal policy optimization-based dynamic client selection for federated ai in 6g-oriented internet of vehicles," *IEEE Transactions on Vehicular Technology*, 2024.
[36] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, "Is independent learning all you need in the starcraft multi-agent challenge?" 2020. [Online]. Available: https://arxiv.org/abs/2011.09533
[37] Z. Yang, K. Adámek, and W. Armour, "Accurate and convenient energy measurements for gpus: A detailed study of NVIDIA gpu's built-in power sensor," in *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC24)*, 2024.
[38] 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, pp. 770–778.
[39] J. Getzner, B. Charpentier, and S. Günnemann, "Accuracy is not the only metric that matters: Estimating the energy consumption of deep learning models," *arXiv preprint arXiv:2304.00897*, 2023.
[40] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *2017 51st asilomar conference on signals, systems, and computers*. IEEE, 2017, pp. 1916–1920.
[41] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1078–1088, 2021.

Anna Lackinger received the MSc degree from the Technical University of Vienna, Austria, in 2023, with distinction in the field of computer science. She is now working toward a PhD degree in the Distributed Systems Group at the Technical University of Vienna in Austria. Her research interests include edge intelligence and machine learning.

Andrea Morichetta Andrea Morichetta is a postdoctoral researcher at the Distributed Systems Group of TU Wien, specializing in machine learning, distributed systems, and edge-to-cloud computing. Dr. Morichetta holds a Ph.D. in Electrical, Electronics, and Communication Engineering from Politecnico di Torino and possesses a strong international profile. He has collaborated with leading institutions and industry partners such as Cisco (San Jose, CA, US), AIT (Vienna, AT), Futurewei Technologies (Seattle, WA, US), and Tsinghua University (Beijing, CN). With extensive teaching experience, Dr. Morichetta lectures in Distributed Systems at TU Wien, where he leads both Bachelor and Master-level courses. His research is contributing to the study of complex, large-scale distributed systems, through modular, multi-modal coordination.

Pantelis A. Frangoudis is a researcher with the Distributed Systems Group, TU Wien, Austria. He has been with the Communication Systems Department, EURECOM, France (2017–2019), and with team DIONYSOS at IRISA/INRIA Rennes, France (2012–2017), which he originally joined under an ERCIM "Alain Bensoussan" post-doctoral fellowship. He has a Ph.D. (2012) in Computer Science from AUEB, Greece. His interests include mobile and wireless networking, edge and cloud computing, and Internet multimedia.

Schahram Dustdar (Fellow, IEEE) is a Full Professor of Computer Science (Informatics) with a focus on Internet Technologies heading the Distributed Systems Group at the TU Wien. He is Editor-in-Chief of Computing (Springer) and an Associate Editor of IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, ACM Computing Surveys, ACM Transactions on the Web, and ACM Transactions on Internet Technology, as well as on the editorial board of IEEE Internet Computing and IEEE Computer. He is recipient of multiple awards: TCI Distinguished Service Award (2021), IEEE TCSVC Outstanding Leadership Award (2018), IEEE TCSC Award for Excellence in Scalable Computing (2019), ACM Distinguished Scientist (2009), ACM Distinguished Speaker (2021), IBM Faculty Award (2012). He is a Member of Academia Europaea und IEEE Fellow.