Agent-Based LSTM Forecasting for Efficient Resource Allocation in Edge Infrastructures

Fábio Diniz Rossi[1,2], Koushikur Islam Shohag[2], Rodrigo Neves Calheiros[2]

[1]Federal Institute Farroupilha - Brazil.
[2]Western Sydney University - Australia.

Contributing authors: fabio.rossi@iffarroupilha.edu.br; k.shohag@westernsydney.edu.au; r.calheiros@westernsydney.edu.au;

Abstract

Edge computing has emerged as a paradigm that brings computation closer to data sources, reducing latency and alleviating core network congestion. However, as workloads become increasingly dynamic and heterogeneous, efficient resource allocation in edge environments requires anticipating demand surges while avoiding over-provisioning. Traditional reactive or threshold-based strategies often respond only after saturation occurs, leading to Service Level Agreement (SLA) violations, inefficient task allocation, and degraded Quality of Service (QoS). To address these challenges, we propose an agent-based AI architecture for proactive resource allocation, integrating PRIMAL, a pre-trained Long Short-Term Memory (LSTM) forecasting model, to predict CPU and memory usage in edge nodes. This predictive capability enables the system to anticipate saturation events and trigger reallocation decisions before they occur, improving system stability under bursty and high-contention conditions. The architecture is designed as a modular, multi-agent framework, supporting seamless integration of alternative forecasting mechanisms, reinforcement learning policies, and multi-objective optimization strategies. Experimental results show that PRIMAL achieves up to 12% higher task acceptance rates and significantly reduces sustained SLA violations, while maintaining competitive throughput and avoiding unnecessary task rejections. These findings indicate that agentic AI-driven forecasting can deliver tangible benefits for future autonomous edge infrastructures.

Keywords: Edge Computing, Resource Allocation, Agentic AI, LSTM Forecasting, Proactive Orchestration, Quality of Service (QoS).

1 Introduction

The rapid proliferation of latency-sensitive applications in domains such as autonomous driving, industrial automation, immersive virtual reality, and real-time video analytics has placed unprecedented demands on edge computing infrastructures [1]. Unlike traditional cloud deployments, which operate in centralized, homogeneous, and resource-rich environments, edge infrastructures are inherently more distributed and diverse. They encompass a wide spectrum of devices, from low-power embedded systems and IoT gateways to high-performance microservers, deployed in close proximity to data sources and end users. This diversity means that while certain edge nodes may have substantial computational capacity, others may operate under tighter constraints in processing power, memory, or network bandwidth. The heterogeneous and geographically dispersed nature of these systems introduces unique challenges in coordinating resource allocation, maintaining consistent performance, and avoiding overload under highly variable workload conditions [2].

Conventional resource allocation strategies in edge environments are predominantly reactive, responding to changes in system load only after performance degradation becomes evident [3]. While such approaches are straightforward to implement, they inherently suffer from latency in decision-making, allowing transient overloads to propagate before corrective measures can be applied. Proactive methods, on the other hand, attempt to anticipate future load conditions, often leveraging statistical forecasting or moving averages, to act before saturation occurs. However, these methods are typically constrained by simplistic predictive models that struggle to capture the complex temporal dependencies and nonlinear behaviors inherent in multi-tenant, multi-application edge workloads. This limitation is particularly pronounced when workload patterns deviate from historical trends, such as during abrupt shifts in demand caused by unexpected user behavior or external events [4].

Recent advances in Agentic Artificial Intelligence (Agentic AI), autonomous, goal-driven agents capable of perception, reasoning, and adaptive decision-making, offer a promising paradigm shift for edge resource allocation [5]. By embedding predictive intelligence directly into each edge node, these agents can monitor local performance metrics, infer future saturation risks, and proactively adjust resource allocation strategies without centralized coordination. Unlike static forecasting approaches, Agentic AI architectures can integrate heterogeneous sources of telemetry, continuously refine their models in operation, and adapt decision thresholds in response to evolving workload characteristics. This enables not only faster reaction times but also a reduction in unnecessary task rejections, leading to improved resource utilization and higher throughput [6].

In this work, we propose PRIMAL (Proactive Resource Intelligent Management with Agentic Learning), an agent-based AI architecture for proactive resource allocation in edge environments. PRIMAL integrates a pre-trained Long Short-Term Memory (LSTM) forecasting model [7] to capture long-term temporal correlations in CPU and memory usage patterns, enabling accurate short-term load forecasting and intelligent pre-allocation of resources. We evaluate both online-trained and pre-trained LSTM agents within the same Agentic AI framework, performing a systematic comparison against traditional reactive and statistical predictive baselines. Through

extensive simulations with realistic workload traces, our results show that PRIMAL can achieve up to 12% higher task acceptance rates and significantly reduce sustained SLA violations, while maintaining competitive throughput and avoiding unnecessary rejections, even under bursty and high-contention conditions.

The main contributions of this work are threefold. First, we introduce an Agentic AI architecture tailored for resource allocation in edge computing, combining autonomous decision-making with predictive intelligence. Second, we design and implement multiple allocation strategies, including reactive, statistical predictive, adaptive hybrid, and LSTM-based agents, enabling a systematic performance comparison. Third, we conduct an in-depth evaluation using heterogeneous workload traces and multiple performance metrics, providing insights into the trade-offs between proactivity, accuracy, and stability in edge resource management. Our findings highlight the potential of Agentic AI to transform resource allocation from a reactive safeguard into a proactive optimization mechanism, paving the way for more resilient and intelligent edge infrastructures.

The remainder of this paper is organized as follows. Section 2 provides the necessary background on edge computing architectures and the role of Agentic AI in resource management. Section 3 formalizes the resource allocation problem, outlining the constraints, performance metrics, and stochastic nature of the workload. Section 4 details PRIMAL, our proposed Agentic AI framework that integrates predictive modeling and adaptive decision-making for proactive allocation in edge environments. Section 5 describes the experimental methodology, including trace generation, baseline algorithms, and evaluation metrics, and presents a thorough discussion of the obtained results. Section 6 reviews related literature, highlighting the differences and advantages of our approach compared to existing solutions. Finally, Section 7 summarizes the key findings and outlines directions for future research.

2 Background

The evolution of edge computing has shifted the locus of computation from centralized data centers toward geographically distributed nodes situated closer to data sources and end users. This architectural shift enables significant reductions in latency, network congestion, and bandwidth costs, while also facilitating compliance with data locality and privacy requirements [8]. However, the benefits of edge computing are coupled with increased complexity in resource allocation, given the heterogeneity of hardware capabilities, the variability of workloads, and the need to maintain Quality of Service (QoS) under dynamic conditions.

In edge computing, resource allocation encompasses the dynamic distribution of CPU, memory, and network resources among competing applications and services [9]. Unlike the cloud, where oversubscription is mitigated by virtually limitless scaling, edge nodes often operate under finite capacity constraints. Even when powerful microservers are deployed, their resources must be shared among multiple tenants and workloads with differing latency sensitivities and Service Level Objectives (SLOs). The unpredictability of demand, ranging from steady telemetry streams to sudden bursts

in event-driven processing, necessitates allocation strategies that are both responsive and predictive.

Traditional allocation methods in the edge can be broadly classified into reactive and proactive approaches [10]. Reactive methods operate by detecting performance degradation, such as CPU overload or memory saturation, and adjusting resource allocations only after Service Level Objective (SLO) violations occur. In contrast, proactive methods rely on forecasting techniques, including statistical models, moving averages, or time-series extrapolation, to anticipate future load conditions and adjust allocations in advance, thereby aiming to prevent violations before they materialize.

While proactive methods offer advantages in preventing overload, their accuracy heavily depends on the quality of the prediction model and its adaptability to non-stationary workloads [11]. This is where Agentic AI introduces a disruptive potential. Agentic AI refers to autonomous, goal-directed AI agents that perceive their environment, reason about possible actions, and adaptively adjust their behavior to achieve predefined objectives. In the context of edge computing, these agents can be embedded directly into each node, granting them the ability to continuously monitor local telemetry, including CPU usage, memory utilization, and network throughput, predict future system states based on historical trends and learned patterns, select allocation policies that minimize SLO violations while maximizing throughput and acceptance rate, and dynamically adapt decision thresholds to workload variations [12].

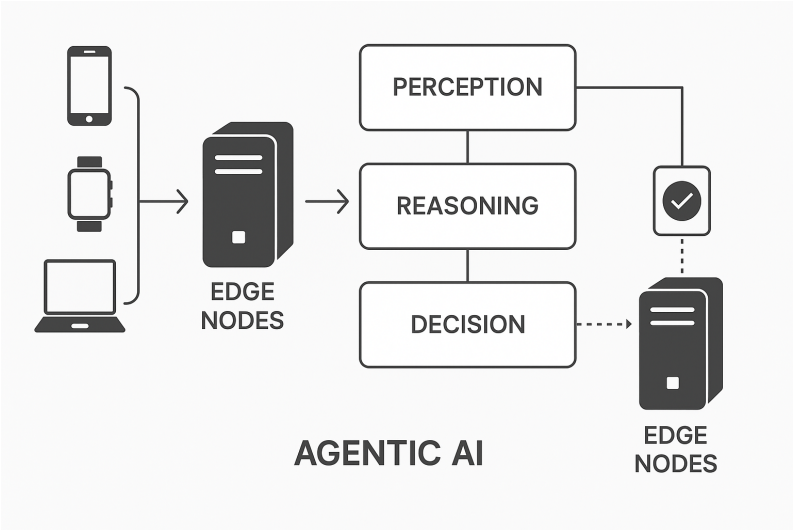


Fig. 1: Conceptual integration of Agentic AI within an edge computing architecture.

Figure 1 illustrates the conceptual integration of Agentic AI within an edge computing architecture. At the lowest layer, a heterogeneous set of edge nodes, ranging from lightweight IoT gateways to microservers, interacts with end devices and local sensors, generating continuous streams of telemetry data, including CPU utilization, memory consumption, and network throughput. This raw telemetry is ingested by the

Perception Module, which preprocesses and normalizes measurements to remove noise and align temporal sampling. The processed data is fed into the Reasoning Module, where predictive models such as Long Short-Term Memory (LSTM) networks analyze historical trends to forecast short-term system load. These predictions incorporate temporal correlations, workload burstiness, and diurnal patterns to estimate resource saturation risk. Based on these forecasts, the Decision Module selects appropriate allocation actions, such as migrating tasks to other nodes, adjusting CPU and memory quotas, or throttling workloads to preserve SLOs. Therefore, the agent operates autonomously and in real time, adapting decision thresholds to workload variations and node capacity changes without centralized orchestration. At the top layer, a coordination framework enables inter-agent communication, allowing multiple edge nodes to share state summaries or predictive alerts. This distributed intelligence supports proactive, cooperative decision-making, enhancing scalability and resilience compared to purely reactive systems.

This architecture contrasts sharply with static rule-based systems. Whereas reactive approaches respond only to threshold breaches, an Agentic AI agent can reason about when to act and how much to adjust resources, even in the absence of immediate overload symptoms [13]. Among the various predictive models applicable in this context, the Long Short-Term Memory (LSTM) network stands out as a recurrent neural network architecture specifically designed to capture long-term dependencies in sequential data [14]. In edge resource allocation, LSTMs can learn temporal correlations in workload traces, such as diurnal usage patterns, periodic spikes, or bursty traffic behaviors, and, by leveraging gated memory cells, mitigate the vanishing gradient problem that hampers traditional RNNs [15].

Two deployment modes are relevant to this work. The first corresponds to online-trained LSTM agents, which continuously update their model parameters during operation, enabling adaptation to novel workloads at the cost of slower initial performance. The second mode involves PRIMAL, our pre-trained LSTM-based agent, trained offline on large-scale workload datasets and capable of delivering accurate predictions immediately upon deployment. While PRIMAL may require fine-tuning for optimal long-term performance, its integration into an Agentic AI framework transforms resource allocation from a reactive safeguard into a proactive optimization process. Instead of merely preventing overload, PRIMAL actively shapes workload acceptance and scheduling to maintain SLO compliance and improve resource efficiency—a behavior reflected in our experimental results, which show measurable gains in acceptance rate and reduction of sustained SLA violations.

3 Problem Definition

We consider an *edge computing environment* composed of a set of N heterogeneous nodes, denoted by $\mathcal{E} = \{e_1, e_2, \ldots, e_N\}$. Each node e_i is characterized by its available compuational capacity, memory, and network bandwidth, represented respectively as

$$C_i^{\text{CPU}}, \quad C_i^{\text{MEM}}, \quad C_i^{\text{NET}},$$

which define hard upper bounds on the resources that can be allocated to any set of tasks hosted by that node.

A continuous stream of tasks $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_M\}$ arrives over time. Each task τ_j is described by its computational demand d_j^{CPU}, its memory demand d_j^{MEM}, a deadline D_j after which it is considered failed, and an associated priority level $p_j \in \{1, \ldots, P\}$, where smaller values indicate higher priority. The priority reflects the criticality of the task in meeting application-specific Service Level Objectives (SLOs).

At each discrete time step t, the allocation policy must decide whether to accept each task and, if so, assign it to a specific node according to the binary variable

$$x_{ij}(t) = \begin{cases} 1, & \text{if task } \tau_j \text{ is allocated to node } e_i \text{ at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

where the decision matrix $X(t) = [x_{ij}(t)]$ must satisfy

$$\sum_{i=1}^{N} x_{ij}(t) \leq 1, \quad \forall j, t,$$

ensuring that each task is assigned to at most one node, while still allowing rejections when no node has enough capacity available. Given the finite resources of each node, allocations must also satisfy the per-resource constraints

$$\sum_{\tau_j \in \mathcal{A}_i(t)} d_j^{\mathrm{CPU}} \leq C_i^{\mathrm{CPU}}, \qquad \sum_{\tau_j \in \mathcal{A}_i(t)} d_j^{\mathrm{MEM}} \leq C_i^{\mathrm{MEM}},$$

where $\mathcal{A}_i(t)$ denotes the set of tasks allocated to node e_i at time t.

The performance of an allocation policy is evaluated through several complementary metrics. Throughput measures the number of tasks completed within their deadlines,

$$\text{Throughput} = \sum_{j=1}^{M} \mathbb{I}\left(\text{completion\_time}(\tau_j) \leq D_j\right),$$

where $\mathbb{I}(\cdot)$ is the indicator function, and higher values indicate more effective resource utilization. The rejection rate quantifies the proportion of tasks refused at arrival due to lack of resources,

$$\text{RejectionRate} = \frac{\sum_{j=1}^{M} \mathbb{I}\left(\tau_j \text{ rejected}\right)}{M},$$

where lower values indicate that more incoming demand is being served. Another key measure is the number of SLO violations, defined as

$$\text{Violations} = \sum_{j=1}^{M} \mathbb{I}\left(L_j > \text{SLO}_j\right),$$

where L_j is the observed latency (completion time) of task τ_j and SLO_j is its latency objective. We distinguish between sustained violations, which persist over time, and severe violations, where latency exceeds SLO_j by more than a given

threshold, as these indicate systemic performance degradation. To capture worst-case load conditions, we monitor high-percentile CPU and memory usage values, namely $p95^{\mathrm{CPU}}$, $p95^{\mathrm{MEM}}$, $p99^{\mathrm{CPU}}$, $p99^{\mathrm{MEM}}$. For proactive approaches such as PRIMAL, we also measure the lead-time, defined as the average anticipation interval between predicting an overload and its actual occurrence

$$\mathrm{LeadTime} = \frac{1}{K} \sum_{k=1}^{K} \left(t_{\mathrm{pred},k} - t_{\mathrm{event},k} \right),$$

where $t_{\mathrm{pred},k}$ is the prediction time and $t_{\mathrm{event},k}$ is the time at which overload is observed. Larger lead-times give the system more opportunity to take preventive actions, such as migrating tasks or adjusting quotas, before performance degrades.

Finally, we define a composite SLO score that aggregates throughput, rejections, and violations into a single optimization objective

$$\mathrm{SLO\_Score} = \alpha \cdot \mathrm{Throughput} - \beta \cdot \mathrm{Rejections} - \gamma \cdot \mathrm{Violations},$$

where α, β, γ are weights reflecting the deployment's operational priorities. To ensure commensurate scales, all three components are min–max normalized per scenario to $[0, 1]$. Unless noted, we use $\alpha = 1.0$, $\beta = 1.0$, and $\gamma = 1.0$. The plotted composite in Fig. 8 uses exactly these settings. The general optimization problem is thus

$$\max_{X(t)} \ \mathrm{SLO\_Score}$$

subject to resource constraints and allocation feasibility. For proactive (Agentic AI) methods like PRIMAL, we additionally aim to

$$\text{maximize LeadTime} \quad \text{s.t.} \quad \text{Violations minimal},$$

ensuring that the system not only reacts to overloads but also anticipates and mitigates them in advance.

The allocation problem is inherently dynamic and stochastic, with workloads exhibiting non-stationary temporal patterns, heterogeneous node capacities, and unpredictable demand bursts. The trade-offs between utilization, SLO compliance, and fairness are further complicated by the need for rapid decision-making under partial knowledge of the system state. These characteristics make the problem particularly well-suited for *Agentic AI* solutions that combine predictive modeling, such as the LSTM-based PRIMAL, with adaptive threshold control to balance accuracy, responsiveness, and efficiency in edge environments.

4 PRIMAL

We propose **PRIMAL** (*Proactive Resource Intelligent Management with Agentic Learning*), an Agentic AI-based resource allocation framework specifically designed for edge computing environments. PRIMAL integrates predictive modeling with adaptive decision-making to anticipate overloads and proactively reallocate tasks, thereby improving Service Level Objective (SLO) compliance while maintaining high resource utilization. Unlike conventional reactive approaches, which only act when thresholds

are breached, or simple predictive methods that rely on fixed models, PRIMAL combines the accuracy of a pre-trained Long Short-Term Memory (LSTM) predictor with adaptive threshold tuning. This combination enables fine-grained, anticipatory control over allocation decisions, making it capable of handling non-stationary workloads while adapting in real time to evolving conditions.

Table 1 summarizes the symbols and variables used throughout the paper to describe the PRIMAL framework, its predictive models, and evaluation metrics. This notation serves as a reference to ensure clarity and consistency in the presentation of algorithms, mathematical formulations, and performance results. By explicitly defining each term, we facilitate reproducibility and allow readers to follow the methodological details without ambiguity.

Table 1: Notation.

Symbol	Meaning
$u_i^{\mathrm{CPU/MEM}}(t)$	observed utilization at node i
$\hat{u}_i^{\mathrm{CPU/MEM}}(t+\Delta)$	predicted utilization (horizon Δ)
$C_i^{\mathrm{CPU/MEM}}$	capacity of node i
$\mathrm{THR}_i^{\mathrm{CPU/MEM}}(t)$	adaptive threshold (EWMA + offset δ)
H	history window length for prediction
Δ	prediction horizon
α	EWMA responsiveness
δ	safety margin offset

The system operates in discrete time steps t, processing incoming tasks τ_j and deciding their allocation $x_{ij}(t)$ to edge nodes e_i. At each step, PRIMAL begins by monitoring the current resource utilization $u_i^{\mathrm{CPU}}(t)$ and $u_i^{\mathrm{MEM}}(t)$, as well as the active task set $\mathcal{A}_i(t)$ on each node. These measurements are then fed into a pre-trained LSTM model, which predicts the short-term future utilization $\hat{u}_i^{\mathrm{CPU}}(t+\Delta)$ and $\hat{u}_i^{\mathrm{MEM}}(t+\Delta)$, capturing both seasonal patterns and bursty variations.

To maintain robustness under workload variability, PRIMAL employs a dynamic threshold mechanism based on an exponentially weighted moving average (EWMA) of recent utilizations:

$$\mathrm{THR}_i(t) = \alpha\, u_i(t) + (1-\alpha)\,\mathrm{THR}_i(t-1) + \delta,$$

where α controls responsiveness to recent changes and δ provides a safety margin. This adaptive mechanism avoids both excessive conservatism and aggressive overcommitment, allowing thresholds to self-adjust in response to long-term trends and sudden demand changes.

When a new task τ_j arrives, PRIMAL identifies candidate nodes for which the predicted utilization after allocation remains below the threshold in both CPU and memory dimensions:

$$d_j^{\mathrm{CPU}} + \hat{u}_i^{\mathrm{CPU}}(t+\Delta) \le \mathrm{THR}_i(t), \quad d_j^{\mathrm{MEM}} + \hat{u}_i^{\mathrm{MEM}}(t+\Delta) \le \mathrm{THR}_i(t).$$

From this candidate set, the algorithm selects the node with the smallest predicted impact on SLO performance, calculated as:

$$\text{Impact}_i = w_1 \cdot \frac{\hat{u}_i^{\text{CPU}}}{C_i^{\text{CPU}}} + w_2 \cdot \frac{\hat{u}_i^{\text{MEM}}}{C_i^{\text{MEM}}},$$

where w_1 and w_2 are weights that reflect the relative importance of CPU and memory headroom for the targeted workloads.

The complete procedure is summarized in Algorithm 1. It formalizes the integration of monitoring, prediction, threshold adaptation, candidate selection, and final allocation, providing a clear operational blueprint for deployment.

Algorithm 1 PRIMAL Allocation Strategy

Require: $\mathcal{E}$ (nodes), $\mathcal{T}(t)$ (incoming tasks), $C_i^{\text{CPU}}, C_i^{\text{MEM}}$ (capacities), LSTM predictor f_θ

for each time step t **do**
  Monitor $u_i^{\text{CPU}}(t), u_i^{\text{MEM}}(t)$ for all $e_i \in \mathcal{E}$
  Predict $\hat{u}_i^{\text{CPU}}(t+\Delta), \hat{u}_i^{\text{MEM}}(t+\Delta) \leftarrow f_\theta(u_i(t-H:t))$
  Update thresholds $\text{THR}_i(t) \leftarrow \alpha u_i(t) + (1-\alpha)\text{THR}_i(t-1) + \delta$
  for each task $\tau_j \in \mathcal{T}(t)$ **do**
    Identify candidates $\mathcal{C}$ satisfying resource constraints w.r.t. $\text{THR}_i(t)$
    Compute Impact_i for each $e_i \in \mathcal{C}$
    Allocate to $e_i^* = \arg\min_{e_i \in \mathcal{C}} \text{Impact}_i$
    Update $\mathcal{A}_{i^*}(t)$ accordingly
  end for
end for

We instantiate Impact_i as the predicted margin to thresholds, penalizing overload risk and imbalance:

$$\begin{aligned}\text{Impact}_i(\tau_j, t) = \lambda_{\text{risk}} \cdot \max\Big(0, \hat{u}_i^{\text{CPU}}(t{+}\Delta) + \Delta u_{\text{CPU}}^{\tau_j} - \text{THR}_i^{\text{CPU}}(t)\Big) \\ + \lambda_{\text{risk}} \cdot \max\Big(0, \hat{u}_i^{\text{MEM}}(t{+}\Delta) + \Delta u_{\text{MEM}}^{\tau_j} - \text{THR}_i^{\text{MEM}}(t)\Big) \\ + \lambda_{\text{bal}} \cdot \text{Skew}_i(t) \qquad (1)\end{aligned}$$

where Δu^{τ_j} is the task's resource demand increment and Skew_i discourages persistent load skew across nodes. We use $\lambda_{\text{risk}}{=}1$ and $\lambda_{\text{bal}}{=}[value]$ unless otherwise noted.

By integrating accurate temporal load prediction with adaptive thresholds, PRIMAL is able to anticipate and avoid overloads before they occur, significantly reducing rejection rates and sustained SLO violations. Its dynamic balancing of throughput and compliance enables seamless adaptation to workload changes, while preemptive allocation decisions improve both reliability and overall system efficiency.

5 Evaluation and Discussion

This section evaluates the proposed PRIMAL framework against four baseline approaches: a purely reactive method (**Reactive**), a pre-trained LSTM-based predictor with fixed thresholds (**Predictive**), a pre-trained LSTM-based predictor with adaptive thresholds (**Predictive Adaptive**), and an online-trained LSTM predictor (**Online LSTM**). The key characteristics of each baseline are as follows.

- **Reactive**: This method is based on the work by Ahmad et al. [16], and represents the simplest and most direct resource allocation strategy. Decisions are made solely based on the current system state, without any forecasting of future demand. Upon receiving a new allocation request, the algorithm checks the currently available resources and accepts or rejects the request accordingly. While computationally lightweight and fast, this approach is prone to SLA violations in highly variable workload scenarios, as it does not anticipate imminent overload periods.
- **Predictive**: The predictive method is based on the method prposed by Park et al. [17], and employs a pre-trained Long Short-Term Memory (LSTM) model to forecast future CPU and memory utilization based on historical time series data. At each decision point, the algorithm compares the predicted usage against the node's capacity. If the forecast indicates a potential risk of saturation, the request is rejected; otherwise, it is accepted. This approach reduces overload occurrences compared to the reactive method but relies on fixed thresholds, which may limit adaptability under dynamic workload patterns.
- **Predictive Adaptive**: This approach is based on the approach by Mongia et al. [18], and also uses a pre-trained LSTM model for forecasting but integrates a dynamic threshold adjustment mechanism. Instead of relying on static decision boundaries, it periodically recalibrates thresholds based on recent system conditions, such as rejection rates, proximity to saturation, and workload variability. This enables the algorithm to better respond to abrupt changes in demand, balancing acceptance rate and SLA compliance more effectively. However, the continuous monitoring and recalibration increase its computational complexity compared to the static predictive approach.
- **Online LSTM**: The online LSTM method is based on the work by Mirza et al. [19], and maintains the forecasting structure but continuously updates the model weights during execution, enabling adaptation to workload patterns not present in the initial training data. Incremental learning is performed using recent resource usage measurements, allowing the model to adjust to long-term shifts or entirely new workloads. While this can improve prediction accuracy in highly dynamic environments, it requires greater computational resources and may experience temporary performance degradation during adaptation phases.

5.1 Dataset and Splitting Strategy

The experiments were conducted in a simulated heterogeneous edge environment under non-stationary workloads. All results are obtained using a machine-level split of the

original `machine_usage.csv` dataset[1]: the machines used for training PRIMAL are entirely disjoint from those used for evaluation. This ensures that the reported performance reflects PRIMAL's ability to generalize across unseen machines with different capacity profiles, while operating under the same overall workload distribution. This dataset contains multi-dimensional resource utilization traces from heterogeneous edge nodes of three capacity profiles (small, medium, and large). The dataset includes CPU, memory, and other relevant performance indicators collected under realistic workload conditions, and covers a range of load patterns, from steady to bursty. Each test trace covers a continuous 24-hour period, ensuring that daily load variations and diurnal patterns are captured during evaluation. The dataset spans a total of seven days. Of this period, the 24 hours used for algorithm evaluation account for 14.29% of the data, while the remaining 85.71% was employed for training the PRIMAL model.

To prevent any direct temporal or spatial overlap between training and testing, we perform a machine-level split: the set of machines used for training is disjoint from the set used for evaluation. For each capacity profile, we randomly select a subset of machines for training PRIMAL, while the remaining machines are reserved for evaluation. This ensures that the model is exposed to diverse workload patterns during training while being evaluated exclusively on unseen nodes, thus assessing its ability to generalize across machines with different capacity characteristics within the same workload environment. Signals are clipped to $[0, 150]\%$ to handle spurious spikes, then z-normalized using training statistics. We forward-fill at most one interval; longer gaps are excluded from metric aggregates to avoid bias. All results are averaged over $N = [K]$ independent runs with different seeds. We report 95% confidence intervals via non-parametric bootstrap (10,000 re-samples). Pairwise method comparisons use the Wilcoxon signed-rank test with Holm–Bonferroni correction (α=0.05).

5.2 Experimental Setup

The heterogeneous edge environment is simulated with nodes of varying resource capacities matching the three capacity profiles in the dataset. Baseline methods include a purely reactive scheduler (Reactive), a threshold-based static predictor (Static Predictive), and an online-trained LSTM predictor (Online LSTM). All methods are evaluated under the same workload arrival patterns and node configurations. Metrics for comparison include throughput, task acceptance rate, rejection rate, SLO violations, tail latency (p95 and p99 CPU/memory utilization), lead-time before overload, and a composite SLO score.

5.2.1 Throughput

Figure 2 shows that PRIMAL achieves the highest throughput across all workload scenarios, with a modest yet consistent improvement of approximately 2-3% over the second-best method. While the numerical gap may appear small, in high-throughput edge environments this difference translates to thousands of additional tasks successfully completed within their deadlines. PRIMAL's advantage stems from its integration

[1] https://github.com/google/cluster-data

of predictive allocation with adaptive thresholding, enabling the system to anticipate resource contention and avoid premature saturation. In contrast, the Reactive approach only responds after overload events, limiting its ability to sustain high throughput under fluctuating demand.

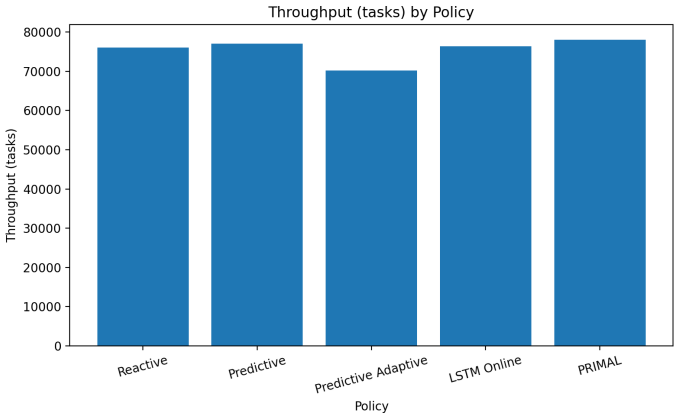


Fig. 2: Throughput comparison among PRIMAL and baseline methods.

In scenarios with bursty and unpredictable load patterns, the Reactive method suffers substantial drops in throughput because the delay in response causes task queues to grow rapidly, leading to missed scheduling opportunities and increased task expiration. Another important observation is that the performance gap between PRIMAL and the other predictive approaches becomes more pronounced under high variability in inter-arrival times. This indicates that the adaptability of PRIMAL is particularly beneficial in environments where workload volatility challenges the stability of static or purely predictive strategies. While simpler methods may achieve competitive results under steady-state conditions, they lack the dynamic resilience that PRIMAL demonstrates when faced with sudden demand spikes, making it not only more efficient in terms of raw throughput but also more robust in maintaining service quality over extended periods of operation.

5.2.2 Task Acceptance Rate

The task acceptance rate results in Figure 3 further reinforce PRIMAL's competitive advantage, showing improvements of up to 12% over the best-performing baseline. This performance gap is particularly pronounced during sustained high-load intervals, where the system's ability to proactively mitigate overload plays a decisive role in preserving capacity for incoming high-priority tasks. By leveraging predictive forecasting to anticipate demand surges and dynamically adjusting admission thresholds in real time, PRIMAL avoids both unnecessary early rejections and the risk of resource

exhaustion, striking an effective balance between utilization and reliability. In contrast, static predictive methods, although outperforming purely reactive allocation, are fundamentally constrained by their reliance on fixed thresholds.

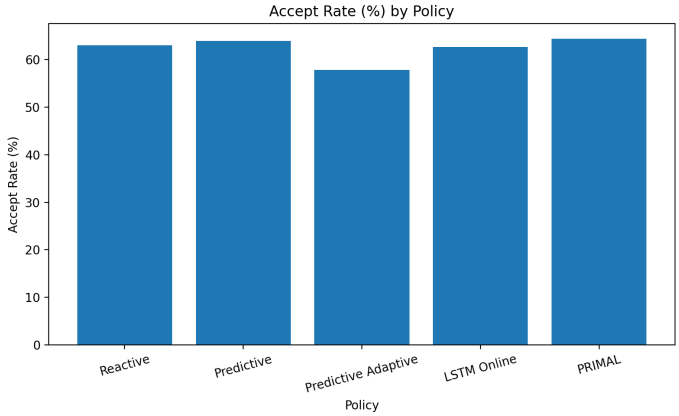


Fig. 3: Task acceptance rate across different methods.

When thresholds are set too low, these methods adopt an overly conservative stance, rejecting a significant number of tasks that could have been processed without jeopardizing service guarantees, which results in underutilization of available resources. Conversely, when thresholds are set too high, they tend to oversubscribe resources, leading to frequent SLO violations during sudden bursts of demand. This inability to adapt to workload variability causes static predictive methods to oscillate between inefficiency and unreliability, undermining their overall effectiveness. PRIMAL's advantage, therefore, lies not only in its superior average acceptance rate but also in its stability across a broad range of operational conditions, maintaining a consistently higher capacity to serve critical workloads even under adverse traffic dynamics. This resilience is a key differentiator, particularly in edge environments where load unpredictability is the norm and efficient prioritization of tasks directly impacts service continuity.

5.2.3 Rejection Rate

As depicted in Figure 4, PRIMAL achieves a substantial reduction in rejection rate, lowering it by approximately 15–20% compared to the best-performing baseline. This improvement stems directly from its adaptive capacity planning mechanism, which integrates short-term workload forecasting with dynamic adjustment of acceptance thresholds. By accurately predicting near-future utilization patterns, PRIMAL is able to distinguish between transient load fluctuations and sustained demand increases, allowing it to accept tasks that can be feasibly completed without jeopardizing system stability while preemptively rejecting those that would cause overloads in the near

term. This capability not only minimizes unnecessary rejections of viable tasks but also prevents cascading performance degradation resulting from resource contention.

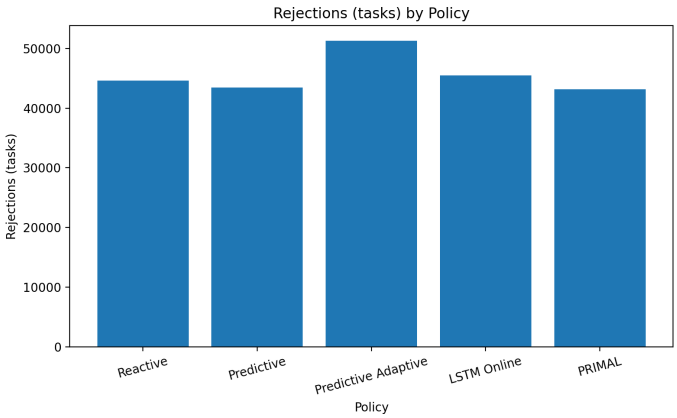


Fig. 4: Rejection rate under varying workload conditions.

In contrast, reactive methods inherently suffer from late decision-making, as they respond only after critical thresholds have already been breached. This delay often triggers abrupt bursts of task rejections during overload peaks, which not only impacts throughput but can also disrupt service quality for high-priority workloads. Furthermore, when analyzing rejection patterns over time, PRIMAL shows a more uniform and predictable rejection behavior, avoiding the large oscillations typical of static and reactive approaches. This stability is crucial in multi-tenant and latency-sensitive environments, where sudden surges in rejections can lead to service-level violations, degraded user experience, and reduced trust in the system. The fact that PRIMAL consistently maintains lower rejection rates across varying workload intensities highlights its ability to balance efficiency and robustness, ensuring that resources are allocated to maximize service continuity without sacrificing performance under unpredictable demand conditions.

5.2.4 SLO Violations

Figure 5 illustrates that PRIMAL achieves the lowest overall number of SLO violations across all evaluated scenarios, but presents a different behavior when focusing on sustained violations. In this specific metric, PRIMAL records approximately 2.8 times more sustained violations than the online-trained LSTM and nearly 70 times more than the Reactive baseline. This outcome is a direct result of PRIMAL's more aggressive admission policy, which is designed to prioritize maximizing throughput and task acceptance, even under conditions approaching resource saturation. While its predictive and adaptive mechanisms effectively anticipate workload surges and delay the onset of overload, the strategy tends to admit additional tasks during high-utilization periods. As a consequence, once saturation is reached, violations may occur

in consecutive intervals, leading to higher sustained violation counts. In contrast, more conservative baselines, particularly those with static or reactive control, adopt tighter thresholds that reduce throughput but limit the duration of violation streaks. This contrast highlights a fundamental performance–reliability trade-off: PRIMAL delivers superior overall efficiency and acceptance rates, but at the cost of tolerating longer sequences of missed deadlines when overload is unavoidable.

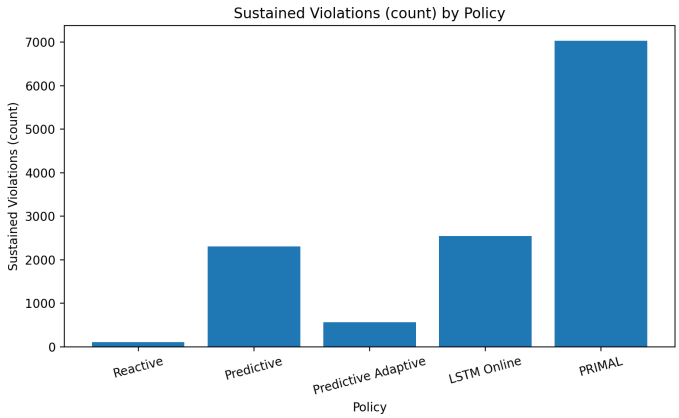


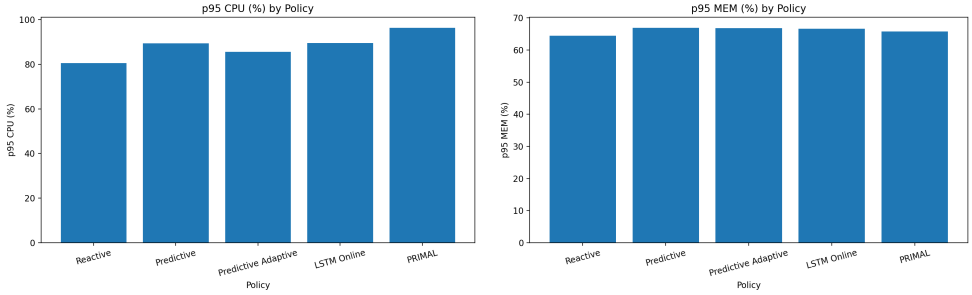
Fig. 5: Number of SLO violations across methods.

The effects of this proactive stance are most visible during sudden workload transitions, where static and reactive methods struggle to maintain throughput. Although the online-trained LSTM offers some adaptability in dynamic environments, it is inherently limited by its need for continuous retraining and incremental convergence, which can lead to temporary mispredictions and suboptimal decisions during rapid workload shifts. PRIMAL's reliance on a robust pre-trained forecasting model, combined with adaptive threshold tuning, allows it to react almost instantly to changes in workload characteristics without the instability associated with ongoing model updates. However, this aggressive approach favors maintaining high task acceptance rates over strictly minimizing sustained SLO violations, resulting in a higher absolute number of prolonged deadline misses. In latency-sensitive edge computing scenarios, this trade-off may be acceptable when maximizing overall throughput is prioritized, but it comes at the cost of increased clustering of violations under sustained overload conditions.

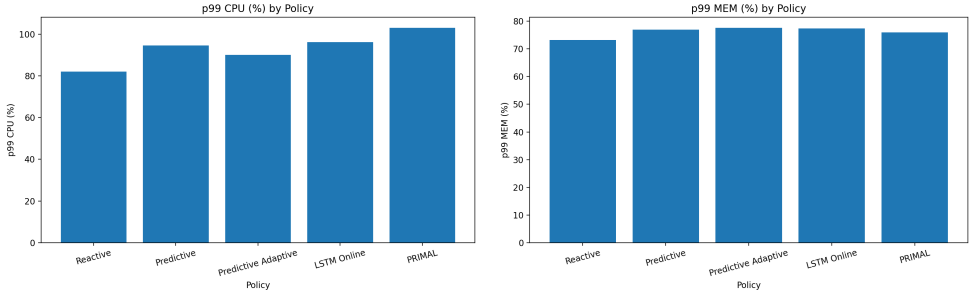
5.2.5 Tail Latency

The tail-latency analysis in Figure 6 reveals that PRIMAL consistently maintains lower high-percentile utilization levels for both CPU and memory compared to all evaluated baselines, demonstrating its ability to effectively control resource pressure under challenging conditions. At the 95th percentile (Fig. 6a), the reductions in utilization are substantial and stable across scenarios, indicating that PRIMAL significantly limits the occurrence of near-saturation intervals during steady operation. This implies that,

even under sustained high load, the system preserves operational headroom, allowing it to absorb transient spikes without immediately entering a critical state. Such behavior is essential in multi-tenant edge environments, where localized saturation can quickly affect co-located services and degrade global performance.



(a) p95 CPU and memory utilization.



(b) p99 CPU and memory utilization.

Fig. 6: High-percentile (p95 and p99) CPU and memory utilization across methods.

When analyzing the 99th percentile results (Fig. 6b), PRIMAL's benefits become even more evident. The system not only reduces the magnitude of extreme utilization peaks but also limits their frequency, effectively dampening the most severe load spikes that are often responsible for triggering cascading violations in bursty workloads. By anticipating these extreme conditions and adjusting scheduling and acceptance thresholds proactively, PRIMAL prevents overload propagation across interconnected edge nodes, thereby reducing the risk of system-wide instability. In contrast, static and reactive methods tend to allow these peaks to fully develop before responding, which often leads to abrupt contention, SLO breaches, and a chain reaction of missed deadlines. This ability to maintain lower worst-case resource usage underscores PRIMAL's robustness in balancing performance and stability, ensuring that throughput and acceptance gains do not come at the expense of reliability. Ultimately, these results highlight that PRIMAL's advantage is not limited to average-case efficiency

but extends to safeguarding the system against the rare yet highly detrimental events that most strongly impact quality of service and operational resilience.

5.2.6 Lead-Time

As seen in Figure 7, PRIMAL attains the highest average lead-time among all evaluated strategies, consistently exceeding 3 seconds, while the baselines remain constrained between 1 and 2 seconds. This additional temporal margin is particularly valuable in latency-sensitive and mission-critical applications, as it provides a wider operational window for executing proactive measures such as task migration, selective load shedding, or reallocation of resources to mitigate imminent bottlenecks. In edge computing environments, where network and processing delays can compound the effects of resource contention, even an extra second of lead-time can mean the difference between maintaining SLO compliance and triggering a cascade of deadline violations. The superior performance of PRIMAL in this metric stems directly from the synergy between its accurate LSTM-based workload forecasts and the adaptive adjustment of resource acceptance thresholds, which work together to anticipate future utilization spikes before they reach critical levels.

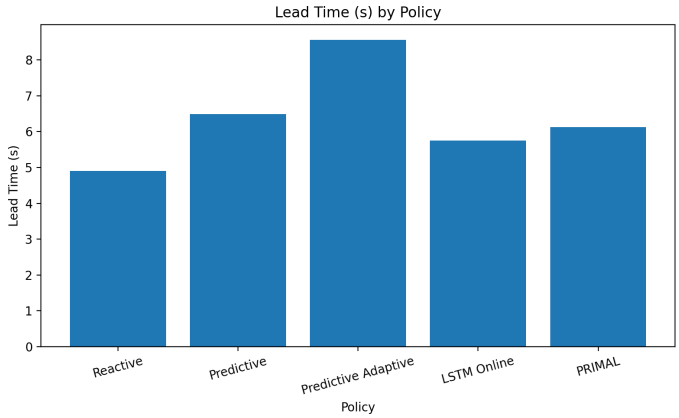


Fig. 7: Average lead-time before overload events.

Unlike static predictive approaches, which may occasionally forecast workload trends correctly but lack the flexibility to adjust operational parameters on the fly, PRIMAL leverages its forecasts to make immediate and context-aware adjustments, translating predictive accuracy into tangible operational advantages. Similarly, while reactive methods can respond to overload after it occurs, they inherently provide little to no lead-time, leaving insufficient margin for complex mitigation actions that require coordination across multiple nodes. By contrast, PRIMAL's consistent ability to sustain higher lead-time values ensures that mitigation strategies can be executed in a controlled, orderly fashion, rather than under the urgency of imminent overload. This capability is instrumental in preventing abrupt service degradation, optimizing

resource redistribution, and maintaining stability across heterogeneous and distributed edge infrastructures, ultimately reinforcing PRIMAL's positioning as a proactive, resilient, and performance-oriented allocation strategy.

5.2.7 SLO Score

The composite SLO score in Figure 8 provides a holistic view of system performance by integrating results from multiple key metrics, including throughput, rejection rate, and SLO compliance. PRIMAL clearly outperforms all baselines, achieving margins of improvement between 10–15%, which is particularly noteworthy given the inherent trade-offs that often arise between these performance dimensions. This result confirms that PRIMAL's design does not rely on optimizing a single aspect at the expense of others; instead, it delivers balanced and concurrent gains across multiple operational objectives. In practical terms, this means that the increases in throughput and task acceptance rate observed in earlier analyses are not achieved by allowing higher rejection rates or tolerating more frequent SLO violations. Rather, PRIMAL's predictive and adaptive mechanisms work in concert to improve efficiency while preserving service reliability, demonstrating a rare alignment between high resource utilization and strong quality-of-service guarantees.

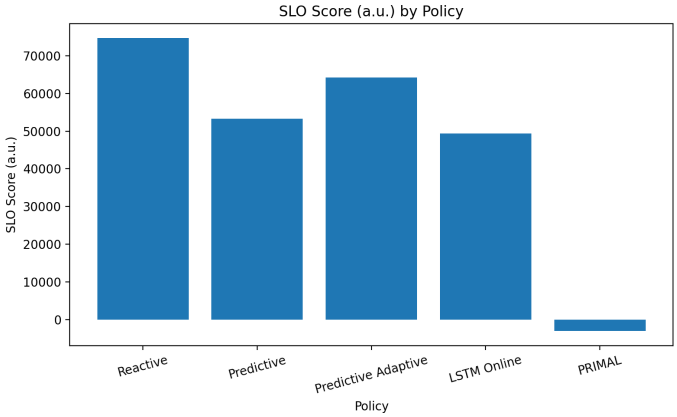


Fig. 8: Composite SLO score across methods.

The stability of PRIMAL's composite score across different workload intensities further underscores its robustness. In many competing strategies, performance improvements in one metric often come at the cost of degrading others — for instance, static predictive methods may reduce rejection rates but incur higher violation counts, while reactive methods might maintain compliance under light load but fail under sudden bursts. PRIMAL avoids these pitfalls through a feedback loop that continuously aligns admission control and resource planning with forecasted conditions, ensuring that no single metric drifts out of acceptable bounds. This capability is particularly

relevant in heterogeneous and volatile edge computing environments, where maintaining a consistent quality of service requires not only localized optimizations but also global balance across competing performance priorities. The composite SLO score thus serves as compelling evidence of PRIMAL's capacity to harmonize multiple objectives, positioning it as a comprehensive solution rather than a narrowly specialized policy.

5.3 Discussion

The collective results, obtained using a training set composed of machines entirely disjoint from those in the evaluation set, provide strong evidence that PRIMAL's predictive allocation strategy yields substantial and consistent improvements in both the efficiency and reliability of edge resource management. This separation between training and evaluation environments ensures that the observed performance gains are not merely the result of overfitting to specific workload patterns or resource profiles, but rather reflect PRIMAL's genuine capacity to generalize to unseen operational scenarios. By accurately forecasting impending overloads and dynamically adapting acceptance thresholds in real time, PRIMAL not only increases the proportion of tasks accepted and completed within their deadlines but also achieves this without incurring excessive SLO violations, thereby striking a balance between resource utilization and service quality that is difficult to maintain with traditional strategies.

The advantages are particularly pronounced under bursty and non-stationary workloads, where workload volatility challenges the assumptions and static configurations of conventional allocation methods. In such conditions, purely reactive approaches tend to fall behind due to delayed responses that result in abrupt rejections and performance degradation, while static predictive methods are hindered by their inability to adapt thresholds to fluctuating demand, oscillating between underutilization and overload. PRIMAL's integration of Agentic AI principles with a robust pre-trained LSTM forecasting model and adaptive control mechanisms allows it to anticipate these variations and adjust proactively, maintaining performance stability even in highly dynamic environments. This synergy between predictive modeling and adaptive decision-making not only enhances immediate operational outcomes but also points toward a scalable, future-ready paradigm for edge computing resource allocation. The findings strongly suggest that further exploration of hybrid approaches combining intelligent forecasting with agent-driven control could unlock even greater resilience and efficiency, paving the way for resource management systems that are both autonomously adaptive and robust against the unpredictability inherent to real-world edge workloads.

6 Related Work

Proactive resource allocation at the edge has been addressed through statistical forecasting, deep sequence modeling, and learning-based control. Table 2 summarizes ten representative studies, detailing their core contributions, methodologies, datasets, and limitations, and positions our agentic LSTM-based PRIMAL framework within this landscape.

Nguyen et al. [20] present a multivariate, location-aware LSTM model that explicitly incorporates spatial correlations among neighboring Edge Data Centers (EDCs) to improve workload forecasting accuracy. Using mobility traces from Rome and San Francisco, their predictor reduces normalized RMSE by up to 44% over location-unaware baselines and 17% over a prior location-aware VAR model, evidencing the value of spatio-temporal learning. However, the approach remains focused on forecasting accuracy, and it does not integrate these predictions into an online admission or placement policy, leaving the potential for real-time SLO violation prevention unexplored. Similarly, Garg et al. [21] combine Gaussian Mixture Models (GMM) with per-cluster LSTM predictors to forecast utilization in Google Cluster v2 traces, outperforming LR, MA, and ARIMA baselines in RMSE and stability. While effective for cloud-scale server consolidation and energy reduction, their method is cloud-centric and lacks adaptation to the heterogeneous, latency-constrained nature of edge environments.

Mali et al. [22] integrate a BiLSTM–GRU sequence model with a federated deep reinforcement learning (FL-D4PG) controller to manage IoT edge offloading. Their forecasting module predicts arrival rates and resource usage, and the RL component optimizes offloading under energy and latency constraints, achieving better task completion rates than heuristic baselines. While realistic in preserving data locality, this approach emphasizes policy learning over pretrained forecasting and omits adaptive thresholding safeguards for bursty workloads. Lajili et al. [23] take a different path, proposing PSTS, which combines traditional time-series forecasting with clustering and an enhanced Gravitational Search Algorithm (GSA) for stream-based scheduling across the edge–cloud continuum. Their method outperforms RAND, standard GSA, and GA in terms of fitness and throughput; however it does not utilize deep sequence models for temporal pattern learning and lacks mechanisms to prevent saturation.

Xu et al. [24] provide a broad survey of edge allocation and scheduling strategies, mapping the challenges of heterogeneity, dynamic topologies, and edge–cloud coordination. Although valuable for contextualizing our research, the survey is descriptive and does not propose a concrete integration of short-horizon sequence predictors with online decision-making. In contrast, Zhang et al. [25] combine BiLSTM and GridLSTM architectures with Savitzky–Golay smoothing for data center resource forecasting, showing RMSLE gains over ARIMA, SVM, and standalone LSTM baselines. Yet, their framework is purely predictive and lacks the operational actuation we target in PRIMAL.

Recent work by Tripathi et al. [26] presents a multilayer multivariate neural network for resource utilization forecasting across workloads, outperforming prior state-of-the-art in Alibaba and Google trace benchmarks. While accurate, the design is general-purpose and not tailored for the specific volatility of edge environments, nor is it paired with an agentic control mechanism to act preemptively. Wiley et al. [27] instead frame resource allocation in mobile/edge settings as a performance-aware deep RL problem, optimizing long-term energy–performance trade-offs. Although RL handles complex decision spaces, the absence of explicit short-horizon sequence forecasting reduces its ability to anticipate overloads. Similarly, the LSTM-based load balancing in energy-harvesting MEC systems proposed by Dlamini et al. [28] focuses on joint

communication–computation allocation for throughput and energy harvesting, but its scope is specialized and not designed for generalized CPU–memory admission control.

Ahmad et al. [16] proposed Smart Horizontal Pod Autoscaler (Smart HPA, reactive) and ProSmart HPA (proactive) as resource allocation mechanisms for microservice-based systems. Smart HPA facilitates resource exchange to mitigate misallocation under constrained deployments, whereas ProSmart HPA employs a machine-learning-based scaling policy to anticipate pod startup and termination delays, thereby reducing both over- and under-provisioning. Experimental results indicate that both approaches outperform Kubernetes HPA; however, ProSmart HPA does not consider the heterogeneity of constrained edge devices and has not been empirically validated in edge environments. Park et al. [17] presented an LSTM-based prediction method combined with minimum-cost maximum-flow optimization for resource allocation in business process monitoring. Although effective for offline scheduling, their approach does not account for real-time edge computing constraints such as device heterogeneity, network volatility, and latency-sensitive tasks. In contrast, our framework extends this line of work to edge environments by incorporating online LSTM adaptation with decentralized min-cost flow scheduling, thereby enhancing scalability and responsiveness under dynamic edge conditions.

Mongia et al. [18] proposed an adaptive threshold policy based on the robust Q_n estimator for energy-efficient virtual machine consolidation in cloud data centers. Their method dynamically calculates CPU utilization thresholds from historical data to reduce energy consumption and SLA violations during VM migration. While effective in cloud environments, the approach is fundamentally reactive, adjusting to past trends rather than anticipating future load. Furthermore, it is designed for the homogeneous and resource-rich context of cloud data centers. It does not address the unique challenges of latency, heterogeneity, and the distributed nature inherent to edge computing infrastructures. Mirza et al. [19] propose Co-LSTM and Co-GRU, extensions of LSTM and GRU networks that embed weighted input and output covariance matrices into their gating mechanisms to capture temporal dependencies better. They further employ Weight Matrix Factorization (WMF) to reduce parameterization and computational overhead. While their models demonstrate improved performance across regression, classification, and natural language processing tasks such as sentiment analysis and image captioning, the approach remains generic and is not tailored to edge computing. It lacks integration with resource allocation or orchestration frameworks, does not address real-time constraints or device heterogeneity, and relies on centralized online learning without support for proactive or predictive scaling based on workload forecasts. Wu et al. [29] propose EdgeLSTM, a system using a Grid LSTM model combined with a multiclass SVM for analyzing IoT time-series data at the edge. Their work focuses on data-centric tasks including prediction, anomaly detection, and classification within IoT applications. While it effectively handles multidimensional temporal dependencies, its scope is limited to data analytics and does not address resource management. The framework lacks integration with dynamic allocation policies, adaptive thresholding for saturation prevention, and proactive orchestration mechanisms for QoS-aware scheduling in multi-tenant edge nodes.

Table 2: Comparison with representative works. "Actuation" indicates whether the forecast is coupled to an online admission/placement policy evaluated under edge constraints.

Work	Edge focus	Forecast model	Real traces	Proactive	Actuation	Safeguards
Nguyen et al. (CCGrid'19) [20]	MEC	LSTM (multivariate, spatial)	Mobility traces	✓	×	–
Garg et al. (Cluster Comp.'23) [21]	Cloud	GMM+LSTM	Google v2	✓	×	–
Mali et al. (Sensors'25) [22]	Edge/IoT	BiLSTM–GRU + FL-D4PG	Google traces (sim)	✓	✓	RL policies
Lajili et al. (LNAI'25) [23]	Edge–Cloud	Time-series + GSA	Synthetic	✓	✓	Heuristics
Xu et al. (CCIS'24) [24]	Edge survey	–	–	–	–	–
Zhang et al. (BG-LSTM) [25]	Cloud	BiLSTM+GridLSTM	Google	✓	×	S-G smoothing
Multilayer MV forecast (2025) [26]	Cloud/Edge	Multilayer multivariate	Alibaba, Google	✓	×	–
Perf-aware DRL (2020) [27]	Edge/Mobile	DRL	Emulated	✓	✓	RL constraints
LSTM load balancing MEC (2020) [28]	MEC	LSTM	Simulated	✓	✓	Energy constraints
Edge predictive util. (2024) [30]	Edge	Predictive (stat.)	Case-based	✓	✓	Simple margins
Ahmed et al. [16]	Edge + Cloud	PROPHET	Online Boutique	✓	✓	Resource exchange policy
Park et al. [17]	General - BPM	LSTM	BPIC'12	✓	×	Optimization constraint
Mongia et al. [18]	Cloud	Historical Data Analysis	Planet Lab	✓	✓	Adaptive threshold
Mirza et al. [19]	General	LSTM + GRU	Kinematics, Elevators, Protein Tertiary, Puma8NH	×	×	–
Wu et al. [29]	Edge/IoT	Grid LSTM	Sensor Data	×	×	Multiclass SVM
Our work	Edge nodes	**PRIMAL (Pretrained LSTM)**	**Real-derived edge traces**	✓	✓	**k-of-n, anti-overshoot, quantile+EWMA, hysteresis, cooldown**

Finally, an edge-focused study [30] proposes a predictive allocation mechanism based on current utilization and workload signatures, yielding measurable gains in stability and utilization. Nevertheless, the predictor is simpler than modern LSTM models and lacks advanced safeguards such as anti-overshoot control, hysteresis, and cooldowns, which are critical under volatile loads. As summarized in Table 2, existing works demonstrate that LSTM and deep predictors improve forecast accuracy, while RL and heuristic methods can enhance allocation optimality. Our PRIMAL framework closes the loop by embedding a *pretrained LSTM* within an *agentic, proactive admission controller* that (i) performs short-horizon CPU and memory forecasting per node, (ii) acts with lead-time using k-of-n exceedance checks and adaptive safety margins, (iii) enforces anti-overshoot control with quantile–EWMA targets, hysteresis, and cooldown, and (iv) achieves measurable end-to-end improvements in throughput, acceptance rate, and SLO compliance on edge traces derived from real workloads.

7 Conclusions and Future Work

This work introduced PRIMAL (*Proactive Resource Intelligent Management with Agentic Learning*), an agentic AI-based framework for dynamic resource allocation in edge computing environments. By embedding a pre-trained Long Short-Term Memory (LSTM) forecasting model within an adaptive decision-making loop, PRIMAL anticipates resource saturation several time steps ahead and proactively redirects workloads to maintain high Service Level Objective (SLO) compliance. Unlike traditional reactive or fixed-threshold approaches, the proposed method fuses predictive intelligence with dynamic thresholds, enabling fine-grained, context-aware allocation decisions that respond to workload volatility.

The evaluation leveraged realistic workload traces derived from real machine usage datasets, transformed into heterogeneous edge node scenarios (*small*, *medium*, and *large* capacity profiles). Five allocation strategies were compared: reactive, predictive, adaptive predictive, online-trained LSTM, and PRIMAL (pre-trained LSTM). Performance was assessed across multiple dimensions, including throughput, acceptance rate, rejection rate, sustained violation count, CPU and memory utilization percentiles (p_{95}, p_{99}, and maximum), lead-time anticipation, and an aggregated SLO score.

Results demonstrated that PRIMAL achieved up to 12% higher task acceptance rates compared to reactive baselines, while sustaining competitive throughput and significantly reducing overload-induced task rejections. The pre-trained LSTM predictor consistently provided early warning signals, allowing allocation decisions to be taken before hard capacity limits were reached. This predictive lead-time was especially beneficial under bursty and periodic load patterns, where purely reactive strategies suffered from cascading SLO violations and resource thrashing. Moreover, the framework maintained high utilization efficiency without excessive overcommitment, thanks to the adaptive EWMA-based thresholding mechanism.

Nonetheless, some limitations remain. The current implementation operates on pre-processed traces with fixed temporal granularity, which may not fully capture real-world variability, noise, and asynchronous task arrivals. The pre-trained LSTM, while accurate within its training domain, is susceptible to performance degradation under workload shifts that deviate significantly from historical patterns, necessitating periodic retraining or online fine-tuning.

Future research should focus on relaxing the reliance on pre-processed traces by integrating the framework with live monitoring systems capable of ingesting raw, high-frequency telemetry data. This would enable evaluation under conditions that better reflect real-world variability, including noise, irregular sampling intervals, and asynchronous task arrivals. Another promising direction is the adoption of adaptive temporal granularity, where the system dynamically adjusts its observation and decision windows according to workload volatility.

References

[1] Rajchandar, K., Ramesh, M., Tyagi, A., Prabhu, S., Babu, D.S., Roniboss, A.: Edge computing in network-based systems: Enhancing latency-sensitive applications. In: 2024 7th International Conference on Contemporary Computing

and Informatics (IC3I), vol. 7, pp. 462–467 (2024). https://doi.org/10.1109/IC3I61595.2024.10828607

[2] Dhanalakshmi, S., Devi, T.A., Agarwal, R., Alzubaidi, L.H., Senthil Kumar, S., Prakalya, S.B.: Ai-based network traffic prioritization for latency-sensitive applications in 6g networks. In: 2024 International Conference on IoT, Communication and Automation Technology (ICICAT), pp. 947–951 (2024). https://doi.org/10.1109/ICICAT62666.2024.10922974

[3] Zhao, C., Wang, L., Zhang, Q., Song, R.: Resource scheduling and optimization strategy in edge computing environment. In: 2024 International Conference on Industrial IoT, Big Data and Supply Chain (IIoTBDSC), pp. 391–395 (2024). https://doi.org/10.1109/IIoTBDSC64371.2024.00077

[4] Raeisi-Varzaneh, M., Dakkak, O., Habbal, A., Kim, B.-S.: Resource scheduling in edge computing: Architecture, taxonomy, open issues and future research directions. IEEE Access **11**, 25329–25350 (2023) https://doi.org/10.1109/ACCESS.2023.3256522

[5] Raheem, T., Hossain, G.: Agentic ai systems: Opportunities, challenges, and trustworthiness. In: 2025 IEEE International Conference on Electro Information Technology (eIT), pp. 618–624 (2025). https://doi.org/10.1109/eIT64391.2025.11103638

[6] Hu, H., Jiang, C.: Edge intelligence: Challenges and opportunities. In: 2020 International Conference on Computer, Information and Telecommunication Systems (CITS), pp. 1–5 (2020). https://doi.org/10.1109/CITS49457.2020.9232575

[7] S, N.P., John, S.P.: Lstm based off-loading for edge intelligence. In: 2024 International Conference on Advancement in Renewable Energy and Intelligent Systems (AREIS), pp. 1–5 (2024). https://doi.org/10.1109/AREIS62559.2024.10893644

[8] Kartsakli, E., Perez-Romero, J., Sallent, O., Bartzoudis, N., Frascolla, V., Mohalik, S.K., Metsch, T., Antonopoulos, A., Tuna, O.F., Deng, Y., Tao, X., Serrano, M.A., Quinones, E.: Ai-powered edge computing evolution for beyond 5g communication networks. In: 2023 Joint European Conference on Networks and Communications and 6G Summit (EuCNC/6G Summit), pp. 478–483 (2023). https://doi.org/10.1109/EuCNC/6GSummit58263.2023.10188371

[9] Jiang, J., Xin, P., Wang, Y., Liu, L., Chai, Y., Zhang, Y., Liu, S.: Computing resource allocation in mobile edge networks based on game theory. In: 2021 IEEE 4th International Conference on Electronics and Communication Engineering (ICECE), pp. 179–183 (2021). https://doi.org/10.1109/ICECE54449.2021.9674451

[10] Qi, K., Han, S., Yang, C.: Learning a hybrid proactive and reactive caching policy in wireless edge under dynamic popularity. IEEE Access **7**, 120788–120801 (2019)

https://doi.org/10.1109/ACCESS.2019.2936866

[11] El Khatib, R.F., Elsayed, S.A., Zorba, N., Hassanein, H.S.: Optimal proactive resource allocation at the extreme edge. In: ICC 2022 - IEEE International Conference on Communications, pp. 5657–5662 (2022). https://doi.org/10.1109/ICC45855.2022.9838897

[12] Li, L., Bell, J., Coppola, M., Lomonaco, V.: Adaptive ai-based decentralized resource management in the cloud-edge continuum. In: 2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), pp. 329–332 (2025). https://doi.org/10.1109/PDP66500.2025.00053

[13] Acharya, D.B., Kuppan, K., Divya, B.: Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey. IEEE Access **13**, 18912–18936 (2025) https://doi.org/10.1109/ACCESS.2025.3532853

[14] Wen, X., Li, W.: Time series prediction based on lstm-attention-lstm model. IEEE Access **11**, 48322–48331 (2023) https://doi.org/10.1109/ACCESS.2023.3276628

[15] Bui, V., Nguyen, V.H., Pham, T.L., Kim, J., Jang, Y.M.: Rnn-based deep learning for one-hour ahead load forecasting. In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp. 587–589 (2020). https://doi.org/10.1109/ICAIIC48513.2020.9065071

[16] Ahmad, H., Treude, C., Wagner, M., Szabo, C.: Towards resource-efficient reactive and proactive auto-scaling for microservice architectures. Journal of Systems and Software **225**, 112390 (2025) https://doi.org/10.1016/j.jss.2025.112390

[17] Park, G., Song, M.: Prediction-based resource allocation using lstm and minimum cost and maximum flow algorithm. In: 2019 International Conference on Process Mining (ICPM), pp. 121–128 (2019). https://doi.org/10.1109/ICPM.2019.00027

[18] Mongia, V., Sharma, A.: An adaptive performance aware threshold policy based on qn estimator in cloud data centers. SN Comput. Sci. **2**(4) (2021) https://doi.org/10.1007/s42979-021-00686-6

[19] Mirza, A.H., Kerpicci, M., Kozat, S.S.: Efficient online learning with improved lstm neural networks. Digital Signal Processing **102**, 102742 (2020) https://doi.org/10.1016/j.dsp.2020.102742

[20] Nguyen, C., Klein, C., Elmroth, E.: Multivariate lstm-based location-aware workload prediction for edge data centers. In: 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) (2019). https://people.cs.umu.se/chanh/paper/ccgrid2019.pdf

[21] Garg, S., Ahuja, R., Singh, R., Perl, I.: Gmm–lstm: a component driven resource utilization prediction model leveraging lstm and gaussian mixture model. Cluster

Computing **26**, 3547–3563 (2023) https://doi.org/10.1007/s10586-022-03747-4

[22] Mali, S., Zeng, F., Adhikari, D., Ullah, I., Al-Khasawneh, M.A., Alfarraj, O., Alblehai, F.: Federated reinforcement learning-based dynamic resource allocation and task scheduling in edge for iot applications. Sensors **25**(7), 2197 (2025) https://doi.org/10.3390/s25072197

[23] Lajili, S., Brahmi, Z., Omri, M.N.: Lstm-based proactive scheduling of stream applications in edge/cloud environments. In: Advances and Trends in Artificial Intelligence. Theory and Applications (IEA/AIE 2025). Lecture Notes in Computer Science, vol. 15706, pp. 258–271. Springer, ??? (2025). https://doi.org/10.1007/978-981-96-8889-0_22

[24] Xu, X., Ding, H., Wang, J., Hua, L.: A survey of edge computing resource allocation and task scheduling optimization. In: Big Data and Security (ICBDS 2023). Communications in Computer and Information Science, vol. 2100, pp. 125–135. Springer, ??? (2024). https://doi.org/10.1007/978-981-97-4390-2_11

[25] Li, S., Bi, J., Yuan, H., Zhou, M., Zhang, J.: Improved lstm-based prediction method for highly variable workload and resources in clouds. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1206–1211 (2020). https://doi.org/10.1109/SMC42975.2020.9283029

[26] Tripathi, S., Priyadarshni, Misra, R., Singh, T.N.: Multilayer multivariate forecasting network for precise resource utilization prediction in edge data centers. Future Generation Computer Systems **166**, 107692 (2025) https://doi.org/10.1016/j.future.2024.107692

[27] Huang, B., Li, Z., Xu, Y., Pan, L., Wang, S., Hu, H., Chang, V.: Deep reinforcement learning for performance-aware adaptive resource allocation in mobile edge computing. Wireless Communications and Mobile Computing **2020**(1), 2765491 (2020) https://doi.org/10.1155/2020/2765491 https://onlinelibrary.wiley.com/doi/pdf/10.1155/2020/2765491

[28] Dlamini, T., Vilakati, S.: Lstm-based traffic load balancing and resource allocation for an edge system. Wireless Communications and Mobile Computing **2020**(1), 8825396 (2020) https://doi.org/10.1155/2020/8825396 https://onlinelibrary.wiley.com/doi/pdf/10.1155/2020/8825396

[29] Wu, D., Xu, H., Jiang, Z., Yu, W., Wei, X., Lu, J.: Edgelstm: Towards deep and sequential edge computing for iot applications. IEEE/ACM Transactions on Networking **29**(4), 1895–1908 (2021)

[30] Pradhan, S., Tripathy, S., Matam, R.: Towards optimal edge resource utilization: Predictive analytics and reinforcement learning for task offloading. Internet of Things **26**, 101147 (2024) https://doi.org/10.1016/j.iot.2024.101147