Local-Cloud Inference Offloading for LLMs in Multi-Modal, Multi-Task, Multi-Dialogue Settings

Liangqi Yuan*, Dong-Jun Han†, Shiqiang Wang‡, and Christopher G. Brinton*
*Purdue University, †Yonsei University, ‡IBM T. J. Watson Research Center
Email: *{liangqiy,cgb}@purdue.edu, †djh@yonsei.ac.kr, ‡shiqiang.wang@ieee.org

Abstract

Compared to traditional machine learning models, recent large language models (LLMs) can exhibit multi-task-solving capabilities through multiple dialogues and multi-modal data sources. These unique characteristics of LLMs, together with their large model size, make their deployment more challenging. Specifically, (i) deploying LLMs on local devices faces computational, memory, and energy resource issues, while (ii) deploying them in the cloud cannot guarantee real-time service and incurs communication/usage costs. In this paper, we design TMO, a local-cloud LLM inference system with Three-M Offloading: Multi-modal, Multi-task, and Multi-dialogue. TMO incorporates (i) a lightweight local LLM that can process simple tasks at high speed and (ii) a large-scale cloud LLM that can handle multi-modal data sources. We develop a resource-constrained reinforcement learning (RCRL) strategy for TMO that optimizes the inference location (i.e., local vs. cloud) and multi-modal data sources to use for each task/dialogue, aiming to maximize the long-term reward (response quality, latency, and usage cost) while adhering to resource constraints. We also contribute M4A1, a new dataset we curated that contains reward and cost metrics across multiple modality, task, dialogue, and LLM configurations, enabling evaluation of offloading decisions. We demonstrate the effectiveness of TMO compared to several exploration-decision and LLM-as-Agent baselines, showing significant improvements in latency, cost, and response quality.

CCS Concepts

• **Computing methodologies** → **Distributed artificial intelligence**.

Keywords

Large Language Model, Reinforcement Learning, Multi-modal, Inference Offloading, Resource Constraint.

ACM Reference Format:
Liangqi Yuan*, Dong-Jun Han†, Shiqiang Wang‡, and Christopher G. Brinton*. 2025. Local-Cloud Inference Offloading for LLMs in Multi-Modal, Multi-Task, Multi-Dialogue Settings. In *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '25), October 27–30, 2025, Houston, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3704413.3764429

ACM ISBN 979-8-4007-1353-8/25/10
https://doi.org/10.1145/3704413.3764429

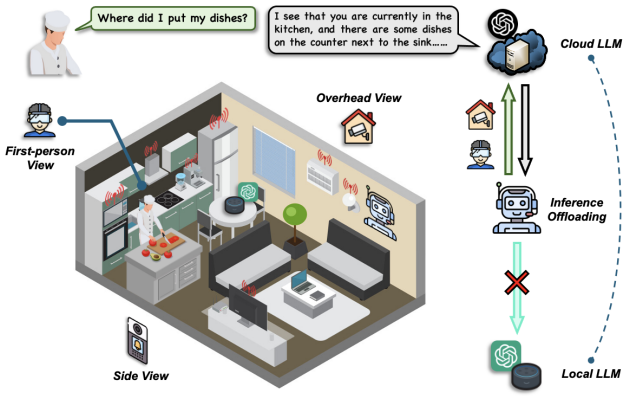


Figure 1: Application scenario of the TMO system as an LLM Assistant. In this example, cloud LLM is selected along with first-person and overhead views, as the query requires visual information.

1 Introduction

Large language models (LLMs) have demonstrated remarkable general-purpose task-solving capabilities across various fields and have been gradually incorporated into daily life. The development of LLMs is thriving, yet deploying LLMs on local devices presents practical challenges, including computational, memory, and energy resource limitations [23, 24]. These issues hinder the deployment of large-scale LLMs on users' local devices. Industry giants have developed strategies that either (i) involve lightweight versions of LLMs, such as Google's Gemma and Meta's LLaMA, which offer solutions with varying numbers of parameters (typically 3B, 8B, 13B, etc.) for fast and cost-efficient response, or (ii) place LLMs in the cloud, where users interact with them via the Internet, as seen with the ChatGPT application on mobile phones. The former often suffers from inferior inference performance, while the latter requires substantial network bandwidth and incurs high costs from service providers. This creates a pressing need to build systems that combine the advantages of both strategies, i.e., enhancing response quality while ensuring lower latency and usage costs.

1.1 Research Challenges

Optimizing LLMs over networks introduces unique challenges beyond traditional machine learning models, as LLMs operate with (i) multi-modal data sources (e.g., text, images), (ii) multi-task environments (e.g., editing, recommendations), and (iii) multi-dialogue contexts (i.e., ongoing conversations with users). Although these elements enrich the capabilities of LLMs, they introduce significant challenges, particularly in the inference stage:

1. Determining the optimal inference location: Determining whether to process tasks using a local LLM or to offload them to a

cloud LLM is non-trivial. This decision critically impacts not only the response quality but also the latency and costs. For example, local LLM inference may limit response quality due to the lightweight model deployed on local devices, whereas cloud LLM inference, although more powerful and capable of multi-modal processing, can introduce significant communication/computation delay.

2. Understanding relationships between modalities, tasks, and dialogues: There are intricate associations between multi-modal data sources, inference tasks, and their temporal variation across multiple dialogues. Although various types of input data can improve the response quality of LLMs, not all tasks require comprehensive multi-modal data. For example, a message editing task may not require images as inputs, whereas some human activity tasks require various types of image modalities [10]. Furthermore, when considering multi-dialogue scenarios, where each dialogue represents a task in a sequential order, earlier uploaded data modalities may provide sustained informational advantages, yet they might necessitate re-uploading if the information becomes outdated.

3. Uncertainty issues in offline training with LLM inference data: It is often desirable to train auxiliary models using LLM inference results, e.g., to learn modality-task associations. The computationally expensive nature of LLMs renders it impractical to conduct online training of such models. Consequently, generating datasets for offline training becomes necessary. However, due to inherent uncertainty in LLM inference, which arises from the probabilistic nature of language model outputs [9], such datasets may contain multiple identical samples with varying response scores. This uncertainty substantially complicates the evaluation of LLM outputs, particularly in multi-task or generative scenarios where standard answers or evaluation metrics do not exist.

Motivated by these challenges, we aim to answer the following research questions. *In the context of local-cloud LLM systems:*

(RQ1) *How can we optimally select the appropriate LLM for inference (i.e., local LLM vs. multi-modal cloud LLM)?*

(RQ2) *How can we identify and utilize the most informative and relevant multi-modal data when necessary for tasks in dialogues, while balancing response quality, latency, and usage cost?*

(RQ3) *How can we effectively address the inherent uncertainty in LLM inference data during offline training?*

1.2 Summary of Contributions

In this paper, we develop TMO, a local-cloud LLM inference offloading optimization system shown in Fig. 1. TMO orchestrates "Three-M" Offloading – multi-modal, multi-task, and multi-dialogue – aimed at enhancing response quality and reducing latency and usage costs. We address the above research questions through a resource-constrained reinforcement learning (RCRL) methodology that learns optimal actions (i.e., RQ1 and RQ2) to respond to mutli-task and multi-dialogue scenarios. We develop an estimator of LLM inference through a nearest neighbor strategy for effective offline learning (i.e., RQ3). Our main contributions are as follows:

- We construct TMO to account for adaptations between device-side LLM and cloud LLM inference within multi-modal, multi-task, multi-dialogue settings. This includes formalizing the interaction with both types of LLMs to dynamically adapt with diverse conversational demands **(Sec. 3.1 and Sec. 3.2)**.
- We formulate TMO's decision engine in terms of resource-constrained reinforcement learning (RCRL) to maximize cumulative reward across multi-dialogue interactions by selecting the optimal LLMs and modalities for inference. This approach enables a concrete optimization of trade-offs between response quality, latency, and usage costs while incorporating task-modality associations into the decision-making process **(Sec. 3.3)**.
- We devise a nearest neighbor strategy to overcome state-action pair uncertainty in estimating response quality during offline RL training. Our approach addresses two types of uncertainty that manifest in LLM evaluation contexts: *non-deterministic evaluation (NDE)*, where identical state-action pairs can yield varying response quality evaluations, and *out-of-distribution (OOD)*, where the system estimates the response quality for unseen state-action pairs (**Sec. 3.4**).
- We curate a new dataset, termed M4A1, which contains samples from three multi-modal data sources, four inference tasks, 2-5 dialogues, and four LLMs, with real-world measurements of latency and usage costs. Through extensive evaluation with this dataset, we demonstrate TMO's superior performance in response quality, latency, and usage cost compared to two SOTA exploration-decision and 15 LLM-as-Agent baselines **(Sec. 4)**.

2 Related Work

Efficient On-Device and Over-Network LLM. The rapid development of LLMs has profoundly influenced human-AI interaction while introducing significant challenges related to operational costs, energy consumption, and environmental impact. Researchers propose various solutions to enable deployment in resource-constrained devices [4], such as architectural optimization [6] and quantization [11]. Furthermore, network-based approaches have emerged, such as [21] proposed the client-edge co-inference method, which partitions LLM into lightweight layers running on devices and parameter-heavy layers operating at the edge to enhance inference efficiency.

However, existing frameworks primarily focus on either single-modal data sources or single-query accuracy [27], neglecting the complex trade-off optimization of key resource indicators (e.g., response quality, latency, costs) [7] and failing to address the intricate relationships between multi-modal, multi-task, and multi-dialogue context (i.e., RQ1 and RQ2). Furthermore, [8, 27] employ datasets with fixed evaluation metrics, which may not fully represent the complexity in many real-world LLM applications. Moreover, online training methods incur prohibitively expensive costs and lead to irreproducible resource consumption, particularly when considering human subjects or alternative approaches for evaluating LLM inference (i.e., RQ3). In contrast to previous work, we propose an offline training approach to preserve LLM inference results. While this inevitably introduces uncertainties in the LLM inference data, we address this challenge by further incorporating an uncertainty estimator for response scores.

Inference Selection and Offloading Optimization. More generally, there is a rich set of literature on strategies for optimizing local-edge-cloud inference location selection/offloading, based on metrics such as performance, cost, and latency [22]. These works

explore diverse methodological approaches that can be categorized into network optimization strategies, such as confidence-based device-server selection [1] and energy-delay considerations for LLM training [12], as well as RL frameworks, such as accuracy-delay-fairness device-server selection [2] and multi-device offloading with quality-latency-energy considerations [5].

However, existing hierarchical inference approaches [1, 2], which orchestrate offloading between local devices and cloud servers, only consider model size differences. They neglect fundamental distinctions in their input modalities, for example, one scenario is device-side LLMs being limited to text-only inputs, while server-side models can process multimodal inputs (i.e., RQ1). Furthermore, recall that the unique challenge in our settings is multi-dialogue interactions, yet these methods primarily focus on meeting specific performance standards or finding *immediate* optimal solutions without considering *cumulative* reward maximization throughout the decision-making process. [8] employed active inference for LLM inference offloading but disregarded multi-modal data and mutli-task in multi-dialogues, failing to align with practical LLM scenarios (i.e., RQ2). Differently from prior works, we implement these concepts in a multi-dialogue LLM scenario, not only maximizing cumulative rewards, but also considering the modality-task associations to select the optimal LLM and multi-modal data sources.

3 Proposed TMO System

3.1 Multi-Modal, Multi-Task, Multi-Dialogue Interaction

We consider a setup in which a user is aiming to solve multiple types of inference tasks through a series of dialogues with LLMs. Each dialogue t involves:

- a text prompt P_t which describes the user's intent, e.g., "Recommend dishes that I can cook with my ingredients," and
- a set of other modalities $\mathcal{M}_t \subseteq \mathcal{M}$, where $\mathcal{M}$ represents all modalities available to the user (e.g., different image views, 3D points, or other data formats describing the current situation).

During the conversation with LLMs, a user may ask additional questions using prompts potentially related to the previous dialogues (e.g., "Where did I put my dishes?" or "Please turn on all the lights in the kitchen"). Here, we assume each text prompt belongs to one of N different task categories (e.g., assistant, recommendation, query, and message editing). Moreover, distinct tasks have varying levels of difficulty and require different types of multi-modal information to solve. For example, the prompt "Where did I put my dishes?" requires additional modalities (e.g., images) for the LLM to solve the task, while the prompt "Please turn on all the lights in the kitchen" may not require other sources of data, indicating that the task is simpler. These characteristics within a multi-task, multi-dialogue scenario with multi-modal data sources distinguish our setup from existing works [8, 26, 27] that consider simpler settings (e.g., a single dialogue with text only), motivating the need for a new solution.

3.2 Local-Cloud LLM Inference

As shown in Fig. 1, a key objective of TMO is to strategically distribute tasks between local and cloud platforms, to exploit their respective strengths. TMO consisiders two different LLMs:

- The **local LLM** is a lightweight LLM (e.g., Phi-3-mini) deployed on local devices, intended to handle simple tasks efficiently and process text prompt understanding with high speed.
- The **cloud LLM** is a large-scale multi-modal LLM (e.g., GPT-4o) situated in the cloud capable of managing complex tasks and processing information that exceeds local device capabilities, integrating multi-modal data input to provide richer and more accurate responses where required.

TMO determine whether to process LLM inference locally or offload to the cloud, depending on the task difficulty, available modalities, and resource requirements. Offloading computationally intensive tasks to the cloud enables preservation of local resources while still benefiting from the cloud powerful computational abilities.

Formally, we represent the user-LLM interaction at each dialogue t in terms of three possibilities:

$$R_t = \begin{cases} \mathrm{LLM}_{\mathrm{Local}}(P_t), & \text{for local inference,} \\ \mathrm{LLM}_{\mathrm{Cloud}}(P_t), & \text{for cloud text-only inference,} \\ \mathrm{LLM}_{\mathrm{Cloud}}\Big(P_t, \bigcup\limits_{m\in\mathcal{M}_t}\{D^{(m)}\}\Big), & \\ \qquad \text{for cloud inference with modalities in } \mathcal{M}_t, \end{cases} \tag{1}$$

where R_t represents the response to the particular combination of LLM and modalities provided for dialogue t. $D^{(m)}$ denotes the data source of modality $m \in \mathcal{M}_t$, while $\mathrm{LLM}_{\mathrm{Local}}(\cdot)$ and $\mathrm{LLM}_{\mathrm{Cloud}}(\cdot)$ are representations of the local LLM and the cloud LLM in functional form. The transformer architecture of LLM is known for its capability of maintaining a contextual memory over extended dialogues, thereby allowing the LLM to generate responses for the current prompt based on the accumulated context of all previous prompts and responses, though the quality of the responses will differ between local and cloud LLMs. Hence, Eq. (1) can be rewritten to reflect the cumulative nature of the prompts. For example, in the case of the cloud LLM with multi-modal inputs,

$$R_t = \mathrm{LLM}_{\mathrm{Cloud}}\left(P_t, \bigcup_{i=0}^{t}\bigcup_{m\in\mathcal{M}_i}\{D^{(m)}\}, \bigcup_{i=0}^{t-1}\{P_i, R_i\}\right), \tag{2}$$

where $t = 1, 2, \ldots, T$. This further shows the dependency on the previously selected modalities as well as prior prompts and responses from the series of earlier dialogues $i = 0, 1, \ldots, t-1$.

TMO Objective and Rationale. The primary objective of TMO is to strategically decide (i) whether to use the local LLM or cloud LLM and (ii) which multi-modal data sources to utilize, for each task and dialogue. This decision-making process aims to enhance response quality while efficiently managing latency and costs throughout extended interactions. The goal is to ensure high response quality in complex multi-dialogue environments, particularly under the system constraints (e.g., latency and costs).

The complexity of LLM and modality selection can be illustrated by the human decision-making process in such a setting. For example, iPhone users encounter choices among multiple LLMs (e.g., local Siri and the ChatGPT application), while also navigating various multi-modal data sources on mobile devices (e.g., front camera, rear camera, and screenshots). Although users can make short-term decisions for current tasks (for example, intuitively deciding to use the local LLM for simple message editing), the challenge intensifies when considering the entire spectrum of interactions

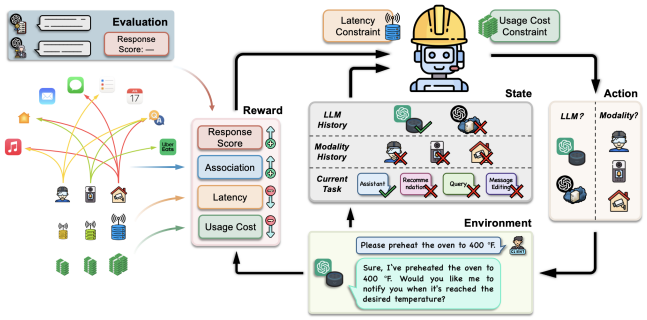


Figure 2: Illustration of RCRL within the TMO System.

over prolonged multi-dialogues and evolving contexts. Individuals must determine which of many multi-modal data sources to transmit and balance the quality of the response with other metrics, such as latency and various costs. Furthermore, early uploading of modality data to the cloud LLM can facilitate quicker and more nuanced understanding of the environment, enhancing long-term multi-dialogue interaction management. Optimal decision-making in these multi-modal, multi-task, and multi-dialogue scenarios is thus difficult by human inspection.

3.3 LLM and Modality Selection

We propose a resource-constrained RL (RCRL) approach to orchestrate the TMO system. Its key components are shown in Fig. 2, and Alg. 1 summarizes the process.

3.3.1 RL Formulation. We first formulate different components of the RL policy as follows.

State: The state captures both the history of previous LLM selections and the history of uploaded modalities, along with the current inference task category $n \in \{1, ..., N\}$. Formally, each state in the state space $\mathcal{S}$ can be visualized as a repeated sequence of the tuple $(\ell, \mathcal{M}^\ell, n)$. Thus, we can represent the state space $\mathcal{S}$ over τ consecutive dialogues as follows:

$$\mathcal{S} = (\ell, \mathcal{M}^\ell, n)^\tau, \tag{3}$$

where $\ell \in \{\text{LLM}_{\text{Local}}, \text{LLM}_{\text{Cloud}}\}$ indicates the choice between local LLM and cloud LLM. $\mathcal{M}^\ell = \mathcal{M}$ if $\ell = \text{LLM}_{\text{Cloud}}$ and $\mathcal{M}^\ell = \emptyset$ otherwise. (We will often refer to $\mathcal{M}^\ell$ as simply $\mathcal{M}$ when the context is clear.) The superscript τ indicates that there are τ state components over τ consecutive dialogues, showing the history of model and modality selections. The state $s_t \in \mathcal{S}$ for dialogue t is then represented by the tuple $s_t = (\ell, \mathcal{M}, n)_t$.

Action: The action taken for the next dialogue consists of the choice between utilizing local or cloud LLM services, as well as the selection of one or multiple data modalities for inference processing. Therefore, the action space $\mathcal{A}$ can be described as follows:

$$\mathcal{A} = (\ell, \mathcal{M}^\ell), \tag{4}$$

with the action $a_t = (\ell, \mathcal{M}^\ell)_t$ for dialogue t consisting of a choice of LLM and modalities to upload.

Reward: In designing the reward function, we consider several key metrics to optimize the decision-making process regarding the choice of LLM service and modalities. Mathematically, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that we propose is expressed as follows:

$$r_t = \alpha S_{R_t} + \beta_\Lambda \sum_{m \in \mathcal{M}_t} \Lambda\left(P_t, D^{(m)}\right) - \beta_\psi \psi_{a_t} - \beta_\phi \phi_{a_t}, \tag{5}$$

where:

- The **response score** S_{R_t} assesses how effectively the LLM response R_t (from either local or cloud) meets the requirements of the text prompt. The response score is used solely for RL training purposes and is not required during the inference phase. A more detailed description of this metric and how we estimate it is provided in Sec. 3.4.
- The **association score** $\Lambda(P_t, D^{(m)})$ quantifies the relevance of multi-modal data $D^{(m)}$ to prompt P_t. We aggregate the association scores for the uploaded modality set $\mathcal{M}_t$, which depend on the current action a_t. Further details are provided in Sec. 3.5.
- The **latency** ψ_{a_t} represents the latency of the LLM inference under different actions a_t. This includes the computation time of the local LLM or the interaction time with the cloud LLM for uploading the selected data modalities. Refer to Sec. 3.6 for the formal definition.
- The **usage cost** ϕ_{a_t} reflects the cost incurred by the LLM under action a_t, including the energy consumption of the local LLM or the service fee of the cloud LLM for various uploaded modalities, as described in Sec. 3.7.

The response score S_{R_t} and the association $\Lambda(P, D^{(m)})$ are thus positive indicators (higher is better, ↑). In contrast, latency ψ_{a_t} and usage cost ϕ_{a_t} are negative indicators (lower is better, ↓). Variables α, β_Λ, β_ψ, and β_ϕ are weighting factors that adjust the influence of each component on the total reward.

3.3.2 MDP and Optimization. Based on our state and action definitions, we model this problem as a markov decision process (MDP), where the learning objective is to maximize the cumulative reward. In each dialogue t, RL algorithms observe the state s_t, select an action a_t, obtain a transition probability $P(s_{t+1}|s_t, a_t)$, receive a reward r_t, and transition to the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$. Given the current state s_t, the action a_t is selected according to a specific policy π as $a \sim \pi(\cdot|s)$, where $\pi(a|s)$ represents the probability of selecting action a in state s. The value function $V^\pi(s)$ predicts the expected discounted rewards of states following the policy π:

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s\right] \tag{6}$$

with discount γ. The optimization of policy π_θ with respect to its parameterization θ aims to maximize the expected discounted rewards in LLM inference offloading and modality selection:

$$\max_\theta J(\theta) \triangleq \mathbb{E}_{s,a\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]. \tag{7}$$

In deep RL optimization, various strategies can be employed to adjust the parameters θ of the policy $\pi_\theta(a|s)$ with the goal of maximizing $J(\theta)$. To achieve this, a loss function $f(\theta)$ is defined such that minimizing $f(\theta)$ leads to maximizing $J(\theta)$. Taking Advantage Actor Critic (A2C) [15] as an example, the loss function $f_{\text{A2C}}(\theta)$ includes both the policy loss and the value loss, along with an entropy term to encourage exploration:

$$f_{\text{A2C}}(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[-\log \pi_\theta(a|s)A(s,a) + \tfrac{1}{2}(V^{\pi_\theta}(s) - r)^2 - \zeta H(\pi_\theta(\cdot|s))], \tag{8}$$

Algorithm 1 Local-Cloud Inference Offloading in TMO

Input: M4A1 Dataset ($\mathcal{D}$), initial policy parameters (π_θ), reward weights (α, β_Λ, β_ψ, β_ϕ), time span (τ), resource constraint penalty coefficient (λ), number of neighbors for response score estimation (k), learning rate (η)

Output: Refined policy π_θ for LLM and modality selection

1: **for** each time step **do**
2: Sample a mini-batch of transitions $\{(s, a, s')\}$ and corresponding metrics $\{(\Lambda, S_R, \psi_a, \phi_a)\}$ from $\mathcal{D}$.
3: Predict actions a' for new states s' via the policy $\pi_\theta(s')$.
4: Estimate uncertain response scores S'_R for pending state-action pairs (s', a') via nearest neighbors. ▷ Eq. (11)
5: Compute rewards r based on the estimated response scores S'_R, associations Λ, latencies ψ_a, and usage costs ϕ_a. ▷ Eq. (5)
6: Calculate the RL loss $f(\theta)$, latency penalty $g_1(\theta)$, and usage cost penalty $g_2(\theta)$. ▷ Eq. (10)
7: Update the policy parameters: $\theta \leftarrow \theta - \eta \nabla_\theta F(\theta)$.
8: **end for**

where $A(s, a) = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$ for $s' \neq s$ is an estimate of the advantage function, providing a measure of the relative value of taking action a in state s. The coefficient ζ controls the weight of the entropy bonus $H(\pi_\theta(\cdot|s))$. In Sec. 4.2, we also consider Proximal Policy Optimization (PPO) [20] and Deep Q Network (DQN) [16] versions of Eq. (8).

3.3.3 Resource-Constrained RL. We consider RCRL to enhance decision-making capabilities for TMO in environments with limited resources. RCRL involves not only maximizing cumulative rewards, but also satisfying certain constraints on resource consumption such as latency or usage cost incurred from processing different modalities. Formally, we consider a resource consumption function $C_j(a)$, which quantifies the amount of resource $j \in \mathcal{J}$ consumed when action a is taken. RCRL then aims to minimize the loss function $f(\theta)$ (e.g., Eq. (8) for the A2C loss) while ensuring compliance with resource constraints, by solving

$$\min_\theta f(\theta) \quad \text{s.t.} \quad \sum_{i=t}^{t+\tau} C_j(a_i) \leq \xi_j, \text{ for all } j \in \mathcal{J}, \tag{9}$$

where $C_j(a)$ denotes the consumption of the j-th constrained resource with ξ_j as the corresponding budget over the horizon, and $\mathcal{J}$ is the set of all constraints. τ is the planning horizon length, defined in Eq. (3). We transform resource constraints into a policy optimization problem using a penalty term to enforce constraints. Specifically, we define $g_j(\theta) \triangleq \sum_{i=t}^{t+\tau} C_j(a_i) - \xi_j$ to penalize constraint violations and aim to minimize the overall loss $F(\theta)$:

$$F(\theta) = f(\theta) + \lambda \sum_{j \in \mathcal{J}} \max\left\{g_j(\theta), 0\right\}, \tag{10}$$

where $\lambda \geq 0$ denotes the penalty associated with the constraints. In our experiments, we will consider two constraint penalty terms $g_1(\theta)$ and $g_2(\theta)$ on latency and usage cost, respectively.

3.4 Response Score Uncertainty Estimation

Due to the vast number of parameters in LLMs, offline training approaches have become essential to mitigate substantial inference costs and enable future reusability [17]. TMO's RCRL procedure requires a specialized dataset for learning and exploration of LLM performance in multi-modal, multi-task, and multi-dialogue settings. Moreover, given the nature of multi-task settings, particularly in generative tasks, there often exist no explicit evaluation metrics, such as accuracy, task success rate, or error rate. Furthermore, the exploratory nature of RL, which frequently encounters new state-action pairs, presents significant evaluation challenges.

To address these issues, we propose leveraging the notion of *LLM-as-Judge* to evaluate response quality. Following this approach, we employ a nearest neighbor strategy to estimate uncertain response scores, providing a method for handling the inherent uncertainties in four LLMs: the outputs of the local LLM, the cloud LLM, a reference LLM, and a judge LLM.

3.4.1 Response Quality Evaluation. The response score S_{R_t} measures the response quality generated by the selected LLM in each dialogue. In multi-task scenarios, evaluating the response quality requires assessing the actual utility of responses in aiding human users, particularly in query tasks. We employ the quantitative evaluation scheme outlined by [13], illustrated on the right side of Fig. 3. This involves using two additional *text-only* LLMs specifically to assess the quality of candidate LLM services:

- The *reference LLM* processes a prompt and a reference response to produce a polished reference response, serving as a benchmark for evaluation.
- The *judge LLM* then evaluates both the candidate's response and the reference's response, scoring them based on helpfulness, relevance, accuracy, and detail.

The response score S_{R_t} for the dialogue t is then calculated as the ratio of the candidate's LLM score to the reference score.

3.4.2 Handling Uncertainty in Response Score Estimation. Integrating RCRL directly into real-time LLM inference is impractical, especially in our scenario where four LLMs operate in tandem: local and cloud-based LLMs generate responses, while reference and judge LLMs evaluate their quality. Consequently, we archive the LLM inference processes into a dataset for offline RCRL. However, this approach introduces two main forms of uncertainty:

(i) *non-deterministic evaluation (NDE)*, in which identical state-action pairs can yield different response quality scores; and
(ii) the potential presence of *out-of-distribution (OOD)* state-action pairs in the dataset.

In contrast to [19], which aims to avoid selecting OOD actions, our goal is to more accurately estimate the response quality under OOD actions rather than constrain them. This adds flexibility for datasets where the states and actions are randomly generated (e.g., the current task, selected LLM, and selected modality), making it inevitable that some state-action pairs will not appear in the dataset (such as for our M4A1 dataset described in Sec. 4.1).

To address these uncertainties, we estimate the response score S'_R for a pending state-action pair (s', a') by leveraging its nearest neighbors in the dataset. Specifically, let $\{(s^i, a^i)\}_{i=1}^k$ be the k state-action pairs nearest to (s', a') in Euclidean distance, and let $\{S_{R^i}\}_{i=1}^k$ denote their known response scores. We estimate S'_R as a weighted

average over its neighbors:

$$S'_R = \frac{\sum_{i=1}^{k} S_{R^i}/d_i}{\sum_{i=1}^{k} 1/d_i}, \quad \text{where} \quad d_i = \left\| [s'\ a'] - [s^i\ a^i] \right\| \tag{11}$$

represents the distance between the k nearest state-action pairs and the pending state-action pair (s', a'), with $[x\ y]$ denoting vector concatenation. In this way, we achieve a more robust offline estimation of response quality, effectively handling both NDE and OOD uncertainties within RCRL for LLM inference.

3.5 Task-Modality Association

TMO is designed to operate in multi-modal, multi-task scenarios. In such settings, quantifying the relevance between users' text prompts and multi-modal data sources enables selection of the most pertinent multi-modal data sources to offload to the cloud LLM. Consequently, we employ feature extractors to transform both tasks and modalities into feature vectors of identical dimensionality for similarity computation. Given this premise, we leverage the pre-trained CLIP model [18], a transformer-based architecture with text and image encoders, to extract features from prompts and multi-model data sources. We employ a normalized cross-modality similarity metric to calculate the association score between a text prompt P and data modality m, which is defined as:

$$\Lambda\left(P, D^{(m)}\right) = \frac{\left\langle \omega_T(P), \omega_I(D^{(m)}) \right\rangle}{\|\omega_T(P)\| \cdot \left\|\omega_I(D^{(m)})\right\|}, \tag{12}$$

where $\omega_T : P \to \mathbb{R}^d$ and $\omega_I : D^{(m)} \to \mathbb{R}^d$ represent the text and image encoders, respectively, mapping inputs to a shared d-dimensional embedding space.

3.6 Latency Model

3.6.1 Local Computation Latency ψ_{Local}. The computational latency of the local LLM, or say inference time, is influenced not only by the number of LLM parameters, but also significantly by the length of the prompt, denoted $|P|$, and of the response, $|R|$. Following [3], we assume that the computational latency is proportional to the number of parameters. If the local LLM is comprised of $|\text{LLM}_{\text{Local}}|$ parameters and requires $2|\text{LLM}_{\text{Local}}|$ floating point operations (FLOPS) for forward propagation, the computational demand for each token in a reference processing length $|P|_{\text{ref}}$ is calculated as $\frac{2|\text{LLM}_{\text{Local}}|}{|P|_{\text{ref}}}$ FLOPS. The latency calculation considers the theoretical peak performance of the local device's GPU, TF_{peak} FLOPS. The computational latency can be computed as:

$$\psi_{\text{Local}} = \frac{2|\text{LLM}_{\text{Local}}|(|P| + |R|)}{|P|_{\text{ref}} \times TF_{\text{peak}}}. \tag{13}$$

3.6.2 Cloud Interaction Latency ψ_{Cloud}. The interaction time with the cloud LLM encompasses the entire process from sending the prompt to receiving the response. This includes uploading the prompt and modalities, inference by the cloud LLM, and downloading the response. We directly recorded the interaction time with the cloud LLM for different sets of uploaded modalities $\mathcal{M}$ during the creation of our M4A1 dataset.

3.7 Usage Cost Model

During the inference process, LLMs incur computational costs. From the user's perspective, for the local LLM, this manifests as energy consumption, while for the cloud LLM, it translates into the service fee paid to the provider.

3.7.1 Local LLM Usage Cost ϕ_{Local}. The energy consumption of the local LLM is influenced by inference time and the power consumption of the computing hardware, which are in turn affected by the local device's computational capacity, the sizes of the prompt and response, and the size of the LLM, all encapsulated within the inference time ψ_{Local}. Let the maximum power consumption for a given local device hardware be $W_{\max}$ Watts. By utilizing a normalized energy cost factor κ (capturing e.g., local electricity rates and device wear-and-tear), after appropriate unit conversion, the local LLM usage cost (in USD) is defined as:

$$\phi_{\text{Local}} = \psi_{\text{Local}} \times W_{\max} \times \kappa. \tag{14}$$

3.7.2 Cloud LLM Usage Function ϕ_{Cloud}. For cloud-based LLMs, in addition to the inference costs of prompts and responses, the inference costs of data modalities must also be considered. These costs are typically available on the service provider's website. Taking GPT-4o as an example, the prompt cost rate is $\varphi_P = \$0.005/1K$ tokens, the response cost rate is $\varphi_R = \$0.015/1K$ tokens, and the data modality costs are proportional to their size and resolution at a rate φ_m for modality $m \in \mathcal{M}$. Therefore, the usage cost of the cloud LLM can be defined as follows:

$$\phi_{\text{Cloud}} = \varphi_P \cdot |P| + \varphi_R \cdot |R| + \sum_{m \in \mathcal{M}} \varphi_m \cdot \left|D^{(m)}\right|. \tag{15}$$

Summary of Reward Function. Revisiting our initially defined reward function in Eq. (5), we substitute the direct response score S_{R_t} with the estimated response score S'_{R_t} defined in Eq. (11) to achieve a more precise representation. The association metric is quantified through Eq. (12). Both the latency formulations (ψ_{Local} and ψ_{Cloud}) and usage cost formulations (ϕ_{Local} and ϕ_{Cloud}) are incorporated as ψ_{a_t} and ϕ_{a_t} in the reward function, respectively.

4 Experimental Results

4.1 M4A1 Dataset

To evaluate performance in practical settings, we develop M4A1, the first multi-modal, multi-task, multi-dialogue, and multi-LLM dataset, as shown in Fig. 3. M4A1 consists of 21,014 dialogues, each meticulously compiled by randomly and sequentially choosing tasks, prompts, modalities, and candidate LLMs to ensure comprehensive coverage across different configurations. The key characteristics of M4A1 can be summarized as follows:

- **Three Multi-Modal Data Sources** consist of images from different views in a unified scenario, including first-person, side, and overhead views.
- **Four Tasks** include Assistant, Recommendation, Query, and Message Editing, represent varying levels of task difficulty, relevance to modalities, and the types of actions that the LLM will undertake.
- **Two to Five Dialogues** include pairs of text prompts and responses within a continuous context.
- **Four LLMs** are a local LLM (Phi-3-mini), a cloud LLM (GPT-4o), a reference LLM (GPT-4o), and a judge LLM (GPT-4o).
- **Four Metrics** include response score (Sec. 3.4), association (Sec. 3.5), latency (Sec. 3.6), and usage cost (Sec. 3.7).

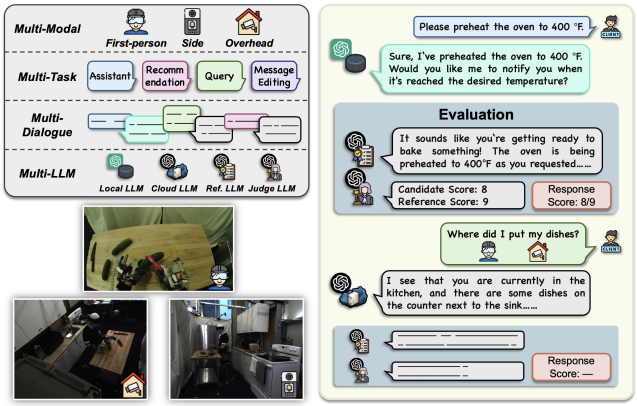


Figure 3: Illustration of M4A1 Dataset.

4.2 Experiment Setup

Training Setup. The M4A1 dataset is split into a training set and a test set in an 8:2 ratio. The training set is used to train RLs while the test set is used to evaluate the performance at inference. When implementing our approach, we consider several popular discrete RL algorithms, including PPO [20], DQN [16], and A2C [15]. We consider these methods with and without resource constraints (corresponding to cases with $\lambda > 0$ and $\lambda = 0$ in Eq. (10)), and denote RC-PPO, RC-DQN, and RC-A2C as the methods with the resource constraints. The policies are parameterized by multilayer perceptrons (MLP), with a total of 30,000 time steps set for training. For each experiment, we repeat the process three times and report the standard deviation as the error bar. We implement the TMO system and the baselines in PyTorch and conducted the experiments on an NVIDIA A100 GPU with 40 GB of memory.

Baselines. We compare the RL policies with three naive baselines: random (with random LLM and modality selection), local (using only the local LLM), and cloud (using only the cloud LLM, but with random modality selection). Inspired by recent advances in *LLM-as-Agent* [14, 25], we substitute the RL agent with an LLM agent as TMO's decision engine for the selection of LLMs and modalities. Specifically, current states and possible actions are translated into natural language descriptions as contextual input for the LLM agents, which then output an action in response to this prompt, following an RL-like procedure. The LLM agents include Phi-3-mini, Phi-3.5-mini, LLaMA-3.2-3B, LLaMA-3.1-8B, Mistral-7B-v0.3, FLAN-T5-large, FLAN-T5-xl, Gemma-2-2B, Gemma-2-9B, GPT-3.5-turbo, GPT-4o-mini, GPT-4o, OpenAI o1-mini, OpenAI o1, and OpenAI o3-mini. We reiterate that none of the existing works can be directly applied to our setting, which features multi-modal, multi-task, and multi-dialogue characteristics. Therefore, we reimplement AIwRG [8], an active inference method, and PerLLM [26], a bandit-based approach using upper confidence bound (UCB) for exploration-exploitation trade-off, integrating the settings introduced in this paper to adapt them to our research problem.

Default Settings. The default experimental settings are as follows unless otherwise stated; we also study the effects of various system parameters. Specifically, we use RC-A2C as the RL policy, with $\alpha = 1$, $\beta_\Lambda = 1/3$, $\beta_\psi = 1/3$, and $\beta_\phi = 1/3$ for the reward function weights, a 30 second latency constraint, a 0.05 USD usage cost constraint, and Jetson TX2 as the device type. These settings serve as the baseline from which variations are systematically introduced to assess their effects. Additionally, we set the time span $\tau = 5$, the penalty coefficient $\lambda = 10$, $k = 5$ nearest neighbors for estimating the response score, and $\kappa = 4.63 \times 10^{-8}$ (Joules per USD) based on the average electricity price in the US [1].

4.3 Main Experimental Results

Table 1 presents the main results of this study, with key takeaways described from four perspectives.

Comparison with Naive Baselines. Compared to the local baseline, the proposed methods can significantly improve the response score by sacrificing latency and usage cost, thus enhancing the overall reward. Compared to the cloud baseline, we also boost the response score and conserve resources by selecting the appropriate modalities, which leads to improved rewards. In contrast to the random baseline, our approach strategically explores the relationships between tasks/dialogues and LLMs/modalities, thereby optimizing decision-making processes to better align with the specific requirements and constraints of each task.

Comparison with SOTA LLM-as-Agent Baselines. We replace the RL agent in TMO with an LLM agent for decision-making processes. Our goal is to evaluate the zero-shot LLM agent's ability to consider task-modality associations, understand contexts, and handle numerically sensitive tasks (e.g., avoiding resource constraint violations). As shown in Table 1, different LLM agents exhibit substantial variability in performance. Some agents ignore all modalities (e.g., Mistral-7B-v0.3, Gemma-2-2b, Gemma-2-9b, GPT-4o-mini, GPT-4o, OpenAI o1-mini, OpenAI o1, and OpenAI o3-mini), while others choose to upload all available modalities for each task (e.g., FLAN-T5-large). Although some agents (e.g., Phi-3-mini, Phi-3.5-mini, and FLAN-T5-xl) appear to select different LLMs and modalities without violating any constraints, their response scores and overall rewards do not confer significant advantages. Thus, these zero-shot LLM agents do not demonstrate clear benefits. One reason is that they are not fine-tuned in the training dataset. Another reason is that, as generative models, LLM agents lack precise numerical calculation capabilities. Consequently, some agents violate constraints, whereas others adopt overly conservative strategies.

Comparison with SOTA Exploration-Decision Baselines. Existing works do not consider multi-task and multi-dialogue settings in a multi-modal scenario, and cannot be trivially deployed to our setting. For example, [8] utilize a benchmark with a fixed evaluation metric Pass@k that measures success rates. In contrast, our method addresses a multi-modal multi-task and multi-dialogue setting which introduces significant challenges related to response score evaluation and uncertainty. From the results, our TMO system significantly outperforms these SOTA baselines. [8] fails to efficiently select LLMs and modalities, frequently opting to upload all three types of modality data sources for most actions. This results in substantial resource waste and redundancy in uploading unnecessary information to the cloud. Conversely, [26] rarely considers any modality data, relying solely on the local LLM and text-only cloud LLM. Although this approach avoids violating any resource

[1] https://www.energybot.com/electricity-rates/

Table 1: Main Result - TMO with RC-A2C optimally balances local/cloud LLMs and adaptively selects modalities, outperforming in response quality, latency, and costs without constraint violations. The highlighted row indicates the highest overall reward without constraint violations.

Method	Response Score (↑)	Latency (s) (↓)	Usage Cost (1e-3 USD) (↓)	Overall Reward (↑)	Local	Cloud - Num. Selected Modalities				Constraint Violation (↓)	
						0 (text-only)	1	2	3	Latency	Usage Cost
Random	0.86 ±0.02	10.00 ±0.25	12.32 ±0.23	0.71 ±0.01	2097	261	775	771	255	0.95 ±0.13	0.89 ±0.83
Local	0.74 ±0.04	0.04 ±0.00	0.00 ±0.00	0.74 ±0.04	4203	0	0	0	0	0.00 ±0.00	0.00 ±0.00
Cloud	1.04 ±0.01	20.24 ±0.18	25.14 ±0.32	0.75 ±0.01	0	515	1583	1540	544	1.11 ±0.05	2.69 ±0.34
LLM-as-Agent [14, 25] for Inference Offloading (Ignored LLM Agent's Latency and Usage Cost)											
Phi-3-mini	0.81 ±0.04	5.25 ±0.31	7.54 ±0.44	0.74 ±0.04	3089	0	613	207	295	0.00 ±0.00	0.00 ±0.00
Phi-3.5-mini	0.83 ±0.04	5.63 ±0.38	8.39 ±0.57	0.76 ±0.04	3118	0	42	1044	0	0.00 ±0.00	0.00 ±0.00
LLaMA-3.2-3B	1.05 ±0.02	22.04 ±0.27	35.08 ±0.39	0.74 ±0.02	78	101	58	2955	1011	2.11 ±0.04	4.02 ±0.16
LLaMA-3.1-8B	0.89 ±0.02	10.84 ±0.18	12.41 ±0.31	0.74 ±0.02	1933	381	1017	545	326	1.54 ±0.22	3.79 ±1.02
Mistral-7B-v0.3	0.83 ±0.03	14.46 ±0.28	1.10 ±0.02	0.64 ±0.03	1519	2684	0	0	0	0.00 ±0.00	0.00 ±0.00
FLAN-T5-large	1.01 ±0.04	24.45 ±0.28	47.91 ±0.55	0.68 ±0.04	0	0	0	0	4203	4.90 ±0.01	11.14 ±0.36
FLAN-T5-xl	0.85 ±0.05	12.95 ±0.15	9.59 ±0.27	0.65 ±0.05	1452	1053	1408	0	289	0.00 ±0.00	0.00 ±0.00
Gemma-2-2b	0.74 ±0.04	0.04 ±0.00	0.00 ±0.00	0.74 ±0.04	4203	0	0	0	0	0.00 ±0.00	0.00 ±0.00
Gemma-2-9b	0.74 ±0.04	0.04 ±0.00	0.00 ±0.00	0.74 ±0.04	4203	0	0	0	0	0.00 ±0.00	0.00 ±0.00
GPT-3.5-turbo	0.99 ±0.02	19.96 ±0.51	33.75 ±0.66	0.73 ±0.01	527	190	504	692	2292	2.97 ±0.15	5.86 ±0.39
GPT-4o-mini	0.74 ±0.04	0.04 ±0.00	0.00 ±0.00	0.74 ±0.04	4203	0	0	0	0	0.00 ±0.00	0.00 ±0.00
GPT-4o	0.85 ±0.05	12.92 ±0.29	0.98 ±0.02	0.67 ±0.05	1807	2396	0	0	0	2.26 ±0.01	0.00 ±0.00
OpenAI o1-mini *	0.77 ±0.03	8.21 ±0.43	0.62 ±0.03	0.66 ±0.03	222	127	0	0	0	1.13 ±0.81	0.00 ±0.00
OpenAI o1 *	0.76 ±0.02	5.18 ±0.16	0.39 ±0.01	0.69 ±0.02	269	80	0	0	0	0.38 ±0.54	0.00 ±0.00
OpenAI o3-mini *	0.86 ±0.04	16.91 ±1.18	1.28 ±0.09	0.63 ±0.03	87	262	0	0	0	2.09 ±0.06	0.00 ±0.00
Comparison with SOTA Exploration-Decision Baselines											
AIwRG [8]	1.02 ±0.05	23.73 ±0.98	44.01 ±3.50	0.69 ±0.06	113	244	0	0	3852	4.68 ±0.18	9.26 ±1.83
PerLLM [26]	0.97 ±0.06	21.12 ±0.08	1.60 ±0.01	0.66 ±0.06	281	3922	0	0	0	0.00 ±0.00	0.00 ±0.00
TMO (PPO)	1.02 ±0.00	17.89 ±0.82	22.48 ±1.48	0.76 ±0.01	142	40	2499	1355	121	0.70 ±0.77	1.66 ±2.35
TMO (DQN)	1.02 ±0.08	19.57 ±2.18	26.23 ±6.36	0.75 ±0.05	0	5	1820	2367	0	0.81 ±0.57	0.00 ±0.00
TMO (A2C)	1.05 ±0.03	18.81 ±3.44	26.44 ±13.05	0.79 ±0.07	98	0	2753	210	1168	1.48 ±2.09	2.38 ±3.36
TMO (RC-PPO)	1.03 ±0.03	18.33 ±1.87	22.43 ±3.57	0.77 ±0.04	90	105	2599	1309	110	0.45 ±0.64	0.00 ±0.00
TMO (RC-DQN)	1.05 ±0.05	18.87 ±0.81	23.29 ±3.59	0.78 ±0.06	46	174	2249	1636	85	0.53 ±0.33	0.42 ±0.60
TMO (RC-A2C)	1.09 ±0.08	16.63 ±0.67	17.97 ±1.18	**0.85 ±0.07**	74	12	3871	225	6	0.00 ±0.00	0.00 ±0.00
Our TMO System - Ablation Study (Using RC-A2C as Backbone)											
w/o LLM Sel.	0.85 ±0.00	9.25 ±1.02	11.67 ±3.14	0.72 ±0.02	2128	6	1228	830	1	0.00 ±0.00	0.00 ±0.00
w/o Modality Sel.	1.04 ±0.02	20.13 ±0.07	24.82 ±0.21	0.75 ±0.02	0	528	1591	1519	528	1.17 ±0.07	2.74 ±0.87
w/o Score Est.	1.01 ±0.01	16.71 ±0.28	17.76 ±0.59	0.76 ±0.01	0	0	4095	89	0	0.00 ±0.00	0.00 ±0.00

* Due to the considerable inference time and usage cost of OpenAI o1-mini, OpenAI o1, and OpenAI o3-mini, we randomly sample 100 instances for evaluation.

constraints, it leads to lower response scores, as the LLM lacks access to any information derived from modality data.

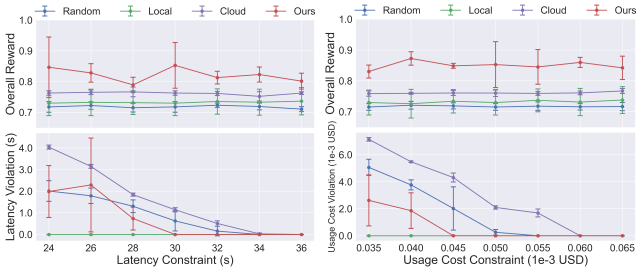
Analysis of Our TMO System. Most RL algorithms prefer to utilize the cloud LLM rather than the local LLM, as the gain in response score from the cloud LLM is much greater than the resource savings achieved by the local LLM. Regarding modality selection, few RL algorithms choose a text-only cloud LLM, as it often results in interaction times comparable to or even longer than those with multi-modal inputs. This largely depends on the inference speed of the service provider rather than factors like network bandwidth. A potential reason is that text-only and multi-modal cloud LLMs may operate on different workflows and versions. However, in cases involving multi-modal inputs, we observe that single modality selection often receives preferential treatment by RL algorithms, as it provides valuable information with less latency and lower usage costs compared to selecting two or three modalities. RC-A2C demonstrates the best response scores and overall rewards while ensuring compliance with latency and usage cost constraints.

4.4 Ablation Studies

To validate our approach, we conduct ablation studies (Table 1, last three rows) by removing key components: RQ1 – LLM selection (substituting with random LLM selection), RQ2 – modality selection (using random modality selection instead), and RQ3 – uncertain response score estimation (using average action scores). Random LLM selection degrades response scores as local LLMs often lack task capabilities. Random LLM selection degrades response scores as local LLMs often lack task capabilities. Lastly, using average response scores fails to account for contextual information, particularly historical modality uploads, leading to biased estimations and degraded performance. Our results demonstrate that effective LLM selection matches tasks with appropriately capable LLMs, while modality selection enhances modality-task alignment, minimizing latency and costs. Additionally, uncertain response score estimation leverages contextual information for more accurate estimations and informed decisions.

Table 2: Trade-off between Metrics in Reward Function.

β_Λ	β_ψ	β_ϕ	Response Score (↑)	Association (↑)	Latency (s) (↓)	Usage Cost (1e-3 USD) (↓)
0	0.5	0.5	0.76 ±0.06	0.02 ±0.03	1.59 ±2.20	1.63 ±2.31
0.5	0	0.5	1.05 ±0.07	0.39 ±0.10	21.04 ±1.77	31.91 ±7.11
0.5	0.5	0	0.98 ±0.04	0.55 ±0.05	23.26 ±0.92	42.97 ±3.27
1/3	1/3	1/3	1.09 ±0.08	0.21 ±0.01	16.63 ±0.67	17.97 ±1.18



(a) Latency Constraint (b) Usage Cost Constraint

Figure 4: Effect of Constraints. TMO demonstrates a faster reduction in resource constraint violations compared to baselines.

4.5 Trade-off Between Metrics

In Table 2, we investigate the trade-offs among different metrics in the reward function by adjusting the weights for association, latency, and usage cost in Eq. (5), denoted as β_Λ, β_ψ, and β_ϕ, respectively. We fix the weight of the response score to $\alpha = 1$. Due to the presence of resource constraints, simply setting $\beta_\psi = 0$ or $\beta_\phi = 0$ not only causes high latency or high usage cost, respectively, but also does not provide a benefit to the response score. Although the RL policy is penalized by the resource constraint in the loss function, it lacks explicit action-level penalties. As a result, the policy may only account for cumulative costs rather than recognizing individual action expenses, because uploading three modality data sources sequentially or simultaneously incurs roughly the same total resource cost. Consequently, when we balance the weights by setting $\beta_\Lambda = \frac{1}{3}, \beta_\psi = \frac{1}{3}, \beta_\phi = \frac{1}{3}$, we can attain a high response score while maintaining relatively lower latency and usage costs.

4.6 Effect of Resource Constraints

In Fig. 4, we evaluate the performance of different methods under varying latency and usage cost constraints, reflecting real-world scenarios where applications operate under different budgetary considerations. As latency and usage cost constraints increase, all methods demonstrate reduced constraint violations, with ours achieving more rapid reduction. This improvement stems from resource constraints incorporated into the loss function to balance the long-term cumulative rewards. In all different constraint settings, the proposed approach achieves the highest overall rewards, demonstrating its effectiveness in practical system deployments by considering response quality, latency, and usage cost altogether. However, under extremely limited resources, ours still struggles to effectively balance the trade-off between response score and resource costs. Therefore, in practical system deployments, beyond employing resource constraints to optimize long-term cumulative reward, establishing safety layers is also essential.

Table 3: Local Devices Setting.

Local Device	Performance (TFLOPS)	Power (Watts)	Latency (s)	Usage Cost (1e-3 USD)
Raspberry Pi-4B	0.0135	8	1.12593	4.17e-4
Raspberry Pi-5	0.0314	12	0.48408	2.69e-4
Jetson Nano	0.472	10	0.03220	1.49e-5
Jetson TX2	1.33	15	0.01143	7.94e-6
Jetson Xavier NX	21	20	0.00072	6.71e-7
Jetson Orin NX	100	25	0.00015	1.76e-7
iPhone 15 (A16)	15.8	15	0.00096	6.69e-7
iPhone 15 Pro (A17 Pro)	35	15	0.00043	3.02e-7

(a) Response Score & Overall Reward (b) Latency & Usage Cost

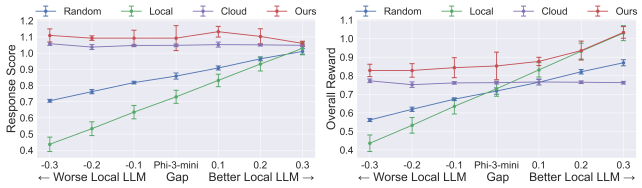
Figure 5: Effect of local device performance. TMO achieves superior performance on all local device compared to baselines.

4.7 Effect of Different Local Devices

To further confirm the advantage of our approach, in Fig. 5, we explore the effects of using different types of local devices, with detailed specifications available in Table 3. We infer that the performance of different devices does not significantly affect the operation of the TMO system, since their response scores and overall rewards do not show statistical significance. The underlying rationality is that for any device, latency and usage cost are normalized through min-max scaling and incorporated as part of the overall reward. The results are consistent with those in Table 1, showing that our approach achieves (i) significant latency and cost savings compared to the cloud LLM baseline and (ii) significant improvements in response quality compared to the local LLM baseline, leading to the highest overall reward. Therefore, when the latency and usage cost of a local LLM are substantially lower than those of a cloud LLM, the TMO system can operate effectively on any device, a precondition that forms part of the background for the TMO system.

4.8 Effect of Local LLM Response Quality

Finally, we evaluate the TMO system's performance by simulating varying local LLM response qualities. Fig. 6 demonstrates these results, wherein we modify the response scores in the M4A1 dataset by applying specific adjustments (represented as the "gap" in the figure) to simulate different local LLM capabilities. The TMO system consistently outperforms all three baselines regardless of local and cloud LLM response quality, demonstrating its effectiveness across any combination of local and cloud LLMs. When the local LLM response quality is substantially lower than the cloud LLM quality, TMO consistently selects the cloud LLM and proceeds with the modality selection. As the quality of the local LLM response approaches the quality of the cloud LLM, TMO balances between the local and cloud options. An extreme scenario would be using identical LLMs locally and in the cloud, TMO is still capable of delivering a superior response quality. This holds under the assumption that the



(a) Response Score (b) Overall Reward

Figure 6: Effect of response quality of local LLM. TMO outperforms baselines across various deployments of local and cloud LLMs.

local LLM does not support multi-modal inputs (or, considering a broader perspective, the cloud LLM cannot access user multi-modal data due to privacy concerns), which reflects a distinctive challenge described in RQ1. These results confirm our methodology's applicability across diverse local LLM levels.

5 Conclusion

In this paper, we developed TMO, a novel local-cloud LLM inference offloading system designed specifically for multi-modal, multi-task, and multi-dialogue scenarios. We formulated the joint LLM and modality selection process as a resource-constrained reinforcement learning (RCRL) problem that maximizes long-term cumulative reward (response quality, latency, and usage cost) under practical user-defined constraints. Additionally, we curated a novel dataset, M4A1, enabling evaluation of TMO system performance across various configurations of modality availability, tasks and dialogues, resource constraints, and LLM response qualities. We demonstrated improvements in the overall reward obtained by TMO over several baselines. Future research will investigate multi-user collaborative inference through resource sharing and examine multi-server adaptive selection based on latency, cost, and other considerations.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under grant CPS-2313109, by the Office of Naval Research (ONR) under grant N00014-23-C-1016, and by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-24-1-0083.

References

[1] Ghina Al-Atat, Puranjay Datta, Sharayu Moharir, and Jaya Prakash Champati. 2024. Regret bounds for online learning for hierarchical inference. In *Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 281–290.

[2] Hasan Burhan Beytur, Ahmet Gunhan Aydin, Gustavo de Veciana, and Haris Vikalo. 2024. Optimization of offloading policies for accuracy-delay tradeoffs in hierarchical inference. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 1989–1998.

[3] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. 2016. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678* (2016).

[4] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176* (2023).

[5] Ning Chen, Sheng Zhang, Yu Liang, Jie Wu, Yu Chen, Yuting Yan, Zhuzhong Qian, and Sanglu Lu. 2024. TileSR: Accelerate On-Device Super-Resolution with Parallel Offloading in Tile Granularity. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2538–2547.

[6] Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. 2023. Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886* (2023).

[7] Qifei Dong, Xiangliang Chen, and Mahadev Satyanarayanan. 2024. Creating edge ai from cloud-based llms. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications*. 8–13.

[8] Ying He, Jingcheng Fang, F Richard Yu, and Victor C Leung. 2024. Large language models (llms) inference offloading and resource allocation in cloud-edge computing: An active inference approach. *IEEE Transactions on Mobile Computing* (2024).

[9] Mengting Hu, Zhen Zhang, Shiwan Zhao, Minlie Huang, and Bingzhe Wu. 2023. Uncertainty in natural language processing: Sources, quantification, and applications. *arXiv preprint arXiv:2306.04459* (2023).

[10] Jaewook Lee, Jun Wang, Elizabeth Brown, Liam Chu, Sebastian S Rodriguez, and Jon E Froehlich. 2024. GazePointAR: A Context-Aware Multimodal Voice Assistant for Pronoun Disambiguation in Wearable Augmented Reality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–20.

[11] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems* 6 (2024), 87–100.

[12] Chang Liu and Jun Zhao. 2024. Resource allocation for stable llm training in mobile edge computing. In *Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 81–90.

[13] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems* 36 (2024).

[14] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* (2023).

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[17] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[19] Yuhang Ran, Yi-Chen Li, Fuxiang Zhang, Zongzhang Zhang, and Yang Yu. 2023. Policy regularization with dataset constraint for offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 28701–28717.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[21] Yifei Shen, Jiawei Shao, Xinjie Zhang, Zehong Lin, Hao Pan, Dongsheng Li, Jun Zhang, and Khaled B Letaief. 2024. Large language models empowered autonomous edge ai for connected intelligence. *IEEE Communications Magazine* (2024).

[22] Chetna Singhal, Yashuo Wu, Francesco Malandrino, Marco Levorato, and Carla Fabiana Chiasserini. 2024. Resource-aware deployment of dynamic dnns over multi-tiered interconnected systems. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 1621–1630.

[23] Ningxin Su, Chenghao Hu, Baochun Li, and Bo Li. 2024. TITANIC: Towards production federated learning with large language models. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 611–620.

[24] Zefan Wang, Yitong Wang, and Jun Zhao. 2024. Resource Allocation and Secure Wireless Communication in the Large Model based Mobile Edge Computing System. In *Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 111–120.

[25] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864* (2023).

[26] Zheming Yang, Yuanhao Yang, Chang Zhao, Qi Guo, Wenkai He, and Wen Ji. 2024. PerLLM: Personalized Inference Scheduling with Edge-Cloud Collaboration for Diverse LLM Services. *arXiv preprint arXiv:2405.14636* (2024).

[27] Xuechen Zhang, Zijian Huang, Ege Onur Taga, Carlee Joe-Wong, Samet Oymak, and Jiasi Chen. 2024. TREACLE: Thrifty Reasoning via Context-Aware LLM and Prompt Selection. *arXiv preprint arXiv:2404.13082* (2024).