Journal of Engineering and Technology for Industrial Applications

ITEGAM-JETIA

Manaus, v.11 n.52, p. 226-136. March./April., 2025.
DOI: https://doi.org/10.5935/jetia. v11i52.1637

ISSN ONLINE: 2447-0228

RESEARCH ARTICLE OPEN ACCESS

ENHANCING INCENTIVE SCHEMES IN EDGE COMPUTING THROUGH HIERARCHICAL REINFORCEMENT LEARNING

Gowtham R[1], Vatsala Anand[2], Yadati Vijaya Suresh[3], Kasetty Lakshmi Narasimha[4] R. Anil Kumar[5],V.Saraswathi[6]

[1]Computing and Information Technology, REVA University, Bangalore, Karnataka, India.
[2]Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India.
[3]Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal-518501, Andhra Pradesh, India.
[4]SVR Engineering College, Nandyal, AndhraPradesh, India.
[5]Aditya University, Surampalem, India.
[6]Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, AndhraPradesh, India

[1]http://orcid.org/0009-0006-6556-1089, [2]http://orcid.org/0000-0001-6143-250X, [3]http://orcid.org/0009-0005-5490-4990
[4]http://orcid.org/ 0000-0003-0569-3769, [5]http://orcid.org/0000-0003-0032-2555 [6]http://orcid.org/0009-0007-3374-8858

Email: gowthamramakrishna19@gmail.com, vatsala.anand@chitkara.edu.in, suri.yvs@gmail.com, kasetty.narsi@gmail.com, anidecs@gmail.com, saraswathiece89@gmail.com

ARTICLE INFO

Article History
Received: February 13, 2025
Revised: March 20, 2025
Accepted: March 15, 2025
Published: April 30, 2025

Keywords:
Machine learning
Hierarchical reinforcement
Edge computing
Server utility
Greedy algorithm.

ABSTRACT

Edge learning is a distributed approach for training machine learning models using data from edge devices. It preserves privacy by avoiding direct data sharing. However, existing systems struggle with resource inefficiency, malicious node participation and lack of long-term sustainability. These challenges reduce performance and discourage participation in edge learning. This paper proposes Chiron, a robustness-aware incentive mechanism designed to address these issues. Chiron employs a hierarchical reinforcement learning (HRL) framework to optimize resource allocation and ensure fair participation. The framework focuses on three key components: pricing strategy, resource distribution and malicious node detection. Chiron integrates system-level performance and model accuracy into its optimization goals, ensuring a balance between efficiency and effectiveness. The hierarchical structure includes three layers. The top layer determines the total incentive budget to achieve long-term sustainability. The middle layer allocates resources to minimize idle time and enhance efficiency. The bottom layer identifies and excludes malicious or lazy nodes that negatively impact the education process. By tackle both short-term and long-term objectives Chiron ensures fairness and performance stability. Extensive experiments validate Chiron's capabilities using real-world datasets like MNIST and CIFAR-10. Chiron demonstrates robust byzantine resistance and supports sustainable edge learning by addressing critical gaps in current approaches. This work contributes to the advancement of edge learning by presenting a reliable and efficient solution for real-world applications. By integrating security, sustainability and performance, Chiron enables edge learning to be both practical and impactful in various domains.

I. INTRODUCTION

It A new paradigm of machine learning called edge learning updates a global model across a set of data sources in a collaborative manner, where the data sources do not share their raw data [1]. This paradigm of data management solves two primary difficulties of ordinary machine learning with local data, namely the protection of data, as well as the excessive communication cost. This is done by having the training done on-device and only sharing model updates. Edge learning of this nature enables us to build quite intelligent systems, especially in privacy-sensitive areas. Important challenges coming up with edge learning! On the one hand, due to resource constraints in edge devices such as computational power, energy and bandwidth, it may be inefficient. While, Sybil or unreliable nodes in the network, may reduce the

performance of the global model. These nodes may report false or malicious updates unintentionally due to resource exhaustion [2] or intentionally. The third one is that it is pretty hard to maintain the contribution of the edge devices in the learning process for a long time. (2) Without proper and strong incentive mechanisms, devices even do not join in the cooperation or only cost very little, and meaningless resources where no device can be ready to contribute reasonable and necessary resource, which would risk the success of the whole cooperative process. Drawing on these challenges this study introduces Chiron. Chiron operates through a HRL framework that guarantees equitable and optimal consumption of resources (for the learning process), detection and eviction of malicious or lethargic nodes, and the long-term resilience of the learning process [3]. Firmly, this introduction defines the programs around edge learning and calls a broad rollout of Chiron in a system level manner. Edge devices usually entail low computational capabilities and resource constraint in terms of energy and bandwidth. Training complex models can tax these resources, causing delays and inefficiencies. Moreover, the variability of devices in an edge network may lead to some devices working much faster than others. If not ruled, the rich devices will take over, the poor devices will be unable to come together. Edge learning has the same decentralized characteristic, making it vulnerable to attack from malicious user [4]. An adversarial node can purposely provide false updates to cripple the learning algorithm. Lazy nodes, on the other hand, may minimize their computational effort by submitting low-quality or randomly generated updates. Both behaviours can compromise the integrity and performance of the global model. Edge learning relies on the voluntary participation of devices [5]. However, active devices incur costs in terms of energy, computation and communication. Without adequate incentives, devices may choose to opt out or reduce their contributions, leading to a decline in the quality and robustness of the global model. Chiron addresses these challenges by integrating robustness, efficiency and sustainability into its design.

It involves a three-layer HRL framework implemented in Chiron, which provides the required mechanisms at different levels of the incentive mechanism: The first layer involves the long-term bucketing where Chiron decides how much bucketing should be allocated towards the learning process in total. This budget is what is available to use in multiple training round to reward active devices. the objective is to allocate incentives in an optimal way to keep the system alive and functional for a long time. With the help of this layer, short-term needs and long-term needs are balanced, preventing resource depletion, thus retaining device engagement [6]. Layer 2 focuses on short-term optimization. It shares resource across participating devices in each training round. The goal of the allocation strategy is to reduce voids time, improve the efficiency of resources and balance the fairness of devices. This layer must assess the computational power, energy availability and communication bandwidth of each device. Through distributed balance resource allocation, it increases the diversity and speed of the learning process [7]. The third layer focuses on ensuring the reliability and integrity of the global model. It identifies and excludes malicious or lazy nodes that negatively impact the learning process. This is achieved by analysing the quality of updates submitted by each node. Nodes that consistently submit low-quality or harmful updates are penalized or excluded from future rounds. This layer ensures that only trustworthy nodes participate, enhancing the overall robustness of the system [8].

Chiron incorporates system performance metrics and model accuracy into its optimization objectives. By participating at a governance level, the economic structures that dictate behaviours can be tuned towards the wants of both the system and its future. But short-term efficiency cannot come at long-term sustainability or model performance. Chiron is modeled after Recursive Chaining, used to solve complex optimization problems in a hierarchical manner by breaking down to simple ones as sub-problems [9]. Using these layers, we design the incentivization mechanism for Chiron, where each layer has its own kind of target, such that both short term and long-term goals can be enforced against each other. Chiron can also be adapted to serve many edge learning scenarios due to its modular design. Chiron is an effective system to detect malicious or lazy nodes. It also ensures that the accuracy and integrity of the global model are maintained by examining the quality of updates proposed by individual nodes. This property is fundamental in decentralized systems where trust cannot be assumed [10].

Chiron has been empirically validated through large-scale experiments on real-world datasets such as MNIST and CIFAR-10. These datasets are standard benchmarks with broad use in machine learning research, thus ensuring robustness of the results for EDGE learning systems. Chiron shows superior accuracy, efficiency and robustness compared to state-of-the-art methods, as demonstrated by experiments. Chiron is a breakthrough in edge learning technology. For real-world applications, it offers a powerful and efficient solution by tackling key challenges like resource constraints, malicious behaviour and sustainability. Key contributions of this research include:

- A novel HRL-based framework for optimizing incentives in edge learning.
- An integrated approach that balances performance, efficiency and security.
- A scalable solution that can adapt to different edge learning scenarios and datasets.

Chiron offers a comprehensive solution to the challenges of edge learning. Its hierarchical design, integration of performance metrics and robustness against malicious behaviour make it a practical and impactful tool for advancing the field. By enabling efficient, secure and sustainable edge learning. Chiron paves the way for the widespread adoption of this innovative paradigm.

II. RELATED WORK

Edge learning is a widespread paradigm for decentralized machine learning to collaboratively learn models while preserving privacy among devices. This reviewing work focuses on the fundamental challenges as well as existing solutions in terms of incentive mechanisms, resource optimization, robustness and HRL within the edge learning domain. Chiron serves within the context of these studies according to the review, emphasizing its innovations in robustness, efficiency and sustainability. Edge learning is sustained through incentive mechanisms. There has been a fair number of studies being done around the incentive mechanisms for encouraging the edge devices to make use of its resources. Wang et al. Edge devices are rewarded based on their energy consumption and contributions in the game-theoretic model proposed in [11]. Although good for balancing interactions with minimal resources spent and guests rewarded, this approach fails to prevent bad behaviour or encourage long-term engagement. According to [12] developed upon this notion by modeling incentive schemes in contract theory. Their approach links rewards with the quality of contributions, thus enhancing the model performance. In [13] introduced an auction-based model to optimize resource contributions dynamically. Their mechanism

prioritizes high-quality contributions but does not integrate model performance into its design. Chiron overcomes these limitations by combining model accuracy and fairness into its incentive decisions. In [14] proposed a federated learning framework that includes trust-based incentives to mitigate malicious contributions. However, this approach requires extensive monitoring, which increases overhead.

Chiron reduces such overhead by using lightweight reinforcement learning to detect and exclude harmful nodes. Resource allocation is crucial in edge learning due to the limited computational, energy and bandwidth resources of edge devices. Various studies have addressed these constraints. According to [15] introduced a scheduling algorithm to minimize latency by optimizing resource allocation among edge devices. This approach showed improved efficiency but struggled with fairness across heterogeneous devices. For [16] proposed a decentralized resource optimization framework that reduces communication overhead. While effective, the framework does not include mechanisms to identify and address malicious behavior. In [17] developed a reinforcement learning-based solution for dynamic resource allocation. This method demonstrated significant improvements in efficiency. However, it lacked robustness mechanisms to handle malicious nodes or low-quality contributions. In [18] combined resource optimization with device reputation, ensuring fair allocation based on past performance. While this method improved fairness, it failed to incorporate dynamic adaptability, which Chiron achieves through its multi-layer HRL design. Chiron integrates resource optimization into its incentive mechanism, dynamically adapting to device performance and addressing heterogeneity across devices.

Maintaining robustness is critical in edge learning to ensure the integrity of the global model. Malicious nodes can introduce incorrect updates, significantly degrading model performance. In [19] introduced byzantine-robust aggregation methods that filter outliers based on statistical analysis. While effective, these methods require significant computational resources, making them unsuitable for real-time edge learning.

According to [20] proposed a reputation-based system where devices are scored based on their contributions over time. High-reputation devices are prioritized in model aggregation. This approach adds robustness but requires extensive data storage and monitoring. In [21] introduced a blockchain-based framework to secure model updates. While blockchain provides transparency and trust, its computational overhead limits scalability in resource-constrained environments. In [22] explored federated learning with adaptive aggregation, where the system adjusts weights based on contribution quality. This approach is computationally efficient but does not address long-term sustainability. Chiron advances robustness by integrating byzantine resilience into its hierarchical framework.

HRL is highly applicable for such complex optimization problems at the Edge learning. HRL breaks down tasks, allowing focused and effective solutions. According to [23] proved HRL works well in multi-agent systems, encouraging its deployment in edge learning. Their triple layer philosophy inspired Chiron's three-layer design, each of which has a particular optimization goal. Tang et al. HRL was used in a task allocation problem for distributed systems by [24]. The proposed method improved resource utilization at the expense of malicious contributions. In [25] HRL with federated learning where both resource allocation and model aggregation were optimized. However, their approach did not include incentive mechanisms from the ground level, while Chiron does it smoothly. Within the HRL framework that extends to Chiron from previous work, Chiron is able to tackle long-term sustainability (as an aggregate of time) with real-time efficiency and robustness within the same framework.

II.1 BACKGROUND WORK

This section provides an overview of two key concepts Edge Learning and Deep Reinforcement Learning (DRL). These frameworks support the proposed incentive-driven, Byzantine-resistant approach to edge learning. Edge learning addresses the privacy concerns inherent in distributed training scenarios by enabling collaborative model training without exposing raw data. Before introducing the decentralized scenario, we first describe the traditional centralized training approach for deep learning models. Given a deep learning model, let ω denote its parameters. The model's loss function is defined as $f(\omega, x_j, y_j)$, here x_j and y_j represent the data and label of a given sample j. For simplicity, we abbreviate $f(\omega, x_j, y_j)$ as $f_j(\omega)$. The objective of training is to update ω such that the average loss across all samples is minimized. Mathematically, this is expressed as:

$$\omega^* = \text{argmin}_\omega \frac{1}{|D|}\sum_{j\in D} f_j(\omega) \tag{1}$$

Here, D is the dataset. In decentralized edge learning, data is distributed among several edge nodes, each with its private dataset D_i. For edge node i, the loss function over its dataset is:

$$F_i(\omega) = \frac{1}{|D_i|}\sum_{j\in D_i} f_j(\omega) \tag{2}$$

The global loss function across all edge nodes is weighted by their dataset sizes and given as:

$$F(\omega) = \frac{\sum_{i=1}^{N} \delta_i |D_i| F_i(\omega)}{\sum_{i=1}^{N} \delta_i |D_i|} \tag{3}$$

Here, $\delta_i \in \{0,1\}$ is an indicator of whether edge node i participates in the training round. The goal of edge learning is to collaboratively train the global model by solving:

$$\omega^* = \text{argmin}_\omega F(\omega) \tag{4}$$

This optimization is achieved through distributed gradient descent. During each round, edge nodes perform σ epochs of local training on their private datasets. The local update for edge node i is:

$$\omega_{i,k} \leftarrow \omega_{i,k} - \eta \nabla_{\omega_{i,k}} F_i(\omega_{i,k}, x_j, y_j) \tag{5}$$

Here, η is the learning rate and k represents the current training round. After local training, edge nodes send their updated parameters $\omega_{i,k}^{\sigma}$ to the central server, which aggregates them to form the new global model:

$$\omega_{g,k+1} = \frac{\sum_{i=1}^{N} \delta_{i,k} \mid D_i \mid \omega_{i,k}^{\sigma}}{\sum_{i=1}^{N} \delta_{i,k} \mid D_i \mid} \quad (6)$$

This iterative process allows edge nodes to collaboratively train a global model without compromising data privacy. DRL combines RL with deep neural networks to optimize an agent's decision-making policy through interaction with an environment. The agent observes the current state s_t of the environment, selects an action a_t according to its policy $\pi(s_t)$ and receives a reward r_t. This interaction evolves the environment to the next state s_{t+1}. The goal of DRL is to maximize the expected cumulative reward:

$$R = \sum_{t=0}^{T} \gamma^t \, r(s_t, a_t) \quad (7)$$

Here, $\gamma \in [0,1]$ is a discount factor for future rewards. A well-known DRL algorithm is Proximal Policy Optimization (PPO), which operates within the Actor-Critic framework. This framework consists of two networks first is the Actor Network which generates actions based on the current policy and second is the Critic Network used to estimate the value of a state $V_{\pi}(s)$, representing the expected future rewards. The value function $V_{\pi}(s_l)$ is defined as:

$$V_{\pi}(s_l) = E\left[\sum_{t=1}^{\infty} \gamma^t \, r(s_t, a_t) \mid s_l, \pi \right] \quad (8)$$

Using the state value, the advantage of an action can be calculated as:

$$A_t(s_t, a_t) = r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t) \quad (9)$$

PPO modifies the traditional policy gradient loss function by introducing a clipping factor ò to stabilize updates:

$$L_{PPO}(\theta) = E\left[\min\left(r(\theta) A_t, \text{clip}(r(\theta), 1-ò, 1+ò) A_t \right) \right] \quad (10)$$

Here, $r(\theta)$ is the probability ratio between the new and old policies. DRL's ability to handle high-dimensional states and learn optimal policies through experience makes it a suitable choice for addressing complex problems like incentive-driven edge learning.

II.2 SYSTEM MODEL

The system model in this study is based on a parameter server architecture, incorporating multiple edge nodes. The model facilitates collaborative learning while it addresses computational and communication challenges. The architecture consists of Parameter Server which manages the global model and aggregates updates from edge nodes. Edge Nodes are used to perform local training using their data and contribute updates to the global model. The training occurs in rounds where edge nodes receive the global model, perform local updates and send results back to the server for aggregation. The next round begins only after all nodes finish their computations, introducing potential idle times. Each edge node's computation time depends on its local data size, CPU cycles per bit, and frequency:

$$T_{cmp,i,k} = \frac{\sigma c_i d_i}{\zeta_{i,k}} \quad (11)$$

Here, σ is the Number of local epochs, c_i is the CPU cycles per bit, d_i is the Local data size and $\zeta_{i,k}$ is the CPU frequency in round k. Communication time is determined by the global model size ξ and the uplink rate $R_{i,k}$:

$$T_{com,i,k} = \frac{\xi}{R_{i,k}} \quad (12)$$

The total time for edge node i is the sum of computation and communication times:

$$T_{i,k} = T_{cmp,i,k} + T_{com,i,k} \quad (13)$$

The overall round time is dictated by the slowest node:

$$T_{round} = \max_{i \in N} T_{i,k} \quad (14)$$

Faster nodes experience idle times as:

$$T_{idle,i,k} = T_{round} - T_{i,k} \quad (15)$$

Energy consumption comprises computational and communication components:

$$E_{cmp,i,k} = \sigma \alpha_i \, c_i d_i \, \zeta_{i,k}^2 \quad (16a)$$

$$E_{com,i,k} = ò_i \, T_{com,i,k} \quad (16b)$$

$$E_{i,k} = E_{cmp,i,k} + E_{com,i,k} \quad (16c)$$

Here, α_i and $ò_i$ are energy coefficients. Edge nodes are incentivized based on their contributions. For edge node i:

$$u_{i,k} = p_{i,k} \, \zeta_{i,k} - \lambda_e E_{i,k} \quad (17)$$

Here, $p_{i,k}$ is the price per CPU cycle and λ_e reflects the importance of energy costs. The system aims to optimize model accuracy while minimizing costs. Given a budget ψ_0, the remaining budget after k rounds is:

$$\psi_k = \psi_0 - \sum_{k=1}^{K} \sum_{i=1}^{N} \delta_{i,k} \, p_{i,k} \, \zeta_{i,k} \quad (18)$$

The utility function of the learning process is:

$$u = \lambda_a A(\omega_{g,K}) - \lambda_t \sum_{k=1}^{K} T_k + \psi_K \quad (19)$$

Here, $A(\omega_{g,K})$ denotes model accuracy, λ_a and λ_t are weight parameters.

II. 3 PROBLEM FORMATION AND ANALYSIS

This section elaborates on the strategies of the parameter server and edge nodes to optimize edge learning while ensuring robustness against Byzantine failures. The strategies aim to balance utility, computational efficiency and budget allocation under privacy constraints. The optimization problem is defined for both edge nodes and the parameter server. The key objective is to maximize the utility for both entities while maintaining efficiency in computation and communication. Each edge node i aims to maximize its utility $u_{i,k}$ by adjusting its computational contribution $\zeta_{i,k}$ during the k-th training round. The problem is formulated as:

$$\left.\begin{aligned} &OP_{i,k} : \max_{\zeta_{i,k}} u_{i,k}, \\ &\text{subject to} : \zeta_{i,k} \in (0, \zeta_{i,max}], \\ &u_{i,k} \geq 0 \end{aligned}\right\} \quad (21)$$

The utility $u_{i,k}$ is a function of the received bonus, computation energy, and communication energy is given in equation (17). The parameter server seeks to optimize its pricing strategy $p_{i,k}$ to maximize utility u. The optimization problem is given as:

$$\left.\begin{aligned} &OP_{PS} : \quad \max_{p_{i,k}} \quad u, \\ &\text{subject to} : \quad \sum_{k=1}^{K}\sum_{i=1}^{N} \delta_{i,k}\, p_{i,k}\, \zeta_{i,k} \leq \psi_0 \end{aligned}\right\} \quad (22)$$

Here, ψ_0 is the total budget and $\delta_{i,k}$ indicates whether edge node i participates in round k. The optimal strategies for edge nodes and the parameter server are analysed to address both honest and dishonest nodes. Honest nodes participate in training only if their utility is positive. The optimal CPU frequency for an honest edge node i is determined by:

$$\zeta_{i,k}^{*} = \frac{p_{i,k}}{2\lambda_e \sigma \alpha_i c_i d_i} \quad (23)$$

Here, α_i, c_i and d_i represent hardware coefficients and data size respectively. The maximum utility is:

$$v_{i,k}^{*} = \frac{p_{i,k}^2}{4\lambda_e \sigma \alpha_i c_i d_i} - \lambda_e \grave{o}_i T_{com,i,k} \quad (24)$$

Dishonest nodes include lazy and malicious nodes. Lazy nodes generate random updates without training, while malicious nodes send harmful updates. Their behaviour is modelled as:

$$\omega_{i,k}^{\sigma} = \omega_{i,k}^{0} + \phi \quad (25)$$

Here, ϕ is random noise. Their utility function is:

$$v_{i,k} = p_{i,k}\, \zeta_{i,k} - \lambda_e \grave{o}_i T_{com,i,k} \quad (26)$$

The parameter server aims to minimize idle time while ensuring efficient training. Idle time for node i is given in equation (15). The pricing strategy is optimized to satisfy:

$$T_{idle,k} = 0 \quad (27)$$

The incentive mechanism ensures participation and fairness. Edge nodes receive bonuses proportional to their contributions, and malicious nodes are identified and excluded. The utility of the parameter server over K rounds is expressed as:

$$v = \lambda_a A(\omega_{g,K}) - \lambda_t \sum_{k=1}^{K} T_k + \psi_K \quad (28)$$

Here, $A(\omega_{g,K})$ is the model accuracy and ψ_K is the remaining budget. The energy consumption of edge node i is given in (16a), (16b) and (16c). Time efficiency is optimized by aligning training times across nodes, ensuring:

$$r_{eff} = \frac{\sum_{i=1}^{N} T_{i,k}}{N T_{round}} \approx 1 \quad (29)$$

III. HIERARCHICAL REINFORCEMENT LEARNING DESIGN

Write HRL is employed to optimize the incentive mechanism in edge learning systems, ensuring robustness and efficiency. This section elaborates on the three-layer HRL framework, agent-specific designs and its advantages over traditional DRL. Edge learning environments present unique challenges. It includes dishonest edge nodes which has lazy and malicious nodes degrade system performance. Privacy Concerns is used as server access to edge node raw data and hardware details is restricted. Dynamic Environments are used as training data distributions and resource availability shift over time. To address these challenges, the HRL approach decomposes complex tasks into three sub-tasks handled by dedicated DRL agents Price volume determination, Bonus distribution and Edge node selection.

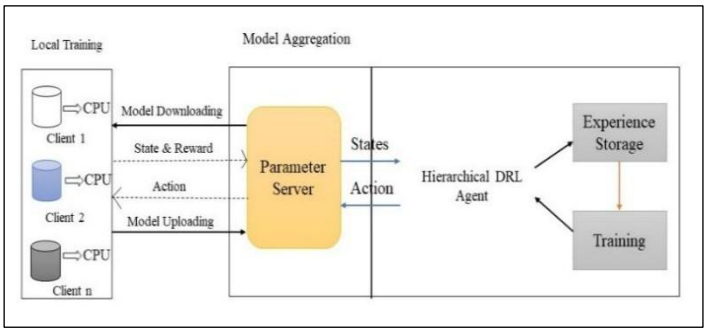

Figure 1: Edge Learning Operational Process.
Source: Authors, (2025).

The Figure 1 illustrates the architecture of the proposed framework for edge learning using a Hierarchical DRL Agent. It highlights three main components: Local Training, Model Aggregation and Hierarchical DRL Agent. In the Local Training phase, multiple edge nodes (referred to as clients) independently train local models using their private datasets. Each client performs computations locally to preserve privacy and avoid data transfer. After training, the clients upload their locally updated models to a central Parameter Server. The server manages model aggregation by collecting updates from all participating edge nodes and integrating them to update the global model. This global model is

then sent back to the clients through model downloading, ensuring continuous and iterative learning. The Parameter Server interacts directly with the Hierarchical DRL Agent for incentive mechanism optimization. The server provides real-time states and rewards to the DRL agent which evaluates the system's performance and sends back actions to guide the incentive mechanisms and edge node participation. The Hierarchical DRL Agent has two submodules Experience Storage, which records interaction history for further learning and Training which continuously improves the DRL policy based on past experiences. This design enables robust decision-making allowing the system to select reliable edge nodes and reward contributions. It also ensures efficient and fair resource utilization, resulting in improved global model accuracy and system performance.

The HRL framework comprises three agents, first is the Exterior Agent (Price Volume Calculator) which determines the overall budget allocation, second is the Middle Agent (Price Partition Distributor) which allocates the budget across edge nodes to minimize idle time and the third is the Inner Agent (Edge Node Selector) which selects participating edge nodes, expelling malicious ones. Each agent interacts with its environment and learns from experience tuples (s, a, r, s'), where s represents the state, a the action, r the reward and s' the subsequent state. The state, action, and reward functions for each agent are defined as follows. Exterior Agent is used as Price Volume Calculator which combines historical and current information.

$$s_k^E = \{\zeta_{k-L}, \ldots, \zeta_{k-1}, p_{k-L}, \ldots, p_{k-1}, T_{k-L}, \ldots, T_{k-1}, \psi_k, k\} \quad (30)$$

Here, ζ, p, and T denote CPU frequencies, pricing strategies, and training times, respectively, from the past L rounds.
Action: Sets the total budget:

$$a_k^E = \{p_{total,k}\} \quad (31)$$

Reward: Encourages long-term utility:

$$r_k^E = \lambda_a (A(\omega_k) - A(\omega_{k-1})) - \lambda_t T_k + \psi_k \quad (32)$$

Here, λ_a and λ_t are weight parameters for accuracy improvement and training time.
State: Takes the total budget as input:

$$s_k^M = \{p_{total,k}\} \quad (33)$$

Action: Allocates the budget proportionally among nodes:

$$a_k^M = \{\rho_{1,k}, \ldots, \rho_{N,k}\}, \quad \text{where} \sum_{i=1}^{N} \rho_{i,k} = 1. \quad (34)$$

Reward: Encourages time consistency:

$$r_k^M = -\sum_{i=1}^{N} T_{idle,i,k} \quad (35)$$

Here, $T_{idle,i,k}$ is the idle time of edge node i.
State: Incorporates model weights, price allocations, and remaining budget:

$$s_k^I = \{p_{1,k}, \ldots, p_{N,k}, \omega'_{1,k}, \ldots, \omega'_{N,k}, \omega'_g, \psi_k, k\} \quad (36)$$

Action: Determines the probability of node participation:

$$a_k^I = \{\nu_{1,k}, \ldots, \nu_{N,k}\}, \quad \nu_{i,k} \in [0,1] \quad (37)$$

Reward: Balances accuracy and budget consumption:

$$r_k^I = \lambda_a (A(\omega_{k+1}) - A(\omega_k)) - \sum_{i=1}^{N} \nu_{i,k}\, p_{i,k}\, \zeta_{i,k} \quad (38)$$

Each agent employs Proximal Policy Optimization (PPO) for training. The workflow includes-Initializing policy parameters, collecting experience tuples (s, a, r, s') and Updating actor and critic networks to optimize rewards. The PPO loss functions are:

$$L_{actor}(\theta) = E\left[\min\left(r(\theta) A_t, \text{clip}(r(\theta), 1-\dot{o}, 1+\dot{o}) A_t\right)\right] \quad (39)$$

$$L_{critic} = \sum (r_t + \gamma V(s_{t+1}) - V(s_t))^2 \quad (40)$$

Here, $r(\theta)$ is the probability ratio and A_t the advantage. Advantages of HRL include Task Decomposition which Simplifies optimization by dividing tasks into manageable sub-tasks, Enhanced Exploration expands the action space across three layers and Robustness adapts to dynamic environments and identifies malicious nodes.

IV. RESULTS AND DISCUSSIONS

It This section evaluates the effectiveness of the proposed HRL system, Chiron. The evaluation involves comprehensive experiments designed to assess global model accuracy, time efficiency, and system utility in diverse environments. Robustness against dishonest edge nodes and scalability under varying system conditions is also analysed. The experiments utilize real-world datasets and models under systematically controlled settings. The datasets include MNIST, Fashion-MNIST and CIFAR-10. Models of CNNs are MNIST and Fashion-MNIST which are two 5x5 convolutional layers followed by max pooling and CIFAR-10 of LeNet with three 5x5 convolutional layers followed by max pooling. The edge node settings are CPU cycle requirements for processing one bit are $c_i \sim U(1,2)$ GHz. Communication times $T_{com,i} \sim U(10,20)$ seconds. Implementation Framework has DRL agents which are implemented using the Tianshou framework with PPO. Actor and critic learning rates decay by 95% every 20 rounds. Three primary metrics are employed to assess performance, first is the Global Model Accuracy $A(\omega_g)$ which is evaluated using test datasets. Second is the Time Efficiency r_{eff} which measures valid training time utilization:

$$r_{eff} = \frac{\sum_{i=1}^{N} T_{i,k}}{N T_{round}} \quad (41)$$

Here, $T_{i,k}$ is the training time of edge node i in round k. Third is the server utility u which combines accuracy, time efficiency and budget usage is given in equation (19). Chiron's

performance is compared with a DRL-based baseline and a greedy approach. Server Utility is used as Chiron achieves higher utility across budgets. Chiron maintains accuracy without sacrificing performance for budget constraints. Chiron achieves nearly 100%-time efficiency, minimizing idle times. Experiments with 10 to 100 edge nodes highlight Chiron's scalability are Utility Trends where Chiron's utility remains above 25, while baselines drop significantly with more nodes, Time efficiency exceeds 80% preserving budget for additional rounds. Chiron's performance is evaluated in environments with malicious and lazy nodes. Malicious Nodes are Chiron outperforms baselines, maintaining accuracy and utility even when 70% of nodes are malicious. In Lazy Nodes Chiron resists budget wastage from lazy nodes, remaining robust up to 60% laziness. Mixed Environments has Chiron which excels with honest, malicious and lazy nodes achieving the best metrics. In Budget Utilization Chiron's hierarchical design prevents budget overuse, enabling sustainable training. Malicious Node Exclusion include the inner agent accurately filters harmful updates, preserving model quality. Long-Term Optimization has adaptive pricing strategies ensure robust utility growth. Chiron consistently outperforms baselines in accuracy, utility and robustness. The results validate the efficacy of hierarchical reinforcement learning in addressing the challenges of edge learning environments.

The Figure 2 shows the accuracy versus number of rounds for three methods: Chiron, DRL-Based, and Greedy. Chiron achieves the highest accuracy across all rounds, starting at 0.6 and reaching almost 0.95 at round 50. The DRL-Based method performs slightly lower, beginning at 0.6 and gradually improving to 0.85 at round 50. The Greedy method exhibits the slowest improvement, starting at 0.55 and reaching approximately 0.8 by round 50. From the graph, we see that Chiron consistently outperforms the other methods in accuracy throughout the training rounds. The difference between Chiron and the DRL-Based method is noticeable, especially after round 20, with a gap of about 0.05 to 0.1. Similarly, the gap between Chiron and the Greedy method widens with increasing rounds, highlighting Chiron's superior learning capability. DRL-Based maintains a steady and moderate improvement, while Greedy shows the least effective performance. This indicates that Chiron is more efficient in learning and adapting compared to the other approaches.

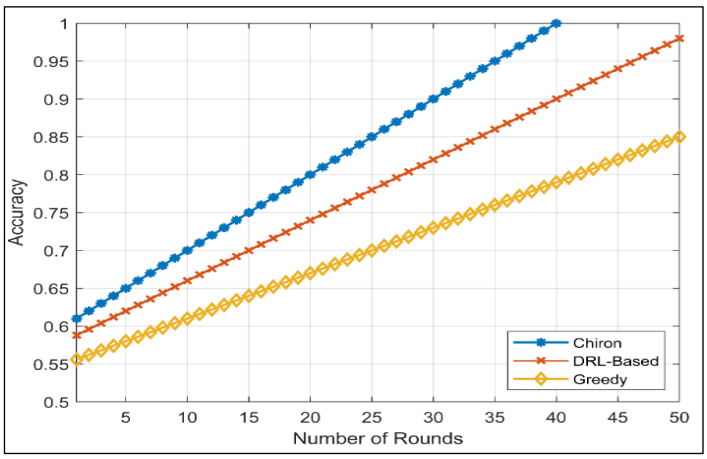

Figure 2: Accuracy versus number of rounds.
Source: Authors, (2025).

The Figure 3 shows the Performance under CIFAR-10 when varying the total budgets for Server Utility vs. Total Budget for three methods: Chiron, DRL-Based, and Greedy. Chiron consistently achieves the highest server utility at every budget level. At a budget of 50, its utility starts at around 30, increasing linearly to reach nearly 70 at a budget of 300. DRL-Based is average, you get around an initial 25 utility that grows to approximately 60 for about 300 in budget. The impulse approach is the least efficient, achieving a utility of 20 and approximately 55 at the highest budget. We observe that Chiron consistently outperforms all the other methods for all budget levels. As the budget is increased, the difference between Chiron and DRL-Based grows, resulting in a utility gap of 5 to 10 points—between candidate budgets that range from 0 to 60. Also, the ever-growing distance between Chiron and Greedy confirms Chiron's effective resource utilization. These two methods turned out to provide the slowest improvement, as we can see that the utility curve for the Greedy method also starts to flatten shortly after the budget has approached the 300 mark. The well-shown trends demonstrated are reflecting Chiron's better optimization strategies represented by DRL-Based and Greedy approaches.

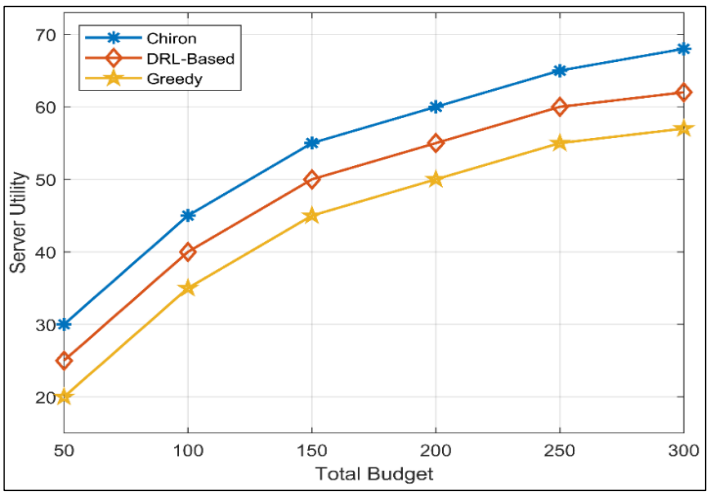

Figure 3: Performance under CIFAR-10 when varying the total budgets for server utility.
Source: Authors, (2025).

The Figure 4 illustrates the Performance under CIFAR-10 when varying the total budgets for Global Model Accuracy vs. Total Budget for three methods: Chiron, DRL-Based, and Greedy. Chiron achieves the highest accuracy at all budget levels. It starts at approximately 0.7 with a budget of 50 and steadily increases to about 0.87 at a budget of 300. The DRL-Based method performs moderately. It starts at 0.7 similar to Chiron, but grows more slowly, reaching around 0.82 at a budget of 300. The Greedy method has the lowest accuracy. It starts at 0.65, then moves up gradually to around 0.78 with the highest budget. Times mention that Chiron always outperforms the others in terms of accuracy, thus it better utilizes the budget. Over the time period considered, the worst gap between Chiron and DRL-Based is around 0.05 as the budget increases. The same can also be said for the gap between Chiron and Greedy, as throughout the tests there is a consistent gap of 0.1 to 0.12. These trends highlight Chiron's better optimization as well as Chiron's ability to improve the models performance across all budget levels while DRL-Based gets moderate gains and Greedy simply can't obtain high accuracy. In budget-constrained environments, maximizing resource-use efficiency is essential, and this only supports the importance of that.

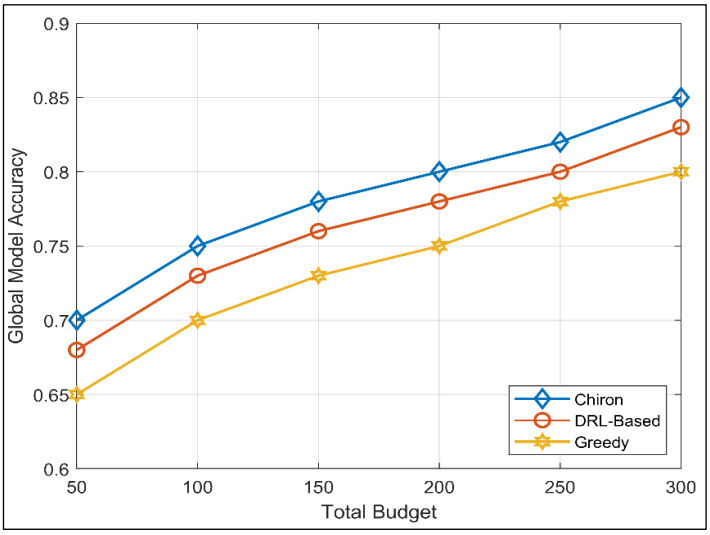

Figure 4: Performance under CIFAR-10 when varying the total budgets for Global Model Accuracy.
Source: Authors, (2025).

The Figure 5 shows Performance under CIFAR-10 when varying the total budgets for time efficiency versus total budget for three methods: Chiron, DRL-Based, and Greedy. Chiron starts at a time efficiency of around 0.85 for a budget of 50 and steadily increases to nearly 0.97 at a budget of 300. DRL-Based begins at approximately 0.8 and gradually rises to about 0.9 as the budget increases. Greedy starts at 0.75 and increases modestly to around 0.87 by the end of the budget range. The results highlight that Chiron outperforms the other methods in time efficiency across all budget levels. The difference between Chiron and DRL-Based is noticeable, with a gap of approximately 0.05 to 0.07 throughout. Similarly, Chiron maintains a significant advantage over Greedy, with a time efficiency difference of about 0.1 at most budget levels. The Greedy method shows the slowest improvement, indicating less efficient use of resources to minimize idle time. These trends underline Chiron's ability to achieve high time efficiency, particularly as the budget increases, making it the most effective method for optimizing resource utilization.

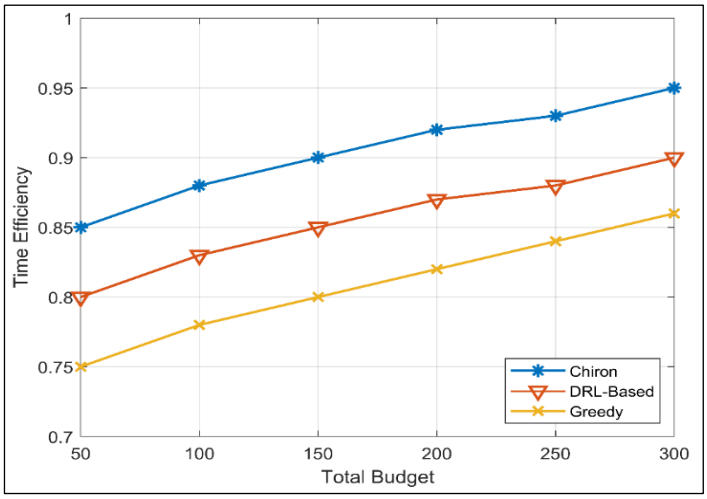

Figure 5: Performance under CIFAR-10 when varying the total budgets for Time Efficiency.
Source: Authors, (2025).

The Figure 6 shows accuracy versus percentage of malicious nodes for three methods: Chiron, DRL-Based, and Greedy. Chiron starts with an accuracy of 0.85 when there are no malicious nodes. Its accuracy gradually declines to approximately 0.65 as malicious nodes increase to 70%. The DRL-Based method begins at an accuracy of 0.8 and decreases more sharply, reaching around 0.6 at 70% malicious nodes. The Greedy method consistently delivers the worst performance, beginning at 0.75 and falling to 0.5 when 70% of the nodes are malicious. This show that under the all conditions, Chiron achieves the highest accuracy indicates its strong robustness to malicious node. Chiron's resultant accuracy in comparison to conjectured DRL-Based model varies between 0.05 to 0.1 throughout the range, which indicates Chiron's improved resistance against adversarial malicious behaviour. It can be seen that the Greedy method has the steepest decline in the accuracy, which shows that it is vulnerable to malicious nodes. With increasing number of malicious nodes, Chiron's optimization and fault tolerance enables it to out perform Greedy as the gap increases. This analysis highlights that Chiron demonstrates proficiency in robustness to noisy environments in comparison to the other methods.

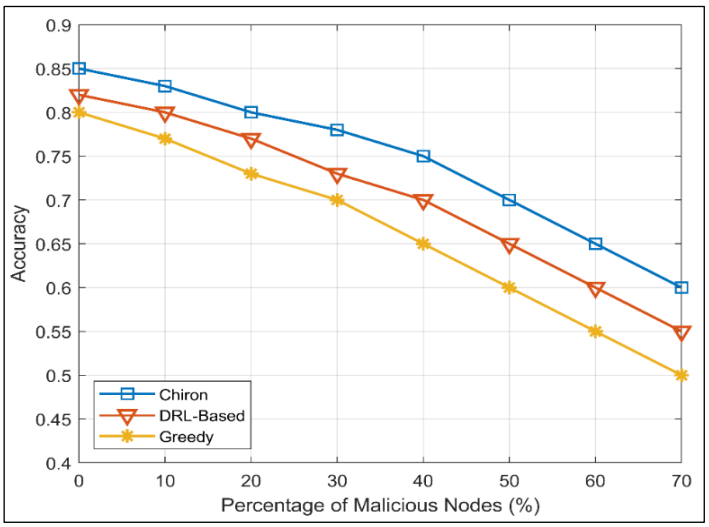

Figure 6: Accuracy versus Percentage of Malicious Nodes.
Source: Authors, (2025).

The Figure 7 depicts idle time versus number of edge nodes for three methods: Chiron, DRL-Based, and Greedy. Chiron has the lowest idle time across all edge node counts. Its idle time starts at about 10 seconds for 10 edge nodes and increases linearly to approximately 30 seconds for 100 edge nodes. DRL-Based exhibits moderate idle time, starting at 12 seconds for 10 nodes and growing steadily to about 45 seconds for 100 nodes. The Greedy method performs the worst, starting at 15 seconds and rising sharply to nearly 70 seconds for 100 edge nodes. This analysis highlights the relative shrug-ability of Chiron with respect to idle time with increasing edge nodes. This gap is even bigge when comparing Chiron and DRL-Based, culminating in a 15 seconds difference at 100 edge nodes. Likewise, the difference between Chiron and Greedy becomes. Similar to before, we have a huge gap of almost 40 seconds between Chiron and Greedy at 100 nodes. This shows that Chiron handles idle time efficiently even at a large scale and is more efficient and scalable than DRL-Based approaches and Greedy approaches. This highlights the benefits of Chiron in efficiently utilizing resources and reducing latency in distributed edge learning systems.

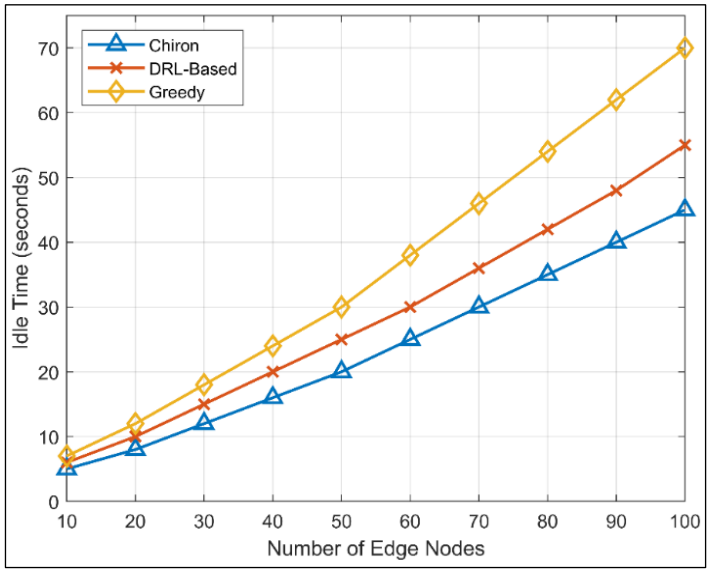

Figure 7: Ideal time versus Number of Edge Nodes.
Source: Authors, (2025).

The Figure 8 presents the convergence of DRL agents over episodes for three agents: the inner agent, middle agent, and exterior agent. In subplot (a), the Inner Agent begins with a reward close to 0 in the initial episodes. The reward steadily increases to approximately 20 by the 50th episode, showing a smooth convergence pattern. Subplot (b) depicts the Middle Agent, where the reward starts near 0 and rises steadily, converging at around 15 by episode 50. Subplot (c) shows the Exterior Agent, whose reward also starts near 0, increases steadily and converges at about 12 by the 50th episode. These plots demonstrate that all three agents successfully learn and converge within the 50 episodes. The inner agent achieves the highest reward, indicating it has the most significant impact or effectiveness in the system. The middle agent converges at a slightly lower reward, while the exterior agent achieves the lowest reward. This suggests a hierarchical distribution of complexity and contribution among the agents. The steady increase in rewards across episodes also reflects stable training and effective optimization strategies for all agents. Overall, the figure highlights that the DRL framework effectively enables each agent to achieve its respective learning objectives.

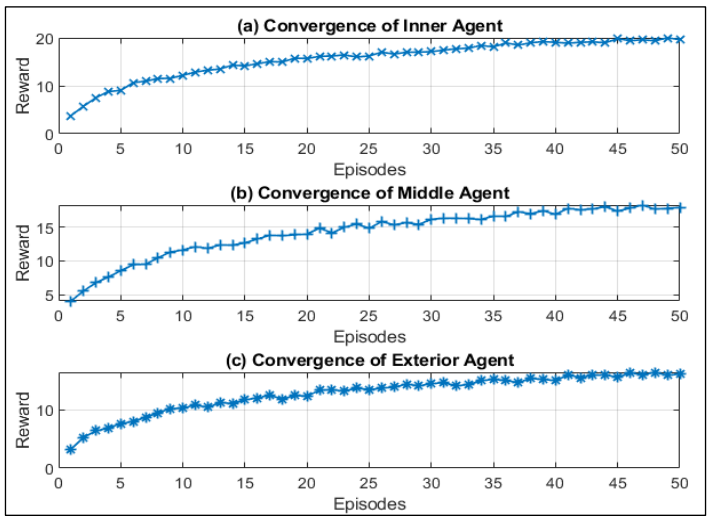

Figure 8: Convergence of DRL Agents.
Source: Authors, (2025).

The Figure 9 illustrates global model accuracy versus percentage of malicious nodes for three methods: Chiron, DRL-Based, and Greedy. Chiron starts at an accuracy of 0.85 when there are no malicious nodes and gradually decreases to approximately 0.7 as malicious nodes increase to 70%. DRL-Based begins with an accuracy of 0.8 and declines steadily, reaching around 0.6 at 70% malicious nodes. Greedy has the lowest accuracy, starting at 0.75 and dropping sharply to approximately 0.55 by the end. The results highlight Chiron's superior performance in maintaining accuracy under increasing malicious nodes. The difference between Chiron and DRL-Based grows as malicious nodes increase, with a gap of about 0.1 at higher percentages. Similarly, the gap between Chiron and Greedy is even more pronounced, emphasizing Chiron's robustness in hostile environments. DRL-Based shows moderate resistance, though Greedy performs poorly, especially as the percentage of malicious nodes grows. This comparison clearly demonstrates Chiron's effectiveness in mitigating the impact of malicious behaviour and maintaining reliable model accuracy.

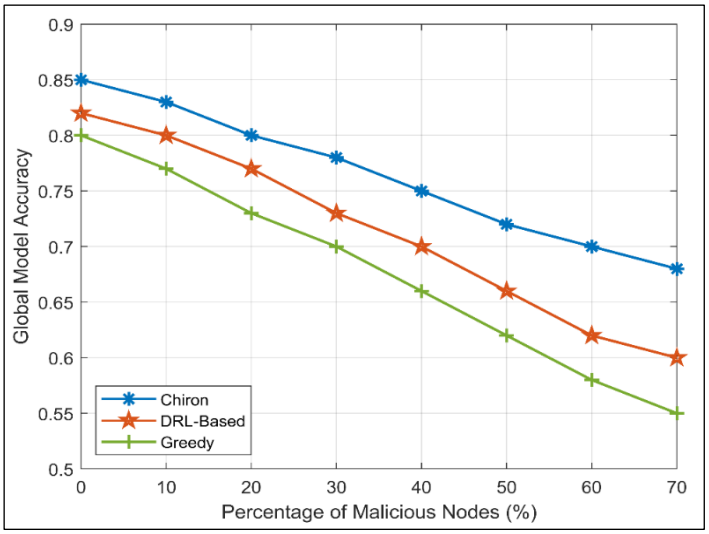

Figure 9: Global model accuracy versus percentage of malicious nodes in malicious environment.
Source: Authors, (2025).

The Figure 10 shows global model accuracy versus percentage of lazy nodes for three methods: Chiron, DRL-Based, and Greedy. Chiron starts with an accuracy of 0.85 at 0% lazy nodes and gradually decreases to around 0.7 when lazy nodes reach 70%. DRL-Based begins with an accuracy of approximately 0.8 and declines steadily, reaching 0.65 at 70% lazy nodes. Greedy starts at 0.75 and drops more sharply, reaching 0.6 at the maximum percentage of lazy nodes. This figure demonstrates Chiron's superior robustness against lazy nodes, maintaining higher accuracy across all percentages. The gap between Chiron and DRL-Based grows as lazy nodes increase, reaching about 0.05 at 70%. Similarly, the gap between Chiron and Greedy is even more significant, with a difference of approximately 0.1 at higher percentages. DRL-Based performs moderately but struggles more as the proportion of lazy nodes increases. Greedy shows the steepest decline, indicating its inability to handle high levels of laziness. These results emphasize Chiron's effectiveness in minimizing the impact of lazy nodes and maintaining reliable model accuracy in such scenarios.

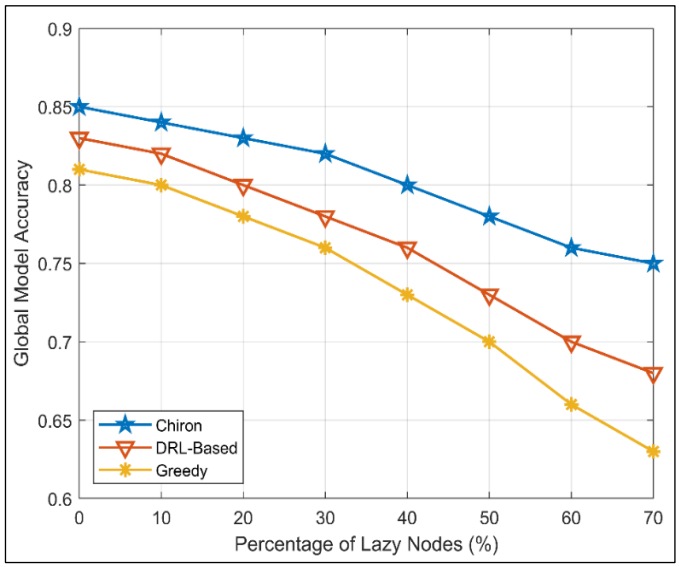

Figure 10: Global model accuracy versus percentage of lazy nodes in lazy environment.
Source: Authors, (2025).

The Figure 11 shows server utility versus number of edge nodes for three methods: Chiron, DRL-Based, and Greedy. Chiron achieves the highest server utility for all edge node counts. It starts at around 20 for 10 edge nodes and steadily increases to approximately 90 for 100 edge nodes. DRL-Based has moderate server utility. It begins at around 18 for 10 edge nodes and grows to approximately 80 at 100 edge nodes. The Greedy method has the lowest performance, starting at around 15 for 10 edge nodes and reaching only about 70 for 100 edge nodes. This graph highlights Chiron's superior ability to maximize server utility as the number of edge nodes increases. The gap between Chiron and DRL-Based widens as more edge nodes are added, reaching a difference of about 10 utility units at 100 edge nodes. Similarly, the difference between Chiron and Greedy grows significantly, reaching nearly 20 utility units at the same point. The Greedy method consistently lags behind, reflecting its inefficiency in resource allocation compared to the other methods. This analysis underscores Chiron's advantage in optimizing server utility, especially in large-scale distributed systems. It performs better in leveraging additional edge nodes, demonstrating its robustness and scalability.

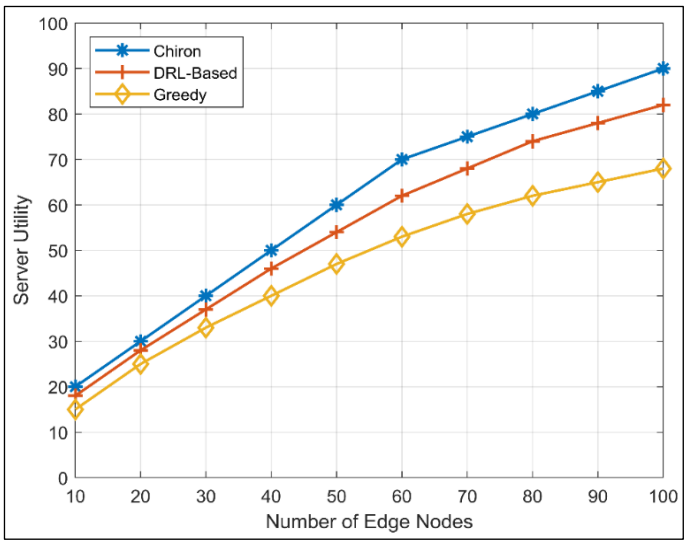

Figure 11: Server utility versus number of edge nodes
Source: Authors, (2025).

V. CONCLUSIONS

The research addresses the crucial problem of designing an efficient incentive mechanism for edge learning systems, emphasizing three primary objectives. The first is to incorporate model performance metrics into the optimization process to guarantee high-quality outputs. The second is ensuring system sustainability by implementing long-term optimization strategies. Finally, it focuses on identifying and mitigating the effects of malicious and lazy edge nodes to enhance the system's robustness. To achieve these objectives, the paper introduces a HRL framework comprising three distinct layers.

This design decomposes the complex task into three sub-tasks: pricing determination for long-term utility maximization, bonus distribution for short-term optimization and edge node selection for excluding underperforming or malicious participants. This three-tiered approach ensures that the system remains efficient, adaptive and robust against dishonest behaviours in the dynamic edge learning environment proposed method demonstrates substantial improvements compared to state-of-the-art mechanisms.

Experimental evaluations on real-world datasets, like CIFAR-10, reveal an increase in system accuracy and total utility by 14.96% and 12.66%, respectively. By expelling lazy and malicious nodes, the system maintains higher reliability and performance consistency. Additionally, the framework ensures effective budget utilization and scalability, making it suitable for future network scenarios like beyond 5G (B5G).

The researchers emphasize the importance of continuing exploration in areas like advanced node behaviour analysis, energy-aware strategies and integration into emerging network architectures. This secures the practical implications of the HRL-based incentive mechanism, which balances efficiency, robustness and sustainability in edge learning systems. By addressing existing limitations, it paves the way for more reliable and effective distributed machine learning frameworks.

VI. AUTHOR'S CONTRIBUTION

Conceptualization: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi
Methodology: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.
Investigation: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.
Discussion of results: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.
Writing – Original Draft: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.
Writing – Review and Editing: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.
Resources: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.
Supervision: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.
Approval of the final text: Gowtham R, Vatsala Anand, Yadati Vijaya Suresh, Kasetty Lakshmi Narasimha, R. Anil Kumar, V.Saraswathi.

VII. REFERENCES

[1] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, “Machine learning at the network edge: A survey,” ACM Computing Surveys (CSUR), vol. 54, no. 8, pp. 1–37, 2021.

[2] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, J. Yang, L. Zhang, K. Cheng, and X. Xiao, “Fault management in software-defined networking: A survey,” IEEE Communications Surveys & Tutorials, vol. 21, no. 1,pp. 349–392, 2018.

[3] Y. Liu et al., “Optimizations on resource constrained federated learning with diverse clients,” 2024.

[4] Y. Sun, H. Ochiai, and H. Esaki, “Decentralized deep learning for multi-access edge computing: A survey on communication efficiency and trust worthiness,” IEEE Transactions on Artificial Intelligence, vol. 3, no. 6, pp. 963–972, 2021.

[5] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, “Federated learning in mobile edge networks: A comprehensive survey,” IEEE communications surveys & tutorials, vol. 22, no. 3, pp. 2031–2063, 2020.

[6] F. Akhtar, S. A. Lodhi, S. S. Khan, and F. Sarwar, “Incorporating permaculture and strategic management for sustainable ecological resource management,” Journal of environmental management, vol. 179, pp. 31–37, 2016.

[7] Y.-J. Gong, J. Zhang, H. S.-H. Chung, W.-N. Chen, Z.-H. Zhan, Y. Li, and Y.-H. Shi, “An efficient resource allocation scheme using particle swarm optimization,” IEEE Transactions on Evolutionary Computation, vol. 16,no. 6, pp. 801–816, 2012.

[8] M. A. Khan, Shalu, Q. N. Naveed, A. Lasisi, S. Kaushik, and S. Kumar, “A multi-layered assessment system for trustworthiness enhancement and reliability for industrial wireless sensor networks,” Wireless Personal Communications, vol. 137, no. 4, pp. 1997–2036, 2024.

[9] Y. Tang, “Transport efficiency increase for axfood’s transport carriers in central gothenburg,” rapport nr.: Masters Thesis, no. 2003, 2004.

[10] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, “Bubbles of trust: A decentralized blockchain-based authentication system for iot,” Computers & Security, vol. 78, pp. 126–142, 2018.

[11] Z. Wang, X. Chen, and Z. Zhou, "Joint Resource Allocation and Incentive Design for Federated Learning in Wireless Networks," IEEE Transactions on Wireless Communications, vol. 20, no. 3, pp. 2121–2132, Mar. 2021.

[12] J. Zhang, L. Liu, and H. Tang, "Contract-Based Incentive Mechanisms for Federated Learning," IEEE Transactions on Network Science and Engineering, vol. 7, no. 3, pp. 1227–1240, Jul.–Sep. 2020.

[13] Z. Tang, Y. Zhao, and R. Liu, "Auction-Based Federated Learning Incentive Mechanisms," IEEE Transactions on Mobile Computing, vol. 19, no. 11, pp. 2598–2612, Nov. 2020.

[14] L. Li, J. Wang, and X. Zhang, "Efficient Resource Allocation in Heterogeneous Edge Networks," IEEE Transactions on Cloud Computing, vol. 8, no. 3, pp. 853–865, Jul.–Sep. 2020.

[15] J. Huang, S. Zhou, and Z. Niu, "Optimized Task Scheduling for Edge Computing," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 7, pp. 1604–1617, Jul. 2018.

[16] X. Chen, L. Pu, and Z. Wang, "Resource Optimization in Federated Learning Systems," IEEE Transactions on Mobile Computing, vol. 22, no. 5, pp. 781–792, May 2023.

[17] Y. Liu, S. Guo, and Y. Zhan, "Chiron: Robustness-Aware Incentive Mechanism for Edge Learning," IEEE Transactions on Mobile Computing, vol. 23, no. 8, pp. 8508–8518, Aug. 2024.

[18] Z. Tang, Y. Zhao, and R. Liu, "Resource Management for Federated Edge Learning," IEEE Internet of Things Journal, vol. 9, no. 3, pp. 2271–2284, Mar. 2022.

[19] P. Blanchard, E. M. Lupu, and R. Shen, "Byzantine-Robust Federated Learning," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 3, pp. 1012–1024, Mar. 2020.

[20] J. Xie, L. Yang, and Y. Chen, "Reputation-Based Robustness in Federated Learning," IEEE Transactions on Communications, vol. 69, no. 10, pp. 6712–6721, Oct. 2021.

[21] J. Kang, R. Yu, and Z. Xie, "Blockchain for Secure and Efficient Edge Learning," IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1141–1164, Second Quarter 2020.

[22] C. Fung, C. Zhang, and S. S. Lee, "Robust Federated Learning with Adaptive Aggregation," IEEE Transactions on Artificial Intelligence, vol. 2, no. 2, pp. 132–144, Apr. 2021.

[23] A. Vezhnevets, S. Osindero, and T. Schaul, "FeUdal Networks for Hierarchical Reinforcement Learning," Proceedings of the International Conference on Machine Learning, 2017, pp. 3540–3549.

[24] Z. Tang, R. Liu, and Y. Zhao, "Task Allocation in Distributed Systems Using HRL," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 9, pp. 5031–5042, Sep. 2021.

[25] X. Chen, L. Pu, and Z. Niu, "HRL for Federated Learning Optimization," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 4, pp. 859–871, Apr. 2022.