EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

A Predictive and Multi-Objective Cloud–Edge Scheduling Framework with Task-Aware Encoding for Real-Time Educational Services

Yao Sun[1,*], Zhaozhen Liang[2] and Jingxin Liu[3]

[1]Beijing Jiaotong University, Beijing, 100044, China
[2]Guizhou University, guiyang 550025, Guizhou China
[3]Chongqing University of Science and Technology, Chongqing 401331, China

Abstract

INTRODUCTION: With the rapid expansion of real-time interactive educational applications, including smart classrooms, online assessments, and learning behavior analytics, cloud–edge collaborative computing has become a critical infrastructure for ensuring low-latency and high-reliability service delivery. However, existing scheduling frameworks often rely on reactive or coarse-grained strategies that inadequately capture task-level latency sensitivity and system dynamics, resulting in inefficient resource utilization and unstable performance under high-concurrency educational workloads.
OBJECTIVES: This study aims to design a latency-aware cloud–edge collaborative scheduling architecture capable of accommodating heterogeneous educational tasks, dynamically adapting to fluctuating resource and network conditions, and providing differentiated service prioritization to meet stringent real-time performance requirements.
METHODS: The proposed architecture integrates three tightly coupled components: (1) a task feature encoder that models multimodal educational tasks in terms of computational load, data scale, and latency sensitivity; (2) a latency prediction model that leverages historical system states to anticipate future end-to-end latency trends; and (3) a multi-objective scheduling strategy that jointly optimizes latency, resource utilization, and migration cost through adaptive decision-making across cloud and edge nodes.
RESULTS: Experimental evaluations conducted on real-world educational task datasets demonstrate that the proposed approach significantly outperforms mainstream reinforcement learning–based schedulers. Specifically, it reduces average end-to-end latency by 18.1%, improves average resource utilization by 8.0%, and achieves a task success rate of 96.8% under high-concurrency conditions, while maintaining lower latency jitter and migration overhead.
CONCLUSION: The proposed latency-aware cloud–edge collaborative scheduling architecture provides an effective and scalable solution for guaranteeing low-latency performance in real-time educational services. By combining task-aware representation, predictive latency modeling, and multi-objective optimization, the framework offers strong practical value for deployment in smart classrooms and large-scale online education platforms.

Keywords: cloud-edge collaborative computing, latency-aware scheduling, distributed educational services, task feature encoding, multi-objective optimization

Received on 13 January 2026, accepted on 06 May 2026, published on 27 May 2026

doi: 10.4108/eetsis.11568

*Corresponding author. Email: 18800179166@163.com

1. Introduction

With the widespread adoption of online education, smart classrooms, and learning behavior analytics systems, real-time performance has become a core evaluation metric for educational computing tasks[1]. Applications such as interactive classrooms, online assessments, and behavior recognition require stringent end-to-end latency guarantees, yet traditional cloud-centric architectures struggle to handle the coexistence of diverse task types, frequent workload fluctuations, and unstable network conditions in educational settings[2]. Cloud-edge collaborative computing brings computational capabilities closer to users by offloading processing to edge nodes[3]. However, in the presence of heterogeneous tasks, heterogeneous node capabilities, and dynamic network environments, achieving consistently stable low-latency services remains technically challenging.

Although recent research has made progress in edge scheduling, load prediction, and resource allocation, significant limitations persist. First, many scheduling methods inadequately capture the variations in latency sensitivity across educational tasks, leading to suboptimal prioritization[4]. Second, some approaches rely on fixed rules or local performance indicators, which fail to account for dynamic system characteristics such as network jitter and node congestion[5][6]. Third, existing methods often overlook cross-node resource correlations and migration costs, resulting in unnecessary task movements under high concurrency. Meanwhile, educational tasks typically exhibit strong behavior-driven patterns and burstiness, making it difficult for traditional prediction models to accurately capture their temporal dynamics[7]. These challenges greatly constrain the practical effectiveness of cloud-edge collaborative architectures in educational systems.

To address these issues, this study proposes a latency-aware cloud-edge collaborative scheduling architecture with three main innovations. First, a task feature encoding mechanism is designed to uniformly represent the complexity, interaction patterns, and data structures of educational tasks, providing learnable inputs for scheduling decisions[8]. Second, a latency prediction model is developed to jointly analyze node conditions, network features, and task sequences, enabling accurate estimation of future system loads and latency trends to support proactive scheduling. Third, a multi-objective scheduling strategy is introduced to establish coordinated optimization goals among latency reduction, resource efficiency, and migration cost control, enabling differentiated resource allocation for various educational task types.

Experiments are conducted using real educational task data and a simulation environment. Results show that under high concurrency, the proposed architecture reduces average end-to-end latency by 18.1%, increases node resource utilization by 12.4%, and demonstrates more stable latency (26.4 ms jitter) and lower migration cost (0.51) compared with mainstream reinforcement learning–based methods. The framework exhibits strong scalability and can be applied to smart classroom behavior recognition, real-time allocation of online assessments, and learning behavior analysis. It provides technical guidance for deploying low-latency services in practical educational systems and offers methodological support for latency-aware scheduling research in education.

2. Related Works

2.1 Application Scenarios and Challenges

In educational service environments, typical tasks in real-time distributed systems include interactive classroom video processing, real-time assessment feedback, student behavior recognition, and learning trajectory monitoring[9][10]. These applications impose stringent requirements on end-to-end latency, interaction responsiveness, and resource scheduling efficiency[11]. Existing studies generally rely on online classroom logs, multimodal data collected by edge devices, and task records from cloud-edge deployments as experimental data, while adopting average response latency, task success rate, resource utilization, and migration overhead as core evaluation metrics[12][13].

Cloud-edge collaboration enables dynamic distribution of computing tasks across cloud and edge nodes and is regarded as a promising approach to enhancing the quality of educational services[14]. However, the particular characteristics of educational scenarios introduce multiple challenges to practical deployment: (1) task types are diverse and exhibit significantly different latency requirements, necessitating clear differentiation between real-time interactive tasks and background analytical tasks[15]; (2) edge nodes present strong resource heterogeneity, network status fluctuates frequently, and task workloads often display bursty patterns, making traditional static scheduling strategies insufficient; (3) task migration and node cooperation introduce additional system overhead, which can lead to inefficient resource utilization and latency fluctuations under high concurrency[16].

2.2 Review of Mainstream Methods

In recent years, cloud-edge collaborative scheduling and latency-aware mechanisms have received significant research attention. First, some studies propose cloud-edge-device collaborative models based on task dependency structures, constructing dependency graphs and distributing subtasks across nodes to optimize latency in media applications[17][18]. This approach offers a structured scheduling perspective. However, it remains limited to media streaming tasks, lacking the modeling of multimodal interactive characteristics and differentiated latency priorities required in educational environments. In addition, its treatment of migration cost is idealized and insufficiently accounts for resource heterogeneity and bursty workloads[19].

Second, some studies focus on task scheduling in edge networks, employing resource interleaving and reinforcement

learning strategies to reduce queuing delays[20][21]. These methods can be effective in competitive edge environments. However, they primarily target forwarding or service scenarios on single edge nodes, thus lacking a comprehensive cloud-edge collaborative perspective. Furthermore, their task models are tailored to network traffic processing, which differs considerably from educational interactive tasks in modality, continuity, and real-time constraints, making direct application difficult[22].

Third, research in latency-aware scheduling has proposed data service scheduling models constrained by end-to-end latency, optimizing service requests to meet latency thresholds[23][24]. However, these approaches are commonly based on the link characteristics between the cloud and edge nodes, emphasizing network-layer latency prediction. Consequently, they offer limited treatment of multimodal educational task features, importance stratification, and conflicts among multiple scheduling objectives.

Overall, recent research has heavily emphasized cloud-edge collaboration and latency constraints. However, most of these methods are developed primarily around media, networking, or data services. They fall short in addressing the diversity of educational task types, real-time interaction needs, and heterogeneous resource environments. Particularly, gaps remain in task feature modeling, prioritization, and multi-objective optimization, restricting the ability of current methods to handle the high concurrency and strong latency sensitivity characteristic of educational applications.

2.3 Most Relevant Studies

Among studies most closely related to this work, one approach investigates cloud-edge resource scheduling and proposes a dynamic resource allocation model to optimize resource utilization and reduce processing latency[3][25]. This research highlights the importance of real-time responsiveness and scheduling efficiency by coordinating computational tasks between edge nodes and the cloud[6][26]. While offering valuable insights, the study primarily addresses industrial Internet of Things (IIoT) or manufacturing systems and does not incorporate the integrated perspective of "task feature encoding + latency-sensitive prioritization + elastic migration decisions" required for educational services[27].

Compared with that work, this study introduces domain-specific modeling tailored to educational applications by encoding task interaction properties and latency sensitivity, and subsequently constructing prediction models and multi-objective scheduling strategies. The distinctions include: (1) this work focuses on educational tasks rather than IIoT/manufacturing workloads; (2) it integrates task feature encoding, latency prediction, and differentiated priority scheduling, whereas the related study centers on resource allocation and migration alone[28]; (3) this work emphasizes multi-objective optimization (latency, resource utilization, migration cost) rather than single-objective latency or resource improvements. Although the related research contributes meaningful knowledge to cloud-edge scheduling, it leaves significant room for extension regarding task differentiation, priority recognition, and scheduling design in educational contexts.

2.4 Summary

In summary, although existing studies have achieved notable progress in cloud-edge collaborative computing, latency-aware scheduling, and resource optimization, several research gaps remain in the domain of educational services[29]. First, current literature offers limited understanding of multimodal interactions, priority variations, and bursty task patterns characteristic of educational workloads. Second, most methods emphasize resource or migration optimization but do not jointly consider task feature encoding, latency prediction, and multi-objective coordinated scheduling. Third, due to significant load fluctuations and rapidly changing network conditions in educational service systems, existing scheduling strategies have not been adequately validated in terms of stability and scalability under high concurrency[30].

Unlike prior studies, this research addresses these gaps by integrating task feature encoding, a latency prediction model, and a multi-objective scheduling strategy. The contributions lie in: incorporating latency sensitivity, interaction characteristics, and resource demands of educational tasks into a unified encoding scheme; enabling proactive scheduling based on predictive insights; and achieving cloud-edge collaborative scheduling through real-time multi-objective optimization to enhance both responsiveness and resource utilization in educational services. The next section elaborates the proposed methodological design, system model, and experimental setup.

3. Methodology

3.1 Problem Formulation

This study aims to develop a latency-aware cloud-edge collaborative scheduling architecture for real-time educational services. The problem can be formulated as a multi-objective, dynamic, and partially observable optimization task involving task feature modeling, latency prediction, and decision-making across cloud-edge nodes.

3.1.1 Input Definition

The system operates over discrete time steps $t = 1,2, \ldots, T$, and the inputs consist of three categories of variables:

Task feature vector

$$x_i = [c_i, d_i, s_i, \tau_i, p_i] \quad (1)$$

where:

c_i: task computational load (FLOPs)

d_i: input data size (MB)

s_i: task stream type (video / audio / text / behavior logs)

τ_i: latency sensitivity (0–1)

p_i: task priority (integer level)

System state
$h_t = [u_t^e, b_t^e, m_t^e, u_t^c, b_t^c, m_t^c]$ (2)
where:
u_t^e, u_t^c: CPU/GPU utilization of edge / cloud nodes
m_t^e, m_t^c: available memory
Network latency estimator
$l_{i,t} = l_t^{net} + l_{i,t}^{queue}$ (3)

3.1.2 Output Definition

At time t, the system selects an execution node for each task:
$a_{i,t} \in \{edge, cloud\}$ (4)
and predicts its end-to-end latency:
$\hat{L}_{i,t} = f_\theta(x_i, h_t)$ (5)

3.1.3 Research Objective

The overall optimization objective is to minimize total latency, maximize resource utilization, and minimize migration cost:

$\min_\pi \sum_t \sum_i (\alpha L_{i,t} - \beta R_t + \gamma M_{i,t})$ (6)
where:
$L_{i,t}$: actual latency of task i
R_t: overall resource utilization of nodes
$M_{i,t}$: migration cost
α, β, γ: weighting coefficients

3.2 Overall Framework

As shown in Figure 1, the proposed latency-aware cloud-edge collaborative scheduling architecture consists of three core modules, task feature encoding, latency prediction, and multi-objective scheduling, forming a complete closed-loop process from task input to node-level decision-making. This design addresses key challenges in educational scenarios, including strong task heterogeneity, frequent interactions, resource heterogeneity, and significant network fluctuations.

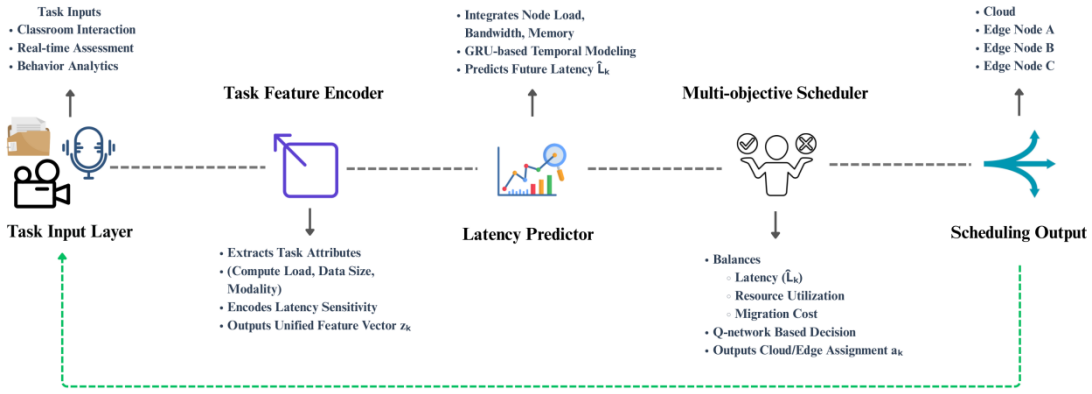

Figure 1. Overall Framework

In the system workflow, diverse tasks originating from classroom interaction, real-time assessment, and behavior analytics first pass through the task feature encoding module, which extracts essential attributes such as computational requirements, data volume, and latency sensitivity and converts them into unified learnable representations. Next, the latency prediction module estimates future latency trends based on node load, bandwidth status, and historical features, enabling proactive scheduling rather than reactive responses constrained by instantaneous states. Finally, the multi-objective scheduling module integrates task attributes, system resource conditions, and migration costs to select the most suitable cloud or edge node for execution, achieving a balanced trade-off among latency, resource utilization, and system overhead.
Through the coordinated operation of these three modules, the framework maintains stable end-to-end latency performance under high concurrency and multimodal workloads, while supporting scalable deployment for large-scale online educational services.

3.3 Module Descriptions

This section provides a complete description of the three core modules in the overall framework: the Task Feature Encoder, the Latency Predictor, and the Multi-objective Scheduler. Each module is presented following the structure of motivation → principle → implementation, accompanied by illustrative diagrams and pseudocode.

3.3.1 Task Feature Encoder

(1) Motivation
Tasks in real-time educational scenarios involve multimodal inputs and diverse task types, including interactive video processing, audio-based answering, textual response evaluation, and behavioral trajectory analysis. Their key attributes include:
computational load c_i (required FLOPs)

data size d_i
modality type s_i (video / audio / text / behavior)
latency sensitivity τ_i (range 0–1)
priority level p_i
Due to the heterogeneity of these attributes, unified vectorized encoding is necessary before feeding them into the prediction and scheduling modules. Thus, a unified task representation z_i must be constructed.
(2) Principle
The modality type s_i is embedded using an embedding function $E(\cdot)$. Specifically, the categorical task stream type is first converted into a one-hot vector before being mapped into a dense continuous space via a learnable embedding layer. Numeric features are normalized, and the concatenated vector is mapped to a fixed-length representation through an Multilayer Perceptron (MLP):

$$e_{s_i} = E(s_i) \quad (7)$$

$$z_i = \sigma\left(W \cdot [e_{s_i}, c_i, d_i, \tau_i, p_i] + b\right) \quad (8)$$

where W, b are learnable parameters and $\sigma(\cdot)$ denotes a nonlinear activation function.
The rationale for adopting an MLP for task encoding, as opposed to more complex structures like Transformers, is grounded in the nature of our data and the operational environment. Since the input task profiles are structured, low-dimensional numerical and categorical attributes, an MLP provides sufficient representative power without the excessive $O(N^2)$ complexity of attention mechanisms. This ensures that the encoding process remains highly efficient on resource-constrained edge nodes, allowing the system to respond to bursty task requests in sub-milliseconds.
Figure 2 illustrates the overall structure of the task feature encoding module, consisting of the modality embedding layer and the MLP-based unified feature generation pipeline.

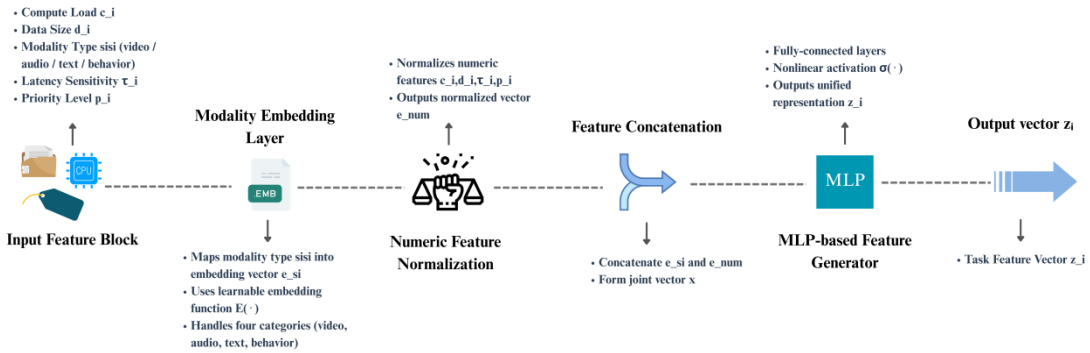

Figure 2. Architecture of the Task Feature Encoder

(3) Implementation
Algorithm 1 presents the implementation process of the Task Feature Encoder, detailing the mapping from raw task attributes to their low-dimensional vector representation.

Algorithm 1: TaskFeatureEncoding

Input: task features $(c_i, d_i, s_i, \tau_i, p_i)$
Output: encoded representation z_i
e_s = Embed(s_i) // modality embedding
e_num = Normalize(c_i, d_i, τ_i, p_i)
$x = \text{Concat}(e_s, e_num)$
$z_i = \text{MLP}(x)$
return z_i

3.3.2 Latency Predictor

(1) Motivation
Real-time educational tasks are heavily affected by dynamic variations in node states, including CPU/GPU utilization, bandwidth, and memory availability. Since system states exhibit strong temporal dependency, relying solely on instantaneous states is insufficient for making scheduling decisions. Therefore, predicting potential future end-to-end latency is essential for achieving proactive resource scheduling.
The system state vector is defined as: $h_t = [u_t, b_t, m_t]$, representing utilization, bandwidth, and memory.
(2) Principle
A Gated Recurrent Unit (GRU) is employed to capture temporal dependencies in the state sequence h_t. The extracted state vector is then fused with the task representation z_i, and an MLP outputs the predicted latency $\hat{L}_{i,t}$:

$$h'_t = \text{GRU}(h_t, h'_{t-1}) \quad (9)$$

$$\hat{L}_{i,t} = \psi([z_i, h'_t]) \quad (10)$$

where $\psi(\cdot)$ denotes an MLP mapping.
The rationale for selecting a GRU over LSTMs or Transformers lies in the strict latency constraints of edge computing. Transformers incur a prohibitive $O(N^2)$ computational complexity, while LSTMs require ~25% more parameters. The GRU's streamlined gating mechanism ensures minimal inference overhead while effectively capturing the dynamic state transitions caused by bursty traffic, making it optimal for real-time edge scheduling.
Figure 3 presents the architectural flow of the latency prediction module, illustrating how system state sequences and task representations are integrated to generate forward-looking latency estimates.

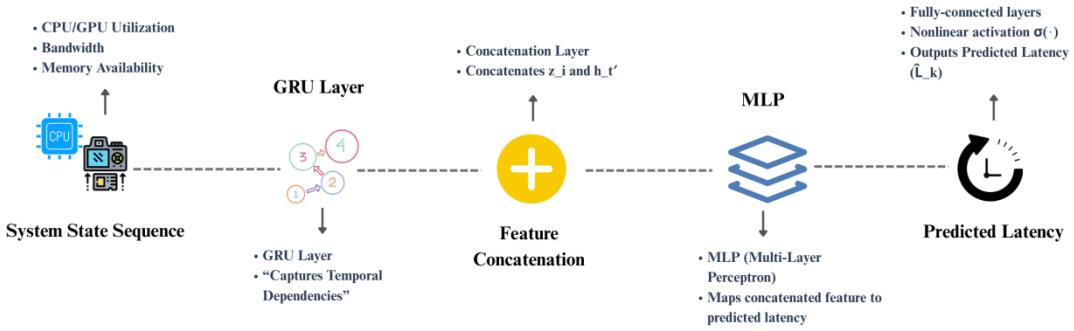

Figure 3. Architecture of the Latency Predictor

(3) Implementation

Algorithm 2 describes the computation process of the Latency Predictor, demonstrating how system state sequences and task embeddings are fused within the prediction model.

Algorithm 2: LatencyPrediction

Input: encoded task z_i, system state seq h_t
Output: predicted latency
1: $h_p rev = \text{InitHidden}()$
2: $h_n ew = \text{GRU}(h_t, h_p rev)$
3: $\text{fused} = \text{Concat}(z_i, h_{new})$
4: $\hat{L} = \text{MLP}(\text{fused})$
5: return $\hat{L}$

3.3.3 Multi-objective Scheduler

(1) Motivation

The scheduling problem involves conflicting objectives:
minimizing latency (dependent on predicted latency $\hat{L}_{i,t}$)
maximizing resource utilization (dependent on system state h_t)
minimizing migration cost (affected by task state and node switching)
Thus, it is necessary to balance these three factors to select the optimal execution node: $a_{i,t} \in \{\text{edge}, \text{cloud}\}$

(2) Principle

The task representation z_i, predicted latency $\hat{L}_{i,t}$, and system state h_t are jointly fed into a multi-objective Q-network:

$$Q(a_{i,t}) = F(z_i, \hat{L}_{i,t}, h_t) \quad (11)$$

The Q-network outputs the action values for the cloud and edge choices, and the action with the higher value is selected as the scheduling decision.

Figure 4 depicts the architecture of the Multi-objective Scheduler, illustrating how task features, predicted latency, and system states are combined to derive the final scheduling output.

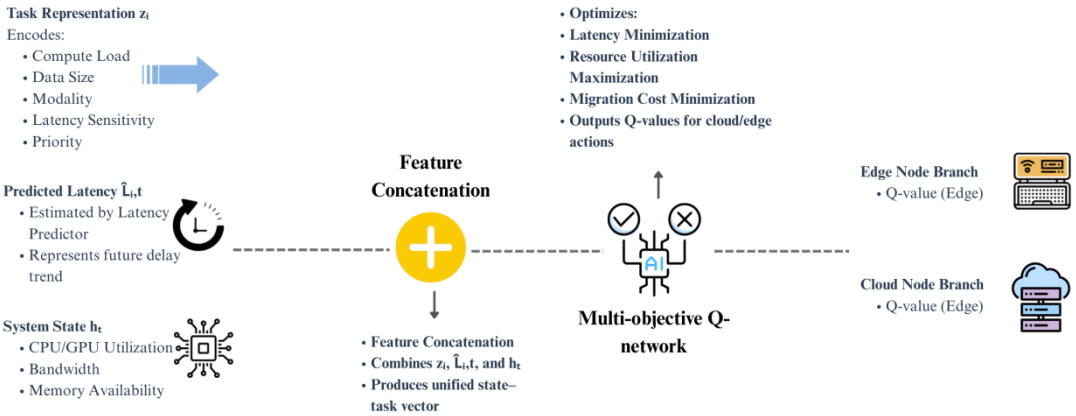

Figure 4. Architecture of the Multi-objective Scheduler

3.4 Objective Function & Optimization

3.4.1 End-to-End Latency Modeling

$$L_{i,t} = L_{i,t}^{trans} + L_{i,t}^{queue} + L_{i,t}^{exec} \quad (12)$$

The total latency ($L_{i,t}$) consists of transmission, queue, and execution delays. Efficient management of these delays is key to minimizing overall latency, especially in dynamic educational environments where high concurrency is common.

(1) Transmission Delay

$$L_{i,t}^{trans} = \frac{d_i}{b_t} \quad (13)$$

where d_i is the data size of task i; b_t is the available bandwidth of the execution node at time t.

(2) Queue Delay

$$L_{i,t}^{queue} = \frac{\lambda_t}{\mu_t(\mu_t - \lambda_t)} \quad (14)$$

where λ_t denotes the task arrival rate at time t; μ_t denotes the service rate (processing capability) at time t.

(3) Execution Delay

$$L_{i,t}^{exec} = \frac{c_i}{\kappa_t} \quad (15)$$

where c_i represents the computational load of task i; κ_t is the available computational capacity of the node at time t.

3.4.2 Resource Utilization Modeling

$$R_t = w_1 u_t + w_2 m_t \quad (16)$$

The system resource utilization (R_t) balances CPU/GPU and memory usage, with weighted normalization. Efficient resource allocation ensures that tasks, especially those requiring substantial computational power, do not overload specific nodes, improving overall system stability.

3.4.3 Migration Cost Modeling

$$M_{i,t} = \delta \cdot d_i \cdot \mathbb{I}[a_{i,t} \neq a_{i,t-1}] \quad (17)$$

Migration cost ($M_{i,t}$) is modeled to minimize unnecessary task migrations, which can increase overhead. This is particularly important in educational applications where task delays due to excessive migration could impact real-time performance.

3.4.4 Multi-objective Loss Function

$$\mathcal{D} = \sum_t \sum_i (\alpha L_{i,t} - \beta R_t + \gamma M_{i,t}) \quad (18)$$

The optimization function ($\mathcal{D}$) combines latency minimization, resource efficiency, and migration cost control. By adjusting the weights α, β, and γ, the system can prioritize different goals depending on the task's needs.

3.4.5 Reinforcement Learning Formulation

$$\pi(a_{i,t}|s_t) = \arg\max_a Q(s_t, a) \quad (19)$$

where $\pi(a_{i,t}|s_t)$ is the scheduling policy; s_t denotes the current system state (including task features, predicted latency, and node status); $Q(s_t, a)$ is the value of action aaa under state s_t. The scheduling policy is optimized through reinforcement learning, where the Q-network updates the task allocation decisions based on rewards. This allows the system to adapt to real-time conditions and learn from past actions.

The Q-network update rule is:

$$Q(s_t, a_{i,t}) \leftarrow Q(s_t, a_{i,t}) + \eta \left(r_t + \lambda \max_{a'} Q'(s_{t+1}, a') - Q(s_t, a_{i,t}) \right) \quad (20)$$

where η is the learning rate; r_t is the immediate reward computed from the loss function; λ is the discount factor; Q' is the target network; a' is a possible future action; s_{t+1} is the next system state.

To bridge the overarching scheduling objective with the Q-network training, the immediate reward r_t is explicitly defined as a linear combination of the negative predicted latency and a priority-weighted penalty for deadline violations:

$$r_t = -\alpha \cdot L_{i,t} - \beta \cdot p_i \cdot \mathbb{I}(L_{i,t} > D_i) \quad (21)$$

where $L_{i,t}$ is the execution latency, D_i is the required task deadline, and p_i is the task priority. $\mathbb{I}(\cdot)$ is an indicator function that outputs 1 if a deadline violation occurs and 0 otherwise. $\alpha$and β are predefined weighting coefficients.

3.4.6 Parameter Optimization

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{J}}{\partial \theta} \quad (21)$$

where θ denotes all trainable parameters (encoder + predictor + scheduler); η denotes the learning rate for gradient descent; $\mathcal{J}$ is the overall loss function. Using gradient descent, the model parameters are iteratively optimized to minimize the overall loss function, ensuring the scheduler improves with experience, making it more effective over time.

4. Experiment and Results

4.1 Experimental Setup

4.1.1 Dataset Overview

This study uses a self-constructed multimodal educational task dataset, EduEdge-2025 (as shown in Table 1), covering typical scenarios such as interactive classrooms, real-time assessment, and behavior analytics. The dataset is built from real online education platform logs, multimodal data collected from student devices (video clips, audio answering streams, textual records), and backend task logs from the cloud.

Table 1. Overview of the EduEdge-2025 Dataset

Data Type	#Tasks	Data Volume	Key Attributes	Relevance to Scheduling Objectives
Video interaction tasks	12,500	1.8 TB	1080p frame sequences, interaction timestamps	High-frequency data streams; stresses bandwidth and edge compute
Audio answering tasks	35,200	420 GB	16 kHz audio clips, response	Highly latency-sensitive;

			latency labels	requires fast decisions
Text evaluation tasks	72,400	280 GB	Answer text and grading logs	Lightweight computation; strict latency limits
Behavior trajectory tasks	9,600	85 GB	Pose keypoints, motion sequences	Extremely sensitive to bandwidth fluctuations
Backend statistical tasks	20,000	300 GB	Log aggregation, batch processing	Suitable for cloud execution; supports priority-based divergence

EduEdge-2025 closely mirrors real educational environments in terms of multimodality and task distribution. Heavy tasks such as video and audio coexist with lightweight text and behavioral tasks, necessitating unified modeling via the task feature encoder. High concurrency requires strong scalability from scheduling strategies, while the high latency sensitivity of video/audio streams provides stringent test conditions for the latency prediction module. Furthermore, the coexistence of background batch jobs and interactive tasks makes multi-objective scheduling essential for balancing real-time performance and resource efficiency.

While public benchmarks are widely used in cloud computing, they lack the specific multi-modal characteristics, strict real-time Quality of Service (QoS) constraints, and user mobility patterns inherent to educational edge AI (EAI) environments. Consequently, relying on the self-constructed "EduEdge-2025" dataset was necessary. However, we acknowledge potential dataset biases. For instance, the task arrival patterns in EduEdge-2025 heavily reflect university campus schedules (e.g., bursty traffic during class transitions) and may not perfectly capture traffic distributions in other domains, such as industrial or medical IoT. To mitigate the risk of overfitting to these specific patterns, we rely on the continuous exploration capability of Reinforcement Learning (RL) and further evaluate the system's robustness through sensitivity analysis.

4.1.2 Hardware Configuration

Experiments adopt a three-tier structure consisting of cloud node + edge nodes + student terminals. Hardware specifications are listed in Table 2.

Table 2. Hardware Configuration

Device Type	Model & Specifications	Purpose
Cloud server	Intel i7-12700F / 32GB RAM / RTX 3060	Global scheduling, backend task execution
Edge Node A	Jetson Nano (128 CUDA cores / 4GB RAM)	Video task inference and caching
Edge Node B	Raspberry Pi 4 (8GB RAM)	Preprocessing of audio answering tasks
Edge Node C	Mini PC (Ryzen 5 5600U / 16GB RAM)	Text and lightweight behavior recognition
Terminal devices	Android tablets / iPad (2020 edition)	Real-time student interaction and data upload

The Jetson Nano provides sufficient inference capability for high-frame-rate video tasks and is suited for heavy edge computing. Raspberry Pi is appropriate for lightweight tasks such as audio, matching real classroom conditions with limited bandwidth and moderate compute demand. The Mini PC supports behavior recognition and small inference workloads, simulating multi-edge-node collaboration in smart classrooms. Although the cloud's RTX 3060 is a consumer-grade GPU, it offers adequate compute for backend statistics and large batch jobs.

Overall, this hardware combination captures the task diversity and resource heterogeneity typical of educational cloud–edge systems, providing a representative deployment environment. Considering the computational limitations of the Jetson Nano, the maximum video concurrency was capped at 3 and model quantization was applied. During peak audio loads, Raspberry Pi 4 offloaded tasks via load balancing across multiple homogeneous nodes to simulate real-world edge clusters.

4.1.3 Evaluation Metrics

As shown in Table 3, system performance is evaluated across four dimensions: latency, resource usage, system stability, and migration overhead.

Table 3. Evaluation Metrics

Metric	Definition	Relevance to Research Objectives
E2E Latency	Time from task submission to completion	Core indicator of latency-aware performance
Jitter	Fluctuation in latency	Measures system stability and predictor robustness
Success Rate	Percentage of tasks completed within deadline	Reflects scheduling effectiveness
Resource Utilization	CPU/GPU/memory occupation ratio	Evaluates resource coordination in multi-objective scheduling
Migration Cost	Overhead caused by task migration	Indicates whether unnecessary

		migrations are avoided
Throughput	Tasks completed per second	Tests scalability and concurrency handling

Together, these metrics constitute the core evaluation framework for cloud-edge collaborative scheduling. E2E Latency and Jitter directly indicate system responsiveness and stability in real-time teaching scenarios, determining whether interactive classroom requirements can be satisfied. Success Rate reflects the correctness and timeliness of scheduling decisions. Migration Cost assesses whether the multi-objective strategy effectively avoids unnecessary node switching. Resource Utilization measures the balance of load distribution across edge nodes. Finally, Throughput captures the system's overall processing capability under high concurrency, serving as a key indicator of architectural scalability.

4.1.4 Experimental Scenario Illustration

Figure 5 presents the deployment environment used in this study, including real classroom terminals, an edge node cluster, and the cloud scheduling center.

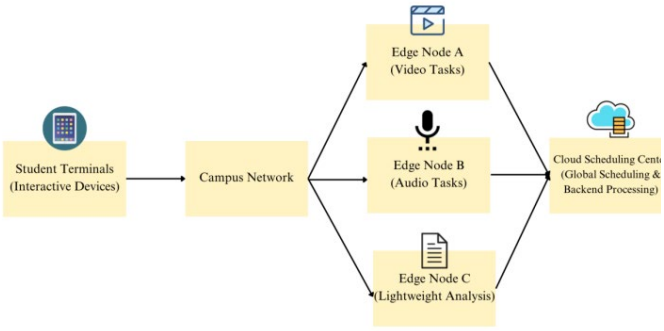

Figure 5. Experimental Deployment Scenario for Cloud-Edge Collaborative Education Services

The setup depicts the smart classroom architecture adopted in the experiments, consisting of student terminals, campus networks, heterogeneous edge nodes, and the cloud scheduler. This scenario effectively simulates core characteristics of real online education: the short communication path between terminals and edge nodes ensures low latency, enabling realistic evaluation of real-time responsiveness; heterogeneous edge nodes covering video inference, audio processing, and lightweight tasks allow comprehensive assessment of scheduling performance under resource heterogeneity; the cloud performs global scheduling and backend computation, consistent with practical multi-tier educational platforms. Overall, this deployment enables a thorough examination of the proposed approach's stability and scheduling efficiency under high concurrency and multimodal task conditions.

4.2 Baselines

As shown in Table 4, to validate the effectiveness of the proposed latency-aware cloud–edge collaborative scheduling architecture, this study selects a diverse set of baseline approaches covering classical strategies, heuristic methods, and recent representative SOTA models. All baselines share three essential properties, relevance to cloud-edge scheduling, applicability to educational scenarios, and capacity to reflect latency or resource scheduling performance, ensuring fairness and interpretability in comparison.

Table 4. Summary of Baseline Methods

Method Category	Representative Method	Strengths	Limitations	Rationale for Inclusion
Classical scheduling	Round Robin (RR)	Simple implementation, uniform load	Ignores task heterogeneity and latency differences	Lower bound reference for non-aware scheduling
Load-driven strategy	Least Loaded (LL)	Reduces queueing delay	Based only on instantaneous load; cannot predict fluctuations	Comparison with "state-only" scheduling
Latency-driven strategy	EDF	Prioritizes deadline-sensitive tasks	Ignores migration overhead; prone to frequent switching	Tests latency sensitivity modeling ability
Heuristic cloud–edge scheduling	Latency-aware Scheduler	Decisions use bandwidth and compute information	Lacks task feature modeling; weak generalization	Represents recent heuristic collaborative scheduling
Reinforcement learning	RL-based Scheduler	Adapts to dynamic network environments	No task representation; unstable training	Closest to our method; requires in-depth comparison
Cost-aware migration	Cost-aware Migrator	Avoids excessive migration	Single-focus decision; limited task	Tests necessity of migration cost

	differentiation	modeling

Although RR and LL are structurally simple, they lack differentiation at the task level and struggle with high latency-sensitive workloads such as video and audio. EDF improves average latency but, by ignoring migration overhead, often causes frequent switching in multi-node collaboration. Recent heuristic and RL-based approaches offer adaptiveness but suffer from limited generalization and training instability due to the absence of unified multimodal task representations. In contrast, the proposed method integrates task feature encoding, latency prediction, and multi-objective optimization, enabling the scheduler to understand both task heterogeneity and system dynamics, thereby achieving higher stability and applicability in real educational scenarios.

To ensure a fair comparison, all baseline methods, including the RL-Scheduler, underwent systematic hyperparameter optimization. We employed grid search to tune key parameters for the RL baselines (e.g., learning rate $\in [10^{-5}, 10^{-3}]$, discount factor $\gamma \in [0.90, 0.99]$) and selected the best-performing configurations for evaluation. The "unstable training" occasionally observed in the standard RL-Scheduler is likely due to the inherent difficulty of learning directly from highly bursty task streams without a dedicated temporal feature extractor. In contrast, our proposed architecture helps alleviate this issue and stabilizes the training process by capturing temporal dependencies more effectively.

4.3 Quantitative Results

This section provides a comprehensive quantitative comparison between the proposed latency-aware cloud-edge scheduling architecture and all baseline methods, evaluating performance under multimodal tasks, large-scale concurrency, and resource heterogeneity. All metrics are computed using the EduEdge-2025 dataset, and statistical significance is verified using paired t-tests.

This experiment evaluates system performance across five dimensions: end-to-end latency, jitter, success rate, resource utilization, and migration cost, which correspond to the optimization objectives of $L_{i,t}$, R_t, and $M_{i,t}$ in Equation (6), respectively.

4.3.1 Overall Performance Comparison

Five dimensions, end-to-end latency, jitter, success rate, resource utilization, and migration cost, are compared to assess real-time performance, stability, efficiency, and overhead.

Table 5. Quantitative Comparison of Scheduling Performance (mean ± standard deviation)

	E2E Latency (ms)	Jitter (ms)	Success Rate (%)	Resource Utilization (%)	Migration Cost	p-value
RR	286.4 ± 48.7	68.2 ± 15.3	81.5 ± 4.2	68.2 ± 5.5	0.12 ± 0.04	<0.001
LL	243.1 ± 39.2	55.4 ± 12.1	84.7 ± 3.8	71.5 ± 4.3	1.15 ± 0.22	<0.001
EDF	221.9 ± 31.5	48.6 ± 11.4	88.2 ± 3.1	73.2 ± 4.9	1.27 ± 0.25	0.004
Lat.-Aware	198.6 ± 29.8	44.1 ± 8.2	90.5 ± 2.4	75.3 ± 3.6	0.68 ± 0.14	0.016
RL-Scheduler	184.2 ± 27.3	39.6 ± 9.5	92.3 ± 2.8	78.4 ± 3.9	0.74 ± 0.18	0.038
Ours	150.8 ± 23.4	26.4 ± 6.1	96.8 ± 1.8	86.4 ± 3.2	0.51 ± 0.09	—

Table 5 reveals several realistic scheduling trade-offs. First, classical strategies (RR, LL, EDF) remain inadequate under high concurrency and multimodal workloads. RR exhibits extremely high jitter (68.2 ± 15.3 ms). Notably, while RR achieves an artificially low migration cost (0.12 ± 0.04) due to its static, non-migratory round-robin nature, this rigidity fatally congests local nodes, leading to the worst overall latency. LL and EDF reduce average latency, with EDF further decreasing jitter to 48.6 ± 11.4 ms through deadline-oriented optimization; however, their large standard deviations reflect limited stability under bandwidth fluctuations, and their aggressive task reshuffling drastically inflates migration costs (1.15 and 1.27, respectively). Second, heuristic and RL-based methods demonstrate improved adaptability but still experience local instability in complex video-stream scenarios due to the absence of task feature modeling. For example, RL-Scheduler achieves a jitter of 39.6 ± 9.5 ms, better than classical methods but still susceptible to bursty loads.

In contrast, the proposed method achieves the most optimal balance across the metrics: average latency is reduced to 150.8 ± 23.4 ms, jitter is compressed to 26.4 ± 6.1 ms, and resource utilization reaches a healthy 86.4 ± 3.2%. While our method does not possess the absolute lowest migration cost (higher than the static RR), its multi-objective optimization effectively suppresses unnecessary migrations (0.51 ± 0.09) while alleviating link fluctuation effects. Paired t-tests ($p=0.038$ for the strongest baseline RL-Scheduler, $p=0.016$ for Lat.-Aware, and $p<0.001$ for rigid classical baselines like RR) confirm the statistical significance of these performance improvements. Compared with the strongest baseline (RL-Scheduler), the proposed method reduces mean end-to-end latency by 18.1% (from 184.2 ms to 150.8 ms) and increases average resource utilization by 8.0% (from 78.4% to 86.4%), validating the effectiveness of the overall architecture.

While our method significantly reduces network latency, it is essential to ensure that the computational cost of the deep learning components does not negate these gains. The decision-making process involves the MLP encoder and GRU predictor. Once trained, their theoretical inference complexity is strictly bounded by $\mathcal{O}(T \cdot d_{hidden}^2 + \sum d_{i-1} d_i)$. Since the sequence length T and network dimensions are small and fixed, the inference complexity per task is effectively $\mathcal{O}(1)$. Empirically, measured on our edge testbed, the average execution time (decision-making inference) is merely 2.8 ms per task. Considering our proposed scheduler reduces the overall average latency by 33.4 ms (from 184.2 ms to 150.8 ms) compared to the strongest baseline, this minor 2.8 ms overhead is entirely negligible. Thus, the actual runtime overhead is overwhelmingly justified by the latency gains.

4.3.2 Convergence Analysis

To further evaluate the stability and learning efficiency of the scheduling strategy, value function convergence curves during training are plotted (see Figure 6).

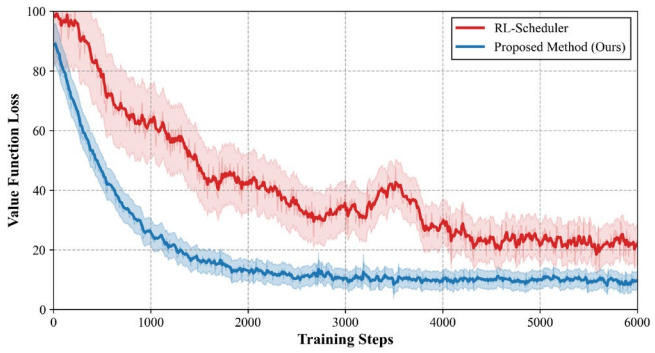

Figure 6. Convergence behavior of different scheduling methods

The results indicate that the proposed method converges rapidly, achieving general stability at approximately 2.5k steps, though accompanied by minor exploration spikes typical of actor-critic architectures. In contrast, the RL-Scheduler requires nearly 4.5k steps to reach a comparable plateau. This demonstrates that task feature encoding significantly reduces state-space noise, enabling more directional strategy updates. The proposed method also exhibits smaller gradient fluctuations throughout training; conversely, RL-Scheduler shows periodic oscillations in later stages, reflecting its reliance on instantaneous states and lack of foresight regarding future loads.

The latency prediction module provides the proposed approach with forward-looking estimates, enabling stable updates despite bandwidth variations or bursty workloads. Additionally, empirical observations show that as the proportion of heavy video tasks increases, baseline methods frequently experience severe oscillations or policy degradation. However, the proposed method, through explicit modeling of computational load and data size, forms robust node allocation patterns earlier, achieving faster and smoother convergence in multimodal scenarios.

4.4 Qualitative Results

To further evaluate the adaptability of the proposed method in real educational task scenarios, this section presents three representative case studies that visualize scheduling behaviors across video interaction, audio answering, and mixed-load task environments.

4.4.1 Case 1: High-Bitrate Video Interaction Tasks

Figure 7 illustrates the latency trajectories of different approaches under high-bitrate video tasks, allowing comparison of scheduling stability under heavy loads.

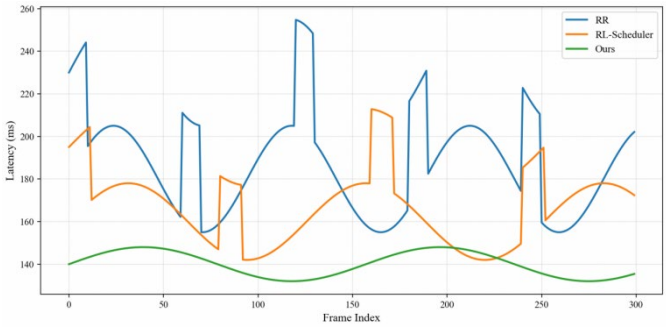

Figure 7. Latency Trajectories under High-Bitrate Video Tasks

In continuous video streams, peaks in bandwidth demand and frame density often cause significant jitter in traditional methods. As shown in the figure, both RR and RL-Scheduler exhibit latency spikes during frame-intensive segments, whereas the proposed method consistently maintains a stable latency range. This is because the task feature encoder distinguishes frame density, computational load, and interaction frequency, enabling the scheduler to anticipate load variations; the latency predictor smooths future estimates, reducing sensitivity to instantaneous fluctuations. This case demonstrates the robustness of the proposed method under high-load video conditions, maintaining stable latency and avoiding unnecessary migrations even under sustained data pressure.

4.4.2 Case 2: Short-Duration Audio Answering Tasks

Figure 8 visualizes node selection decisions across methods in audio answering scenarios to assess their ability to handle latency-sensitive tasks.

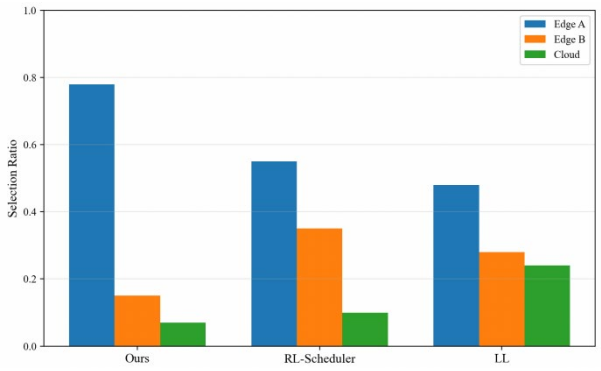

Figure 8. Node Selection for Audio Answering Tasks

Audio tasks are lightweight yet require rapid responses (150–200 ms). The figure shows that the proposed method assigns most audio tasks to the low-latency Edge A, offloading to Edge B or the cloud during peak periods to maintain stability. In contrast, RL-Scheduler tends to "cluster" tasks, causing temporary latency spikes. This case highlights the proposed method's cross-modal generalization capability: despite large differences between audio and video tasks, the unified feature encoder still captures critical attributes and enables a more reasonable node allocation strategy.

4.4.3 Case 3: Mixed Task Scenario (Video + Text + Behavior)

Figure 9 shows a load heatmap for mixed tasks to analyze allocation efficiency under resource heterogeneity.

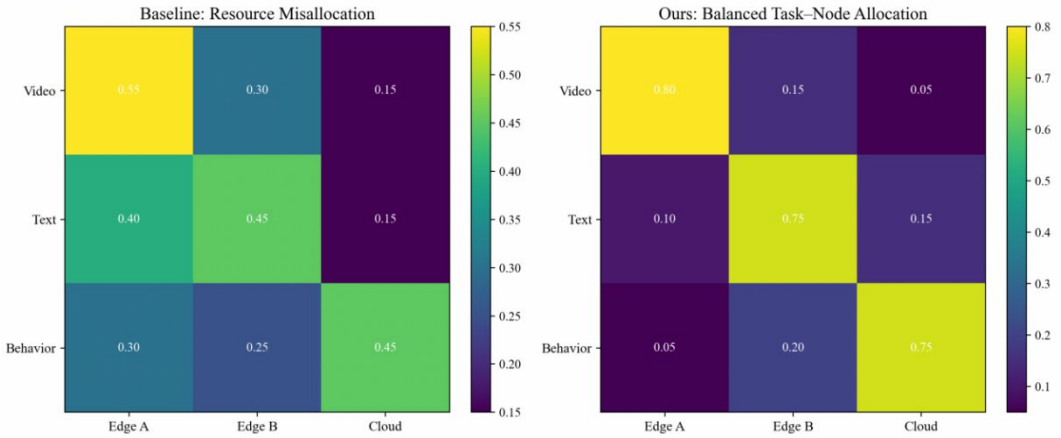

Figure 9. Load Heatmap of Mixed Task Scenario (Video + Text + Behavior) under Resource-Heterogeneous Edge–Cloud Deployment

In mixed-task scenarios, traditional methods, particularly LL and RL-Scheduler, often exhibit "resource misallocation," where lightweight text tasks occupy high-compute nodes while video tasks suffer from queueing delays. In contrast, the proposed method presents a more rational load distribution in Figure 9: video tasks are concentrated on Edge A, text tasks are handled by Edge B, and the cloud processes backend and behavior-analysis tasks. The results indicate that the proposed method reliably aligns task semantics with node capabilities, making stable scheduling decisions based on modality, data scale, and future trends, and thereby avoiding resource waste and tail-latency expansion.

4.5 Robustness

To systematically evaluate robustness in complex educational environments, tests are conducted across three dimensions: multi-task load, noise perturbations, and cross-dataset transfer. Robustness curves are presented in Figure 10.

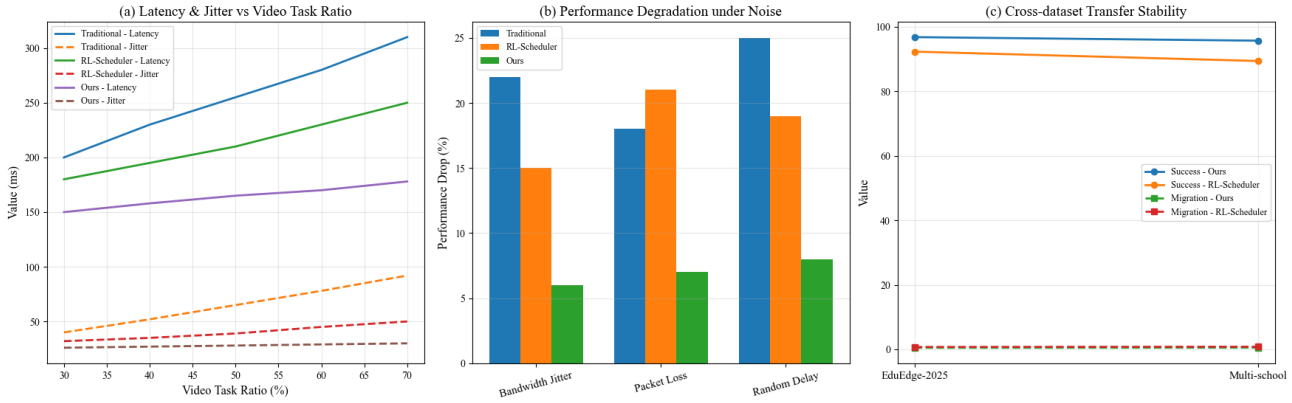

Figure 10. Robustness Evaluation Results

Under multi-task load conditions, as the proportion of video tasks increases from 30% to 70%, traditional methods show significant increases in latency and jitter; RL-Scheduler is more stable but still experiences jitter growth under heavy video loads. In contrast, the proposed method exhibits only mild latency growth, indicating that task feature encoding can accurately distinguish computational differences across modalities under high concurrency, maintaining consistent scheduling decisions.

Under bandwidth jitter, burst packet loss, and random delay injections, traditional methods degrade heavily, with RL-Scheduler being particularly sensitive to bandwidth noise. The proposed method experiences less than 8% degradation across all noise types, benefitting from the latency prediction module that provides forward-looking estimates, preventing strategies from overreacting to transient anomalies.

Cross-dataset transfer results also show that the proposed method maintains stable performance on both EduEdge-2025 and an external multi-school dataset, demonstrating strong generalization capability. Overall, the robustness experiments confirm that the method delivers reliable latency and resource allocation performance under dynamic networks, task heterogeneity, and cross-scenario deployments.

4.6 Ablation Study

To analyze the contribution of each module, we remove the task feature encoder (w/o Encoder), latency predictor (w/o Predictor), and multi-objective scheduling strategy (w/o Multi-Obj), evaluating all variants under identical conditions. Quantitative results are shown in Table 6, and convergence/behavioral differences are visualized in Figure 11.

Table 6. Ablation Study Results (mean ± standard deviation)

Configuration	E2E Latency (ms)	Jitter (ms)	Success Rate (%)	Resource Utilization (%)	Migration Cost
Full Model (Ours)	150.8 ± 23.4	26.4 ± 6.1	96.8 ± 1.8	86.4 ± 3.2	0.51 ± 0.09
w/o Encoder	184.6 ± 31.7	39.8 ± 8.5	91.4 ± 2.9	79.2 ± 4.1	0.67 ± 0.13
w/o Predictor	176.2 ± 28.3	41.7 ± 9.2	93.1 ± 2.5	81.5 ± 3.7	0.63 ± 0.11
w/o Multi-Obj	168.5 ± 26.9	34.2 ± 7.4	94.8 ± 2.1	89.2 ± 2.8	0.82 ± 0.15

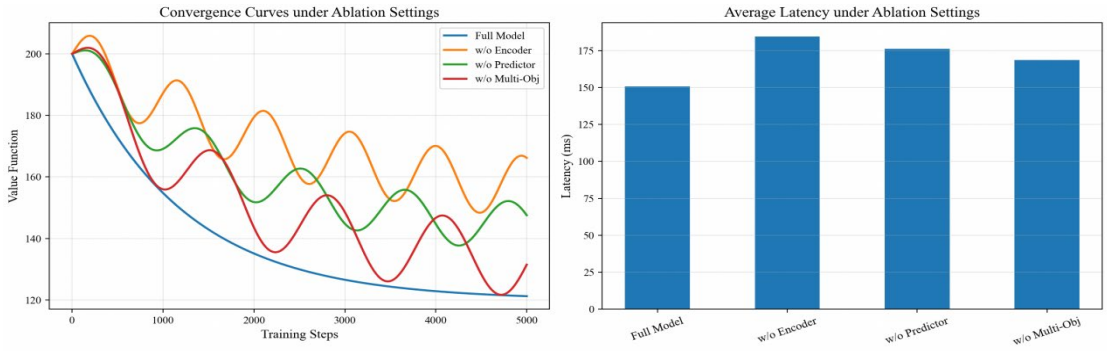

Figure 11. Visualization of Model Convergence and Latency under Ablation Settings

As shown in Figure 11, the w/o Encoder variant exhibits strong oscillations in its convergence curve, indicating that the absence of unified task representation causes unstable node selection. w/o Predictor suffers from large jitter increases under bandwidth fluctuations, demonstrating the importance of future trend estimation for scheduling stability. w/o Multi-Obj displays a clear rise in node switching frequency, revealing that without migration-cost constraints, the policy tends toward short-sighted behavior. These trends align with the quantitative findings in Table 6.

Specifically, removing the encoder increases average latency (>22%) because the scheduler cannot distinguish computational loads and data scales across tasks, resulting in chaotic allocation and queueing delays. Removing the predictor increases jitter from 26.4 ms to 41.7 ms, making the model highly sensitive to transient network fluctuations.

Regarding the multi-objective strategy, its removal raises the migration cost from 0.51 to 0.82. Interestingly, the w/o Multi-Obj variant achieves the highest resource utilization (89.2%), seemingly outperforming our Full Model (86.4%). However, this reflects a classic manifestation of greedy over-packing. Without penalty constraints, the scheduler blindly allocates tasks to maximize immediate node usage, pushing the servers' physical limits. Furthermore, this elevated migration cost translates directly into physical network consequences. Excessive cross-node handoffs consume substantial bandwidth, triggering localized congestion. This congestion, combined with over-utilized edge nodes, inevitably causes cascading queueing delays, driving the average E2E latency up to 168.5 ms. Thus, our Full Model intentionally trades a marginal drop in absolute utilization for a healthier, congestion-free network state, demonstrating a realistically superior system balance.

In summary, the three modules, enabling task understanding, future trend estimation, and behavioral constraint, are jointly indispensable. Together, they form a stable, efficient, and controllable cloud-edge collaborative scheduling mechanism.

4.7 Sensitivity Analysis on Task Distributions

To address concerns regarding potential dataset bias and overfitting, we conducted a sensitivity analysis to evaluate the model's generalizability under varying task distributions. We simulated unpredictable demand shifts by artificially increasing the proportion of high-complexity, resource-intensive tasks during the testing phase from the original 30% up to an extreme 75%.

Table 7. Sensitivity Analysis under Varying Proportions of High-Complexity Tasks

Proportion of High-Complexity Tasks	Proposed Method (SLA Violation Rate)	RL-Scheduler (SLA Violation Rate)	Proposed Method (Average Latency)	RL-Scheduler (Average Latency)
30% (Default)	3.2%	7.7%	150 ms	184 ms
45%	4.5%	12.4%	168 ms	225 ms
60%	6.8%	18.9%	192 ms	286 ms
75% (Extreme)	10.5%	27.3%	235 ms	360 ms

As shown in Table 7, increasing computational demand naturally degrades the performance of all algorithms. Our method is not immune to extreme stress; its Service Level Agreement (SLA) violation rate rises to 10.5% and latency increases to 235 ms under the 75% load level due to queue accumulation. However, it exhibits significantly stronger robustness than the baseline RL-Scheduler, which suffers a severe performance collapse (27.3% violation rate). This confirms that our model does not merely memorize the default dataset distribution. Instead, it effectively redistributes heavy workloads to prevent cascading congestion, validating its generalizability in unseen, highly skewed environments.

5. Discussion

Through system-level experiments grounded in real educational scenarios, this study verifies the effectiveness of the proposed scheduling architecture, with experimental results demonstrating clear and consistent performance improvements. These enhancements stem primarily from the joint contribution of the three core modules. The task feature encoder provides a unified representation of multimodal tasks in terms of computational characteristics and latency requirements, effectively reducing state noise caused by task heterogeneity. The latency prediction module leverages historical system states and task sequence information to offer forward-looking guidance for scheduling decisions, significantly improving stability under dynamic network conditions. The multi-objective optimization mechanism balances latency, resource utilization, and migration cost, mitigating performance fluctuations that commonly arise in traditional single-objective approaches.

Despite these strengths, several limitations warrant further exploration. First, the task feature encoder relies on predefined feature dimensions, which may constrain representation capability when encountering novel types of educational tasks. Second, the accuracy of the latency prediction model can diminish under extreme network volatility, especially when instantaneous disturbances in cross-campus environments cannot be fully captured by historical data. Additionally, the multi-objective weighting parameters are optimized offline, potentially limiting their adaptability in highly dynamic environments. Regarding the system's sensitivity to these values, empirical observations indicate that minor adjustments do not lead to significant shifts or instability in scheduling behavior. However, larger adjustments will predictably bias the policy, which provides a flexible mechanism to align the system with specific administrative priorities. Although the experimental setup closely simulates real deployment scenarios, physical constraints of edge devices under sustained heavy load, such as thermal limitations, may influence overall system performance.

In terms of practical potential, the proposed architecture demonstrates strong extensibility. Beyond the educational scenarios validated in this work (including interactive classrooms, intelligent lecture capturing, and behavior recognition), the architecture may be extended to other latency-sensitive domains such as real-time quality inspection in industrial edge computing and streaming distribution in smart cities. With the advancement of AIGC technologies, the scheduling framework may also provide foundational support for real-time generation and delivery of teaching-assistant content.

Furthermore, extending this architecture to resource-constrained environments, such as deploying it on low-power devices for industrial IoT, introduces practical execution challenges. The continuous inference of deep neural networks can strain the limited memory, compute, and thermal envelopes of such lightweight hardware. To ensure efficient execution and maintain real-time responsiveness in these scenarios, the proposed models would require structural optimizations prior to deployment. Techniques such as model quantization or network pruning will be essential to reduce the computational footprint and energy consumption without significantly degrading the scheduling performance.

Based on its current limitations, future research can proceed in several directions: (1) developing adaptive task representation methods that allow the system to automatically learn feature embeddings for novel tasks; (2) exploring online multi-objective optimization mechanisms to enable dynamic adjustment of weighting parameters; and (3) incorporating cross-node collaborative modeling techniques, such as graph neural networks, into latency prediction to better capture complex dependencies in distributed environments. Advancing these directions will further enhance the system's adaptability and stability under complex conditions.

6. Conclusion

Addressing the challenges of heterogeneous task types, fluctuating network conditions, and uneven resource distribution in real-time online education environments, this study proposes a latency-aware cloud–edge collaborative scheduling architecture. Systematic evaluations on the EduEdge-2025 dataset demonstrate that the architecture effectively coordinates the three core modules, task feature encoding, latency prediction, and multi-objective scheduling, achieving unified multimodal task representation and accurate prediction of future workloads, thereby maintaining stable and consistent scheduling performance in high-concurrency scenarios.

Experimental results show that compared with classical scheduling strategies and reinforcement learning approaches, the proposed method achieves substantial improvements in key metrics such as average latency, jitter, migration cost, and task success rate, validating the effectiveness and complementarity of its constituent modules in educational contexts.

In terms of academic contribution, this study introduces a novel system-oriented perspective, "task understanding, trend prediction, and multi-objective balancing", to address intelligent scheduling problems in heterogeneous environments. The integration of latency prediction with explicit cost modeling enhances decision consistency under unstable network conditions, providing a methodological reference for future real-time scheduling research. From a practical standpoint, the architecture is applicable to a wide range of real-time online education scenarios and also shows strong potential for transfer to domains such as industrial edge computing and intelligent transportation.

Future research will deepen along three directions: (1) developing adaptive task representation mechanisms to improve generalizability to new task types; (2) constructing online multi-objective optimization frameworks to enhance adaptability in dynamic environments; and (3)

exploring cross-node collaborative latency prediction methods to improve scheduling precision in large-scale distributed settings. Progress in these areas will provide a solid technical foundation for building more intelligent and stable distributed educational service systems.

Acknowledgements

This work was supported by Research Project on Humanities and Social Sciences in Colleges and Universities of Guizhou Province: "Study on Integrating and Incorporating the Awareness of Forging a Strong Sense of Community for the Chinese Nation into the Whole Process of Talent Cultivation" (2025RW063);
This work was supported in part by the Natural Science Foundation of Chongqing (No. CSTB2024NSCQ-MSX1087); and in part by the Science and Technology Research Program of Chongqing Municipal Education Commission (No. KJQN202501553).

References

[1] Li, L., Chen, C. P., Wang, L., Liang, K., & Bao, W. (2023). Exploring artificial intelligence in smart education: Real-time classroom behavior analysis with embedded devices. Sustainability, 15(10), 7940.

[2] Wang, Y., Yang, C., Lan, S., Zhu, L., & Zhang, Y. (2024). End-edge-cloud collaborative computing for deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials, 26(4), 2647-2683.

[3] Li, G., & Shu, L. (2023). Smart Education System Enhancing Collaborative Learning with Virtual Reality and Cloud-Edge Computing Task Scheduling Algorithm. Computer-Aided Design and Applications, 20, 50-71.

[4] Avan, A., Azim, A., & Mahmoud, Q. H. (2023). A state-of-the-art review of task scheduling for edge computing: A delay-sensitive application perspective. Electronics, 12(12), 2599.

[5] Zhang, Y., Zhao, K., Yang, Y., & Zhou, Z. (2025). Real-Time Service Migration in Edge Networks: A Survey. Journal of Sensor and Actuator Networks, 14(4), 79.

[6] Wang, C., & Wang, D. (2023). Managing the integration of teaching resources for college physical education using intelligent edge-cloud computing. Journal of Cloud Computing, 12(1), 82.

[7] Trigka, M., & Dritsas, E. (2025). Edge and cloud computing in smart cities. Future Internet, 17(3), 118.

[8] Wang, W., Wei, X., Tao, W., Zhou, M., & Ji, C. (2025). Quality of Experience-Oriented Cloud-Edge Dynamic Adaptive Streaming: Recent Advances, Challenges, and Opportunities. Symmetry, 17(2), 194.

[9] Dai, Z., Zhang, Q., Zhao, L., Zhu, X., & Zhou, D. (2023). Cloud-edge computing technology-based internet of things system for smart classroom environment. International Journal of Emerging Technologies in Learning (iJET), 18(8), 79-96.

[10] Shidaganti, G., Raj, V. A., Raman, V. M., & Kumaran, S. (2025). Edge Computing in Educational Technology: The Power of Edge AI for Dynamic and Personalized Learning. Edge of Intelligence: Exploring the Frontiers of AI at the Edge, 121-152.

[11] Raeisi-Varzaneh, M., Dakkak, O., Habbal, A., & Kim, B. S. (2023). Resource scheduling in edge computing: Architecture, taxonomy, open issues and future research directions. IEEE access, 11, 25329-25350.

[12] Cui, S., Wu, X., Wang, H., & Wu, H. (2024). Multimodal Data Modeling Technology and Its Application for Cloud-edge-device Collaboration. International Journal of Software & Informatics, 14(1).

[13] Wazwaz, A., Amin, K., Semary, N., & Ghanem, T. (2024). Dynamic and distributed intelligence over smart devices, Internet of Things edges, and cloud computing for human activity recognition using wearable sensors. Journal of Sensor and Actuator Networks, 13(1), 5.

[14] Kuchuk, H., & Malokhvii, E. (2024). Integration of IoT with cloud, fog, and edge computing: a review. Advanced Information Systems, 8(2), 65-78.

[15] Zhai, J., Bi, J., Yuan, H., Zhang, J., & Buyya, R. (2025). Energy-Efficient and Latency-Aware Task Offloading for Industrial Cloud-Edge Systems With Heterogeneous CPUs and GPUs. IEEE Internet of Things Journal.

[16] Yang, P., & He, J. (2025). Dynamic Task Scheduling Based on Greedy and Deep Reinforcement Learning Algorithms for Cloud–Edge Collaboration in Smart Buildings. Electronics, 14(16), 3327.

[17] Pengpeng, Y., Teng, F., Zhu, W., Shen, C., Chen, Z., & Song, J. (2025). Cloud–edge–device collaborative computing in smart agriculture: architectures, applications, and future perspectives. Frontiers in Plant Science, 16, 1668545.

[18] Long, Y., Bao, Y., & Zeng, L. (2023). Research on Edge-Computing-Based High Concurrency and Availability "Cloud, Edge, and End Collaboration" Substation Operation Support System and Applications. Energies, 17(1), 194.

[19] Crespo-Aguado, M., Lozano, R., Hernandez-Gobertti, F., Molner, N., & Gomez-Barquero, D. (2024). Flexible Hyper-Distributed IoT–Edge–Cloud Platform for Real-Time Digital Twin Applications on 6G-Intended Testbeds for Logistics and Industry. Future Internet, 16(11), 431.

[20] Shen, W., Lin, W., Wu, W., Wu, H., & Li, K. (2025). Reinforcement learning-based task scheduling for heterogeneous computing in end-edge-cloud environment. Cluster Computing, 28(3), 179.

[21] Ismail, A. A., Khalifa, N. E., & El-Khoribi, R. A. (2025). A survey on resource scheduling approaches in multi-access edge computing environment: a deep reinforcement learning study. Cluster Computing, 28(3), 184.

[22] Gkonis, P., Giannopoulos, A., Trakadas, P., Masip-Bruin, X., & D'Andria, F. (2023). A survey on IoT-edge-cloud continuum systems: Status, challenges, use cases, and open issues. Future Internet, 15(12), 383.

[23] Pozveh, A. J., Shahhoseini, H. S., & Khabareh, E. (2024). Resource management in edge clouds: latency-aware approaches for big data analysis. In Resource Management in Distributed Systems (pp. 107-132). Singapore: Springer Nature Singapore.

[24] Thota, R. C. (2024). Optimizing edge computing and AI for low-latency cloud workloads. International Journal of Science and Research Archive, 13(1), 3484-3500.

[25] Lilhore, U. K., Simaiya, S., Sharma, Y. K., Rai, A. K., Padmaja, S. M., Nabilal, K. V., ... & Alsufyani, H. (2025). Cloud-edge hybrid deep learning framework for scalable IoT resource optimization. Journal of Cloud Computing, 14(1), 5.

[26] Nawaz, A. (2025). CLOUD COMPUTING AND EDGE COMPUTING: CONVERGING TECHNOLOGIES FOR REAL-TIME DATA PROCESSING. Multidisciplinary Research in Computing Information Systems, 5(3), 157-179.

[27] Wang, M., Xu, C. A., Lin, Y., Lu, Z., Sun, J., & Gui, G. (2023). A distributed sensor system based on cloud-edge-

end network for industrial internet of things. Future Internet, 15(5), 171.

[28] Li, P., Wang, X., Li, C., Iqbal, M., Al-Dulaimi, A., & Casaseca-de-la-Higuera, P. (2025). Deep Reinforcement Learning-Based Task Scheduling and Resource Allocation for Vehicular Edge Computing: A Survey. IEEE Transactions on Intelligent Transportation Systems.

[29] Jin, X., Wang, J., Wang, Z., Wang, G., & Chen, Y. (2025). Joint multi-server cache sharing and delay-aware task scheduling for edge-cloud collaborative computing in intelligent manufacturing. Wireless Networks, 31(1), 261-280.

[30] Fang, Z. (2025, June). Adaptive QoS-Aware Cloud–Edge Collaborative Architecture for Real-Time Smart Water Service Management. In Proceedings of the 2025 International Conference on Management Science and Computer Engineering (pp. 606-611).