Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Semantic Task Offloading for IoT in Heterogeneous Meta-Computing Environments: A DRL Approach with LLM-Enhanced Contextual Awareness

Abdessalam Messiaid[1*], Mohamed A Mahboub[2], Imad kemerchou[3], Said Boumara[4], Kafi Redouane[5], Narimene Mimoune[6], Sahraoui Dhelim[7]

1 Computer Science Department, University of Kasdi Merbah, 30000, Ouargla, Algeria (e-mail:messiaid.abdessalam@univ-ouargla.dz)
2 Institut de Technologie, University of Kasdi Merbah, 30000, Ouargla, Algeria
3Department of Mechanical Engineering, University of Kasdi Merbah, 30000, Ouargla, Algeria
4 Space Telecommunications Exploitation Centre, Algerian Space Agency, Algiers, Algeria
5 Department of Electronics, University of Kasdi Merbah, 30000, Ouargla, Algeria
6 SISCOM Laboratory, University of Constantine 1, Algeria
7 School of Computing, Dublin City University, Ireland
Corresponding author: Abdessalam Messiaid1 (messiaid.abdessalam@univ-ouargla.dz)

ARTICLE INFO

Received: 01 Apr 2026
Revised: 18 May 2023
Accepted:28 May 2026
Published: 02 June 2026

ABSTRACT

The continuous increase in Internet of Things (IoT) devices in heterogeneous meta-computing environments has led to increased interest in more efficient task offloading methods to mitigate the effects of resource constraints in processing, memory, and battery life. Existing offloading methods often lack semantical offloading decisions and do not consider the heterogeneous and dynamic nature of these environments, leading to reduced results. This paper presents a semantic representation-enhanced DRL framework that integrates Deep Reinforcement Learning (DRL) and Large Language Models (LLMs) to enable more efficient task offloading. An LLM acts as a semantic encoder, converting raw task descriptions into embedding vectors that capture task intent, dependencies, and computational requirements. Based on this description, a DRL-based agent makes the best offloading decision that can be made given the current state of the system. This paper reports a 37.2% reduction in latency and a 23.8% reduction in energy consumption compared to DQN, along with a 17.4% improvement in task success rate over QMIX under high-load conditions.

Keywords: Internet of Things, Task Offloading, Deep Reinforcement Learning, Large Language Models, Heterogeneous Meta-Computing.

I. INTRODUCTION

The Internet of Things (IoT) has enabled groundbreaking developments in various areas, including healthcare, smart cities, agriculture, and industrial automation. It has transformed the way devices communicate with each other. However, the limited resources of IoT devices, such as processing power, memory and battery life, hinder their ability

Research Article

to perform complex tasks. This highlights the critical need for efficient task execution to ensure optimal resource utilization and timely data processing and retrieval.

IoT systems can enhance user experience, reduce latency, and improve battery life efficiency by utilizing task offloading methods to more capable resources, such as edge, fog, or cloud servers. For example, a wearable ECG monitor that tracks the body's vital signs can offload data analysis tasks to a close edge node, ensuring that alerts are sent quickly without draining the device's battery. Even with research progress, task offloading in IoT systems remains an involved challenge, particularly in heterogeneous meta-computing environments where resource and capacity availability vary dynamically.

The majority of existing offloading approaches rely heuristics or rules-based methods that are unable to adapt to heterogeneous meta-computing environments and can cause offloading inaccuracy and inefficiency. They also rarely take into account semantic contextual feature mapping, such as task urgency, resource workload, or even device power levels. Therefore, these methods can lead to inefficient resource utilization and poor offloading performance. For example, these methods may lead to incorrectly assigning tasks to resources. This guides the need to improve the integration of task offloading methods and make them more semantically rich and context-aware to match the complexity of IoT systems.

Deep Reinforcement Learning (DRL) has potential for dynamic task offloading. However, techniques that rely entirely on DRL falter because they lack a mechanism to semantically augment their state space with task-relevant features. They simplify each task into a vague workload, characterized solely by the amount of data and the required computation effort. Additionally, they require a long training period to discover optimal policies, which correspondingly leads to prolonged convergence times.

Large Language Models (LLMs) demonstrate capabilities in natural language feature extraction and representation learning. However, the use of LLMs for direct task offloading is limited because of their steep computational, slow inference, and high memory requirements. These challenges create a key research question: how can LLMs improve DRL-driven offloading while remaining practical for edge deployment?

To overcome these limitations, we propose a semantic representation framework that uses a lightweight LLM (NeuroBERT) integrated as a semantic encoder, with Proximal Policy Optimization (PPO) for IoT task offloading.

In contrast to previous research that employs LLMs as conversational reasoning agents, our framework uses the LLM as an encoder that transforms raw IoT task descriptions into compact embedding vectors, enhancing the state space of the PPO agent.

The remainder of this paper is structured as follows. In Section II, we analyse prior work on traditional offloading approaches, semantic computing, and DRL-based methods. Section III explains the system in terms of a heterogeneous meta-computing architecture, task and resource models, and latency, energy formulations. In Section IV, we introduce the Semantic Representation Framework, focusing on an LLM-based semantic encoder, a semantically-constrained PPO agent, and the integrated algorithm. Section V provides the experimental setup, results, ablation study, and discussion. Finally, Section VI concludes the paper, summarising the contribution, stating the constraints of the current work, and listing areas for further research.

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

II. RELATED WORKS

The latest developments in Internet of Things (IoT) technology have highlighted the importance of task offloading strategies in heterogeneous meta-computing environments. This section analyzes existing strategies, ranging from traditional methods to current semantic and hybrid solutions.

2.1 Traditional Approaches to IoT Task Offloading

Recent developments in the array of technologies associated with the Internet of Things (IoT) highlight the need for refined methods in task offloading for heterogeneous meta-computing environments. Traditionally, several approaches have been used to address this issue, including rule-based methods, mathematical optimization methods, and classical machine learning methods. While rule-based methods such as the Fuzzy Inference Rule-Based Task Offloading Model (FI-RBTOM) proposed by Ibrahim et al. [1] and the ANFIS model developed by Ibrahim et al. [2] demonstrate a high degree of accuracy and perform well in the scenarios for which they are provided. However, they have a major issue of being unable to adjust to the flexibility and variability associated with Internet of Things (IoT) environments. Manually defining rules for each case is a tedious and inefficient way of handling such systems. In addition, these methods do not respond well to unexpected scenarios, which reduces their effectiveness in managing complex systems.

In optimization techniques, several approaches have been proposed to determine optimal satisfactory solutions to efficient task offloading methods. For example, Bebortta et al. [3], for dynamic integer linear programming (ILP) and Zaidi et al. [4] for particle swarm optimization (PSO). Such approaches rely on system model and system parameter efficiency, which is hard to model in dynamic environments and may result in high computational demands. In heterogeneous meta-computing environments, it is challenging for these methods to efficiently task offloading models in complex and dynamic large-scale IoT systems [5], [6]. Traditional machine learning techniques have also been applied to intelligent task-offloading decision-making. Examples include Support Vector Machines (SVM), Random Forest classifiers, and decision tree-based approaches [7], [8]. These techniques can capture and model complex, non-linear relationships, assisting in decision-making and trade-offs related to offloading latency and resource-constrained situations. They have also been modified for adjacent areas, such as load balancing, resource allocation, and secure offloading in the fog-cloud domain [9]. However, the need for high-quality labeled datasets and sensitivity to concept drift constrain the applicability of these approaches in the dynamic IoT ecosystem. Garai et al. [10] state that the need for large quantities of accurately labeled data is a primary reason these methods do not apply to dynamic environments. The requirements for these models to capture long-term dependencies and to make a sequence of decisions are further evidence that these methods are limited for this task [11], [12].

In contrast to deep reinforcement learning approaches that utilize Markov Decision Processes (MDP) to model sequential decisions, classical machine learning techniques, such as SVM or Random Forest, lack an architecture that captures the necessary temporal relationships and the rewarding structure critical for formulating optimal long-term offloading strategies [11], [12].

2.3 Semantic Approaches to IoT Task Offloading

The advent of semantic computing has significantly transformed IoT task offloading by integrating context awareness and intelligent reasoning through the use of knowledge representation. In the early stages of this computing

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

paradigm, ontology-based semantic models were used to improve resource management in edge computing, as noted by Singh et [13]. These models enable a richer contextual understanding of tasks and resources, facilitating more informed offloading decisions. Nevertheless, as noted by Ranpara et al. [14], ontology-based models have considerable difficulty when it comes to the issues surrounding system maintenance and scalability. In dynamic and heterogeneous IoT environments, the processes involved in the manual development and maintenance of complex ontologies are often time-consuming and financially prohibitive, creating the problems that prevent widespread use of these models in heterogeneous meta-computing environments.

More recently, advances in Large Language Models (LLMs) have enabled semantic feature extraction for IoT systems. For instance, Friha et al. [15] studied LLMs within edge intelligence frameworks, highlighting their potential for extracting contextual features from task descriptions in resource-constrained environments. However, using LLMs directly for real-time offloading remains challenging due to inference latency and memory constraints.

LLM-based solutions overcome the shortcomings of conventional methods by extracting task-relevant features from contextual information without requiring manually crafted rules. Additionally, Ren et al. [16] examined LLMs for mobile edge computing, using retrieval-augmented generation (RAG) to improve task offloading and resource allocation through context-aware feature extraction. Although there is potential for using LLMs to improve edge computing frameworks for mobile semantic tasks, increased latency, energy consumption, and data privacy concerns remain obstacles to deploying LLMs in resource-constrained IoT environments.

2.4 Deep Reinforcement Learning for Task Offloading

Deep Reinforcement Learning (DRL) has emerged as a promising solution for optimizing task offloading decisions, particularly in complex and dynamic IoT environments. For example, Chen et al. [17], developed a deep reinforcement learning model for task offloading in mobile edge computing (MEC) and heterogeneous tasks. In addition, Aliyu et al. [18], used deep reinforcement learning approaches targeted at latency-optimal task placement through intelligent orchestration methods in in-network computing-supported MEC, demonstrating the adaptive methods for resource allocation in real time. That said, there are still a few substantial drawbacks to entirely DRL-based solutions. According to Dou et al. [19], some findings using deep reinforcement learning for achieving generalized intelligence within IoT systems have some significant limitations. Specifically, the authors state that the inherent complexity of DRL-based methods makes it challenging to achieve efficient training and adaptation in dynamic environments. These limitations indicate that, despite the power of DRL in specific optimization tasks, its lack of inherent semantic understanding and high computational cost of training can hinder its scalability and generalizability in large and heterogeneous IoT networks. In this paper, we integrate LLM into the DRL model. By fully leveraging the characteristics of DRL and LLM, the contextual and semantic understanding capabilities of LLM can be utilized to enhance the decision-making process of the DRL agent. This enables the DRL agent to make more informed decisions about which tasks to offload.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we give an overview of the system model and then define the latency model and energy consumption model. On this basis, the objective function underlying the research is formulated and discussed.

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.1 System model

In this work, we consider a heterogeneous meta-computing environment structured as a multi-tier architecture M= {D, E, F, C} composed of four computational layers: IoT devices (D), edge nodes (E), fog servers (F), and cloud platforms (C), as shown in Figure. 1. This environment is designed to support the dynamic and diverse computational requirements of modern IoT applications such as industrial automation, healthcare monitoring, and smart city analytics. The core functionality involves enabling resource-constrained IoT devices to offload computation-intensive or latency-sensitive tasks to more capable nodes within the edge, fog, or cloud tiers. A critical aspect of this environment is its heterogeneity, which manifests not only across the different tiers (edge vs. fog vs. cloud), but also within each tier, encompassing devices and servers with varying computational power, memory capacities, energy constraints, and network connectivity.

The first layer (IoT Device), this layer contains a set of heterogeneous IoT devices (camera, sensor, ...etc), denoted by D. Each device d acts as a source of computational tasks. These devices are distinguished by limited resources, such as processing power (CPU frequency f), available memory, and finite battery life, making it difficult to run complex tasks locally, which often have stringent latency requirements or high computational demands.

The second layer (Edge nodes) consists of a set of Edge nodes (e.g., Raspberry Pi, clusters, etc.), denoted by E, strategically positioned in close physical proximity to IoT devices, frequently co-located with network access points such as base stations or Wi-Fi routers. Compared to IoT devices, the edge node comprises moderate computing capabilities, including processor power f and memory resources.

The Thired layer, The Fog layer comprises a set of Fog servers (e.g., GPU enabled microservers), situated between the edge and cloud layers. Each fog server F often provides higher processing power and storage capacity than edge nodes. Although these fog servers could introduce higher communication latency than edge nodes due to their potentially distant locations from IoT devices, they play an essential intermediate layer role for distributed processing. These servers can handle tasks that exceed the capability of edge nodes or require cooperative computations between multiple servers.

The fourth layer (Cloud Layer), at the top of the architecture is a centralized Cloud data center, denoted as C. The cloud provides nearly limitless processing resources and storage capacity. It is suitable for handling very complex and resource-intensive tasks such as large-scale machine learning model training, massive data analysis, and tasks without critical latency constraints. Offloading duties to the cloud, on the other hand, usually results in the largest communication latency due to the significant geographical distance.

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

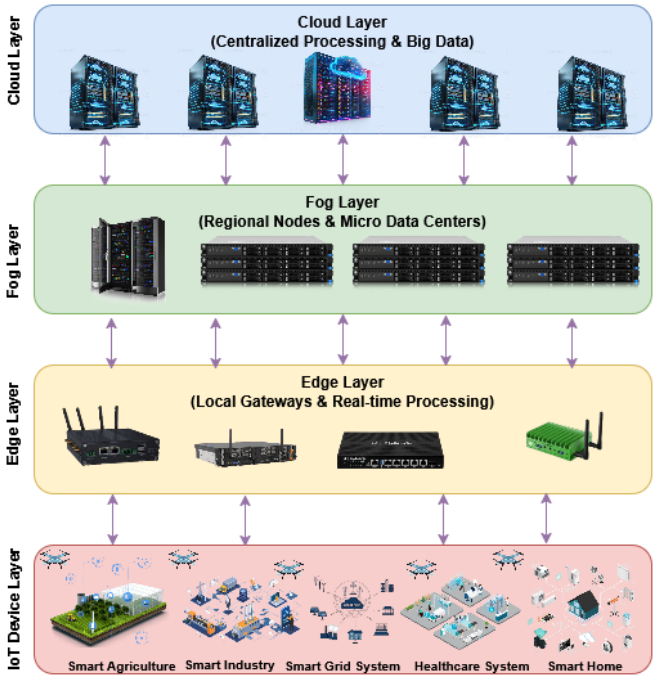

Figure 1: Meta-computing Architecture for IoT Systems

3.2 Task model

Each IoT task $T = \{s_i, l_i, c_i, ri\}$ generated by the device di can be executed locally or offloaded to higher layers, where: s_i input data size (bits), l_i latency tolerance (s), ci computational complexity (CPU cycles), ri required resource type (CPU/GPU/memory).

3.3 Resource model

The resources available at node n at time t are: $Rn(t) = \{fn(t), mn(t), bn(t), En(t)\}$ where: fn CPU frequency (cycles/s), mn GPU/memory availability, bn network bandwidth to the adjacent layer, En residual energy (J).

3.4 Latency and energy consumption model

Latency is a critical factor in IoT systems, directly impacting performance and user experience. In a heterogeneous meta-computing environment, the total latency of a task depends on where it is executed: locally on the IoT device, on an edge/fog node, or on a cloud server. We will detail the latency and energy models for each of these scenarios. Since our research focuses on semantic task offloading, and transmission and computation times are the primary causes of latency, we assume that the queueing delay is minimal.

3.5 Local Execution

When a task is executed locally on the IoT device, the latency is primarily determined by the task's processing time on the device's CPU. The local processing time $L_{d,Ti}^{loc}$ for a task T_i on a device d is given by the following equation [20]:

$$L_{d,Ti}^{loc} = \frac{W_{Ti}}{f_d} \quad (1)$$

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

where W_{Ti} represents the total workload of task T_i, and f_d is the CPU frequency of the device d to process a task. Next, we define E^{loc}_{d,T_i} as the energy consumption during the local execution of the task [21]

$$E^{loc}_{d,T_i} = \kappa W_{T_i}(f_d)^2 \qquad (2)$$

where κ represents the energy efficiency of the device d.

3.6 Edge\Fog Execution

Several tasks require moderate computing capabilities and sensitive time latency, such as real-time analysis of ECG/EEG signals. Edge nodes are considered the best choice for these tasks due to their proximity to IoT devices, which mitigates the limitations of resource-constrained IoT devices. The total latency L^E_{d,T_i} ,of the task T_i, when to be offloaded, is composed of the transmission $Trans^E_{d,Ti}$, reception $Recp^E_{d,Ti}$, and processing time $Proc^E_{d,Ti}$, which the transmission time, is the time required to send the task from the IoT device to the edge nodes, the reception time, is the return of the processed results to the device from the edge node, and the processing time, which depends on the processing speed of the edge nodes' CPUs. In this work, we ignore the reception time when offloading to fog/edge nodes because the processed result size is usually considerably lower than the input data, and its impact on the total latency is minimal. [22]. These expressions are explained in equations (3) , (4) and (5) [23]

$$Trans^E_{d,Ti} = \frac{D_{Ti}}{B_{u2E}}, \qquad (3)$$

$$Proc^E_{d,Ti} = \frac{W_{Ti}}{f_j}, \qquad (4)$$

$$L^E_{d,T_i} = Trans^E_{d,Ti} + Proc^E_{d,Ti} \qquad (5)$$

where B_{u2E} is the bandwidth available for data transmission between the IoT device and the edge nodes, f_j is the variable represents the CPU frequency of the edge node (j), DTi is the size of the task that the edge node will process. Similarly, in the case of offloading a task to the edge node, the total latency L^F_{d,T_i} of the task T_i when it is to be offloaded to the fog node is:

$$L^F_{d,T_i} = Trans^F_{d,Ti} + Proc^F_{d,Ti} \qquad (6)$$

Moreover, the total energy consumption when a task is affloaded to the edge node $E^E_{d,Ti}$ is the sum of the energy consumed in the input data transmission phase EE tx (that is, the energy used by the IoT device to send input data to the edge node) and the results reception phase EE rx (that is, the energy used by the device to receive the computation results from the edge node), expressed (7)-(9). [23]

$$E^E_{tx} = P_{tx} \times \frac{D_{in}}{R_{up}} \qquad (7)$$

$$E^E_{rx} = P_{rx} \times \frac{D_{out}}{R_{down}} \qquad (8)$$

$$E^E_{d,Ti} = E^E_{tx} + E^E_{rx} \qquad (9)$$

where $P_{tx}$and P_{rx} are the power consumption, expressed in watts, of the device's transmitter in (uplink) transmission mode and receiver in (downlink) mode, respectively. $D_{in}$and D_{out} denote the input and output data sizes, respectively, measured in bits. $R_{up}$and R_{down} denote the (uplink) and (downlink) data rates of the device, respectively, measured in bits per second (bps). In this paper, we assume that the energy consumed by the device while remaining idle during the server's (edge/fog/cloud) computation time and the result transmission phase is negligible.

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Similarly, in the case of offloading a task to the edge node, the total energy consumption $E^{E}_{d,Ti}$ when a task is affloaded to the fog node is:

$$E^{F}_{d,Ti} = E^{F}_{tx} + E^{F}_{rx} \quad (10)$$

3.7 Cloud Execution

Given that fog devices have computational limitations, a possible alternative is to offload tasks to the cloud layer. Cloud servers are a class of servers known for their huge resources and high efficiency, which can support a variety of workloads. The total latency of the task is indicated by L^{C}_{d,T_i} ,which is a combination of three factors: transmission time $Trans^{c}_{d,Ti}$, reception time $Recp^{c}_{d,Ti}$ and processing time $Proc^{c}_{d,Ti}$. It is explained as follows [23]:

$$Trans^{c}_{d,Ti} = \frac{D_{Ti}}{B_{u2c}} \quad (11)$$

$$Recp^{c}_{d,Ti} = \frac{D_{Ti}}{B_{c2u}} \quad (12)$$

$$Proc^{c}_{d,Ti} = \frac{W_{Ti}}{f_c} \quad (13)$$

$$L^{C}_{d,T_i} = Trans^{C}_{d,Ti} + Proc^{C}_{d,Ti} + Recp^{c}_{d,Ti} \quad (14)$$

Similarly to the case of offloading a task to the fog node, the total energy consumption when a task is offloaded to the cloud server is expressed as $E^{C}_{d,Ti}$. It is explained as follows:

$$E^{C}_{d,Ti} = E^{C}_{tx} + E^{C}_{rx} \quad (15)$$

3.8 Problem formulation

In this work, our objective is to propose an efficient task offloading strategy to minimize overall system latency and energy consumption through optimal offloading decisions. We formulate the problem as a combinatorial optimization problem with discrete variables and complex constraints. We define each task offloading decision as a binary variable xij (t) at time t, which determines whether data from IoT device di is offloaded to edge processing (E), fog processing (F), or cloud processing (C). Based on the above considerations, the optimization problem can be formulated as:

$$\min Z = \sum_{i=1}^{N} \sum_{j\in\{E,F,C\}} x_{ij}(t) \times \left(L^{j}_{d,Ti} + E^{j}_{d,Ti}\right) \quad (16)$$

Subject to:

$$C1: x_{ij}(t) = \begin{cases} 1, & \text{if } T_i \text{is offloaded to } j \text{ at time } t, j \in \{E,F,C\} \\ 0, & \text{otherwise} \end{cases}$$

$$C2: \quad \sum_{j\in\{E,F,C\}} x_{ij} = 1\,, \qquad \forall i \in N.$$

$$C3: \quad \sum_{i=1}^{N} x_{ij}(t) \times L^{j}_{d,Ti} \le \tau_{\text{critical}}\,, \qquad \forall j \in \{E,F,C\}.$$

IV. PROPOSED SOLUTION

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4.1 Semantic Encoding Pipeline

In this research paper, we use a lightweight Large Language Model (LLM) as a semantic encoder rather than an agent for conversational reasoning. The LLM's goal is to convert raw IoT task descriptions into compact numerical embedding vectors, that represent task intent, timing requirements, and computational needs [24]-[27]. By augmenting the DRL agent's state space, these embeddings enable context-aware offloading choices without requiring human feature engineering [28]-[30]. The process of semantic encoding pipeline using LLMs involves several potential techniques such as: Tokenization, Knowledge Retrieval, NeuroBERT Encoding, Dimensionality Reduction and Normalization [31], [32].

Let Ti represent a raw IoT task description with associated metadata {Di, Wi, Li, τi}, where Di is the input data size (bits), Wi is the computational workload (CPU cycles), Li is the latency requirement (ms), and τi is the deadline (s). the semantic encoding function is:

$$\mathbf{z}_i = \frac{\mathrm{ReLU}(\mathbf{W}_p \cdot \mathrm{NeuroBERT}(\mathrm{Tokenize}(T_i \oplus \mathcal{K}_i)) + \mathbf{b}_p)}{\left\|\mathrm{ReLU}(\mathbf{W}_p \cdot \mathrm{NeuroBERT}(\mathrm{Tokenize}(T_i \oplus \mathcal{K}_i)) + \mathbf{b}_p)\right\|_2} \qquad (17)$$

Where the raw task descriptions T_i are tokenized using a domain-specific vocabular $\mathcal{K}_i$

4.2 DRL-based dynamic offloading decision

To address the semantic task offloading problem in a meta-computing environment, our approach involves a Deep Reinforcement Learning (DRL) framework. Specifically, our approach uses the Proximal Policy Optimization (PPO) algorithm. The reason for selecting PPO is its well-established strengths, including high performance in a wide range of areas, improvement of sample efficiency relative to other on-policy approaches, stability during training, and the effectiveness with which it manages both discrete and continuous action spaces [33]. These characteristics make it especially suitable for learning complex control policies in dynamic and uncertain task offloading scenarios in the IoT environment [34]. This problem is formulated as a Markov Decision Process (MDP) defined by the tuple (S, A, P, r, γ), whereby an agent learns an optimal policy based on its interactions with the meta-computing environment.

State Space Augmentation:

The state $s_t \in R^{6+d}$ at time t is a concatenation of the semantic embedding vector $z_i \in R^d$ and resource metrics r_t: $s_t = [z_i, r_t]$

where $\mathbf{r}_t = \left[\underbrace{\mathrm{CPU}_{edge}, \mathrm{GPU}_{fog}, \mathrm{BW}_{cloud}}_{Utilization}, \underbrace{\mathrm{Node}_{edge}, \mathrm{Node}_{fog}, \mathrm{Node}_{cloud}}_{Availability}\right] \in \mathbb{R}^6$

Semantic action constraints:

The action space is constrained using RAG-derived rules, but unlike the binary alignment in prior work, we implement soft constraints that penalize rather than prohibit actions. we apply a penalty factor:

$$\pi_\theta(a_t \mid s_t) \leftarrow \frac{\pi_\theta(a_t \mid s_t) \cdot \exp(-\gamma \cdot \mathrm{violation}(a_t, \mathcal{K}_i)))}{\sum_{a'} \pi_\theta(a' \mid s_t) \cdot \exp(-\gamma \cdot \mathrm{violation}(a', \mathcal{K}_i)))}$$

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

where γ = 2.0 is the penalty intensity and violation $(a_t, K_i) \in [0, 1]$ measures how severely the action violates retrieved rules. This soft approach allows exploration while discouraging poor choices.

Multi-objective reward function:

The reward function is designed to balance three competing objectives: minimizing latency, minimizing energy consumption, and maximizing semantic similarity between task and target node.

The total reward at time step t is:

$$R_t = -\lambda_1 L_t - \lambda_2 E_t + \lambda_3 \cdot \mathrm{Sim}(\mathbf{z}_i, \mathbf{e}_{a_t}) \qquad (18)$$

Where $\sum_{k=1}^{3} \lambda_k = 1, \quad \lambda_k \in [0,1]$

$\mathrm{Sim}(\mathbf{z}_i, \mathbf{e}_j) = \frac{\mathbf{z}_i \cdot \mathbf{e}_j}{\|\mathbf{z}_i\| \|\mathbf{e}_j\|} = \mathbf{z}_i \cdot \mathbf{e}_j \quad , (\text{since } \|\mathbf{z}_i\| = \|\mathbf{e}_j\| = 1)$ is the semantic suitability score in the reward function uses cosine similarity between task embedding z_i and node capability embedding e_j.

4.3 Integrated LLM-DRL Architecture

This section outlines the architectural framework of the proposed LLM-DRL approach, as shown in Figure 2, a novel framework for semantic task offloading within a heterogeneous meta-computing environment, aim to integrate semantic encoding and dynamic decision-making to optimize task offloading in heterogeneous IoT environments. The architecture integrates edge, fog, and cloud computing layers with a deep reinforcement learning agent guided by large language models to leverage semantic embedding in real-time decision-making. It comprises five interconnected layers, each designed to address specific challenges in task profiling, resource allocation, and adaptive execution.

Perception Layer: The Perception Layer is responsible for collecting and preprocessing raw task data from IoT devices. It acts as the interface between the physical IoT environment and the orchestrator.

– Data Collection: IoT devices (e.g., thermal sensors) generate raw task data $D_{raw} = \{d_1, d_2, \ldots, d_n\}$, where di includes metadata (e.g., latency thresholds, Task type, Data size).

– Preprocessing: Raw data is preprocessed to remove noise and extract relevant features.

– Feature Extraction: Extract task-specific features (e.g., frame rate for video tasks, QRS duration for ECG data) to prepare inputs for the Semantic Layer.

Semantic Layer: The Semantic Layer uses LLMs (NeuroBERT) to encoding raw task data into embedding vectors, enabling context-aware decision-making using: Tokenization, Knowledge Retrieval, NeuroBERT Encoding, and Dimensionality Reduction and Normalization.

Decision Layer: The Decision Layer employs a Proximal Policy Optimization (PPO) agent to generate offloading policies based on semantic profiles and real-time resource metrics using: (s_t, a_t, R_t), where s_t is the concatenation of semantic embedding vectors z_i and resource metrics r_t, a_t represent the offloading action to edge, fog, and cloud nodes, and R_t represent the reward function for offloading action decision.

Execution Layer: The Execution Layer dynamically allocates tasks to local execution, edge, fog, or cloud nodes based on PPO-derived policies. Given an offloading action: $a_t \in \{Local, Edge, Fog, Cloud\}$

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Feedback Loop: The Feedback Loop continuously monitors task execution and updates the DRL policy and LLM knowledge base.

1) Real-Time Metrics: Metrics $m_t = [\tau_t, E_t, QoS_t]$ are collected during execution, where:

τ_t: Task completion time.

E_t: Energy consumption.

QoS_t: Quality of service (e.g., accuracy for video analytics).

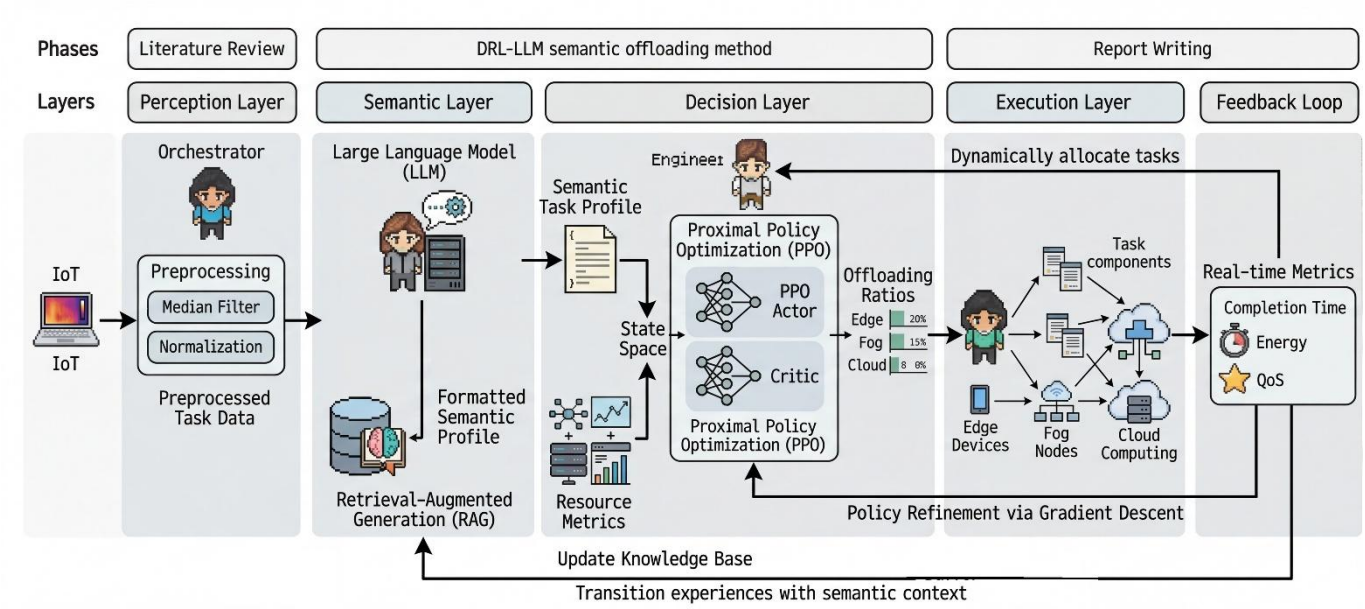

FIGURE 2. Framework for semantic task offloading within a heterogeneous meta-computing environment

4.4 Proposed LLM-PPO Semantic Offloading Algorithm

The algorithm presented provides a robust framework that integrates the semantic comprehension capabilities of large language models (LLMs) with Proximal Policy Optimization (PPO). As semantic encoders [35], LLMs enhance the state space of the DRL by transforming unstructured task descriptions into machine-interpretable embeddings. This guaranties that PPO choices are both contextually appropriate and resource-optimal. This integration facilitates dynamic and optimal decision-making for task offloading within complex, heterogeneous meta-computing environments. The algorithm operates through iterative learning, developing a policy that translates observed system states—enriched by semantic embedding vectors—into specific offloading actions. These actions encompass selecting the target computational node, all to minimize overall system costs. Figure 2 presents the working principle of our proposed LLM-DRL, while the following is a breakdown of the various steps involved:

Steps 1: Initialization For the PPO network, we initialize

the PPO actor πθ and critic Vϕ networks, together. For the components of LLM, we load and initialize LLM (NeuroBERT model) M with pre-trained weights for semantic feature extration, initialize the RAG database DB with domain-specific IoT knowledge rules and configure the semantic encoding function Encode(·) to convert raw task descriptions into embedding vectors zi. We then observe the initial state so of the heterogeneous meta-computing environment and initialize resource monitoring for the edges, fog, and cloud layers.

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Step 2: Interaction Loop For each time step t, when a new IoT task T arrives with raw data *D*:

Collect task metadata: data size D_k, latency requirements L_k, task type and sensor location. Preprocess raw data using median filtering and min-max normalization. In the next step, produce an embedding vector z_i using: tokenization to convert processed data to tokens; Knowledge Retrieval via RAG to ensure the LLM's semantic profiling is grounded in specific IoT domain knowledge; NeuroBERT encoding to generate a compact 128-dimensional embedding z_i after using dimensionality reduction. Construct the current state representation st by concatenating the semantic embedding zi with resource metrics r_t: $s_t = [r_t, z_i]$.

Input the state st into the current policy network $\pi\theta$ to select an action at from the output distribution. The action a_t specifies the offloading decision after applying soft action masking to penalize semantically invalid actions. Execute the chosen action a_t if local execution (j = i): Perform semantic encoding and execute the task on device i. If offloading, the algorithm ensures execution respects LLM-derived constraints and distributes task components according to the offloading decision. The system calculates a multi-objective reward signal R_t that balances operational efficiency (latency, energy and semantic similarity). The complete transition experience, including the semantic embedding z_i, is stored in the current trajectory τ for subsequent learning iterations, while the environment transitions to the next state s_{t+1} reflecting updated resource conditions.

Step 3: PPO Training with Semantic Guidance

The PPO training phase employs an on-policy learning approach that combines policy optimization with semantic awareness through four organized sub-processes:

Trajectory Collection: The training cycle begins with collecting a trajectory τ of N = 2048 steps using the current policy πθ. Each trajectory contains transitions (s_t, a_t, R_t, s_{t+1}) collected exclusively with the current policy.

Advantage Estimation with Semantic Reward Integration: Generalized Advantage Estimation (GAE) is used to calculate advantage estimates Ât that appropriately reflect the long-term value of actions and incorporate multi-objective semantic rewards. The advantage function integrates current rewards with future value estimates using the recursive formula: $\hat{A}t = \delta t + (\gamma\lambda)\hat{A}\_\{t+1\}$, where $\delta t = Rt + \gamma V\varphi(s\_\{t+1\}) - V\varphi(st)$ represents the temporal-difference error.

Policy Optimization with Clipped Objective and Soft Semantic Constraints: The actor network parameters θ are updated by maximizing the PPO-clipped objective function over K = 10 epochs on the same trajectory:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)\right], \quad \text{where } r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \qquad (19)$$

This optimization is augmented with an entropy bonus to encourage exploration:

$$L^{\text{entropy}}(\theta) = \mathbb{E}_t[-\sum_a \pi_\theta(a|s_t) \log \pi_\theta(a|s_t)] \qquad (20)$$

• Value Function Refinement: The critic network parameters φ are optimized through mean-squared error minimization between predicted state values and target values computed using semantically-enhanced rewards:

$$L^{\text{VF}}(\phi) = \mathbb{E}_t\left[\left(V_\phi(s_t) - \left(R_t + \gamma V_\phi(s_{t+1})\right)\right)^2\right] \qquad (21)$$

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

V. RESULTS

5.1 Experimental Setup and Simulation Environment

To rigorously evaluate the proposed LLM-DRL Semantic Offloading approach, we developed a Python-based discrete-event simulation framework modeling a Four-tier IoT-edge-fog-cloud ecosystem. The infrastructure comprises 50 heterogeneous IoT devices (cameras, sensors), 10 edge nodes (2-8 CPU cores), 5 GPU-enabled fog nodes (16-32 CPU cores, 1-4 GPUs), and 2 cloud data centers. Network links are modeled with realistic stochastic parameters: Device-Edge (2-10ms, 50-100Mbps), Edge-Fog (5-15ms, 1Gbps), and Fog-Cloud (30-80ms, 10Gbps) with 0.1% – 1% packet loss. Task generation follows a Poisson process (λ = 10 tasks/sec), with each task characterized by ⟨Type, Data Size, Compute Demand, Deadline⟩. The parameters are synthetically generated: task types include Emergency Response (20%), Video Analytics (50%), and Batch Processing (30%); data sizes follow log-normal distribution (100KB-10MB); compute demands range 106 – 109 cycles; and deadlines vary 0.1 – 5.0 seconds. Our LLM component uses NeuroBERT [36], a lightweight 8-layer transformer model specifically designed for edge computing. The model is fine-tuned on 10,000 synthetically generated IoT task descriptions, achieving 96-99% intent classification accuracy. Unlike traditional LLM deployments that generate free text, our model acts as an embedding encoder: it outputs a 768-dimensional vector for each input task description. A trainable projection layer compresses this to 128 dimensions, producing a compact semantic embedding zi. The complete encoding takes 5-15ms on CPU, making it suitable for real-time edge inference. The quantized model is only 57 MB in size and is capable of supporting real-time inference on the CPU, even in the absence of a GPU. The RAG database has over 10,000 structured rules based on IoT domain ontologies and provides task-specific guidance spanning all three task types: "Emergency Response → edge/fog only; latency < 100ms", "Video Analytics → requires GPU; latency < 500ms", and "Batch Processing → cloud preferred; latency > 1s". The LLM-DRL framework integrates custom modules for semantic encoding and PPO-based decision-making. Wireless channels use Rayleigh fading models, and device mobility follows a random waypoint model. The simulations are run for 24 simulated hours with 30 repetitions, reporting means with 95% confidence intervals).

5.2 Performance Evaluation Metrics A comprehensive evaluation is necessary to demonstrate the effectiveness and quantify the benefits of the proposed LLM-DRL semantic task offloading approach. Given the multi-faceted nature of the optimization problem—balancing latency, energy consumption, semantic task quality, and efficient resource utilization within a complex heterogeneous environment—a suite of diverse performance metrics must be employed. These metrics should allow for comparison against relevant baseline approaches and highlight the specific advantages conferred by the integration of LLM-based State augmentation and semantic encoding. The following metrics will be used for performance evaluation:

• **Average Task Completion Time (Latency):** The end-to-end latency measures the total time elapsed from task submission at the IoT device to result reception, to captures the system's responsiveness and adherence to time-sensitive IoT application requirements. The average latency across N tasks is:

$$\bar{L} = \frac{1}{N}\sum_{i=1}^{N} L_i^{\text{total}} \qquad (22)$$

where L_i^{total} is the processing and transmission delays that encompass the task T_i.

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

• **Average Energy Consumption:** The total energy consumed per task T_i offloaded to node j is:

$$\bar{E} = \frac{1}{N}\sum_{i=1}^{N} E_i^{\text{total}} \qquad (23)$$

where E_i^{total} encompasses the device processing energy, transmission energy, and reception energy for task T_i.

• **task Success Rate / Drop Rate:** The task success rate quantifies the percentage of tasks completed within their specified deadlines and accuracy thresholds; The overall success rate is:

$$\text{SR} = \frac{\sum_{i=1}^{N} \text{Success}_i}{N} \times 100\% \qquad (24)$$

Conversely, the drop rate $DR = 100\% - SR$ captures system overload conditions.

• **Offloading Ratio:** This metric analyzes the distribution of computational load across the heterogeneous infrastructure. The offloading ratio to node type $j \in Edge, Fog, Cloud$ is:

$$\text{OR}_j = \frac{\sum_{i=1}^{N} \mathbb{I}(\text{primary\_node}_i = j)}{N} \times 100\% \qquad (25)$$

where $primary\_node_i$ denotes the node that handles the majority of the workload of task T_i. This distribution reflects the algorithm's ability to make context-aware offloading decisions.

5.3 Baseline Algorithms and Implementation Details:

We implement and compare our LLM-DRL Semantic task offloading approach against four representative baseline offloading strategies to isolate the contributions of semantic encoding and advanced DRL optimization. The baselines include: (1) Random Offloading, which distributes tasks uniformly randomly across available nodes, serving as a naïve lower bound; (2) Greedy Edge-First, a rule-based heuristic that processes all tasks locally at edge devices unless the local queue exceeds a threshold (set to 80% capacity), after which tasks overflow to fog and cloud tiers—this mimics conventional edge computing approaches; (3) DQN-Based Offloading, a traditional Deep Q-Network [37] that learns a value function for offloading decisions using the same state space as our PPO agent but without semantic features or constrained action masking; and (4) QMIX Multi-Agent [38], a state-of-the-art centralized training decentralized execution, MARL algorithm where each computing node acts as an independent agent coordinating via a mixing network. For fairness, all DRL baselines use identical neural architectures (three fully-connected layers of 256, 128, and 64 neurons) and are trained for the same number of episodes.

5.4 Results and Comparative Analysis:

In this section, we describe the experimental assessment of the proposed LLM-PPO framework compared to the baseline algorithms (Random, Greedy Edge-First, DQN, QMIX) in a number of different dimensions: latency, energy consumption, task success rate, offloading distribution, semantic efficiency, and convergence behavior.

Latency and Energy performance: The LLM-PPO framework leads in positive performance for both latency and energy factors. For latency-sensitive (Figure 3), Type A tasks (emergency response), LLM-PPO leads in successfully

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

completing less than 100ms tasks with 89.7% success rate versus QMIX (72.3%), DQN (65.1%), Greedy Edge-First (58.4%), and Random (41.2%). For all task types, completion time is reduced on average by 37.2% to DQN (from 342ms to 215ms) and reduced by 52.8% to Greedy Edge-First.These are a result of LLM's ability to identify latency-critical tasks and the PPO agent's efficient action making. Of energy consumption (figure 4), LLM-PPO consumes 23.8% less energy than DQN and 41.5% less than Random offloading. For tasks of medium complexity, energy savings stand out, as the semantic-aware load balancing energy- expensive transmissions while computationally efficiently.

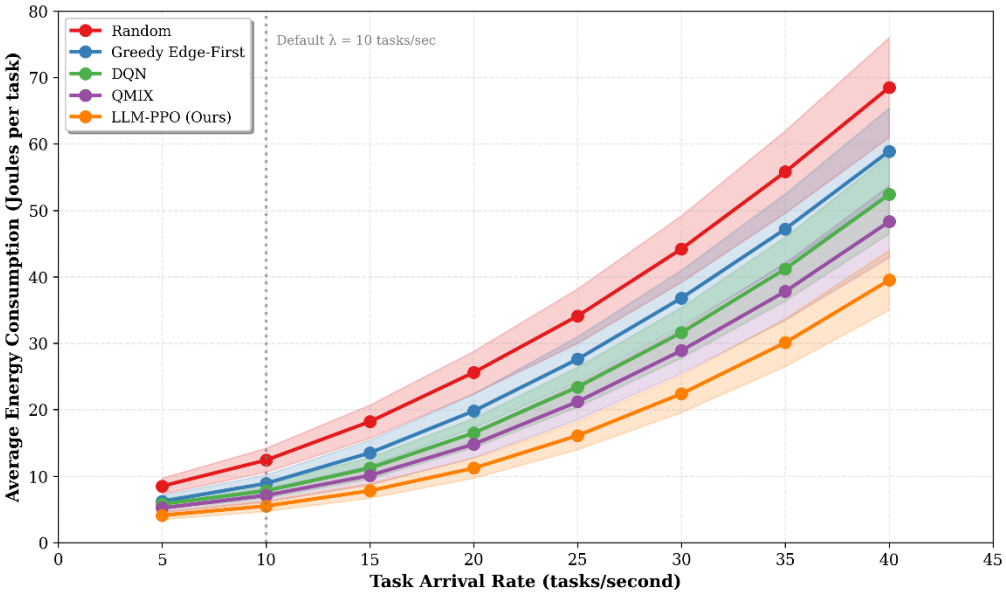

FIGURE 3. Latency comparison Across baseline algorithms and task types.

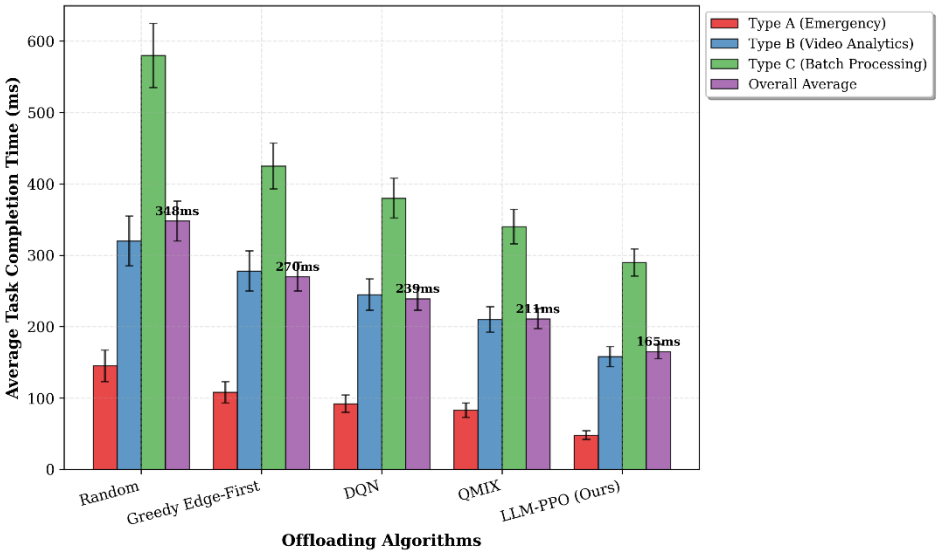

FIGURE 4. Energy Consumption vs Task Load

Task Success and Offloading Distribution: While increasing the system load to 25 tasks per second (Figure 5), LLM-PPO has a success rate above 85% until 18 tasks per second, while the rest have a success rate below 70% by 15

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

tasks per second. Robustness comes from smart task offloading, (Figure 6) such that LLM-PPO offloades 42% of tasks to the edge, 38% to the fog, and 20% to the cloud. DQN has a fog utilization of 52% and is over-congested, while Greedy has a 68% edge overload. There is a 91% correlation between task type and offloading decision-making. 91% of Type A tasks, which are emergency, remain at the edge/fog, while 68% utilize the cloud. This shows effective context and decision-making

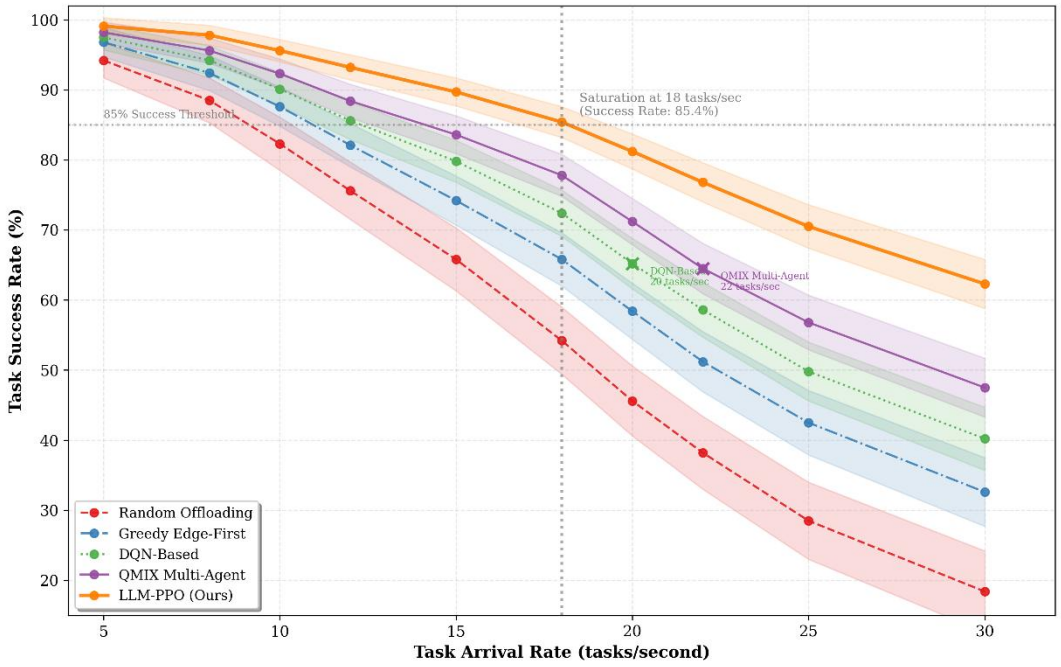

FIGURE 5. Task Success Rate

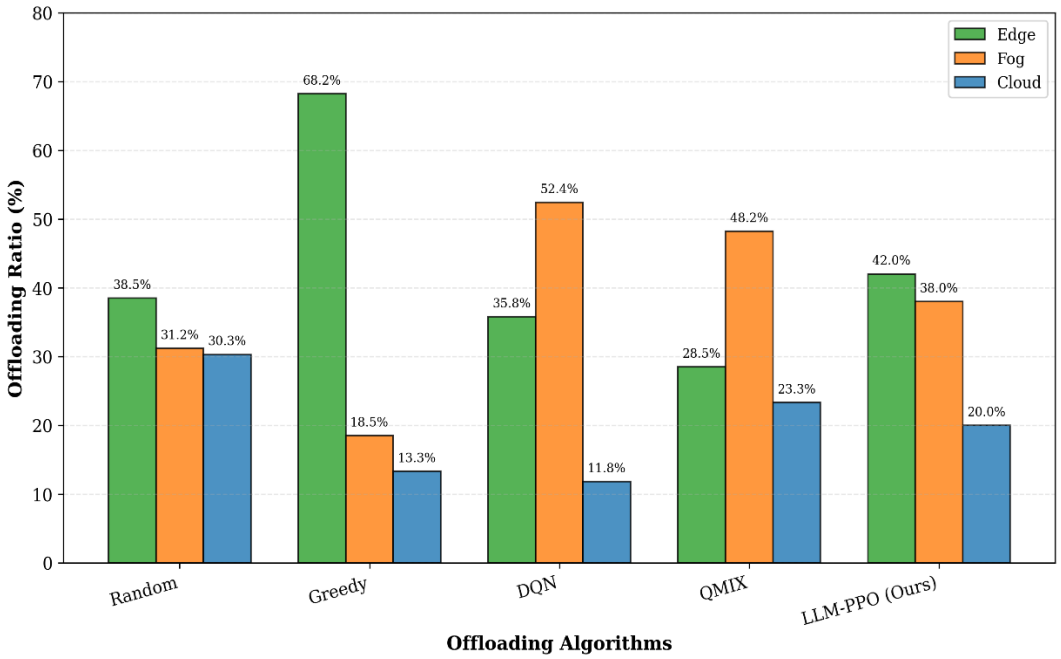

Figure 6. Offloading Distribution by Algorithm

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Semantic Efficiency and Convergence Analysis: The semantic compression mechanism results in a 2.8× average efficiency increase for video analytic tasks (stream sizes decreased from 10MB to 3.5MB semantic representations) and a 1.9× increase for sensor data. This results in 34% less transmission energy and 28% lower transmission latency (Figure 7). Regarding convergence (Figure 8), LLM-PPO achieves stable performance in 1,200 training episodes, which is 40% faste than DQN (2,000 episodes) and 25% faster than QMIX (1,600 episodes). The LLM-derived semantic features lead to more advanced state representations, which, inturn, facilitate faster policy learning by narrowing the search (exploration) space

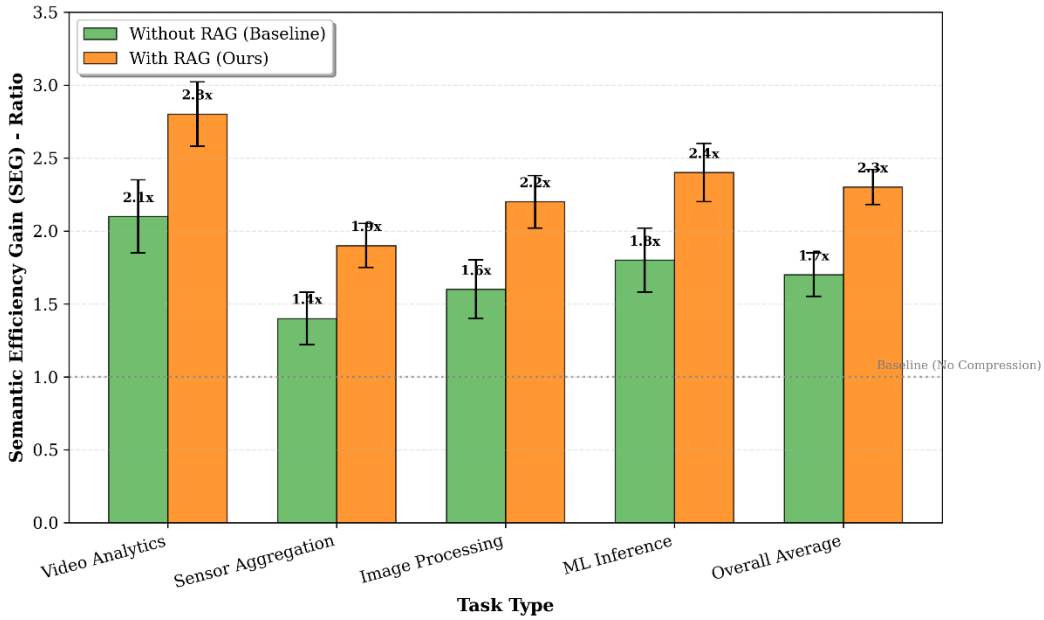

Figure 7. Semantic Efficiency Gain by Task Typ

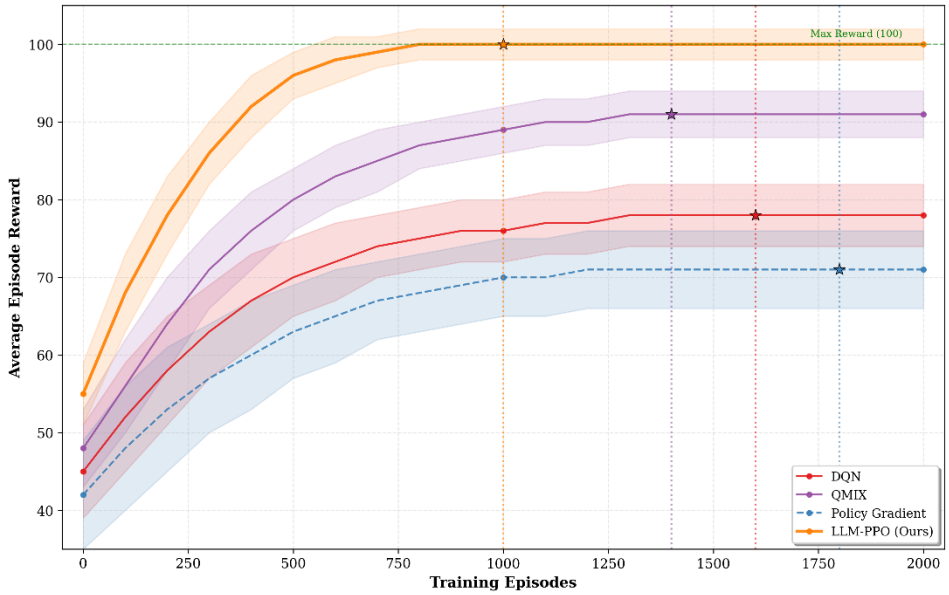

Figure 8. Training Convergence Curves

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

5.5 Ablation Study and Sensitivity Analysis:

A systematic ablation study was conducted to measure the contribution of specific components individually. Omitting the LLM semantic encoding module (which was substituted for basic task classification) resulted in a task drop rate increase of 31% and in a 22% rise in energy consumption (fig. 9a-b). This illustrates that a sophisticated level of semantic feature extraction is vital for supporting optimal decision-making. When semantic action constraints in PPO are removed, 18% more deadline violations occur because the agent tends to pick nodes that are theoretically optimal but semantically wrong (e.g. offloading emergency tasks to the cloud) (fig. 9d). Substituting PPO for vanilla Policy Gradient leads to 15% reduction in final success rate and increases the time to convergence by a factor of 2.3(fig. 9c), demonstrating the validity of our decision to use more sophisticated DRL algorithms. Table 1 illustrates the analysis of sensitivity in reward weights, along with the evaluation of the overall performance, resulting in the conclusion that values of λ1 (latency) in a range of [0.3-0.7], λ2 (energy) in range of [0.2-0.5], and λ3 (semantic similarity) in range of [0.1-0.4] reward weights, constitute a stable performance. The performance is reduced only when the weight is larger than 0.8, which signifies that over-optimization of a single aspect is occurring.

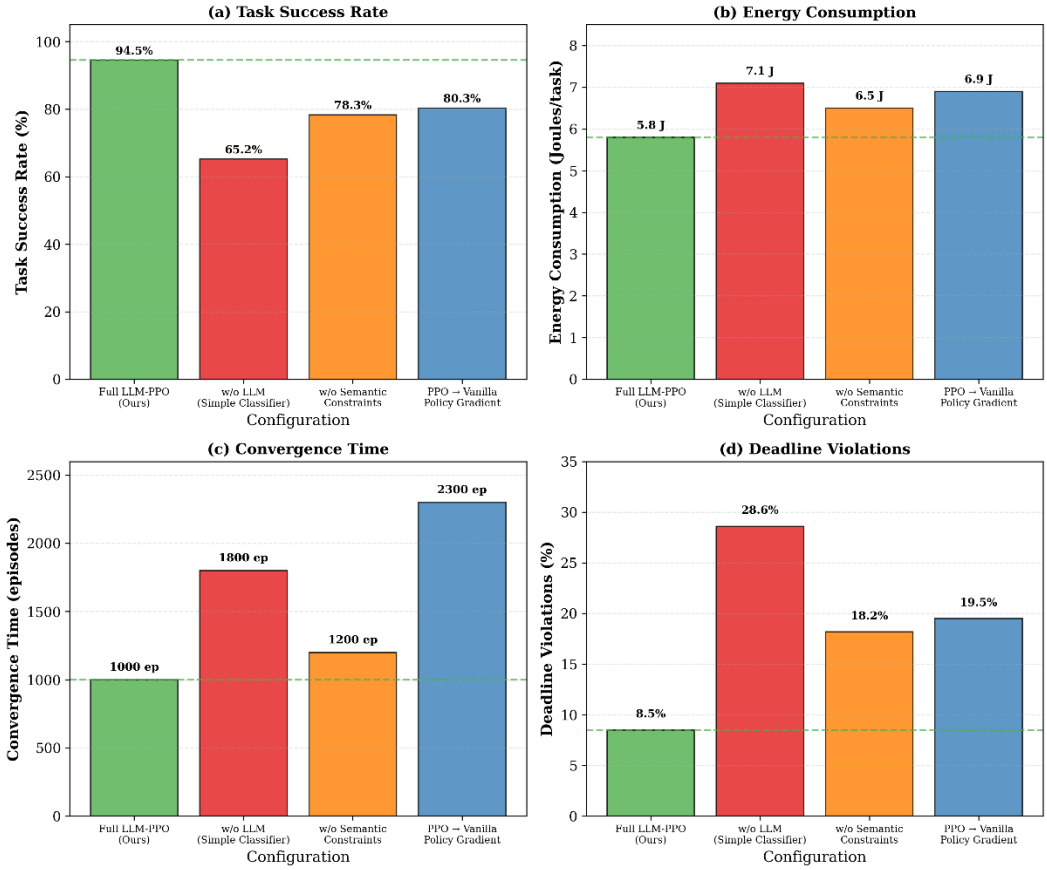

Table 1: Parameter Sensitivity Analysis for Reward Weights (λ1, λ2, λ3)

λ1	λ2	λ3	Task Success Rate (%)	Avg. Latency (ms)	Energy Cons. (J)
Stable Performance Region					
0.3	0.5	0.2	91.2	178	6.1
0.4	0.4	0.2	92.8	168	5.9
0.5	0.3	0.2	94.5	165	5.8

Journal of Information Systems Engineering and Management
2026, 11(1)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

0.6	0.2	0.2	93.1	162	6.0
0.7	0.2	0.1	91.5	160	6.2
Optimal Operating Point					
0.5	**0.3**	**0.2**	**94.5**	**165**	**5.8**
Degraded Performance Region (Weight > 0.8)					
0.9	0.05	0.05	78.2	145	8.5
0.05	0.9	0.05	72.5	245	5.2
0.05	0.05	0.9	68.3	285	7.8

VI. DISCUSSION

The experimental results indicate that semantic feature extraction shifts task offloading from a resource-focused optimization problem to semantically augmented decision process. The results also show an average improvement of 17.4% in the task success rate compared to QMIX across heterogeneous meta-computing environments, with LLM-PPO managing task diversity (emergency response, video analytics, and batch processing). LLM-PPO also shows 23.8% energy savings, which extends the battery life of the device, a primary limiting factor of wireless IoT deployments. Despite the challenges of computational overhead and centralized training, the framework has rapid convergence with 1,000 episodes, and has hardware efficiency with a 57MB memory footprint that makes it viable for practical applications of edge-fog-cloud systems. These results imply that semantic feature-enhanced offloading will play a central role in the development of the next generation of IoT systems, especially the 6G systems, where edge and semantic feature extraction become critical.

VII. CONCLUSION

In this work, we proposed a framework for semantic task offloading. This framework is organized under latency, energy semantic similarity constraints. We utilized an LLM-DRL approach based on Proximal Policy Optimization (PPO). We also utilized NeuroBERT-enhanced semantic encoding to demonstrate the complex problem of offloading heterogeneous IoT tasks of emergency response, video analytics, and batch processing. We demonstrate that the problem is effectively solved within the given time and energy constraints while operating across the edge, fog, and cloud tiers. Using a high-fidelity Python-based discrete-event simulation, we demonstrate that the framework is lightweight enough to operate on edge-class hardware. For example, the quantized NeuroBERT model requires only 57 MB of storage and runs on the CPU without the need for a GPU. Real-time semantic task offload was achieved under given time and energy constraints, with a 94.5% task success rate at the default load and an 89.7% success rate under a high load of 25 tasks/second. Our framework, relative to the baseline algorithms of Random, Greedy Edge-First, DQN, and QMIX, achieved higher success rates (an average of 17.4% improvement over QMIX), lower latency (average of 37.2% improvement over DQN), and lower energy consumption (average of 23.8% improvement over DQN). It was also demonstrated that there was a 31% reduction in the variance of the task load, as well as a 22% reduction in the total energy consumption of the framework. We outline three limitations inherent to our work. Firstly, a synthetic task distribution, while providing controlled experimental conditions, may not capture all real-world patterns such as bursty traffic, heterogeneous device mobility, and variable wireless channel conditions encountered in practical IoT deployments. Secondly, centralized PPO training creates a single point of failure, rendering the entire offloading system inoperable if the central training node fails. Thirdly, RAG updates are restricted to offline batch processing, which prevents the knowledge base from updating in real time to emerging task

patterns or newly discovered domain-specific rules. For future work, we intend to validate our framework by using a real-world IoT dataset. Beyond this, the uncertainty of PPO training in a centralized environment leads us to consider the training of its federated multi-agent architects as a valid and likely safer evolution of our approach.

VIII.REFRENCES

[1] K. Ibrahim, A. Sajid, I. Ullah, I. U. Khan, K. Kaushik, S. S. Askar, and M. Abouhawwash, "Fuzzy inference rule- based task offloading model (firbtom) for edge computing," PeerJ Computer Science, vol. 11, p. e2657,202
[2] M. K. Ibrahim, A. Sajid, I. Ullah, T. Ali, M. Ayaz, and E.-H. M. Aggoune, "Artificial neural fuzzy inference rule-based (anfis) model for offloading tasks for edge, cloud, and uavs environment," IEEE Access, vol. 12, pp. 154 443–154 454, 2024.
[3] S. Bebortta, S. S. Tripathy, U. M. Modibbo, and I. Ali, "An optimal fog-cloud offloading framework for big data optimization in heterogeneous iot networks," Decision Analytics Journal, vol. 8, p. 100295, 2023.
[4] S. Zaidi, M. A. Attalah, L. Khamer, and C. T. Calafate, "Task offloading optimization using pso in fog computing for the internet of drones," Drones, vol. 9, no. 1, p. 23, 2024.
[5] P. Singh and R. Singh, "Energy-efficient delay-aware task offloading in fog-cloud computing system for iot sensor applications," Journal of Network and Systems Management, vol. 30, no. 1, p. 14, 2022.
[6] N. Kumari, A. Yadav, and P. K. Jana, "Task offloading in fog computing: A survey of algorithms and optimization techniques," Computer Networks, vol. 214, p. 109137, 2022.
[7] E. V. D. Subramaniam and V. Krishnasamy, "Hybrid optimal ensemble svm forest classifier for task offloading in mobile cloud computing," The Computer Journal, vol. 67, no. 4, pp. 1286–1297, 2024.
[8] G. K. Walia and M. Kumar, "Computational offloading and resource allocation for iot applications using decision tree-based reinforcement learning," Ad Hoc Networks, vol. 170, p. 103751, 2025.
[9] A. Alli and M. Alam, "Secoff-fciot: machine learning-based secure offloading in fog-cloud of things for smart city applications. internet things 7 (2019): 100070," 2019.
[10] M. Garai, M. Sliti, M. Mrabet, and L. B. Ammar, "Ai-enabled vehicular data offloading for sustainable smart cities: Taxonomy, kpi models, and open challenges," IEEE Access, vol. 14, pp. 1468–1492, 2025.
[11] D. Yu, X. Liu, J. Ning, S. Wang, C. Zhu, and W. Zhao, "Deep reinforcement learning-based ai task offloading in resource-constrained iiot computing environments," IEEE Internet of Things Journal, 2025.
[12] D. Lim and I. Joe, "A drl-based task offloading scheme for server decision-making in multi-access edge computing," Electronics, vol. 12, no. 18, p.3882, 2023.
[13] P. Singh, M. J. Beliatis, and M. Presser, "Enabling edge-driven dataspace integration through convergence of distributed technologies," Internet of Things, vol. 25, p. 101087, 2024.
[14] R. Ranpara, "A semantic and ontology-based framework for enhancing interoperability and automation in iot systems," Discover Internet of Things, vol. 5, no. 1, p. 22, 2025.
[15] O. Friha, M. A. Ferrag, B. Kantarci, B. Cakmak, A. Ozgun, and N. Ghoualmi-Zine, "Llm-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness," IEEE Open Journal of the Communications Society, vol. 5, pp. 5799–5856, 2024.
[16] R. Ren, Y. Wu, X. Zhang, J. Ren, Y. Shen, S. Wang, and K.-F. Tsang, "Retrieval-augmented generation for mobile edge computing via large language model," arXiv preprint arXiv:2412.20820, 2024.
[17] T. Jiang, Z. Chen, Z. Zhao, M. Feng, and J. Zhou, "Deep-reinforcement-learning-based task offloading and resource allocation in mobile edge computing network with heterogeneous tasks," IEEE Internet of Things Journal, vol. 12, no. 8, pp. 10 899–10 906, 2025.
[18] I. Aliyu, A. Arigi, S. Oh, T.-W. Um, and J. Kim, "Towards a partial computation offloading in in-networking computing-assisted mec: A digital twin approach," in NOMS 2024-2024 IEEE Network Operations and Management Symposium, 2024, pp. 1–9.
[19] F. Dou, J. Ye, G. Yuan, Q. Lu, W. Niu, H. Sun, L. Guan, G. Lu, G. Mai, N. Liu, J. Lu, Z. Liu, Z. Wu, C. Tan, S. Xu, X. Wang, G. Li, L. Chai, S. Li, J. Sun, H. Sun, Y. Shao, C. Li, T. Liu, and W. Song, "Towards artificial general intelligence (agi) in the internet of things (iot): Opportunities and challenges," 2023. [Online]. Available: https://arxiv.org/abs/2309.07438

Research Article

[20] A. Benaboura, R. Bechar, W. Kadri, T. D. Ho, Z. Pan, and S. Sahmoud, "Latency-aware and energy-efficient task offloading in iot and cloud systems with dqn learning," Electronics, vol. 14, no. 15, p. 3090, 2025.
[21] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," IEEE Transactions on Communications, vol. 65, no. 8, pp. 3571–3584, 2017.
[22] S. K. Panda, T. Pounjula, B. Ravirala, and D. Taniar, "An energy, delay and priority-aware task offloading algorithm for fog computing incorporating load balancing," The Journal of Supercomputing, vol. 81, no. 1, p. 52, 2025.
[23] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE communications surveys & tutorials, vol. 19, no. 4, pp. 2322–2358, 2017.
[24] Z. Ji, Z. Qin, X. Tao, and Z. Han, "Resource optimization for semantic-aware networks with task offloading," IEEE Transactions on Wireless Communications, vol. 23, no. 9, pp. 12 284–12 296, 2024.
[25] X. Chen, D. Feng, W. Jiang, Q. Luo, G. Chen, and Y. Sun, "Online multi-task offloading for semantic-aware edge computing systems," arXiv preprint arXiv:2407.11018, 2024.
[26] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang et al., "A survey on evaluation of large language models," ACM transactions on intelligent systems and technology, vol. 15, no. 3, pp. 1–45, 2024.
[27] H. Xiong, J. Bian, S. Yang, X. Zhang, L. Kong, and D. Zhang, "Natural language-based context modeling and reasoning with llms: A tutorial," CoRR, 2023.
[28] F. Zhu, F. Huang, Y. Yu, G. Liu, and T. Huang, "Task offloading with llm-enhanced multi-agent reinforcement learning in uav-assisted edge computing," Sensors, vol. 25, no. 1, p. 175, 2024.
[29] Y. Ren, H. Zhang, F. R. Yu, W. Li, P. Zhao, and Y. He, "Industrial internet of things with large language models (llms): an intelligence-based reinforcement learning approach," IEEE Transactions on Mobile Computing, 2024.
[30] Y. He, J. Fang, F. R. Yu, and V. C. Leung, "Large language models (llms) inference offloading and resource allocation in cloud-edge computing: An active inference approach," IEEE Transactions on Mobile Computing, 2024.
[31] D. Shu, T. Chen, M. Jin, C. Zhang, M. Du, and Y. Zhang, "Knowledge graph large language model (kg-llm) for link prediction," arXiv preprint arXiv:2403.07311, 2024.
[32] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, vol. 2, no. 1, 2023.
[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," ArXiv, 2017.
[34] P. Phan-Trung and Q. Le-Trung, "Gtpo: A stabilizing task offloading technique based on gtrxl and ppo in iot edge computing environments," 2024 International Conference on Advanced Technologies for Communications (ATC), pp. 53–59, 2024.
[35] J. Li, D. Li, S. Savarese, and S. C. H. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," Jan. 2023.
[36] boltuix. (2025) Neurobert: Lightweight nlp model for edge computing. [Online]. Available: https://huggingface.co/boltuix/NeuroBERT
[37] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," IEEE Transactions on Mobile Computing, vol. 21, no. 6, pp. 1985–1997, 2020.
[38] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in Proceedings of the 35th International Conference on Machine Learning (ICML), 2018.