Uncertainty-Aware Opportunistic Hybrid Language Model in Wireless Robotic Systems

Jeyoung Park [1] **Yeonsub Lim** [2] **Seungeun Oh** [2] **Jihong Park** [3] **Jinho Choi** [4] **Seong-Lyun Kim** [2]

Abstract

The hybrid language model (HLM) is an emerging architecture that efficiently distributes computation between on-device small language models (SLMs) and remote large language models (LLMs). In HLM, an SLM drafts tokens and its paired LLM validates and refines them, thereby achieving higher token throughput than LLMs and higher inference accuracy than SLMs. Recently, the uncertainty-aware opportunistic HLM (U-HLM) has been proposed to improve communication and computation efficiency by skipping LLM verification when the SLM's uncertainty is low. However, this approach has only been evaluated on simple text prediction tasks under a statistical channel model for theoretical analysis. To validate the practical feasibility of U-HLM, in this paper, we implement U-HLM on a real-world robot testbed, where an industrial-grade robotic manipulator (high-precision robot arm with gripper) runs an SLM and communicates with a remote LLM over Wi-Fi. In this experimental setup, we observe that computing uncertainty itself incurs non-negligible latency. To mitigate this, we propose a conditional uncertainty calculation omission method, which skips the uncertainty calculation when a lightweight logistic regression model predicts the uncertainty to be sufficiently low. Experimental results show that, compared to HLM, the proposed U-HLM improves token throughput by 24.9% and 41.8% under strong and weak Wi-Fi coverage conditions, respectively, while maintaining a 98.11% F1 score.

[1]Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, Ontario, Canada [2]School of Electrical and Electronic Engineering, Yonsei University, Seoul, Republic of Korea [3]ISTD Pillar, Singapore University of Technology and Design, Singapore [4]School of Electrical and Mechanical Engineering, University of Adelaide, Australia. Correspondence to: Seong-Lyun Kim <slkim@yonsei.ac.kr>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

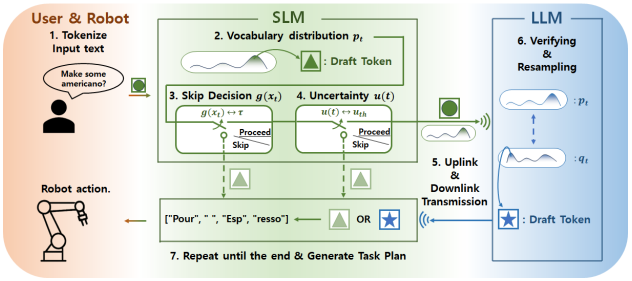


Figure 1. Overall architecture uncertainty-aware opportunistic hybrid language model (U-HLM) consists of local SLM and remote LLM. For every generated token, SLM estimates uncertainty; if the uncertainty exceeds a predefined threshold, uplink transmission to the server occurs for LLM to verify the draft token and resample it if rejected.

1. Introduction

Recent advancements in large language models (LLMs) have led to systems that can provide high-level task planning for performing long-horizon robot tasks (Firoozi et al., 2025). However, task planning frameworks with LLMs still confront key limitations such as computational constraints and inference time, even though onboard processing units have limited resources and real-time capability is an essential requirement for any robotic system interacting with the environment (Ahn et al., 2022; Yang et al., 2024; Bommasani et al., 2021; Bender et al., 2021). Meanwhile, the Uncertainty-Aware Opportunistic Hybrid Language Model (U-HLM) (Oh et al., 2025) has been proposed as a practical framework that not only reduces the computational overhead of LLMs by leveraging both an on-device small language model (SLM) and a remote LLM, but also improves token throughput—by enhancing overall communication and computation efficiency. As shown in Figure 1, U-HLM leverages uncertainty—the model's self-assessed confidence in its outputs—to decide whether uplink communication is necessary, enabling the system to skip transmitting the full vocabulary distribution and avoid remote LLM computation for verification and resampling when uncertainty is low. These characteristics make U-HLM a feasible way to improve latency and reduce computational load in LLM-integrated robotic systems.

While promising, prior evaluations of U-HLM have relied on simulated networks and ignored the cost of on-device uncertainty estimation, which may impact feasibility in real-world deployments. Specifically:

(i) Simulated evaluations do not reflect real wireless channel variability, making it unclear to assess U-HLM's robustness under unstable or degraded network conditions.

(ii) Computing uncertainty via temperature perturbation (Gao et al., 2024) adds non-trivial latency on resource-constrained devices.

To this end, we implement a wireless robotic testbed that consists of a local device, a remote server, and a robot in which U-HLM is deployed for task sequence generation over a wireless network to address (i) and verify its effectiveness as a task planner for robotic systems, while further reducing computational overhead by conditionally omitting uncertainty calculation, resolving (ii).

Our scenario for U-HLM on the wireless robotic testbed is to generate task plans for beverage preparations in a café-style environment. For example, when given a natural-language request "Prepare a medium caramel macchiato with an extra shot of espresso," U-HLM decomposes this into a sequence of atomic actions such as "pour espresso," "add ice," and "drizzle caramel syrup." This setting exposes the model to variability in user orders and real-world constraints, enabling an extensive evaluation of both planning accuracy and execution latency.

In this work, we make the following contributions:

- We implement the wireless robotic testbed in which U-HLM is deployed as a task planner, consisting of a local device, a remote server, and a robot—all connected via the wireless network.
- From the wireless robotic testbed, we identify computational overhead from uncertainty calculation with temperature perturbation and propose the conditional uncertainty omission strategy.
- We empirically evaluate the performance of U-HLM with conditional uncertainty calculation omission across varying task complexities and Wi-Fi conditions, measuring improvements in throughput, latency, and task success.

2. System Model

As proposed in (Oh et al., 2025), U-HLM combines locally deployed SLM with a remote server-based LLM, where *tokens* are their basic units drawn from a shared vocabulary $\mathcal{V}$. For every token generation step, the SLM's uncertainty is monitored, and uplink transmission is invoked only when necessary.

2.1. Local SLM's Token Generation and Uncertainty Calculation

At step t to generate response token $r(t)$ given a input token sequence $\mathbf{s}(t-1)$, the SLM computes logits $\mathbf{z}(t) = \big[z_1(t),\, z_2(t),\, \ldots,\, z_{|\mathcal{V}|}(t)\big]^{\mathsf{T}}$ and derives a vocabulary distribution $\mathbf{p}(t) = \big[p_1(t),\, p_2(t),\, \ldots,\, p_{|\mathcal{V}|}(t)\big]^{\mathsf{T}}$ where it samples *draft token* d from. In parallel, with the use of the logits $\mathbf{z}(t)$, N different vocabulary distributions with temperatures $T_1,\ T_2,\ \ldots,\ T_N$ are computed where the v-th element of n-th distribution is given as follows:

$$\tilde{p}_{v,n}(t) = \frac{\exp\big(z_v(t)/T_n\big)}{\sum_{i=1}^{\mathcal{V}} \exp\big(z_i(t)/T_n\big)}, \quad \forall v \in \mathcal{V}. \tag{1}$$

From the vocabulary distributions, N different *sample tokens*, denoted by $d_1,\ d_2,\ \ldots,\ d_N$, are obtained. Consequently, the uncertainty for the draft token d is calculated as:

$$u(t) \;=\; \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}\big(d_n \neq d\big) \tag{2}$$

where $\mathbb{I}\big(d_n \neq d\big)$ returns 1 if $d_n \neq d$ and 0 otherwise. If the uncertainty $u(t)$ does not exceed a predefined threshold u_{th}, the draft token is immediately accepted as the response token $r(t)$. Otherwise, uplink transmission to send the draft token and the vocabulary distribution to the server occurs.

2.2. Remote LLM's Draft Token Verification and Resampling

Once the draft token and the vocabulary distribution are transmitted to the server, LLM generates its own vocabulary distribution $\mathbf{q}(t) = \big[q_1(t),\ q_2(t),\ \ldots,\ q_{|\mathcal{V}|}(t)\big]^{\top}$. The draft token's probability by the SLM and the LLM is each referred to as $p_d(t)$ and $q_d(t)$. If $p_d(t) \leq q_d(t)$, the draft token is accepted as the t-th response token $r(t)$; otherwise, the draft token is rejected with a probability of $1 - \frac{p_d(t)}{q_d(t)}$, in which case a target token d^* is sampled from an adjusted distribution, where the v-th probability is given as follows:

$$P_v(t) = \frac{\max\big(q_v(t) - p_v(t),\, 0\big)}{\sum_{i=1}^{\mathcal{V}} \max\big(q_i(t) - p_i(t),\, 0\big)}, \quad \forall\, v \in \mathcal{V}. \tag{3}$$

The target token d^* is transmitted back to the local device and accepted as the t-th draft token $r(t)$ to be concatenated

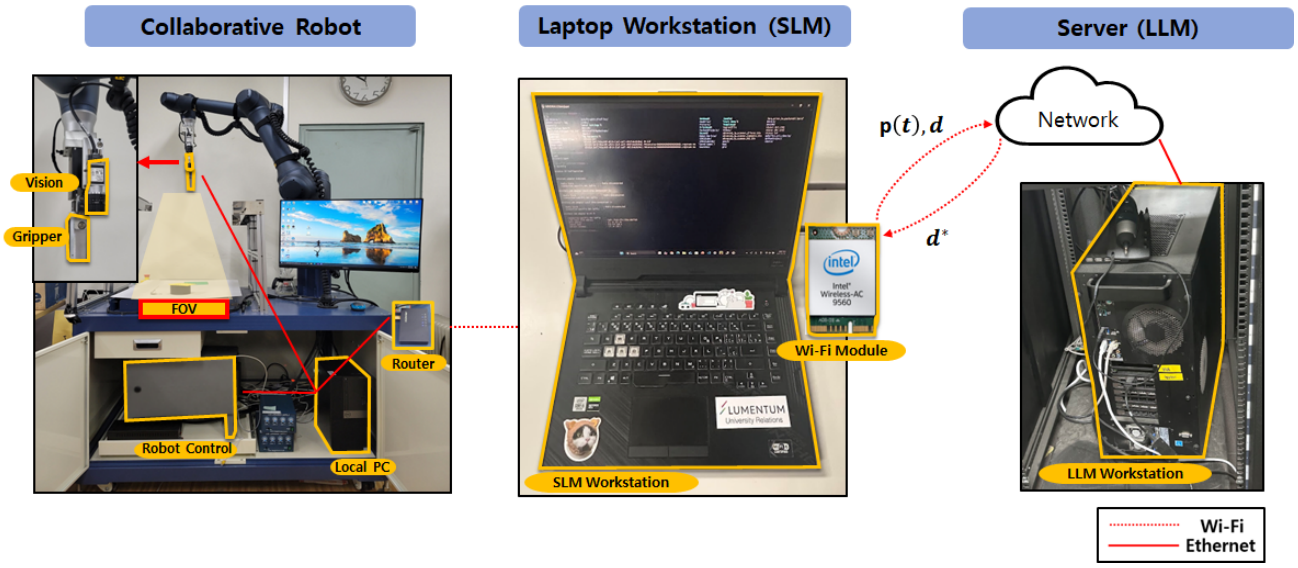


Figure 2. Testbed implementation of U-HLM, consisting of laptop, remote server, and robotic manipulation.

into the token sequence, completing a single round of token generation. This repeats until one of two stop conditions is met: either the End-of-Sequence token is selected or the maximum sequence length is reached.

Although this approach is an optimal solution in theory and in a simulated environment, its effectiveness must be empirically validated on an operational wireless robotic testbed. Thus, in the following sections, we implement a testbed to not only deploy U-HLM on an actual wireless network but also to verify its ability to reliably generate robotic task plans, and compare U-HLM as a task planner against established baselines.

3. Wireless Robotic System

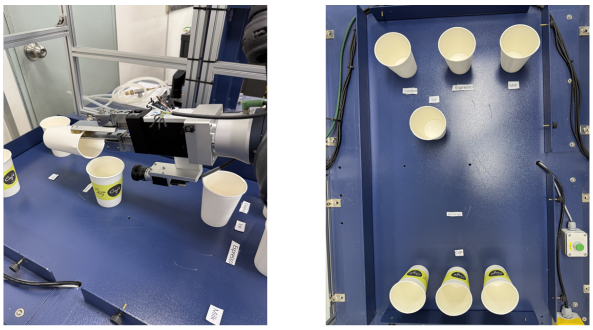
3.1. Testbed Implementation for U-HLM

In this proof-of-concept study to verify U-HLM's effectiveness on an actual wireless network and as a robotic task planner, we implement a testbed consisting of the following three main components: a laptop (local device), a remote server, a the robot, connected over a wireless network, as shown in Figure 2. U-HLM deployed on the testbed serves as a task planner, generating sequences of action-object pairs corresponding to given natural language orders to be performed by the robot.

Experimental Setup. The local SLM, Tiny-Llama 1.1B (Zhang et al., 2024), executes on a Windows-based laptop, serving as the local device, equipped with a 6-core Intel Core i7-10750H CPU, 8 GB of DDR4 RAM, and an NVIDIA GeForce GTX 1650 Ti GPU connected to IEEE 802.11ac Wi-Fi on a 5 GHz band, whereas the remote LLM, Llama 2 7B (Touvron et al., 2023), runs on a Linux server featuring an 8-core Intel Xeon Silver 4215R CPU, 64 GB of DDR4 RAM, and three NVIDIA GeForce RTX 3090 GPUs, connected to Ethernet. A Doosan A0912s robot arm equipped with a GEP2016IO-00-A gripper interfaces with the same Wi-Fi network directly with the laptop. Both the SLM and the LLM are finetuned with LoRA (Hu et al., 2022) to produce structured and executable task sequences consisting of action-object pairs for a given natural language order.

During inference, each draft token generated by SLM is paired with its vocabulary distribution, serialized as a JSON object, and sent from the laptop to the LLM server via a Flask REST endpoint over the Wi-Fi link. The network switch forwards this JSON payload over Ethernet to the server where the verification and probabilistic resampling of the draft token occur before returning the finalized token—again as a JSON object via Flask—over Ethernet and Wi-Fi back to the laptop. Once all tokens are generated to form a complete task plan, the laptop sends it in a single JSON batch again via Flask over Wi-Fi to the robot.

In our task scenario, U-HLM receives a natural language request for a beverage preparation task such as "Can I have caramel macchiato?" and decomposes it into a structured, numbered set of action-object pairs: "1. Place cup 2. Drizzle caramel syrup 3. Pour milk 4. Pour espresso 5. Garnish caramel 6. Serve beverage 7. Done." The robot processes each numbered instruction sequentially: for every step (e.g. "Drizzle caramel syrup"), the on-board controller parses the first word as the action ("Drizzle") and uses the remaining phrase ("caramel syrup") to identify the target object, which are ingredients targeted in cups in our task scenario. Upon encountering the "Done" instruction, the controller recog-



(a) robot pouring ingredient to a cup (b) experimental workspace

Figure 3. (a) An A0912s manipulator executing an ingredient-pouring action into a selected cup under the guidance of the task planner. (b) Top-down view of the experimental workspace, showing six cups positioned at predefined, labeled target locations.

Table 1. Summary of per-token timing measurements

Stage	Min (ms)	Mean (ms)	Max (ms)
Logit generation	53.156	95.040	151.210
Draft token sampling	0.0	0.871	2.003
Uncertainty calculation	5.995	8.066	16.025

nizes it as a termination command and signals successful completion of the task sequence.

As shown in Figure 3a, cups representing the ingredients are located on the experimental workspace with distinct labels denoting the corresponding ingredients. To manipulate them, an onboard RGB-D sensor captures the top-down view of the workspace as shown in Figure 3b, and the locations of the cups are identified by the onboard perception module, while each detected cup is associated with its corresponding label. Upon receiving a target ingredient from the task planner, a cup with the corresponding label is selected, and a predefined action, specified by the task planner and parameterized by the cup's pose and location, is executed by the Doosan A0912s robot arm.

3.2. Latency and Computational Overhead during Temperature Perturbation

During experiments over the testbed discussed in detail in Section 3.1, we observed that computing uncertainty through temperature perturbation in parallel significantly increases memory-bandwidth usage. To isolate each perturbation's effect, we compute uncertainty with temperature perturbation sequentially and break down the local device's per-token latency into the following three components: forward propagation for logit generation, draft token sampling, and uncertainty computation.

While logit generation accounts for most per-token processing time as shown in Table 1—averaging 95.04 ms, uncertainty computation through full temperature perturbation contributes an additional 8.066 ms on average, peaking at 16.025 ms. Given that each temperature perturbation incurs significant computational latency, approximately 10 % of the total, the corresponding computational cost can be reduced if it is possible to omit the temperature perturbation for uncertainty calculation.

4. Conditional Uncertainty Calculation Omission with Probabilistic Classifier

As mentioned earlier, uncertainty computation with temperature perturbation incurs non-negligible computational overhead. A way to approach this is to evaluate the probability that a draft token exceeds the predefined uncertainty threshold and to evaluate the risk associated with the decision to omit full uncertainty calculation based on the computed probability.

4.1. Probability Estimation with Logistic Regression

To estimate the likelihood of the draft token d exceeding the predefined uncertainty threshold u_{th}, we train the logistic regression model (Cox, 1958) by extracting the tokens from the previously generated sequences.

For every draft token d from the previously generated token sequences to train the logistic regression model, we construct a fixed-length feature vector as follows:

$$\begin{aligned}\mathbf{x}_t = [\, & u(t-1), \ldots, u(t-k), \\ & p(t-1), \ldots, p(t-k), d].\end{aligned} \tag{4}$$

where $u(t-1), \ldots, u(t-k)$ and $p(t-1), \ldots, p(t-k)$, denote uncertainty estimates and their probabilities of k tokens preceding d, respectively. At the same time, we label each feature vector $\mathbf{x}_t$ as:

$$y_t = \begin{cases} 1, & \text{if } u(d) > u_{th}; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

where $u(d)$ denotes the uncertainty of d. We then train a logistic regression model $g(\mathbf{x}_t)$ with the cross-entropy loss defined as:

$$\min_{w,b} \sum_{t=1}^{n} \Big[-y_t \ln g(\mathbf{x}_t) - (1-y_t) \ln\big(1-g(\mathbf{x}_t)\big) \Big] + \frac{1}{2}\|w\|_2^2 \tag{6}$$

where $g(\mathbf{x}_t) = \sigma(w^\top \mathbf{x}_t + b)$ is the predicted probability, m is the number of input-label pairs, $\mathbf{x}_t \in \mathbb{R}^{2k+1}$ is the feature vector at d, $y_t \in \{0, 1\}$ is the true label, and $w \in \mathbb{R}^{2k+1}$ and $b \in \mathbb{R}$ are the weight vector and bias.

4.2. Threshold for Uncertainty Calculation Omission Decision

To construct the probabilistic classifier with the trained logistic regression model, we define the rule that uses the probability the regression model outputs to decide whether to omit the uncertainty calculation or not:

$$\tau = \frac{C_u}{C_e}, \quad \text{omit calculation if } g(\mathbf{x}_t) < \tau. \tag{7}$$

This rule is derived from Chow's rejection threshold of minimum-risk rule (Chow, 1970) as the rejection threshold τ is given by

$$\tau = \frac{W_r - W_c}{W_e - W_c} \tag{8}$$

where W_e is the cost incurred when the system makes a recognition error, W_r is the cost of issuing a reject decision, and W_c is the cost of a correct recognition. Mapping our decision into this tri-cost framework , we observe that $W_c = 0$ as there is no realistic cost or delay for classifier operation as its computational cost is negligible compared to logit generation and uncertainty calculation, and W_r, which we denote to be C_u, is the time of calculating uncertainty via temperature perturbation. Lastly, W_e, which we denote C_e, is the average latency encompassing both the initial generation of the token without uncertainty perturbation with erroneously omission of temperature perturbation and the re-generation of the token with proper uncertainty calculation:

$$\tau = \frac{W_r - W_c}{W_e - W_c} = \frac{C_u - 0}{C_e - 0} = \frac{C_u}{C_e}. \tag{9}$$

4.3. Integration to the U-HLM Framework

For every token generation step t, the trained logistic regression model outputs the probability that the uncertainty of the draft token d exceeds u_{th} right before it continues to the uncertainty calculation with the temperature perturbation. If $g(\mathbf{x}_t) < \tau$ for d, the uncertainty calculation with temperature perturbation is omitted, and all subsequent operations are skipped, accepting draft token d as the response token $r(t)$ of the round. Otherwise, it proceeds to regular operation, calculating the actual uncertainty and opportunistically verifying and resampling d with the LLM.

5. Numerical Evaluations

This section demonstrates the effectiveness of U-HLM as a task planner for wireless robotic systems. First, we examine the effectiveness of conditional uncertainty calculation omission and then calibrate the uncertainty threshold to maximize the overall utility. Consequently, we compare the accuracy and the latency of U-HLM deployed on the testbed implementation discussed in Section 3.1 under varying tasks and network conditions against established baselines.

Table 2. Conditional Temperature Perturbation Omit Performance

Metric	Count	Rate (%)
Total tokens evaluated	2500	-
Uncertainty calculation omissions	227	9.1
Wrong omission decisions	20	0.8

5.1. Conditional Uncertainty Calculation Omission for Latency Reduction

As mentioned in Section 4.1, the probabilistic classifier $g(\mathbf{x}_t)$ computes the probability that the next token's uncertainty exceeds the calibration threshold $u_{th} = 0.15$. $w \in \mathbb{R}^{21}$ is the learned weight vector, b is the scalar bias term, and $\mathbf{x}_t \in \mathbb{R}^{21}$ is the input that concatenates a window of the $k = 10$ preceding tokens' uncertainty and probability values with the current token embedding. At each token generation step, the output $g(\mathbf{x}_t) \in [0, 1]$ is compared against $\tau = \frac{C_u}{C_e} \approx 0.228$, and if it does not exceed τ, the uncertainty calculation with the temperature perturbation is omitted, with following actions skipped and the draft token being accepted as the response token.

In this subsection, we numerically examine the performance of the conditional uncertainty calculation omission. For every draft token generated, the classifier assesses the feature vector $\mathbf{x}_t$, and the full uncertainty calculation is executed regardless of the classifier's output. If the classifier decides to omit the full uncertainty calculation and the measured uncertainty does not exceed the uncertainty threshold, it is recorded as a correct omission, while it is recorded as a wrong omission otherwise.

As shown in Table 2, the classifier chooses to omit the uncertainty computation with temperature perturbation for 227 or 9.1% of tokens. Of these, 207 are correct and 20 are wrong, yielding a false-skip rate of 8.8% among skip decisions. These results demonstrate that the proposed strategy can eliminate nearly 10% of full uncertainty evaluations, which is substantial considering that uncertainty calculation with temperature perturbation takes on average 8.066 ms, as mentioned earlier and in Table 1, while keeping the risk of erroneous skips below 1% of total token generations, indicating that performance degradation from skipping is negligible.

5.2. Threshold Selection via Utility Maximization

Here, we derive the uncertainty threshold value that balances the accuracy and the token throughput. To quantify

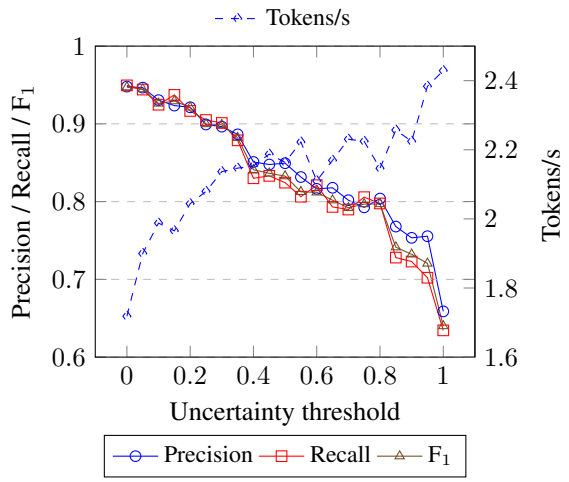


Figure 4. Threshold sensitivity: Precision, Recall, and F_1 (left axis) vs. token throughput (right axis).

the accuracy of the generated plans, we first parse both the model's output and ground truth recipe into lists of the action-object pairs. We count the number of predicted pairs exactly matching the ground-truth pairs (true positives, TP), the number of predicted pairs having no counterpart (false positives, FP), and the number of pairs in the ground truth recipe the model failed to generate (false negatives, FN). Then, based on the counted numbers of TP, FP, and FN, precision, which measures the correctness of positive predictions, recall, which measures the completeness of finding all positives, and the F1 score of each generated task plan are calculated (Rijsbergen, 1979):

$$\text{precision} = \frac{TP}{TP + FP}, \tag{10}$$

$$\text{recall} = \frac{TP}{TP + FN}, \tag{11}$$

$$F_1 = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{12}$$

To evaluate the performances of the model at a given uncertainty threshold, uncertainty threshold values ranging from 0 to 1 in increments of 0.05 are tested with 100 sample orders. As shown in Figure 4, increasing u_{th} gradually decreases the performance measured in precision, recall, and F1.

In task planning for robotic systems, any misclassification of action–object pairs can lead to failed executions or physical hazards. We capture this trade-off by constructing the following utility function:

$$U(u_{\text{th}}) = \alpha_P\, P(u_{\text{th}}) + \alpha_R\, R(u_{\text{th}}) + \alpha_F\, F(u_{\text{th}}) - \lambda\left[1 - \frac{K(u_{\text{th}})}{K_{\max}}\right] \tag{13}$$

Table 3. Utility $U(u_{th})$ computed across uncertainty thresholds for robotic task planning under reliability-driven weighting

u_{th}	$P(u_{th})$	$R(u_{th})$	$F_1(u_{th})$	$K(u_{th})$	$U(u_{th})$
0.00	0.9479	0.9495	0.9472	1.7184	0.8748
0.05	0.9466	0.9441	0.9448	1.9000	0.8906
0.10	0.9305	0.9246	0.9263	1.9878	0.8815
0.15	**0.9235**	**0.9372**	**0.9293**	**1.9653**	**0.8821**
0.20	0.9213	0.9166	0.9189	2.0439	0.8793
0.25	0.8992	0.9049	0.8998	2.0796	0.8650
0.30	0.8964	0.9011	0.8988	2.1367	0.8687
0.35	0.8865	0.8788	0.8821	2.1469	0.8533
0.40	0.8510	0.8300	0.8400	2.1531	0.8119
0.45	0.8477	0.8331	0.8364	2.1874	0.8135
0.50	0.8494	0.8244	0.8328	2.1638	0.8076
0.55	0.8316	0.8063	0.8126	2.2224	0.7945
0.60	0.8164	0.8211	0.8120	2.1102	0.7826
0.65	0.8178	0.7932	0.8018	2.1694	0.7769
0.70	0.8018	0.7898	0.7920	2.2306	0.7735
0.75	0.7926	0.8057	0.7968	2.2245	0.7769
0.80	0.8039	0.7976	0.7969	2.1464	0.7697
0.85	0.7679	0.7280	0.7414	2.2571	0.7270
0.90	0.7534	0.7229	0.7322	2.2245	0.7141
0.95	0.7555	0.7021	0.7205	2.3837	0.7200
1.00	0.6588	0.6344	0.6400	2.4290	0.6433

where P, R, F are precision, recall, and F1 at uncertainty threshold u_{th}; $K(u_{th})$ is the token throughput; K_{max} its maximum across u_{th} values, which is 2.429 tokens/sec) at $u_{th} = 1$; $\alpha_{P,R,F}$ and λ weight the contributions of accuracy metrics and latency.

For a robotic system—where consistent, reliable execution is essential while throughput reduction is desirable—we choose the the following values:

$$\alpha_F = 0.50, \quad \alpha_P = 0.25, \quad \alpha_R = 0.25, \quad \lambda = 0.25.$$

thereby placing greatest emphasis on F1 while still valuing precision and recall, and limiting the throughput penalty to at most 25% of the normalized loss. Using these values, we evaluate $U(u_{th})$ across candidate thresholds and obtain.

As Table 3 shows, F1 reaches their global maxima at $u_{th} = 0.15$, while throughput K remains 80.9% of K_{max}. Most importantly, $U(u_{th})$ is maximized at $u_{th} = 0.15$. This confirms 0.15 as the optimal uncertainty threshold value, delivering peak reliability in robotic task planning with only a negligible throughput trade-off.

Table 4. Wi-Fi performance under strong vs. weak coverage

Metric	Strong Coverage	Weak Coverage
Average RTT (ms)	12.0540	17.5905
RTT jitter (ms)	0.3020	3.5568
Throughput (Mbit/s)	55.4	7.58

Table 5. Model Accuracy (Precision, Recall, F1)

Model	Precision	Recall	F1-score
In-Distribution			
U-HLM	0.9235	0.9372	0.9293
HLM	0.9479	0.9495	0.9472
Rand-HLM	0.7152	0.7028	0.7030
Out-of-Distribution			
U-HLM	0.8320	0.8335	0.8297
HLM	0.8303	0.8351	0.8316
Rand-HLM	0.7196	0.6968	0.7031

5.3. Comparative Performance and Latency Breakdown of Baselines under Varying Network Conditions

In this section, we compare the established baselines by measuring the average per-token latency and the accuracy under varying network conditions and two types of tasks—in-distribution and out-of-distribution. In-distribution tasks include natural language orders for requesting beverages the model has seen during the finetuning of the SLM and the LLM, while out-of-distribution tasks contain requests for beverages that are not included in the training datasets. In terms of network conditions, we characterized the Wi-Fi link between the local device and the remote verification server in two conditions—strong coverage and weak coverage—to account for network fluctuations. For each condition, we derive the round trip time (RTT), the RTT jitter, and the throughput. Table 4 summarizes the key wireless performance indicators. At the same time, we define the per-token latency as follows:

$$\overline{T}_{\text{total}} = \overline{T}_{\text{SLM}} + \overline{T}_{\text{LLM}} + \overline{T}_{\text{comm}}, \tag{14}$$

where $\overline{T}_{\text{SLM}}$ denotes the mean on-device SLM compute time including token drafting and uncertainty estimation, $\overline{T}_{\text{LLM}}$ denotes the mean remote LLM compute time, and $\overline{T}_{\text{comm}}$ denotes the mean round-trip communication latency. The baseline methods include SLM, HLM (Hao et al., 2024), and Rand-HLM, which randomly transmits draft tokens for LLM verifications. In Rand-HLM, the transmission rate is fixed at 0.20 to match the average transmission rate of U-HLM, and the uplink transmission for LLM verification and sampling occurs randomly. Each baseline generates 100 task sequences each.

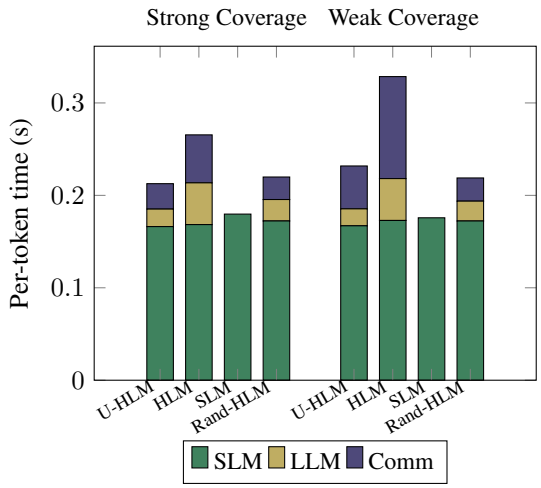


Figure 5. Per-token latency breakdown for four methods under the strong coverage and the weak coverage.

As Figure 5 shows, under the strong coverage, U-HLM requires on average 0.213 s per token—roughly 20 % less than HLM's 0.266 s and only marginally over Rand-HLM's 0.220 s. Under the weak coverage, on the other hand, U-HLM maintains 0.232 s of per token latency, which is 29 % faster than HLM's 0.329 s and still comparable to Rand-HLM's 0.219 s. Transitioning from the strong to the weak coverage, per-token latencies of HLM and U-HLM have been increased by 23.7% and 8.9% respectively, while that of Rand-HLM remains essentially flat. As such, by opportunistically skipping uplink transmission and conditionally omitting uncertainty calculation, U-HLM remains far less sensitive to network degradation compared to HLM while remaining comparable to Rand-HLM.

From an accuracy standpoint as presented in Table 5, U-HLM achieves F1 of 0.9293 for in-distribution tasks—marginally lower than HLM's 0.9472 while significantly outperforming Rand-HLM's 0.7030. For out-of-distribution tasks, U-HLM's F1 only falls to 0.8297, trailing HLM's 0.8316 and still outperforming Rand-HLM's 0.7031.

Thus, as summarized by Figure 6, U-HLM effectively balances accuracy and latency, unlike HLM, which sacrifices throughput for marginal gains in F1, and SLM and Rand-HLM, which achieve high token rates only by compromising accuracy. This balance between accuracy and latency and the reliability in varying network conditions ensure that U-HLM meets the strict timing and accuracy required by the wireless robotic systems to serve as a task planner.

5.4. Robotic System Integration and Deployment on Wireless Network

In this section, we evaluate U-HLM as a task planner by determining how reliably it outputs task plans that can be

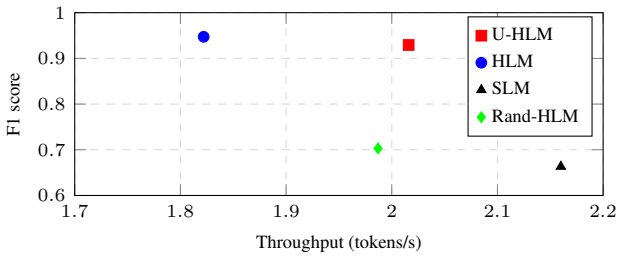


Figure 6. F1 vs. Token Throughput.

Table 6. Comparison of U-HLM, SLM, Rand-HLM and HLM on Planning Success Rate (PSR), Inference latency (in seconds), True Skip Rate (TSR) and Transmission Rate (TR).

Method	Difficulty	PSR	Latency	TSR	TR
U-HLM	Easy	0.80	17.3461	0.9480	0.0967
	Medium	0.65	26.0011	0.9427	0.2036
	Hard	0.16	17.3494	0.9311	0.2499
Rand-HLM	Easy	0.14	17.8705	0.9930	0.2036
	Medium	0.13	26.5631	0.9891	0.2046
	Hard	0.10	16.7155	0.9915	0.2007
HLM	Easy	0.81	18.1062	0.9629	0.2916
	Medium	0.70	28.8667	0.9320	0.4972
	Hard	0.16	21.8640	0.9346	0.5335

correctly executed by the wireless robotic system under varying difficulties of the tasks, as implemented in Section 3.1.

Similar to the previous experiments, we compare the proposed architecture for the following methods: U-HLM, Rand-HLM, and HLM. There are four evaluation metrics to be considered. The first is the planning success rate (PSR), calculated as follows:

$$\text{PSR} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\big(\hat{\pi}_i = \pi_i^*\big) \tag{15}$$

where N is the total number of plans generated, $\hat{\pi}_i$ is the plan produced by the U-HLM for task i, and π_i^* is the corresponding ground-truth plan for task i. The second is the inference latency, which is defined as the total time taken to generate a plan $\hat{\pi}_i$. Additionally, true skip ratio (TSR), the probability of skipping an uplink transmission, and transmission rate (TR), the proportion of tokens that undergo an uplink transmission, are considered.

We categorize tasks into three difficulties: easy, medium, and hard. The easy tasks involve known beverage requests to measure in-distribution performance, the medium tasks involve known beverages with simple modifications to assess adaptability, and the hard tasks consist of requests for beverages unseen during training.

Table 6 shows that U-HLM significantly cuts uplink traffic, preserving low latency across all difficulties. Compared to HLM, U-HLM reduces the transmission rate by 66% from 0.2916 to 0.0967 and lowers inference latency from 18.11 seconds to 17.35 seconds for the easy tasks. On the medium tasks, U-HLM halves the transmission rate from 0.4972 to 0.2036 and speeds up planning by approximately 10%, cutting the latency from 28.87 seconds to 26.00 seconds. A similar trend is shown in the hard tasks as the transmission rate is reduced from 0.4967 to 0.2499, while decreasing the latency from 21.86 seconds to 17.35 seconds. These communication and latency savings result in only a modest loss in accuracy. U-HLM's planning success rate for the medium tasks falls from 0.70 under HLM to 0.65, while it remains unchanged for easy and hard tasks.

By contrast, Rand-HLM achieves a low planning success rate below 0.15 across all difficulties. Those poor planning success rates stem directly from the strict requirement that the generated plan must match the ground-truth plan exactly. This experiment again demonstrates U-HLM's reliability in generating accurate task plans while lowering latency against baselines with comparable accuracy.

6. Conclusions

In this paper, we proposed conditional skipping of uncertainty calculation in U-HLM to further reduce computational overhead, verified its relative performance against baseline methods on a real wireless network, and showcased a practical implementation in robotics. Experimental results confirmed that our implemented U-HLM can significantly reduce both end-to-end latency and uplink traffic with a negligible drop in success rate, demonstrating its real-time viability on robotic platforms and motivating future work on further reducing communication overhead in interference-prone channel environments and expanding to broader task scenarios.

Impact Statement

This paper presents work to advance the field of resource-aware, language-driven robotic planning by demonstrating U-HLM under realistic wireless conditions. By combining U-HLM with a robotic manipulator, it can streamline automation in manufacturing and service robotics. However, incorrect or maliciously altered plans could damage equipment or harm bystanders; to mitigate this, we require rigorous testing before any real-world deployment. We also recognize that large-scale adoption could displace some manual-labor roles even as it creates new opportunities.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., et al. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pp. 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., et al. On the opportunities and risks of foundation models. *ArXiv*, 2021. URL https://crfm.stanford.edu/assets/report.pdf.

Chow, C. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970. doi: 10.1109/TIT.1970.1054406.

Cox, D. R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958. ISSN 00359246. URL http://www.jstor.org/stable/2983890.

Firoozi, R., Tucker, J., Tian, S., Majumdar, A., Sun, J., Liu, W., et al. Foundation models in robotics: Applications, challenges, and the future. *International Journal of Robotics Research*, 44(5):701–739, April 2025. ISSN 0278-3649. doi: 10.1177/02783649241281508. Publisher Copyright: © The Author(s) 2024.

Gao, X., Zhang, J., Mouatadid, L., and Das, K. SPUQ: Perturbation-based uncertainty quantification for large language models. In Graham, Y. and Purver, M. (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2336–2346, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.143/.

Hao, Z., Jiang, H., Jiang, S., Ren, J., and Cao, T. Hybrid slm and llm for edge-cloud collaborative inference. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, EdgeFM '24, pp. 36–41, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706639. doi: 10.1145/3662006.3662067. URL https://doi.org/10.1145/3662006.3662067.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Oh, S., Kim, J., Park, J., Ko, S.-W., Quek, T. Q. S., and Kim, S.-L. Uncertainty-aware hybrid inference with on-device small and remote large language models, 2025. URL https://arxiv.org/abs/2412.12687.

Rijsbergen, C. J. V. *Information Retrieval*. Butterworth-Heinemann, USA, 2nd edition, 1979. ISBN 0408709294.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Yang, Y., Jiao, L., and Xu, Y. A queueing theoretic perspective on low-latency llm inference with variable token length. In *2024 22nd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 273–280, 2024.

Zhang, P., Zeng, G., Wang, T., and Lu, W. Tinyllama: An open-source small language model, 2024. URL https://arxiv.org/abs/2401.02385.