ReLIC: A recipe for 64k steps In-Context Reinforcement Learning for Embodied AI

Ahmad Elawady **Gunjan Chhablani** **Ram Ramrakhya** **Karmesh Yadav**
Dhruv Batra **Zsolt Kira** **Andrew Szot**
Georgia Tech

Abstract

Intelligent embodied agents need to quickly adapt to new scenarios by integrating long histories of experience into decision-making. For instance, a robot in an unfamiliar house initially wouldn't know the locations of objects needed for tasks and might perform inefficiently. However, as it gathers more experience, it should learn the layout and remember where objects are, allowing it to complete new tasks more efficiently. To enable such rapid adaptation to new tasks, we present ReLIC, a new approach for in-context reinforcement learning (RL) for embodied agents. With ReLIC, agents are capable of adapting to new environments using 64,000 steps of in-context experience with full attention mechanism while being trained through self-generated experience via RL. We achieve this by proposing a novel policy update scheme for on-policy RL called "partial updates" as well as a Sink-KV mechanism which enables effective utilization of long observation history for embodied agents. Our method outperforms a variety of meta-RL baselines in adapting to unseen houses in an embodied multi-object navigation task. In addition, we find that ReLIC is capable of few-shot imitation learning despite never being trained with expert demonstrations. We also provide a comprehensive analysis ReLIC, highlighting that the combination of large-scale RL training, the proposed partial updates scheme, and the Sink-KV are essential for effective in-context learning.

1 Introduction

A desired capability of intelligent embodied agents is to rapidly adapt to new scenarios through experience. An essential requirement for this capability is integrating a long history of experience into decision-making to enable an agent to accumulate knowledge about the new scenario that it is encountering. For example, a robot placed in an unseen house initially has no knowledge of the home layout and where to find objects. The robot should leverage its history of experiences of completing tasks in this new home to learn the home layout details, where to find objects, and how to act to complete tasks successfully.

To achieve adaptation of decision-making to new tasks, prior work has leveraged a technique called in-context reinforcement learning (RL) where an agent is trained with RL to utilize past experience in an environment [1–5]. By using sequence models over a history of interactions in an environment, these methods adapt to new scenarios by conditioning policy actions on this context of interaction history without updating the policy parameters. While in-context RL has demonstrated the ability to scale to a context length of a few thousand agent steps [2, 4], this falls short of the needs of embodied AI where single tasks by themselves can span thousands of steps [6]. As a result, the agent cannot learn from multiple task examples because the context required for multiple tasks cannot be accommodated within the policy context. Furthermore, prior work typically focuses on non-visual tasks [4, 5, 7], where larger histories are easier to incorporate due to the compact state representation.

Preprint. Under review.

In this work, we propose a new algorithm for in-context RL, which enables effectively utilizing and scaling to 64,000 steps of in-context experience in partially observable, visual navigation tasks. Our proposed method Reinforcement Learning In Context (ReLIC), achieves this by leveraging a novel update and data collection technique for training with long training contexts in on-policy RL. Specifically, using a long context for existing RL algorithms is prohibitively sample inefficient, as the agent must collect an entire long context of experience before updating the policy. In addition, the agent struggles to utilize the experience from long context windows due to the challenge of learning long-horizon credit assignment and high-dimensional visual observations. To address this problem, we introduce "partial updates" where the policy is updated multiple times within a long context rollout over increasing context window lengths. We also introduce Sink-KV to further increase context utilization by enabling more flexible attention over long sequences by adding learnable *sink key and value* vectors to each attention layer. These learned vectors are prepended to the input's keys and values in the attention operation. Sink-KV stabilizes training by enabling the agent to not attend to low information observation sequences.

We test ReLIC in a challenging indoor navigation task where an agent in an unseen house operating only from egocentric RGB perception must navigate to up to 80 *small* objects in a row, which spans tens of thousands steps of interactions. ReLIC is able to rapidly in-context learn to improve with subsequent experience, whereas state-of-the-art in-context RL baselines struggle to perform any in-context adaptation. We empirically demonstrate that partial updates and Sink-KV are necessary components of ReLIC. We also show it is possible to train ReLIC with 64k context. Surprisingly, we show ReLIC exhibits emergent few-shot imitation learning and can learn to complete new tasks from several expert demonstrations, despite only being trained with RL and never seeing expert demonstrations (which vary in distribution from self-generated experiences) during training. We find that ReLIC can use only a few demonstrations to outperform self-directed exploration alone. In summary, our contributions are:

1. We propose ReLIC for scaling in-context learning for online RL, which adds two novel components of partial updates and Sink-KV. We empirically demonstrate that this enables in-context adaptation of over 64k steps of experience in visual, partially observable embodied AI problems, whereas baselines do not improve with more experience.
2. We demonstrate ReLIC is capable of few-shot imitation learning despite only being trained with self-generated experience from RL.
3. We empirically analyze which aspects of ReLIC are important for in-context learning and find that sufficient RL training scale, partial updates, and the Sink-KV modification are all critical.

2 Related Work

Meta RL. Prior work has explored how agents can learn to quickly adapt to new scenarios through experience. Meta-RL deals with how agents can learn via RL to quickly adapt to new scenarios such as new environment dynamics, layouts, or task specifications. Since Meta-RL is a large space, we only focus on the most relevant Meta-RL variants and refer the readers to Beck et al. [8] for a complete survey of Meta-RL. Some Meta-RL works explicitly condition the policy on a representation of the task and adapt by inferring this representation in the new setting [9–11]. "In-context RL" is a Meta-RL paragidm where the policies implicitly infer the context by taking an entire history of interactions as input. Our work fits under "in-context RL". RL^2 [3] trains an RNN that operates over a sequence of episodes with RL and the agent implicitly learns to adapt based on the RNN hidden state. Other works leverage transformers for this in-context adaptation [2, 5, 12]. Most similar to our work is AMAGO [4], an algorithm for in-context learning through off-policy RL. AMAGO modifies a standard transformer with off-policy loss to make it better suited for long-context learning, with changes consisting of: a shared actor and critic network, using Leaky ReLU activations, and learning over multiple discount factors. Our work does not require these modifications, instead leveraging standard transformer architectures, and proposes a novel update scheme and Sink-KV for scaling the context length with on-policy RL. Empirically, we demonstrate our method scaling to $8\times$ longer context length and on visual tasks, whereas AMAGO focuses primarily on state-based tasks.

Scaling context length. Another related area of research is the efforts to scale the context length of transformers. Prior work [13–15] extend the context length using a compressed representation of the old context, either as a recurrent memory or a specialized token. Other work address the memory and computational inefficiencies of the attention method by approximating it [16, 17] or by doing

system-level optimization [18]. Another direction is context extrapolation at inference time either by changing the position encoding [19, 20] or by introducing attention sink [21]. Our work utilizes the system-level optimized attention [18] and extend the attention sinks to fit the Embodied AI settings.

Embodied AI. Prior work in Embodied AI has primarily concentrated on the single episode evaluation setting, where an agent is randomly initialized in the environment at the beginning of each episode and is tasked with taking the shortest exploratory path to a single goal specified in every episode [22, 23]. In contrast, [24] introduced the multi-ON benchmark, which extends the complexity of the original task by requiring the agent to navigate to a series of goal objects in a specified order within a single episode. Here, the agent must utilize information acquired during its journey to previous goals to navigate more efficiently to subsequent locations. Go to anything (GOAT) [25], extended this to the multi-modal goal setting, providing a mix of image, language, or category goals as input. In comparison, we consider a multi-episodic setting where the agent is randomly instantiated in the environment after a successful or failed trial but has access to the prior episode history.

3 Method

We introduce Reinforcement Learning In Context (ReLIC) which enables agents to in-context adapt to new episodes without any re-training. ReLIC is built using a transformer policy architecture that operates over a long sequence of multi-episode observations and is trained with online RL. The novelty of ReLIC is changing the base RL algorithm to more frequently update the policy with increasingly longer contexts within a policy rollout and adding Sink-KV to give the model the ability to avoid attending to low-information context. In Section 3.1, we provide the general problem setting of adapting to new episodes. Section 3.2 details the transformer policy architecture. Section 3.3 describes the novel update scheme of ReLIC. Finally, Section 3.4 goes over specific implementation details.

3.1 Problem Setting

We study the problem of adaptation to new scenarios in the formalism of meta-RL [8]. We have a distribution of training POMDPs $\mathcal{M}_i \sim p(\mathcal{M})$, where each $\mathcal{M}_i$ is defined by tuple $\left(\mathcal{S}_i, \mathcal{S}_i^0, \mathcal{O}_i, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}_i\right)$ for observations $\mathcal{O}_i$, states $\mathcal{S}_i$ which are not revealed to the agent, starting state distribution $\mathcal{S}_i^0$, action space $\mathcal{A}$, transition function $\mathcal{T}$, discount factor γ, and reward $\mathcal{R}_i$. In our setting, the states, observations, and reward vary per POMDP, while the action space, discount factor, and transition function is shared between all POMDPs.

From a starting state $s_0 \sim \mathcal{S}_i^0$, a policy π, mapping observations to a distribution over actions, is rolled out for an *episode* which is a sequence of interactions until a maximum number of timesteps, or a stopping criteria. We refer to a *trial* as a sequence of episodes within a particular $\mathcal{M}_i$. The objective is to learn a policy π that maximizes the expected return of an episode. At test-time the agent is evaluated on a set of holdout POMDPs.

3.2 ReLIC Policy Architecture

Similar to prior work [2, 4], our method ReLIC implements in-context RL via a transformer sequence model that operates over a history of interactions spanning multiple episodes. At step t within a trial, ReLIC predicts current action a_t based on the entire sequence of previous observations $o_1, \ldots, o_t$ which may span multiple episodes. In the embodied AI settings we study, the observation o_t consists of an egocentric RGB observation from the robot's head camera along with proprioceptive information and a specification of the current goal. Each of these observation components are encoded using a separate observation encoding network, and the embeddings are concatenated to form a single observation embedding e_t. A causal transformer network [26] h_θ inputs the sequence of embeddings $h_\theta(e_1, \ldots, e_t)$. A linear layer that predicts the actions from the transformer output.

The transformer model h_θ thus bears the responsibility of in-context learning by leveraging associations between observations within a trial. This burden especially poses a challenge in our setting of embodied AI for the transformer attending over a history of thousands of egocentric visual observations. Subsequent visual observations are highly correlated, as the agent only takes one action between observations. Knowing which observations are relevant to attend to in deciding the current action is thus a challenging problem. In this work, we build our architecture around full attention

transformers using the same architecture as the LLaMA language model [27], but modify the number of layers and hidden dimension size to appropriately reduce the parameter count for our setting. We then propose methods for scaling in-context RL to long contexts of visual observations.

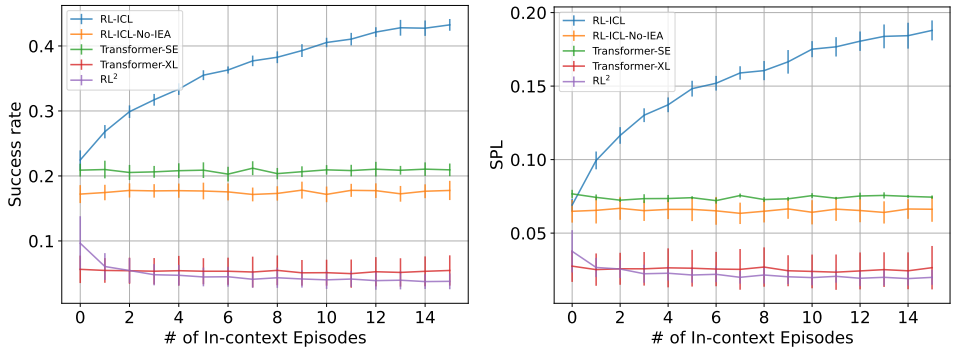
We also introduce an architectural modification to the transformer called **Sink-KV** to improve the transformer's ability to attend over a long history of visual experience from an embodied agent. Building off the intuition that learning to attend over a long sequence of visual observations is challenging, we introduce additional flexibility into the core attention operation by prepending the key and value vectors with a per-layer learnable sequence of *sink KV vectors*. Specifically, recall that for an input sequence $X \in \mathbb{R}^{n \times d}$ of n inputs of embedding dimension d, the attention operator projects X to keys, queries and values notated as K, Q, V respectively and all elements of $\mathbb{R}^{n \times d}$ where we assume all hidden dimensions are d for simplicity. The standard attention operation computes $\text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$. We modify calculating the attention scores by introducing learnable vectors $K_s, V_s \in \mathbb{R}^{s \times d}$ where s is the specified number of "sinks". We then prepend K_s, V_s to the K, V of the input sequence before calculating the attention. Note that the output of the attention operation is still $n \times d$, as in the regular attention operation, as the query vector has no added component. We repeat this process for each attention layer of the transformer, introducing a new K_s, V_s in each attention operation.

Sink-KV gives the sequence model more flexibility on how to attend over the input. Prior works observe that due to the softmax in the attention, the model is forced to attend to at least one token from the input [21, 28]. Sink-KV removes this requirement by adding the learnable vectors to the key and value. In sequences of embodied visual experiences, this is important as attention heads can avoid attending over any inputs when there is no new visual information in the current observations. Adding this flexibility into the attention helps the agent operate over longer sequence lengths.

3.3 ReLIC Learning

ReLIC is updated through online RL, namely PPO [29]. However, for the agent to be able to leverage a long context window for in-context RL, it must also be trained with this long context window. PPO collects a batch of data for learning by "rolling out" the current policy for a sequence of T interactions in an environment. To operate on a long context window spanning an entire trial, the agent must collect a rollout of data that consists of this entire trial. This is challenging because, in the embodied tasks we consider, we seek to train agents on trials lasting over 64k steps, which consists of at least 130 episodes. As typical with PPO, to speed up data collection and increase the update batch size we use multiple environment workers each running a simulation instance that the policy interacts with in parallel. With 32 environment workers, this corresponds to $\approx 130k$ environment steps between every policy update. PPO policies trained in common embodied AI tasks, such as ObjectNav, have only 128 steps between updates and require $\approx 50k$ updates to converge (for 32 environment workers, 128 steps per worker between update and 200M environment steps required for convergence) [23]. Executing a similar number of updates would require ReLIC to collect ≈ 6 billion environment interactions.

ReLIC addresses this problem of sample inefficiency by introducing a *partial update scheme* where the policy is updated multiple times throughout a rollout. First, at the start of a rollout of length T, all environment workers are reset to the start of a new episode. Define the number of partial updates as K. At step $i \in [0, T]$ in the rollout, the policy is operating with a context length of $i - 1$ previous observations to determine the action at step i. Every T/K samples in the rollout, we update the policy. Therefore, at update N within the rollout, the agent has collected NT/K of the T samples in the rollout. The agent is updated using a context window of size NT/K, however, the PPO loss is only applied to the final T/K outputs. The policy is changing every T/K samples in the rollout, so the policy forward pass must be recalculated for the entire NT/K window rather than caching the previous $(N-1)T/K$ activations. In the last update in the rollout, after collecting the last T/K steps, we update the policy with the loss applied on all steps in the rollout. We refer to this step as *full update*. At the start of a new rollout, the context window is cleared and the environment workers again reset to new episodes.



(a) Success Rate (b) Efficiency

Figure 1: Comparing the in-context learning capability of ReLIC and baselines on EXTOBJNAV. The number of episodes in the trial is displayed on the x-axis. The y-axis displays the success or efficiency at that episode count. Agents capable of in-context learning will increase in success and efficiency when encountering more episodes. Each method is run for 3 random seeds and evaluated on 10k distinct sequences. Error bars are standard deviations over trial outcomes between the 3 seeds.

3.4 Implementation Details

The transformer is modeled after the LLaMA transformer architecture [27] initialized from scratch. We only update the parameters of the transformer and projection layers while freezing the visual encoder since prior work shows this is an effective strategy for embodied AI [30, 31]. For faster policy data collection, we store the transformer KV cache between rollout steps. To fit long context during the training in limited size memory, we used low-precision rollout storage, gradient accumulation [32] and flash-attention [18]. After each policy update, we shuffle the older episodes in the each sequence and update the KV-cache. Shuffling the episode serves as regularization technique since the agent sees the same task for a long time. It also reflects the lack of assumptions about the order of episodes, an episode should provide the same information regardless of whether the agent experiences it at the beginning or at the end of the trial.

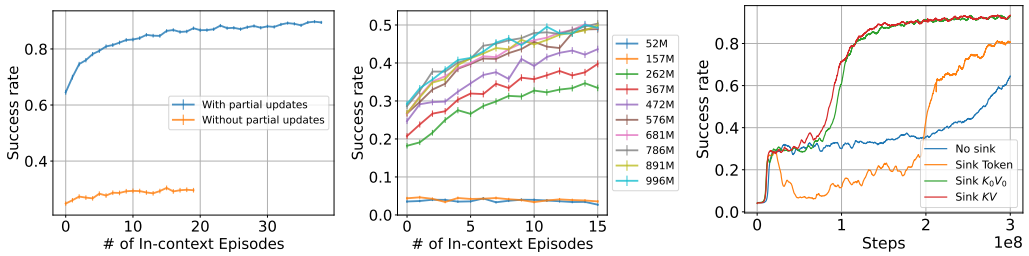
We use the VC-1 visual encoder and with the ViT-B size [31]. We found the starting VC-1 weights performed poorly at detecting small objects, which is need for the embodied AI tasks we consider. We therefore finetuned VC-1 on a small objects classification task. All baselines use this finetuned version of VC-1. We provide more details about this VC-1 finetuning in Appendix D and details about all hyperparameters in Appendix B.2.

4 Experiments

We first introduce the Extended Object Navigation (EXTOBJNAV) task we use to study in-context learning for embodied navigation. Next, we analyze how ReLIC enables in-context learning on this task and outperforms prior work and baselines. We then analyze ablations of ReLIC and analyze its behaviors. Finally, we show ReLIC is capable of few-shot imitation learning.

4.1 EXTOBJNAV: Extended Object Navigation

To evaluate ICL capabilities for embodied agents, we introduce EXTOBJNAV, an extension of the existing Object Navigation (OBJECTNAV) benchmark. EXTOBJNAV assesses an agent's ability to find a sequence of objects in a house while operating from egocentric visual perception. For each object, the agent is randomly placed in a house and must locate and navigate to a specified object category. The agent used is a Fetch robot equipped with a 256×256 RGB head camera. Additionally, the agent possesses an odometry sensor to measure its relative displacement from the start of the episode. Navigation within the environment is executed through discrete actions: move forward 0.25 meters, turn left or right by 30 degrees, and tilt the camera up and down by 30 degrees. The agent also has a stop action, which ends the episode.



(a) Update Scheme Ablation (b) Learning vs. ICL (c) Attention Sink Ablation

Figure 2: Ablations and analysis for ReLIC ICL capabilities. Fig. 2a shows a high number of updates per rollout and overlapping data batches within the rollout are crucial for ICL over long trials. Fig. 2b shows scaling RL training results in not only agents that are more proficient at finding objects, but also agents with stronger ICL capabilities. Fig. 2c shows learning curves for different attention sinks on ReplicaCAD. The Sink KV and Sink K_0V_0 [28] is stable and sample-efficient. However, the learning is slow without attention sinks and unstable with Sink Token [21]. The error bars represent the standard error.

EXTOBJNAV uses scenes from the Habitat Synthetic Scenes Dataset (HSSD) [33] along with a subset of the YCB object dataset [34] containing 20 objects types. Note that EXTOBJNAV requires navigating to *small* objects unlike other OBJECTNAV variants [35] that uses large receptacles as goals. This allows us to increase the dataset diversity by sampling objects randomly in the environment, unlike OBJECTNAV, where the receptacles are fixed parts of the scanned meshes. The random sampling also precludes the agent from using priors over object placements in scenes, forcing it to rely on the experience in its context.

EXTOBJNAV defines a *trial* as a sequence of episodes within a fixed home layout, where a home layout is defined by a combination of a floorplan, a furniture layout and set of object placements. A home layout contains on an average 22 objects where multiple object instances may be of the target category. Within an episode in the trial, a target object category is randomly selected and the agent is randomly placed in the house. The episode is successful if the agent calls the stop action within 2 meters of the object, with at least 10 pixels of the object in the current view. If the object is not found within 500 steps the episode counts as a failure.

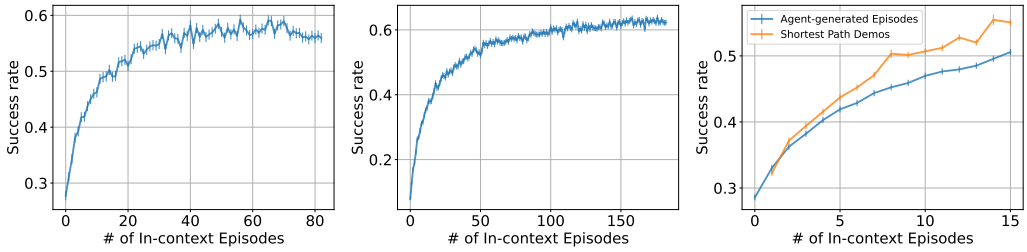
We evaluate the agents on unseen scenes from HSSD and report the success rate (SR) and Success-weighted by Path Length (SPL) [36] metrics. Specifically, we look at the SR and SPL of an agent as it accumulates more episodes in-context. Ideally, with more in-context episodes within a home layout, it should be more adept at finding objects and its SR and SPL should improve. See Appendix A for further details on the EXTOBJNAV.

4.2 In-Context Learning on EXTOBJNAV

In this section, we compare the ability of ReLIC and baselines to in-context learn in a new home layout. We compare ReLIC to the following baselines:

- **RL**^{2} [3]: Use an LSTM and keep the hidden state between episode resets within a trial.
- **Transformer-XL (TrXL)** [13]: Use Transformer-XL and updates the constant-size memory recurrently. This is the model used in [2] trained in our setting. Following [2] we use PreNorm [37] and gating in the feedforward layers [38].
- **ReLIC-No-IEA**: ReLIC without Inter-Episode Attention (IEA). Everything else, including the update scheme is the same as ReLIC.
- **Transformer-SE**: A transformer-based policy operating over only a single episode (SE) and without the update schemes from ReLIC.

All baselines are trained for 500M steps using a distributed version of PPO[22]. Methods that utilize multi-episode context are trained with a context length of 4k, and use 8k context length during inference (unless mentioned otherwise, e.g. in our long-context experiments). The results in Figure 1 demonstrate ReLIC achieves better performance than baselines on 8k steps of ICL, achieving 43% success rate v.s. 22% success rate achieved by the closest performing baseline (Transformer-SE).



(a) Context length generalization (b) 64k Train Context Length (c) Few-Shot Imitation Learning

Figure 3: (a) ReLIC trained with context length 4k (1B steps) generalizes to operating at 32k steps of in context experience in a new home layout. (b) ReLIC trained at 64k context length (500M steps) shows ICL abilities over 175 episodes. (c) ReLIC can do few-shot imitation learning despite not training for it. The error bars represent the standard error.

Additionally, ReLIC is able to effectively adapt to new home layouts throughout the course of the trial. In the first episode of the trial, transformer-based baseline methods attain a similar base performance of around 20% success rate. However, as more episodes arrive, the performance of ReLIC increases. The recurrent models, Transformer-XL and RL^2, have lower base performance at 10% success rate and show no in-context learning. The performance of RL^2 degrades with more in-context episodes, which is aligned with the inability of the LSTM to model long sequences.

The failure to in-context learn in the recurrent models can be explained by the bottleneck in the recurrent memory and the lack of gradient between the current observations and in-context observations. In order for the recurrent memory to be useful in a new episode, it needs to have a useful representation of the in-context episodes for all spawning positions and target object types since the agent is spawned in a random position and tasked to reach a random object type in each episode, which is an easier task with direct access to observations as in the Transformer. In the absence of a gradient flow between the current observations and the in-context observations, recurrent models are unable to optimize the in-context representation for the current step which is not the case in the Transformer.

After 15 episodes of in-context experience, the success rate of ReLIC increases from 23% to 43%. The baselines do not possess this same ICL ability and maintain constant performance with subsequent in-context episodes. ReLIC also in-context learns to navigate faster to objects, as measured by the gap in SPL[36]. As the trial progresses, the agent is able to more efficiently navigate to objects in the house with the SPL of ReLIC increasing from 0.07 to 0.188. The baselines are unable to improve efficiency in-context and maintain a SPL of 0.025 to 0.075 throughout the entire trial.

4.3 ReLIC Ablations and Analysis

We demonstrate that the partial updates in ReLIC and Sink-KV are crucial to learning with RL over long context windows and acquiring ICL capabilities. We then show that ICL emerges later in the training and ReLIC works with large context.

No Partial Updates. [1] Firstly, we switch off the partial updates in ReLIC and find that it performs poorly without the partial updates (Figure 2a), achieving 40% lesser SR at the first episode. This model also shows little ICL abilities with the SR only increasing 5% by the end of the trial versus a 25% increase when using partial updates.

Sink KV.[1] Next, we demonstrate that using Sink KV is necessary for sample-efficient in-context RL learning. We trained the model on ReplicaCAD with and without attention sinks. The learning curves in Figure 2c shows that learning is more stable and faster with Sink KV which achieves 90% success rate at 200M steps. It also shows that Sink KV performs similar to Softmax One [28], referred to as Sink K_0V_0. Without attention sink mechanisms, learning is slow and achieves less than 40% success rate after 200M steps and reaching 64% at 300M steps. Using sink token [21], the training becomes unstable, achieving 40% success rate at 200M steps training and reaching 80% success rate at 300M

[1]To simplify the analysis, we run this expeirment using the ReplicaCAD [6] scenes rather than HSSD because training on HSSD is compute intensive. The smaller scenes make the task easier, and methods faster to train, but other details of the task remain the same.

steps. The details of the different sink attentions, their implementations and how the attention heads use the Sink-KV can be found in Appendix E.

Training Steps v.s. ICL Abilities. We find that ReLIC only acquires ICL capabilities after sufficient RL training. As demonstrated in Figure 2b, the agent is only capable of ICL after 157M steps of training. Models trained for 52M and 157M remain at constant success with more in-context experience. Further training does more than just increase the base agent performance in the first episode of the trial. From 262M steps to 367M steps, the agent base performance increases by 2%, yet the performance after 15 episodes of ICL performance increases 10%. This demonstrates that further training is not only improving the base capabilities of the agent to find objects, but also improving the agent's ability to utilize its context across long trials spanning many episodes.

Context length generalization. Next, we push the abilities of ReLIC to in-context learn over contexts much larger than what is seen during training. In this experiment, we evaluate ReLIC model, trained with 4k context length, on 32k steps of experience, which is enough to fit 80 episode trials in context. Assuming that the simulator is operating at 10Hz, this is almost 1 of agent experience within the context window. Note that for this experiment, we use our best checkpoint, which is trained for 1B steps. The results demonstrate that ReLIC can generalize to contexts $8\times$ larger at inference. Figure 3a shows ReLIC is able to further increase the success rate to over 55% after 80 in context episodes and consistently maintains performance above 50% after 20 in-context episodes.

64k steps trials. Finally, we investigate scaling *training* ReLIC with 64k context length. We use the same hyperparameters as Section 4.2, but increase the number of partial updates per rollout such that the policy is updated every 256 steps, the same number of steps used in ReLIC. We train the model for only 500M steps due to the slower training speed. Fig. 3b shows that the model can in-context learn over 175 episode and continue to improve success rate. More details are available at Appendix C.2.

4.4 Emergent Few-Shot Imitation Learning

In addition to learning in-context from self-generated experience in an environment, ReLIC can also use its context to learn from demonstrations provided by an external agent or expert, despite never being trained on demonstrations and only learning from self-generated experience. We consider the setting of few-shot imitation learning [39, 40] where an agent is given a set of trajectories $\{\tau_1, \ldots, \tau_N\}$ demonstrating behavior reaching desired goals $\{g_1, \ldots, g_N\}$. The agent must then achieve a new g_{N+1} in new environment configurations using these demonstrations. ReLIC is able to few-shot imitation learn by taking the expert demonstration as input via the context. Specifically, we generate N expert shortest path trajectories navigating to random objects from random start positions in an unseen home layout[2]. These N trajectories are inserted into the context of ReLIC and the agent is instructed to navigate to a new object in the environment.

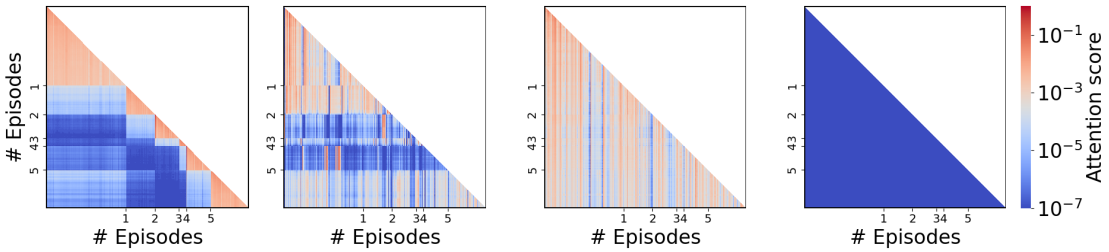
In Figure 3c we show that ReLIC can utilize these expert demonstrations despite never seeing such shortest paths during training. Figure 3c shows the success rate of ReLIC in a single episode after conditioning on some number of shortest path demonstrations. More demonstrations cover more of the house and the agent is able to more successfully navigate to objects. We also compare to the success rate of an agent that has N episodes of experience in the house as opposed to N demonstrations. Using the demonstrations results in better performance with 5% higher success rate for $N = 16$.

4.5 Qualitative Results

In this section, we show that the agent is able to utilize the in-context information by inspecting the attention scores patterns in the attention heads. We generate the data by letting the agent interact with an unseen environment for 19 episodes which produced a sequence of 2455 steps. A random object type is selected as a target in each episode. By inspecting the attention scores of the attention heads, we found 4 patterns shown in Figure 4.

- **Intra-episode attention**: In this pattern, the agent attends only to the running episode, Figure 4a.

[2]The success rate of the shortest path demos is 80% since it's sometime challenging to get a view where the target object is visible. Some objects are obscured behind other objects, inside a drawer or on a chair and only visible from a specific angle.



(a) Intra-episode attn (b) Inter-episodes attn (c) Episode-invariant attn (d) Zero attn

Figure 4: Attention scores patterns of a sequence with 1024 steps. We found 4 attention patterns in the heads of a trained policy: (a) Intra-episode attention where the attention head assigns high score to the running episode, (b) Inter-episode attention pattern where the attention head assigns high score to the context, without being constrained to the running episode, (c) the episode-invariant pattern where the attention head attends to the same tokens regardless of the episode structure in the context, and (d) the zero attention pattern where the attention head assign all attention scores to the Sink-KV.

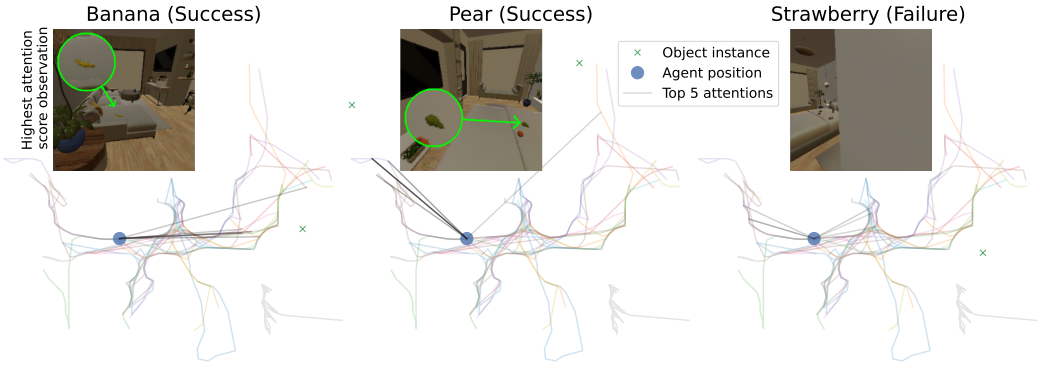


Figure 5: Visualization of an inter-episode attention head, see Section 4.5. The colored curves are the trajectories of previous episodes. The blue circle is the agent's position. The green Xs are the instances of the target object type. The black lines represent the agent's attention when the target is the object type mentioned above the image. The lines connect the agent with the point in history that it attends to, the opacity of the line represents the attention score. The overlaid image is visual observation with the highest attention score.

- **Inter-episodes attention**: Inter-episodes attention is where the agent accesses the information from previous episodes, Figure 4b.
- **Episode-invariant attention**: The agent is able to attend to certain tokens which do not change on changing the episode, Figure 4c.
- **Zero attention**: Some heads have 0 attention scores for all tokens which would not be possible with the vanilla attention.

We further analyze the attention pattern between episodes. We collect 2455 steps in a trial and then probe the agent's attention scores by querying each object type by adding a new observation with the desired object type at the final step. Figure 5 shows that the agent is able to recall multiple instances of the target object types in its history. Further analysis is in Appendix F.

5 Conclusion and Limitations

The ability of an agent to rapidly adapt to new environments is crucial for successful Embodied AI tasks. We introduced ReLIC, an in-context RL method that enables the agent to adapt to new environments by in-context learning with up to 64k environment interactions and visual observations. We studied the two main components of ReLIC: *partial updates* and the *Sink-KV* and showed both are necessary for achieving such in-context learning. We showed that ReLIC results in significantly better performance on a challenging long-sequence visual task compared to the baselines.

Limitations of the approach are that we found for ICL to emerge, it requires a diverse training dataset on which the model can not overfit. There is no incentive for the model to learn to use the context if

it can overfit the task. We were able to address that in the dataset generation by creating different object arrangements for each scene which made it challenging for the model to memorize the objects arrangements. Another is that our study only focuses on the embodied AI navigation task. Future work can explore this same study in more varied environments such as a mobile manipulation task where an agent needs to rearrange objects throughout the scene.

References

[1] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[2] Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

[3] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[4] Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. *arXiv preprint arXiv:2310.09971*, 2023.

[5] Luckeciano C Melo. Transformers are meta-reinforcement learners. In *International Conference on Machine Learning*, pages 15340–15359. PMLR, 2022.

[6] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34, 2021.

[7] Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *arXiv preprint arXiv:2307.03864*, 2023.

[8] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

[9] Tony Z Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. Meld: Meta-reinforcement learning from images via latent state models. *arXiv preprint arXiv:2010.13957*, 2020.

[10] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2):2950–2957, 2020.

[11] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[12] Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

[13] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[14] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention, 2024.

[15] Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*, 2024.

[16] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[17] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[18] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023.

[19] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[20] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation, 2022.

[21] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[22] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[23] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. *arXiv preprint arXiv:2204.13226*, 2022.

[24] Saim Wani, Shivansh Patel, Unnat Jain, Angel Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *NeurIPS*, 2020.

[25] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, et al. Goat: Go to any thing. *arXiv preprint arXiv:2311.06430*, 2023.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[27] 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. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[28] Evan Miller. Attention Is Off By One. https://www.evanmiller.org/attention-is-off-by-one.html, 2023. [Online; accessed 11-May-2024].

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

[30] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *CVPR*, 2022.

[31] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? *arXiv preprint arXiv:2303.18240*, 2023.

[32] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019.

[33] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Schacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. *arXiv preprint arXiv:2306.11290*, 2023.

[34] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[35] Habitat Team. Habitat CVPR challenge, 2021. URL https://aihabitat.org/challenge/2021/.

[36] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[37] Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning, 2019.

[38] Noam Shazeer. Glu variants improve transformer, 2020.

[39] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

[40] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[41] Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.

[42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

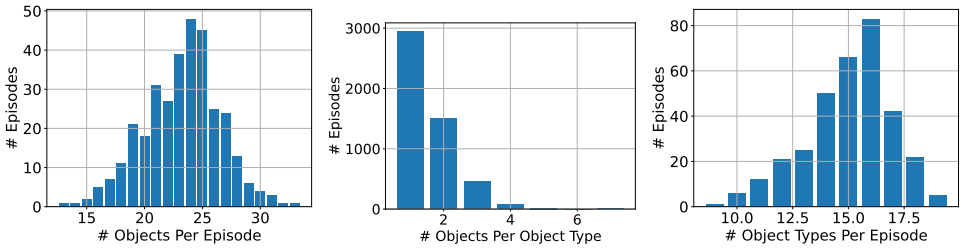
[43] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[44] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[45] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016.

[46] Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang, Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, Vladimir Vondrus, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

[47] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. *Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations*, page 240–248. Springer International Publishing, 2017. ISBN 9783319675589. doi: 10.1007/978-3-319-67558-9_28. URL http://dx.doi.org/10.1007/978-3-319-67558-9_28.



(a) # Objects Per Episode (b) # Objects Per Object Type (c) # Object Types Per Episode

Figure 6: The distribution of the objects and object types in the data.

A Additional EXTOBJNAV Details

The EXTOBJNAV is a small object navigation task. We use the same training (37) and validation (12) scenes as [41]. The data is generated by randomly placing objects from the 20 object types, a subset of the YCB [34] dataset, on random receptacles. The data is generated by sampling between 30 to 40 object instances and placing them on receptacles, and subsequent filtering of the objects that are not reachable by the agent. The filtering is done by placing the agent in front of the object and evaluating whether the agent can meet the success criteria. If the agent can meet the critera, we retain the object. Otherwise, we discard the object. The distribution of the objects and object types are in Figure 6.

The reward function is defined as follows:

- Change in geodesic distance to the closest object $r_d = -\Delta d$ where d is the geodesic distance to the closest object. The closest object can change across the episode.
- Slack reward of -0.001.
- Succes reward of 2.

The episode is considered a success if the agent selects the *Stop* action while it is within 2 meters of an instance of the target type and has 10 pixels of this instance in the view.

B Additional Method Details

B.1 Model training

In this section, we discuss the training setup for ReLIC experiment.

The workers and the batch size. We use 20 environment workers per GPU. Since we use 4 GPUs in parallel, there are 80 environment workers in total. The micro batch size is 1 and we accumulate the gradient for 10 micro batches on the 4 GPUs which makes the effective batch size 40.

RL algorithm. We use PPO [29] to train the model with $\gamma = 0.99$, $\tau = 0.95$, entropy coefficient of 0.1 and value loss coefficient of 0.5.

Optimizer. We use Adam [42] optimizer to learn the parameters.

Learning rate schedule. We use learning rate warm up in the first 100,000 environment interactions. The learning rate starts with $LR_0 = 2e-7$ and reaches $LR = 2e-4$ at the end of the warm up. Cosine decay [43] is used after the warm-up to decay the learning rate to 0 after 1B environment interactions.

Precision. We use FP16 precision for the visual encoder and keep the other components of the model as FP32.

Rollout Storage. The rollout storage size is 4096. We store the observations and the visual embeddings in rollout in low-precision storage, specifically in FP16 precision.

Regularization. We follow [44] in using depth dropout [45] with value 0.1 as regularization technique. We also shuffle the in-context episodes after each partial updates.

Hardware Resources and Training Time. The model is trained for 1B steps on 4x Nvida A40 for 12 days.

B.2 Hyperparameters

We list the hyperparameters for the different experiments discussed in section Section 4.2.

ReLIC: The hyperparameters used in ReLIC can be found in Table 2 and the hyperparameters of the transformer model used in the training can be found in Table 1.

RL^2: For implementing RL^2, we build on the default PPO-GRU baseline parameters in Habitat 3.0 [46]. We set the number of PPO update steps to 256, and the hidden size of the GRU to 512. The scene is changed every 4096 steps during training, and the hidden state is reset to zeros after every scene change.

ReLIC-No-IEA: We use the same model and ReLIC hyperparameters described in ReLIC. The only difference is that we set the attention mask to restrict the token to only access other tokens within the same episode.

Transformer-SE: We use the same model described in ReLIC. However, we limit the training sequence to a fixed size 385 old observations + 256 new observations. The choice of the old number of observations is made such that we never truncate an episode which is at most 500 steps. The attention mask is set to restrict the tokens to only access other tokens in the same episode.

Transformer-XL (TrXL) [13]: Use Transformer-XL and update the constant-size memory recurrently. We follow [2] in that we use PreNorm [37] and use gating in the feedforward layers [38]. We experiment with two values for the memory size, 256 and 1024, using TrXL without gating and found that the model is able to learn with 256 memory but is unstable with 1024 memory. We use 256 memory size which gives the agent context of size $L \times N_m = 4 \times 256 = 1024$ where L is the number of layers. Except for the memory, we use the same number of layers and heads and the same hidden dimensions as ReLIC.

Table 1: Transformer hyperparameters

Hyperparameter	Value
# Layers	4
# Heads	8
Hidden dimensions	256
MLP Hidden dimensions	1024
# Sink-KV	1
Attention sink	Sink KV_0
Episode index encoding	RoPE [19]
Within-episode position encoding	Learnable
Activation	GeLU [38]

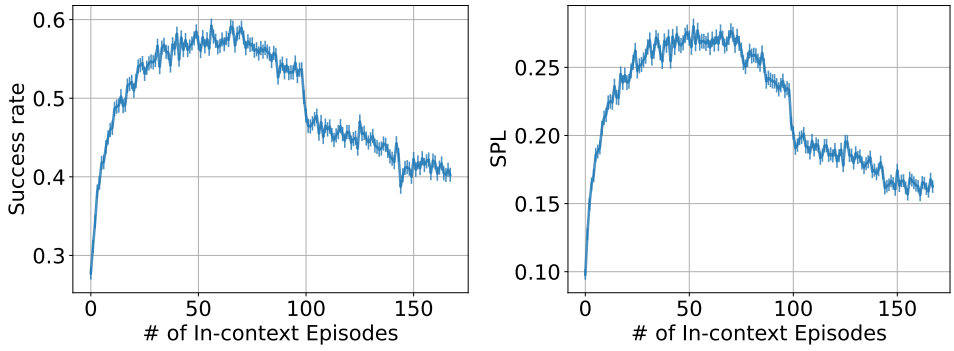
Table 2: ReLIC hyperparameters

Hyperparameter	Value
Rollout size	4096
total # updates per rollout	16
# partial updates	15
# full updates	1

C More Experiments

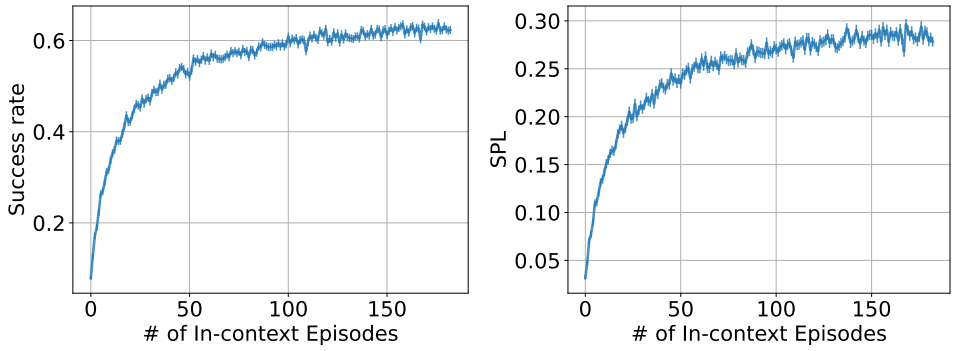
C.1 Inference time generalization

In this section, we evaluate the main experiment, which is trained with 4k sequence length, for extended sequence length. Specifically, we evaluate it on 64k sequence length. Figure 7 shows the performance of the model as the number of in-context episodes increase. The performance increases until the number of in-context episodes reach 80 episodes. This is generalization to 32k steps which is 8x the training sequence length. After 80 episodes, the performance starts to suffer as the number of episode increases. With the drop in performance, the model is still able to perform higher than the base performance, 10% higher success rate at 150 in-context episodes. We show in Appendix C.2 that ReLIC works well with 64k sequence length and shows consistently increasing performance as the number of in-context episodes increase.



(a) Success Rate (b) Efficiency

Figure 7: The result of training on 4k and evaluating on 64k sequence length using ReLIC.



(a) Success Rate (b) Efficiency

Figure 8: The result of training and evaluating on 64k sequence length using ReLIC.

C.2 Training with 64k sequence length

In the main experiment, we showed that we can train on 4k steps and inference for 32k steps. In this experiment, we show that our method ReLIC is able to train with 64k sequence length. We used the same hyperparameters in the main experiment, except the training sequence length which we set to 64k and the number of updates per rollout is increased so that we do updates every 256 steps, same as the main experiment.

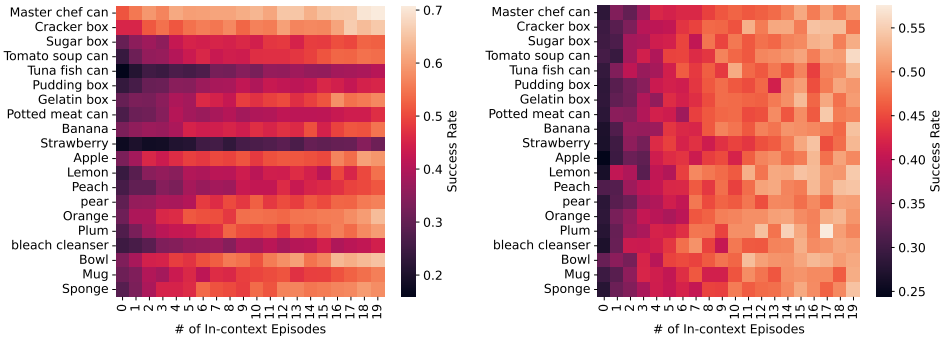
The result in Figure 8 shows that the model is able to learn and generalize on 64k sequence length.

C.3 ReLIC per Object Type

In Figure 9a, we analyze the ICL performance of ReLIC per object type. Specifically, we specify the same object type target for the agent repeatedly for 19 episodes. Similar to the main experiments, the agent is randomly spawned in the house. As Figure 9a illustrates, ReLIC becomes more capable at navigating to all object types in subsequent episodes. The agent is good at adapting to finding some objects such as bowls, cracker box, and apples. Other objects, such as strawberry and tuna fish can, remain difficult. In Figure 9b, we show that with 19 episodes of ICL, the agent is can reliably navigate to any object type in the house despite having different object types as target in the context. This demonstrates the agent is able to utilize information about other object targets from the context.

D Visual encoder finetuning

We finetuned the visual encoder on a generated supervised task before freezing it to be used in our experiments. Each sample, Figure 10, in the data is generated by placing the agent in front of a random object then the RGB sensor data is used as input X. The output y is a binary vector of size



(a) Fixed Object Type Per Trial

(b) RL-ICL per Object Type

Figure 9: Analysis of how ReLIC learns to navigate to particular object types through ICL. (a) compares the number of consecutive episodes within a trial an object appears v.s. the success rate. The agent becomes more capable at navigating to that object type for subsequent episodes. (b) shows the episode index within the trial that the object first appears v.s. the average success rate for different objects. As the agent acquires more experience in-context, it can proficiently navigate to any object type.

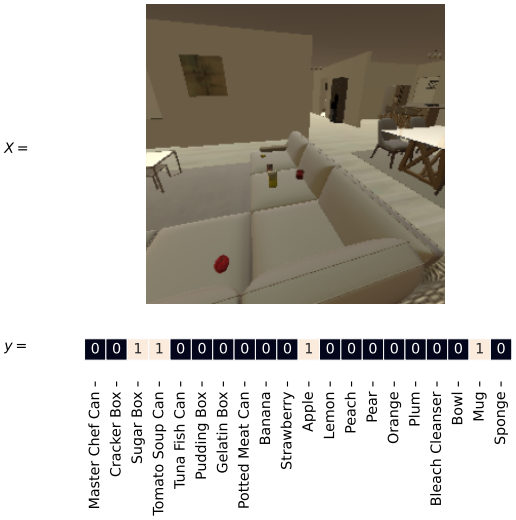


Figure 10: Sample of the finetuning data

20, the number of available object types, where each element represents whether the corresponding object type is in the image or not. The object type is considered in the image if there is an instance of this object in the image with more than 10 pixels. 21k samples are generated from the training scenes and object arrangements. The 21k samples are then split to training and validation data with ratios 90% to 10%.

The VC-1 model is finetuned using the Dice loss [47] by adding a classification head to the output of '[CLS]' token using the generated data. The classification head is first finetuned for 5 epochs with $LR = 0.001$ while the remaining of the model is frozen. Then the model is unfrozen and finetuned for 15 epochs with $LR = 0.00002$.

E Sink KV

We introduce Sink KV, a modification to the attention calculation in the attention layers. We first describe the vanilla attention [26], the issue and the motivation to find a solution. Then we discuss

the proposed solutions and introduce the Sink KV. Finally, we analyse different variants of the Sink KV.

E.1 Motivation

The vanilla attention is the component responsible for the interaction between the tokens in the sequence. The output for each token is calculated by weighting the value of all tokens. The input to the attention layer is the embeddings of the input tokens $E \in \mathbb{R}^{n \times d}$ where n is the number of input tokens and d is the dimension of the embeddings.

First the embeddings E are linearly projected to the Key K, Value V and Query Q. Then the attention scores are calculated using $S = softmax(QK^T/\sqrt{d_k})$ where d_k is the dimensions of the keys. The output A is calculated as a weighted sum of the values V, $A = SV$.

The calculation of the attentions scores S using the $Softmax$ forces the tokens to attend to values V, even if all available values do not hold any useful information, since the sum of the scores is 1 [28]. This is especially harmful in cases where the task requires exploration. As the agent explores more, a more useful information may appear in the sequence. If the agent is forced to attend to low information tokens at the beginning of the exploration, it will introduce noise to the attention layers.

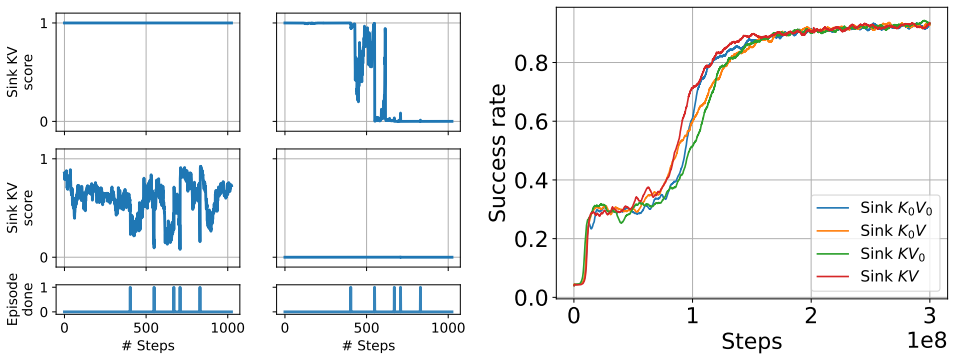
E.2 Solutions

Softmax One [28] addresses this issue by adding 1 to the denominator of the $Softmax$, $softmax_1(x_i) =: exp(x_i)/(1 + \sum_j exp(x_j))$, which is equivalent to having a token with $k = 0$ and $v = 0$. This gives the model the ability to have 0 attention score to all tokens, we refer to Softmax One as Sink K_0V_0.

Sink tokens [21] is another approach to address the same issue by prepending learnable tokens to the input tokens $E = [E_s \circ E_{input}]$ where E is the input embedding to the model and $[A \circ B]$ indicates concatenation along the sequence dimension of the A and B matrices .

Sink KV is a generalization of both approaches. It modifies the attention layer by adding a learnable Key $K_s \in \mathbb{R}^{n \times d_k}$ and values $V_s \in \mathbb{R}^{n \times d}$. In each attention layer, we simply prepend the learnable K_s and V_s to the vanilla keys K_v and values V_v to get the $K = [K_s \circ K_v]$ and $V = [V_s \circ V_v]$ used to calculate the attention scores then the attention output.

In the case $K_s = 0$ and $V_s = 0$, Sink KV becomes equivalent to Softmax One. It can also learn the same Ks and Vs corresponding to the Sink Token since our model is casual and the processing of the Sink Token is not affected by the remaining sequence.



(a) Different patterns of Sink KV scores for 1k input tokens.

(b) The learning curve for the Sink KV variants.

Figure 11: Sink KV analysis.

E.3 Sink KV variants

We tried a variant of Sink KV where the either the Value or the Key is set to 0, referred to as Sink KV_0 and Sink K_0V respectively. All variants perform similarly in terms the success rate Figure 11b.

Figure 11a shows different patterns the model uses the Sink KV_0. The model can assign all attention scores to the Sink KV_0, which yields a zero output for the attention head, or assign variable scores at different time in the generation. For example, one the attention heads is turned off during the 1st episode of the trial by assigning all attention score to the Sink KV_0 then eventually move the attention to the input tokens in the new episodes. The model is also able to ignore the Sink KV_0 by assigning it 0 attention scores as shown in the figure.

F What does the agent see?

Figure 12 shows the attention scores for all 20 object types when selected in the 1st step of a new episode after 19 episodes. See Section 4.5 for explanation on how the result is generated.

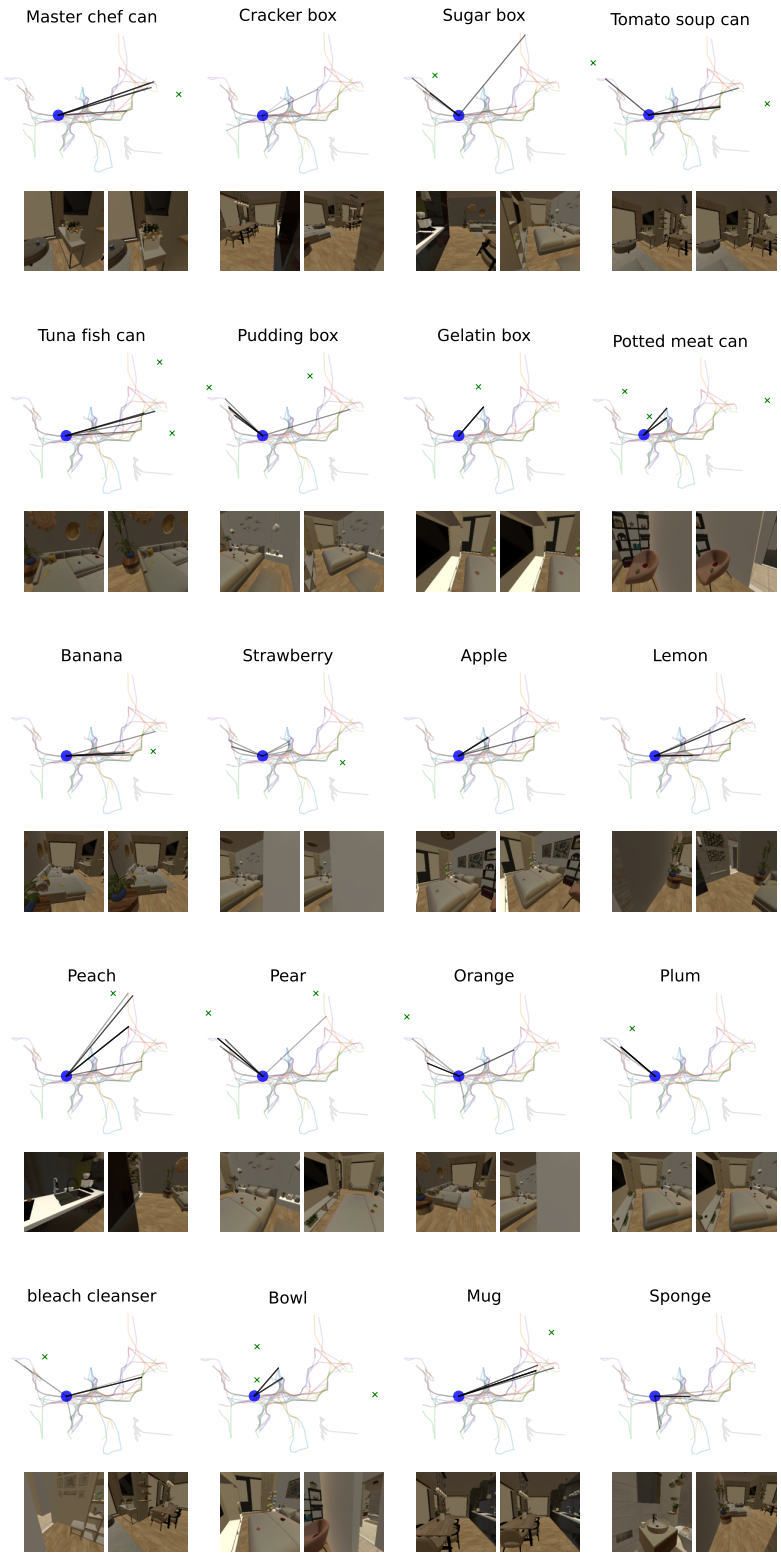


Figure 12: Attention scores of the object detection head, see Section 4.5. The colored curves are the trajectories of previous episodes. The blue circle is the agent's position. The black lines represent the agent's attention when the target is the type in above the image. The lines connect the agent with the point in history that it attends to, the opacity of the line represents the attention score. The two images with highest attention score are shown in the 3rd row.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction reflects the paper contribution. We introduce RL-ICL and show that it scales to 64,000 steps. We also show that the RL-ICL is capable of few-shot imitation learning and we study the important components of the RL-ICL.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We added a section discussing the limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We don't have theoretical result.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We added sufficient details to reproduce the result and we plan to release the code and the datasets.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We plan to release the source code and release the data.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We describe in details how the data is generated and the hyperparameters and training details. We also plan to release the code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We ran all models in the main experiment with 3 seeds and show the standard deviation on these seeds.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We mention the required Computational resources and the time required for the training of the main experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics and our work confirms with it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There's no societal impact of our work since we are studying general approach to increase training sequence length in an Embodied AI task.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no risk since it's a general method to train long sequence length for Embodied AI tasks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the owners of the datasets and tools we used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We document the data generation process and the method details. We'll also release the new task's dataset and the code to generate it.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.