AgentMerge: Enhancing Generalization in Fine-Tuned LLM Agents

Megh Thakkar[1,2] **Léo Boisvert**[1,2,3] **Thibault Le Sellier De Chezelles**[1,3]
Alexandre Piché[1] **Maxime Gasse**[1,2,3] **Alexandre Lacoste**[1] **Massimo Caccia**[1]
[1]ServiceNow Research [2]Mila – Quebec AI Institute [3] Polytechnique Montréal
{megh.thakkar, massimo.caccia}@servicenow.com

Abstract

Recent advancements in large language models (LLMs) have spurred interest in developing autonomous agents capable of performing complex tasks in a human-like manner. Despite progress, even the most advanced models often face challenges with robustness across benchmarks, while smaller, open-source models lag in performance. This study introduces a novel merging-based fine-tuning approach to enhance the capabilities of smaller, cost-efficient LLMs by combining agentic fine-tuning with instruction-tuning, using successful traces from stronger models as a guide. We outline a comprehensive pipeline for data collection, filtering, and supervised fine-tuning, examining key behavior cloning parameters. Our experiments reveal that simply predicting expert trajectories does not consistently improve task performance, highlighting issues like catastrophic forgetting and the loss of reasoning abilities. To address these challenges, we propose AgentMerge, model merging using agentic vectors as a solution, demonstrating its effectiveness in improving generalization and mitigating forgetting. Additionally, we provide an open-source codebase and a 140M-token dataset for the research community.

1 Introduction

Recent advancements in large language models (LLMs) have spurred interest in developing methods to improve performance on complex tasks through fine-tuning. Behavior cloning, where models learn from expert-generated data to replicate decision-making processes, has shown potential in enhancing efficiency and accuracy. However, fine-tuning still faces significant challenges, such as catastrophic forgetting, which leads to degradation of reasoning abilities learned during pretraining.

In this study, we introduce a novel method that addresses these challenges through model merging, AgentMerge. AgentMerge extends task vectors to 'agentic vectors' obtained from agentic fine-tuning. By combining agentic vectors with instruction-tuning, we aim to retain the strengths of both phases while minimizing catastrophic forgetting. We propose a new merging approach, where models fine-tuned on expert trajectories are merged with instruction-tuned models, thereby approximating the ideal scenario of simultaneous instruction and behavior cloning. Our method improves generalization while mitigating the loss of reasoning abilities during fine-tuning.

Our contributions include:

1. Empirical evidence demonstrating the effectiveness of model merging to alleviate issues like catastrophic forgetting in LLM fine-tuning.
2. Insights into the disconnect between expert trajectory prediction and downstream task success, highlighting the need for more robust fine-tuning strategies.
3. Results showing that merging models, rather than relying solely on behavior cloning, can significantly improve task performance.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

4. An open-source 140M token dataset of successful expert traces and a codebase for further research and experimentation.

2 Methodology

2.1 Agent Design

We mostly follow the agent design of Drouin et al. [2024]. See §B for an explanation of the input, prompt and output space of the agents. See §E for details and a sample prompt.

2.2 Finetuning Pipeline

The finetuning pipeline enhances agents by addressing challenges in reasoning, planning, and executing complex tasks in diverse environments. It consists of six key stages. See Fig. 1 for a diagram and summary.

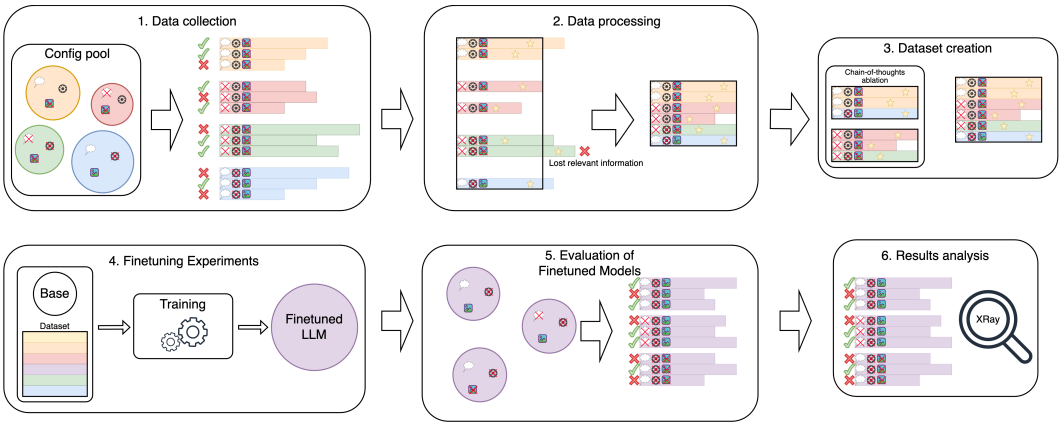


Figure 1: Our generic pipeline: 1) Trajectories are generated using different configurations (Chain-of-thoughts: 💭, use error logs: ⚙, use screenshot: 🖼) and different LLMs (highlighted by different colors). 2) Only the successful trajectories are kept. As each prompt is truncated to fit in our trained model's window, some key information (☆) might get lost in the process. Those samples are discarded. 3) The pipeline now has a pool of data, which can be used to build training sets with different properties. Here, we build an ablation dataset that separates data with and without chain-of-thoughts, and a dataset that keeps both. 4) After selecting a dataset, we train our model starting from a base model to make a stronger finetuned LLM. 5) The latter is used along with different agent configurations to assess the finetuning quality. 6) Finally, we can leverage AgentLab's tools to manually analyze the traces produced by the model.

2.3 AgentMerge

Model merging has been used effectively to ensemble models [Wortsman et al., 2022], combine learned skills Yadav et al. [2023], and as a regularization technique to mitigate forgetting [Ramé et al., 2024]. In our work, we apply model merging to address expert overfitting and enhance generalization. We propose a novel merging approach where we first *agent fine-tune* the pretrained, non-instruction-tuned model, obtaining 'agentic vectors', task vector equivalent of agentic behavior. We then interpolate them with the instruction-tuned task vectors, before eventually adding the interpolated vector back to the base model. This method approximates a model that has undergone both instruction fine-tuning and behavior cloning simultaneously. We call this technique AgentMerge.

3 Experiments

We evaluate the fine-tuning performance of Llama3.1-8B on the WorkArena benchmarks. Experiments focus on learning rate ablation, dataset ablation (with and without CoT), and generalization to unseen tasks. We measure success rates on held-out task configurations and unseen tasks, using success rate and likelihood of expert trajectories as evaluation metrics. The data collection models

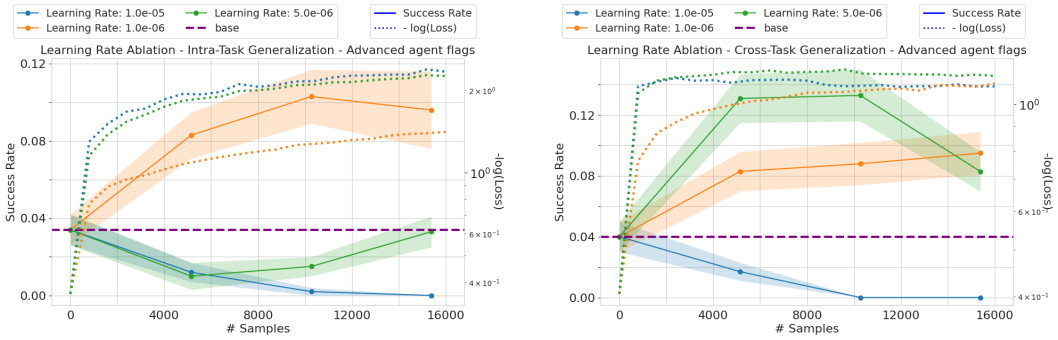


Figure 2: Success rate (left y-axis) and modified likelihood of expert trajectories (right y-axis) in the inter-task (left) and cross-task (right) generalization setup, throughout the fine-tuning phase. Interestingly, the model's improved ability to predict expert trajectories does not directly translate to better performance on downstream tasks.

are LLAMA3.1-70b and Mistral-Large-2. Additionally, we will fully open-source the WorkArena training dataset, which consists of 32K successful episodes and 140M tokens.

We use two agent configurations: `Advanced` and `Basic`. The `Advanced` configuration employs a more powerful set of prompt options, including CoT reasoning and the ability to generate multiple actions simultaneously. This set of flags was optimized for the Llama-3-8B Instruct model through hyperparameter search. For each reported experiment, we performed two fine-tuning runs and averaged the results. The shaded area represents one standard error in each direction.

3.1 Empirical analysis

In Fig. 2, we present a learning rate ablation study on both intra-task and cross-task generalization. For the `Advanced` agents, we observe that in the intra-task generalization scenario, the learning rate that performs the worst in predicting expert trajectories actually yields the best performance on WorkArena. We speculate that certain fine-tuning optimizations might overfit to expert behavior, leading to catastrophic forgetting of reasoning abilities learned during pretraining or instruction-following abilities from instruction-tuning.

In the cross-task generalization regime, while all learning rates converge to similar levels of expert trajectory prediction, the model with intermediate predictive accuracy on expert traces shows a significant drop in performance on WorkArena tasks. Furthermore, as shown in § E.2, extending training time or reducing the learning rate does not lead to performance improvements. For results on the `Basic` agents, please refer to § E.3.

This result is surprising, as one would typically expect a model that excels at predicting expert behavior to also perform well on downstream tasks. However, our findings indicate the opposite. This raises key questions: *"How does a model lose the ability to perform WorkArena tasks while improving at expert imitation?"* and *"What modifications in the data or training process could improve web agent performance?"*. Ideally, we would have conducted agentic fine-tuning simultaneously with the instruction-tuning step post-pretraining. Since that is no longer feasible, a potential solution is to continue pretraining the base model on agent traces and then merge it with the instruction-tuned version. This could approximate the desired scenario. In the next section, we provide empirical support for this approach as a way to mitigate forgetting and produce a more capable agent.

3.2 Merging experiments with AgentMerge

In Tab. 1, we present AgentMerge, comparing it to standard fine-tuning of the instruction-tuned model and to Model Soup, a popular approach that ensembles models. Specifically, Model Soup merges multiple fine-tuned models to create a stronger overall model. We observe that AgentMerge significantly improves performance over the instruct-then-finetuned model, supporting our hypothesis that 1) incorporating agent trajectories into post-training could have mitigated catastrophic forgetting, and 2) merging the agentic vector with the instruction-tuned one approximates this ideal scenario. Note

that we did not compute Model Soup for the learning rate that showed no performance improvements. Additionally, we observe that while Model Soup helps mitigate forgetting, it does so to a lesser extent than our proposed approach. In Fig. 3, we plot the learning curves comparing standard fine-tuning with AgentMerge. In §F we provide some analysis

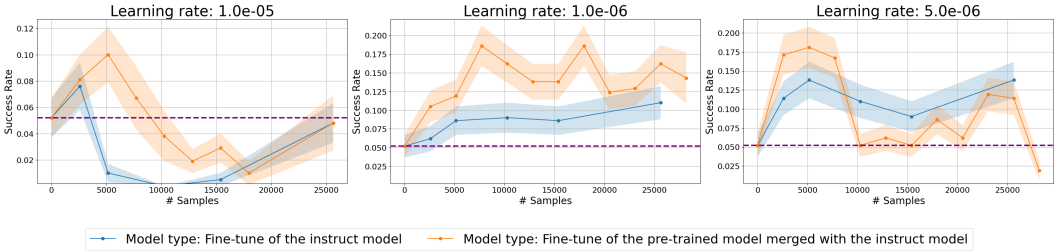


Figure 3: Success rate on WorkArena in the cross-task generalization setup throughout the fine-tuning phase for agentic fine-tuning of the instruction-tuned model vs AgentMerge. Merging consistently provides the fastest learning and achieves the highest peak performance. However, it eventually experiences some forgetting, likely due to the growing divergence between the agentic fine-tuning and instruction-tuning, which becomes increasingly difficult to merge effectively.

4 Related Works

Knowledge Distillation (KD) [Bucila et al., 2006, Hinton, 2015] has emerged as a successful technique for transferring the knowledge of a larger or more complex model to a more efficient model. In text generation, distillation approaches have attempted to either train the student model for token-level predictions using outputs of the teacher model [Sanh et al., 2019], or train the student model to make predictions at the sequence-level [Kim and Rush, 2016, Chiang et al., 2023, Peng et al., 2023]. However, for pre-trained LLMs, knowledge distillation of domain-specific data can lead to forgetting, which is a common problem in continual learning, and might require specialized methods for mitigation [Wu et al., 2022].

LR	Method	Test reward$_{\pm \text{std err}}$
5.0e-6	Instruct-fine-tuned	0.136 $_{\pm 0.03}$
	Model soup	0.156$_{\pm 0.04}$
	AgentMerge	**0.178**$_{\pm 0.03}$
1.0e-6	Instruct-fine-tuned	0.123$_{\pm 0.03}$
	Model soup	0.156$_{\pm 0.04}$
	AgentMerge	**0.171**$_{\pm 0.03}$
1.0e-5	Instruct-fine-tuned	0.078$_{\pm 0.02}$
	AgentMerge	**0.103**$_{\pm 0.02}$

Table 1: Comparing fine-tuning the instruction-tuned model and the merged model with a model soups of the fine-tuned versions of the instruction-tuned models.

Model merging involves interpolating some or all the parameters of different models together. One of the initial works for merging in the era of large pre-trained models uses merging as a form of regularization across hyperparameters, called model soups [Wortsman et al., 2022]. Model ratoutille [Ramé et al., 2023] uses fine-tuning on similar tasks before fine-tuning on the target task for a more generalized model. These methods generally average all the parameters of the model. Task arithmetic [Ilharco et al., 2023] recommends interpolating only amongst the 'task vectors' - parameters that have changed when fine-tuning a base model, which has proven to be more effective for pre-trained models. Methods like TIES [Yadav et al., 2023] and DELLA [Deep et al., 2024] add further advancements over standard task arithmetic specific to their settings.

5 Conclusion

We investigated fine-tuning open-source LLMs to function as agents on benchmarks like MiniWoB and WorkArena. While fine-tuning yielded performance improvements, challenges such as catastrophic forgetting arose. Notably, improved prediction of expert trajectories did not consistently enhance downstream task performance, suggesting that behavior cloning overfits to expert behavior, leading to the forgetting of useful statistical dependencies learned during pretraining. We propose addressing this issue by incorporating agent trajectories into the post-training process and merging the fine-tuned models with their instruction-tuned counterparts to better balance imitation and generalization.

References

C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Knowledge Discovery and Data Mining*, 2006. URL https://api.semanticscholar.org/CorpusID:11253972.

W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

P. T. Deep, R. Bhardwaj, and S. Poria. Della-merging: Reducing interference in model merging through magnitude-based sampling, 2024.

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.

H. He, W. Yao, K. Ma, W. Yu, Y. Dai, H. Zhang, Z. Lan, and D. Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. *arXiv*, abs/2401.13919, 2024. URL https://arxiv.org/abs/2401.13919.

G. Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

G. Ilharco, M. T. Ribeiro, M. Wortsman, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mixtral of experts, 2024.

G. Kim, P. Baldi, and S. McAleer. Language models can solve computer tasks. *arXiv*, abs/2303.17491, 2023. URL https://arxiv.org/abs/2303.17491.

Y. Kim and A. M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, 2016.

Meta. Llama 3: Meta's latest large language model. https://github.com/meta-llama/llama3, 2024. Accessed: 2024-06-03.

B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with gpt-4. *ArXiv*, abs/2304.03277, 2023. URL https://api.semanticscholar.org/CorpusID:257985497.

A. Ramé, K. Ahuja, J. Zhang, M. Cord, L. Bottou, and D. Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization, 2023. URL https://arxiv.org/abs/2212.10445.

A. Ramé, J. Ferret, N. Vieillard, R. Dadashi, L. Hussenot, P.-L. Cedoz, P. G. Sessa, S. Girgin, A. Douillard, and O. Bachem. Warp: On the benefits of weight averaged rewarded policies, 2024. URL https://arxiv.org/abs/2406.16768.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019. URL https://api.semanticscholar.org/CorpusID:203626972.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, and L. Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

T. Wu, M. Caccia, Z. Li, Y.-F. Li, G. Qi, and G. Haffari. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=figzpGMrdD.

P. Yadav, D. Tam, L. Choshen, C. Raffel, and M. Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

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

A Web Agent Pipeline

Our experiments rely on an ecosystem of tools for web agents, which we release as open-source contributions to the community to facilitate prototyping, evaluation, training, and reproducibility.

WorkArena[1] (Fig. 4a, § D) is a benchmark for evaluating web agents on the ServiceNow platform [Drouin et al., 2024]. It measures their ability to perform basic tasks using the main UI components of its user interface. For example, one of the tasks consists in filling out a form after receiving the explicit list of desired values for each field. It is the first benchmark to measure the performance of web agents at solving work-related tasks in the enterprise setting.

BrowserGym[2] (Fig. 4b, § D) is a gym environment that facilitates the design and evaluation of web agents in a unified framework. The salient features of BrowserGym include: i) chat-based agent-human interactions, ii) enriched multimodal observations: HTML, AXTree [Zhou et al., 2023], screenshot, set-of-marks [He et al., 2024], element coordinates, etc. and iii) a standard and flexible action space: `click`, `type`, etc..

AgentLab[3] offers a full pipeline for the large-scale evaluation of web agents. It offers features such as parallel evaluation, standardized data collection, and visual trace analysis and inspection tools.

B Agent Design

Input Our agents receive the task goal, the current page's accessibility tree[4] (AXTree) [Zhou et al., 2023], and an error message resulting from the execution of the previous action, if any. Our study focuses on fine-tuning and merging pure LLM-based agents without using screenshot observations.

Prompt We use a tool called dynamic prompting to build our agent's prompt in a modular manner, using different flags to activate or deactivate the desired features. For example, it allows us to activate or deactivate chain-of-thoughts (CoT) reasoning [Wei et al., 2022], a technique that encourages LLMs to generate intermediate reasoning steps to improve their performance on tasks requiring complex problem-solving, rather than directly producing a final answer. It can also activate error or history logging, amongst other things. We use different configurations of those flags in our experiments.

Output Our agent produces a textual reasoning (when CoT is active) plus an action in the form of a function call. We use the high-level action space from BrowserGym[5] , which allows sending messages to the chat, and interacting with the webpage using element identifiers (`bid` attribute).

C Finetuning Pipeline

The proposed finetuning pipeline for enhancing web agents systematically addresses the challenges of developing models capable of reasoning, planning, and executing complex tasks in enterprise environments. The process is structured into seven key stages:

Step 1: Data Collection We initiate by deploying a collection of web agents within the WorkArena environment. These agents vary across multiple axes, such as the foundational LLMs, observation modalities (AX tree vs. HTML), action spaces (high-level UI actions vs. Python API calls), and prompting techniques like CoT reasoning. Each agent operates in real-world scenarios, collecting interaction traces reflecting diverse approaches to solving tasks like form-filling and list manipulation.

The outcome of this step is a comprehensive corpus of agent-generated task traces that encapsulate different strategies for tackling UI interactions.

[1] `https://github.com/ServiceNow/WorkArena`

[2] `https://github.com/ServiceNow/BrowserGym`

[3] `https://github.com/ServiceNow/AgentLab`

[4] AXTree is a simplified representation of the page in text format for visually impaired users. It contains about 10x less token than the HTML and it is sufficient for most tasks in WorkArena

[5] `https://github.com/ServiceNow/BrowserGym`

Step 2: Data Processing The collected data undergoes rigorous filtering and transformation. Successful traces — where agents complete tasks per WorkArena's validation criteria — are retained for training. To adapt the data to finetuning, we simulate interactions using smaller context windows, trimming prompts when necessary. If key information (such as field identifiers) is missing, the trace is discarded. This ensures training data consistency and high relevance to the finetuning models.

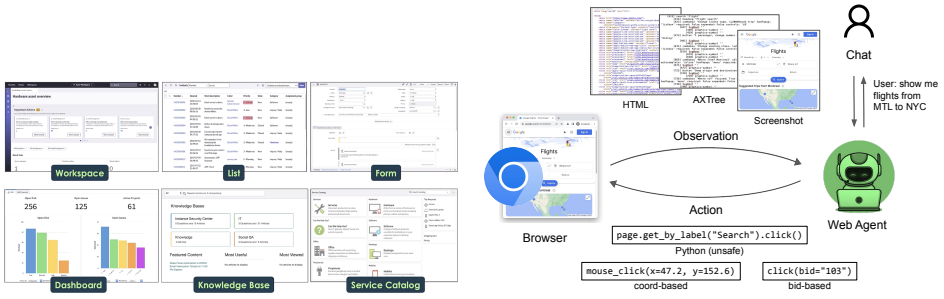
Step 3: Dataset Creation Next, we curate multiple datasets for finetuning. One comprehensive dataset includes all traces, while additional ablation datasets focus on specific variables like CoT reasoning. These ablation studies maintain uniform dataset sizes to avoid biasing results due to data quantity, allowing us to isolate the impact of certain features on learning performance.

Step 4: Finetuning Experiments Finetuning is performed on selected base models supported by our framework. We explore two primary experiment types. In *Dataset Ablations*, we finetune the same model across various datasets, maintaining consistent hyperparameters. This experiment evaluates which datasets or features are most beneficial for learning. In *Hyperparameter Ablations*, we hold the dataset constant while systematically varying one hyperparameter (e.g., learning rate) to assess its influence on model generalization. The outcome is a series of finetuned models, saved as checkpoints for further evaluation.

Step 5: Evaluation of Finetuned Models The finetuned models are converted into web agents, and configured with various evaluation flags. These agents are tested on unseen task instances and configurations to assess their generalization abilities across two levels. In *Inter-task Generalization*, we evaluate agent performance on unseen configurations (seeds) of tasks previously encountered during training. In *Cross-task Generalization* we test the agents on entirely novel tasks that were not part of the training set. The resulting evaluation data provides insights into the models' robustness and flexibility in handling diverse enterprise workflows.

Step 6: Analysis of Results The evaluation results are analyzed to extract key insights into model performance. We generate visualizations and plots (as discussed in § 3.1) to highlight trends, such as the impact of different datasets or finetuning strategies on agent success rates. Additionally, tools like AgentLab's `AgentXray` facilitate deeper inspection of agent behaviors, allowing us to identify strengths and weaknesses in decision-making processes. These insights guide further model refinements and inform future research directions.

D Software



(a) WorkArena **(b)** BrowserGym

Figure 4: (a) WorkArena is a collection of tasks which measure the ability of web agents to interact with basic UI components in the ServiceNow platform. (b) BrowserGym is a framework to execute web agents that receive a natural-language goal from a human user via chat, perceive the environment (web browser) through a set of multimodal observations (e.g., HTML and screenshot), and control it via a standardized set of available actions.

E Agent Design

Below are the general design choices of our LLM-based web agent with chain-of-thought prompting [Wei et al., 2022].

Language models: Our study focuses on open-source LLMs. For data collection we use both **Llama3.1-70b** [Meta, 2024] (`meta-llama-3.1-70b-instruct`, 70B parameters, 128K context) and **Mistral Large 2** [Jiang et al., 2024] (`mistral-large-2407`, 123B parameters, 128K context). For fine-tuning we consider a smaller **Llama3.1-8b** [Meta, 2024] model (`meta-llama-3.1-8b-instruct`, 70B parameters, 128K context). These LLMs are deployed using Hugging Face's Text Generation Inference (TGI) library on 4 A100 GPUs.

Observation space: Our observation space is composed of the goal, the current page's HTML and/or AXTree,[6] the currently focused element, and the error from the previous action if any. We also augment each element with two extra boolean properties provided by BrowserGym, `clickable` and `visible`.

Action space: We use BrowserGym's high-level action space with `chat` and `bid` primitives [Drouin et al., 2024] which respectively allow the agent to send messages to the chat (‘`send_msg_to_user(text)`‘, necessary for information retrieval tasks), and to interact with the page's HTML elements using their unique identifiers (e.g., `click(bid)`, `type(bid, text)` etc.). The `bid` primitives rely on the unique `bid` attribute given by BrowserGym to each HTML element, which is made available textually in the HTML and AXTree. The full action space is described to the agent in the prompt, with individual examples of valid function calls for each primitive. For an example prompt, see Fig. 5.

History: To extend the horizon window of our agent, at each time step we re-inject into the agent's prompt the history of all previous actions and thoughts (from chain-of-thought) since the start of the episode. This gives our agent a chance to recall its previous thoughts, thereby providing a crude memorization mechanism to otherwise memory-less agents.

Zero-shot examples: In the prompt, we provide a single generic example of how the chain-of-thought and action outputs should be formatted. This contrasts with other methods [Kim et al., 2023] where task-specific few-shot examples are provided, yet aligns with our objective of developing zero-shot agents able to solve a large range of new tasks.

Parse and retry: Once the LLM provides an answer, we have a parsing loop that can re-prompt the agent up to 4 times to make it aware of a parsing mistake. This can save the agent from making basic mistakes and is mainly useful for less capable LLMs. Once parsed, the action is executed via BrowserGym, which moves to the next step.

Prompt truncation: When the prompt is too large for the context window of our agent, we progressively truncate the HTML and AXTree from the end until it fits the maximum allowed number of tokens.

E.1 More results

E.2 Basic Agents' results

E.3 Basic Agents' results

F Analysing the Finetuned Model's behaviour

When performing fine-tuning and particularly evaluating on unseen tasks, we observe that the model is in-fact able to imitate the behavior based on the traces of the training data. For example, we see that the model struggles significantly with tasks requiring navigation either due to not having encountered them at all during training or having observed very similar navigation tasks, failing to acquire the skill to solve them. Parallely, the model becomes powerful at previously unseen form filling tasks, having observed and learnt from similar tasks in the training data. These observations indicate that the model is able to imitate, or learn with fine-tuning on individual observation-action instances and apply them for sequential decision making. Interestingly, it is often unable to improve over types of tasks that were impossible for the base model to solve.

[6] On WebArena and WorkArena we only use AXTrees because HTML is prohibitively large. On MiniWoB we use both AXTree and HTML as it consistently gives the best performance.

Example Prompt - Order Sales Laptop task

```
# Instructions
Review the current state of the page and all other information to find the best
possible next action to accomplish your goal. Your answer will be interpreted
and executed by a program, make sure to follow the formatting instructions.

## Goal:
Go to the hardware store and order 6 "Sales Laptop" with configuration
{'Additional software requirements': 'Slack, Zoom, Google Workspace, HubSpot, Adobe Creative Cloud',
'Adobe Acrobat': True, 'Adobe Photoshop': False, 'Microsoft Powerpoint': False, 'Siebel Client': False}

# Observation of current step:

## AXTree:
Note: [bid] is the unique alpha-numeric identifier at the beginning of lines for each element in the
AXTree. Always use bid to refer to elements in your actions.
Note: You can only interact with visible elements. If the "visible" tag is not
present, the element is not visible on the page.

RootWebArea 'Catalog | ServiceNow'
...
    [a] Iframe 'Main Content', visible
      RootWebArea 'Catalog', focused
...
                      [a251] heading 'Hardware', clickable, visible
                        [a252] link 'Hardware', clickable, visible
...
                [a261] link '', clickable, visible
                  [a262] table '', visible
                    [a263] rowgroup '', visible
                      [a264] row '', visible
                        [a265] gridcell '', visible
                        [a268] gridcell 'Hardware. Order from a variety of hardware to meet your business
                        needs, including phones, tablets and laptops. Order from a variety of hardware to meet
                        your business needs, including phones, tablets and laptops.', clickable, visible
                          [a269] link 'Hardware. Order from a variety of hardware to meet your business
                          needs, including phones, tablets and laptops.', clickable, visible
                            [a270] heading 'Hardware', visible
...

## Focused element:
bid='a85'

# History of interaction with the task:
...

# Action space:

Note: This action set allows you to interact with your environment. Most of them
are python functions executing playwright code. The primary way of referring to
elements in the page is through bid which are specified in your observations.
13 different types of actions are available.
...
fill(bid: str, value: str)
    Description: Fill out a form field. It focuses the element and triggers an input event with the
    entered text. It works for <input>, <textarea> and [contenteditable] elements.
    Examples:
        fill('237', 'example value')
        fill('45', 'multi-line\nexample')
        fill('a12', 'example with "quotes"')
...
send_msg_to_user(text: str)
    Description: Sends a message to the user.
    Examples:
        send_msg_to_user('Based on the results of my search, the city was built in 1751.')

Only a single action can be provided at once. Example:
fill('a12', 'example with "quotes"')
...
# Concrete Example

Here is a concrete example of how to format your answer.
Make sure to follow the template with proper tags:

<think>
From previous action I tried to set the value of year to "2022",
using select_option, but it doesn't appear to be in the form. It may be a
dynamic dropdown, I will try using click with the bid "a324" and look at the
response from the page.
</think>

<action>
click('a324')
</action>
```

Figure 5: Example prompt of our LLM-based agent. Some parts are truncated (...) for clarity.

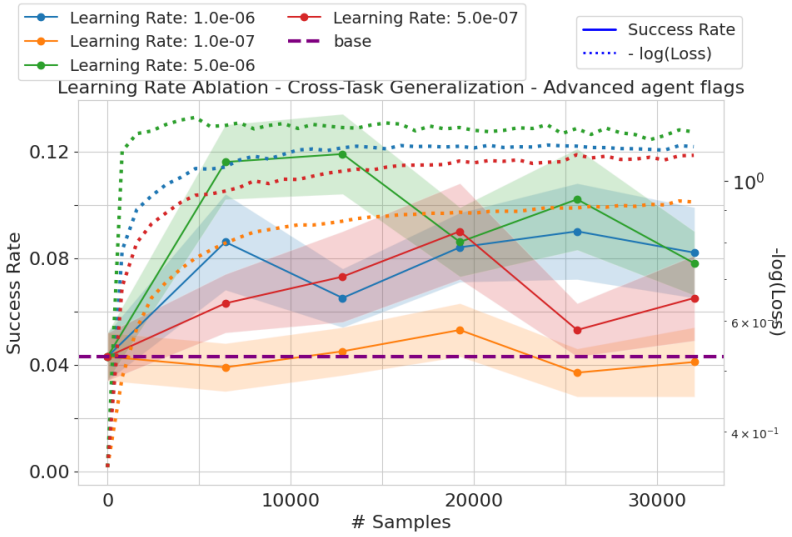


Figure 6: Learning rate ablation on WorkArena in the cross-task generalization setup. Training longer or using smaller learning rates does not improve downstream performance, indicating that these strategies may not enhance the agent's generalization to unseen tasks.

Error Analysis As mentioned previously, the validation loss alone is insufficient to predict downstream task success. To understand why, we explored the traces from agent $1e-05$ in Fig. 2 at the first 3 checkpoints of its training, respectively at 5,120, 10,240 and 15,360 samples. At 5,120 samples, the model performs best, while the performance deteriorates markedly in subsequent checkpoints.

For instance, in all instances of a task asking the agent to change the current user, the model appears to have memorized examples rather than reasoning through the task. At both 10,240 and 15,360 samples, all the initial actions consistently start with `click(a324)`, which is the action provided as an example in the prompt. This suggests the model is not distinguishing between the observed data and the examples, highlighting a deficiency in its reasoning capabilities. Additionally, for these 2 checkpoints, very few traces include a "think" step.

At 10,240 samples, the model manages to complete only one task—likely by coincidence. This task, the dashboard task, resembles a standard question-answering task and requires minimal agentic abilities, mostly involving reading from the AxTree. A notable observation is that the model no longer generates "think" steps at this stage, jumping directly into actions without reasoning.

By 15,360 samples, the model's reasoning capabilities remain absent. It continues to default to actions from the examples, such as frequently outputting the bid "a324" from the training set, further reinforcing that the model is failing to adapt its actions based on actual observations. Overall, the progression from 5,120 to 15,360 samples indicates a significant decline in the model's agentic and reasoning abilities, with increasing reliance on memorized examples rather than understanding the task context and observations.

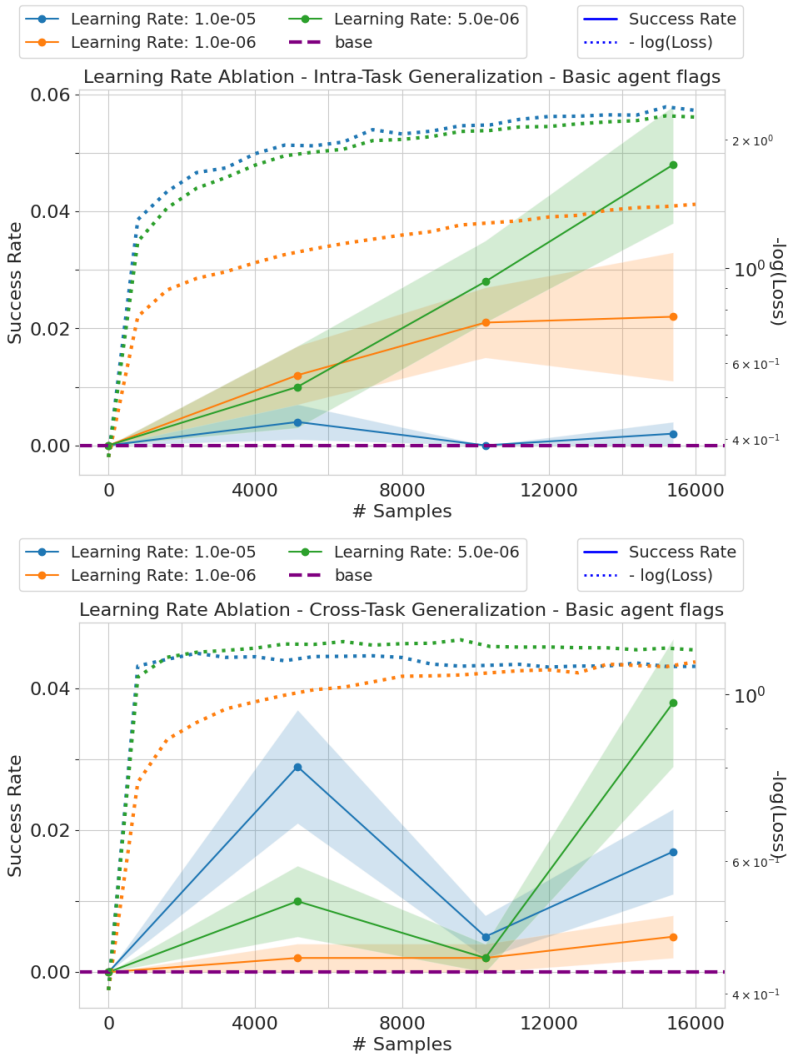


Figure 7: Learning rate ablation for the Basic agent in WorkArena