MemCtrl: Using MLLMs as Active Memory Controllers on Embodied Agents

Anonymous ACL submission

Abstract

Foundation models rely on in-context learning for personalized decision making. The limited size of this context window necessitates memory compression and retrieval systems like RAG. These systems however often treat memory as large offline storage spaces, which is unfavorable for embodied agents that are expected to operate under strict memory and compute constraints, online. In this work, we propose MemCtrl, a novel framework that uses Multimodal Large Language Models (MLLMs) for pruning memory online. MemCtrl augments MLLMs with a *trainable* memory head μ that acts as a gate to determine which observations or reflections to retain, update, or discard during exploration. We evaluate with training two types of μ, 1) via an offline expert, and 2) via online RL, and observe significant improvement in overall embodied task completion ability on μ-augmented MLLMs. In particular, on augmenting two low performing MLLMs with MemCtrl on multiple subsets of the EmbodiedBench benchmark, we observe that μ-augmented MLLMs show an improvement of around 16% on average, with over 20% on specific instruction subsets. Finally, we present an qualitative analysis on the memory fragments collected by μ, noting the superior performance of μ augmented MLLMs on long and complex instruction types.

1 Introduction

An overarching goal of Embodied AI is the development of a generalist agent that can perform consistently well with high success on diverse tasks, environments and instructions (Szot et al., 2025). A common paradigm to achieve this has been to utilize foundation models to develop task solving frameworks (Mu et al., 2023; Yang et al., 2025). While a few of these methods generalize well to diverse tasks and instructions (Driess et al., 2023; Zawalski et al., 2024), they are constrained by high training costs, prohibiting them from quickly being able to adapt to novel real-time settings where the data is out of distribution. Further, finetuning large foundation models incurs significant computational capacity, proving to be a significant hurdle in the democratization of these methods (Liang et al., 2022), especially in the context of robotics, where computation on the edge is of vital importance.

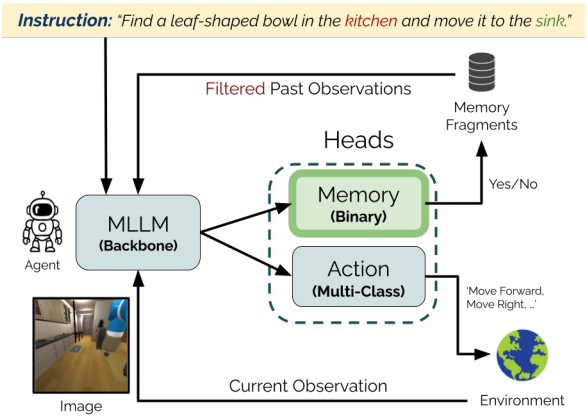

Figure 1: **Overview**: We present MemCtrl, a novel memory filterting scheme to improve decision making performance on **small** MLLMs tackling embodied tasks. Our approach proposes a *trainable memory head* (green box labeled "Memory") that learns to actively filter out redundant observations on-the-go. This form of *active* filtering alleviates issues with inefficient retrieval from stored observations, while also enabling scalability as a detachable memory head.

An alternative, more feasible paradigm has been a modular system where foundation models are used in conjunction with memory banks (Zhong et al., 2023; Wang et al., 2024) of past experiences and reflections. Foundation models including very large Multimodal Large Language Models (MLLMs) such as LLaMA 4 (Touvron et al., 2023) and Deepseek V3 (DeepSeek-AI et al., 2025) are limited by the size of their con-

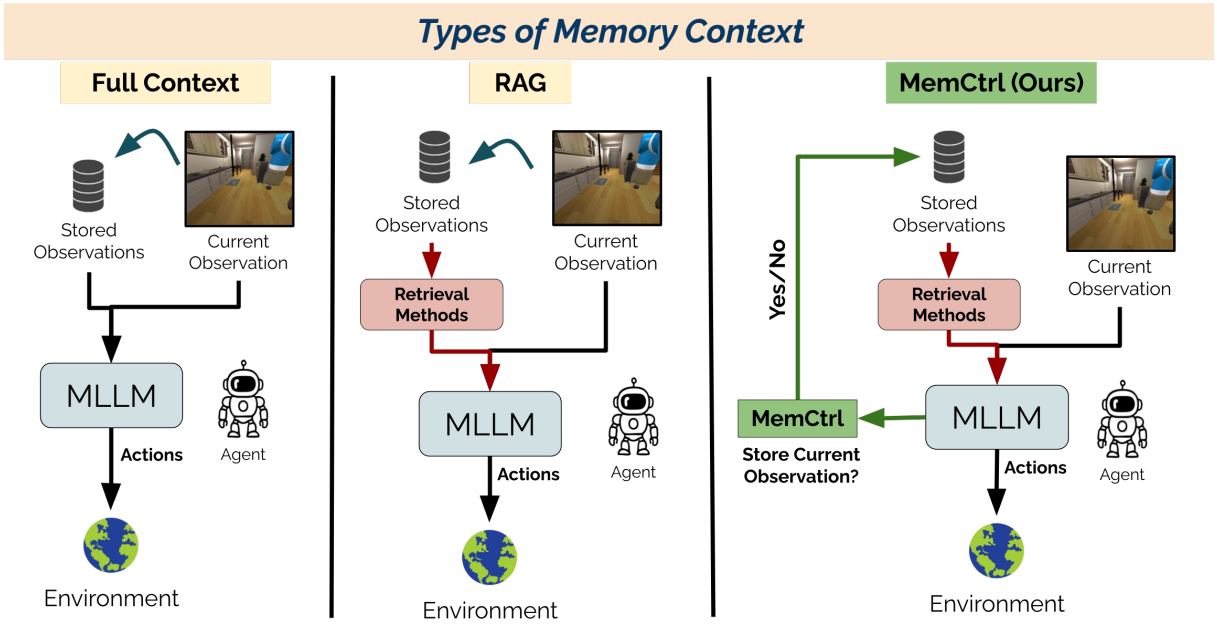

Figure 2: **Comparison with Prior Work**: We present MemCtrl, a novel approach to train *"memory heads"* to filter observations on the go. Prior work either used the entirety of stored observations as context (left) or filtered them via a variety of Retrieval Augmented Generation (RAG) based schemes (red arrows), both of which assume the parsing of large amounts of data offline. MemCtrl introduces transferrable heads to use on MLLM backbone (green arrows) to actively filter observations.

text window, and developing methods to refine and selectively pass memory as context is an active area of research (Wu et al., 2025). Prior work in this area includes use of intrinsic model editing techniques for memory injection (Mitchell et al., 2022; Meng et al., 2023), or extrinsic interactions with episodic logs, Retrieval-Augmented Generated (RAG) (Gao et al.), or long-range latent states (Park et al., 2023; Wang et al., 2023).

While both paradigms have shown improved performance in multi-step reasoning, their implementation on embodied robot agents raises practical issues. Embodied agents providing assistance often use small models ($<$ 20B parameters) that work locally on-device, with often limited or only cloud access to large memory storage. Moreover, these agents need to generalize to novel settings, making it preferable to have modular, lightweight segments that are easily *transferrable*.

To model a more efficient memory framework for embodied agents, we draw inspiration from how humans store memories of experiences. While performing various embodied tasks, humans do not accumulate every observation for later retrieval, but rather learn to actively filter out only certain vital fragments that we assume to be relevant to our task (He et al., 2025). When queried, we reconstruct the missing fragments of memory through commonsense reasoning. This makes us humans highly efficient reasoners even with limited storage.

We aim to endow compact embodied agents with a similar ability: rather than relying on large external memory banks or complex retrieval pipelines, the agent must actively learn to store vital memories while filtering redundant ones on the go. Learning this skill across a wide range of tasks would enable scalable self-improvement under tight computational and memory budgets.

Main Results: To address these issues, we present *MemCtrl*, a transferrable memory augmentation scheme that aims to improve the embodied decision making performance of **small** models. MemCtrl introduces a trainable memory head that learns to selectively store *memories of importance*, increasing both parameter and memory efficiency for self-improving embodied agents. Our contributions are as follows:

- **Active Memory Filtering:** We introduce two lightweight memory heads μ trained on top of a frozen MLLM backbone to actively filter observations to determine which to keep and which to discard in memory. Unlike prior retrieval-based work involving filtering

large observational data offline, μ enables the MLLM to engage in real-time filtering, which is particularly useful in memory-constrained settings involving small models.

- **Transferrable Heads:** μ is model-agnostic and attaches to any off-the-shelf MLLM without having to finetune or edit the backbone to remove redundant observations for more prudent decision making. This modular design allows MemCtrl to transfer across embodied setups and vision-language backbones, enabling its scalable transfer across embodied agents in diverse settings.

- **Improving Small Model Performance:** Finally, attaching μ to the worst performing agents on the Habitat (Puig et al., 2023) and ALFRED (Shridhar et al., 2020) splits of EmbodiedBench (Yang et al., 2025) shows a significant improvement on task performance of around 16% on average, while also storing significantly fewer observations. This efficiency makes MemCtrl favorable for real-world deployment.

2 Related Work

2.1 MLLMs in Embodied AI

Several works in recent literature have leveraged large language models (LLMs) for high-level robot planning. For example, Ahn et al. (2022) introduce a framework (SayCan) where an LLM translates natural-language instructions into feasible robot actions constrained by a set of learned skills. This approach demonstrated that LLMs can provide semantic task knowledge, but it relies on a fixed library of affordance-grounded skills and struggles with adapting to novel situations. Recent multimodal LLMs extend this idea by directly integrating visual inputs. For instance, PaLM-E (Driess et al., 2023) is a vision-language model that outputs robotic actions, achieving good generalizability across manipulation and navigation tasks. However, PaLM-E requires a lot of training data, limiting its real-time adaptability. Similarly, RT-2 (Zitkovich et al., 2023) augments a vision-language model with web-scale pretraining to create a vision-language-action (VLA) agent for improved zero-shot object understanding. However, it makes use of short temporal contexts without an explicit memory mechanism. In contrast, our work uses an MLLM as an active memory controller, enabling the agent to retain and recall cross-modal information over long horizons. Our design empowers embodied agents with small models to handle complex, extended tasks without requiring large-scale training.

2.2 Memory-Augmented Agents

Recent works have explored augmenting LLM-based embodied agents with memory to address challenges with long-horizon task solving. Mai et al. (2023) propose using an LLM itself as a 'robotic brain' that maintains an egocentric memory of the agent's observations and dialogue. Their system shows that a textual memory stream can help the agent refer back to important context, improving consistency in multi-step tasks. Another approach is to attach an external memory to the agent. In the HELPER framework (Sarch et al., 2023), they maintain a repository of dialogue-to-action examples, retrieving relevant past interactions to condition the LLM when parsing new instructions. They show that dynamic memory of prior events or user preferences can overcome the limitations of fixed prompts or short context windows to improve task completion success. However, this still relies on a separate module for populating memory, with the foundation model having a passive role. In contrast, our work introduces a framework that enables the MLLM to *actively* control what and what not to store in the agent's memory. Figure 2 highlights this idea. The MemCtrl module takes in the embedding of the current observation from the MLLM and determines whether or not the observation should be stored.

3 MLLM-based Embodied Agents

Let $\mathcal{M}$ be an MLLM, $\mathcal{O}_c$ be the current observation, and $\mathcal{I}$ be the instruction. A baseline method utilizes $\mathcal{M}$ to translate the observations and instructions into actionable outputs for the agent as follows:

$$a = \mathcal{M}(\mathcal{O}_c, \mathcal{I}),$$

where $a \in \mathcal{A}$ such that $\mathcal{A}$ represents a set of feasible actions for the agent in the environment. Using $\mathcal{M}$ as a prior in this "zero-shot" manner leads to subpar performance, since the agent does not have any continual context of the environment, and takes actions solely based off of the current image observed and its capacity for commonsense reasoning (Dorbala et al., 2022; Majumdar et al., 2022; Gadre et al., 2023; Shah et al., 2023).

3.1 Memory-Augmented MLLM Agents

One way to improve zero-shot performance of $\mathcal{M}$ is to provide continued context to the agent as a set of past observations, i.e., $\{\mathcal{O}_c, \mathcal{I}, \mathcal{C}\}$, where $\mathcal{C}$ is the context passed to $\mathcal{M}$:

$$\mathcal{C} = \{R_i, \mathcal{O}_i\}_{i:1 \to n}.$$

R_i and $\mathcal{O}_i$ respectively represent the i^{th} reflection and observation in n timesteps.

EmbodiedBench (Yang et al., 2025) highlights that adding history as context greatly influences performance, particularly when it comes to embodied tasks involving long-horizon instructions. Further, they highlight benefits to adding memory in the form of agent reflections, where the agent reflects on its past interactions with the environment to refine its future plan.

While adding history improves zero-shot performance, $\mathcal{M}$ is limited by a context window h, and $n < h$. To circumvent this issue, several approaches explore retrieval pipelines to compress memory to obtain context $c = F(\mathcal{C}, \mathcal{I})$ for $\mathcal{M}$. F here is retrieval function that selects the most relevant parts of the whole context given an objective, which in this case is the instruction $\mathcal{I}$.

Inefficient Retrieval: In alignment to other reported results (Liu et al., 2024; Packer et al., 2023), we similarly observe that as the size of context $n = sizeof(\mathcal{C})$ increases it leads to more inefficient retrieval, especially since we are limited by a context window of size $h << n$. Robot agents running in the wild often collect observations at high frequencies (> 1 observation per second), which quickly increases the size of n. Further, having redundant observations in memory only adds to this issue, prompting the development of better strategies for write-time memory control, especially on small models.

To achieve such active memory control, we propose *MemCtrl*, a learnable memory augmentation scheme that allows active write-time memory control on the model $\mathcal{M}$. We describe this in the following section.

4 MemCtrl: Training Memory Heads (μ)

To achieve write-time memory control, we consider three natural augmentations to $\mathcal{M}$, illustrated in Figure 3. Our objective is to improve decision making on **small** visual-language models, which also translates to small context windows

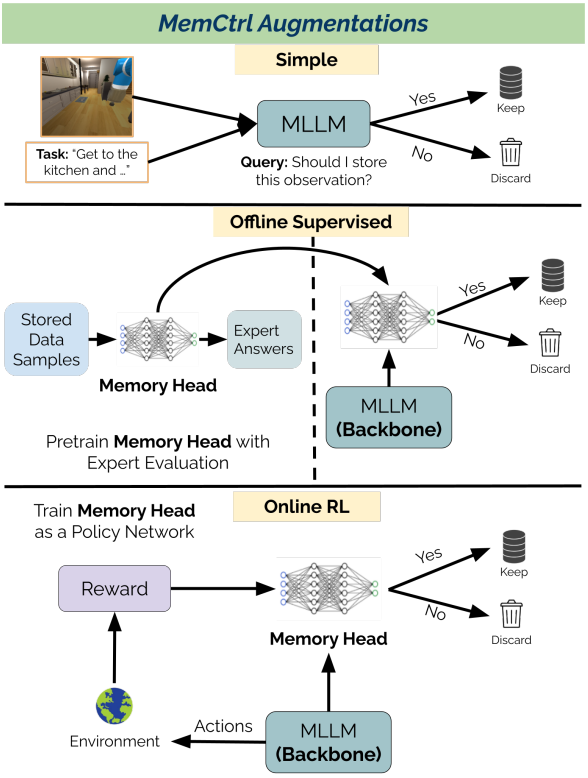

Figure 3: **MemCtrl**: We experiment with 3 augmentations. The simple case acts as a non-trained baseline, where the MLLM is directly queried about storage. In the offline supervised case, μ is first pretrained using expert answers from a high performing, expert MLLM (GPT-4o here). This trained binary classifier then acts as a head on top of the MLLM backbone. In the Online RL case, we train the memory head online as a policy network. We use a sparse reward on task success and a dense reward on action success. Note that MemCtrl is trained as a detachable head that takes the visuolingual MLLM embeddings as input.

$\mathcal{C}$'s, by empowering them to better filter observations prior to storing them in memory.

Filtering observations in this way fully avoids the problem of inefficient retrieval described in the previous section, since the agent's context is now driven by its own decisions, much akin to humans deciding only to remember moments of importance that might be meaningful to them in the future.

For this, we propose a **trainable memory head** μ as a simple binary classifier that learns to either keep or discard memory. μ integrates with the $\mathcal{M}$ backbone, to make decisions online about whether or not to store the current observation. Following

on the our definitions so far, we get

$$\begin{aligned} c &= F(\mathcal{C}, \mathcal{I}), \\ a &= \mathcal{M}_a(\mathcal{O}_c, \mathcal{I}, c), \\ b &= \mathcal{M}_\mu(\mathcal{O}_c, \mathcal{I}, c). \end{aligned} \tag{1}$$

where $b \in \{0, 1\}$ is a binary classifier that determines if the current observation must be added or discarded. The updated context, C' can then be written as,

$$C' = \begin{cases} C \cup \{(\mathcal{O}_c, a)\} & b = 1, \\ C & \text{otherwise.} \end{cases} \tag{2}$$

We consider 2 ways to integrate μ onto $\mathcal{M}$:

- **Offline, Fully-Supervised**: We first gather offline data from a high-performing $\mathcal{M}$ treated as an expert. Using the gathered negative and positive samples, we train μ offline as a binary classifier to determine which observations led to success and which led to failure. We transfer this *pretrained* memory head onto a low-performing model $\mathcal{M}$, as an expert supervision gate. We train the network with a binary cross-entropy loss function:

$$L(y, \hat{p}) = y \log(\hat{p}) + (1 - y) \log(1 - \hat{p}), \tag{3}$$

  where $y_i \in [0, 1]$ is the ground-truth label and $\hat{p}_i$ is the predicted probability from μ of whether the current observation should be stored.

- **Online RL**: We directly train the memory head and the action head via an online RL policy. We model two rewards: 1) a sparse reward for episode success, and 2) a dense reward for picking valid actions:

$$R(r, a) = r + \mathbf{1}_{a \in \mathcal{A}}, \tag{4}$$

  where $r \in \{0, 1\}$ is the binary reward signal for task completion, and $\mathbf{1}$ is an indicator function. In our approach, $r = 0$ for all steps except for goal-completing ones. These reward functions ensure that μ picks helpful observations and the action heads make valid decisions.

In both these cases, the memory head μ empowers the MLLM to play an active role in filtering observations, as highlighted in Figure 3. Further, μ is a head, and can be transferred across arbitrary MLLMs helping alleviate the cost of directly finetuning MLLMs. Algorithm 1 and 2 in the Appendix present the details of our algorithm for the supervised and RL variants, respectively.

5 Experimental Setup

Datasets. Our objective is to improve the performance of an MLLM by augmenting it with a memory head. For this, we choose EmbodiedBench (Yang et al., 2025) as a benchmark for evaluation, since it provides us with tools to automate embodied task evaluation on both ALFRED (Shridhar et al., 2020) and Habitat (Puig et al., 2023) simulators, making it easy to evaluate the performance of multiple MLLMs on various tasks. We modify this evaluator with memory heads for our task.

LLM backbones. To showcase improvement, we choose two low-performing models on EmbodiedBench, *Qwen2-VL-7B-Ins* and *Gemma-3-12B-IT* aiming to showcase an improvement in performance when augmented with MemCtrl. We run all models locally on a NVIDIA A5000 GPU.

Baseline: Simple, In-Context Learning. We prompt $\mathcal{M}$ for a binary output on whether or not to store the current observation in its memory. We do not train a memory head here, but ablate with combinations of μ and $\mathcal{M}$.

Memory Heads. Both μ's are paramterized as Linear MLP's with 3 layers, that map the backbone MLLM's embedding to a binary output.

For the offline, supervised μ, we first gather expert data using a high performing MLLM, GPT-4o which gives us a set of $X = [x_1, x_2, \ldots, x_n]$ embeddings per episode mapped to a binary episode success or failure l, giving us $[n, 1]$ training pairs for each episode. We ensure balancing of the dataset with negative and positive samples and then train the MLP to overfit using a cross-entropy loss.

For the online μ, we similarly define an MLP as the policy network that predicts a binary outcome. We define a sparse and a dense reward as described in the previous section, and train using REINFORCE (Williams, 1992).

6 Results

Table 1 presents the main results, showcasing the benefits of MemCtrl across two different LLM

Model	EB-ALFRED						EB-Habitat					
	Avg	Base	Common	Complex	Spatial	Long	Avg	Base	Common	Complex	Spatial	Long
Gemma-3-12B-IT	25.6	32	26	38	20	12	23.0	58	10	24	24	4
Gemma-3-12B-IT + μ_{Simple}	28	41	27	34	16	22	31.2	62	15	30	31	18
Gemma-3-12B-IT + $\mu_{\text{Offline Sup.}}$	**32.2**	48	31	32	23	27	31.8	66	14	35	27	17
Gemma-3-12B-IT + $\mu_{\text{Online RL}}$	27.8	38	29	41	21	26	**33.8**	60	13	37	35	24
Qwen2.5-VL-7B-Ins	4.7	10	8	6	0	2	14.3	32	2	26	14	2
Qwen2.5-VL-7B-Ins + μ_{Simple}	9.6	15	10	12	2	14	21.4	39	10	31	14	13
Qwen2.5-VL-7B-Ins + $\mu_{\text{Offline Sup.}}$	12.2	9	10	14	2	26	**22.8**	39	5	33	17	20
Qwen2.5-VL-7B-Ins + $\mu_{\text{Online RL}}$	**14.2**	10	13	21	3	24	22.2	37	3	37	16	18

Table 1: **Results**: We augment *Qwen2.5-VL-7B-Ins* and *Gemma-3-12B-IT* with the 3 variations of MemCtrl on 5 subsets of EB-ALFRED and EB-Habitat (Yang et al., 2025). Note the improve performance overall of adding the memory head μ. In particular, we note superior performance on long context and complex instructions, which tend to be long horizon where memory helps. Performance on Habitat is also much better than ALFRED overall, hinting that navigation heavy tasks that are common on the Habiat dataset might benefit from memory augmentation more than manipulation ones common in ALFRED.

backbones, *Gemma-3-12B-IT* and *Qwen2.5-VL-7B-Ins*. We pick these two models as they perform among the worst on the EmbodiedBench benchmark, aiming to show improved performance with the inclusion of our memory head.

Increased Performance with μ. The overall performance improves across EB-Alfred and EB-Habitat with adding any type of memory head μ. This is expected, since any form of memory provides continual context that is more meaningful for the MLLM.

In particular, we observe huge improvements on Long instructions, with results on *Gemma-3-12B-IT* bumping from 12 on the baseline to 26 with the μ_{RL} augmentation, and *Qwen2.5-VL-7B-Ins* going from 2 to 24. We believe this to be a result of a more strategic memory storage needed for longer-horizon tasks, like with EB-Habitat.

We also observe an overall improvement in the task performance on complex instructions, where the instructions are not just long, but also contain irrelevant information. This is very evident with the Qwen model, where it goes up 3x from 6 to 21. For instance, the following is the difference between a base and complex instruction:-

> **Base**: "Move one of the pear items to the indicated sofa."
>
> **Complex**: "When you find the fridge door open, go ahead and move an bowl to the sofa; otherwise, transport an hammer to the sofa."

The agent is expected to finish these tasks in a fixed set of timesteps, and over time, gathers more and more information about the environment as potential context for determining its next action. The base query here is fairly simple, requiring to track just a single object ('pear'). In contrast, the complex query not only has multiple objects to track ('fridge, sofa, bowl, hammer'), but is also sophisticated in its framing, requiring better reasoning. While more context would help with better reasoning, it also leads to more redundant information storage, which a trained memory head can help actively filter.

Performance across Gemma-3 and Qwen2.5. The baseline performance of Qwen2.5 as reported by EmbodiedBench is far lower than Gemma-3. We note much higher performance gains on Qwen2.5 compared to Gemma-3. Being one of the worst performing models on EmbodiedBench due to it's small size, adding a lightweight μ for active memory control bumps its performance up. Qwen2.5-VL + μ_{RL} is comparable to Ovis2-16B, a model with over twice the number of parameters that had 16% on the original benchmark.

Alfred vs Habitat: Finally, we also notice better performance on Habitat overall compared to Alfred. Tasks in habitat tend to be more navigation centric, requiring more long-horizon planning. In contrast, Alfred focuses more on reasoning on current observations for manipulating objects. We infer that memory filtration is potentially more impactful on long-horizon navigation tasks, since it helps reduce redundant frames gathered.

7 Qualitative Analysis

Figure 6 in the Appendix shows a qualitative example of MemCtrl in action.

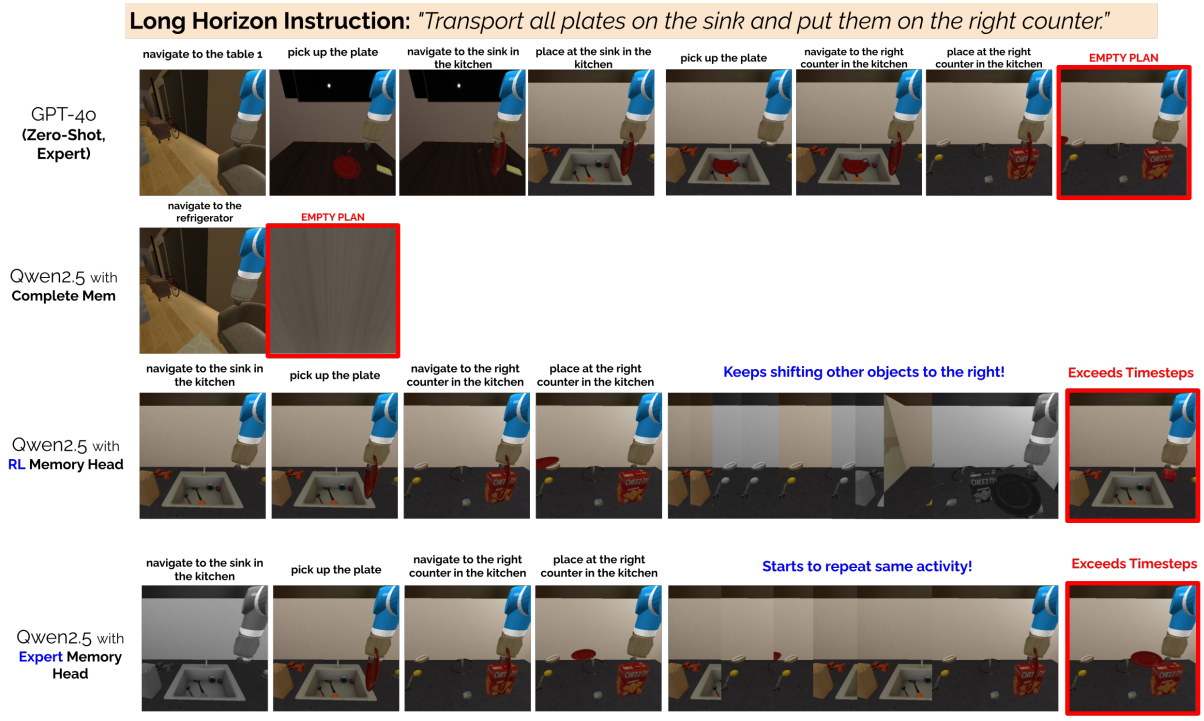

Figure 4: **Long Horizon performance on EB-Habitat**: We notice that on long horizon tasks, the expert tends to end the task early by hastily assuming that it is done (finishing after placing *one* plate instead of *all* plates). Memory heads highlight unique performance improvements, with μ_{RL} exhibiting a more **exploratory** nature by continuing to place *new* objects at the right counter, and $\mu_{\text{Exp.}}$ being more **exploitative** by repeating the same activity over and over, with a single plate. Note: Grayed out images indicate discarded memories.

7.1 Visualization

Figure 5 and Figure 4 compare a base and long episode, using GPT-4o and Qwen with and without memory. Note Qwen performs poorly on the EB-Habitat baseline, with only 14.3% average vs GPT-4o that has 59%. Augmenting Qwen with a lightweight memory head μ bumps this up to around 22%, both in the case of μ_{RL} and μ_{Expert}.

In the base case, note the good zero shot performance of the GPT-4o agent in being able to complete the task. The complete memory agent however shows an invalid action after the first step, and this is also seen with the case of Qwen + μ_{Expert}. However, with Qwen + μ_{RL} note the improved performance in being able to complete the task in fewer number of steps. Qwen is a much smaller model than GPT-4o, and with a small memory head augmentation, it is able to perform at par with its GPT-4o counterpart.

In the long horizon case, we make a few interesting observations. First, none of the 4 methods seem to be able to successfully complete the task. The instruction requires *all* plates to be transported to the right corner, requiring long-horizon task planning to keep track of which *instances* of plates have been moved.

- The **zero-shot expert** assumes that the task has ended after transferring the first plate, and hence sends no executable plan after being done. This stops the episode earlier than expected, hence failing the task.
- In the **complete memory case**, the agent is overloaded with too many experiences from the past in its memory, and this floods the MLLM with too much context, leading to a bad action being selected (*navigate to the refrigerator*).
- In the **Qwen + μ_{RL}** case, the agent successfully places **one** plate from the sink onto the right counter. While not ending the episode early like with the expert, it continues to place other objects including apples and wrenches in the vicinity to the right, ultimately running out of timesteps. This behavior can be associated with more active *exploration*, as the agent seeks to explore new directions to complete the instruction. Note that the memory head filters out most of these extraneous observations (images in gray).

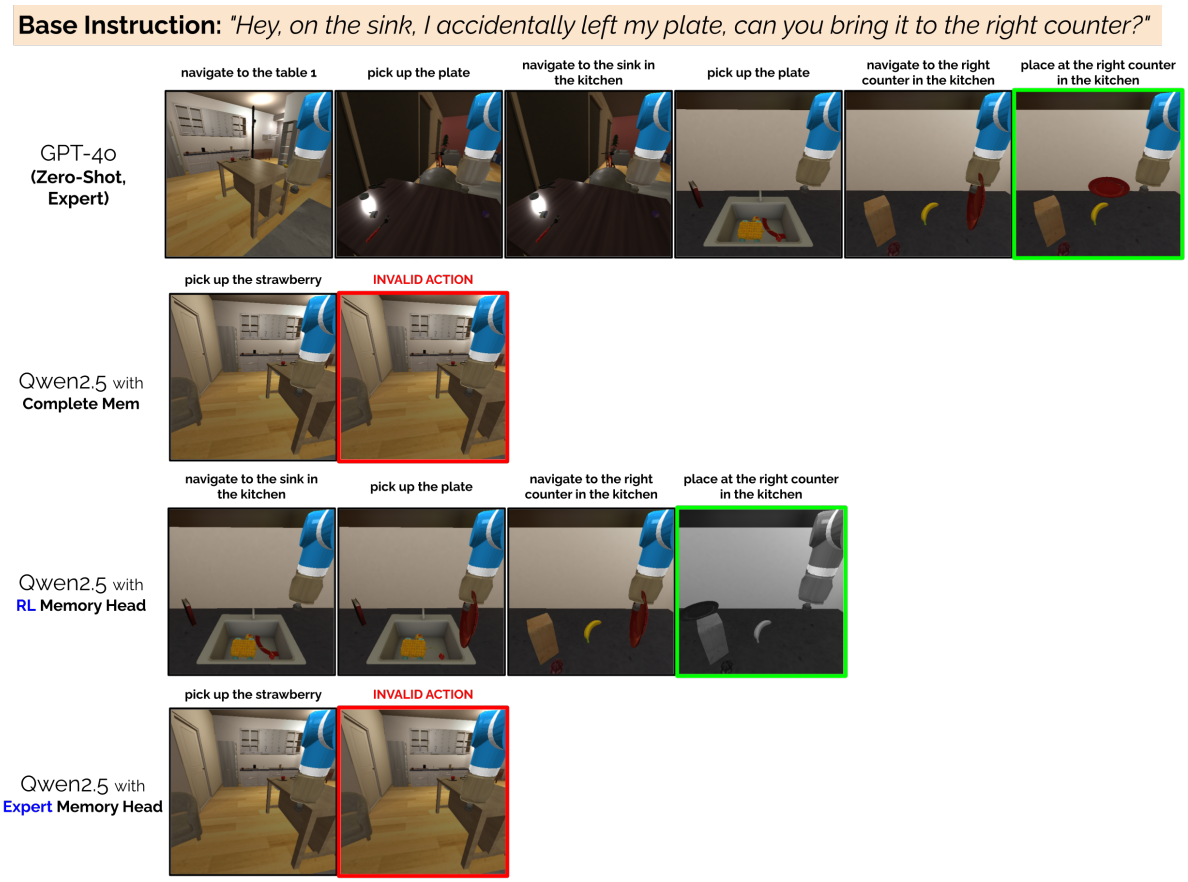

Figure 5: **Base Performance on EB-Habitat**: Here, we compare the performance of GPT-4o vs Qwen2.5-VL-7B-Ins, with various memory augmentations. While GPT-4V gives superior zero-shot performance, it is a very large model that is not easily finetunable. In this instance it also takes more steps to complete the task. On the other hand, μ boosts the performance of a significantly weaker model, and in this scenario, even doing it quicker with μ_{RL}. The expert head fails here however, similar to the complete memory, causing the episode to end early. Note: Grayed out images indicate discarded memories.

- In the **Qwen + μ_{Expert}** case, we observe the opposite behaviour or *exploitation*, where the agent just starts to repeat the same activity over and over again. It navigates to the sink, picks up a plate, transfers it to the right counter, then goes back to the sink, and so on. This form of exploitative behavior seems desirable, especially since the agent is tasked with transporting all plates to the sink. The expert memory head as a result decides to keep almost all of the observations, as the agent keeps doing the right thing in following the instruction. However, as the **same plate** is constantly being moved, it highlights a limitation of the expert memory head in being too conservative with classifying significant memories.

8 Conclusions

In this work, we present MemCtrl, a lightweight, transferrable memory framework that introduces a memory head μ on a backbone MLLM to actively filter *memories of importance*. Instead of editing the MLLM directly or using external retrieval methods like RAG, μ is trained as a binary classifier decide whether or not to store current observations on the go. We introduce two ways to train μ, and present a qualitative and quantitative analysis of μ-augmented low parameter Qwen and Gemma models. Our results show significant performance improvement of around 16% on average, across ALFRED and Habitat splits of the EmbodiedBench dataset. Further, we note the superior performance on instructions involving complex or long-horizon language.

9 Limitations

Our work has a few limitations. The supervised learning method requires expert demonstrations from a stronger model to understand which observations contribute to success, and the RL variants suffer from the inefficiencies that come with sparse reward structures. Future work could look into designing better reward functions that can capture the *interesting-ness* of an observation (Gardezi et al., 2021). Furthermore, the benefits of MemCtrl degrade over short horizons, suggesting a limited need to train a memory head to filter observations in more basic settings. Another possible avenue we are excited to explore is to incorporate audio observations to increase the complexity of observations stored. Sim-to-real transfer of our work via real world experiments is also a potential extension.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2022. Do as i can, not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

Kwesi Adu Cobbina and Tianyi Zhou. 2025. Where to show demos in your prompt: A positional bias of in-context learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29548–29581, Suzhou, China. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Vishnu Sashank Dorbala, Gunnar Sigurdsson, Robinson Piramuthu, Jesse Thomason, and Gaurav S Sukhatme. 2022. Clip-nav: Using clip for zero-shot vision-and-language navigation. *arXiv preprint arXiv:2211.16649*.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, , and 1 others. 2023. PaLM-E: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. 2025. Context length alone hurts llm performance despite perfect retrieval. *Preprint*, arXiv:2510.05381.

Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. 2023. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey.

Maham Gardezi, King Hei Fung, Usman Mirza Baig, Mariam Ismail, Oren Kadosh, Yoram S Bonneh, and Bhavin R Sheth. 2021. What makes an image interesting and how can we explain it. *Frontiers in psychology*, 12:668651.

Zizhan He, Maxime Daigle, and Pouya Bashivan. 2025. Building spatial world models from sparse transitional episodic memories. *arXiv preprint arXiv:2505.13696*.

Sanghwan Kim, Daoji Huang, Yongqin Xian, Otmar Hilliges, Luc Van Gool, and Xi Wang. 2024. Palm: Predicting actions through language models. In *European Conference on Computer Vision*, pages 140–158. Springer.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2022. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Jinjie Mai, Jun Chen, Bing Li, Guocheng Qian, Mohamed Elhoseiny, and Bernard Ghanem. 2023. LLM as A robotic brain: Unifying egocentric memory and control. *arXiv preprint arXiv:2304.09349*.

Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. 2022. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *Advances in Neural Information Processing Systems*, 35:32340–32352.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass editing memory in a transformer. *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022.

Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 15817–15831. PMLR.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Yu Qiao, and Ping Luo. 2023. EmbodiedGPT: Vision-language pre-training via embodied chain of thought. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith R. Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 1–22.

Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, and 1 others. 2023. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*.

Gabriel Sarch, Yue Wu, Michael Tarr, and Katerina Fragkiadaki. 2023. Open-ended instructable embodied agents with memory-augmented large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3468–3500, Singapore. Association for Computational Linguistics.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. 2023. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Andrew Szot, Bogdan Mazoure, Omar Attia, Aleksei Timofeev, Harsh Agrawal, Devon Hjelm, Zhe Gan, Zsolt Kira, and Alexander Toshev. 2025. From multimodal llms to generalist embodied agents: Methods and lessons. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10644–10655.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi "Jim" Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, and 1 others. 2024. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*.

Mengge Xue, Zhenyu Hu, Liqun Liu, Kuo Liao, Shu'ang Li, Honglin Han, Meng Zhao, and Chengguo Yin. 2024. Strengthened symbol binding makes large language models reliable multiple-choice selectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4331–4344.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja V. Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. 2025. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. 2024. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. *Preprint*, arXiv:2309.03882.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*. Preprint. See also _AAAI_ 2024 version.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, and 35 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning (CoRL)*, volume 229 of *PMLR*, pages 2165–2183.

A Algorithms

Algorithm 1 Training Memory Heads μ with Offline, Supervised Learning.

Require: Labeled ground-truth expert answers $E = \{(y_i, \mathcal{O}_i^E)\}_{i=1}^K$, Memory Head $\mathcal{M}_\mu$, Action Head $\mathcal{M}_a$

Pretraining μ
for i in $[1, K]$ **do**
    $\hat{p}_i = \mu(\mathcal{O}_i^E)$
    Calculate loss $L(y_i, \hat{p}_i)$ with Equation 3
    Update μ
end for
Test-time with trained μ
$c \leftarrow F(\mathcal{C}, I)$
$a \leftarrow \mathcal{M}_a(\mathcal{O}_c, \mathcal{I}, c)$
$b \leftarrow \mathcal{M}_\mu(\mathcal{O}_c, \mathcal{I}, c)$
if $b = 1$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\mathcal{O}_c, a)\}$
end if

Algorithm 2 Training Memory Heads μ with Online, RL.

Require: Current observation $\mathcal{O}_c$, Instruction $\mathcal{I}$, Total Context $\mathcal{C}$, Memory Head $\mathcal{M}_\mu$, Action Head $\mathcal{M}_a$

$c \leftarrow F(\mathcal{C}, I)$
$a \leftarrow \mathcal{M}_a(\mathcal{O}_c, \mathcal{I}, c)$
$b \leftarrow \mathcal{M}_\mu(\mathcal{O}_c, \mathcal{I}, c)$
if $b = 1$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\mathcal{O}_c, a)\}$
end if
Calculate reward with Equation 4
Update μ

B Experimental Details

We train the memory head μ both via offline supervised learning with an expert, and online RL. In both cases, μ is an MLP initialized with 3 layers to map the hidden MLLM dimension to a binary output.

Expert, Offline Supervised: Here, we first gather a dataset using an expert model that performs well on EmbodiedBench. We use GPT-4o for this, which has high success rates of 56.3% and 59.0% on EB-ALFRED and EB-Habitat respectively (Yang et al., 2025). We gather observations, actions predicted, and episode success. We use this to create labels for our loss function, where if the action predicted was valid (*last_action_success* variable in EmbodiedBench is true) or the episode was successful, we associate the image and it's visual state description (text) to a positive label, and negative otherwise. We make sure to balance out this dataset for optimal training. We then train μ with the gathered labels as ground truth, and embeddings from the MLLM backbone (Qwen or Gemma) model taking in the image and visual scene description as input. We use a Binary Cross-Entropy Loss with the Adam optimizer. Our learning rate is $1e-3$.

RL, Online: Here, we train online using REINFORCE. At each timestep, we sample a binary action from the current policy, which in this case is whether to keep or discard the memory. We then compute the cumulative reward after executing the actions predicted by the action head, and update the policy. To compute the cumulative reward, we use 1) a dense reward from the action head predicting a valid action, and 2) a sparse reward from episode success. Observe that both these rewards are not directly related to keeping or discarding memory, but are instead a result of the agent performing expected behavior to improve success. Modeling direct rewards to determine which memory to keep or discard is challenging, as they are tied to the nature of the task. For instance, an observation containing a white wall might not be useful for tasks involving picking up objects, but becomes necessary when it comes to answering questions about the environment. However, given enough training data, an agent can learn about the types of tasks being asked and corresponding memory fragments to keep in order to successfully complete them. This is analogous to a lifelong learning agent that must continually adapt to its surroundings to provide personalized assistance.

C MLLM Prompts & Decoder Modifications

We modify the base prompt provided by EmbodiedBench with stricter constraints on choosing actions by adding the following:

> ***Important Rules***:
> *The action_id must be picked from the available ids provided above.*
> *Make sure the action_id matches the corresponding action_name.*
> *A valid path is guaranteed to exist. If the image does not contain the required object for completing the task, you may*

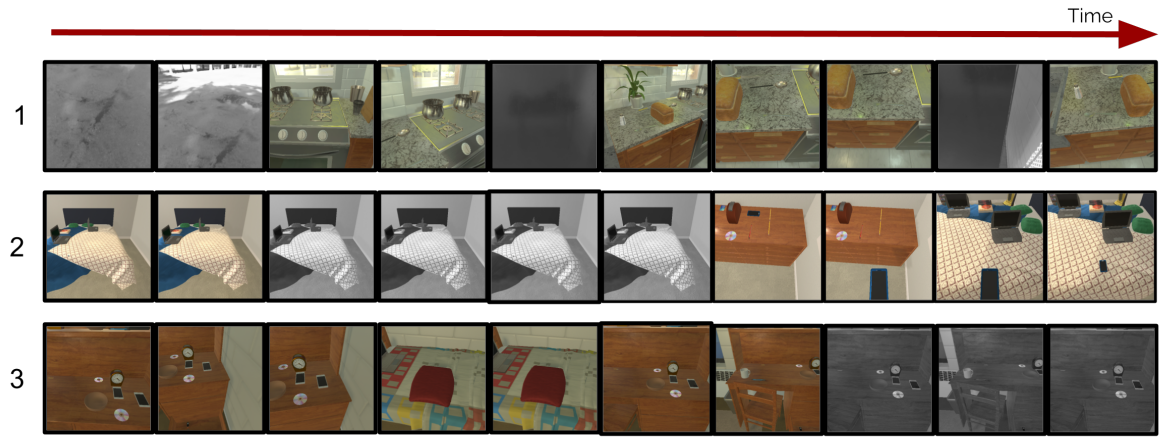

Figure 6: **Active Filtering**: The transferrable memory head performs active filtering, i.e., filters observations on the go. The grayed out images represent discarded ones. In sequence 1, notice MemCtrl filters out redundant images taken at odd angles while the agent looks around. In sequence 2, the agent makes a bunch of invalid actions in the middle of the sequence, before ultimately completing its task of placing a phone on the bed. These extra observations are filtered out as an outcome of our negative dense reward on invalid actions. In sequence 3, notice the repeating pattern towards the end, which starts to get filtered out by μ. In each of these cases, the active involvement of the MLLM in filtering allows for better write-time memory control.

have to navigate there.

We also decode by action name instead of action id, and note that this leads to a significant increase in the number of valid actions taken while planning. For instance, Qwen outputs:

> *visual_state_description: "The scene ..."*
> *reflection_and_reasoning: "The user wants ... "*
> *executable_plan: {51: "pick up the hammer", ...}*

We note that the action of "pick up the hammer" exists and is valid, but it is linked to a different action_id, 28, causing the action decoding to fail on the base EmbodiedBench code.

We speculate that this may be connected to prior literature showing that LLMs struggle with multiple choice selection (Xue et al., 2024; Zheng et al., 2024). The action selection of this work can be viewed as a multiple choice problem; therefore, some of the issues of selecting a natural language phrase with an ID can exist here as well. Furthermore, the action to ID mapping is at the beginning of the prompt, and there is research that shows that where information is positioned in the prompt affects reasoning ability (Cobbina and Zhou, 2025). Another reason could be that smaller sized LLMs are better at generating descriptive language with a larger token counts, and lack the capacity for integer action mapping requiring consolidation towards a smaller token count (Kim et al., 2024).

For this work, we modify the decoding function to map the *action name* instead of the ID and notice improved performance, especially on the Qwen model. Further work could potentially try moving the mapping information to different locations in the prompt. Another potential cause could be prompt length, where irrelevant information or even increased whitespace can degrade LLM accuracy, as shown in Du et al. (2025). Reducing the prompt to remove unnecessary whitespace or characters could potentially show improvement in outputting both the correct action name and the corresponding ID.

D Analyzing Memory

The memory head enables a form of selective memory as an alternative to passing a complete history of observations. To highlight its effectiveness, we perform an ablation with complete memory, where none of the observations are discarded, and all of them are passed back to the MLLM, while maintaining a token horizon to prevent overflow. Table 2 presents results for the complete memory case on EB-Habitat and EB-ALFRED, in comparison to μ_{RL}.

Model	EB-ALFRED						EB-Habitat					
	Avg	Base	Common	Complex	Spatial	Long	Avg	Base	Common	Complex	Spatial	Long
Qwen2.5-VL-7B-Ins	4.7	10	8	6	0	2	14.3	32	2	26	14	2
Qwen2.5-VL-7B-Ins + $\mu_{\textbf{Complete}}$	7.8	8	7	18	1	5	8.8	28	0	8	8	0
Qwen2.5-VL-7B-Ins + $\mu_{\textbf{RL}}$	**14.2**	10	13	21	3	24	**22.2**	37	3	37	16	18

Table 2: **Complete vs Selective**: We compare the performance of complete memory $\mu_{\textbf{Complete}}$ where all observations are passed as context to our best performing selective memory agent, μ_{RL}. Note the improved performance of our selective memory agent, highlighting the importance of being picky about what to store in memory, especially when model capacity is limited.

Method	EB-ALFRED						EB-Habitat					
	Avg	Base	Common	Complex	Spatial	Long	Avg	Base	Common	Complex	Spatial	Long
Memory Efficiency $\mu_{\mathcal{E}}(\%)\uparrow$												
Qwen2.5 (Baseline, No Mem.)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Qwen2.5 + μ_{RL}	39.42	35.6	42.8	39.9	38.7	40.1	27.56	39.2	13.7	15.7	22.9	46.3
Qwen2.5 + μ_{Expert}	**38.66**	37.9	40.1	45.6	36.4	33.3	**26.38**	37.2	10.2	14.8	36.5	33.2
Qwen2.5 + μ_{Complete}	100	100	100	100	100	100	100	100	100	100	100	100
Invalid Actions $\mathcal{I}\downarrow$												
Qwen2.5 (Baseline, No Mem.)	3.50	3.1	2.9	4.6	3.8	3.1	3.0	0.5	5.2	4.8	2.7	1.8
Qwen2.5 + μ_{RL}	2.22	2.0	1.8	2.9	2.1	2.3	1.36	0.4	3.0	2.4	0.6	0.3
Qwen2.5 + μ_{Expert}	**2.10**	2.7	1.5	2.4	1.8	2.1	**1.02**	0.6	2.2	1.3	0.6	0.4
Qwen2.5 + μ_{Complete}	3.10	2.7	4.2	3.3	2.3	3.0	2.12	1.5	4.6	2.8	0.8	0.9

Table 3: **Statistics**: Memory efficiency (↑) and invalid actions (↓) across all five splits for EB-Habitat and EB-Alfred. $\mu_{\mathcal{E}}$ for the expert memory head is slightly better on average, but is much worse on ALFRED. This can be attributed to tasks in ALFRED being slightly harder than Habitat. μ augmented Qwen models also make lesser number of invalid actions per episode. Overall, adding a memory head shows significant improvement over no memory and complete memory baselines.

D.1 Statistics

Table 3 highlights statistics gathered across all 5 splits on EB-Habitat and EB-Alfred for Qwen and the memory augmentations. We measure the following across 20 randomly chosen episodes:

Memory Efficiency $\mathcal{E}$: To determine the effectiveness of our approach, we compute the memory efficiency per episode as,

$$\mathcal{E} = 1 - \frac{\text{Number of Memories Kept}}{\text{Total Steps Taken}}$$

This gives us the fraction of memories that were stored in the memory bank per episode. When all the memories have been store, $\mathcal{E}$ resolves to 0, meaning the agent was inefficient in its memory management.

Invalid Actions $\mathcal{I}$: This is the average number of times that the MLLM responds with an invalid action for execution. For instance, the MLLM asks the agent to *'pick up a spoon'*, but there is no spoon visible in the observation.

Inferences: The table highlights the improved memory efficiency of both our models. In our main results table, we noted the improved performance of the online RL memory head on Qwen. By multiplying values from that table with the efficiency values here, we get a *weighted efficiency* score that aims to capture both success and frugality of memory usage. This can be written as,

$$\mathcal{W}^{(m)_b} = \text{Succ.}^{(m)} \cdot \left(1 - \frac{\mu_{\mathcal{E}}^{(m)}}{100}\right),$$

where $\text{Succ}^{(m)}$ is the task success (in %) of method m on benchmark $b \in \{\text{Alfred, Habitat}\}$, and $\mu_{\mathcal{E}}^{(m)}$ is the corresponding memory efficiency. We then aggregate this into a single score per method as,

$$\mathcal{W}^{(m)} = \frac{\mathcal{W}_{Alf.}^{(m)} + \mathcal{W}_{Hab.}^{(m)}}{2}$$

Substituting the values from Table 1 in the main text and Table 2 in this supplementary, we get $\mathcal{W}^{\text{Exp.}} = 12.13$ and $\mathcal{W}^{\text{RL}} = 10.95$, meaning that the expert has slightly better overall weighted efficiency.

We also note that the invalid actions $\mathcal{I}$ are significantly lower on both our memory heads when compared to the baseline and complete memory approaches. This further highlights the effectiveness of selective memory where μ actively decides to populate the context for the MLLM on the go.

Across the 3 sequences in Figure 6, note how our approach actively filters redundant and repeated observations. A trained memory head can

be transferred to novel settings, learning to distinguish between interesting and non-interesting observational data (Gardezi et al., 2021).