Fine-tuning LLM Agents with Retrospective In-Context Online Learning

Wen-Tse Chen[1], **Jiayu Chen**[1], **Fahim Tajwar**[1], **Hao Zhu**[2], **Xintong Duan**[1],
Ruslan Salakhutdinov[1], **Jeff Schneider**[1]
Carnegie Mellon University[1], Stanford University[2]
`{wentsec, jiayuc2, xintongd}@andrew.cmu.edu`
`{ftajwar, rsalakhu, Jeff.Schneider}@cs.cmu.edu`
`zhuhao@stanford.edu`

Abstract

Fine-tuning large language models (LLMs) using online learning, where models learn from self-sampled data and environmental feedback, presents a promising but challenging research direction due to the typically sparse nature of rewards. Traditional methods for addressing this challenge often involve training domain-specific Q-functions to convert sparse rewards into dense signals. However, these methods suffer from poor sample efficiency and limited generalizability. In this work, we propose a novel framework that leverages the pre-trained knowledge of LLMs to transform sparse rewards into dense supervised signals through in-context learning. Specifically, we introduce a retrospective in-context learning approach, where LLMs assign temporal credit to past actions based on feedback. Unlike previous approaches, which rely heavily on extensive feedback data or intricate prompt engineering, our method uses online learning to iteratively update the policy by combining in-context learning with gradient-based fine-tuning. We empirically demonstrate the effectiveness of our approach on the BabyAI benchmark, showing that it is significantly more sample-efficient than traditional online reinforcement learning (RL) algorithms while achieving comparable performance to imitation learning. Our findings suggest that LLM-based agents can refine their policies using sparse feedback in an online manner, making them more adaptive to dynamic environments.

1 Introduction

Fine-tuning large language models (LLMs) using online learning, where they learn from self-sampled data and environment feedback, is a promising research direction [Dong et al., 2024]. However, the task is inherently difficult due to the typically sparse nature of environment feedback [Andrychowicz et al., 2017, Sukhbaatar et al., 2017]. This sparse environmental feedback not only increases the sample complexity of the algorithm but also raises its variance and instability [Chaudhari et al., 2024, Cao et al., 2024]. In this work, we propose leveraging the pre-trained knowledge of LLMs to transform sparse rewards into dense signals, enabling more efficient learning from environment feedback.

In-context learning, which allows agents to adapt quickly to new tasks with limited samples [Chen et al., 2024a], is a powerful method for utilizing the pre-trained knowledge of LLMs. Despite its success in decision-making tasks, LLM-based agents have faced criticism for their limited ability to self-correct, even when provided with environmental feedback [Kamoi et al., 2024]. Previous works have mostly relied on LLMs' generalization abilities for reflection, typically using either (1) large amounts of feedback generation data to train the reflector [Chen et al., 2024b], or (2) intricate prompt engineering [Shinn et al., 2024].

To reduce the reliance on LLMs' in-context learning capacity for self-correction, we propose a retrospective approach to assign temporal credit using in-context learning. Specifically, when a sparse environment feedback is received, LLMs use in-context learning to update all of its prior actions on the trajectory. Empirically, we show that retrospectively in-context policy updating is more effective than general policy improvement.

In this work, we propose an online learning framework that embraces the instability of LLMs while leveraging their pre-trained knowledge. We use LLMs as both the initial policy and a reflector that evaluates whether past actions were beneficial or detrimental based on environmental feedback. While retrospective in-context learning simplifies policy refinement for observed trajectories, it does not generalize to unseen states. To address this, we apply gradient updates and retrospective in-context learning iteratively to store the in-context updated policy. Given the dynamic programming nature of decision-making tasks, the proposed framework can continuously learn from environmental feedback in an online manner.

In summary, our contribution are as follow:

(1) Empirically showed that LLM agents can improve their policy on the given trajectory through in-context learning when provided with environment feedback.

(2) Proposed an online learning framework where agents utilize retrospective in-context learning to transform sparse environment feedback into dense supervised signals, followed by supervised fine-tuning to embed these feedback into the model's parameters.

(3) We empirically demonstrate the effectiveness of our algorithm on the BabyAI benchmark, showing it is more sample efficient than online RL algorithms, while achieving comparable performance to imitation learning.

2 Using In-context Learning to Accelerate Online Learning

In this section, we outline the rationale behind the algorithm. First, we revisit the classic online RL algorithm, demonstrating how to effectively integrate in-context learning with online RL. Second, we explain the role of in-context learning in the proposed algorithm and show how it can generate dense supervised signals from sparse environment feedback and how to use the in-context updated policy to refine the base policy.

2.1 Notation

We consider a Markov decision process [Puterman, 1990], defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the actio space, $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ is an unknown transition function, and r is the rewards function. We will use $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t q(s_t = s|\pi)$ to represent the unnormalized discounted state distribution induced by the policy π, where $q(s_t = s|\pi)$ is the likelihood of the agent being in state s after following π for t time-steps. The goal of policy learning is to maximizes the expected sum of rewards, $\mathbf{E}_{s\sim d_\pi(s) a\sim\pi(\cdot|s)}[r(s, a)]$.

2.2 Revisit Online RL

Online RL operates in two phases: policy evaluation and policy improvement. During policy evaluation, the agent uses environment feedback to update the Q-function, resulting in more accurate Q-value estimates for the state-action pairs along the trajectory. In the policy improvement phase, the agent uses the updated Q-function to refine the policy, where the Q-values serve as dense supervised signals, making it easier to train the policy.

We proposed two insights here. First, the success of online RL suggests that environment feedback can be effectively used to update the Q-value for the trajectory just experienced, though its impact on improving estimation accuracy for other states is uncertain. During policy evaluation, the RL agent updates the Q-value only for the states along the observed trajectory. However, the dynamic programming nature of the algorithm leads to overall improvements in Q-value estimation. Similarly, in in-context learning with environment feedback, we focus on using the feedback to back-propagate updates to the policy based on the trajectory just taken. This approach is simpler than trying to learn an improved policy across all possible states, thereby reducing the proposed method's reliance on LLM's in-context learning capabilities.

The second insight is that in-context learning enables direct policy improvement from environment feedback, bypassing the traditional value function updates. Online RL uses rewards to guide the training of value function, and uses well-trained value function as a dense supervised signal to train the policy. By leveraging in-context learning, we can directly produce a better policy compared to the base policy using environment feedback. This approach allows us to bypass value function learning, and indicates that we have to use in-context learning as the way we use value function. While in-context learning does not guarantee monotonic improvement like online RL, it leverages the pre-trained knowledge of LLMs and can accommodate non-numerical feedback, such as text-based input, significantly accelerating the training process.

2.3 In-context Learning as Dense Rewards Generator

This subsection explains how in-context learning improves policy using environment feedback. The goal of in-context learning is to use the environment feedback to back-propagate updates to the policy along the trajectory just traversed. Following Shinn et al. [2024], in-context learning can be broken down into two key steps: generating verbal feedback in response to trial-and-error interactions, and subsequently using that verbal feedback to in-context update the policy.

We highlight three important considerations for in-context learning. First, the LLM used to generate verbal feedback should be distinct from the policy model, as fine-tuning the policy could diminish its generalization ability [Kirkpatrick et al., 2017]. Second, while previous works generated one universal verbal feedback for all states, our framework allows the agent to produce individualized verbal feedback for each state, which further simplifies tasks solved by in-context learning. Third, similar to centralized training with decentralized execution [Yu et al., 2022] in multi-agent RL, global or privileged states can be utilized to generate feedback, since they are only needed during training and not during inference.

After getting the in-context updated policy, the next step is applying gradient updates to align the base policy with it. Similar to online RL [Schulman, 2015], we incorporate a trust region term into the optimization objective to enhance the stability of online training. That is, we don't want the current policy deviate a lot from the sampling policy. The rationale for introducing the trust region term is twofold. Firstly, feedback is based on trajectories collected by π_k. When π diverges significantly from π_k, the feedback becomes less informative. Secondly, due to the noisy nature of verbal feedback, the trust region acts as a regularizer to avoid collapsing into sub-optimal solutions.

In sum, while our primary contribution is demonstrating that in-context learning can accelerate online training, much of our discussion focuses on how to manage the instability of in-context learning. We tackle this by simplifying the tasks assigned to in-context learning and designing the online learning algorithm to be robust against noisy outputs from in-context learning.

Algorithm 1 Retrospective In-Context Online Learning(RICOL)

```
Input: $\pi_0, K$
for $k = 0 \cdots K-1$ do
    $\tau_k \sim \pi_k(\cdot|s_0)$                                   ▷ Sample new trajectory
    for $s_t = s_0 \cdots s_T \in \tau_k$ do
        feedback$_t \sim \pi_{reflect}(\cdot|s_{t:T}, r_{t:T})$          ▷ Feedback generation
        $\pi'_{k+1}(\cdot|s_t) \leftarrow \pi_k(\cdot|s_t, \text{feedback}_t)$   ▷ In-context update
    end for
    $\pi_{k+1} \leftarrow arg\min_\pi \mathbf{E}_{s\in\tau}[\text{Distance}\left(\pi(\cdot|s), \pi'_{k+1}(\cdot|s)\right)]$   ▷ Gradient update
end for
```

3 Methodology

In this section, we introduced a practical algorithm Retrospective In-Context Online Learning (RICOL). As shown in Algorithm 1, RICOL is an online learning algorithm, switching between data collection, in-context learning and gradient update. We will elaborate each steps in the following paragraph. The prompt can be found in Appendix D.

3.1 Feedback Generation

The first step is to use policy π_k to collect trajectories τ_k. For each time step t, we use the sequence $s_{t:T}, a_{t:T}$, and $r_{t:T}$ as a prompt to query the critic model π_{critic}, where π_{critic} is an LLM distinct from π_k. The critic model is prompted to evaluate whether the action a_t was good, given the future trajectory. The output of π_{critic} is a verbal feedback, denoted as feedback$_t$.

3.2 In-context Update

After getting feedback$_t$, we append it to the prompt and query π_k to obtain the in-context updated policy π'_{k+1}. Note that π'_{k+1} is not used to generate a new roll-out. Instead, we store the probability output by π'_{k+1} on all the state of τ_k and use them to guide policy training later.

3.3 Gradient Update

After obtaining the dense supervised signal π'_{k+1}, we apply a gradient update to update the current policy π, moving it closer to π'_{k+1}. At the same time, we don't want it to deviate too much from the sampling policy π_k. Similar to Peng et al. [2019], the loss function can be defined as follow:

$$\min_{\pi} \mathbf{E}_{s\sim d_{\pi_k}(s)} \left[D_{KL} \left(\frac{1}{Z(s)} \cdot \pi_k(\cdot|s) \odot \exp(\frac{\log \pi'_{k+1}(\cdot|s)}{\alpha}) || \pi(\cdot|s) \right) \right], \tag{1}$$

where $\odot$ represents the element-wise multiplication and α is a hyper-parameter defining the size of the trust region. The derivation process can be found in Appendix B.

4 Experiments

This section addresses three research questions. **Q1**: Is retrospective in-context policy updating with environment feedback easier than updating the general policy? **Q2:** Does the proposed online learning framework rely heavily on the model's in-context learning ability? **Q3:** How does RICOL's performance and sample efficiency compare to baseline algorithms?

We evaluated our method using the BabyAI benchmark [Chevalier-Boisvert et al., 2018]. BabyAI is a 2D grid-world navigation task where the agent follows language-based instructions. This task is challenging for LLM agents due to their known limitations in spatial reasoning [Wu et al., 2024]. By default, we use Llama-3.1-8B-Instruct as policy π, and use Llama-3.1-70B-Instruct as critic π_{critic}.

4.1 Retrospective In-Context Policy Updating(Q1)

In this subsection, we empirically show that using environmental feedback to backpropagate and update the policy in-context based on the trajectory just taken is easier than updating for states the agent has not encountered. Figure 1 compares the performance of retrospective in-context learning (Retro-ICL) with traditional in-context learning (ICL). Since retrospective in-context learning cannot alter the policy distribution during evaluation, we do not use win rate as an evaluation metric. Instead, we measure how accurately the policy predicts the expert's behavior in a given trajectory.

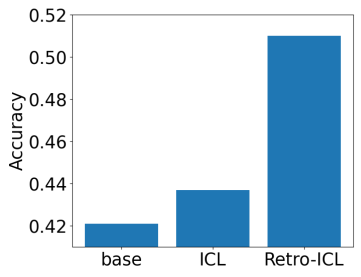
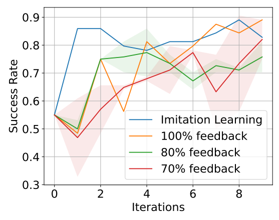
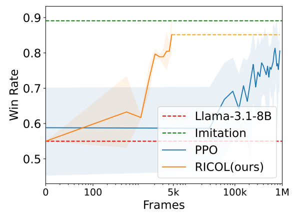


Figure 1: Using in-context learning to retrospectively update the policy (Retro-ICL) is more effective than updating a general policy (ICL).

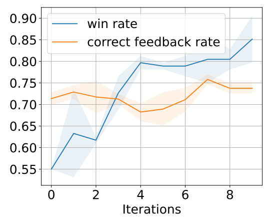
The base policy reflects the performance of zero-shot LLMs. In ICL, we use in-context learning to generate verbal feedback from one trajectory and evaluate the in-context updated policy (which takes the verbal feedback as input) on a different trajectory. Retrospective-ICL, on the other hand, evaluates the updated policy on the same trajectory used to generate the feedback, and provides different verbal feedback for each state. Our empirical results show that Retro-ICL outperforms ICL by 7.2%, confirming that retrospectively updating the policy is easier than updating the general policy. The reported results are averaged over 1,000 data points.



(a) robustness.



(b) main results.



(c) feedback accuracy.

Figure 2: (a) shows that the proposed online algorithm is able to learn from noisy verbal feedback, where 80% feedback implies training with feedback that is accurate only 80% of the time. (b) shows that our method has similar performance with imitation learning but doesn't need expert policy and is more efficient than online RL. (c) shows that the accuracy of the generated feedback is stable during training, indicating that we are not exploiting critic model's extra information but learn from environment feedback.

4.2 Robustness of Online Learning (Q2)

In this subsection, we discuss how the accuracy of verbal feedback impacts the performance of the online learning algorithm. Specifically, we train the agent using hand-written verbal feedback and then manually adjust its accuracy to evaluate the robustness of the algorithm. Figure 2a illustrates that the proposed online learning algorithm is capable of training effectively even when the verbal feedback accuracy is as low as 70%, where random feedback has an accuracy of 50%. This result demonstrates that the algorithm does not heavily depend on highly accurate in-context learning for training.

4.3 Main Results(Q3)

In this subsection, We compared our method against imitation learning and online RL. Figure 2b demonstrates that our method exhibits performance comparable to imitation learning, despite being an online approach that trains on self-sampled data rather than an expert policy. Additionally, our method is more sample-efficient than online RL methods, which must learn a domain-specific Q function from scratch. In contrast, our approach utilizes LLMs as a pre-trained policy and generates dense rewards to accelerate training. The implementation detail can be found in Appendix C.

Figure 2c shows that the accuracy of verbal feedback remains relatively constant throughout the training process. This stability suggests that the agent is effectively learning from environmental feedback rather than relying on additional information stored in the critic network. If the agent were primarily benefiting from corrections provided by the critic, we would expect the accuracy of verbal feedback to decline over time, due to the critic's limited knowledge capacity.

5 Conclusion and Limitation

We proposed an online learning framework that uses retrospective in-context learning to convert sparse rewards into dense signals, allowing LLM agents to improve policies based on past trajectories. By combining this with gradient-based fine-tuning, our method addresses in-context learning instability and enables continuous policy refinement. Empirical results on the BabyAI benchmark show our approach is more sample-efficient than traditional online RL while performing comparably to imitation learning.

The proposed method has the following limitations. First, we did not incorporate chain-of-thought prompting Wei et al. [2022] when designing the algorithm pipeline; the current version only allows the agent to output actions directly. Second, we tested the method on a single navigation benchmark and with one LLM, leaving its broader applicability to other benchmarks, including token-level MDPs and other LLMs, uncertain.

References

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

M. Cao, L. Shu, L. Yu, Y. Zhu, N. Wichers, Y. Liu, and L. Meng. Drlc: Reinforcement learning with dense rewards from llm critic. *arXiv preprint arXiv:2401.07382*, 2024.

T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

S. Chaudhari, P. Aggarwal, V. Murahari, T. Rajpurohit, A. Kalyan, K. Narasimhan, A. Deshpande, and B. C. da Silva. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *arXiv preprint arXiv:2404.08555*, 2024.

Y. Chen, S. Zhang, G. Qi, and X. Guo. Parameterizing context: Unleashing the power of parameter-efficient fine-tuning and in-context tuning for continual table semantic parsing. *Advances in Neural Information Processing Systems*, 36, 2024a.

Z. Chen, K. Zhou, W. X. Zhao, J. Wan, F. Zhang, D. Zhang, and J.-R. Wen. Improving large language models via fine-grained reinforcement learning with minimum editing constraint. *arXiv preprint arXiv:2401.06081*, 2024b.

M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

H. Dong, W. Xiong, B. Pang, H. Wang, H. Zhao, Y. Zhou, N. Jiang, D. Sahoo, C. Xiong, and T. Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

R. Kamoi, Y. Zhang, N. Zhang, J. Han, and R. Zhang. When can llms actually correct their own mistakes? a critical survey of self-correction of llms. *arXiv preprint arXiv:2406.01297*, 2024.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

H. Li, X. Yang, Z. Wang, X. Zhu, J. Zhou, Y. Qiao, X. Wang, H. Li, L. Lu, and J. Dai. Auto mc-reward: Automated dense reward design with large language models for minecraft. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16426–16435, 2024.

Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

R. Y. Pang, W. Yuan, K. Cho, H. He, S. Sukhbaatar, and J. Weston. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*, 2024.

X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

J. Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

X. Wang, J. Chen, Z. Wang, Y. Zhou, Y. Zhou, H. Yao, T. Zhou, T. Goldstein, P. Bhatia, F. Huang, et al. Enhancing visual-language modality alignment in large vision language models via self-improvement. *arXiv preprint arXiv:2405.15973*, 2024.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

W. Wu, S. Mao, Y. Zhang, Y. Xia, L. Dong, L. Cui, and F. Wei. Visualization-of-thought elicits spatial reasoning in large language models. *arXiv preprint arXiv:2404.03622*, 2024.

C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35: 24611–24624, 2022.

W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

W. Yuan, R. Y. Pang, K. Cho, S. Sukhbaatar, J. Xu, and J. Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

A Related works

A.1 Using LLMs as Critic or Reflector

Several methods Yuan et al. [2024]Pang et al. [2024], Wang et al. [2024] utilized LLMs' self-critique capabilities to label a preference dataset, then trained the agent using DPO Rafailov et al. [2024]. Our method also employs LLMs to label the dataset, but rather than requesting binary labels in text form, we use the log probabilities output by LLMs as the supervised signal. Chen et al. [2024b] proposed using LLMs as reflectors to output dense rewards. Half of their contribution focuses on training the reflector model, whereas our work embraces the inherent instability of LLMs. Furthermore, they address token-level MDPs, while we tackle tasks involving LLM agents. Lastly, after receiving dense rewards, they train a value-function and use the PPO objective to update their policy. In contrast, we bypass the need for a value-function and learn directly from the generated dense supervised signals.

A.2 Updating LLMs with environmental feedback

Several methods have improved LLMs by using in-context learning as a dense rewards generator. Ma et al. [2023], Kwon et al. [2023], Yu et al. [2023]Li et al. [2024] utilized LLMs to output rewards either as code or prompts, and then updated these rewards with environmental feedback through in-context learning. Shinn et al. [2024], Zhao et al. [2024] encoded prior experiences into text-form memory and incorporated this memory into the prompt to improve decision-making in subsequent trials. These approaches require LLMs to fully comprehend the dynamics of the environment with only a few-shot task transition provided, while our method only requires LLMs to improve performance on the trajectory they have just experienced.

A.3 Fine-tuning LLMs as Multi-tasks Agent

Chen et al. [2024a] integrates in-context learning with low-rank fine-tuning to encode few-shot examples into model parameters, addressing catastrophic forgetting in multi-task continuous learning. Our approach shares a similar pipeline but focuses on encoding environmental feedback into parameters, utilizing online learning rather than expert demonstrations.

B Derivation of The Loss Function

The derivation is mainly from Peng et al. [2019]. The goal of the gradient update step is to bring π closer to π'. One way to achieve this is by maximizing the following objective:

$$\eta(\pi) = \mathbf{E}_{s\sim d_\pi(s), a\sim\pi(\cdot|s)} \left[\log \pi'(a|s)\right], \tag{2}$$

where $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s|\pi)$ represents the unnormalized discounted state distribution induced by the policy π, and $p(s_t = s|\pi)$ is the likelihood of the agent being in state s after following π for t time-steps.

In practice, to enhance sample efficiency, we aim to use $d_{\pi_k}(s)$ instead of $d_\pi(s)$ when estimating $\eta(\pi)$, where π_k is the policy used to sample new data. Therefore, similar to , we can employ $\hat{\eta}(\pi)$ as an approximation of $\eta(\pi)$ in practical implementations.

$$\hat{\eta}(\pi) = \int_s d_{\pi_k}(s) \int_a \pi(a|s) \left[\log(\pi'(a|s))\right] da ds. \tag{3}$$

$\hat{\eta}(\pi)$ is a reasonable estimator of $\eta(\pi)$ only when π and π_k are sufficiently similar. Therefore, rather than directly solving the optimization problem $\max_\pi \eta(\pi)$, we can instead reformulate it as the following optimization problem:

$$\begin{aligned} arg\max_{\pi} & \int_s d_{\pi_k}(s) \int_a \pi(a|s) \left[\log(\pi'(a|s))\right] da ds \\ \text{s.t.} & \int_s d_{\pi_k}(s) D_{KL}\left(\pi(\cdot|s)||\pi_k(\cdot|s)\right) \leq \epsilon \\ & \int_a \pi(a|s) da = 1, \forall s. \end{aligned} \tag{4}$$

By applying the method of Lagrange multipliers to the constrained optimization problem, the optimal policy can be expressed as follows:

$$\pi^*(a|s) = \frac{1}{Z(s)} \pi_k(a|s) \exp\left(\frac{1}{\alpha} \log \pi'(a|s)\right), \tag{5}$$

where $Z(s) = \int_{a'} \pi_k(a'|s) \exp\left(\frac{1}{\alpha} \log \pi'(a|s)\right)$ normalizes the optimal policy and α is the Lagrange multiplier.

Finally, we need to project π^* back onto the manifold of parameterized policies. This can be achieved by optimizing the following objective:

$$\begin{aligned} & arg \min_{\pi} \mathbf{E}_{s \sim d_{\pi_0}(s)} \left[D_{KL} \left(\pi^*(\cdot|s) || \pi(\cdot|s) \right) \right] \\ =& arg \min_{\pi} \mathbf{E}_{s \sim d_{\pi_0}(s)} \left[D_{KL} \left(\frac{1}{Z(s)} \cdot \pi_k(\cdot|s) \odot \exp(\frac{\log \pi'_{k+1}(\cdot|s)}{\alpha}) || \pi(\cdot|s) \right) \right], \end{aligned} \tag{6}$$

where $\odot$ represents the element-wise multiplication. In practice, we treat α as a hyper-parameter that controls the size of the trust region.

C Experiment

In this section, we provide the implementation detail of the environment, baseline algorithms and the training setup.

We evaluate our algorithm in the BabyAI environment [Chevalier-Boisvert et al., 2018], a 2D grid world where the player navigates an agent to accomplish specified tasks. We use Chevalier-Boisvert et al. [2023]'s implementation of the environment, and use Carta et al. [2023]'s caption function to get the text form observation. Our primary focus is on the *GoToLocal* scenario. As shown in Figure 3, the player directs the agent (a red triangle) toward a specified local object. The environment is partially observable, with the agent having a 7x7 window of visibility in front of it, viewed from an egocentric perspective. The agent can perform six actions: Turn Left, Turn Right, Move Forward, Pickup, Drop, and Toggle. Both input and output for the tasks are presented in text form.

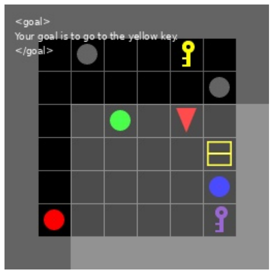

<goal>
Your goal is to go to the yellow key
</goal>

Figure 3: Screenshot of the environment.

We use the open-source project rl-starter-files to implement the PPO algorithm as reported in the paper, leveraging BERT and CNN as the model backbones. To ensure a fair comparison, we train a policy that achieves a 55% win rate, matching the zero-shot win rate of Llama-3.1-8B-Instruct, using PPO. This policy serves as the starting point for our training, while the value network is trained from scratch. We admit that the baseline algorithm utilizes a much smaller model size than our method. However, we emphasize that this baseline is used to illustrate the slow pace of training a value function from scratch, and any RL-based method will require this training process.

Finally, we use the open-source project LLaMA-Factory to implement our algorithm. The training process requires 4 NVIDIA RTX A6000 GPUs and takes 3 hours to complete 10 iterations.

D Prompt Template

D.1 Agent Prompt

```
You are a helpful navigation agent in a 2D grid world with these rules:

1. "Forward" moves you "1 step" in the direction you're facing.
2. "Left" makes you turn 90 degrees in place to the left.
3. "Right" makes you turn 90 degrees in place in the right direction.
4. You cannot move onto a grid with an item.
```

```
I will provide you with the goal and the current state,
which includes the location of all nearby objects relative to you.
You should then respond to me with the next action you should take
to achieve your goals.
The action needs to be in possible actions.

<possible actions>
Pickup.
Drop.
Toggle.
Forward.
Left.
Right.
</possible actions>

DESIRED FORMAT:
<next action>
YOUR ACTION HERE
</next action>
```

D.2 Critic Prompt

```
You are a helpful navigation agent in a 2D grid world with these rules:

1. "Forward" moves the agent "1 step" in the direction it is facing.
2. "Left" makes the agent turn 90 degrees in place to the left
from its current orientation.
3. "Right" makes the agent turn 90 degrees in place to the right
from its current orientation.
4. The agent cannot move on grids occupied by walls or other objects.
5. The coordinates of the object are represented by (x, y).

I'll provide you with a trajectory that includes your goal (destination),
actions taken, and states encountered.
The map is presented at state t, and in the following state,
we only provide the agent's position and orientation
At the end of the trajectory, it shows whether you successfully reach the goal.
You have to analyze the trajectory and determine if you would change or maintain
your action at state t based on the information provided.


Now you have another chance to retry the task.
You are placed at state t again.
Your goal is to move the agent beside the goal and face it or approch
the goal as close as possible.
Reply with whether you would change or maintain your action t.
Consider environmental feedback and goal when making decisions.
Explain your reasoning, then conclude with your decision.
Be concise and clear.

DESIRED FORMAT:
<feedback you should follow>
In the given state t, the agent is at (X,Y), facing DIRECTION,
with the goal located at (X, Y).
The goal is located to the DIRECTION-DIRECTION of the agent.
YOUR REASONING HERE.
</feedback you should follow>
<conclusion>
```

```
In the previous attempt, I chose action <<ACTION T HERE>>.
This time, I will maintain/change the selected action.
</conclusion>
```