FedGCS: A Generative Framework for Efficient Client Selection in Federated Learning via Gradient-based Optimization

Zhiyuan Ning[1,2], **Chunlin Tian**[4], **Meng Xiao**[1,2], **Wei Fan**[5], **Pengyang Wang**[4], **Li Li**[4*], **Pengfei Wang**[1,2*] and **Yuanchun Zhou**[1,2,3]
[1]Computer Network Information Center, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Hangzhou Institute for Advanced Study, UCAS
[4]Department of Computer and Information Science, IOTSC, University of Macau
[5]University of Oxford
ningzhiyuan@cnic.cn, yc27402@um.edu.mo, shaow@cnic.cn, wei.fan@wrh.ox.ac.uk,
{pywang, llili}@um.edu.mo, {pfwang, zyc}@cnic.cn

Abstract

Federated Learning faces significant challenges in statistical and system heterogeneity, along with high energy consumption, necessitating efficient client selection strategies. Traditional approaches, including heuristic and learning-based methods, fall short of addressing these complexities holistically. In response, we propose FedGCS, a novel generative client selection framework that innovatively recasts the client selection process as a generative task. Drawing inspiration from the methodologies used in large language models, FedGCS efficiently encodes abundant decision-making knowledge within a continuous representation space, enabling efficient gradient-based optimization to search for optimal client selection that will be finally output via generation. The framework comprises four steps: (1) automatic collection of diverse "selection-score" pair data using classical client selection methods; (2) training an encoder-evaluator-decoder framework on this data to construct a continuous representation space; (3) employing gradient-based optimization in this space for optimal client selection; (4) generating the final optimal client selection via using beam search for the well-trained decoder. FedGCS outperforms traditional methods by being more comprehensive, generalizable, and efficient, simultaneously optimizing for model performance, latency, and energy consumption. The effectiveness of FedGCS is proven through extensive experimental analyses.

1 Introduction

Federated Learning (FL), as introduced in [McMahan *et al.*, 2017], enables multiple client devices to collaboratively train a shared model without exposing users' raw data, preserving privacy. Despite its advantages, the practical deployment of FL still faces various challenges: (1) statistical heterogeneity, *i.e.*, data owned by different clients may come from different distributions, resulting in non-independent, identically distributed (Non-IID) training data, which can seriously affect the convergence and overall performance of the global model [Hsieh *et al.*, 2020]; (2) system heterogeneity, *i.e.*, clients participating in FL have different degrees of computational resources, communication capabilities and fault tolerance, which may result in stragglers that hinder the FL training process inflicting intolerable latency [Bonawitz *et al.*, 2019]; (3) the devices perform many compute-intensive data processing and model training, and the devices and the central server frequently exchange model parameters and gradients, which will lead to substantial energy consumption. Thus, the inclusion of too many clients may lead to suboptimal performance, latency, and wasted energy. The selection of a "good" subset of clients as FL participants for each training round is critical to mitigating all three of these issues [Lai *et al.*, 2021].

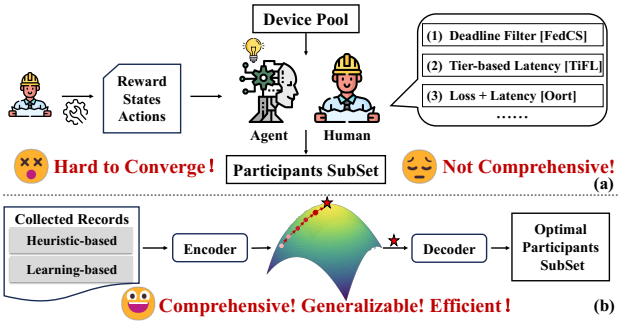


Figure 1: (a) Right: heuristic-based selection, which utilizes manually developed heuristic policies to select participating devices. (a) Left: learning-based selection, which employs RL to train continuously self-optimizing agents for optimal client selection. (b): FedGCS formulates client selection as a generative task.

Existing client selection methods only partially address the above three challenges and can be categorized into two types: (1) *heuristic-based selection*, as shown in Figure 1(a) right, in

*Corresponding author.

which manually-engineered heuristic rules are used to choose participating devices to improve the model performance and training efficiency [Chai *et al.*, 2020; Lai *et al.*, 2021]. However, such methods are subjectively designed by human experts and are not comprehensive enough that consider only a limited number of optimization objectives and deployment scenarios. As a result, when confronted with unfamiliar and complex scenarios, a great deal of domain expertise and manual tuning is often required, and the final performance may be suboptimal; (2) *learning-based selection*, as shown in Figure 1(a) left, which explores the use of reinforcement learning (RL) [Sutton and Barto, 2018] to train continuously self-optimizing agents for optimal client selection [Wang *et al.*, 2020; Zhang *et al.*, 2022]. However, the objectives designated by human experts for RL training still do not handle the evolving and complex real-world deployment scenarios well [Kim and Wu, 2021]. Moreover, RL is developed based on making decisions in massive discrete spaces and therefore has difficulty converging compared to solving continuous optimization problems [Dulac-Arnold *et al.*, 2021].

Recently, large language models (LLMs) trained on large volumes of text data have shown impressive abilities to encode world knowledge into model parameters and solve a wide variety of natural language processing tasks via text generation [Brown *et al.*, 2020; Touvron *et al.*, 2023]. Such great success has driven various domains to migrate to generative models [Ramesh *et al.*, 2022; Gruver *et al.*, 2023], where all the information and knowledge required for a task is encoded into the model parameters and the task is ultimately solved in a generative manner. This is a general, flexible, comprehensive, and efficient modeling style that prompts us to consider: ***can client selection in FL also be effectively addressed in a similar generative way?*** To this end, we propose **FedGCS**, a novel **G**enerative **C**lient **S**election framework for **Fed**erated learning, that formulates the discrete client device selection task as a generative task. Like LLMs, FedGCS tries to encode the wide knowledge of discrete decisions (*i.e.*, selection or deselection of clients) into a continuous representation space, where gradient-based optimization can be efficiently applied to find the optimal representation that will eventually be output as the discrete selection format via generation (as shown in Figure 1(b)). Specifically, FedGCS includes four steps: (1) utilizing classical client selection approaches, *i.e.*, heuristic-based and learning-based selection, to automatically collect sufficient, diverse and high-quality "selection-score" pair data as training data for subsequent models; (2) training an encoder-evaluator-decoder framework by simultaneously optimizing the sequence-to-sequence loss [Sutskever *et al.*, 2014] and score estimation loss based on the collected pair data, and thereby obtaining a continuous representation space on which selection optimization will be performed; (3) adopting gradient-based optimization in the continuous representation space to find the optimal client selection representation; (4) applying the beam search strategy [Freitag and Al-Onaizan, 2017] to the well-trained decoder to generate the optimal client selection based on the optimal representation.

FedGCS has the following advantages over previous methods: (1) FedGCS encodes abundant experience of various different classical algorithms into the same continuous space

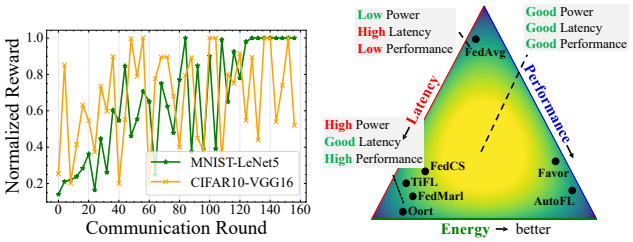


Figure 2: Left: reward convergence curve of Value Decomposition Network [Sunehag *et al.*, 2017] for RL agent in two client selection tasks. Right: ternary depiction of the 3D optimization space.

to determine the final decision, and simultaneously considers the three metrics of model performance, latency and energy, making it more ***comprehensive*** than the traditional methods; (2) through data-driven learning, FedGCS only needs the automatically collected "selection-score" pair data as input and avoids the overly complex and cumbersome manual interventions and tuning of previous methods, therefore FedGCS is more flexible and ***generalizable***; (3) modeling the process of determining the optimal client selection as executing gradient-based optimization in a continuous space is more ***efficient*** than performing a heuristic rule-based search in a large discrete space or training hard-to-converge RL agents. Finally, we conduct extensive experiments and analyses to demonstrate the effectiveness and superiority of FedGCS.

2 Related Works

2.1 Client Selection in Federated Learning

Client selection in FL is critical to optimizing communication costs, computational resources, and overall performance [Lai *et al.*, 2021], which falls into two main categories: (1) *heuristic-based selection* that predominantly relies on heuristics rooted in isolated considerations such as data heterogeneity and energy efficiency [Li *et al.*, 2019; Cho *et al.*, 2020]. Specifically, Oort [Lai *et al.*, 2021] employs analytical strategies that comprehensively address the multifaceted nature of device selection. However, adapting these heuristics to unseen scenarios often demands lots of domain-specific expertise and extensive tuning. (2) *learning-based selection* that devises policies for device selection through RL [Sutton and Barto, 2018] to formulate decisions as Markov decision processes. Specifically, AutoFL [Kim and Wu, 2021] leverages a Q-table, Favor [Wang *et al.*, 2020] adopts Q-learning, and FedMarl [Zhang *et al.*, 2022] utilizes multi-agent RL to achieve their objectives. However, training RL agents is difficult to converge [Dulac-Arnold *et al.*, 2021], especially in the face of noisy and complex real-world environments (as shown in Figure 2 left). Finally, as shown in Figure 2 right, all these methods mentioned above usually focus on only one or two optimization metrics and lack a comprehensive global consideration of performance, latency, and energy.

2.2 Generative Models

Recent advancements in LLMs and generative models mark a shift in the machine learning paradigm, attesting to the evolution from task-specific architectures to highly generalized

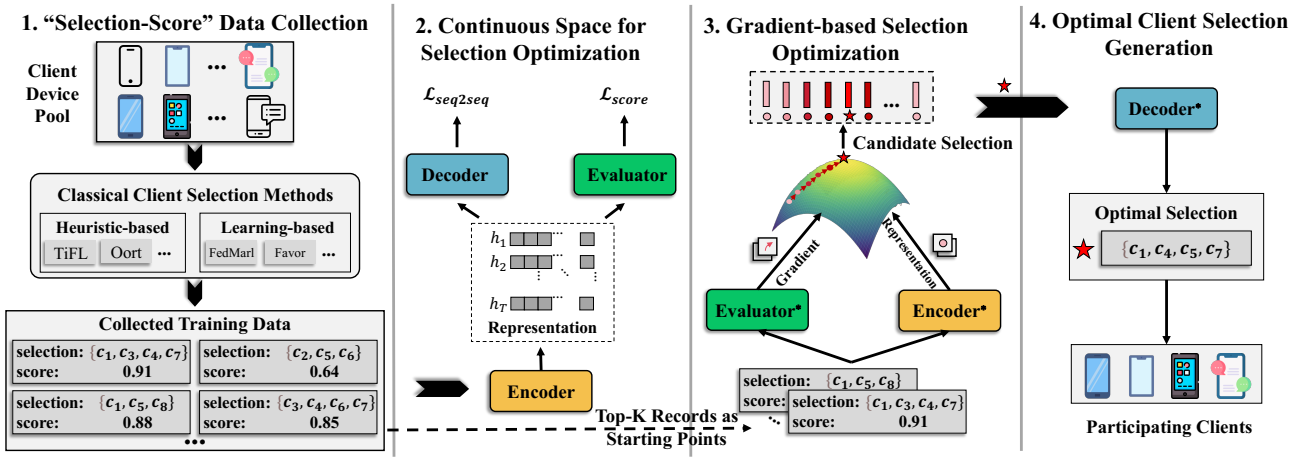


Figure 3: Framework overview of FedGCS: (1) efficiently collecting sufficient, diverse, comprehensive and high-quality training data; (2) preserving the knowledge of classical client selection methods into a global continuous representation space; (3) searching for better representation in the learned space via gradient-based optimization; (4) outputting the optimal device subset via generation.

generic models. These models, exemplified by the likes of GPT-3 [Brown *et al.*, 2020] in the language domain, along with DALL·E [Ramesh *et al.*, 2022] in the multi-modal domain, have driven the migration to generative models across various domains [Gruver *et al.*, 2023; Xiao *et al.*, 2023; Wang *et al.*, 2024]. Generative models have showcased an unparalleled ability to encode multifaceted knowledge representations from expansive data into their parameters [Brown *et al.*, 2020]. This encoded knowledge can be adeptly maneuvered to generate contextually coherent and semantically rich outputs, addressing a wide spectrum of downstream tasks. Despite the wide range of applications in language and vision, we have not seen generative models applied extensively in decision-making tasks like the client selection task in FL.

3 Problem Formulation

In this section, we formulate the discrete client selection task in FL as a generative task and conduct gradient-based optimization in a continuous representation space to get the optimal selection. Firstly, at the beginning of a training round, given a candidate client device pool set $\mathcal{C} = \{c_1, c_2, \dots c_J\}$ of J client devices, we use classic client selection algorithms to collect n "selection-score" pair as training data for subsequent model training. We denote the collected records as $\mathcal{R} = \{(\mathbf{s_i}, p_i)\}_1^n$, where $\mathbf{s_i} = \{s_1, s_2, \dots s_T\}$ is the client selection record made by a classic algorithm ($\mathbf{s_i} \subset \mathcal{C}$, $s_{1 \le i \le T} \in \mathcal{C}$ is the selected device IDs, and T is not a fixed number that is determined by the classical algorithm used for this selection), and p_i is the corresponding comprehensive score of the FL performance for making the client selection $\mathbf{s_i}$. To characterize p_i of a given $\mathbf{s_i}$, unlike previous FL approaches [Li *et al.*, 2019; Zhan *et al.*, 2020; Wang *et al.*, 2020] that focus on different metrics and typically consider only one or two metrics to guide the selection, in this paper, we simultaneously focus on: (1) *good performance* of the global model; (2) *low processing latency*; (3) *low energy consumption* to perform comprehensive and multi-dimensional optimization. Here, the comprehensive metric function M of a client selection $\mathbf{s_i}$ is defined as:

$$p_i = M(\mathbf{s_i}) = p_{perf.} \times \left(\frac{L}{p_L}\right)^{\mathbf{1}(L<p_L)\times a} \times \left(\frac{E}{p_E}\right)^{\mathbf{1}(E<p_E)\times b}, \quad (1)$$

where $p_{perf.}$ denotes the downstream task performance. L and E are developer-specified latency and energy budgets for the devices. p_L and p_E represent the actual total latency and energy consumption, encompassing communication and computation processes, for a given client selection $\mathbf{s_i}$. $\mathbf{1}(x)$ is an indicator function that takes the value 1 if x is true and 0 otherwise. Therefore, client selections exceeding the desired latency L and energy E will be penalized by developer-specified factors a and b [Lai *et al.*, 2021], both set to 2 in our implementation. Then, we aim to learn a continuous representation space $\mathbb{S}$ for "selection-score" pair data. Specifically, we learn three modules: (1) an encoder ϕ which can map a selection $\mathbf{s_i}$ to its corresponding continuous representation $E_{\mathbf{s_i}}$; (2) an evaluator ω which can evaluate the comprehensive score p_i of a selection $\mathbf{s_i}$ based on its representation $E_{\mathbf{s_i}}$; (3) a decoder ψ which can map a continuous representation $E_{\mathbf{s_i}}$ back to its corresponding discrete client selection $\mathbf{s_i}$. We optimize all these modules simultaneously based on the collected records $\mathcal{R}$. After getting the space $\mathbb{S}$ and the well-learned three modules, we can adopt gradient-based optimization in $\mathbb{S}$ to find the optimal client selection $\mathbf{s}^*$, given by:

$$\mathbf{s}^* = \psi\left(E_{\mathbf{s}^*}\right) = \psi(\text{argmax}_{E_{\mathbf{s_i}} \in \mathbb{S}}\, \omega(E_{\mathbf{s_i}})). \quad (2)$$

4 Methodology

Figure 3 shows the overview of our framework, which will be described in detail in this section.

4.1 "Selection-Score" Data Collection

To ensure that subsequent models and algorithms perform well, we need to collect sufficient, diverse, comprehensive and high-quality "selection-score" pair data $\mathcal{R} = \{(\mathbf{s_i}, p_i)\}_1^n$

as training data: training data is large enough to represent the entire distribution; training data includes high-performance selection cases as well as certain random exploration samples. Intuitively, we can use random selections and compute their corresponding comprehensive scores as training data. However, this strategy is inefficient because the number of random selections is exceptionally large and most of them are low-performance samples. Therefore, we also must consider the efficiency of our data collection process.

With the help of classical client selection algorithms, we can automatically and efficiently collect training data that fulfills our requirements in two ways: (1) *heuristic-base collection:* using heuristic-base selection methods (e.g., Oort [Lai *et al.*, 2021]) to generate records. This category of algorithms consists of expert manually-engineered heuristic rules, and different algorithms reflect different experts' considerations, based on which we can produce comprehensive and high-quality "selection-score" records reflecting various expert intelligence. (2) *learning-based collection:* treating learning-based selection methods (e.g., FedMarl [Zhang *et al.*, 2022]) as exploration-based training data collectors. This stochastic and self-optimizing learning process can be viewed as an exploratory and automated data collection tool that leverages machine intelligence to help collect diverse yet quality records. For the collected selection samples, the comprehensive metric function M allows us to obtain their corresponding scores easily, thus finally finishing the pair data collection.

4.2 Continuous Space for Selection Optimization

After collecting "selection-score" pair data $\mathcal{R}$ that reflects the broad and abundant experience of classical client device selection algorithms, we train an encoder-evaluator-decoder framework based on $\mathcal{R}$ to embed these records into a continuous representation space $\mathbb{S}$ for follow-up gradient-based selection optimization. Each point in $\mathbb{S}$ is associated with a client selection and its corresponding comprehensive score, denoted below by $\mathbf{s}$ and p, respectively.

Data Augmentation. Sequence-to-sequence (seq2seq) architecture [Sutskever *et al.*, 2014] is an important machine learning method: it uses an encoder to capture the context of the input sequence and sends it to a decoder, which then produces the final output sequence. Its flexibility and powerful capabilities have led to a wide range of applications where the data can be naturally organized as a sequence [Raffel *et al.*, 2020]. The encoder and decoder together in our framework essentially belong to the seq2seq paradigm as well, with the particularity that the inputs and outputs (*i.e.*, client selections) are a special kind of sequences—the sets. Each client selection is a subset of the device pool that contains an unordered sequence of device IDs. To model the order-independent properties of sets in the seq2seq component, we propose to use data augmentation to increase the diversity of sequences corresponding to the same set. Specifically, given a selection and its corresponding score from $\mathcal{R}$, we randomly shuffle the device IDs contained in the selection to obtain a new order, then we pair the new shuffled device IDs sequence with the original score and add them to the training data.

Encoder ϕ. The encoder aims to map any given selection $\mathbf{s} = \{s_1, s_2, \dots s_T\}$ to continuous representation $E_\mathbf{s} = \phi(\mathbf{s})$ that is considered as the input context. Here, we adopt a single layer long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] as encoder ϕ. Specifically, we input the device IDs from $\mathbf{s}$ sequentially into the encoder and collect their corresponding output embeddings to form $E_\mathbf{s} = [h_1, h_2, \dots, h_T] \in \mathbb{R}^{T\times d}$ (T is the length of the selection $\mathbf{s}$, and d is the embedding dimension).

Decoder ψ. The decoder aims to generate the device IDs of a client selection $\mathbf{s}$ based on $E_\mathbf{s}$, denoted by $\mathbf{s} = \psi(E_\mathbf{s})$. Similar to the encoder, we employ a single-layer LSTM to implement the decoder and train it in an autoregressive way [Sutskever *et al.*, 2014; Brown *et al.*, 2020]. Firstly, the initial input of the decoder is the last device ID embedding in $E_\mathbf{s}$ (*i.e.*, h_T). Then, at step i of the decoding process, we can get the current decoder hidden state $\hat{h}_i$ from the LSTM, and utilize dot product attention [Luong *et al.*, 2015] to aggregate the input context $E_\mathbf{s}$ from encoder, and finally obtain an enhanced input embedding $\tilde{h}_i$ for this step, defined as:

$$\tilde{h}_i = \sum_{h_j \in E_\mathbf{s}} a_{ij} h_j, \text{ where } a_{ij} = \frac{\exp\left(\hat{h}_i \cdot h_j\right)}{\sum_{h_k \in E_\mathbf{s}} \exp\left(\hat{h}_i \cdot h_k\right)}, \quad (3)$$

where a_{ij} is the attention weight between $\hat{h}_i$ and h_j. Later, we concatenate $\hat{h}_i$ and $\tilde{h}_i$ together and fed them into a fully connected layer followed by a softmax layer to produce the predictive distribution of step i, formulated as:

$$P_\psi\left(s_i \mid \mathbf{s}_{<i}, E_\mathbf{s}\right) = \frac{\exp\left(W_{s_i}\left[\hat{h}_i; \tilde{h}_i\right]\right)}{\sum_{c\in\mathcal{C}} \exp\left(W_c\left[\hat{h}_i; \tilde{h}_i\right]\right)}, \quad (4)$$

where $s_i \in \mathbf{s}$ is the i-th ID in $\mathbf{s}$, $\mathbf{s}_{<i}$ represents the prediction of the previous or initial step, $\mathcal{C}$ is the candidate client device pool set (*i.e.*, the token set in seq2seq models), and W stand for the parameter of the fully connected layer. By multiplying the probability in each step, we can derive the distribution of the whole client selection $\mathbf{s}$ as follows:

$$P_\psi(\mathbf{s} \mid E_\mathbf{s}) = \prod_{t=1}^{T} P_\psi\left(s_i \mid \mathbf{s}_{<i}, E_\mathbf{s}\right). \quad (5)$$

In order to make the generated sequence similar to the true sequence, we minimize the negative log-likelihood of the distribution, which is defined as:

$$\mathcal{L}_{seq2seq} = -\log P_\psi(\mathbf{s} \mid E_\mathbf{s}) = -\sum_{t=1}^{T} \log P_\psi\left(s_i \mid \mathbf{s}_{<i}, E_\mathbf{s}\right). \quad (6)$$

Evaluator ω. The evaluator aims to evaluate the corresponding comprehensive score p of a selection $\mathbf{s}$ based on its representation $E_\mathbf{s}$. Specifically, we first conduct mean value operation on device ID embeddings $h_{(.)}$ in $E_\mathbf{s}$ to aggregate the information and obtain the integrated selection embedding $\bar{E}_\mathbf{s} \in \mathbb{R}^d$. We then fed $\bar{E}_\mathbf{s}$ into the evaluator ω (a feedforward neural network) to estimate the score, given as $\hat{p} = \omega(\bar{E}_\mathbf{s})$. To minimize the difference between the estimated score $\hat{p}$ and the collected real one p, we leverage the Mean Squared Error (MSE), defined by:

$$\mathcal{L}_{score} = \text{MSE}(p, \hat{p}) = (p - \hat{p})^2. \quad (7)$$

Joint Training Loss $\mathcal{L}$. We optimize the encoder ϕ, decoder ψ and evaluator ω simultaneously by integrating Equation 6 (*i.e.*, the seq2seq loss) and Equation 7 (*i.e.*, the score estimation loss) to construct the joint training loss $\mathcal{L}$:

$$\mathcal{L} = \alpha \mathcal{L}_{seq2seq} + (1-\alpha)\mathcal{L}_{score}, \tag{8}$$

where α is the trade-off hyperparameter to control the contribution of $\mathcal{L}_{seq2seq}$ and $\mathcal{L}_{score}$ during the training process.

4.3 Gradient-based Selection Optimization

After obtaining the trained encoder, evaluator, and decoder, we can adapt the gradient-based optimization method in $\mathbb{S}$ to find the optimal client selection. Good initialization is crucial for the gradient-based optimization approaches [Glorot and Bengio, 2010], so we first select top-K client selections ranked by their comprehensive score p and use the encoder to embed these selections into a continuous representation which will later be used as starting points for subsequent optimization. Assuming that one starting point representation is $E_{\mathbf{s}}$, in order to obtain a selection that possesses a better comprehensive score, we optimize from $E_{\mathbf{s}}$ towards the gradient direction induced by the evaluator ω:

$$E_{\mathbf{s}}^{+} = E_{\mathbf{s}} + \eta \frac{\partial \omega(E_{\mathbf{s}})}{\partial E_{\mathbf{s}}}, \tag{9}$$

where $E_{\mathbf{s}}^{+}$ is the optimized selection representation, η is the step size. The comprehensive score of $E_{\mathbf{s}}^{+}$ is supposed to be better than $E_{\mathbf{s}}$ due to $\omega(E_{\mathbf{s}}^{+}) \geq \omega(E_{\mathbf{s}})$. For all the K starting points, we perform the above optimization several times to get the set of candidate selection representation $\{\tilde{E}_{\mathbf{s}_i}\}_1^K$. Finally, we choose the optimal selection representation $E_{\mathbf{s}^*}$ in $\{\tilde{E}_{\mathbf{s}_i}\}_1^K$ through the estimated candidates' comprehensive score, *i.e.*, $E_{\mathbf{s}^*} = \text{argmax}_{\tilde{E}_{\mathbf{s_i}}} \{\omega(\tilde{E}_{\mathbf{s_i}})\}_1^K$.

4.4 Optimal Client Selection Generation

To identify the optimal client selection s^*, we fed $E_{\mathbf{s}^*}$ into the well-trained decoder ψ, this process can be denoted by: $s^* = \psi(E_{\mathbf{s}^*})$. Specifically, we adopt the beam search [Freitag and Al-Onaizan, 2017] to generate the client selection, just like text generation in natural language processing [Sutskever *et al.*, 2014]. Instead of setting the length of the selection in advance, we use the decoder to iteratively generate device IDs until it encounters the stop token <EOS>, which makes our selection process more adaptive.

5 Experiments

5.1 Experimental Setup

Infrastructure. To evaluate the performance and effectiveness of FedGCS, we first build a simulator with the server/client architecture based on PyTorch [Paszke *et al.*, 2019], where distinct processes emulate the central server and participating devices. To emulate data heterogeneity, we allocate training data with different distributions across devices. To mimic system heterogeneity, we establish a FL system comprising six device types with diverse hardware configurations, including Google Pixel 6, OnePlus 10 pro, Redmi Note 10, Realme Q3s, NVIDIA Jetson Nano, and NVIDIA Jetson TX2. We utilize Monsoon Power Monitor [mon, 2023] to track latency and energy consumption during training. Furthermore, we integrate end-user interaction traces from LiveLab [Shepard *et al.*, 2011] to emulate concurrent applications that impact the training capability at runtime.

Baselines. FedGCS is compared with three types of client selection methods, including ***random-based*** (FedAvg [McMahan *et al.*, 2017] and FedProx [Li *et al.*, 2021]), ***heuristic-based*** (AFL [Goetz *et al.*, 2019], TiFL [Chai *et al.*, 2020] and Oort [Lai *et al.*, 2021]), and ***learning-based*** (Favor [Wang *et al.*, 2020] and FedMarl [Zhang *et al.*, 2022]).

Datasets and Models. We use typical models and datasets in Computer Vision (CV) and Natural Language Processing (NLP) domains for evaluation, including LeNet5 [LeCun *et al.*, 1998a] on MNIST [Lecun *et al.*, 1998b], ResNet18 [He *et al.*, 2016] on CIFAR10 [Krizhevsky *et al.*, 2009], VGG16 [Simonyan., 2014] on CINIC10 [Darlow *et al.*, 2018], ShuffleNet [Zhang *et al.*, 2018] on TinyImageNet [Deng *et al.*, 2009] for image classification, LSTM [Hochreiter and Schmidhuber, 1997] on Shakespeare [Shakespeare, 2017], GPT-2 [Radford *et al.*, 2019] on Wikitext [Merity *et al.*, 2016] for text generation. For four CV datasets, Dirichlet distribution $p_k \sim Dir_N(\beta)$ is utilized to simulate the Non-IID data distribution on different devices.

Hyperparameter Settings and Reproducibility. We run two heuristic-based methods—Oort and Favor, and one heuristic-based method—FedMarl 100 times each, totaling 300 runs, to collect "selection-score" pair data for subsequent model training. For data augmentation, each selection is randomly shuffled 25 times to model set order-independence. We adopt a single-layer LSTM as the Encoder and Decoder backbones, and two-layer feed-forward networks for the Evaluator. The hidden state sizes are 64 (Encoder), 64 (Decoder), and 200 (Evaluator), with a 32-dimensional embedding for each device ID token. For FedGCS training, we set batch size$= 1024$, learning rate$= 0.001$, $\alpha = 0.8$, and utilize the top-25 device selections as starting points for gradient optimization. In the FL setting, $J = 100$ devices are available, with $T = 10$ ($T = 20$ for FedMarl) selected per round over $r = 10$ rounds for GPT-2 ($r = 50$ for others). Clients perform local $ep = 5$ epoch training each round. The code is available at https://github.com/zhiyuan-ning/GenerativeFL.

5.2 Overall Performance

Model Performance. Table 1 shows the final test performance of all client selection methods. We can observe that FedGCS significantly outperforms other baselines across all domains and tasks. Specifically, for IID setting, FedGCS improves the test accuracy 5.56% over FedAvg and 2.20% over SOTA value on average. Further, FedGCS is particularly more effective in the more challenging Non-IID setting, outperforming the baselines by considerable margins: improving the test accuracy 20.35% over FedAvg and 3.16% over SOTA on average, and reducing 18.55 perplexity (PPL, lower values correspond to stronger text generation capability) over FedAvg and 6.54 over SOTA on the GPT-2 model. Overall, this experiment demonstrates the practical and robust ability of FedGCS to scale to complex workloads and application

Dataset &Model	CV Tasks								NLP Tasks	
	MNIST LeNet5 ↑		CIFAR10 Resnet18 ↑		CINIC10 VGG16 ↑		TinyImageNet ShuffleNet ↑		Shakespeare LSTM ↑	Wikitext GPT-2[1]↓
	IID	Non-IID[2]	IID	Non-IID[2]	IID	Non-IID[2]	IID	Non-IID[2]	Non-IID[2]	Non-IID[2]
FedAvg	98.84	36.70	84.78	42.02	64.78	37.40	34.62	23.82	39.18	31.13
FedProx	$99_{(35)}$[3]	67.67	85.11	52.40	64.51	39.59	35.38	24.72	40.68	30.07
AFL	98.61	89.60	83.33	51.81	65.86	41.03	36.77	25.90	42.67	27.42
TiFL	98.91	91.21	84.96	56.39	67.32	42.16	38.82	25.85	44.12	25.61
Oort	$99_{(49)}$[3]	92.47	85.69	53.51	67.07	43.57	40.57	26.62	45.42	19.12
Favor	98.87	85.48	85.98	51.34	65.47	39.26	37.62	25.34	43.11	28.45
FedMarl	98.82	90.13	86.37	56.73	66.74	44.16	39.48	26.31	44.72	21.84
FedGCS	**$99_{(28)}$**[3]	**95.83**	**88.42**	**61.62**	**69.09**	**46.27**	**43.35**	**29.51**	**47.65**	**12.58**

Table 1: Performance of different selection methods. **Bold** indicates the best one and underline indicates the runner-up.
[1] We use perplexity (PPL) to evaluate GPT-2, which reflects the text generation capability of model.
[2] Set $\beta = 0.01$ for MNIST, $\beta = 0.1$ for others to emulate Non-IID. Shakespeare and Wikitext are naturally Non-IID.
[3] Early exit when target accuracy (99%) is achieved. Numbers in parentheses indicate the exited round.

Schemes	Acc. (%)↑	Latency(min)↓	Energy(KJ)↓
FedGCS^{-c}	86.23	18.2	2.12
FedGCS^{-a}	85.37	24.82	2.53
FedGCS	**88.42**	**15.34**	**1.70**

Table 2: The effect of data collection (FedGCS^{-c}) and augmentation (FedGCS^{-a}) on test accuracy, average latency and energy cost per round when training the ResNet18 model on CIFAR10 (IID).

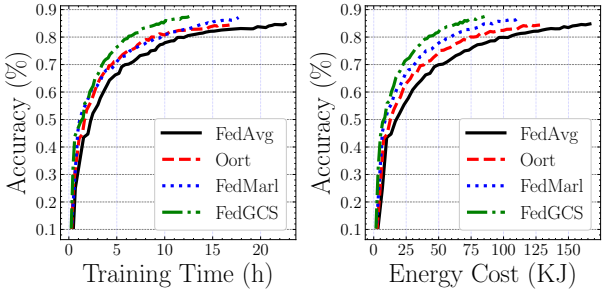


Figure 4: Efficiency comparison of FedGCS with three SOTA baselines to train ResNet18 on CIFAR10 (IID). Left: ToA. Right: EoA.

scenarios. This may be because FedGCS converts extensive discrete decision knowledge into a continuous representation space in a data-driven manner and identifies a superior client selection in that space based on gradient-based optimization.

Latency and Energy Consumption. Then, we evaluate the system efficiency of FedGCS from two perspectives: Time to Accuracy (ToA) and Energy to Accuracy (EoA). Figure 4 left shows FedGCS effectively speeds up the training process with faster convergence and achieves superior accuracy. In addition, FedGCS also achieves significant energy savings shown in Figure 4 right. This can be attributed to FedGCS thoughtfully integrating considerations of both latency and energy consumption during training. In contrast, Oort emphasizes solely training duration as the system effectiveness metric, and FedMarl takes into account the energy costs associated with communication, but it overlooks the intrinsic energy overheads of the training process itself, a critical oversight, especially for devices operating on battery power.

5.3 Framework Analysis

Impact of Data Collection and Augmentation. To explore the impact of the different modules of FedGCS, we develop two model variants: (1) FedGCS^{-c}, we randomly collect pair data without relying on classical client selection methods. (2) FedGCS^{-a}, we disable the data augmentation process of FedGCS. Table 2 shows the comparison results, we can find that the performance of FedGCS is much better than FedGCS^{-c}. This suggests that the quality of the collected data is critical to constructing a continuous representation space, and that a better space will in turn help to identify better client selection. Moreover, we observe that FedGCS is superior to FedGCS^{-a}. The key driver is that data augmentation can enhance data diversity and model the order-independent properties of client selection, thereby enabling more robust and effective learning for FedGCS. Consequently, these results confirm that data collection and augmentation are essential for sustaining the performance of FedGCS.

Study of Generalization Ability of GCS. Figure 5 outlines when training the ResNet18 model on CIFAR10 (IID), the generalization ability of FedGCS in combination with different client selection methods, and the effectiveness of using multiple selection methods for data collection. The adoption of a single selection method as data collector can significantly improve results over the corresponding selection method, highlighting the merits of continuous space paradigms in enhancing generalization ability. Furthermore, the use of diverse, multiple selection methods as data collectors (*i.e.*, the FedGCS in Figure 5 which uses Oort, Favor and FedMarl as data collectors simultaneously) outperforms other single collector strategies, emphasizing the value of diversity in data collectors for comprehensive space construction.

Study of the Selection Strategy Made by FedGCS. We analyze the selection strategy of FedGCS by comparing its selected device ratio with the best fixed client selection method Oort and the dynamic selection approach FedMarl. Figure 6 shows the results, where we can observe that the ratio of the

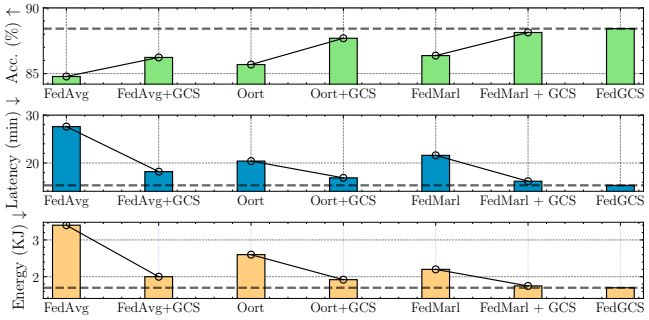


Figure 5: Generalization ability of FedGCS.

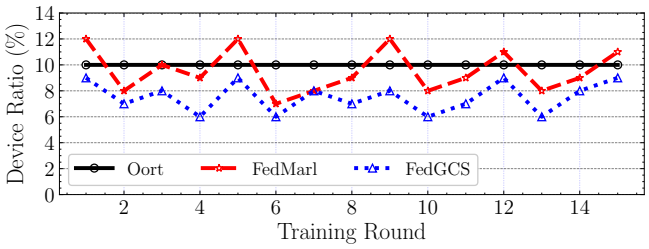
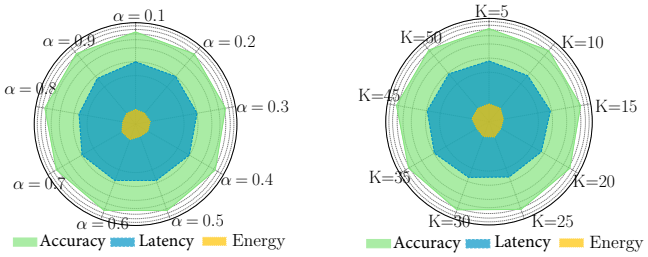


Figure 6: Comparison of the selected devices ratio of FedGCS, FedMarl and Oort.



(a) Trade Off (b) Top-K

Figure 7: Hyperparameter sensitivity of FedGCS.

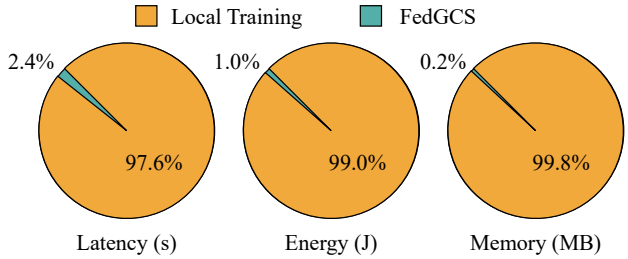


Figure 8: Overhead analysis of the FedGCS framework shows low memory usage and operational effectiveness.

client selection made by FedGCS is dynamically changing each training round, indicating that decision making via generative is more adaptive when compared to the fixed client selection methods. In addition, the ratio of FedGCS is the smallest, suggesting that FedGCS can select the most reasonable number of devices even when compared to the dynamic selection approaches. Overall, such results demonstrate that FedGCS can achieve the best performance with the smallest device selection ratio, indicating that the selection strategy of FedGCS is superior compared to other methods.

Study of the Hyperparameter Sensitivity of FedGCS. There are two major hyperparameters, training trade-off α and the top-K records as the starting points of gradient-based optimization. A higher α will make the model more concentrated on the loss $\mathcal{L}_{seq2seq}$, and a higher K makes the model search more milder. We set α from 0.1 to 0.9, and K from 5 to 50, then train the FedGCS on ResNet18 over CIFAR10 (IID). The model performance is reported in Figure 7. Overall, we observe that model performance and system efficiency vary slightly for different K, but for better results K needs to be greater than 20. Further, setting α as 0.8 will slightly bring a higher model performance. These findings provide insight into how α and K impact the model performance and system efficiency and how to choose the optimal hyperparameters.

5.4 Overhead Analysis

As shown in Figure 8, we delineate the overhead analysis of the FedGCS framework and employ the Monsoon Power Monitor for the empirical measurement of the local training runtime overhead, specifically focusing on the ResNet18-CIFAR10 model augmented with 500 samples on an NVIDIA Jetson Nano platform. The empirical data reveals an average local training duration per round of 28.24 minutes, accompanied by an average energy expenditure of 370.5 Joules. Conversely, the deployment of the FedGCS framework on a server manifests markedly different metrics: a mere 0.68 minutes of processing time and an average energy consumption of 3.9 Joules. This stark contrast underscores the negligible runtime overhead of the FedGCS when operating in a server environment. Further, the memory requirements of FedGCS are remarkably low, with a total memory footprint under 0.1 MB. This is negligible when compared to the substantial memory size of the ResNet18 model, which is approximately 42.70MB, and the expansive 40GB DRAM capacity available on cloud servers equipped with NVIDIA A40 GPUs. Consequently, we assert that the runtime overhead of the FedGCS framework is not a significant factor in affecting the overall efficiency and progress of the training process.

6 Conclusion

FedGCS marks a significant advancement in FL by introducing a generative framework for client selection, effectively addressing statistical and system heterogeneity and high energy consumption challenges. This framework transcends traditional methods by encapsulating decision-making processes within a continuous representation space, enabling precise and efficient identification of optimal client subsets. Our novel approach, which includes data collection, model training, gradient-based optimization, and generation of client selection, not only outperforms existing methods in terms of model performance, latency, and energy efficiency but also simplifies the process, reducing reliance on domain expertise and manual intervention. The empirical validation of FedGCS underscores its superiority and potential to revolutionize client selection strategies in FL.

Acknowledgments

This research is supported by the the Strategic Priority Research Program of the Chinese Academy of Sciences XDB38030300, the Natural Science Foundation of China under Grant No. 61836013, the Postdoctoral Fellowship Program of CPSF (No.GZC20232736), the China Postdoctoral Science Foundation Funded Project (No.2023M743565), the Science and Technology Development Fund (FDCT), Macau SAR (file no. 0123/2023/RIA2, 001/2024/SKL), and the Start-up Research Grant of University of Macau (File no. SRG2021-00017-IOTSC).

Contribution Statement

Zhiyuan Ning and Chunlin Tian contributed equally.

References

[Bonawitz *et al.*, 2019] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Chai *et al.*, 2020] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system. In *Proceedings of the 29th international symposium on high-performance parallel and distributed computing*, pages 125–136, 2020.

[Cho *et al.*, 2020] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

[Darlow *et al.*, 2018] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Dulac-Arnold *et al.*, 2021] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[Freitag and Al-Onaizan, 2017] Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In Thang Luong, Alexandra Birch, Graham Neubig, and Andrew Finch, editors, *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver, August 2017. Association for Computational Linguistics.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[Goetz *et al.*, 2019] Jack Goetz, Kshitiz Malik, Duc Bui, Seungwhan Moon, Honglei Liu, and Anuj Kumar. Active federated learning. *arXiv preprint arXiv:1909.12641*, 2019.

[Gruver *et al.*, 2023] Nate Gruver, Marc Anton Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hsieh *et al.*, 2020] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[Kim and Wu, 2021] Young Geun Kim and Carole-Jean Wu. Autofl: Enabling heterogeneity-aware energy efficient federated learning. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 183–198, 2021.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

[Lai *et al.*, 2021] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. Oort: Efficient federated learning via guided participant selection. In *OSDI*, pages 19–35, 2021.

[LeCun *et al.*, 1998a] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lecun *et al.*, 1998b] Yann Lecun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

[Li *et al.*, 2019] Li Li, Haoyi Xiong, Zhishan Guo, Jun Wang, and Cheng-Zhong Xu. Smartpc: Hierarchical pace control in real-time federated learning system. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pages 406–418. IEEE, 2019.

[Li *et al.*, 2021] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[Luong *et al.*, 2015] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Merity *et al.*, 2016] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[mon, 2023] Monsoon high voltage power monitor. https://www.msoon.com/online-store/High-Voltage-Power-Monitor-p90002590, 2023.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[Ramesh *et al.*, 2022] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[Shakespeare, 2017] Shakespeare. Shakespeare dataset. https://www.gutenberg.org/files/100/, 2017.

[Shepard *et al.*, 2011] Clayton Shepard, Ahmad Rahmati, Chad Tossell, Lin Zhong, and Phillip Kortum. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2011.

[Simonyan., 2014] Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Sunehag *et al.*, 2017] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Touvron *et al.*, 2023] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[Wang *et al.*, 2020] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020.

[Wang *et al.*, 2024] Dongjie Wang, Meng Xiao, Min Wu, Yuanchun Zhou, Yanjie Fu, et al. Reinforcement-enhanced autoregressive feature transformation: Gradient-steered search in continuous space for postfix expressions. *Advances in Neural Information Processing Systems*, 36, 2024.

[Xiao *et al.*, 2023] Meng Xiao, Dongjie Wang, Min Wu, Pengfei Wang, Yuanchun Zhou, and Yanjie Fu. Beyond discrete selection: Continuous embedding space optimization for generative feature selection. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 688–697. IEEE, 2023.

[Zhan *et al.*, 2020] Yufeng Zhan, Peng Li, and Song Guo. Experience-driven computational resource allocation of federated learning by deep reinforcement learning. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 234–243. IEEE, 2020.

[Zhang *et al.*, 2018] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[Zhang *et al.*, 2022] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9091–9099, 2022.