CIFLEX: Contextual Instruction Flow for Sub-task Execution in Multi-Turn Interactions with a Single On-Device LLM

Anonymous ACL submission

Abstract

We present **CIFLEX** (**C**ontextual **I**nstruction **FL**ow with **EX**ecution), a novel execution system for efficient sub-task handling in multi-turn interactions with a single on-device large language model (LLM). As LLMs become increasingly capable, a single model is expected to handle diverse sub-tasks that more effectively and comprehensively support answering user requests. Naive approach reprocesses the entire conversation context when switching between main and sub-tasks (e.g., query rewriting, summarization), incurring significant computational overhead. **CIFLEX** mitigates this overhead by reusing the key-value (KV) cache from the main task and injecting only task-specific instructions into isolated side paths. After sub-task execution, the model rolls back to the main path via cached context, thereby avoiding redundant prefill computation. To support sub-task selection, we also develop a hierarchical classification strategy tailored for small-scale models, decomposing multi-choice decisions into binary ones. Experiments show that **CIFLEX** significantly reduces computational costs without degrading task performance, enabling scalable and efficient multi-task dialogue on-device.

1 Introduction

As Large language models (LLMs) (Dubey et al., 2024; Achiam et al., 2023; Yang et al., 2024a; Jiang et al., 2023) engage in more multi-turn interactions (Hassan and Graham, 2024; Guan et al., 2025; Laban et al., 2025a), user poses various and complex requests. As such, a variety of sub-tasks, such as query rewriting (), API call or complex reasoning (), are necessary for better supporting user request in the middle of a multi-turn interaction session. Moreover, the LLM should keep aware of the historical context of the multi-turn conversation for correct sub-task executions as well as reliable answer for the main request. However, in edge device, while maintaining the context of conversation in a main model, deploying additional side-task specific models for every sub-task is impractical due to its memory and compute constraints. Encouragingly, LLMs progresses at a remarkable pace, and thus a single LLM is expected to handle a variety of tasks.

Despite the rapid progress of the versatility of a single LLM, it has paid limited attention to the efficiency of the scenarios where sub-task execution and decision-making operate *over long, multi-turn* conversations under an LLM. Explicitly including sub-task execution within the conversation history leads to not only excessively long contexts but also exposes unnecessary internal processing steps to the user. Then, as a naive approach, one could perform sub-task execution by reloading the entire conversation history and inserting new task-specific instructions whenever switching from the main task to a sub-task. However, this incurs substantial computational and memory overhead due to repeated prefill operations, which is particularly problematic for on-device deployment. For example, in a conversation of around 20 turns, the prefill required can be 300 times more tokens than the generation. As the number of turns grows, the cost of managing context and instructions across multiple tasks becomes a dominant bottleneck, especially when such operations are repeated throughout a session.

Decoder-based LLMs (Dubey et al., 2024; Achiam et al., 2023; Yang et al., 2024a; Jiang et al., 2023) can alleviate redundancy during generation via key-value (KV) caching, which stores intermediate token representations to avoid recomputation over previously seen inputs. Recently, several works (Lyu et al., 2025a; Liu et al., 2024a) have attempted to exploit KV cache sharing across differently fine-tuned models to reduce overhead. While promising, these methods have not been explored in the context of long-horizon, multi-turn conversations, where cache reuse between different models becomes more fragile and error-prone. In our own

observations, we found that sharing KV caches between differently fine-tuned models becomes increasingly problematic as the conversation grows longer, due to subtle drifts in behavior and context representation. Given the rapidly improving generalization ability of recent LLMs on diverse tasks, we argue that employing a single LLM is more *robust* and *cache-efficient* when managing instruction flow with keeping the conversation history across main and side tasks.

To this end, we propose **CIFLEX** (**C**ontextual **I**nstruction **FL**ow with **EX**ecution), a novel execution strategy that extends the utility of KV cache from token-level reuse to task-level reuse. Rather than reloading the entire context for every sub-task, **CIFLEX** checkpoints the KV cache of the main conversation flow and reuses it by injecting only task-specific instructions into a side path. Sub-tasks are then executed efficiently on top of the preserved context without redundant processing. After execution, the model seamlessly rolls back to the main path using the cached state, avoiding the computational cost of full prefilled context. Additionally, we propose a hierarchical sub-task classification framework designed for smaller-scale LLMs. Instead of relying on multi-class selection, which is often unreliable for device-scale models—we perform a sequence of binary classifications in side paths, guided by our contextual instruction flow.

Since there have been no works considering sub-task execution in multi-turn conversations, we present long multi-turn conversation datasets (*e.g.,* TopiOCQA-Task+ and QReCC-Task+) by combining various task-specific datasets for conversational search, math problem solving, API calls, and casual chats. Then, through extensive experiments, we show that **CIFLEX** achieves significant computational savings while sacrificing less performance, offering a practical and scalable solution for multi-task dialogue systems in resource-constrained environments.

2 CIFLEX: Contextual Instruction FLow with EXecution

2.1 Problem Formulation

In a multi-turn interaction which is the main task m, an LLM $\mathcal{M}$ (here, we consider edge device-scaled model such as LLaMA3 3.1 8B) generates an answer $a^{(t)}$ for the given user query $q^{(t)}$ at each turn t. This main question and answer flow is represented by

$$a^{(t)} = m(\mathcal{M}, C^{(t)}_{\text{main}}, q^{(t)}), \tag{1}$$

where $C^{(t)}_{\text{main}}$ denotes the conversational context of m at turn t, consisting of the main task instruction I_{main} and all previous turns:

$$C^{(t)}_{\text{main}} = \begin{cases} \{I_{\text{main}}\} & \text{if } t = 1, \\ \{I_{\text{main}},\ \{(q^{(t')}, r^{(t')}, a^{(t')})\}_{t'=1}^{t-1}\} & \text{otherwise,} \end{cases} \tag{2}$$

where $r^{(t)}$ represents turn-specific context such as the external retrieval knowledge, which is optional but can lead to substantial increase $|C^{(t)}_{\text{main}}|$.

To produce correct answer $a^{(t)}$, the LLM can optionally execute a proper auxiliary sub-task which supports the main answer but is not directly exposed to the user. The sub-tasks are designed to support m, and are represented as a set $\{s_1, \ldots, s_N\}$, where s_n denotes the nth sub-task. Each sub-task produces an intermediate output $o^{(t)}_{s_n}$ based on its own context:

$$o^{(t)}_{s_n} = s_n(\mathcal{M}, C^{(t)}_{s_n}, q^{(t)}), \tag{3}$$

where a sub-task-specific instruction I_{s_n} is used in place of I_{main} in $C^{(t)}_{s_n}$.

Also, before executing any sub-task s_n, a classification sub-task μ needs to be performed to determine which sub-task is necessary for the current turn. This classification task is also treated as another sub-task.

Then, in a single-LLM setting, whenever switching from the sub-tasks to the main task, the conversational context should be re-prefilled even though much of the context is redundant across tasks and turns. This repeated context reloading imposes significant computational overhead, which motivates our proposed approach for efficient sub-task execution.

2.2 Methodology

To enable efficient sub-task execution, we propose Contextual Instruction Flow with Execution dubbed **CIFLEX**. Fig. 1 illustrates the overall flow. We also provide the set of prompt templates in Appendix.

Sub-Task Execution & Rollback. Once a user query q_t is issued, we create a *checkpoint* at that moment—preserving the key-value (KV) cache of the entire main path up to q_t. For instance, when

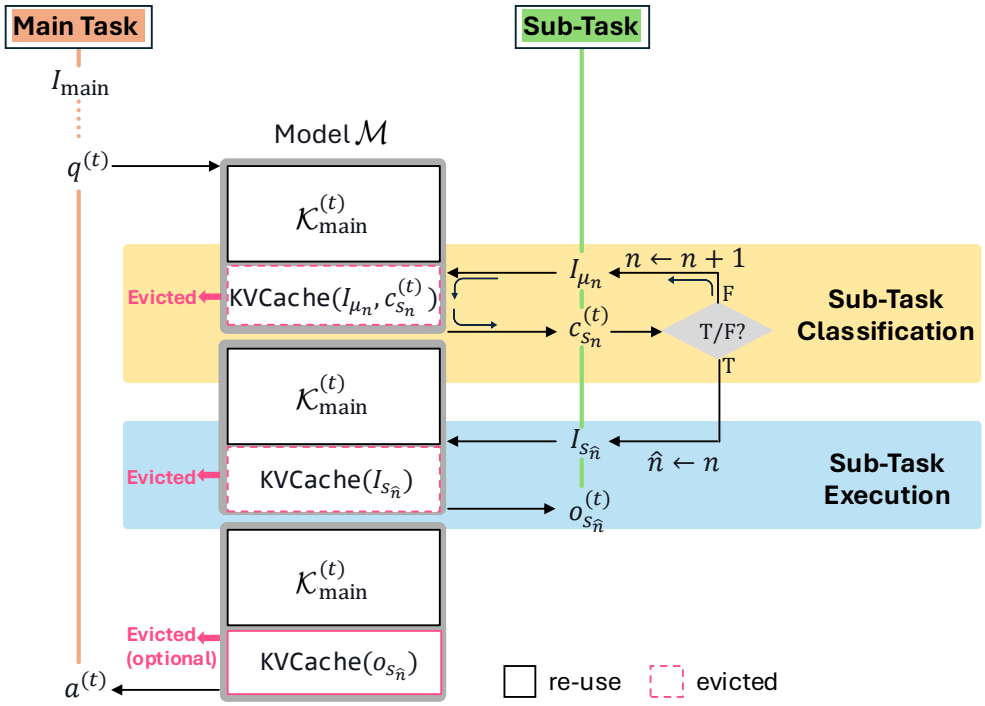


Figure 1: **Overall framework of the proposed CIFLEX.**

$t = 1$

$$\mathcal{K}^{(1)}_{\text{main}} = \text{KVCache}\left(C^{(1)}_{\text{main}}, q^{(1)}\right). \quad (4)$$

Based on this cached state, we branch into the appropriate sub-task path. Instead of reprocessing the full conversation context, we prefill only the sub-task instruction I_{s_n} on top of the preserved KV cache.

$$o^{(1)}_{s_n} = s_n(\mathcal{M}, I_{s_n};\ \mathcal{K}^{(1)}_{\text{main}}) \quad (5)$$

The sub-task s_n is then executed efficiently. Empirically, we observe that although the main instruction I_{main} is implicitly retained via the KV cache, the sub-task reliably adheres to I_{s_n}, without interference from I_{main}.

After completing the sub-task, we evict the KV cache related to the sub-task instruction (e.g., I_{s_n}) or complex reasoning process and, if necessary, retain the cache associated with the sub-task's final output $o^{(1)}_{s_n}$.

$$\mathcal{K}^{(1)}_{\text{rollback}} = \mathcal{K}^{(1)}_{\text{main}} \oplus \text{KVCache}\left(o^{(1)}_{s_n}\right) \quad (6)$$

where $\oplus$ denotes concatenation operation. We can then *roll back* to the main path and the model to continue the main task without re-prefilling the full conversational history.

$$a^{(1)} = m(\mathcal{M}; \mathcal{K}^{(1)}_{\text{rollback}})). \quad (7)$$

Model	LLaMA (8B)	Mistral (7B)	Qwen (7B)	GPT-4 (>100B)
Acc (%)	33.93	44.31	42.13	98.21

Table 1: **Challenge of multi-choice sub-task classification set-up on smaller-scaled models.** Accuracy (%) are reported on dataset extended from TopiocQA.

Here, $r^{(1)}$ can be prefilled, if necessary, i.e. $a^{(1)} = m(\mathcal{M}, r^{(1)}; \mathcal{K}^{(1)}_{\text{rollback}}))$.

Notice that, when $t > 1$, the KV cache is shared across turns as well, then it is simply updated after the model outputs $a^{(t)}$:

$$\mathcal{K}^{(t+1)}_{\text{main}} = \mathcal{K}^{(t)}_{\text{rollback}} \oplus \mathcal{K}^{(t)}_{a} \oplus \text{KVCache}\left(q^{(t+1)}\right) \quad (8)$$

where $\mathcal{K}^{(t)}_{a}$ is the added KV cache during answering in the main task. Therefore, only the KV cache for $q^{(t+1)}$ is computed in prefill.

This structured mechanism is what we refer to as *contextual instruction flow*, where the shared conversational context is reused, and only task-specific instructions are attached or evicted to enable efficient task switching. This contextual instruction flow allows substantially reducing the overall computational cost of switching the main and sub-tasks during multi-turn conversations with a single LLM.

Sub-Task Classification. As described in Section 2.1, the objective of sub-task classification is

to determine which sub-task should be executed. In our work, it is also formulated using the proposed contextual instruction flow strategy.

As shown in Table 1, small-scale language models struggle to comprehend the definitions of multiple sub-tasks simultaneously, failing to make accurate selections under a multi-choice setup. To address this limitation, we propose a hierarchical selection strategy.

Specifically, we first assign priorities to the available sub-tasks using a larger LLM, offline. Then, at runtime, the on-device model performs binary classification per sub-task in descending order of priority, varying only the class-specific instruction in the contextual instruction.

$$c_{s_n}^{(t)} = \mu_n(\mathcal{M}, I_{\mu_n};\ \mathcal{K}_{\text{main}}^{(t)}) \tag{9}$$

Then, if a sub-task with higher priority is classified as necessary, the hierarchical process is terminated, and the selected sub-task $s_{\hat{n}}$ is executed.

It is important to note that these classification decisions are also made using the same single LLM instance. As a result, executing multiple binary classifications can lead to a significant increase in prefilling cost. However, thanks to our contextual instruction flow design, only the task-specific classification instructions need to be prefetched, which minimizes redundant computation and ensures efficient task routing.

3 Multi-turn Dataset Construction

Following (Yi et al., 2024), we consider broadly two kinds of conversations: task-oriented conversation (conversational search, math problem, multi-modal question) and open-domain conversation (casual chat). Then, sub-tasks are mapped as follows:

(1) Conversational Search - Query Rewriting. We employ TopiOCQA (Adlakha et al., 2022) and QReCC (Anantha et al., 2021), both of which consist of multi-turn conversational queries. These datasets contain context-dependent questions involving pronouns, ellipses, and coreference, requiring disambiguation based on prior conversation turns. Hence, in the sub-task, LLM clarifies the original question into a standalone question. Then, an off-the-shelf retriever (e.g., BGE-Large (Xiao et al., 2023)) retrieves evidence passages which is given to the main task.

(2) Math Problem - Reasoning & Answer. To this end, we incorporate the GSM8K dataset (Cobbe et al., 2021), which provides both problem statements and detailed reasoning steps. We randomly sample math problems and concatenate them naturally into existing TopiOCQA and QReCC, respectively. In the side-path execution flow, the LLM is in-context learned, and outputs the reasoning and answer are generated. Then, only the final answer is returned to the main path. This design helps reduce the contextual burden on the main flow while enabling intermediate computation in the sub-task flow.

(3) Multi-Modal - API Call. Here, we exploit the Gorilla dataset (Patil et al., 2023), which contains API specifications and their expected invocation patterns. Based on a randomly selected API, then a larger LLM (e.g., LLaMA3.1 70B (Dubey et al., 2024), GPT4 (Achiam et al., 2023)) generates a multi-modal question like *"What is the color of the whale in the Pacific Ocean [Image]?"*. In sub-task, LLM is instructed to select a proper API call. Then, the textual description for the multi-modal input from an off-the-shelf model is returned to the main task.

(4) On-going Casual Chat - No Sub-Task. In practice, not every user query requires a sub-task. To reflect such cases, we include examples where the user is in the middle of casual conversation with the assistant and no sub-task execution is necessary. Specifically, we append several turns of casual, non-task-oriented turns—covering topics such as weather, hobbies, or daily life—on top of task-oriented interactions from categories (1)–(3). These casual turns are generated using a larger LLM. These cases serve as negative examples for sub-task classification and help balance the task distribution.

(5) Last Casual Chat - Chat Summary. On edge devices, AI assistants are often personalized to the user, making it crucial to understand user preferences and behavior across interactions. To support this, we introduce a summarization sub-task that captures the essence of a casual conversation once it concludes. Namely, in the sub-task, LLM is instructed to summarize the user's behavior or interests based on the conversation. This summary may be saved in an external knowledge hub.

- Commonly, to ensure a coherent multi-turn flow and maintain topic consistency, we include the full conversation history from TopiOCQA or

QReCC within the prompt when generating additional casual chat or API-related turns using large LLMs (the corresponding prompts are included in Appendix). This allows the generated content to remain contextually grounded and avoids unnatural topic shifts across turns. Finally, since we use two different conversational search datasets, the resulting multi-turn, multi-task datasets are dubbed **TopiOCQA-Task+** and **QReCC-Task+**, respectively, which enable us to evaluate our proposed execution framework (**CIFLEX**) in a unified yet diverse conversational setting. We provide more details in Appendix.

4 Experimental Results

4.1 Baselines

To show the effectiveness of our method, we compare the proposed **CIFLEX** with three baselines:

-*Full Re-load*: We assume a single LLM is runnable due to the resource constraints of edge devices. Hence, in this baseline, all the conversation history is re-loaded (pre-filled as textual inputs) whenever main or sub-tasks are switched without considering KV cache reuse across tasks.

-*Recent Re-load*: For less computational burden of re-loading the entire conversation history, only the recent several turns are retained, discarding the earlier turns. Here, the recent five turns are used.

-*Seamless*: To mitigate the burden of the re-loading, we can execute all the tasks in a single flow without task switching. Here, all the sub-task execution instructions and results are included in the main task seamlessly.

-*Chain-of-Model (CoM)* (Lyu et al., 2025a): In this baseline, we train the sub-task-specific LoRA adapters (Hu et al., 2021) starting from the same pre-trained model, and CoM employs learnable prompt tokens to increase adaptability across the differently-trained models in sharing the KV-caches of common parts.

4.2 Evaluation Metrics

Sub-Task Classification. To evaluate the effectiveness of the proposed per-task binary classification, we measure the accuracy of the classification results. Each sub-task is independently predicted in its own path, and overall classification performance is reported as the average accuracy across tasks.

Sub-Task Execution. For the query rewriting sub-task, we use the rewritten query to retrieve passages via an off-the-shelf retriever, BGE-Large (Xiao et al., 2023). Since both TopiOCQA and QReCC provide ground-truth retrieval annotations, we report standard information retrieval metrics: nDCG@3 reflects relevance and order of top-3 results, and Hit@k indicates whether any relevant passage is retrieved within the top-k passages where we set $k = 5$. Since the chat summary and API call have no static, fixed ground truth, we utilize the GPT-eval metric, the coherence (Zhong et al., 2023). This means the side-task output is evaluated if it is aligned logically and contextually with the corresponding side-task instruction. The detailed metric prompt is included in Appendix. For the math problem, the sub-task output is directly used in the main task; therefore, we assess it as part of the main task's answer.

Main-Task Execution. We utilize the GPT-eval metrics, coherence and correctness, based on (Zhong et al., 2023). Correctness metric is used only for the task with the static, fixed ground-truth, i.e. math problem.

4.3 Target models

For the edge-device model, we employ LLaMA3.1-8B-Inst (Dubey et al., 2024), Mistral-v0.2-Inst 7B (Jiang et al., 2023), Qwen2.5-Inst-7B (Yang et al., 2024a) models.

4.4 Results

Sub-Task Classification. As described in Sec. 3, each user query is classified into one of five sub-task categories: query rewriting, math solving (reasoning & answer), API call, chat summary, and no sub-task. Following our per-task binary classification with task priority strategy, we assign a fixed priority order to the five classes based on their class-wise recall rates measured by a larger oracle model. Classes with clearer semantic boundaries receive higher priority, while more ambiguous classes are ranked lower. Specifically, the priority order is: API call $\rightarrow$ Math solving $\rightarrow$ Query rewriting $\rightarrow$ Summarization $\rightarrow$ None. In cases where multiple sub-tasks are simultaneously detected, the one with higher priority is selected (e.g., API call over query rewriting). As shown in Table 2, for all the models and datasets, the proposed approach significantly outperforms the multi-choice strategy, which often struggles to detect the correct sub-task.

Also, in terms of the efficiency, Fig. 2 plots the required total number of pre-fill tokens up to a given turn. Although the per-task classification is effective, it requires an extremely high number of tokens (193.6K for 22 turns) without applying the

Dataset	Method	On-device-scale model			Server-scale model
		LLaMA3.1-Instruct (8B)	Mistral-Instruct-v0.2 (7B)	Qwen2.5-Instruct (8B)	GPT-4 (>100B)
TopiOCQA-Task+	Multi-choice	33.93	44.31	42.13	98.21
	Per-task binary	**88.01**	**83.68**	**84.4**	98.55
QReCC-Task+	Multi-choice	51.83	57.66	59.06	96.52
	Per-task binary	**82.04**	**80.68**	**83.97**	97.31

Table 2: **Sub-task classification results.** Accuracy (%) are reported on multi-choice strategy and the proposed per-task binary classification varying models on TopiOCQA-Task+ and QReCC-Task+. The best results are in bold.

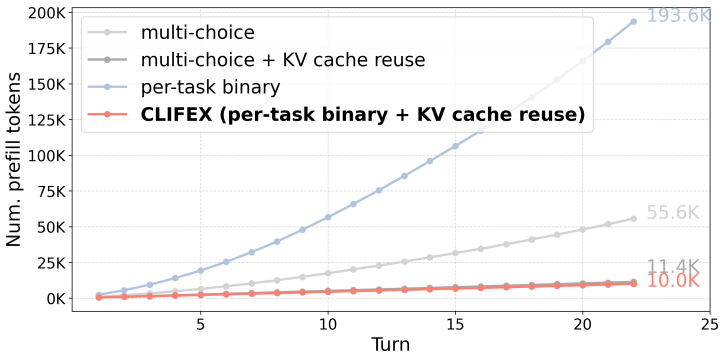


Figure 2: **Total prefilled tokens for sub-task classification until each turn** in LLaMA3.1-Instruct (8B) on TopiOCA-Task+.

proposed **CIFLEX**. With **CIFLEX**, per-task binary (orange) even uses slightly less pre-fill tokens than multi-choice (dark gray), since the instructions of sub-tasks are designed to include shared parts (see the Appendix). Hence, our per-task classification with **CIFLEX** is highly effective and efficient.

Sub & Main-Task Execution. We compare the proposed **CIFLEX** against the four baselines introduced in Sec. 4.1. As shown in Table 3, the Seamless baseline exhibits poor performance across both sub-tasks and the main task. This degradation indicates that accumulated main & sub-task execution history interfere with LLM's contextual understanding ability,highlighting the necessity of independent sub-task execution in separate flows.

The Full Re-load baseline yields reliable performance owing to prefilling clean task-specific instructions, but at the significant cost of context re-loading overhead, as in Fig. 3. Compared to the Full Re-load, the Recent Re-load baseline reduces prefilling cost by reloading only the truncated recent turns; however, it still requires much more prefilling tokens than our **CIFLEX**. Moreover, it suffers from performance degradation. Although the CoM (Lyu et al., 2025a) learns models to increase the adaptability of KV caches across different models, its performance is lower than the single model inference (ours). We can see that cache-sharing across heterogeneous models becomes fragile and error-prone as the number of turns increases.

In contrast, the proposed **CIFLEX** achieves comparable or superior performance to the Full Re-load, while incurring significantly less prefilling cost, as in the orange of Fig. 3. This demonstrates that **CIFLEX** strikes an effective balance in terms of both accuracy and efficiency in multi-turn, multi-task execution under a single LLM. More analyses are in Appendix.

On-Device Latency: Table 4 shows the on-device latency of prefilling and generation. We measured this latency on the mobile edge device equipped with Snapdragon® 8 Elite chipset. Here, prefilling latency encompasses the latencies of sub-task classification, sub-task execution, and main task execution for each turn. Similarly, generation latency is computed. Technically, the generation latency is about 50 times that of the prefilling latency. Nevertheless, in the baseline, the prefilling latency drastically increases beyond generation latency as the number of turns grows. This underscores the importance of an efficient prefilling strategy in multi-turn interactions. Our method achieves very low latency, which is even lower than typical generation latency, making it suitable for practical use. Also, as the per-task classification can be batch-processed, then the latency is further reduced where the classifica-

Dataset	Model	Method	Sub-Task		Main Task
			Query Rewrite (ACC, DCG))	Chat Summary (Cohere)	Answering (Cohere, Correct)
TopiOCQA-Task+	LLaMA3.1-Instruct (8B)	Full Re-load	(63.04, 31.07)	98.77	(86.45, **56.50**)
		Recent Re-load	(62.53, 30.83)	98.65	(85.90, 56.20)
		Seamless	(57.13, 21.34)	97.85	(83.30, 54.10)
		CoM (Lyu et al., 2025a)	(61.41, 29.82)	98.55	(85.40, 55.20)
		Proposed	(**63.04**, **31.34**)	**98.92**	(**87.57**, 55.59)
	Mistral-Instruct-v0.2 (8B)	Full Re-load	(61.10, **29.84**)	99.46	(82.26, **54.30**)
		Recent Re-load	(59.42,27.81)	99.30	(81.10, 53.60)
		Seamless	(53.13, 22.45)	98.50	(78.90, 52.40)
		CoM (Lyu et al., 2025a)	(59.91,29.31)	99.15	(81.00, 53.25)
		Proposed	(**61.45**, 29.83)	**99.75**	(**83.75**, 52.75)
	Qwen2.5-Instruct (8B)	Full Re-load	(**64.29**, **32.18**)	**99.75**	(**75.86**, **57.11**)
		Recent Re-load	(63.70, 31.85)	99.65	(75.50, 56.80)
		Seamless	(58.20, 29.60)	98.85	(73.10, 54.90)
		CoM (Lyu et al., 2025a)	(62.90, 31.30)	99.40	(74.50, 55.80)
		Proposed	(63.12, 31.59)	99.51	(74.18, 55.57)
QReCC-Task+	LLaMA3.1-Instruct (8B)	Full Re-load	(77.37, **28.12**)	**97.97**	(74.81, **65.45**)
		Recent Re-load	(76.70, 27.70)	97.51	(73.88, 64.32)
		Seamless	(71.50, 24.90)	96.11	(71.04, 60.12)
		CoM (Lyu et al., 2025a)	(75.90, 27.30)	97.04	(74.43, 64.10)
		Proposed	(**77.43**, 27.59)	97.74	(**75.32**, 65.04)
	Mistral-Instruct-v0.2 (8B)	Full Re-load	(74.91, 46.72)	**97.14**	(66.94, 57.38)
		Recent Re-load	(73.80, 45.10)	96.04	(65.82, 55.76)
		Seamless	(68.10, 41.00)	95.53	(62.11, 52.71)
		CoM (Lyu et al., 2025a)	(73.90, 44.90)	96.22	(65.32, 55.05)
		Proposed	(**76.04**, **48.48**)	96.99	(**67.08**, **57.95**)
	Qwen2.5-Instruct (8B)	Full Re-load	(**77.08**, **49.26**)	**99.72**	(**80.17**, **70.55**)
		Recent Re-load	(75.90, 47.90)	98.55	(77.31, 68.00)
		Seamless	(69.80, 43.20)	96.42	(74.13, 64.19)
		CoM (Lyu et al., 2025a)	(74.80, 46.60)	98.60	(78.51, 68.72)
		Proposed	(76.04, 48.48)	99.68	(79.57, 69.12)

Table 3: **Evaluation results (%) for sub-task & main-task execution** on TopiOCQA-Task+ and QReCC-Task+. Sub-tasks of API call and math problem are evaluated as answering quality (correctness) in the main task.

Method	Stage	Turn				
		2	5	10	15	Last
Full Re-load	Prefill	4.35	25.38	86.90	165.76	331.76
	Generation	1.94	8.27	18.47	25.13	39.43
Proposed	Prefill	1.26	5.06	11.30	15.88	23.51
	Prefill (*bp*)	**0.95**	**3.84**	**8.56**	**11.60**	**16.79**
	Generation	2.02	8.34	18.80	25.52	41.15

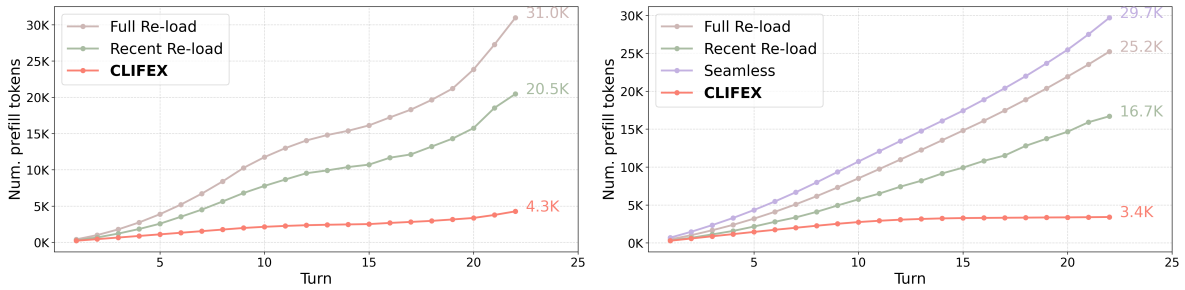
Table 4: **Latency (↓) in terms of seconds on edge device for LLaMA3.1-Instruct (8B).** '*bp*' denotes the batch processing in sub-task classification.

tion latency was reduced by half compared to the case without batch processing.

5 Related Works

Efficient LLM Inference. To reduce the computational and memory overhead in inference, several works have explored various strategies addressing KV caches. Approaches targeting task-agnostic inference efficiency are based on a single LLM. Then, they enhance KV cache efficiency via strategies such as sharing KV caches across layers (Yang et al., 2024b), computing KV caches for a subset of layers (Wu and Tu, 2024), and saving KV caches for fast GPU-to-CPU offloading (Lee et al., 2025).

In contrast, methods assuming different tasks utilize two or more task-specific models. They leverage fine-tuning or model adaptation to facilitate KV cache sharing and reuse. FTHSS (Lyu et al., 2025b) exploited learnable prompt tokens for KV cache sharing in LLMs. Similarly, DroidSpeak (Liu et al., 2024b) optimizes context sharing in fine-tuned LLMs by selectively recomputing critical layers while reusing non-critical KV cache segments. KVLINK (Yang et al., 2025) pre-computes and reuses KV caches for document segments, em-



(a) Sub-Task (b) Main-Task

Figure 3: **Turn-wise prefilled tokens for main-task and sub-task execution until each turns** in LLaMA3.1-Instruct (8B) on TopiOCA-Task+.

ploying mixed-data fine-tuning. While efficient, these methods require model modifications, often fine-tuning. Their applicability in multi-turn interactions, however, remains underexplored.

Multi-turn Conversation. In LLM, the multi-turn conversation can be categorized into task-oriented dialogue (TOD) and open-domain dialogue (ODD) (Yi et al., 2024). In TOD, LLMs assist users in achieving specific goals, often requiring structured interaction and the potential management of external APIs or retrieval of external knowledge if necessary. ODD focuses on engaging in free-form, natural conversations without a predefined task constraint.

While these distinctions exist, LLMs exhibit substantial performance drop in multi-turn conversations (Laban et al., 2025b), which stems from the inherent difficulty for LLMs in effectively managing the dialogue flow over turns. Effectively handling multi-turn conversations necessitates sophisticated capabilities (Laban et al., 2025b)—understanding history, interpreting past exchanges, and adapting to evolving user objectives—which introduce complexities like maintaining coherence, ensuring alignment with shifting intentions, and mitigating cumulative errors and contextual drift. Mixed TOD/ODD dialogues present further complexities, Such interactions necessitate capabilities like dynamic task classification, instruction adaptation based on the identified task, and potentially sub-task execution for TOD. Nevertheless, comprehensive datasets designed to capture the nuances of these complex multi-turn interactions have not yet curated. In this work, we carefully designed datasets for assessing LLMs on such complex multi-turn scenarios.

6 Conclusions

We introduced **CIFLEX**, a new multi-turn interactions strategy that enables efficient multiple sub-task handling under a single on-device LLM. **CIFLEX** reduces redundant computation by reusing KV cache from the main conversation flow and executing sub-tasks through instruction-only prefilling in side paths. After execution, the model seamlessly returns to the main path using the preserved cache, eliminating costly prefill operations. To support sub-task selection, we proposed a hierarchical classification strategy that decomposes multi-choice decisions into binary ones, making it suitable for small-scale models. For validation, we present novel multi-turn, multi-task datasets (TopiOCQA-Task+ & QReCC-Task+). Experiments demonstrated that **CIFLEX** notably lowers computational cost while maintaining task performance, highlighting its practicality for scalable, on-device conversation systems.

7 Limitations

While **CIFLEX** demonstrates strong efficiency gains and robust task handling within multi-turn conversations, several limitations remain in multi-turn conversation system. Due to the current limitations of LVMs, results from image or audio API calls are approximated using textual descriptions in LLM. For retrieval sub-tasks, we rely on off-the-shelf retrievers. Future advances in the related fields may enable seamless integration as a complete multi-turn conversation with AI assistants. Also, our datasets, while covering multi-turn interactions, could be extended to include further longer conversational trajectories. Lastly, although **CIFLEX** significantly reduces prefill latency, generation latency remains a dominant cost, which could be mitigated through further optimization in future edge hardware.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *ArXiv:2303.08774*.

Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy. 2022. TopiOCQA: Open-domain conversational question answering with topic switching. *Transactions of the Association for Computational Linguistics*, 10:468–483.

Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. Open-domain question answering goes conversational via question rewriting. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *ArXiv:2407.21783*.

Shengyue Guan, Haoyi Xiong, Jindong Wang, Jiang Bian, Bin Zhu, and Jian-guang Lou. 2025. Evaluating llm-based agents for multi-turn conversations: A survey. *arXiv preprint arXiv:2503.22458*.

Islam A Hassan and Yvette Graham. 2024. Advancing open-domain conversational agents-designing an engaging system for natural multi-turn dialogue. In *Proceedings of the 1st Workshop on Simulating Conversational Intelligence in Chat (SCI-CHAT 2024)*, pages 75–79.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: low-rank adaptation of large language models. *International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025a. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025b. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*.

Sanghyeon Lee, Hongbeen Kim, Soojin Hwang, Guseul Heo, Minwoo Noh, and Jaehyuk Huh. 2025. Efficient llm inference with activation checkpointing and hybrid caching. *arXiv preprint arXiv:2501.01792*.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, and Esha Choukse. 2024a. Droidspeak: Kv cache sharing for cross-llm communication and multi-llm serving. *arXiv preprint arXiv:2411.02820*.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, et al. 2024b. Droidspeak: Kv cache sharing for cross-llm communication and multi-llm serving. *arXiv preprint arXiv:2411.02820*.

Yuanjie Lyu, Chao Zhang, Yuhao Chen, Yong Chen, and Tong Xu. 2025a. Streamlining the collaborative chain of models into a single forward pass in generation-based tasks. *arXiv preprint arXiv:2502.11083*.

Yuanjie Lyu, Chao Zhang, Yuhao Chen, Yong Chen, and Tong Xu. 2025b. Streamlining the collaborative chain of models into a single forward pass in generation-based tasks. *arXiv preprint arXiv:2502.11083*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Haoyi Wu and Kewei Tu. 2024. Layer-condensed kv cache for efficient inference of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11175–11188.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighof. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. 2025. Kvlink: Accelerating large language models via efficient kv cache reuse. *arXiv preprint arXiv:2502.16002*.

Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. 2024b. Kvsharer: Efficient inference via layer-wise dissimilar kv cache sharing. *arXiv preprint arXiv:2410.18517*.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. *arXiv preprint arXiv:2402.18013*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory.

- Appendix -

CIFLEX: Contextual Instruction Flow for Sub-task Execution in Multi-Turn Interactions with a Single On-Device LLM

A Without $r^{(t)}$

In conversational search (RAG), retrieved passages $r^{(t)}$ often constitute a large portion of the conversation history. While reducing prefill tokens by omitting passages at each turn (maintaining only the full Q&A history) is possible, we observed a severe degradation in retrieval performance: ACC and DCG scores dropped from 63.04/31.34 to 50.12/20.70.

B Token Cost Analysis on QReCC-Task+

As shown in 4, we can see the similar trend of the prefill-token efficiency with TopioCQA-Task+ in sub-task classification, and sub and main task execution.

C Prompt for Evaluation Metric

Correctness metric

```
Given the description for a dialogue, evaluates if the
response can be considered as a correct response of the
dialogue\n(labels: (0 : wrong, 0.5 : partial, 1 : correct)\
n You should return only digit 0, 0.5, or 1. \nresponse: {
response}\n detailed dialogue description: {answer}
```

Coherence metric

```
Each dialogue is multi-turn conversation between user and
AI assistant. Assesses whether the response is naturally
and coherently structured, connecting the prompt\n  (labels:
 0 : not coherent, 0.5 : partially coherent, 1 : coherent)
.\n You should return only digit 0, 0.5, or 1.\nprompt: {
prompt}\nresponse: {response}
```

D Inference Prompt Templates

We also provide the prompt templates used for inference (sub-task classification, sub-task execution, main-task execution) in Fig. 5, 6, 7. In the side-task, we designed that a large parts is shared across sub-tasks as shown in 7.

E Dataset

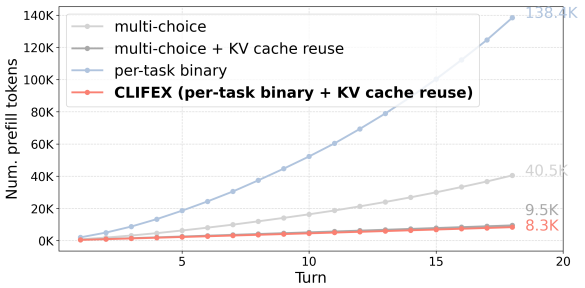
E.1 Data Curation Prompt

We fully provide the dataset curation prompt in Fig. 8, 9, 10, 11.

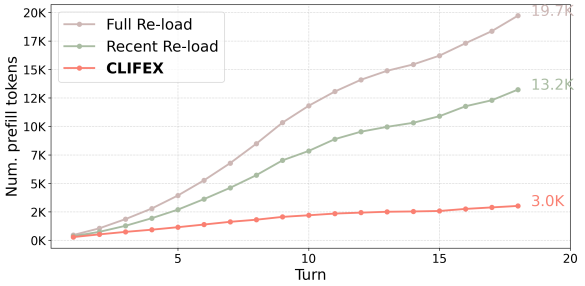
E.2 More Details

In addition, for RAG and math problem solving turns, we use the original datasets themselves where math problems are randomly sampled for each conversational search (RAG) dialogue. Finally, the curated multi-turn conversation is a series of RAG questions, multi-modal questions, casual chat (start, on-going, ending), and math solving. Notice that, as rewriting and retrieval augmented generation (RAG) is more challenging than other tasks, this kind of turns are about 40% in each series of turns.

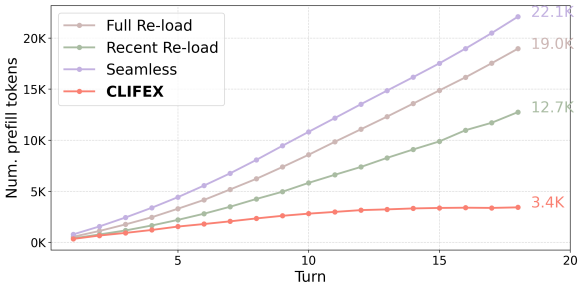
For fairness, we employed the LLaMA3.1-Instruct (70B) for the generated turns since we use the GPT4 for the evaluation.



(a) Sub-task Classification



(b) Sub-task execution



(c) Main-task execution

Figure 4: **Turn-wise prefilled tokens for sub-task classification, and main-task and sub-task execution until each turns** in LLaMA3.1-Instruct (8B) on QReCC-Task+ dataset.

Prompt template for main task

```
A multi-trun or single-turn conversation between user and AI assistant is given. \
Answer the last user question. Ensure that only answer the last user question. If a context is provided for the last question,
use the context as the major source for the answer. \
Answer should be less than 50 words. Do not output conversation history or other unnecessary words as "Here is..."

#question1: {question_1}
#answer1: {answer_1}
.
.
.
#question{n}: {question_n}
```

Figure 5: Prompt template for main-task execution.

Prompt template for sub-task clssi. (per-task binary)

```
Your new goal: for the target last user question, discern if the following sub-task is required or not.

Answer format: Ensure to provide your answer as ONLY "Yes" or "No" without adding unnecessary words or reasons, descriptions.
i.e. "#Answer: ["Yes" or "No"]"

Sub-task:
{Sub-task description}.
```

Figure 6: Prompt template for turns for answering in sub-task.

Prompt template for sub-task execution

```
Note that, now you are a helpful, respectful and honest assistant for a following sub-task.

Sub-task:
{Sub-task description}.
```

Figure 7: Prompt template for turns for sub-task execution (ours).

Dataset: Prompt template for API calling turn

```
You are a helpful conversation simulator between User and AI assistant. \
Given a series of previous conversation (question and answers are seperated by [SEP]), make a follow-up user interaction and AI assistant response based on a hypothetical image.

Assume the image has been processed by a vision-language API (e.g., LLaVA), and the API has returned a detailed textual image description.


Your task is to generate:
1. A realistic **LLaVA-style textual image description sentences**, which clearly and explicitly mentions visual content.
2. A **casual user question** that refers to the image
3. An **AI assistant answer** that responds to the question based only on the given textual description.

---


### Guidelines

- The **user question** must:
  - Explicitly or implicitly refer to the image.
  - Be casual and natural in tone.

- The **assistant answer** must:
  - Use only information that is **present** in the image description.

  - Avoid interpretation, speculation, or reasoning.

---

### Format:

##Textual Image Description: [Generated description]

##User Question: [User - 128) * 64 + (‘ - 128) ‰s question that can be answered with no inference, just by reading the description]
##AI Assistant Answer: [Answer using only what - 128) * 64 + (‘ - 128) ‰s stated in the description]

---

### Example:


##Textual Image Description: A small white dog is sitting on a blue couch with a red blanket draped over one side. A green ball
 lies on the floor in front of the couch, and sunlight is coming in through a nearby window.
##User Question: What color is the couch in this image?
##AI Assistant Answer: The couch is blue.

---

Now generate more examples in this format, strictly following the instructions above.


### Previous Conversation: {conversation_history}

Output:
```

Figure 8: Prompt template data curation for API call turns.

Dataset: Prompt template for start of casual conversation

```
You are a helpful conversation simulator between User and AI assistant. \
Given a series of previous conversation (question and answers are separated by [SEP]), make a follow-up user interaction and AI
 assistant response. \
In the previous conversation, to answer the user questions, information retrieval or API calls are required. \
However, from now on, it's different. The follow-up user interaction should be non-knowledge intensive and not necessarily in
the form of a question. \

It should be a casual chat, focusing on topics such as user's daily life or sentiment. \
The user interaction can be a statement, a greeting, or any other form of casual conversation starter.

The user interaction and AI assistant response should consist of concise one or two sentences.

Make a single turn of user interaction and AI assistant response where each of them is seperated by [SEP]:
[User interaction] [SEP] [AI assistant response]


### Previous Conversation:
{conversation_history}

Output:
```

Figure 9: Prompt template data curation for starting casual chat turns.

Dataset: Prompt template for on-going casual conversation

```
You are a helpful conversation simulator between User and AI assistant. \
Given a series of previous conversation (question and answers are separated by [SEP]), make a follow-up user interaction and AI
 assistant response. \
Similar to the last conversation, the follow-up user interaction and AI assistant response should be casual chat with no need
of knowledge retrieval. \
Namely, it should be a casual chat, focusing on topics such as user's daily life or sentiment.   \

Also, it is not necessarily in the form of a question. \

The user interaction and AI assistant response should consist of concise one or two sentences.

Make a single turn of user interaction and AI assistant response where each of them is seperated by [SEP]:
[User interaction] [SEP] [AI assistant response]

### Previous Conversation:

{conversation_history}

Output:
```

Figure 10: Prompt template data curation for on-going casual chat turns.

Dataset-Prompt template for finishing casual conversation

```
You are a helpful conversation simulator between User and AI assistant. \
Given a series of previous conversation (question and answers are separated by [SEP]), make the final turn of user interaction
and AI assistant response. \
Crucially, the final turn must imply that the conversation is ending.
The user interaction and AI assistant response should consist of concise one or two sentences.


Make a single turn of questions and answers where each of them is seperated by [SEP]:
[User interaction] [SEP] [AI assistant response]

### Previous Conversation:
{conversation_history}

Output:
```

Figure 11: Prompt template data curation for finishing casual chat turns.