How to Train Your LLM Web Agent: A Statistical Diagnosis

Anonymous Authors[1]

Abstract

Large language model (LLM) agents for web interfaces have advanced rapidly, yet open-source systems still lag behind proprietary agents. Bridging this gap is key to enabling customizable, efficient, and privacy-preserving agents. Two challenges hinder progress: the reproducibility issues in RL and LLM agent training, where results often depend on sensitive factors like seeds and decoding parameters, and the focus of prior work on single-step tasks, overlooking the complexities of web-based, multi-step decision-making.

We address these gaps by providing a statistically driven study of training LLM agents for web tasks. Our two-stage pipeline combines imitation learning from a Llama 3.3 70B teacher with on-policy fine-tuning via Group Relative Policy Optimization (GRPO) on a Llama 3.1 8B student. Through 240 configuration sweeps and rigorous bootstrapping, we chart the first compute allocation curve for open-source LLM web agents. Our findings show that dedicating one-third of compute to teacher traces and the rest to RL improves MiniWoB++ success by 6 points and closes 60% of the gap to GPT-4o on WorkArena, while cutting GPU costs by 45%. We introduce a principled hyperparameter sensitivity analysis, offering actionable guidelines for robust and cost-effective agent training.

1. Introduction

Large-language-model (LLM) agents capable of driving web interfaces have seen rapid progress, yet open-source systems continue to trail proprietary agents by a significant margin. Closing this gap is crucial: enabling organizations to train their own agents would allow them to customize workflows, compress models for lower latency and cost, and maintain full control over sensitive data, fostering user trust.

However, progress in this area is hindered by two compounding challenges. First, reproducibility remains an ongoing concern in reinforcement learning (RL) and LLM agent training, with studies showing that reported gains can be sensitive to factors such as random seeds, sampling parameters, and prompt formatting (Hochlehnert et al., 2025). While large industry labs often have the resources to run extensive multi-seed experiments (Abdin et al., 2025), such large-scale evaluations remain challenging for smaller research groups due to the high computational cost of LLM training, making it harder to ensure reliable and generalizable findings.

Second, most existing work on LLMs with RL has focused on single-step tasks such as code generation or mathematical reasoning (Yu et al., 2025; DeepSeek-AI et al., 2025; Roux et al., 2025). While effective in constrained domains, these approaches fail to capture the complexities of real-world tasks requiring multi-step planning and sequential decision-making. Recent benchmarks such as WebArena (Zhou et al., 2023), WorkArena (Drouin et al., 2024), OSWorld (Xie et al., 2024), and The Agent Company (Xu et al., 2024) highlight these challenges, exposing the brittleness of current methods when deployed in realistic, interactive environments.

In this paper, we tackle these gaps by providing a statistically driven diagnosis of training LLM agents for web-based, multi-step tasks. Specifically, we study how to allocate compute between expensive, high-quality off-policy demonstrations from a large teacher model and cheaper on-policy rollouts from a smaller student model. Too much reliance on the former leads to brittle imitation, while too much of the latter drowns learning in sparse web-task rewards.

We address this trade-off with a two-stage pipeline: a LLaMA 3.3 70B teacher generates K successful episodes, and a LLaMA 3.1 8B student first imitates these trajectories before branching into Group Relative Policy Optimization (GRPO) for on-policy fine-tuning.. By sweeping 240 configurations across the SFT-to-RL ratio and key hyperparam-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

eters, and applying a bootstrapping protocol, we chart the first data-driven compute allocation curve for open-source LLM agents.

Our findings reveal clear patterns. On MiniWoB++, our approach pushes the compute-performance frontier by balancing expert demonstrations with on-policy reinforcement learning. This combination outperforms both pure imitation learning and naive supervised fine-tuning, achieving stronger generalization while making more efficient use of compute. On the more challenging WorkArena benchmark, our method emerges as a promising approach in low-compute regimes.

Beyond compute-performance analysis, we introduce a statistically disciplined method for hyperparameter diagnosis and sensitivity analysis. This addresses key reproducibility gaps by estimating the likelihood of hyperparameter configurations being optimal, correcting for imbalanced exploration, and providing actionable guidelines for practitioners (see Section 6).

The study yields several actionable insights. First, branching into RL early—but not immediately—after SFT leads to better outcomes. Second, curriculum learning and error log feedback help when agents start from scratch but become counterproductive once SFT warmup is applied. Third, in GRPO, applying zero-advantage filtering, avoiding standard deviation normalization of the advantage, and skipping importance ratio correction and trust region consistently improve performance. Fourth, decoding temperature is consistently critical, while learning rate and discount rate must also be carefully tuned.

These findings are significant for two reasons. First, they give smaller research groups a reproducible, budget-aware playbook for pushing open LLM agents closer to state-of-the-art without scaling model size. Second, they address a slice of the broader reproducibility concerns recently highlighted in the RL community (Agarwal et al., 2021; Hochlehnert et al., 2025), offering a template for rigorous, statistically-grounded hyperparameter tuning and compute allocation in LLM agent training.

2. Background

This section consolidates the algorithmic ingredients used throughout the paper: (i) the MDP formulation of web-based language agents, (ii) supervised fine-tuning (SFT) on expert traces, (iii) Grouped-Return Policy Optimisation (GRPO) for reinforcement learning, and (iv) curriculum and normalisation techniques that stabilise training.

2.1. Language Agents as MDPs

We model each task as a Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma \rangle$. A *state* $s_t \in \mathcal{S}$ is represented as text, while an *action* $a_t \in \mathcal{A}$ is also expressed as a high-level textual command. The environment returns a scalar reward $r_t \in [-1, 1]$ indicating task progress or completion. The agent's policy $\pi_\theta(a \mid s)$ is parameterized by an LLM with weights θ and is trained to maximize the expected discounted return:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{k \geq 0} \gamma^k r_{t+k}\Big]. \tag{1}$$

Here, $\gamma \in [0, 1]$ is the discount rate, which controls the agent's preference for immediate versus future rewards: a lower γ encourages short-term behaviors, while a γ closer to 1 incentivizes long-term planning.

In practice, the policy $\pi_\theta(a \mid s)$ is instantiated as an autoregressive LLM that generates actions token by token by sampling from a probability distribution over the vocabulary. This introduces inherent stochasticity into the agent's behavior, which is modulated by a *decoding temperature* parameter τ.

2.2. Off-policy boot-strapping via SFT

Generating high-reward web trajectories on a fresh model is prohibitively slow. We therefore first imitate a stronger *expert* policy π_E by minimising the cross-entropy loss

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{|\mathcal{D}_{\text{exp}}|} \sum_{(s,a) \in \mathcal{D}_{\text{exp}}} \log \pi_\theta(a \mid s). \tag{2}$$

where $\mathcal{D}_{\text{exp}}$ contains (s, a) pairs from successful expert episodes, including chain-of-thought tokens. SFT offers a high-quality, low-variance gradient but is inherently *off-policy*: the distribution of states seen at evaluation time drifts away from $\mathcal{D}_{\text{exp}}$ as soon as the student deviates from the expert.

2.3. On-Policy Improvement with GRPO

After supervised fine-tuning (SFT), we apply *Grouped-Return Policy Optimisation* (GRPO), a clipped policy gradient method that operates on sets of G output tokens grouped by task, trajectory, or outcome. Let q denote a prompt segment (or query), and let o_i represent a sampled token from the model's response. Define the importance ratio $r_i = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$, and let A_i denote the advantage estimate, computed using group-normalised returns.

The GRPO objective is:

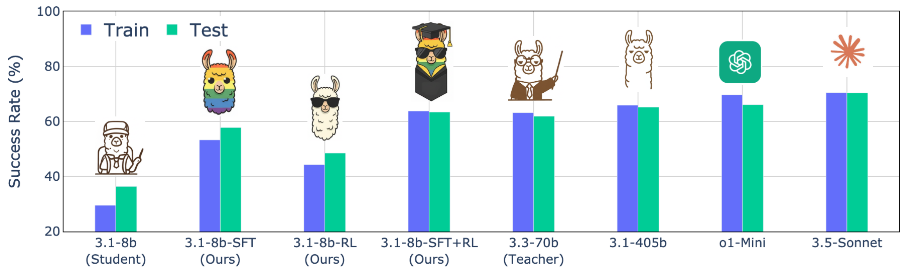


Figure 1: Comparison of our method with baselines from (de Chezelles et al., 2025) on the MiniWoB++(Liu et al., 2018) benchmark. Our 8B Llama 3.1 model is on par with all the larger models such as Llama 3.1 405B, o1-mini and 3.5-sonnet.

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q)}\,\mathbb{E}_{\substack{\{o_i\}_1^G \\ \sim\pi_{\theta_{\text{old}}}}}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big(r_{i,t}\hat{A}_{i,t},\; \text{clip}\big(r_{i,t},1-\varepsilon,1+\varepsilon\big)\hat{A}_{i,t}\Big) - \beta\, D_{\text{KL}}\big(\pi_\theta \| \pi_{\text{ref}}\big)\Bigg] \tag{3}$$

where the clipped importance ratio enforces conservative updates akin to PPO (Schulman et al., 2017), and the KL divergence against a reference policy π_{ref} (typically the SFT model) acts as a trust-region regulariser weighted by β. The objective recovers PPO in the special case $G{=}1$, $\beta{=}0$.

Group-Normalised Advantage. GRPO computes the advantage A_i using only the rewards associated with the G outputs in the same group. Let r_i denote the scalar reward assigned to output o_i, and let $r = (r_1, \ldots, r_G)$ be the vector of all rewards in the group. The normalised reward is computed as $\tilde{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$, and the advantage is set as $A_i = \tilde{r}_i$.

This normalisation encourages intra-group differentiation while being invariant to the absolute reward scale. However, Liu et al. (2025) argue that retaining the per-group standard deviation σ_g in the denominator introduces a variance-inflating bias during training.

Zero-advantage filtering. Tokens with $A_{t,g} = 0$ contribute no learning signal yet still consume memory. Dropping them yields a constant *effective* batch size and modestly accelerates training (Yu et al., 2025).

2.4. Curriculum through Variance-Aware Boltzmann Sampling

To promote steady learning progress, we design a curriculum that prioritizes challenging tasks, neither trivial nor too difficult (Thakkar et al., 2023). Specifically, we select tasks according to a Boltzmann distribution centered around a target return μ_{target} which specifies the desired performance threshold, encouraging focus on partially mastered tasks, with a temperature parameter τ controlling the sharpness of the distribution, with lower values concentrating probability mass tightly around μ_{target}.

This sampling mechanism dynamically adapts the training distribution, concentrating learning on tasks where the agent is neither already proficient nor entirely unskilled. As a result, the agent avoids premature convergence on easy tasks and prevents wasted effort on tasks far beyond its current capabilities.

3. Methodology

Our training recipe comprises two sequential stages—SFT followed by RL—whose compute budgets we treat as a resource allocation problem. We also detail our hyperparameter sweep and the statistical protocol that converts hundreds of runs into reliable conclusions. We evaluate our recipe along two axes: compute cost, measured in FLOPs (using an adapted formula from (ben)), and model performance, evaluated on both *unseen* training goals and testing tasks.

Stage 1 – Supervised Fine-Tuning (SFT). We generate N_E expert episodes using a large/powerful model, filter for successful rollouts, and store (s, a) pairs including chain-of-thought annotations. Optimizing Eq. (2.2) for T_{SFT} steps yields an initial policy π_{θ_0}. Given $t_b \in [0, T_{\text{SFT}}]$ steps of SFT, we calculate the total compute cost $\text{F}_{\text{SFT}}(t_b)$ as the sum of FLOPs used during data generation by the teacher model and the FLOPs used to train the student model.

Stage 2 – RL Fine-Tuning. From π_{θ_0}, we branch B checkpoints $\{t_b\}$ at different intervals. Each branch continues with GRPO (Eq. (3)) for T_{RL} steps. Here the cost of T_{RL} steps $F_{\text{RL}}(T_{\text{RL}})$ is equivalent to the sum of FLOPs for one data generation step and one training step of the student model. The total compute for a given run is then calculated using FLOPs $= F_{\text{SFT}}(t_b) + F_{\text{RL}}(T_{\text{RL}})$, where the total *training FLOPs* are dominated by matrix multiplications, with costs scaling linearly with model parameters, sequence length, and optimizer momentum buffers.

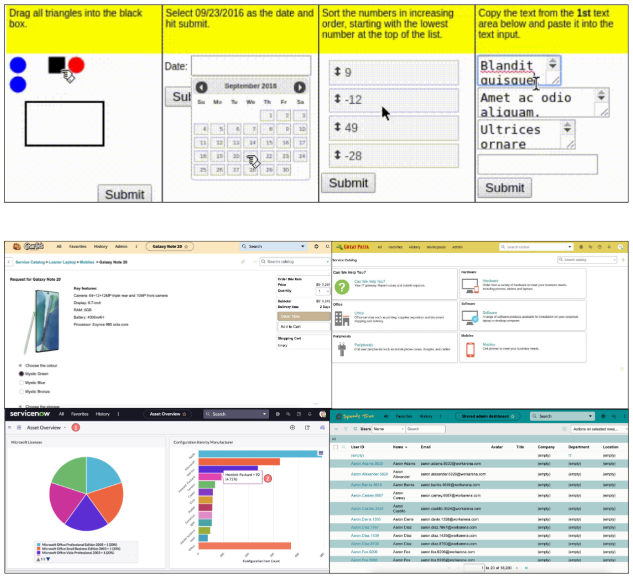

Drag all triangles into the black box.
Select 09/23/2016 as the date and hit submit.
Sort the numbers in increasing order, starting with the lowest number at the top of the list.
Copy the text from the 1st text area below and paste it into the text input.

Figure 2: Example tasks in MiniWoB++ (Liu et al., 2018) (top) and WorkArena (Drouin et al., 2024) (bottom). MiniWoB consists of single-page simple tasks such as selecting a particular date and using a basic text editor, while WorkArena comprises multi-page complex tasks like filling forms and placing orders in an enterprise environment.

We vary the SFT-to-RL ratio by branching out of SFT at different checkpoints. This exposes trade-offs between expensive but high-quality expert supervision and cheap but noisy on-policy exploration.

3.1. Estimating the Uncertainty of the Hyperparameter Selection Process

Across the different SFT checkpoints, we sample 240 distinct configurations with ten varying hyperparameters, (see Section 6 for details).

Our objective is to study the effect of various hyperparameters (HP) on the downstream success rate of our trained agents. This comes with two important considerations. First, if we change the value of, e.g., the batch size, and we want to know if a bigger batch size is better, the learning rate and other parameters need to be readjusted close to their optimal configuration (under a fixed budget). Secondly, to account for noise, we would need to restart the same experiment several times to avoid spurious conclusions. In practice, this is out of reach. For a more computationally friendly approach, we resort to bootstrapping the collection of trials over different hyperparameter configurations.

Bootstrapping the hyperparameter selection process. From the full set of 240 runs, we resample *trials* (with replacement). For each value, we select the configuration with the best validation score, and repeat this 1,000 times. We also extract the fraction of times each HP value wins, which serves as our estimated probability of it being part of the global optimum. This also provides uncertainty estimates for the validation and test metrics. For a more in-depth analysis, we also study this selection process on subsets of the complete trials by fixing the value of some parameters. For example, in Figure 6, we study the behavior of HPs in isolation for different starting checkpoints with varying amounts of SFT.

Balancing unequal coverage. Due to random search, some HP values were explored more than others, biasing the winner toward the larger groups. To correct for this, each trial is sampled with probability $\propto 1/\text{group size}$, approximating an equal compute budget for every HP value.

4. Experimental Setup

We evaluate our approach using Llama 3.3 70B as the expert model to generate demonstration traces, and Llama 3.1 8B as the student model for fine-tuning. Both models operate with a 16k token context window to handle the complexity of web-based tasks.

Our experiments focus on two benchmarks. The first is MiniWoB++, a suite of 30 medium-horizon tasks designed for web interaction, where an optimal policy typically requires 2 to 5 steps per task. The second is WorkArena (Drouin et al., 2024), a more challenging benchmark comprising 33 enterprise knowledge-work tasks, where an optimal policy typically requires 3 to 10 steps. These benchmarks provide a representative spectrum of sequential decision-making challenges faced by interactive LLM agents. Both benchmarks are depicted in Figure 2.

For the observation space, MiniWoB++ provides raw HTML trees, while WorkArena leverages accessibility trees (AxTrees), which we truncate to 16k tokens to meet hardware constraints. The agent operates in a discrete action space composed of high-level UI primitives: `noop`, `fill(node, text)`, `click(node)`, `type(node, text)`, `select_option(node, option)`, `scroll(node)`, and `hover(node)`. This abstraction allows the agent to interact effectively with diverse web interfaces. All agents are using Chain-of-thought prompting (Wei et al., 2023). We also experiment with *error log feedback*, allowing the agent to receive explicit error messages when it takes invalid actions.

To manage the training pipeline, we use BROWSERGYM (de Chezelles et al., 2025) for orchestrating Chromium-based web environments, structuring the agent's action space and we use AGENTLAB (de Chezelles et al., 2025) for

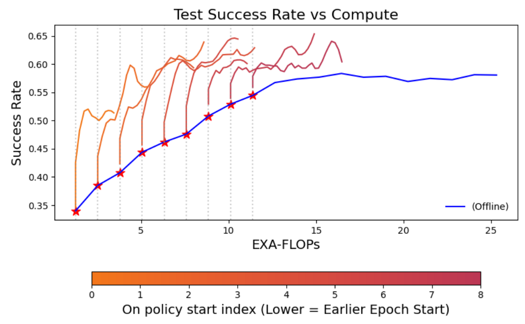


Figure 3: **MiniWoB++ compute analysis** — Test task performance w.r.t total FLOPs. Branching off early from SFT to RL yields superior performance under budget constraints.

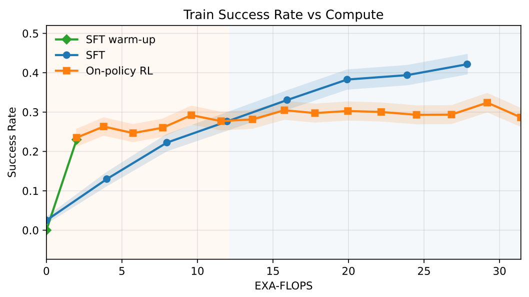


Figure 4: **WorkArena compute analysis** — Despite task difficulty, early RL branching remains valuable for efficient compute usage. Two seeds for each runs are reported using a ema filter with an alpha = 0.7 for smoothing.

agent design. Model fine-tuning is conducted with TORCHTUNE, utilizing Fully Sharded Data Parallelism (FSDP) to enable scalable training across multiple GPUs. Given the high memory demands of long-sequence processing, we apply activation offloading and gradient checkpointing techniques, achieving approximately 40% reduction in memory usage.

Our computational infrastructure comprises 8 × H100-80GB GPUs for expert data generation with the 70B model. For student model training, we allocate 2 × H100 GPUs for MiniWoB++ experiments and 4 × H100 GPUs for WorkArena experiments, reflecting the increased complexity of the latter.

The environments we use in this study are stochastic, hence each task is initialized with a random seed. We define a "goal" as a specific random seed for a given task. The evaluation protocol is designed to assess generalization at two levels: (i) performance on held-out goals within the training tasks, and (ii) performance on entirely new, unseen tasks. The primary metric reported is the average task success rate, which reflects the agent's ability to generalize beyond its training distribution.

To ensure a fair comparison of training strategies, we account for compute efficiency by estimating the total floating-point operations (FLOPs) consumed during both supervised fine-tuning (SFT) and reinforcement learning (GRPO) phases, described in Section 3 and defined in Appendix B.

5. Main Results and Compute Trade-Offs

In this section, we present our primary findings and analyze the trade-offs between supervised fine-tuning (SFT) and reinforcement learning (RL) in terms of both performance and compute efficiency.

Performance overview Table 1 summarizes the headline results across benchmarks. We selected the top-performing runs based on the area under the curve (AUC) of success rates on held-out training goals to mitigate overfitting risks. For each method (SFT, RL, and SFT+RL), we report the mean performance of the top two seeds at convergence.

For the MiniWoB++ benchmark, combining SFT with RL consistently outperforms SFT-only runs, matching the teacher model's performance. Conversely, on the more challenging WorkArena benchmark, SFT-only approaches are generally more effective. RL runs initiated without warm-starting failed to achieve meaningful improvements, highlighting the difficulty of bootstrapping learning from low initial success rates.

Further analysis of saturation behavior for both SFT and SFT+RL is provided at the end of this section. Notably, WorkArena's test set can be easier than its training set for certain agents, adding nuance to the observed performance gaps.

Compute-Performance Trade-off Across Benchmarks The inherent trade-off between high-quality but expensive SFT data and cheaper, noisier on-policy RL samples underpins our analysis. To explore this balance, we conducted a series of experiments where RL fine-tuning branches off from SFT runs at regular intervals (every 1024 SFT training samples).

Figure 3 illustrates this trade-off for both MiniWoB++ and Figure 4 for WorkArena. For MiniWoB++, branching off early into RL not only yields better performance for lower compute budgets but can also surpass SFT-only results. In particular, dedicating roughly one-third of the total FLOPs to teacher traces and the remainder to RL lifts MiniWoB++'s success by +6 percentage points over SFT-only training at identical cost, matching teacher performance. This suggests that self-generated rollouts are crucial for maximizing agent performance, with early branching pushing the compute-performance Pareto front. The warm-started RL runs converge to a performance ceiling unattainable by RL-only approaches, underscoring the synergy between SFT and RL.

Table 1: Comparison of our method with baselines on the held-out train and test splits of WorkArena and MiniWoB++. Baselines are run in the exact same setting as (de Chezelles et al., 2025).

Model	WorkArena		MiniWoB++	
	Train-held-out	Test-held-out	Train-held-out	Test-held-out
Claude-3.5-Sonnet	48.1±3.1	87.1±4.0	70.5±2.0	70.4±4.3
GPT-4o	41.2±3.1	61.4±5.8	65.7±2.1	64.3±4.5
GPT-4o-Mini	23.1±2.6	41.4±5.9	56.2±2.2	66.1±4.4
Llama-3.1-70b-Instruct	25.0±2.7	38.6±5.8	57.0±2.2	65.2±4.4
o1-Mini	49.2±3.1	84.3±4.3	69.7±2.1	66.1±4.4
Llama-3.1-405b-Instruct	35.8±3.0	71.4±5.4	65.9±2.1	65.2±4.4
Llama-3.1-8B *(Student)*	9.0±2.9	5.0±2.2	29.5±3.2	36.4±3.4
Llama-3.1-8b-SFT *(Ours)*	25.0±4.4	39.0±4.8	53.3±2.5	57.8±2.5
Llama-3.1-8b-RL *(Ours)*	0.0±0.0	0.0±0.0	44.3±2.5	48.5±2.5
Llama-3.1-8b-SFT+RL *(Ours)*	24.0±4.2	32.0±4.6	**63.8±2.4**	**63.4±2.4**
Llama-3.3-70b *(Teacher)*	**30.0±4.6**	**50.0±5.0**	63.2±4.8	61.9±4.9

In contrast, WorkArena's greater task complexity posed significant challenges. RL trials struggled to produce a Pareto front akin to MiniWoB++, limiting direct compute-performance trade-off analysis. Nevertheless, early RL branching from SFT still holds promise for lower compute budgets, even if RL runs did not significantly surpass SFT-only baselines. This outcome highlights the need for more advanced RL strategies or alternative optimization methods for high-complexity, sparse-reward tasks.

These insights emphasize the importance of balancing sample efficiency (quality of expert data) with compute efficiency (on-policy exploration), and could inform future research on scalable LLM agent training.

Task Performance Saturation and Analysis Despite extensive post-training, agent performance on the WorkArena benchmark plateaus at around 40% after just 9–10 epochs. This stagnation appears to stem from the intrinsic difficulty of certain tasks—such as sorting and filtering—which even the Llama 3.3 70B teacher model struggles to solve (see Figure 5). A per-task breakdown shows that while both SFT and RL agents gradually close the performance gap with the teacher model, with RL achieving a slightly higher final success rate, a significant portion of tasks (14 out of 33) remain completely unsolved. These failures are attributed to either the limitations of the teacher model or the sparsity of reward signals, both of which hamper the learning process. On-policy RL exploration proves ineffective in overcoming these challenges due to the lack of foundational skills and informative feedback. These findings underscore the need for more effective methods to address complex tasks under sparse reward settings. Additional per-task performance results for WorkArena and Miniwob are provided in Appendix A.

6. Ablation and Sensitivity Analysis

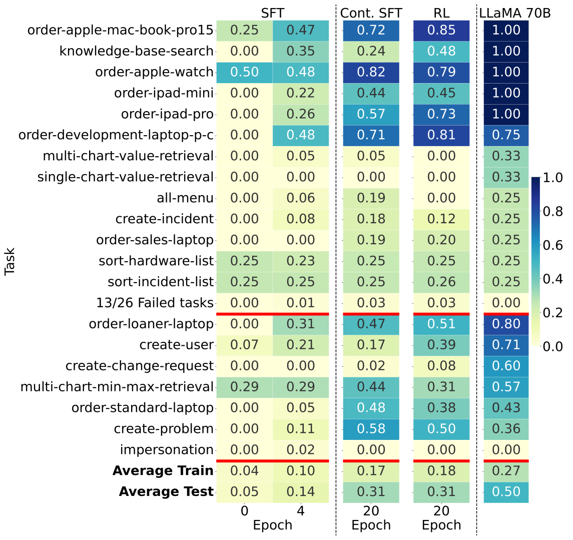
We simulate re-running hyperparameter configurations and selecting the best-performing ones using the method described in Section 3.1. This is done across three checkpoints: the base LLaMA 3.1 8B Instruct model and two warm-started variants with an additional 2.5×10^{18} and 7.6×10^{18} FLOPs of supervised fine-tuning, respectively, to assess variations across compute budgets. We evaluate the test and training performance of 10 hyper-parameters across 240 runs. We report final test performance to verify generalization in Appendix D finding no significant deviations between test and train parameters.

Figure 6 displays our findings, which we summarize as follows. *Curriculum learning* is beneficial when starting RL from scratch but becomes detrimental after warm-starting, likely because warm-started models already perform well on easy tasks, and curriculum forces them to over-focus on harder ones. *Error log feedback* helps when there's no SFT but hurts after pretraining—probably because pretrained models rarely make invalid moves and the extra signals become overwhelming. A *decoding temperature* of 0.25 consistently yields the best results, striking a balance between exploration and exploitation; lower values led to under-exploration and were discarded. *Grouped-relative advantage* helps only after SFT, while using raw rewards works better when starting directly from the Instruct model, possibly due to how advantage scaling interacts with the initial weights. *Zero-advantage filtering* improves training across all settings by ensuring batches focus on informative updates. *Standard-deviation normalized advantage*, as noted by (Liu et al., 2025), consistently hurts performance, especially when models receive less SFT budget. *Importance ratio correction and trust region*, though standard, also hurt models with little or no SFT, likely because conservative updates slow down learning. In contrast, for models that start from a stronger SFT checkpoint, these mechanisms can help stabilize training and avoid catastrophic updates. For the *learning rate*, larger values generally work better, except on the test set for models trained without SFT, where more conservative updates improve generalization. *Effective batch size* shows minimal impact across the board; a batch size of 512 is a safe and robust choice in all our experiments. Finally, regarding the *discount rate*, we find that 0.9 works well in most settings, except for heavily warm-started models, where a lower rate of 0.5 appears beneficial—likely because it encourages the model to optimize aggressively on tasks it already performs well on.

7. Related Work

Best precices in deep RL. Building on the recognition of reproducibility challenges and unstable RL training of LLM agents, recent studies have proposed best practices for training LLM agents using RL methods. Dang and Ngo (2025) recommend leveraging high quality data, balancing easy and hard problems, and controlling length generation with cosine reward. Yu et al. (2025) promote higher clipping in the GRPO loss to promote diversity and avoid entropy collapse, dynamic sampling to improve training efficiency and stability, token level gradients for long CoT sequences, and overlong reward shaping to reduce reward noise. Roux et al. (2025) introduce tapered variant of importance sampling to speed up learning while maintaining stable learning dynamics. The proposed method (TOPR) allows the handling of both positive and negative examples in a fully offline setting. More generally, Hochlehnert et al. (2025) emphasizes the need for greater methodological precision, particularly concerning decoding parameters, random seeds, prompt formatting, as well as the hardware and software frameworks, to guarantee transparent and thorough assessments of model performance. These practices are essential for developing robust and reproducible agents.

LLM Agents trained with RL on multi-step environments. Recent advancements have sought to bridge the gap in training LLM agents for multi-step environments, with approaches like WebRL (Qi et al., 2025) and SWEET-RL (Zhou et al., 2025) demonstrating significant progress. WebRL employs a self-evolving curriculum to address the challenges of sparse feedback and task scarcity, successfully enhancing the performance of open LLMs in web-based tasks (Qi et al., 2025). Similarly, SWEET-RL introduces a hierarchical structure that enables effective credit assignment over multiple turns, improving policy learning and generalization in collaborative reasoning tasks (Zhou et al., 2025). These studies collectively illustrate the necessity of adapting RL techniques to accommodate the complexities of multi-step interactions, paving the way for more capable and versatile LLM agents. An extended version of the related work is provided in Appendix C.



Task	SFT 0	SFT 4	Cont. SFT 20	RL 20	LLaMA 70B
order-apple-mac-book-pro15	0.25	0.47	0.72	0.85	1.00
knowledge-base-search	0.00	0.35	0.24	0.48	1.00
order-apple-watch	0.50	0.48	0.82	0.79	1.00
order-ipad-mini	0.00	0.22	0.44	0.45	1.00
order-ipad-pro	0.00	0.26	0.57	0.73	1.00
order-development-laptop-p-c	0.00	0.48	0.71	0.81	0.75
multi-chart-value-retrieval	0.00	0.05	0.05	0.00	0.33
single-chart-value-retrieval	0.00	0.00	0.00	0.00	0.33
all-menu	0.00	0.06	0.19	0.00	0.25
create-incident	0.00	0.08	0.18	0.12	0.25
order-sales-laptop	0.00	0.00	0.19	0.20	0.25
sort-hardware-list	0.25	0.23	0.25	0.25	0.25
sort-incident-list	0.25	0.25	0.25	0.26	0.25
13/26 Failed tasks	0.00	0.01	0.03	0.03	0.00
order-loaner-laptop	0.00	0.31	0.47	0.51	0.80
create-user	0.07	0.21	0.17	0.39	0.71
create-change-request	0.00	0.00	0.02	0.08	0.60
multi-chart-min-max-retrieval	0.29	0.29	0.44	0.31	0.57
order-standard-laptop	0.00	0.05	0.48	0.38	0.43
create-problem	0.00	0.11	0.58	0.50	0.36
impersonation	0.00	0.02	0.00	0.00	0.00
Average Train	0.04	0.10	0.17	0.18	0.27
Average Test	0.05	0.14	0.31	0.31	0.50

Figure 5: Per-task performance of SFT and SFT+RL agents on WorkArena. The Llama 3.1 8B model is initially fine-tuned for 4 epochs on trajectories from a teacher Llama 3.3 70B model. Training then continues either with additional SFT or with GRPO fine-tuning up to epoch 20. The teacher model's success rate is also shown.

8. Discussion

Limitations. Our focus is on providing a comprehensive perspective on training an LLM-based web agent, studying compute trade-offs, hyperparameter selection, and analyzing failure cases. With this in mind, our results are limited to English-language web interfaces and Llama 3 models in the 8B–70B parameter range, where larger models may alter trade-offs.

Regarding our statistical method, it does not account for the lack of coverage from the random search. A more exhaustive search could discover configurations that would change the conclusions drawn in this study. We note also that a significant portion of the reported uncertainty is due to epistemic uncertainty that could be reduced by evaluating more configurations.

Conclusion. Overall, our study provides a practical and statistically grounded recipe for training open-source LLM agents on complex web tasks, helping close the performance gap with proprietary systems while promoting reproducibility and efficiency. By systematically analyzing compute allocation strategies, hyperparameter sensitivities, and training dynamics, we offer actionable insights that can inform future research and development of interactive agents. We hope these findings will serve both as a reproducible baseline and as a stepping stone for advancing reliable and cost-effective LLM agent training in web-based and other complex sequential environments.

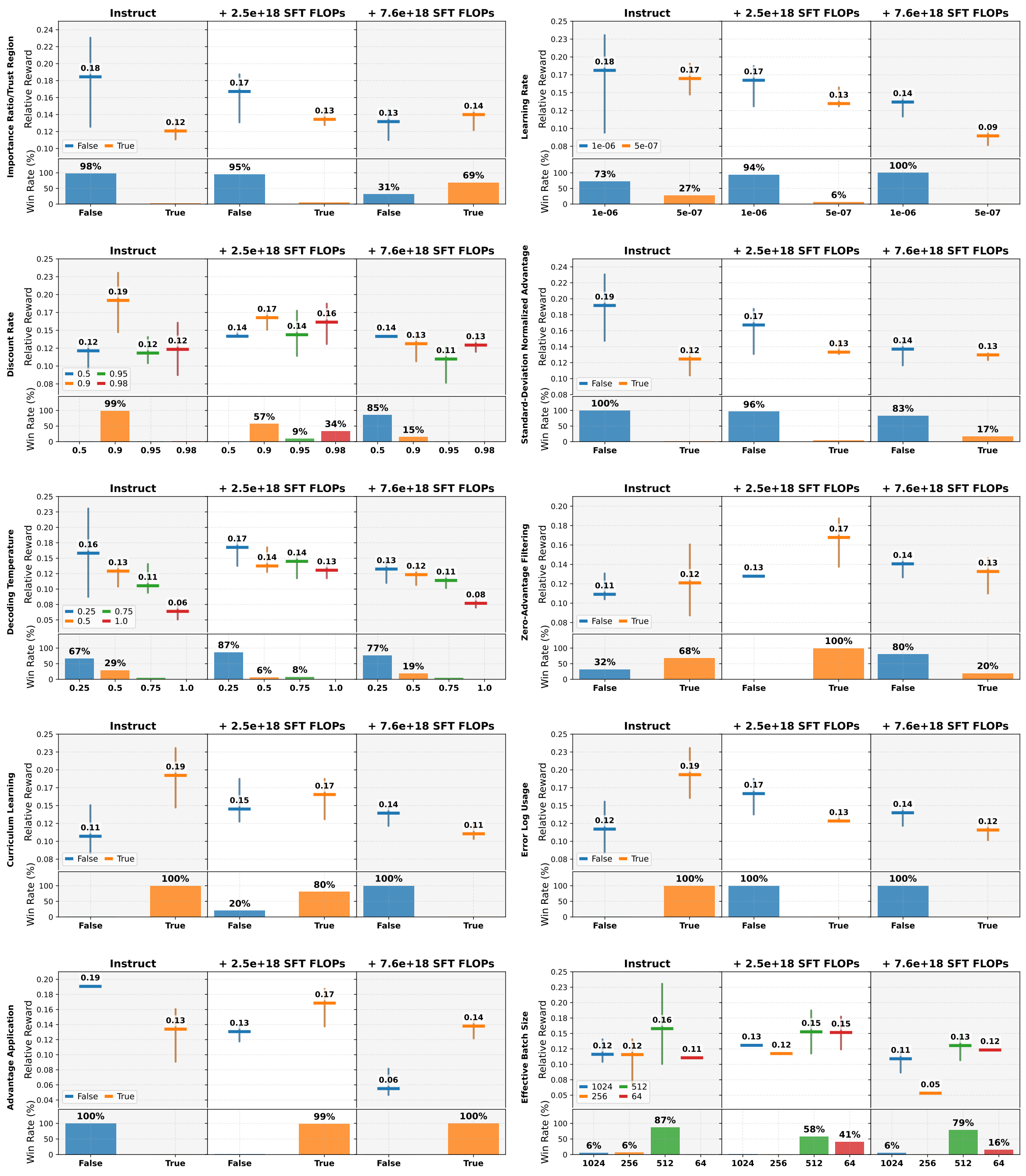


Figure 6: Bootstrap analysis ($n = 1000$ samples) of hyperparameter optimization across different SFT compute budgets on *training* held out tasks. Each subplot examines a different hyperparameter, including increasing SFT compute: the base instruct model (left), +2.5$\times 10^{18}$ SFT FLOPs (middle), and +7.6e$\times 10^{18}$ SFT FLOPs (right). For each hyperparameter-compute combination, the top panel shows relative reward performance with error bars indicating 95% confidence intervals, while the bottom panel displays win rates representing the percentage of bootstrap iterations where each parameter value achieved maximum performance. Results demonstrate that optimal hyperparameter values shift as model pre-training compute increases, suggesting that hyperparameter selection should be adapted to the computational budget allocated to supervised fine-tuning.

References

Dgxc benchmarking. URL https://catalog.ngc.nvidia.com/orgs/nvidia/teams/dgxc-benchmarking/resources/llama31-8b-dgxc-benchmarking-a.

M. Abdin, S. Agarwal, A. Awadallah, V. Balachandran, H. Behl, L. Chen, G. de Rosa, S. Gunasekar, M. Javaheripi, N. Joshi, P. Kauffmann, Y. Lara, C. C. T. Mendes, A. Mitra, B. Nushi, D. Papailiopoulos, O. Saarikivi, S. Shah, V. Shrivastava, V. Vineet, Y. Wu, S. Yousefi, and G. Zheng. Phi-4-reasoning technical report, 2025. URL https://arxiv.org/abs/2504.21318.

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

L. Boisvert, M. Thakkar, M. Gasse, M. Caccia, T. L. S. D. Chezelles, Q. Cappart, N. Chapados, A. Lacoste, and A. Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks, 2024. URL https://arxiv.org/abs/2407.05291.

Q.-A. Dang and C. Ngo. Reinforcement learning for reasoning in small llms: What works and what doesn't. *arXiv preprint arXiv:2503.16219*, 2025. URL https://arxiv.org/abs/2503.16219.

T. L. S. de Chezelles, M. Gasse, A. Lacoste, M. Caccia, A. Drouin, L. Boisvert, M. Thakkar, T. Marty, R. Assouel, S. O. Shayegan, L. K. Jang, X. H. Lù, O. Yoran, D. Kong, F. F. Xu, S. Reddy, G. Neubig, Q. Cappart, R. Salakhutdinov, and N. Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

A. Drouin, M. Gasse, M. Caccia, I. H. Laradji, M. D. Verme, T. Marty, L. Boisvert, M. Thakkar, Q. Cappart, D. Vazquez, N. Chapados, and A. Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024.

A. Hochlehnert, H. Bhatnagar, V. Udandarao, S. Albanie, A. Prabhu, and M. Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility, 2025. URL https://arxiv.org/abs/2504.07086.

J. Y. Koh and et al. Visualwebarena: Evaluating multimodal agents on realistic visually grounded web tasks. In *ACL 2024*, 2024. URL https://aclanthology.org/2024.acl-long.50.

E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1802.08802.

Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. S. Lee, and M. Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

S. Murty and et al. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *arXiv preprint arXiv:2410.02907*, 2025. URL https://arxiv.org/abs/2410.02907.

Z. Qi, X. Liu, I. L. Iong, H. Lai, X. Sun, J. Sun, X. Yang, Y. Yang, S. Yao, W. Xu, J. Tang, and Y. Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

N. L. Roux, M. G. Bellemare, J. Lebensold, A. Bergeron, J. Greaves, A. Fréchette, C. Pelletier, E. Thibodeau-Laufer, S. Toth, and S. Work. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025. URL https://arxiv.org/abs/2503.14286.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

M. Thakkar, T. Bolukbasi, S. Ganapathy, S. Vashishth, S. Chandar, and P. Talukdar. Self-influence guided data reweighting for language model pre-training. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=rXn9WO4M2p.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

T. Xie, D. Zhang, J. Chen, X. Li, S. Zhao, R. Cao, T. J. Hua, Z. Cheng, D. Shin, F. Lei, Y. Liu, Y. Xu, S. Zhou, S. Savarese, C. Xiong, V. Zhong, and T. Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

F. F. Xu, Y. Song, B. Li, Y. Tang, K. Jain, M. Bao, Z. Z. Wang, X. Zhou, Z. Guo, M. Cao, M. Yang, H. Y. Lu, A. Martin, Z. Su, L. Maben, R. Mehta, W. Chi, L. Jang, Y. Xie, S. Zhou, and G. Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2024. URL https://arxiv.org/abs/2412.14161.

Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, W. Dai, Y. Song, X. Wei, H. Zhou, J. Liu, W.-Y. Ma, Y.-Q. Zhang, L. Yan, M. Qiao, Y. Wu, and M. Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, and G. Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://arxiv.org/abs/2307.13854.

Y. Zhou, S. Jiang, Y. Tian, J. Weston, S. Levine, S. Sukhbaatar, and X. Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks, 2025. URL https://arxiv.org/abs/2503.15478.

Broader Impact

Web-based LLM agents have the potential to revolutionize markets by enabling more cost-efficient and effective workflows. Our work focuses on making these agents accessible across a range of compute budgets, empowering not only industrial labs but also smaller research groups and individuals to train their own assistants. This approach promotes data privacy and reduces reliance on costly infrastructure.

Despite their promise, web-based agents face significant challenges that limit their broader adoption. Issues such as reliability, vulnerability to adversarial attacks, and limited access to proprietary data remain key obstacles to realizing their full potential.

Impact Statement

Authors are **required** to include a statement of the potential broader impact of their work, including its ethical aspects and future societal consequences. This statement should be in an unnumbered section at the end of the paper (co-located with Acknowledgements – the two may appear in either order, but both must be before References), and does not count toward the paper page limit. In many cases, where the ethical impacts and expected societal implications are those that are well established when advancing the field of Machine Learning, substantial discussion is not required, and a simple statement such as the following will suffice:

“This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.”

The above statement can be used verbatim in such cases, but we encourage authors to think about whether there is content which does warrant further discussion, as this statement will be apparent if the paper is later flagged for ethics review.

References

Dgxc benchmarking. URL https://catalog.ngc.nvidia.com/orgs/nvidia/teams/dgxc-benchmarking/resources/llama31-8b-dgxc-benchmarking-a.

M. Abdin, S. Agarwal, A. Awadallah, V. Balachandran, H. Behl, L. Chen, G. de Rosa, S. Gunasekar, M. Javaheripi, N. Joshi, P. Kauffmann, Y. Lara, C. C. T. Mendes, A. Mitra, B. Nushi, D. Papailiopoulos, O. Saarikivi, S. Shah, V. Shrivastava, V. Vineet, Y. Wu, S. Yousefi, and G. Zheng. Phi-4-reasoning technical report, 2025. URL https://arxiv.org/abs/2504.21318.

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

L. Boisvert, M. Thakkar, M. Gasse, M. Caccia, T. L. S. D. Chezelles, Q. Cappart, N. Chapados, A. Lacoste, and A. Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks, 2024. URL https://arxiv.org/abs/2407.05291.

Q.-A. Dang and C. Ngo. Reinforcement learning for reasoning in small llms: What works and what doesn't. *arXiv preprint arXiv:2503.16219*, 2025. URL https://arxiv.org/abs/2503.16219.

T. L. S. de Chezelles, M. Gasse, A. Lacoste, M. Caccia, A. Drouin, L. Boisvert, M. Thakkar, T. Marty, R. Assouel, S. O. Shayegan, L. K. Jang, X. H. Lù, O. Yoran, D. Kong, F. F. Xu, S. Reddy, G. Neubig, Q. Cappart, R. Salakhutdinov, and N. Chapados. The browsergym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=5298fKGmv3. Expert Certification.

DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

A. Drouin, M. Gasse, M. Caccia, I. H. Laradji, M. D. Verme, T. Marty, L. Boisvert, M. Thakkar, Q. Cappart, D. Vazquez, N. Chapados, and A. Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024.

A. Hochlehnert, H. Bhatnagar, V. Udandarao, S. Albanie, A. Prabhu, and M. Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility, 2025. URL https://arxiv.org/abs/2504.07086.

J. Y. Koh and et al. Visualwebarena: Evaluating multimodal agents on realistic visually grounded web tasks. In *ACL 2024*, 2024. URL https://aclanthology.org/2024.acl-long.50.

E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1802.08802.

Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. S. Lee, and M. Lin. Understanding r1-zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

S. Murty and et al. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *arXiv preprint arXiv:2410.02907*, 2025. URL https://arxiv.org/abs/2410.02907.

Z. Qi, X. Liu, I. L. Iong, H. Lai, X. Sun, J. Sun, X. Yang, Y. Yang, S. Yao, W. Xu, J. Tang, and Y. Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

N. L. Roux, M. G. Bellemare, J. Lebensold, A. Bergeron, J. Greaves, A. Fréchette, C. Pelletier, E. Thibodeau-Laufer, S. Toth, and S. Work. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025. URL https://arxiv.org/abs/2503.14286.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

M. Thakkar, T. Bolukbasi, S. Ganapathy, S. Vashishth, S. Chandar, and P. Talukdar. Self-influence guided data reweighting for language model pre-training. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=rXn9WO4M2p.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

T. Xie, D. Zhang, J. Chen, X. Li, S. Zhao, R. Cao, T. J. Hua, Z. Cheng, D. Shin, F. Lei, Y. Liu, Y. Xu, S. Zhou, S. Savarese, C. Xiong, V. Zhong, and T. Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

F. F. Xu, Y. Song, B. Li, Y. Tang, K. Jain, M. Bao, Z. Z. Wang, X. Zhou, Z. Guo, M. Cao, M. Yang, H. Y. Lu, A. Martin, Z. Su, L. Maben, R. Mehta, W. Chi, L. Jang, Y. Xie, S. Zhou, and G. Neubig. Theagentcompany: Benchmarking llm agents on consequential real world tasks, 2024. URL https://arxiv.org/abs/2412.14161.

Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, W. Dai, Y. Song, X. Wei, H. Zhou, J. Liu, W.-Y. Ma, Y.-Q. Zhang, L. Yan, M. Qiao, Y. Wu, and M. Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried, U. Alon, and G. Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL https://arxiv.org/abs/2307.13854.

Y. Zhou, S. Jiang, Y. Tian, J. Weston, S. Levine, S. Sukhbaatar, and X. Li. Sweet-rl: Training multi-turn llm agents on collaborative reasoning tasks, 2025. URL https://arxiv.org/abs/2503.15478.

A. You *can* have an appendix here.

You can have as much text here as you want. The main body must be at most 8 pages long. For the final version, one more page can be added. If you want, you can use an appendix like this one.

The `\onecolumn` command above can be kept in place if you prefer a one-column appendix, or can be removed if you prefer a two-column appendix. Apart from this possible change, the style (font size, spacing, margins, page numbering, etc.) should be kept the same as the main body.

A. Extended Learning and Saturation Analysis

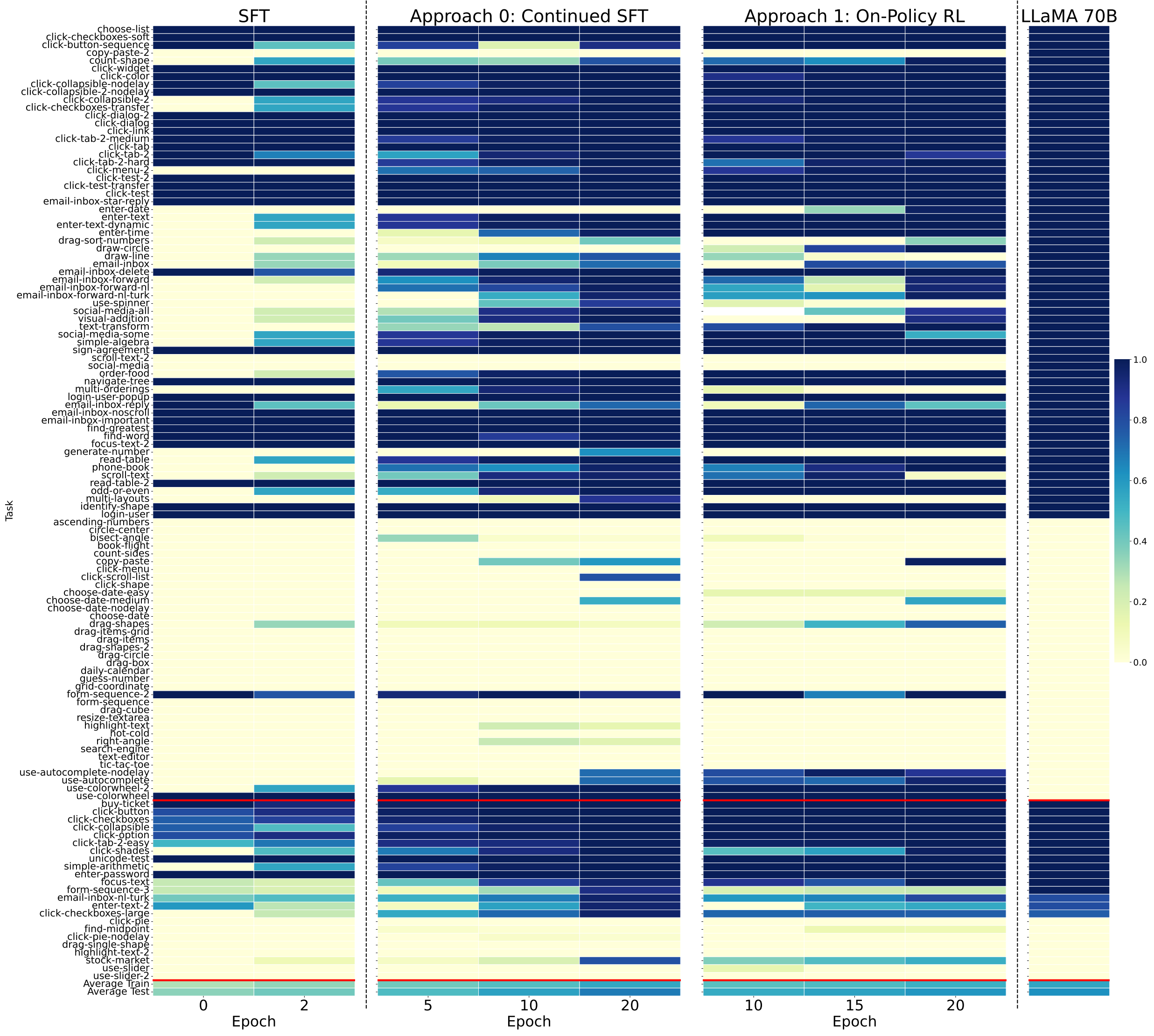


Figure 7: Per task performance of SFT and SFT+RL agents on MiniWob++.

Challenges in Agent-Environment Interaction In this section we talk about the general challenges faced by the agent to interact effectively with the environment.

- **Observation/action space mismatch**: One of the important thing to note specifically in our web environment is the observation space which the agent uses is a bit different from the action space. Multiple times, the agent can see the correct action in the AxTree but the action space, the icon is not visible and to make it visible, the agent needs to scroll

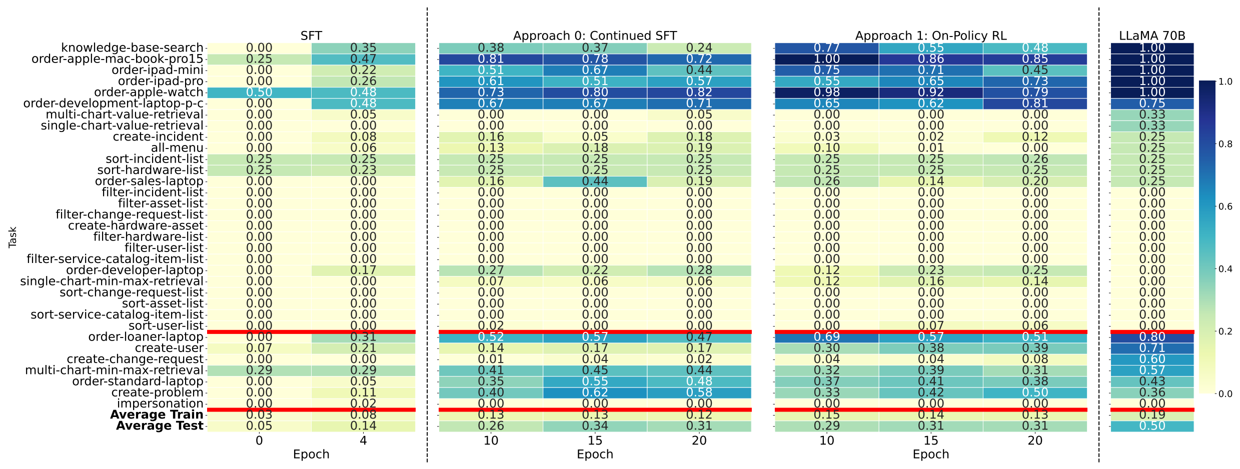


Figure 8: Per task performance of SFT and SFT+RL agents on WorkArena.

down and do the action. This mismatch causes huge problems (Koh and et al., 2024)

- **UI Misuse**: The agent tries to interact with items in the environment in ways that it is not designed. For example, the agent trying to fill in a checkbox with value True while it should just click on it. (Murty and et al., 2025)
- **Repeating actions**: A common issue we observed is the repetition of actions across multiple consecutive steps, often accompanied by verbose and redundant chains of thought. The agent frequently restates similar thoughts or re-executes the same actions unnecessarily, leading to inefficiencies and sometimes getting stuck in loops. (Murty and et al., 2025).

B. Deriving Compute Cost

FLOPs Estimation Methodology Flop calculations are based on model architecture, token counts, and average sequence lengths observed during training and evaluation.

FLOPs per Token

We estimate FLOPs per token using the following formula, adapted from nvidia benchmarking(ben):

$$\text{FLOPs}_{\text{per token}} = (\text{FLOPs}_{\text{attn}} + \text{FLOPs}_{\text{MLP}} + \text{FLOPs}_{\text{embed}}) \times (1 + \text{backward multiplier}) \tag{4}$$

Where:

$$\text{FLOPs}_{\text{attn}} = 12 \times (\text{number of layers}) \times (\text{hidden size})^2 \times \left(1 + \frac{\text{number of query groups}}{\text{number of attention heads}} + \frac{\text{sequence length}}{\text{hidden size}}\right) \tag{5}$$

$$\text{FLOPs}_{\text{MLP}} = 18 \times (\text{number of layers}) \times (\text{hidden size}) \times \text{FFN} \tag{6}$$

$$\text{FLOPs}_{\text{embed}} = 6 \times \text{vocabulary size} \times (\text{hidden size}) \tag{7}$$

On-Policy FLOPs (LLaMA-8B)

We compute the total FLOPs for each on-policy epoch by summing the training and testing FLOPs:

$$\text{FLOPs}_{\text{train}} = N_{\text{train}} \times \text{FLOPs}_{\text{per token}}^{(\text{backward}=3)} \tag{8}$$

$$\text{FLOPs}_{\text{test}} = N_{\text{test}} \times \text{FLOPs}_{\text{per token}}^{(\text{backward}=0)} \tag{9}$$

$$\text{FLOPs}_{\text{epoch}} = \text{FLOPs}_{\text{train}} + \text{FLOPs}_{\text{test}} \tag{10}$$

Where N_{train} and N_{test} are the number of tokens used for training and evaluation respectively. Sequence length S is measured per epoch from logged metrics.

Offline FLOPs (Generation: LLaMA-70B, Training: LLaMA-8B)

Offline training includes two compute components:

- **Data Generation** (LLaMA-70B, forward-only):

$$\text{FLOPs}_{\text{gen}} = N_{\text{gen}} \times \text{FLOPs}_{\text{per token}}^{(70B,\text{backward}=0)} \tag{11}$$

  where $N_{\text{gen}} =$ avg seq len $\times$ samples per epoch (from dataset metadata).

- **Training** (LLaMA-8B, with backward pass):

$$\text{FLOPs}_{\text{train}} = N_{\text{gen}} \times \text{FLOPs}_{\text{per token}}^{(8B,\text{backward}=3)} \tag{12}$$

The total FLOPs per offline epoch is:

$$\text{FLOPs}_{\text{epoch}} = \text{FLOPs}_{\text{gen}} + \text{FLOPs}_{\text{train}} \tag{13}$$

All FLOPs values are reported in **exaFLOPs** by dividing the total FLOPs by 10^{18}.

C. Extended Related Work

The Reproducibility Crisis in RL. The reproducibility crisis in large language models (LLMs) and reinforcement learning (RL) has garnered increasing attention, particularly due to the reliance on single seed results that distort the perceived performance of models. The reproducibility challenge[1] organized every year is a positive step towards addressing this. More concretely, Hochlehnert et al. (2025) provide a critical examination of how such practices undermine the reliability of published findings, revealing that many reported gains are sensitive to implementation choices, such as random seeds and prompt formatting (Hochlehnert et al., 2025).

Bandit-domain RLHF with LLMs. Previous work in RL for LLMs has predominantly focused on single-step tasks, which have shown effectiveness in mathematical reasoning and code generation (Yu et al., 2025; DeepSeek-AI et al., 2025; Roux et al., 2025). While these approaches exhibit promising results, they are limited in their applicability to real-world scenarios, which often require multistep decision-making capabilities. The narrow focus on bandit-style problems fails to address the complexities inherent in tasks that demand sequential interaction, highlighting a significant gap in the current research landscape.

Interactive Agent Benchmarks. To assess the capabilities of LLM agents in more realistic environments, benchmarks such as WebArena (Zhou et al., 2023), WorkArena (Drouin et al., 2024; Boisvert et al., 2024), the Agent Company (Xu et al., 2024), and OSWorld (Xie et al., 2024) have been designed to evaluate agents on multi-step tasks across various domains. These benchmarks expose the limitations of current LLM agents, revealing that while they may perform well in controlled settings, their performance in practical applications remains subpar, underscoring the need for further advancements in agent robustness and generalization to multi-step planning.

Best prectices in deep RL. Building on the recognition of reproducibility challenges and unstable RL training of LLM agents, recent studies have proposed best practices for training LLM agents using RL methods. Dang and Ngo (2025) recommend leveraging high quality data, balancing easy and hard problems, and controlling length generation with cosine reward. Yu et al. (2025) promote higher clipping in the GRPO loss to promote diversity and avoid entropy collapse, dynamic sampling to improve training efficiency and stability, token level gradients for long CoT sequences, and overlong reward shaping to reduce reward noise. Roux et al. (2025) introduce tapered variant of importance sampling to speed up learning while maintaining stable learning dynamics. The proposed method (TOPR) allows the handling of both positive and negative examples in a fully offline setting. More generally, Hochlehnert et al. (2025) emphasizes the need for greater methodological precision, particularly concerning decoding parameters, random seeds, prompt formatting, as well as the hardware and software frameworks, to guarantee transparent and thorough assessments of model performance.

[1] https://reproml.org/

LLM Agents trained with RL on multi-step environments. Recent advancements have sought to bridge the gap in training LLM agents for multi-step environments, with approaches like WebRL (Qi et al., 2025) and SWEET-RL (Zhou et al., 2025) demonstrating significant progress. WebRL employs a self-evolving curriculum to address the challenges of sparse feedback and task scarcity, successfully enhancing the performance of open LLMs in web-based tasks (Qi et al., 2025). Similarly, SWEET-RL introduces a hierarchical structure that enables effective credit assignment over multiple turns, improving policy learning and generalization in collaborative reasoning tasks (Zhou et al., 2025). These studies collectively illustrate the necessity of adapting RL techniques to accommodate the complexities of multi-step interactions, paving the way for more capable and versatile LLM agents.

D. Test Set Hyper-Parameter Bootstrap Analysis

We overall find similar results between the held-out train and test tasks with respect to optimal hyper-parameters. While we see no large deviations, we find that some parameters such as curriculum learning from the instruct model and using error logs can have a larger beneficial effect on the held-out testing tasks.

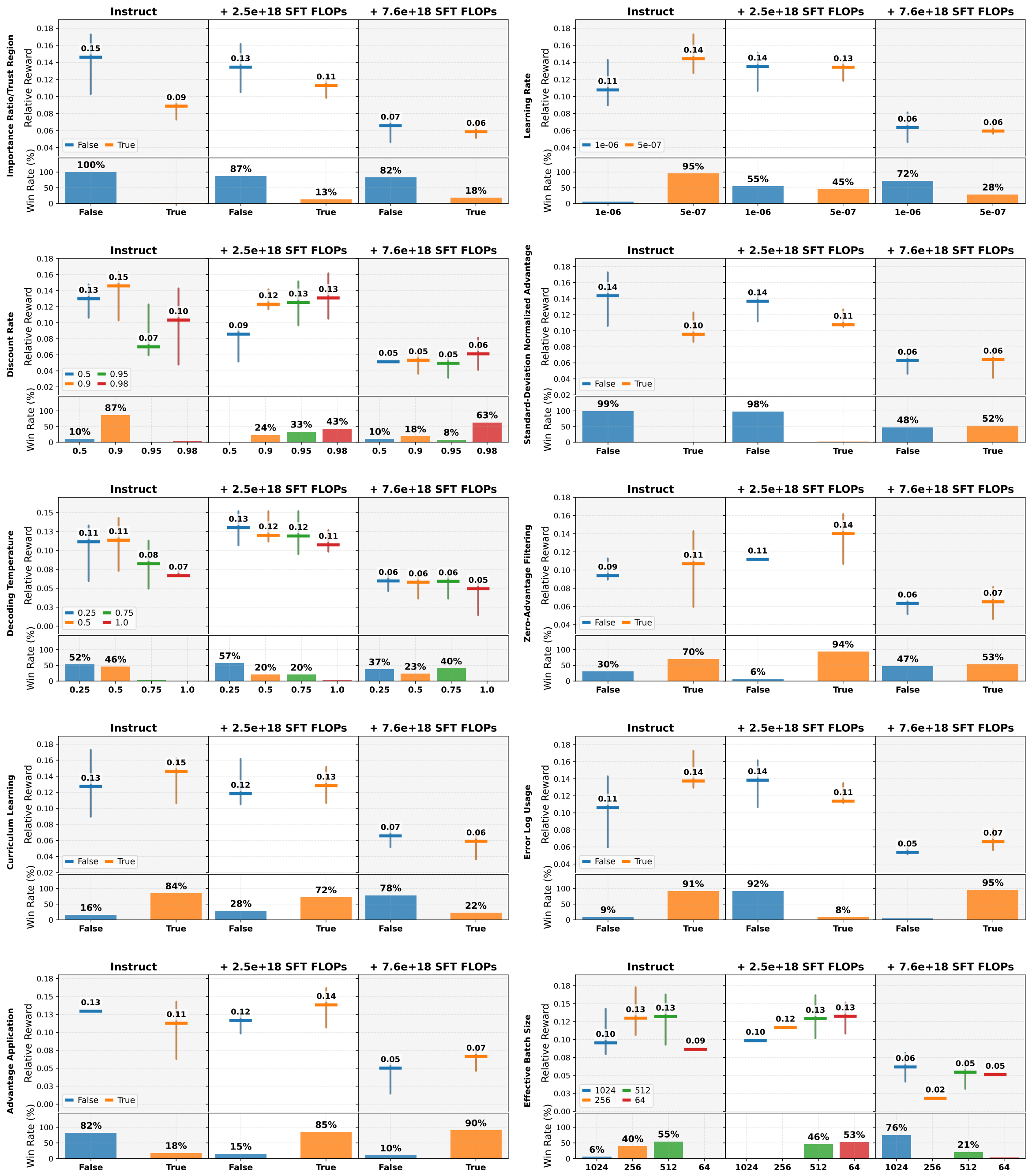


Figure 9: Bootstrap analysis ($n = 1000$ samples) of hyperparameter optimization across different SFT compute budgets on *test* held out tasks. Each subplot examines a different hyperparameter, including increasing SFT compute: the base instruct model (left), +2.5e+18 SFT FLOPs (middle), and +7.6e+18 SFT FLOPs (right). For each hyperparameter-compute combination, the top panel shows relative reward performance with error bars indicating 95% confidence intervals, while the bottom panel displays win rates representing the percentage of bootstrap iterations where each parameter value achieved maximum performance. Results demonstrate that optimal hyperparameter values often shift as model pre-training compute increases, suggesting that hyperparameter selection should be adapted based on the computational budget allocated to SFT.