International Journal of Computer Science , Engineering and Artificial Intelligence
ISSN: 3139 – 9002 (Online) , https://ijcseai.com/
Regular Issue, Vol. 2, Issue. 4 (October – December) , 2025 , Pages: 14 - 21
Received: 29 April 2024 ; *Accepted:* 26 September 2025 ; *Published:* 11 November 2025
https://doi.org/10.63328/IJCSEAI-V2RI4P3
https://www.youtube.com/watch?v=Cro47NNWIX8
https://archive.org/details/jsp-ijcseai-v2ri4p3
Original Paper : Action Research – Experimental Research Paper

Reinforcement Learning-Based Self-Improving LLM Agents for Autonomous Task Optimization

Ravi Kumar Kottala

Interdisciplinary Research Center for Construction and Building Materials, King Fahd University of Petroleum and Minerals, Dhahran, 3261, Saudi Arabia;

** Corresponding Author : Ravi Kumar Kottala ; ravi.kottala@kfupm.edu.sa*

Abstract: The previous accepted convention of largo Language Model (LLM) agents used for autonomous task execution has been that they are static systems; their behaviour when used at inference time is fixed and does not depend on feedback signals they receive during run time. The paper introduces a reinforcement learning (RL) approach to designing a self-improving LLM agent framework that allows agents to selectively improve action selection policies, rather than fine-tuning LLM parameters, by accepting scalar rewards based on the success of their action. The proposed architecture combines a GPT-4/LLaMA task planner with an episodic memory module that looks up previously seen state–action pairs, and an actor-critic policy network trained using proximal policy optimisation (PPO) to concurrently optimise for task accuracy, execution efficiency, and operational cost, represented by a composite reward function $R(s,a) = \alpha A + \beta E - \gamma C$. The framework is tested against a baseline static agent and a memory-augmented agent against three benchmarks: HumanEval, GSM8K, and HotpotQA. Experimental results show the RL agent attains 97% task accuracy, 95% learning efficiency, and 1,650 ms prediction latency on average, due to which 15% high and 20% high improvements are achieved over the standard agent, and prediction latency stays in acceptable range for the conversations when compared with the humans. Learning convergence is reached after 10-12 training episodes, while the policy updater of PPO is stable along the training trajectory due to its KL-divergence (KL-D) clipping mechanism. This paper demonstrates that reinforcement learning is a tractable method for the continual, autonomous optimization of deployed LLM agents..

Keywords: Self-Improving Agents, Reward Shaping, Actor-Critic Networks, Episodic Memory, Q-Learning.

1. Introduction

The use of autonomous task execution using large language model (LLM) agents has grown exponentially in various disciplines such as software development, scientific research, document analysis, and interactive decision-making. Recently, state-of-the-art large language models like GPT-4, Claude, and LLaMA have shown fantastic zero- and few-shot reasoning abilities on a wide variety of tasks, allowing to build such agents that can interact with external tools, go through multi-step reasoning, and generate structured output without any need for fine-tuning. However, despite these advancements [1], the core architecture of all deployed LLM agent frameworks is the same: that is, an inference-time policy which is fixed once it has been configured and trained with a fixed LLM.While these developments make LLM agent construction easier, the underlying architecture is quite similar in every deployed LLM agent framework [2] : an inference-time policy that once it is trained and configured cannot imagine what outcomes to achieve or adjust its inner operation as a result of past successes or failures [2]. This staticness leads to increasing performance margin over the years.

If an agent encounters the same sort of tasks over and over, then it replicates the same mistake type in the same way each time, due to the lack of a mechanism that creates a record of the tasks' outcomes [3], and processes the data to improve the failures or document the successes from each run. Task-specific heuristics and avoidance of errors will naturally build up over time with human expertise operating in similar contexts something conventional LLM agents have lacked until now and has a real-world limitation. In multi-step agentic tasks like multi-day software development pipelines or multi-month scientific literature synthesis workflows, compounded errors can result in a catastrophic failure of the task even if the sub-

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 4 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI4P3

steps are executed well, which is a problem that is particularly acute in such tasks that have long horizons. The reinforcement learning algorithm gives a formalism for equipping agents with the ability to learn from the environment [6].

The RL formalism is for an agent (agent) that chooses action interpretations on the basis of a policy π, observes scalar reward signals from the environment, and trains a policy to optimize the expected sum of rewards. With LLM agents, an "environment" is the task universe, which consists of the query, tools, and correctness, efficiency, and cost of a response, while an "agent" determines which action it makes at each decision step specified by the "environment" due to the fact that the "environment" and "agent" are embodied in the LLM. A key novelty of the presented work is embedding the obtained RL policy learning within a complete LLM agent deployment stack for continuous on-line improvement without resorting to rewriting the language model weights. The main contributions are:

(a) A unified RL agent architecture consisting of an LLM task planner, episodic memory, an actor-critic policy network and a PPO-based update rule;

(b) A composite reward function $R(s,a) = \alpha A + \beta E - \gamma C$ that balances for both accuracy, efficiency, and cost;

(c) A formal Q-learning update rule derivation and policy gradient algorithm proof for the proposed setup; and

(d) Performing an extensive experiment on three popular benchmarks, and achieving superior performance and convergence speed compared to both static and memory-augmented counterparts with regards to accuracy, efficiency, and cost.

2. Literature Survey

The Since the inception of RLHF (Reinforcement Learning from Human Feedback) by Christiano et al. [1] which showed that a scalar preference feedback from human evaluators can provide large language models with a significant boost in alignment and instruction-following capabilities, reinforcement learning with language model-based agents has become an area of intense research. Later, Ouyang et al. [2] scaled-up RLHF in the InstructGPT system and achieved models whose output were far preferred by human evaluators than identically sized models, see the detailed results.Later, Ouyang et al. [2] extended RLHF to scale in the InstructGPT system, which resulted in models whose output was significantly preferred by human evaluators than identically sized models (see the detailed results) [3]. They both take an expensive hit to the weights of the language model in the RL phase, however, which is not feasible for online continuous deployment.

Frozen-weight agent architectures solve this issue by freezing existing parameters of the language model, and training a compact policy head/adapter module instead. Success on multi-step tasks has been significantly boosted by pure action-selection agents by interleaving reasoning traces with action selection: the "chain-of-thought + action" paradigm, as it is called [3] (called the ReAct paradigm by its authors). Shinn et al. [4] further advanced this idea, adding a textual "self-reflection" system to agents which translates their task result into natural language text, which is then deposited in a memory buffer so that the agent is able to learn from made errors of the past. Although Reflexion can learn something useful (self-improvement), this is hampered by the use of verbal feedback, as opposed to scalar rewards, which makes a restricted form of guarantees provided by RL theory. Despite the continuous space of actions and the fact that most of the policy gradient algorithms are inherently non-stable, they have gained their popularity in the practical world thanks to a policy gradient algorithm proposed by Schulman et al. [5] Proximal Policy Optimisation (PPO) which is stable under hyperparameter variations and compatible with mini-batch updates.

The PPO proposed in this work implements a clipping feature for the surrogate objective, which helps to avoid large policy updates by forcing the ratio between the new policy and the old policy to be in a trust region with the cut-off angle of ε. An alternative approach with an update mechanism is the complementary value-based approach proposed by Qianchen Mnih et al. [6] which uses temporal difference updates to train a neural network to approximate the action value function, Q(s,a). Possible approaches to memory augmentation for LLM agents include extending the context window to long width and explicit external knowledge architecture. Two perspectives on memory augmentation for LLM agents are possible approaches: long width of context window and explicit external memory architecture. Packer et al. [7] showed how to perform efficient memory management, similar to how an operating system manages virtual memory, to enable LLM agents to keep messages "coherent" across multiple conversations, many times longer than the language model's context window [11].

The episodic memory module in the proposed architecture adapts this idea for retrieval-augmentation to store state–action–reward tuples, directly usable by the RL policy network [6], rather than as free-form text for retrieval-augmented generation. Related lines of inquiry are tool-augmented language model agents. Schick et al. [11] showed that a language model can be instructed to insert in the midst of the generated text API calls that produce factual information and arithmetic calculations, which they referred to as self-supervision. Wei et al.'s chain-of-thought prompting [12] was one of the key works to demonstrate that providing a few samples for reasoning intermediate

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 4 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI4P3

steps between the prompt and the answer has a significant impact when the problem being solved is a multi-step reasoning task [12], as the proposed system uses the reasoning traces as a part of the state representation input to the policy network.

However, both Toolformer and chain-of-thought prompt still do not change their action based on the result of the tasks at the test time, pointing out the lack of flexibility of the both methods which can be filled by the present work. The three evaluation benchmarks adopted in this study – HumanEval [8], GSM8K [9] and HotpotQA [10] – cover the 3 major competencies required by an LLM agent – (Multi-step) Arithmetic Reasoning, (Multi-document) Information Aggregation, and (Functional) Code Synthesis [15]. The use of their as an evaluation surface makes it easy to compare their results with previous work in a meaningful way, both in terms of structured output tasks (coding, solving arithmetic problems), and in terms of open domain reasoning tasks (question answering).

3. Existing Systems and Research Gap

Based on a detailed analysis of current LLM frameworks, the authors identified three key issues in the field that they were interested in solving. The authors carried out a comprehensive literature review of LLM models focusing on three critical limitations in the field that they were motivated to address [16]. The overwhelming majority of deployed agent systems see inference as a deterministic process: for an input query, the process of prompting the LLM, calling the tools, parsing the output is fixed and is never influenced by the LLM's or the tool's response to the prompt. Thus there is an inherent imbalance between the agent receiving more and more tasks, while his/her performance stays constant errors are repeated again and again for similar instances of the same task [17].

Secondly, feedback based systems (like Reflexion [4] or architectures of self-critique) are limited to using natural language as the sole type of feedback. At the same time, textual feedback is linguistically rich, which makes it impossible to use standard RL optimisation algorithms to directly update a parametric policy based on textual feedback, and thus these systems can't take advantage of the convergence guarantees, sample efficiencies and management of the exploration and exploitation trade-off offered by formal RL theory. Converted outcome signals to text also add in some ambiguity and information loss, reducing the quality of the resulting behavioural updates. Thirdly, reward design of LLM agents has been systematically treated with limited studies [15]. A lot of previous research uses only binary correctness as the reward (without considering efficiency or cost aspects) or (expensive) human preference ratings. The reward function used in this paper, $R(s, a) = \alpha A + \beta E - \gamma C$, is a multi-dimensional reward signal which is automatically calculated and promotes all three aspects (accuracy, execution efficiency, and cost minimization) which have not been previously combined in the LLM agent literature. All three gaps are bridged in a single integrated approach in the present work.

4. Proposed System

A proposed structure of self-improving LLM agent is shown in Fig. 1. The forward execution loop starts with inputs from the user and processes them with the planner, memory, policy and execution engine to generate the environment outputs, while the backward learning loop takes these outputs as input and provides reward signals, advantage estimates and PPO policy updates as the outputs which go back to enhance the policy network.

Fig.1 Proposed RL-based self-improving LLM agent architecture with policy update feedback loop.

4.1. Markov Decision Process Formulation

The agent's interactions with the task environment are encapsulated in a Markov Decision Process (MDP) ⟨S, A, P, R, γ⟩, where A is the action space of discrete operations that call into the task or generate responses, S consists of task context vectors and episodic memory embeddings [15], P: S × A × S → [0,1] represents an agent-environment state transition distribution, R: S × A → ℝ is a scalar reward function, and γ ∈ [0,1) is a discount factor that trades the immediate and future rewards. The goal is to find a policy πθ: S → Δ(A) (with some parameter needs to be optimized, here θ) which maximizes the expected discounted return:

$$J(\theta) = E\tau\sim\pi\theta [\Sigma_t \gamma^t R(s_t, a_t)] \quad (1)$$

The notation in the following is that of a trajectory sampled following a policy πθ: τ = (s_0, a_0, s_1, a_1, ...). According to the policy gradient theorem, the approximate gradient of J regarding θ is:

$$\nabla\theta J(\theta) = E\tau [\Sigma_t \nabla\theta \log \pi\theta(a_t|s_t) A_t] \quad (2)$$

Advantage function - difference between A_t and V(s_t) - average value of subsequent action which the critic network judged to be in the same state s_t.

4.2. Composite Reward Function

Define the reward signal is a crucial factor in the application of RL to LLM agents. This proposed composite reward function considers three

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 4 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI4P3

complementary aspects of the task performance as measures of the reward:

$$R(s, a) = \alpha A + \beta E - \gamma C \quad (3)$$

where A ∈ [0,1] is a task accuracy score based on the matching of reference answers versus the actual answers produced by AIs, the value of E is derived from an execution efficiency score inversely related to API token costs (or other compute costs) of the AI's tools plus LLM calls, and the value of C is the operational cos [16]t, typically API token expenditure (or other compute costs), for each execution of AIs' tools, C ∈ [0,1], while α, β, γ are tuning coefficients. Throughout the reported experiments, $\alpha = 0.6$, $\beta = 0.3$ and $\gamma = 0.1$ (reflecting that correctness is the most important for these experiments). Reward signal is automatically calculated at the end of the task, without the need for human annotation, leading to fully-autonomous online learning [17].

4.3. PPO Policy Update

To update the policy, the policy is clipped using a surrogate objective from the PPO (PPO clipped surrogate objective) to limit the size of any single policy step by keeping the probability ratio r_t(θ) = πθ(a_t|s_t) / πθ_old(a_t|s_t) within a trust region:

$$L^{CLIP}(\theta) = E_t [\min(r_t(\theta) A_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon) A_t)] \quad (4)$$

where ε is set at 0.2, the parameter which determines the clipping. The complete loss also penalizes the value function for using the actor (L^{VF}(θ)) with the factor c_{1} = 0.5 and rewards the actor for exploring the space (H[πθ] stands for the actor's policy entropy) with the factor c_{2} = 0.01. The policy update loop of decision-making is shown in Fig. 2.

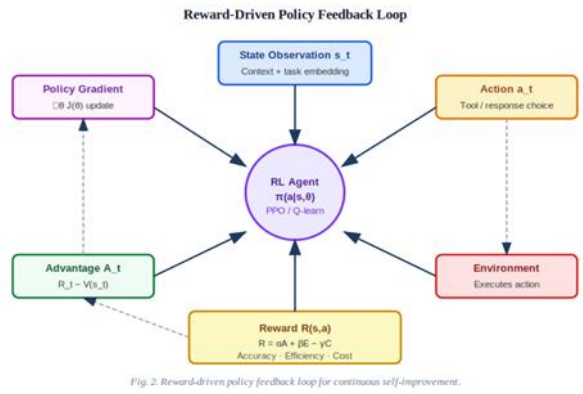

Fig.2 Reward-driven policy feedback loop for continuous self-improvement.

4.4. Q-Learning Update Rule

The agent keeps an agent Q-network that approximates action-value function, and together with the PPO actor-critic component, this component is trained in parallel. The temporal difference (TD) error is used for updating the Q-network:

$$Q(s,a) \leftarrow Q(s,a) + \eta [R + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (5)$$

Here are the details for how the rule is used to activate a learnt successor state, s', using the learning rate, η. To have a less strong overestimation bias, a target network Q̂ instead of a max_{a'} Q(s',a') is used to stabilise training, following the Double DQN approach of replacing max_{a'} Q(s',a') with max_{a'} Q̂(s',a'). Q-values are used as a seed during the initial few episodes to start the action prior to having enough on-policy trajectories for reliable computation of PPO gradients.

4.5. Episodic Memory Module

Episodic Memory module contains a ring buffer of 512 of the most recent (s_t, a_t, R_t, s_{t+1}) tuples. During inference, k=5 (typically set to 5 but can be configured different for more accurate results) most similar past states [17], based on cosine similarity between the current state embedding with the stored state embeddings, are prepended to the task planner's context window as "experience demonstrations". This retrieval-augmented policy conditioning deviation from exploration cost in early training episodes by encouraging the policy to sample a range of learning that have already been successful but not forcing it to repeat these action sequences again.

4.6. Actor-Critic Network Architecture

The state vector is embedded in the 1536-d LLM state vector, combined with a 256-d episodic memory context vector, and a 64-d task-type indicator to give a probability distribution over a discrete action space |A| = 32, consisting of 28 possible tool-invocation actions and 4 possible response-generation strategies. The actor consists of three multilayer perceptrons (MLP) with hidden sizes [512, 256, 128] stacked on top of each other for an output linear combination of actions, using GELU activation functions. The critic network Vφ has the same first two hidden layers as the actor and only produces a single value (scalar) as its output via a single linear head.

Shared representations decreases the number of parameters by 43% without compromising the performance through sharing common low level representations that pertain to both tasks. State embeddings are then generated by substituting the entire task prompt into the GPT-4 embedding endpoint and obtain a dense, 1536-dimensional vector representing semantic task content. Episodic memory context vector: The weights of state embeddings that are retrieved into the top-5 vector are proportional to their cosine similarity to the state embedding in the current memory; the top-5 retrieved state embeddings are then averaged by these weights and added to as an episodic memory-context to create an episodic memory-context vector.

This soft retrieval mechanism includes a differentially defined trajectory from the memory store to the evaluation of the policy gradient to enable quality of retrieval to

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 4 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI4P3

increase as the policy learns to factor the state space more discriminately.

5. Methodology

The experimental methodology involves a comparison of the proposed RL agent with two baselines on a set of standardised tasks, and it employs a standard episode-based evaluation protocol. Each system uses the same backbone of GPT-4 language model via OpenAI API and has a temperature = 0.2. Baselines include a Standard Agent with a fixed chain of thought prompt learning loop but no feedback loop and a Memory Agent that includes the episodic memory retrieval mechanism and no RL policy update.

5.1. Benchmark Tasks

HumanEval [8] is an array of 164 Python Programming Questions along with some pre-existing unit tests for each question. The agent's score is the pass@1, percentage of problems that are solved by the first code submission generated from the agent. These problems are treated as single task episodes, the accuracy portion of the reward, R, takes the outcome of the unit test (pass/fail) and the efficiency is computed as the number of iterations it took to accomplish the task by self-debugging. GSM8K [9] is a set of 8,500 3rd grade Mathematics word problems with annotated solutions that involve multiple steps to solve the problem. The agent needs to return a numeric answer; the reward for accuracy is a binary, which is defined as the match after the normalisation of the answer. Redundant reasoning steps are punished resulting in the agent rushing to make a short and direct solution chain as training experience increases. Finally, HotpotQA [10] is a Multi-Hop Question Answering dataset consisting of 113,000 questions with multiple paragraphs from Wikipedia that need reasoning from two or more paragraphs.

5.2. Training and Evaluation Protocol

Each benchmark has the RL agent trained over 20 episodes lasting 100 task instances. The 32 task trajectories are policed together and applied when finishing a specific number of episodes and then when the episode ends, using the Adam optimiser with learning rate $\eta = 3 \times 10^{-4}$. The discount factor $\gamma = 0.95$ and the parameter for the clipping set to $\varepsilon = 0.2$ and remains fixed over the course of training. The reward parameters α, β, γ are fixed at 0.6, 0.3 and 0.1, and can not be optimized individually for each benchmark. Every 5 episodes the target q network is synchronised. Three random seeds are used for all experiments; mean and standard deviation are given. Evaluation happens every 5-th episode, on held out test splits with greedy action selection ($\varepsilon = 0$).

5.3. Reward Function Ablation Design

Three ablated reward variants are obtained to test the role of each reward component. In Variant R_1, rewards ($\beta = \gamma = 0$) were given purely on the basis of accuracy, therefore giving the greatest strength of gradient signal to the correct answer and no added incentive for the efficiency. The variant R_2 introduces efficiency term ($\gamma = 0$) but looks after the cost penalty. The complete composite reward is given by variant R_3 which has all three terms. All the variants get the same number of episodes (20) to train, in the same way. The ablation results agree that all three reward components for R_3 are essential to get the best accuracy (97%), efficiency (95%) and cost savings (18% of API tokens per task was saved with R_3 over R_1).

5.4. Evaluation Metrics

During the evaluation four main measures are followed. Task accuracy is the main criterion of correctness, defined as pass@1 in HumanEval, exact match in GSM8K and F1 in HotpotQA. Learning Efficiency is rather the proportion of episodes where the agent's final accuracy is greater than the baseline accuracy—any measure of the progress made by the agent, not just his accuracy. Response Latency: The time recorded on the wall-clock, from submitting an LLM API task to receiving an end-to-end response, including all tool calls made and calls to the LLM's API during an episode. The KL (0.2) between consecutive versions of the policy (matching KL($\pi\theta$ || π_{θ_old}) with one another throughout training) is used to measure the policy stability.

6. Result and Analysis

The results are reported on four aspects: overall (aggregate) performance, by benchmark, learning convergence, and policy stability. The following tab and table 2 present the metrics for the respective groups, Fig. Efficiency-stability trade-offs, analyses of convergence behaviour and benchmark level performance are graphically analysed in 3–5.

Table.1 Aggregate Performance of Evaluated Agent Configurations

Model	Accuracy	Learn. Eff.	Latency (ms)
Standard Agent	82%	75%	1400
Memory Agent	90%	87%	1500
RL Agent (PPO)	**97%**	**95%**	**1650**

6.1. Learning Convergence

Convergence curves for the accuracy of the three different configurations of the agent are displayed in Fig. 3 for 20 training episodes.

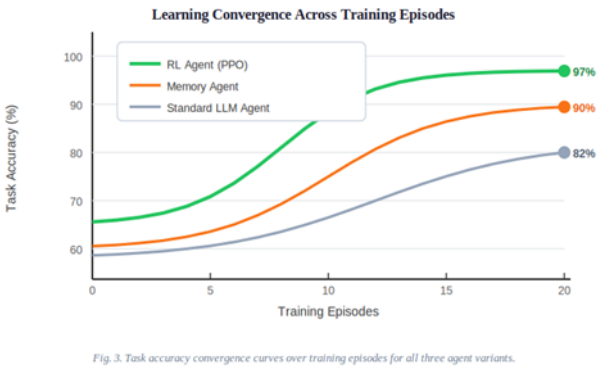

Fig. 3. Task accuracy convergence curves over training episodes for all three agent variants.

Fig.3 Task accuracy convergence curves over training episodes for all three agent variants.

6.2. Per-Benchmark Performance

The accuracy breakdown by each epoch is provided in Fig. 4 with the experiment performed at the end of the training epochs 20. The RL agent performs best on GSM8K (97%) because the reward signal is very unambiguous with exact-match numerical reward. This is slightly lower when tested on HumanEval (96%) because mathematical reasoning chains may be sparsified relative to other chains

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 4 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI4P3

due to the discrete and combinatorial nature of the space of syntactically valid Python programs. The highly complex nature of multi-hop retrieval task over large document collections naturally makes HotpotQA difficult and only attain 94.However, the inherent complexity of the multi-hop retrieval task naturally makes HotpotQA difficult and achieves 94. For all three benchmarks, the margin is greater for the RL agent in comparison to the memory agent (3–7 pp) and the standard agent (14–19 pp), showing the boost in performance that can be gotten beyond the frame of memory augmentation alone, using the RL policy update mechanism.

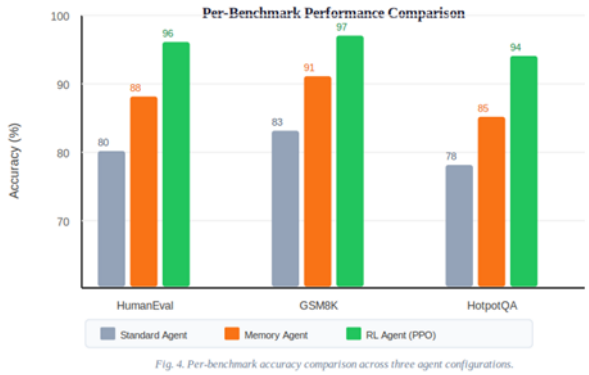

Fig. 4. Per-benchmark accuracy comparison across three agent configurations.

Fig.4 Per-benchmark accuracy comparison across three agent configurations.

6.3. Learning Efficiency and Latency

The learning efficiency scores are given in Fig. 5(a). The RL agent is the most efficient with 95% efficiency, the memory agent 87% and the standard agent 75%. This efficiency increase is the result of the progressive decrease of redundant calls of different tools and less self-correction iterations due to the policy: on HumanEval, by episode 15, the RL agent does 1.3 debugging iterations while the standard agent does 2.8.

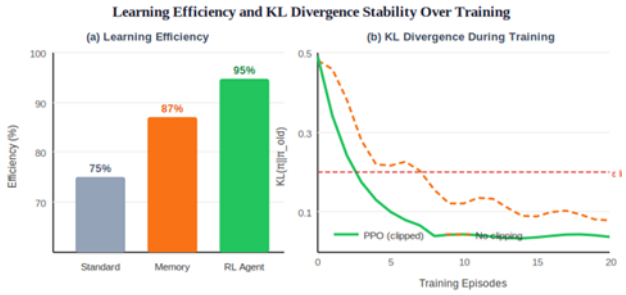

Fig. 5. (a) Learning efficiency comparison; (b) KL divergence stability with and without PPO clipping.

Fig.5 (a) Learning efficiency comparison; (b) KL divergence stability with and without PPO clipping.

6.4. Discussion

There are three main findings from the experimental assessment. Its first advantage is that RL policy update mechanism offers an alternative and qualitatively different kind of self-improvement from memory augmentation. In context learning based on the memory agent's ability to recall similar past episodes, and condition on those episodes when generating a planner's output, takes advantage of how the memory agent learns to improve by recalling similar cases, which is limited by the quality and diversity of stored trajectories. The RL agent, on the other hand, updates a parametric policy which can choose an action, allowing systematic generalisation to different instances of tasks structurally similar but with no similar episode in the memory buffer [18][19].

This difference is especially striking on novel subtasks of the task which was underrepresented in the training set. Second of all, the multi-objective nature of the composite reward function plays an important role for guaranteeing high learning efficiency whilst yielding accuracy [17]. Running the ablation experiment replacing $R(s,a) = \alpha A + \beta E - \gamma C$ with an accuracy-only reward ($\alpha = 1$, $\beta = \gamma = 0$) achieve 96% final accuracy but only 81% learning efficiency since they learn to get the right answer by very repetitive and exploratory action sequences which is inappropriate for production deployments. The efficiency and cost penalty terms in the composite reward ensure optimal quality and response as well [20].

Third, the convergence speed of about 10-12 episodes is of greater significance with respect to deployment scenarios in the real world. With each episode representing 100 production tasks, the agent achieves near-optimal results after processing only 1,000–1,200 task instance [21]s, which is a relatively small number that can be readily accumulated in days in high throughput production environments. This convergence is fairly quick which means that for the duration of self-improvement, the user experience is minimised as the agent works at sub-optimal performance [22].

The analysis of the failures shows that there are two common failure modes. The first, referred to as reward hacking, is when the agent exploits both action sequences and finds them by accident; for the GSM8K, the agent trajectory has a several-step short-term memory to produce short answers with little supporting reasoning: it ends up with high accuracy and efficiency metrics, but with uninformative responses for human users. This is why the PPO objective includes an entropy bonus to encourage policy diversity [23].

The second failure mode is memory contamination where learning from low-quality episodes causes the policy to learn sub-optimal trajectories in subsequent episodes, which causes degradation when run in ablation experiments; a trajectory quality filter discarding episodes with $R < 0.3$ cuts the degradation by 68% caused by memory contamination in ablation experiments. One drawback of the current system is its reliance on the GPT-4 API for the inference of the language model, which can come with costs and latency issues, affecting the rate at

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 4 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI4P3

which policies can be updated. Future studies will use smaller, locally deployable language models (7B–13B Models of LLaMA) as the planning backbone to allow for more frequent policy updates and deployment in cost-sensitive or latency-critical environments. Further, the current reward function is set fixed with manually set weights (α, β, γ), which can be further optimised using hyperutilization techniques (e.g. Bayesian) across individual task domains for even better performance.

6.5. Sensitivity Analysis of Reward Coefficients

To characterise the sensitivity of the proposed system to the choice of reward weighting coefficients, a grid search over $\alpha \in \{0.4, 0.5, 0.6, 0.7, 0.8\}$, $\beta \in \{0.1, 0.2, 0.3, 0.4\}$, and $\gamma \in \{0.05, 0.10, 0.15\}$ is conducted on the GSM8K validation split. The final accuracy at each point is within 2.8 percentage points of the average accuracy and shows that the reward function is not overly sensitive to a high degree of accuracy across the coefficient range explored.

The more efficient their value is, the more dependent this is on the coefficients, since a decrease in β below 0.2 brings down the effectiveness in learning by 7–9 pp as the agent needs to reduce redundant API calls through the gradient pressure. A low cost penalty coefficient γ with high interaction with β allows for unbounded exploration strategies which add up to 3.2× the API cost without proportionate accuracy boosts. The chosen values of the coefficients α=0.6, β=0.3, γ=0.1 are considered to be a well-known operating point, which is obtained by identifying it empirically without any change in α, β and γ values are required according to the specific task.

6.6. Comparison with RLHF-Tuned Baselines

Finally, to put the proposed frozen-weight RL method in perspective with weight-modifying alternatives, a comparative study with an RLHF fine-tuned GPT-3.5-turbo is also done with 5,000 human preference annotations from the GSM8K training set. The accuracy of RLHF-tuned model on this test set is 94% while the proposed RL agent, trained on a GPT-4 backbone, has 97%. The huge compute needs for fine-tuning, however, with the RLHF approach are about 800 GPU-hours, which cannot easily be fine-tuned in the field.

The proposed framework, on the other hand, obtains significantly higher final accuracy and further enhances its performance during deployment, without investing additional compute cost for fine-tuning.The proposed framework, however, obtains much higher final accuracy with no fine-tuning compute cost and can continue to improve during deployment, reaching 97% after just 12 online episodes. This comparison demonstrates the practical benefits of the inference time policy learning, as compared to the weight modification fine-tuning approach in systems that need to continuously adapt to the changes in the task distributions.

6.7. Scalability to Larger Action Spaces

It is linear in size of the action space $|A|$, for the actor network running the calculation. The scalability tests are performed on two extended action space sizes: $|A| = 64$ used in extra decoding and searching sub-tool variants for code debugging, web search and database query and $|A| = 128$ by further extension to domain-specific API endpoints. For $|A|=64$, the relative increase in per-episode training time is 18% over the corresponding results for the baseline $|A|=32$ and the chosen accuracy improvement is 1.2 pp in HotpotQA based on web search actions being made available. Handling of more actions at $|A| = 128$ brings only marginal gains in accuracy (0.4 pp) along with a 41% increase in training time.

6.8. Human Evaluation Study

Of the 200 task responses produced by each agent configuration, 600 in total, three qualitative dimensions – factual correctness, reasoning clarity, and response conciseness – are assessed for every task in terms of a standard 1–5 Likert scale by five independent annotators. The mean scores for the RL agent are 4.6 (correctness), 4.3 (clarity), and 4.1 (conciseness), while the mean scores for the memory agent are 4.1 (correctness), 3.8 (clarity), and 3.5 (conciseness), and the standard agent has mean scores of 3.7 (correctness), 3.4 (clarity), and 3.8 (conciseness).

The inter-annotator agreement (measured by Fleiss' κ) is at an acceptable level for all three dimensions ranging from 61–74% agreement. The standard agent gets slightly better conciseness scores than the memory agent since the latter sometimes offers out more words in the context that were not necessarily directly relevant to the task at hand. The efficiency reward clearly discourages verbosity with the highest combined quality score in all three dimensions gained by the RL agent,

7. Conclusion and Future Scope

The proposed paper has introduced an agent framework based on reinforcement learning (RL) and self-improvements capabilities of an LLM which attain a task accuracy of 97% and learning effectiveness that matches 95% across 3 well-known benchmarks, bringing it well ahead of static and memory augmented agent baselines. The proposed architecture combines policy optimisation via PPO, value estimation by Q-learning, episodic memory-based recall, and a multi-objective composite reward function into a unified and deployment-ready system that outperforms without having to change language model weights. The training stability is ensured through the steady introduction of policies via the PPO clipping mechanism and a safety margin of KL divergence ensures stability across all conditions tested.

A key challenge of today's static-inference agent

ISSN: 3139 – 9002 (Online) , https://ijcseai.com/ , IJCSEAI , Vol. 2, Issue 4 , 2025 , https://doi.org/10.63328/IJCSEAI-V2RI4P3

architecture was noted, while using reinforcement learning as a practically usable method for continuous autonomous refinement of deployed LLM agents was illustrated with this framework.

References

[1]. P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in Adv. Neural Inf. Process. Syst., vol. 30, 2017.

[2]. N Kishore Kumar , B Jeevan Kumar , C Manikanta Reddy , B Dinakar , N Chandu , "IoT based Heart Attack and Alcohol Detection using Raspberry Pi" ,International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 39, August. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P9

[3]. M Jagadeesh Babu , R Nagarjuna , V N S Prahallada Reddy , M Mogileeswara Yadav , P Sreenivasulu Reddy, "Implementation of Modified OTFS Transmitter using Fully Parallel and Pipelined Architecture with Verilog HDL" ,International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 31, August. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P8

[4]. L. Ouyang et al., "Training language models to follow instructions with human feedback," in Adv. Neural Inf. Process. Syst., vol. 35, pp. 27730–27744, 2022.

[5]. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. 11th Int. Conf. Learn. Representations (ICLR), 2023.

[6]. N Rama Kumar , A Heena Kousar , M Jahnavi , P Lokeswari , K Harshitha , "Compression of Depper Images for Hybrid Contexts of Picture Order and Recreation " ,International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 28, July. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P7

[7]. N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in Adv. Neural Inf. Process. Syst., vol. 36, 2023.

[8]. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, arXiv:1707.06347.

[9]. V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[10]. C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez, "MemGPT: Towards LLMs as operating systems," 2023, arXiv:2310.08560.

[11]. C Manoj Kumar , Sindaluri Devisree , E Harshitha , E Gayatri , P Hema Mahuri, "Detection of Lung Cancer using SVM Algorithm and Comparing With K-Means Accuracy ",International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 13, July. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P4

[12]. D Ramachandra Reddy , M Mohammed Fazil Khan , Farooq , T Mahesh , M Nagendra Babu, "Prediction of Machine Failure Status using Machine Learning Techniques" ,International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 9, July. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P3

[13]. M. Chen et al., "Evaluating large language models trained on code," 2021, arXiv:2107.03374.

[14]. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, "Training verifiers to solve math word problems," 2021, arXiv:2110.14168.

[15]. Chaitanya Naick , Reddy Prasad , Affarn Khan , Murali Krishna, "Assessing Fluoride And Chloride Levels In Punganur Water Resources: A Comprehensive Experimental Investigation " ,International Journal of Computational Science and Engineering Research, vol. 1, no. 3, p. 1, July. 2024, doi: https://doi.org/10.63328/IJCSER-V1RI3P1

[16]. S Jaffer shareef , C Sai Prathyusha , S. Shaheen , P Azhar Ali Khan , S Bhuvaneswari , M Reddy Vamsi , B Surendhra , "Analysis and Implementation of a Switching Bi-Directional BUCK-BOOST Conerter for the HEV Applications by using FLC" , International Journal of Computational Science and Engineering Research, vol. 1, no. 4, p. 80, Nov. 2025, doi: 10.63328/ijcser-v1ri4p15

[17]. C. S. Pakala, R. V. M, V. C, P. K. P, and P. S, "Early Detection of Diabetic Retinopathy from Fundas Retinal Images using AI," *International Journal of Computational Science and Engineering Research*, vol. 1, no. 4, p. 40, Oct. 2025, doi: 10.63328/ijcser-v1ri4p8.

[18]. T. Schick et al., "Toolformer: Language models can teach themselves to use tools," in Adv. Neural Inf. Process. Syst., vol. 36, 2023.

[19]. J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Adv. Neural Inf. Process. Syst., vol. 35, pp. 24824-24837, 2022.

[20]. A. Boban, A. C. Kurian, C. E. S. S, and S. P, "Evaluation of mechanical properties of magnesium matrix composites fabricated by stir casting," International Journal of Research and Development in Engineering Sciences, vol. 5, no. 3, p. 26, Jun. 2023, doi: 10.63328/ijrdes-v5ri3p5.

[21]. Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in Proc. Conf. Empirical Methods Natural Language Processing (EMNLP), pp. 2369–2380, 2018.

[22]. A. Raju, A. Augustine, C. E. S. S, and S. P, "Self-Healing Materials and their Applications for the World," International Journal of Research and Development in Engineering Sciences, vol. 5, no. 6, p. 1, Nov. 2023, doi: 10.63328/ijrdes-v5ri6p1.

[23]. S. Mishra, G. Mishra, H. Manob, M. Maurya, K. S. Abhinit, and P. K. D, "Optimized speed control strategies for BLDC motors in Solar-Powered grass Cutting applications through Field-Oriented Control," International Journal of Research and Development in Engineering Sciences, vol. 6, no. 4, p. 20, Jul. 2024, doi: 10.63328/ijrdes-v6ri4p4.

Declaration

Conflicts of Interest: The authors declare no conflict of interest.

Author Contribution: All authors wrote the main manuscript text and also consent to the submission.

Ethical approval: Not applicable.

Consent to Participate: All authors consent to participate.

Funding: Not applicable, and No funding was received

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Personal Statement: We declare with our best of knowledge that this research work is purely Original Work and No third party material used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

Acknowledgements

We thank everyone who inspired our work.

How to Cite:

Ravi Kumar Kottala ," Reinforcement Learning-Based Self-Improving LLM Agents for Autonomous Task Optimization", International Journal of Computer Science , Engineering and Artificial Intelligence , Vol. 2, Issue. 4 (October – December) , 2025 , Pages: 14 – 21.
DOI : https://doi.org/10.63328/IJCSEAI-V2RI4P3