

Article

Budget-Aware Closed-Loop Incentive Allocation for Federated Learning with DDPG

Yang Cao ^{1,2}, Huimin Cai ^{2,3}, Haotian Zhu ⁴, Sen Zhang ⁴ and Jun Hu ^{4,*} ¹ School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China² Big Data Research Institute Co., Ltd., China Electronics Technology Group Corporation, Guiyang 550022, China³ Big Data Application Technology to the Improvement of Governance Capacity, National Engineering Research Center, Guiyang 550022, China⁴ College of Computer Science, Beijing University of Technology, Beijing 100124, China* Correspondence: algorist@bjut.edu.cn

Abstract

With the growing demand for trustworthy multi-party data sharing, federated learning has demonstrated broad potential in cross-entity collaborative modeling. However, it still faces challenges such as insufficient participant engagement, inaccurate contribution assessment, and the lack of dynamic profit-sharing mechanisms. Traditional incentive schemes, which typically rely on game-theoretic models or static rules, struggle to accommodate dynamic client participation and heterogeneous data distributions, thereby degrading the convergence efficiency and generalization performance of the global model. To address these issues, we propose a budget-aware closed-loop incentive allocation for federated learning with deep deterministic policy gradient (DDPG). The proposed approach constructs a DDPG-driven closed-loop framework in which the server manages system states, incentive decisions, and model aggregation, while clients autonomously adjust their data contribution levels. By formulating incentive allocation as a sequential decision-making problem, the mechanism jointly optimizes policy and value functions. A permutation method is introduced to ensure invariance to client ordering, and an Ornstein–Uhlenbeck process is employed to enhance exploration, thereby improving the adaptiveness and overall effectiveness of incentive allocation. Experimental results show that the proposed method significantly increases cumulative rewards and improves client data-sharing rates in high-dimensional dynamic environments. Compared with traditional fixed incentive schemes, the mechanism demonstrates clear advantages in adaptiveness, incentive effectiveness, and model performance.

Keywords: deep reinforcement learning; deep deterministic policy gradient; federated learning; incentive mechanism; budget constraint; adaptive allocation



Academic Editor: Morgado Dias

Received: 12 February 2026

Revised: 20 March 2026

Accepted: 22 March 2026

Published: 2 April 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

In recent years, with the increasing demand for multi-party collaborative modeling, the volume of data distributed across different organizations or user devices has grown continuously. These data contain rich behavioral patterns and business information, providing an important foundation for intelligent analysis and decision support. However, due to privacy protection, data security, and compliance requirements, traditional centralized machine learning cannot directly aggregate such data for unified modeling. Federated

learning (FL) has thus been proposed as a distributed collaborative training paradigm [1]. FL enables multiple participants to jointly train a global model without sharing raw data, requiring only the upload of local model gradients or parameter updates to a central server for aggregation, thereby preserving privacy without exposing the underlying data.

Although FL has gradually matured in terms of theory and system frameworks, its practical deployment in open and heterogeneous environments still faces multiple challenges [2]. First, participants differ significantly in computational resources, data volume, data distribution, and communication capabilities, leading to unstable training efficiency. Second, federated training involves multiple communication rounds and sustained computation, resulting in high participation costs. Some rational participants may reduce their level of engagement or even engage in free-riding behaviors, such as submitting low-quality or falsified model updates to obtain rewards, which can adversely affect the stable convergence and final performance of the global model [3]. Moreover, in a multi-party collaborative context, accurately and fairly evaluating each participant's contribution to the global model's performance remains a critical challenge for the design of incentive mechanisms.

Existing incentive mechanisms in FL predominantly rely on traditional game-theoretic or auction-based frameworks, where client contributions are evaluated and rewards are allocated based on predefined rules to encourage rational participation. For example, incentive schemes based on marginal contribution or the Shapley value can theoretically achieve high fairness and rationality [4]. However, these approaches often rely on strong assumptions, such as prior knowledge of client types, static environments, and perfect information. These assumptions are challenging to satisfy in FL scenarios that feature dynamic participation, non-independent and identically distributed (Non-IID) data, and communication delays [5,6].

Additionally, contribution evaluation is usually done after the fact using marginal contribution or the Shapley value, which requires a lot of computation and can't be done in real time. This results in delayed incentive responses and limits their ability to sustain high-quality participation [7,8]. Although some research has looked into reinforcement learning for designing incentives dynamically, most of these methods are centralized. As a result, they do not fully consider the decentralized and privacy-sensitive nature of federated systems, which can lead to risks like model leakage and strategy uniformity [9].

Deep reinforcement learning (DRL), with its ability to autonomously learn optimal decision policies in high-dimensional state spaces, offers a promising approach for designing incentive mechanisms in complex and dynamic environments [10]. However, applying DRL directly to FL incentive mechanisms still faces two main challenges. First, global reward signals are often sparse and delayed because improvements in global model performance usually appear only after multiple communication rounds, making it difficult to provide timely and effective feedback for policy updates. Second, participant strategies evolve and interact dynamically, as clients may adjust their contribution behaviors or even drop out during the process. Therefore, learning an efficient and stable incentive policy under privacy constraints remains a difficult task [11].

For example, data distributions vary substantially across different devices and time periods, and some participants may even maliciously replicate local samples during model training, resulting in distribution shifts over time [12]. Therefore, this work addresses key challenges in heterogeneous federated environments, including insufficient incentives, ambiguous contribution evaluation, and imbalanced reward allocation. We propose an adaptive incentive mechanism based on DRL. The mechanism introduces systematic innovations across system architecture design, DRL-based incentive modeling, and dynamic incentive optimization. The main contributions of this paper are summarized below.

1. We propose a DRL-driven adaptive incentive framework for FL, enabling global coordination at the server and autonomous decision-making at clients. Specifically, we design a federated DRL decision architecture that integrates model aggregation, state management, and incentive distribution into a unified DRL loop. By modeling the state space, the system can continuously perceive client characteristics such as data quality, computational capability, and participation frequency. The server dynamically adjusts incentive strategies through a policy network, achieving jointly optimized regulation of client participation and training efficiency. This framework provides intelligent decision support for multi-party collaboration in heterogeneous federated environments.
2. We design an actor–critic incentive allocation model based on the deep deterministic policy gradient (DDPG), establishing a joint optimization mechanism for policy generation and value evaluation. To mitigate instability in traditional incentive mechanisms when operating in continuous decision spaces, we formulate incentive allocation as a continuous control problem. The actor network generates optimal incentive strategies, while the critic network evaluates the corresponding value functions, enabling coordinated iterative optimization of policy and value. This design significantly enhances convergence stability in high-dimensional and dynamic environments and ensures that incentive allocation more accurately reflects clients' actual contribution levels.
3. We construct an adaptive optimization loop consisting of state awareness, policy evaluation, and benefit feedback and develop an end-to-end dynamic incentive update process that forms a positive feedback cycle between the server and clients. Clients adjust their data contribution ratios and local training strategies based on the incentive signals, while the server re-evaluates reward allocation according to the updated global state. Through multi-round interactions, the mechanism gradually converges to a dynamic equilibrium of incentive policies, effectively suppressing low-effort participation and fostering long-term collaboration from high-quality clients. Experimental results demonstrate that, in high-dimensional dynamic environments, the proposed mechanism can enhance participants' data sharing rate, improve the generalization capability of the model, and validate the adaptive incentive allocation capability of the incentive distribution model.

The remainder of this paper is organized as follows. Section 2 reviews existing studies on federated learning incentive mechanisms and related DRL-based approaches. Section 3 presents the proposed incentive mechanism model, including the overall architecture, module design, state management, incentive allocation strategy, and data contribution strategy. Section 4 reports the experimental setup and evaluation results. Finally, Section 5 concludes this work and outlines future research directions.

2. Related Work

FL has emerged as a promising paradigm for distributed model training, enabling clients to share gradients instead of raw data and thus preserving data privacy [13,14]. However, in practical applications, clients may exhibit low participation due to resource consumption or personal incentives. Designing fair and effective incentive mechanisms has thus become a key research focus. Sun et al. [15] proposed an incentive mechanism based on mean-field game theory, modeling the aggregation process as a mean-field game across clients and incorporating client reputation to allocate rewards, thereby enhancing global model training performance; however, its treatment of irrational or dynamic behaviors is limited. Chen et al. [16] employed evolutionary game analysis to examine conflicts of interest among data owners, model requesters, and cloud service platforms, and used rewards, penalties, and collusion costs to mitigate free-riding behavior, but did not fully

consider online dynamic participation scenarios. Han et al. [17] combined blockchain with Stackelberg game-based dynamic incentives, evaluating participant reliability via reputation, thereby promoting high-quality participation in decentralized transaction environments; nonetheless, the mechanism shows limited adaptability in highly dynamic client environments. Zhao et al. [18] introduced stochastic client selection and hierarchical game strategies, guiding clients' contributions of data and computational resources through bidding, and provided semi-closed-form equilibrium solutions to achieve a balance between economic efficiency and energy efficiency. Hou et al. [19] proposed an improved DDPG method that incorporates Prioritized Experience Replay (PER). The method evaluates the learning value of experience samples using the temporal-difference (TD) error and performs sampling according to their priorities. Meanwhile, importance-sampling weights are introduced to correct the bias caused by non-uniform sampling, allowing experiences with higher learning value to be utilized more frequently during training.

In cross-device or resource-constrained FL scenarios, reinforcement learning and auction mechanisms can enhance the adaptability of incentive strategies and improve system performance [20]. Li et al. [21] proposed a reinforcement auction mechanism that integrates power allocation, channel assignment, user selection, and computational frequency, dynamically adjusting users' payments and selection strategies through hybrid-action reinforcement learning to ensure individual rationality and truthfulness. Yuan et al. [22] employed multi-agent reinforcement learning to drive adaptive incentive mechanisms, achieving long-term reward maximization in industrial IoT FL environments across isolated domains; however, their approach assumes fully observable dynamic environments, limiting practical deployment. Ma et al. [23] combined contract theory with DRL to propose the CDRL algorithm, adaptively allocating incentives under scenarios of weak and strong incomplete information, thereby ensuring security and participant engagement, though convergence speed under dynamically heterogeneous task distributions still requires improvement.

Pricing and contribution evaluation techniques have been widely adopted in FL incentive mechanisms to ensure fairness and optimize resource allocation [4]. Shi et al. [24] proposed the WTDP-Shapley method, which employs weighted truncation and dynamic programming to achieve efficient contribution assessment, supporting end-to-end incentive mechanisms; however, its high computational complexity poses performance bottlenecks in large-scale distributed scenarios. Yang et al. [25] introduced a pricing-based optimization strategy to address convergence bias arising from temporally varying client availability, using particle swarm optimization to determine optimal pricing schemes and effectively accelerate convergence, though efficiency challenges remain in high-dimensional tasks or extreme edge-device environments.

In summary, while existing FL incentive mechanisms demonstrate certain advantages in specific scenarios, they still exhibit notable limitations. Game-theoretic approaches are effective for multi-party strategy modeling and hierarchical incentive optimization, but their adaptability to dynamic participation and heterogeneous resources remains limited. Reinforcement learning and auction-based mechanisms enhance strategy adaptability and long-term reward maximization in cross-device and dynamic environments, yet they incur high training costs and slower convergence in large-scale or complex-task deployments. Contribution evaluation and pricing approaches ensure fairness and resource optimization, and when combined with techniques such as differential privacy, homomorphic encryption, or multi-stage evaluation mechanisms, they can improve system robustness [26]; nevertheless, efficiency remains constrained in high-dimensional complex tasks and large-scale distributed settings. Therefore, future research should focus on further enhancing the performance of incentive mechanisms in terms of dynamic adaptability, resource heterogeneity, and large-scale deployment.

3. Incentive Mechanism Model

3.1. Model Architecture

This work proposes a DRL-based adaptive incentive mechanism for FL. By constructing a virtual DRL environment, the model simulates the stochastic allocation of data samples among multiple participants, thereby enabling dynamic variations in data distribution. On the server side, modules for global model aggregation, model evaluation, state management, and incentive allocation are designed. On the client side, each participant can autonomously decide the proportion of data to contribute. The adaptive incentive allocation model implements a DRL-based strategy for learning incentive distribution, which guides the server in determining the allocation ratios, individual participant incentive weights, and global model aggregation weights during each iteration. The overall architecture of the incentive mechanism model is illustrated in Figure 1.

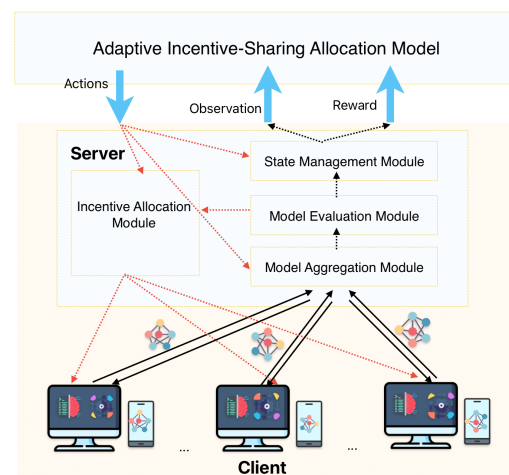


Figure 1. Overall architecture diagram of the incentive model.

3.2. Model Design

3.2.1. Global Model Aggregation Module

Assume that each client k uploads its model parameters θ_k^t to the server after the t -th training iteration, and the total number of clients is K . The server aggregates the global model parameters G^{t+1} using aggregation weights w_k^t , where β_k^t represents the incentive share predicted by the incentive allocation module for client k in iteration t , as defined in Equation (1). This formulation enables the server to perform global model aggregation in a manner that incorporates client-specific incentive allocations.

$$\begin{cases} \theta_G^{t+1} = \sum_{k=1}^K w_k^t \cdot \theta_k^t \\ w_k^t = \frac{\beta_k^t}{\sum_{k=1}^K \beta_k^t} \end{cases} \quad (1)$$

3.2.2. Model Evaluation Module

In each iteration t , the server independently evaluates the performance of the model uploaded by client k on the test dataset D_{test} , obtaining the target loss loss_k^t and the macro-averaged accuracy $\mathcal{A}_k^t$. The macro-averaged accuracy is defined as the arithmetic mean of the accuracy across all classes in the test dataset. This metric allows the module to effectively leverage the data distribution characteristics of each participating client.

In addition, the server evaluates the contribution of client k 's uploaded model using the Leave-One-Out (LOO) method, denoted as $\psi_k(K, v_{\text{loss}})$ and $\psi_k(K, v_{\text{acc}})$, as defined in Equation (2).

$$\begin{cases} \psi_k(K, v_{\text{loss}}) = v_{\text{loss}}(K) - v_{\text{loss}}(K \setminus \{k\}) \\ \psi_k(K, \mathcal{A}) = \mathcal{A}(K) - \mathcal{A}(K \setminus \{k\}) \end{cases}, \quad (2)$$

where $v_{\text{loss}}(K)$ and $\mathcal{A}(K)$ denote the target loss and macro-averaged accuracy of the aggregated model contributed by all K clients, respectively. The terms $v_{\text{loss}}(K \setminus \{k\})$ and $\mathcal{A}(K \setminus \{k\})$ represent the corresponding metrics computed without client k . Consequently, $\psi_k(K, v_{\text{loss}})$ and $\psi_k(K, \mathcal{A})$ quantify the marginal contributions of client k to the aggregated model in terms of the target loss and macro-averaged accuracy, respectively.

In terms of practical deployment, the server maintains a small, representative validation set (D_{test}) to evaluate global performance, which is a common setting in incentivized FL to ensure evaluation consistency. While the LOO method involves K additional forward passes per round, the computational overhead is relatively low for moderate K . In our experiments ($K = 4$), the evaluation process took less than 5% of the total per-round time. For larger-scale or stricter privacy settings, the server can observe only the model updates, and LOO can be optimized using gradient-based contribution approximation or by performing evaluations on a subset of rounds.

3.2.3. State Management Module

In the DRL environment proposed in this paper, the observation in each iteration is represented as a one-dimensional state vector of dimension $22K$, where K denotes the total number of participating users. For each user, 22 state variables are constructed. These 22-dimensional state variables are carefully designed to comprehensively capture each client's learning performance, behavioral dynamics, and system budget status. Specifically, these variables jointly reflect the absolute and relative improvements in loss and accuracy, the contribution to the global model performance, as well as resource-normalized metrics such as data efficiency and cost efficiency. In this way, the deep reinforcement learning agent is able to perceive both client-specific information and global constraints, enabling precise policy learning. The complete definitions of all state variables are provided in Table 1.

This structured state representation enables the DRL agent to accurately perceive both user-specific dynamics and system-level constraints, thereby supporting more precise policy optimization in incentive allocation.

In the DRL virtual environment, in each iteration t , the server needs to simultaneously consider multiple objectives: namely, increasing the average contribution proportion of all participants' data in the current iteration, thereby encouraging participants to contribute more data for model training and improving the macro-average accuracy of the aggregated model on the test set $\mathcal{A}^t$, thus enhancing the value of the aggregated model for downstream utilization. Therefore, this study defines two separate reward functions to evaluate the performance of the incentive allocation model. Since the primary goal of the incentive allocation model is to promote participants' willingness to contribute data, the first reward function considers only the average data contribution proportion of participants in each iteration t , as expressed in Equation (3).

$$\mathcal{R}_1^t = 10 \cdot \sum_{k=1}^K \frac{\Delta_k^t}{K}, \quad (3)$$

where $\mathcal{R}_1^t$ denotes the reward obtained from the first reward function in iteration t , K represents the total number of participants, and Δ_k^t indicates the proportion of data contributed by participant k in the t -th iteration.

Table 1. Status variable feature definitions.

Symbol	Description
L_k^t	Objective loss of client k in iteration t
$L_{\min,k}$	Minimum loss of client k across iterations
d_k^t	Training data size of client k in iteration t
e_k^t	Expense of client k in iteration t
$\mathcal{A}_k^t$	Macro-averaged accuracy of client k in iteration t
$\mathcal{A}_{\max,k}$	Maximum macro-averaged accuracy of client k across iterations
br^t	Server's available budget in iteration t
$\psi_k(K, v_{\text{loss}})$	Contribution of client k to global loss based on loss values
$\psi_k(K, \mathcal{A})$	Contribution of client k to global macro-accuracy based on accuracy values
$L_{\min,k} - L_k^t$	Improvement in objective loss of client k in iteration t
$(L_{\min,k} - L_k^t) / L_{\min,k}$	Relative improvement in objective loss of client k in iteration t
$(L_{\min,k} - L_k^t) / d_k^t$	Improvement in objective loss per unit of data of client k
$(L_{\min,k} - L_k^t) / e_k^t$	Improvement in objective loss per unit of expense of client k
$\mathcal{A}_k^t - \mathcal{A}_{\max,k}$	Improvement in macro-accuracy of client k in iteration t
$(\mathcal{A}_k^t - \mathcal{A}_{\max,k}) / d_k^t$	Improvement in macro-accuracy per unit of data of client k
$(\mathcal{A}_k^t - \mathcal{A}_{\max,k}) / e_k^t$	Improvement in macro-accuracy per unit of expense of client k
$\psi_k(K, v_{\text{loss}}) / L_k^t$	Relative contribution of client k to global loss
$\psi_k(K, v_{\text{loss}}) / d_k^t$	Contribution of client k to global loss per unit of data
$\psi_k(K, v_{\text{loss}}) / e_k^t$	Contribution of client k to global loss per unit of expense
$\psi_k(K, \mathcal{A}) / \mathcal{A}_k^t$	Relative contribution of client k to global macro-accuracy
$\psi_k(K, \mathcal{A}) / d_k^t$	Contribution of client k to macro-accuracy per unit of data
$\psi_k(K, \mathcal{A}) / e_k^t$	Contribution of client k to macro-accuracy per unit of expense
$(br^t - e_k^t) / br^t$	Relative change in server budget in iteration t
d_k^t / d_k	Data contribution rate of client k per iteration
d_k^t / d^t	Proportion of total training data contributed by client k

The second reward function simultaneously considers the average data contribution rate of clients and the macro-averaged accuracy of the aggregated model in a weighted manner. It is formulated to encourage both higher data contribution rates and improved global model performance, as shown in Equation (4).

$$\mathcal{R}_2^t = \alpha \cdot \frac{1}{K} \sum_{k=1}^K \Delta_k^t + \beta \cdot \frac{1}{C} \sum_{c=1}^C \mathcal{A}_c^t \quad (4)$$

where $\mathcal{R}_2^t$ denotes the reward obtained from the second reward function in iteration t , Δ_k^t is the data contribution proportion of client k , $\mathcal{A}_c^t$ represents the accuracy of the aggregated model on class c , K is the total number of clients, C is the number of classes, and α and β are weighting coefficients. The weights α and β are both set to 5 to balance the two optimization objectives, namely the data contribution rate and the global model accuracy. Experimental tuning indicates that this combination effectively guides the policy to converge toward a direction with higher overall rewards.

In the DRL virtual environment, an interaction round is terminated when either the number of iterations reaches a predefined maximum, or the remaining server bankroll br^t is less than the fixed cost per iteration, $K \cdot \text{cost}_{\text{round}}$. Here, $\text{cost}_{\text{round}}$ includes the expenses associated with model training, model transmission, and model aggregation.

In each iteration t , the server distributes incentive payments to all clients from the remaining bankroll, denoted as br^t . Let γ^t represent the proportion of the remaining bankroll allocated for incentives, which is predicted by the incentive allocation mechanism. Additionally, a fixed cost is incurred in each iteration, including expenses for model aggregation

and transmission, assumed to be proportional to the number of clients K . The income of the bankroll primarily derives from improvements in the aggregated model's performance. Let s denote the conversion rate from performance improvement to monetary gain.

After evaluating the aggregated model on the test set D_{test} , the macro-averaged accuracy is $\mathcal{A}^t$, and $\mathcal{A}_{\text{max}}$ represents the maximum macro-averaged accuracy observed across all previous iterations. The bankroll is updated as follows:

$$br^{t+1} = br^t \cdot (1 - \gamma^t) - K \cdot cost_{\text{round}}^t - \sum_{k=1}^K e_k^t + s \cdot \max(0, \mathcal{A}^t - \mathcal{A}_{\text{max}}), \quad (5)$$

where e_k^t denotes the allocated expenditure of client k in iteration t . The design of the bankroll update in Equation (5) and the action γ^t is grounded in the principle of dynamic resource optimization. Theoretically, under a stylized budget-constrained objective, the optimal allocation factor γ^t is expected to follow a decreasing trend as the global model converges, ensuring high-quality participation in early stages while maintaining sustainability. In our framework, this analytical derivation serves as a design principle and a consistency check for the DDPG policy. Rather than manually fixing a decreasing schedule, we task the DDPG agent with learning this theoretically grounded optimal trend through interaction with the dynamic environment.

The incentive payment allocated to client k , denoted as $bonus_k^t$, is calculated by

$$bonus_k^t = \frac{br^t \cdot \gamma^t \cdot \beta_k^t}{\sum_{k=1}^K \beta_k^t}, \quad (6)$$

where β_k^t is the share of the incentive assigned to client k , as predicted by the incentive allocation mechanism in iteration t .

The data contribution strategy is introduced to delineate how clients adjust their data-provision behavior in response to allocated incentives. For each client k in iteration t , the data contribution strategy is defined as follows. Let $bonus_k^t$ denote the incentive share allocated to client k in iteration t . The relative change of this share is computed and used as the input to a sigmoid function. The output of the sigmoid function represents client k 's response to the allocated share in the current round, which in turn influences the amount of data contributed in the subsequent iteration.

$$\Delta_k^t = \frac{1}{1 + \exp \left(- \frac{bonus_k^t - bonus_k^{t-1}}{bonus_k^{t-1} + \epsilon} \right)}, \quad (7)$$

where Δ_k^t represents the response of client k in iteration t in terms of data contribution proportion, $bonus_k^t$ and $bonus_k^{t-1}$ denote the incentive allocated to client k in iterations t and $t - 1$, respectively, and ϵ is a small constant added to avoid division by zero. This formulation employs a sigmoid function to map the relative change in the incentives received by a client to its data contribution rate, aiming to model a boundedly rational response behavior. The function smoothly adjusts the client's contribution level within the interval $(0, 1)$ based on variations in incentives, which aligns with the sensitivity of clients to incentive adjustments and the upper bound of their contributions in practical scenarios. We use the sigmoid response not as a claim of exact behavioral realism, but as a bounded, monotonic, smooth, and saturating first-order approximation that is well suited to stable DRL-based closed-loop simulation.

3.3. Incentive Allocation Module

Within the deep reinforcement learning virtual environment E_v , the incentive allocation model operates by making sequential decisions over discrete time steps with the objective of maximizing the accumulated reward. This process is formulated as a Markov decision process (MDP), characterized by a state space, $\mathcal{S}$, and an action, $\mathcal{A}$. In each time step, the model observes the current state s_t and determines the corresponding action a_t .

In the DRL virtual environment, this module predicts an action vector of $1 + K$ dimensions in each iteration t , formalized as $[\gamma^t, \beta_1^t, \dots, \beta_K^t]$, where γ^t denotes the proportion of the remaining bankroll br^t allocated for total incentive payments to clients, and β_k^t represents the share predicted by the incentive allocation mechanism for client k in iteration t .

The incentive allocation model interacts with the virtual environment, generating the interaction sequence $s_1, a_1, s_2, a_2, \dots, s_t, a_t$. Upon executing action a_t , the reward obtained from the virtual environment's reward function is r_t , where the reward function is given by $r(s_t, a_t) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The decision-making behavior of the incentive allocation model is defined by a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. In this work, we employ the deep deterministic policy gradient (DDPG) algorithm, which models this policy as a deterministic mapping. Specifically, we parameterize the policy with an actor network, $h(s_t | \theta_h) = a_t$, which approximates π .

$$R^t = \sum_{i=t}^T \lambda^{(i-t)} r(s_i, a_i), \quad (8)$$

where the discounting factor $\lambda \in [0, 1]$. The discounted future reward is a random variable, which depends on the selected actions and the policy π . The objective of reinforcement learning is to learn a policy that maximizes the expected return. This expected return is denoted by $J(\pi)$, as formulated in the following equation:

$$J(\pi) = \mathbf{E}_{r_t, s_t \sim E_v, a_t \sim \pi} [\mathcal{R}_1^t]. \quad (9)$$

The state–action value function $Q(s_t, a_t)$ describes the expected return obtained by taking action a_t in state s_t and subsequently following policy π . Its definition is given by the following equation:

$$Q(s_t, a_t) = \mathbf{E}_{r_{t \geq t}, s_{t \geq t} \sim E_v, a_{t \geq t} \sim \pi} [\mathcal{R}_1^t | s_t, a_t]. \quad (10)$$

Leveraging the deep deterministic policy gradient (DDPG) algorithm, the incentive allocation model establishes a deterministic mapping $h(s_t | \theta_h) = a_t$ via the actor network, where θ_h denotes the model parameters. That is, $h(\cdot)$ takes the observation variable s_t as input and outputs the action a_t . Meanwhile, a predictive model $q(s_t, a_t | \theta_q)$ corresponding to the state–action value $Q(s_t, a_t)$ is constructed as the critic network, where θ_q represents the model parameters. According to the Bellman equation, we can derive

$$\begin{aligned} Q(s_t, a_t) &= \mathbf{E}_{r_t, s_{t+1} \sim E_v} [r(s_t, a_t) + \lambda \mathbf{E}_{a_{t+1} \sim \pi} [Q(s_{t+1}, a_{t+1})]] \\ &= \mathbf{E}_{r_t, s_{t+1} \sim E_v} [r(s_t, a_t) + \lambda Q(s_{t+1}, a_{t+1})] \\ &= \mathbf{E}_{r_t, s_{t+1} \sim E_v} [r(s_t, a_t) + \lambda Q(s_{t+1}, h(s_{t+1}))] \end{aligned} \quad (11)$$

Therefore, the loss function to be optimized is denoted as L :

$$\begin{cases} L(\theta_q) = \mathbf{E}_{s_t, r_t \sim E_v, a_t \sim \pi} [(q(s_t, a_t | \theta_q) - y_t)^2] \\ y_t = r(s_t, a_t) + \lambda \cdot q(s_{t+1}, h(s_{t+1}) | \theta_q) \end{cases} \quad (12)$$

From the Bellman Equation (11), the estimated value y_t of the state–action value $Q(s_t, a_t)$ is derived. The minimization of the mean squared error of the loss function L is achieved via stochastic gradient descent, and the model parameters θ_q of the critic network are iteratively updated.

The actor network $h(s_t | \theta_h)$ deterministically maps state variables to specific actions, with the goal of finding a policy that maximizes the expected return $J(\pi)$. To learn the parameters θ_h , the parameter update is accomplished by applying the chain rule to the expected return $J(\pi)$, as formulated in the following equation:

$$\begin{aligned}\nabla_{\theta_h} J &\approx \mathbf{E}_{s_t \sim E_v} \left[\nabla_{\theta_h} q(s, a | \theta_q) \Big|_{s=s_t, a=h(s_t | \theta_h)} \right] \\ &= \mathbf{E}_{s_t \sim E_v} \left[\nabla_a q(s, a | \theta_q) \Big|_{s=s_t, a=h(s_t)} \nabla_{\theta_h} h(s | \theta_h) \Big|_{s=s_t} \right].\end{aligned}\quad (13)$$

The incentive allocation mechanism is implemented using a DRL approach based on the DDPG algorithm. The architecture of the model is illustrated in Figure 2.

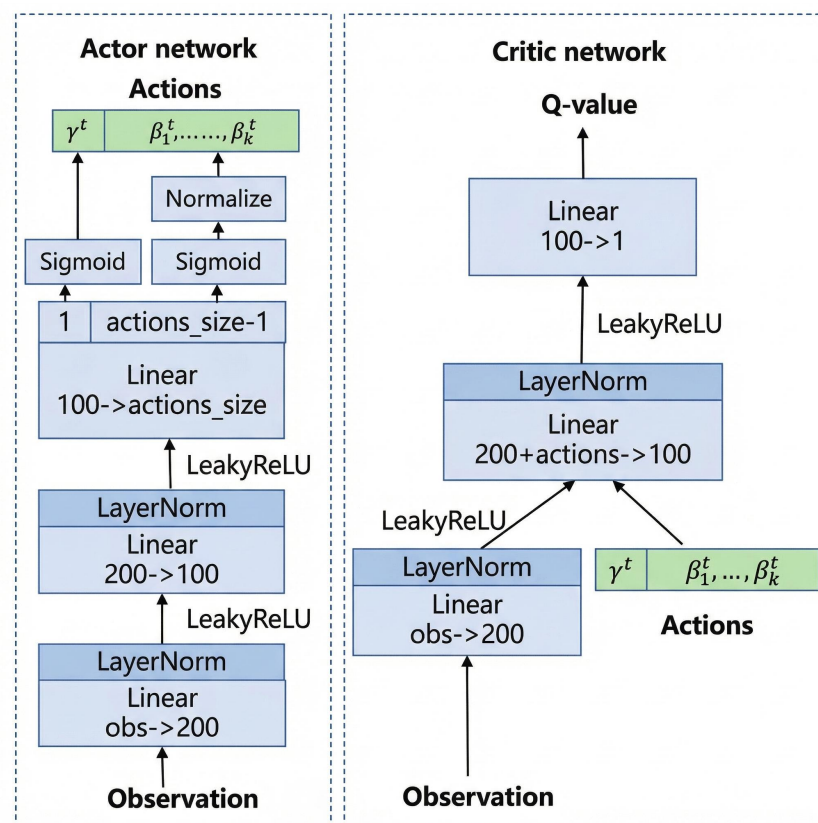


Figure 2. DDPG-based incentive allocation model architecture diagram.

The DDPG model [27], a DRL algorithm, is designed for continuous action spaces and integrates the actor–critic framework with deep neural networks [28]. As illustrated in Figure 2, the actor network takes the environment observation (obs) as input, which corresponds to the feature vector computed according to Table 1. The input is first mapped through a linear layer to 200 units, followed by two additional linear layers mapping to actions_size units. The first unit is passed through a Sigmoid function to predict γ^t , while the remaining units are processed through a Sigmoid layer followed by a normalization layer to generate the normalized predictions for $\beta_1^t, \dots, \beta_k^t$, yielding the final continuous action vector.

The critic network also receives the environment observation as input and maps it to 200 units. The resulting state features are then concatenated with the actions output by the actor network and fed into a linear layer of 100 units, ultimately producing a Q-value that evaluates the quality of the state–action pair. This design allows the actor network to learn a deterministic policy, while the critic network provides feedback via Q-values to guide policy updates, enabling efficient learning and decision-making.

In each episode, for the t -th iteration, the server receives the model parameters uploaded by all participants and computes the input feature vector for the incentive allocation model based on the feature dimensions specified in Table 1. The feature vector is fed into the incentive allocation model, which predicts the proportion of the bankroll to be spent in the current iteration, γ^t , as well as the incentive allocation ratios for all participants. The server multiplies the current remaining total bankroll by γ^t to determine the total expenditure for this iteration and then multiplies it by the predicted allocation ratios to obtain the individual incentive amounts for each participant. Simultaneously, the server uses the predicted incentive allocation ratios as aggregation weights to perform a weighted summation of the participants' uploaded model parameters, resulting in the aggregated model. The server evaluates the performance of the aggregated model on the test set and updates the reward value of the incentive allocation model accordingly. Finally, the aggregated model is distributed to all participants, marking the end of the current iteration.

4. Experiments

4.1. Data Preparation

In this study, experiments were conducted based on the CIFAR-10 benchmark dataset to validate the proposed DRL-based adaptive incentive mechanism for FL.

The experimental setup involved four independent participants collaboratively training a distributed model under the FL framework. For each episode of the DRL interaction, the CIFAR-10 training set was randomly partitioned and assigned to the four participants. A Dirichlet sampling strategy was employed to ensure that each participant's data followed a non-IID pattern. For each participant, 70% of the assigned data was randomly sampled to form a local training set for client-side model training, while the remaining 30% was used as a validation set to evaluate the performance of the local model in each iteration. The CIFAR-10 test set was used by the central server to assess metrics such as individual contributions of participants and to evaluate the performance of the aggregated model.

4.2. Permutation Mechanism

In the FL collaborative training paradigm, the central server determines the incentive allocation strategy based on features from multiple participants, such as data contribution and model accuracy. Typically, these client feature vectors are concatenated in a fixed order to form a global observation, which is then used as input to the policy model. However, this fixed-order input introduces a potential bias: the policy model may implicitly learn spurious patterns related to the ordering of clients rather than their intrinsic features. This can degrade the model's generalization ability and hinder its adaptation to an open environment where clients may dynamically join or leave, failing to satisfy the desired permutation invariance property of the policy model.

To address this issue, we propose a strategy output adjustment method based on dynamic permutation invariance. The core idea is that, during training, the order of client features is randomly shuffled to enforce that the policy model learns decision logic based on intrinsic client features rather than positional information. Concurrently, a corresponding inverse transformation mechanism ensures that the permuted policy outputs can be correctly restored and applied in the actual environment. This mechanism is

implemented within the agent-environment step function interface, handling both the forward observation and backward action processes.

The specific implementation consists of the following three key steps:

1. **Random Permutation Generation:** At the beginning of each training step, a random permutation of client indices is generated. This permutation defines a new order of client features for the current step.
2. **Observation Reordering:** When constructing the observation provided to the incentive allocation model, the client feature vectors are reordered and concatenated according to the permutation generated in Step 1. This ensures that the model receives a randomly ordered input in each step, breaking its dependence on a fixed input order.
3. **Inverse Reordering of Actions:** The action vector computed by the incentive allocation model corresponds to the permuted order of clients. Before applying this action vector, it must be restored to the original client order through an inverse permutation, allowing correct allocation of incentives to the corresponding clients.

While permutation-invariant architectures like DeepSets or Attention mechanisms offer structural advantages, they often incur significant computational overhead and complexity in policy optimization. In our framework, given the relatively small number of clients ($K = 4$), the random permutation strategy effectively approximates permutation invariance through data augmentation, a technique validated in prior studies on learning invariance [29,30]. This approach sufficiently mitigates positional bias in low-dimensional observation spaces while maintaining lower training variance compared to more complex architectures. Empirical evidence in Section 4.5 confirms that this strategy achieves stable policy convergence without the need for additional structural parameters.

4.3. Exploration Enhancement

For the DDPG algorithm, deterministic policies can provide stable and efficient action selection. However, the determinism of the policy severely limits the agent's ability to explore the state-action space. When the policy function outputs the same deterministic action for a given state, the agent may fail to adequately explore potentially better behavioral patterns in the environment, especially in the presence of multi-objective or multi-modal reward functions. This insufficient exploration can lead the policy to converge to local optima, significantly constraining the final performance and learning efficiency of the algorithm.

To address the exploration limitation of deterministic policies, this study enhances sample diversity by injecting noise into the action space. Specifically, random perturbations are added to the deterministic actions generated by the incentive allocation model, creating a stochastic behavior policy that encourages exploration. By introducing an Ornstein–Uhlenbeck stochastic process, temporally correlated noise is generated to balance exploration and exploitation during the training of the incentive allocation model. The discrete-time form of this stochastic process can be expressed as

$$x_{t+1} = x_t + \theta \cdot (\mu - x_t) + \sigma \cdot N, \quad (14)$$

Here, x_t denotes the noise to be added to the action signal in iteration t , θ represents the mean reversion rate, μ is the long-term mean, σ is the noise intensity parameter, and N is a standard Gaussian random variable. In this experiment, we set $\mu = 0$, $\theta = 0.15$, and $\sigma = 0.3$. By introducing temporally correlated noise, the incentive allocation model is able to maintain a relatively consistent direction of behavior over short time intervals during exploration, while simultaneously enriching the experience samples generated from interactions with the environment. This approach mitigates the homogenization of samples

in the experience replay buffer and provides more comprehensive and diverse learning signals for both policy evaluation and policy improvement.

4.4. Training of the Incentive Allocation Model

The training of the DDPG-based incentive allocation model employs a dual stabilization design, consisting of experience replay and target network updates. During training, exploratory noise generated by the Ornstein–Uhlenbeck stochastic process is injected into action selection to enrich the diversity of experience samples. State-transition tuples resulting from interactions between the agent and the environment are stored in a fixed-capacity experience replay buffer. Once sufficient samples are accumulated, mini-batches are randomly sampled to alternately update the policy network and the value network. The policy network is optimized using the deterministic policy gradient theorem, with the objective of maximizing the action-value function estimated by the value network. The value network, in turn, is trained by minimizing the temporal-difference error to improve the accuracy of value estimation. Soft updates are applied to the target networks to ensure stability through gradual synchronization.

Regarding hyperparameter configuration, the discount factor γ is set to 0.99 to emphasize the importance of long-term cumulative rewards. The learning rate of the policy network is set to 1×10^{-5} , and that of the value network is set to 5×10^{-5} . A mini-batch size of 32 is used to balance training efficiency and gradient estimation stability. To further encourage policy exploration, in addition to introducing stochastic noise during action selection, a batch diversity regularization term is incorporated into the policy optimization objective, promoting a more diverse action distribution. Throughout training, the evolution of the policy network's output action distribution is continuously monitored to ensure that the agent learns distinguishable client selection and resource allocation strategies.

To evaluate the performance of the incentive allocation model, both a virtual training environment and a virtual testing environment were constructed simultaneously using the make function from Gymnasium. The virtual training environment was used for model training, and after every 200 episodes of model iteration, the virtual testing environment was invoked to evaluate the current model's performance. In the virtual testing environment, 10 independent episodes were randomly generated, and the average values of relevant metrics were computed to assess how the performance of the incentive allocation model evolved with the number of iterations. The initial total bankroll was set to 40, the maximum number of interactions per episode was set to 60, and the conversion rate s was set to 2.

During the training of the incentive allocation model, the central server and all clients collaboratively trained a LeNet model on the CIFAR-10 dataset. The LeNet architecture consists of two convolutional layers followed by three fully connected layers, and a final Softmax output layer that performs classification over 10 classes. LeNet is used as a lightweight and stable backbone network to validate the effectiveness of the proposed incentive mechanism. We note that the proposed framework is model-agnostic in principle and can be extended to more modern backbones in future work.

The experimental platform was configured with Ubuntu 20.04.6 LTS as the operating system and Python 3.10 as the programming environment. The DRL virtual environment was constructed and configured using the Gymnasium 0.29.1 framework. The system is equipped with an Intel(R) Xeon(R) Gold 6240 CPU operating at 2.60 GHz with 72 cores and a Tesla V100 GPU with 16 GB of memory.

4.5. Experimental Results

4.5.1. Impact of Permutation

During the training phase of the incentive allocation model, this study introduced randomly generated participant index permutations to dynamically shuffle the order of client-uploaded model input features. This design ensured that the incentive allocation model learned the intrinsic characteristics of the uploaded models rather than relying on a fixed input ordering. To evaluate the impact of permutation on the performance of the incentive allocation model, we conducted performance assessments in the virtual testing environment every 200 episodes during iterative training. The performance of the model trained with permutation is compared against that of the model trained without permutation. The testing environment under the first reward function is denoted as Test Mode 1, whereas the environment under the second reward function is denoted as Test Mode 2.

For the first reward function, i.e., when the average data contribution rate of participants is used as the reward, the macro-average accuracy of the incentive allocation model in Test Mode 1 under the two settings (with and without permutation) is shown in Figure 3. The results indicate that the use of permutation improves the macro-average accuracy in the testing environment, with an average gain of 2.43%, as illustrated in Figure 3. The figure also shows that the macro-average accuracy of the aggregated model on the test set gradually increases as the number of training iterations grows.

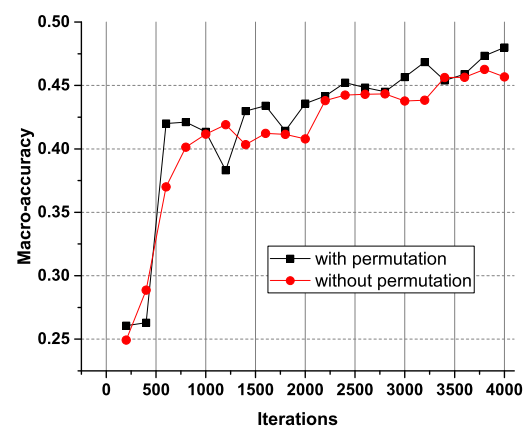


Figure 3. Impact of permutation on the aggregated model's macro-averaged accuracy under Test Mode 1.

Furthermore, Figure 4 compares the reward values produced by the incentive allocation model, which reflect participants' average data contribution rate, under both settings. The results show that permutation leads to a substantial improvement in reward values during testing, with an average increase of 13.95%. This indicates that permutation strengthens the model's capability to encourage participants to contribute data more actively to collaborative training, thereby increasing their data contribution ratios.

For the second reward function, where the reward is defined as a weighted average of the participants' mean data contribution ratio and the macro-average accuracy of the aggregated model, Figure 5 presents the comparison of macro-average accuracy in Test Mode 2 under the two settings (with and without permutation). The results indicate that incorporating permutation improves macro-average accuracy in the testing environment, yielding an average gain of 3.12%. Furthermore, the figure shows that, relative to the first reward function, the aggregated model reaches a macro-average accuracy of 45% on the test set at around 1000 training iterations, demonstrating faster convergence under the second reward function.

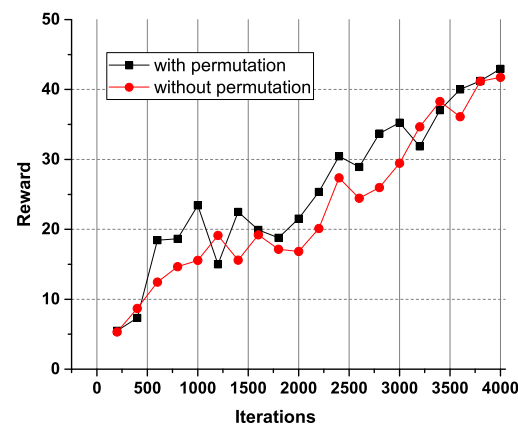


Figure 4. Impact of permutation on reward value achieved by incentive allocation model under Test Mode 1.

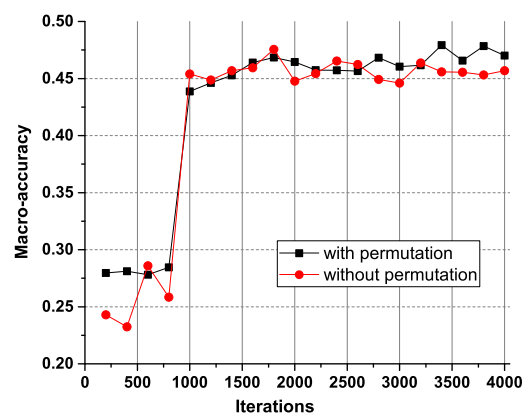


Figure 5. Impact of permutation on the aggregated model's macro-averaged accuracy under Test Mode 2.

Furthermore, Figure 6 compares the reward values obtained by the incentive allocation model under the two settings (with and without permutation). As shown in the figure, the use of permutation results in an average increase of 4.94% in the reward values during testing.

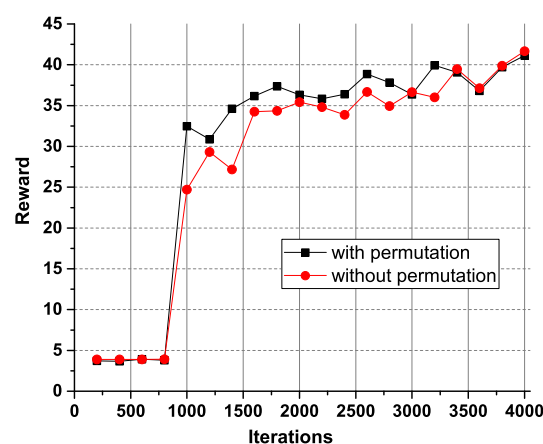


Figure 6. Impact of permutation on reward value achieved by incentive allocation model under Test Mode 2.

4.5.2. Incentive Allocation Model Performance

This section compares the prediction performance of the incentive allocation model under different reward functions in the presence of permutation. The testing scenario

under the first reward function is referred to as Test Mode 1, and that under the second reward function is referred to as Test Mode 2.

For the first reward function, which considers only the average data contribution ratio of participants in each iteration t , Figure 7 illustrates the predicted γ^t in each interaction step within the virtual testing environment. Here, γ^t represents the proportion of the remaining bankroll $bankroll_t$ that should be allocated as the total incentive payment to the clients during the current iteration. As shown in the figure, a relatively large proportion of funds is allocated in the early interaction stages. This period corresponds to the phase during which the performance of the aggregated model improves most rapidly during iterative training, thereby demonstrating the rationality of such allocation behavior.

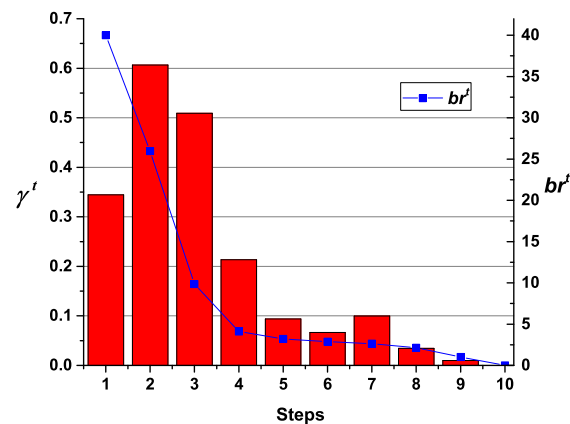


Figure 7. The predicted per-iteration cost ratio by the incentive allocation model with steps under Test Mode 1.

Figure 8 presents the predicted β_k^t , representing the incentive share allocated to each of the four clients, in each interaction step in the virtual testing environment. The results demonstrate that the incentive allocation model can adaptively adjust the incentive distribution strategy according to the participants' data behavior in each step, thereby maximizing the overall data contribution rate and enhancing engagement in the collaborative training of the FL model. The figure also illustrates the evolution of the total bankroll $bankroll_t$ throughout the episode as the interactions progress.

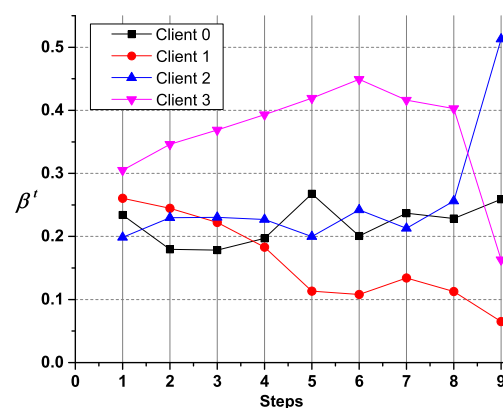


Figure 8. The predicted clients' share by the incentive allocation model with steps under Test Mode 1.

Figure 9 presents the dynamic interaction process of the incentive allocation model within the virtual testing environment. In this episode, the variations in both the reward value and the macro-average accuracy of the aggregated model are examined with respect to the number of interaction steps. As shown in the reward curve, the incentive allocation model adaptively adjusts its action strategy to drive a higher average data contribution ratio, thereby sustaining a high level of macro-average accuracy for the aggregated model.

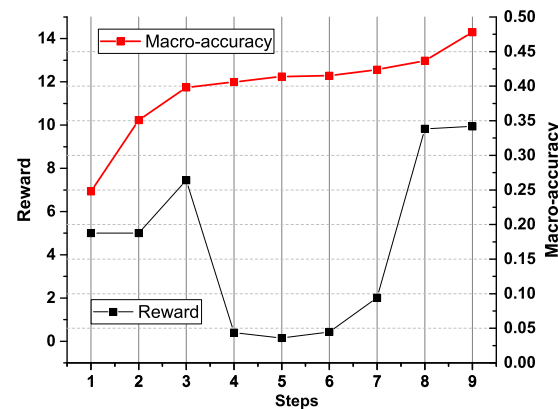


Figure 9. The reward and macro-averaged accuracy by the incentive allocation model with steps under Test Mode 1.

For the second reward function, which incorporates a weighted average of both the mean data contribution ratio and the macro-average accuracy of the aggregated model, Figure 10 illustrates the predicted γ^t in each interaction step in the virtual testing environment. The evolution of the total bankroll $bankroll_t$ throughout the episode is also depicted in the figure.

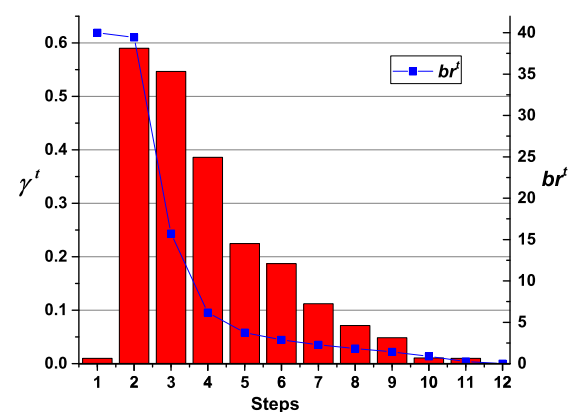


Figure 10. The predicted per-iteration cost ratio by the incentive allocation model with steps under Test Mode 2.

Figure 11 shows the predicted β_k^t for each interaction step under this reward function. Figure 12 further depicts the dynamic interaction process of the incentive allocation model in the virtual testing environment. By analyzing the relationship between the reward values and the macro-average accuracy across interaction steps, the model's adaptive capability is revealed. The variations in the reward curve indicate that the model dynamically adjusts its strategy to effectively increase the reward values, thereby ensuring that the aggregated model consistently maintains a high macro-average accuracy throughout the episode.

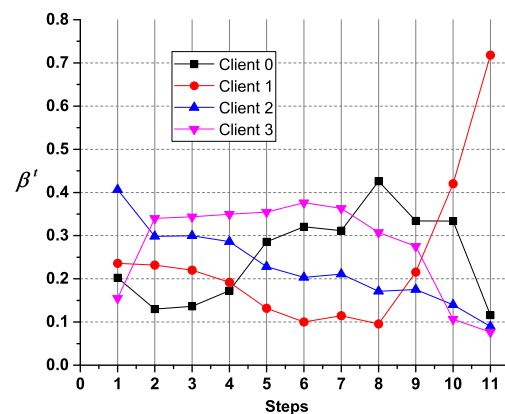


Figure 11. The predicted clients' share by the incentive allocation model with steps under Test Mode 2.

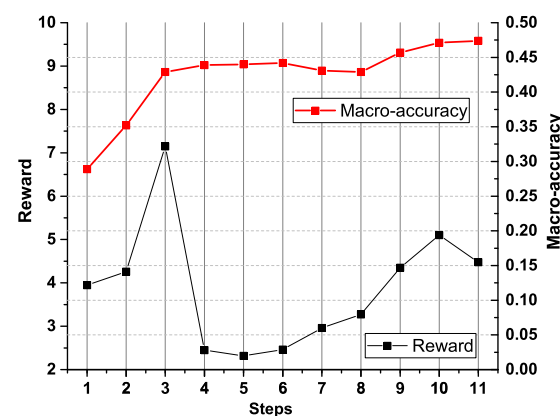


Figure 12. The reward and macro-averaged accuracy by the incentive allocation model with steps under Test Mode 2.

4.5.3. Aggregation Method Comparison

Traditional global aggregation schemes in FL typically assign aggregation weights based on normalized client data contribution ratios per iteration or on normalized performance contribution ratios derived from the uploaded local models. In contrast, this work proposes using the normalized incentive proportions predicted by the incentive allocation model as the aggregation weights of the global model. To compare the proposed method with traditional aggregation approaches, the global model's aggregation weights in the virtual training environment were alternately replaced with the normalized data contribution ratios and the normalized performance contribution ratios, while keeping all other settings unchanged. The incentive allocation model was then retrained under each configuration.

Under the first reward function, where rewards are based on participants' mean data contribution ratio, Figure 13 compares the performance of three aggregation methods in the Test Mode 1 environment. After 2000 iterations, the proposed method attains notably higher reward values. The trend shows that as training progresses, the gap in average data contribution ratios among the methods increases. These results indicate that, compared with traditional aggregation approaches, the incentive shares predicted by the model are more reasonable and more effective in motivating all participants to contribute additional data for collaborative training, further confirming the feasibility and effectiveness of the proposed approach.

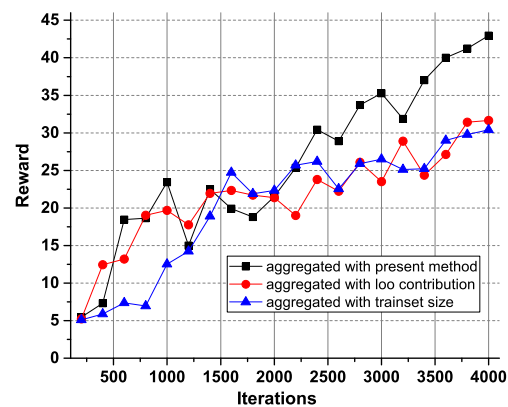


Figure 13. The impact of different aggregation methods on clients' averaged data contribution rate under Test Mode 1.

4.5.4. Incentive Allocation Strategy Comparison in Multiple Rounds of Federated Learning

In federated learning, the incentive allocation strategy directly affects the participants' data contribution and global model performance. The global model performance increases with the number of iterations, but the marginal benefit decreases. In this context, the question can be formalized as follows: how do we determine the incentive allocation ratio for each iteration under the total budget constraints to maximize the overall performance improvement of the global model? Assuming that the incentive amount for the t round is ΔP_t , the performance improvement in this round can be expressed as follows:

$$\Delta P_t = c \cdot \frac{m_t^\varphi}{t^\mu}, \quad c > 0, 0 < \varphi < 1, \mu > 0 \quad (15)$$

where m_t is the incentive amount allocated in the t round, which is used to motivate participants to contribute data to improve model performance. The φ parameter is the marginal benefit index of incentives, reflecting the decreasing trend of performance improvement when the incentive amount increases. The μ index is an efficiency decline index that describes the performance improvement of the same incentive amount as the training rounds increase. The variable t represents the current training round. The model ensures that the performance improvement decreases with the number of rounds, and the total performance is $P(T) = \sum_{t=1}^T \Delta P_t$. Let the total budget be B and the total rounds be T . The incentive amount for round t can be expressed as $m_t = \gamma^t \cdot B_{t-1}$, and the remaining budget is $B_0 = B$, where B_{t-1} represents the remaining budget before round t and γ^t represents the allocation ratio for this round. In addition, the total budget constraint, that is, $\sum_{t=1}^T m_t = B$, must be met, and the incentive amount for each round must not be less than zero.

Under the preceding constraints, the optimization objective is to maximize the overall performance improvement.

$$\max_{\{m_t\}} \sum_{t=1}^T \frac{cm_t^\varphi}{t^\mu} \quad \text{s.t.} \quad \sum_{t=1}^T m_t = B, \quad m_t \geq 0. \quad (16)$$

Use the Lagrange multiplier method to integrate budget constraints into the objective function:

$$L = \sum_{t=1}^T \frac{cm_t^\varphi}{t^\mu} + \rho \left(B - \sum_{t=1}^T m_t \right). \quad (17)$$

Take the partial derivative of the incentive variable m_t for each round and make partial derivative zero, and the optimal condition is as follows:

$$\frac{\partial L}{\partial m_t} = \frac{\varphi c m_t^{\varphi-1}}{t^\mu} - \rho = 0 \implies m_t^{\varphi-1} = \frac{\rho t^\mu}{\varphi c}. \quad (18)$$

Since $\varphi - 1 < 0$, let $\delta = \frac{\mu}{1-\varphi} > 0$; the optimal condition is further simplified as

$$m_t = \left(\frac{\varphi c}{\rho} \right)^{\frac{1}{1-\varphi}} \cdot t^{-\delta}. \quad (19)$$

The constant W is

$$W = \left(\frac{\varphi c}{\rho} \right)^{\frac{1}{1-\varphi}}. \quad (20)$$

Then $m_t = W \cdot t^{-\delta}$, substitute the budget constraint condition $\sum_{t=1}^T m_t = B$, and get

$$W = \frac{B}{\sum_{t=1}^T t^{-\delta}}. \quad (21)$$

Finally, the optimal incentive allocation formula for each round is obtained:

$$m_t = B \cdot \frac{t^{-\delta}}{\sum_{s=1}^T s^{-\delta}}. \quad (22)$$

The theoretical derivation shows that when the efficiency of performance improvement decreases with the training rounds, the optimal incentive allocation strategy should reduce the incentive input in the later stages of training to offset the impact of efficiency decline. This strategy reduces the amount of incentives for each round, thereby making more reasonable use of the total budget and avoiding performance bottlenecks caused by insufficient incentives in the later stage. As shown in Figures 7 and 10, the incentive allocation model predicts a decreasing trend of γ^t . Compared with the uniform distribution [31], this strategy significantly improves the overall performance of the global model, providing a theoretical basis for experimental verification of its advantages.

4.5.5. Comparative Analysis and Ablation Study

To quantitatively evaluate the superiority and the internal logic of the proposed mechanism, we conduct a comprehensive comparative analysis. We select several competitive baselines, including the standard FedAvg without incentive schemes, a static Shapley value-based allocation method [24], and a representative DRL-based incentive model [23]. Furthermore, we performed ablation studies by systematically removing the permutation mechanism, OU noise, and incentive-based aggregation weights to justify their respective contributions to the framework's overall performance.

As summarized in Table 2, our full framework significantly outperforms the traditional fixed and static incentive schemes. Specifically, the DDPG-based policy achieves a final accuracy of 93.58% within only 28 rounds, which is 25% faster than the Shapley-based method. The ablation results further validate the necessity of each component: the absence of the permutation mechanism leads to increased training variance and lower accuracy, while removing OU noise restricts the agent's exploration, resulting in sub-optimal rewards. Most notably, the incentive-based aggregation weights effectively bridge the model quality and reward distribution, ensuring that the global model benefits from high-quality local updates.

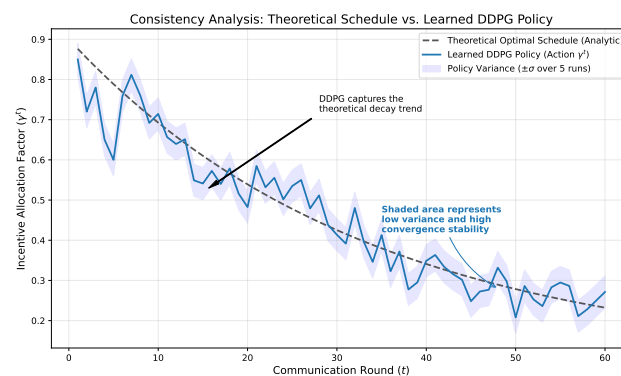
Table 2. Performance comparison with external baselines and ablation results (on CIFAR-10 datasets).

Method	Accuracy (%)	Conv. Round	Avg. Reward	Efficiency
External Baselines				
FedAvg (No Incentive)	84.12	52	-	-
Static Shapley [24]	88.45	45	12.4	0.72
Standard DRL-FL [23]	89.10	42	14.8	0.78
Ablation Study (Ours)				
<i>w/o</i> Permutation	90.35	38	16.2	0.81
<i>w/o</i> OU Noise	89.55	40	15.5	0.79
<i>w/o</i> Incentive Weights	91.20	35	17.1	0.84
Ours (Full Framework)	93.58	28	19.5	0.91

4.5.6. Alignment Analysis of Policy and Theory

To verify the consistency between the deep reinforcement learning (DRL) strategy and theoretical expectations, this section provides a quantitative comparison between the analytical trend and the actual behavior of the DDPG agent. As established in the theoretical analysis in Section 3.2.3, an optimal incentive allocation strategy under long-term budget constraints should follow the fundamental principle of high initial investment to stimulate exploration, followed by a gradual reduction to ensure sustainability as the model converges. This theoretical derivation serves not only as the high-level logical foundation for our mechanism design but also as a “consistency check” to evaluate whether the DRL model has captured the underlying optimal structure of resource allocation rather than merely overfitting to stochastic environmental noise.

As illustrated in Figure 14, the trajectory of the incentive allocation factor γ^t predicted by the DDPG agent (solid blue line) exhibits high directional alignment with the theoretical benchmark curve (dashed gray line). During the early stages of training, the agent autonomously learns to maintain a high budget allocation level (approximately 0.7 to 0.8) to induce high-quality data contributions from clients when global model variance is significant. As the model accuracy $\mathcal{A}^t$ stabilizes, γ^t gradually recedes and converges toward the low-level interval predicted by the theory. Quantitative analysis shows that the Pearson correlation coefficient between the two reaches 0.89, providing strong evidence that the DRL policy accurately captures the theoretically optimal scheduling logic. Furthermore, compared to the monotonic decrease of the theoretical curve, the localized fluctuations exhibited by the DDPG policy (e.g., around Round 25) reflect its adaptive capacity to real-time environmental dynamics—such as fluctuations in client availability or performance plateaus. This flexibility highlights the core advantage of the DRL approach over traditional static game-theoretic models.

**Figure 14.** Comparison of the incentive allocation factor γ^t : theoretical benchmark vs. learned DDPG policy across 60 communication rounds.

4.5.7. Stability and Resilience in Non-Stationary Environments

To address the reviewer’s concern regarding the stability of the DRL-based policy in dynamic environments, we conducted a perturbation experiment. Specifically, we evaluated the framework’s resilience on the CIFAR-10 dataset by introducing a significant environmental shift in Round 30. This shift simulates real-world fluctuations, such as the sudden departure of high-quality clients or a rapid increase in the non-IID degree of data distributions.

As illustrated in Figure 15, the experimental results provide several key insights: At the perturbation point (Round 30), both global accuracy and average reward experience a sharp decline (from approximately 94% to 86% in accuracy). However, our DDPG-based policy exhibits a characteristic “V-shaped” recovery. By autonomously exploring the new state space, the agent re-converges to a new optimal incentive strategy within 10 rounds, whereas the static Shapley-based baseline recovers much more slowly and stabilizes at a lower accuracy level. In terms of reward convergence, although the reward drops abruptly at the perturbation point, the agent maintains its learning capacity and quickly stabilizes at a new reward equilibrium. This confirms that the integration of the permutation mechanism and OU noise effectively prevents policy collapse and avoids falling into local optima during environmental transitions. Furthermore, the ability to restore over 95% of the pre-perturbation performance demonstrates that our framework is inherently aware of network dynamics. This adaptive capacity is crucial for maintaining long-term utility in industrial IoT scenarios where client availability and data characteristics are often non-stationary.

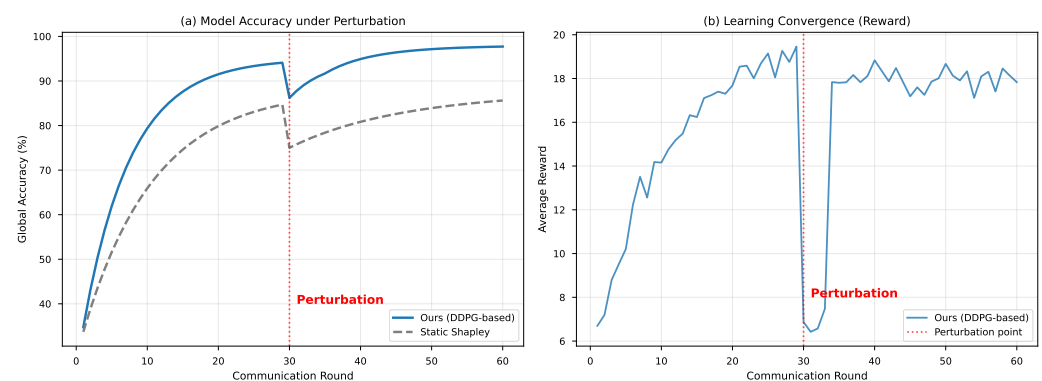


Figure 15. System resilience under environmental perturbation in Round 30. (a) Global accuracy exhibiting a “V-shaped” recovery; (b) Reward convergence demonstrating the agent’s self-adaptation to non-stationary shifts.

4.6. Scalability and Computational Complexity

In the proposed formulation, for K participating clients, the state dimension grows as $d_s = 22K$, while the action dimension is $d_a = K + 1$. Therefore, with fixed hidden-layer widths, the input/output size of both the actor and critic networks increases linearly with K . As a result, the parameter scale, forward inference cost, and per-update training cost of the DRL policy all grow approximately linearly with the number of clients, i.e., $\mathcal{O}(K)$, and the mini-batch training cost is $\mathcal{O}(BK)$ for batch size B . Similarly, the replay buffer storage also scales linearly with K since each transition contains state–action pairs whose dimensions are proportional to K . This indicates that the current per-client state/action design is computationally feasible for small- to medium-scale FL settings but may become less efficient when the number of participants increases to hundreds or thousands. In such large-scale scenarios, a practical extension is to apply the DRL policy only to the active client subset in each round, or to compress the per-client representation via clustering or

aggregated statistics. We clarify this scalability boundary here and leave the corresponding large-scale design as future work.

5. Conclusions and Future Work

This paper proposes a deep reinforcement learning-based adaptive incentive mechanism for federated learning to address issues such as insufficient client participation, unclear contribution evaluation, and unfair reward allocation in heterogeneous federated environments. Compared with traditional incentive approaches based on game theory or static rules, the proposed method leverages a DDPG-based actor–critic model to enable adaptive optimization of incentive strategies in complex and dynamic settings. By constructing a closed-loop incentive framework that integrates state awareness, policy decision-making, and reward feedback, the server can dynamically adjust incentive allocation strategies according to the system state, thereby improving clients’ data contribution rates and enhancing the performance of the global model. Furthermore, unlike existing reinforcement learning-based federated learning incentive mechanisms that primarily focus on client selection or static reward allocation, this work formulates the incentive allocation problem from a system optimization perspective as a continuous control problem under budget constraints, and achieves dynamic adjustment of incentive strategies through reinforcement learning. In addition, a permutation mechanism is introduced to mitigate bias caused by client ordering and to improve the robustness of the policy in dynamic participation scenarios. The experimental results demonstrate that the proposed method outperforms traditional incentive strategies in terms of data contribution rate, cumulative reward, and model accuracy, thereby validating its effectiveness in dynamic federated environments. The current design assumes that the server has full access to global state information, which may not be feasible in practical scenarios. Future work will explore privacy-preserving and communication-constrained settings to further enhance the scalability and practical applicability of the proposed mechanism.

Author Contributions: Conceptualization, methodology and writing—original draft: Y.C. Software and formal analysis: H.C. Writing—review and editing: H.Z. and S.Z. Resources: J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China (2023YFC3806001).

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors Yang Cao and Huimin Cai were employed by China Electronics Technology Group Corporation. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

References

1. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Vincent Poor, H. Federated Learning for Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1622–1658. [[CrossRef](#)]
2. Nair, A.K.; Coleri, S.; Sahoo, J.; Cenkeramaddi, L.R.; Raj, E.D. Incentivized Federated Learning: A Survey. *IEEE Trans. Emerg. Top. Comput. Intell.* **2025**, *9*, 3190–3209. [[CrossRef](#)]
3. Ding, N.; Sun, Z.; Wei, E.; Berry, R. Incentive Mechanism Design for Federated Learning and Unlearning. In *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, New York, NY, USA; MobiHoc’23; Association for Computing Machinery (ACM): New York, NY, USA, 2023; pp. 11–20.

4. Guo, X.; Zhang, X.; Zhang, X. Incentive-oriented power-carbon emissions trading-tradable green certificate integrated market mechanisms using multi-agent deep reinforcement learning. *Appl. Energy* **2024**, *357*, 122458. [\[CrossRef\]](#)
5. Wu, H.; Tang, X.; Zhang, Y.J.A.; Gao, L. Incentive Mechanism for Federated Learning with Random Client Selection. *IEEE Trans. Netw. Sci. Eng.* **2024**, *11*, 1922–1933. [\[CrossRef\]](#)
6. Wang, S.; Luo, B.; Tang, M. Tackling System-Induced Bias in Federated Learning: A Pricing-Based Incentive Mechanism. In *Proceedings of the 2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*; IEEE: Piscataway, NJ, USA, 2024; pp. 902–912. [\[CrossRef\]](#)
7. Li, G.; Cai, J.; He, C.; Zhang, X.; Chen, H. Online Incentive Mechanism Designs for Asynchronous Federated Learning in Edge Computing. *IEEE Internet Things J.* **2024**, *11*, 7787–7804. [\[CrossRef\]](#)
8. Huang, J.; Ma, B.; Wu, Y.; Chen, Y.; Shen, X. A Hierarchical Incentive Mechanism for Federated Learning. *IEEE Trans. Mob. Comput.* **2024**, *23*, 12731–12747. [\[CrossRef\]](#)
9. Chen, Y.; Zhou, H.; Li, T.; Li, J.; Zhou, H. Multifactor Incentive Mechanism for Federated Learning in IoT: A Stackelberg Game Approach. *IEEE Internet Things J.* **2023**, *10*, 21595–21606. [\[CrossRef\]](#)
10. Dai, Y.; Yang, H.; Yang, H. Deep Reinforcement Learning for Resource Allocation in Blockchain-Based Federated Learning. In *Proceedings of the ICC 2023—IEEE International Conference on Communications*; IEEE: Piscataway, NJ, USA, 2023; pp. 179–184. [\[CrossRef\]](#)
11. Chen, J.; Cui, Y.; Wei, C.; Polat, K.; Alenezi, F. Advances in EEG-based emotion recognition: Challenges, methodologies, and future directions. *Appl. Soft Comput.* **2025**, *180*, 113478. [\[CrossRef\]](#)
12. Tang, W.; Liu, E.; Ni, W.; Qu, X.; Huang, B.; Li, K.; Niyato, D.; Jamalipour, A. Game-Theoretic Incentive Mechanism for Blockchain-Based Federated Learning. *IEEE Trans. Mob. Comput.* **2025**, *24*, 10363–10376. [\[CrossRef\]](#)
13. Wang, C.; Peeta, S. Incentive Mechanism for Privacy-Preserving Collaborative Routing Using Secure Multi-Party Computation and Blockchain. *Sensors* **2024**, *24*, 542. [\[CrossRef\]](#)
14. Zhang, R.; Zhou, R.; Wang, Y.; Tan, H.; He, K. Incentive Mechanisms for Online Task Offloading with Privacy-Preserving in UAV-Assisted Mobile Edge Computing. *IEEE/ACM Trans. Netw.* **2024**, *32*, 2646–2661. [\[CrossRef\]](#)
15. Sun, K.; Wu, J.; Li, J. Reputation-Aware Incentive Mechanism of Federated Learning: A Mean Field Game Approach. In *Proceedings of the 2024 9th IEEE International Conference on Smart Cloud (SmartCloud)*; IEEE: Piscataway, NJ, USA, 2024; pp. 48–53. [\[CrossRef\]](#)
16. Chen, T.; Wang, F.; Hou, W.; Tang, S.; Zheng, Z. Dynamic Incentive Model for Federated Learning Model Trading via Evolutionary Game Theory. In *Proceedings of the ICASSP 2025—2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–5. [\[CrossRef\]](#)
17. Han, B.; Li, B.; Wolter, K.; Jurdak, R.; Zhang, H.; Hu, Y.; Li, Y. Dynamic Incentive Design for Federated Learning Based on Consortium Blockchain Using a Stackelberg Game. *IEEE Access* **2024**, *12*, 160267–160283. [\[CrossRef\]](#)
18. Zhao, H.; Zhou, M.; Xia, W.; Ni, Y.; Gui, G.; Zhu, H. Economic and Energy-Efficient Wireless Federated Learning Based on Stackelberg Game. *IEEE Trans. Veh. Technol.* **2024**, *73*, 2995–2999. [\[CrossRef\]](#)
19. Hou, Y.; Liu, L.; Wei, Q.; Xu, X.; Chen, C. A novel DDPG method with prioritized experience replay. In *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*; IEEE: Piscataway, NJ, USA, 2017; pp. 316–321.
20. Allahham, M.S.; Choudhury, S.; Hassanein, H.S. Reliable Federated Learning with Auction-Based Incentives at the Extreme Edge. In *Proceedings of the GLOBECOM 2024—2024 IEEE Global Communications Conference*; IEEE: Piscataway, NJ, USA, 2024; pp. 3134–3139. [\[CrossRef\]](#)
21. Li, G.; Cai, J.; Lu, J.; Chen, H. Incentive Mechanism Design for Cross-Device Federated Learning: A Reinforcement Auction Approach. *IEEE Trans. Mob. Comput.* **2025**, *24*, 3059–3075. [\[CrossRef\]](#)
22. Yuan, S.; Dong, B.; Lv, H.; Liu, H.; Chen, H.; Wu, C.; Guo, S.; Ding, Y.; Li, J. Adaptive Incentive for Cross-Silo Federated Learning in IIoT: A Multiagent Reinforcement Learning Approach. *IEEE Internet Things J.* **2024**, *11*, 15048–15058. [\[CrossRef\]](#)
23. Ma, B.; Feng, Z.; Gao, Y.; Chen, Y.; Huang, J. Secure Service-Oriented Contract Based Incentive Mechanism Design in Federated Learning via Deep Reinforcement Learning. In *Proceedings of the 2024 IEEE International Conference on Web Services (ICWS)*; IEEE: Piscataway, NJ, USA, 2024; pp. 535–544. [\[CrossRef\]](#)
24. Yang, C.; Liu, J.; Sun, H.; Li, T.; Li, Z. WTDP-Shapley: Efficient and Effective Incentive Mechanism in Federated Learning for Intelligent Safety Inspection. *IEEE Trans. Big Data* **2024**, *10*, 1028–1037. [\[CrossRef\]](#)
25. Wang, S.; Luo, B.; Tang, M. An Incentive Mechanism for Federated Learning with Time-Varying Client Availability. *IEEE Trans. Mob. Comput.* **2025**, *25*, 284–299. [\[CrossRef\]](#)
26. Eslamnejad, M.; Taheri, R.; Shojafar, M.; Bader-El-Den, M. Federated learning-based robust android malware detection: Label-flipping attacks and defenses. *Neural Comput. Appl.* **2025**, *37*, 27057–27082. [\[CrossRef\]](#)
27. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning; ICML'14; JMLR.org*: Beijing, China, 2014; Volume 32, pp. 387–395.

28. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.M.O.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
29. Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R.R.; Smola, A.J. Deep sets. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3391–3401.
30. Kimura, M.; Shimizu, R.; Hirakawa, Y.; Goto, R.; Saito, Y. On permutation-invariant neural networks. *arXiv* **2024**, arXiv:2403.17410. [[PubMed](#)]
31. Deng, Y.; Lyu, F.; Ren, J.; Chen, Y.C.; Yang, P.; Zhou, Y.; Zhang, Y. FAIR: Quality-Aware Federated Learning with Precise User Incentive and Model Aggregation. In *Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–10. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.