KIMI-DEV: AGENTLESS TRAINING AS SKILL PRIOR FOR SWE-AGENTS

Anonymous authors

Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) are increasingly applied to software engineering (SWE), with SWE-bench as a key benchmark. Solutions are split into SWE-Agent frameworks with multi-turn interactions and workflow-based Agentless methods with single-turn verifiable steps. We argue these paradigms are not mutually exclusive: reasoning-intensive Agentless training induces skill priors, including localization, code edit, and self-reflection that enable efficient and effective SWE-Agent adaptation. In this work, we first curate the Agentless training recipe and present Kimi-Dev, an open-source SWE LLM achieving 60.4% on SWE-bench Verified, the best among workflow approaches. With additional SFT adaptation on 5k publicly-available trajectories, Kimi-Dev powers SWE-Agents to 48.6% pass@1, on par with that of Claude 3.5 Sonnet (241022 version). These results show that structured skill priors from Agentless training can bridge workflow and agentic frameworks for transferable coding agents.

1 Introduction

Recent days have witnessed the rapid development of Large Language Models (LLMs) automating Software-Engineering (SWE) tasks (Jimenez et al., 2023; Yang et al., 2024a; Xia et al., 2024; Anthropic, 2024; Pan et al., 2024; Wang et al., 2025a; Wei et al., 2025; Yang et al., 2025a; Kimi et al., 2025; OpenAI, 2025c). Among the benchmarks that track the progress of LLM coding agents in SWE scenarios, SWE-bench (Jimenez et al., 2023) stands out as one of the most representative ones: Given an issue that reports a bug in a real-world GitHub repository, a model is required to produce a patch that fixes the bug, the correctness of which is further judged by whether the corresponding unit tests are passed after its application. The difficulty of the task (as of the date the benchmark was proposed), the existence of the outcome reward with the provided auto-eval harness, as well as the real-world economic value it reflects, have made the SWE-bench a focal point of the field.

Two lines of solutions have emerged for the SWE-bench task. Agent-based solutions like SWE-Agent (Yang et al., 2024a) and OpenHands (Wang et al., 2025a) take an interactionist approach: Instructed with the necessary task description, a predefined set of available tools, as well as the specific problem statement, the agent is required to interact with an executable environment for *multiple turns*, make change to the source codes, and determine when to stop autonomously. In contrast, workflow-based solutions like Agentless (Xia et al., 2024) pre-define the solving progress as a pipeline, which consists of steps like localization, bug repair, and test composition. Such task decomposition transforms the agentic task into generating correct responses for a chain of *single-turn* problems with verifiable rewards (Guo et al., 2025; Wei et al., 2025; He et al., 2025).

The two paradigms have been widely viewed as mutually exclusive. On the one hand, SWE-Agents are born with higher potential and better adaptability, thanks to the higher degree of freedom of the multi-turn interaction without the fixed routines. However, it has also proved more difficult to train with such frameworks due to their end-to-end nature (Luo et al., 2025; Cao et al., 2025). On the other hand, Agentless methods offer better modularity and the ease to train with Reinforcement Learning with Verifiable Rewards (RLVR) techniques, but more limited exploration space and flexibility, and difficulty in behavior monitoring as the erroneous patterns appear only in the single-turn long reasoning contents (Pan et al., 2024). However, we challenge the dichotomy from the perspective of training recipe: We argue that Agentless training should not be viewed as the ultimate deliverable, but rather as a way to induce skill priors – atomic capabilities such as the localization of buggy im-

plementations and the update of erroneous code snippets, as well as self-reflection and verification, all of which help scaffold the efficient adaptation of more capable and generalizable SWE-agents.

Guided by this perspective, we introduce Kimi-Dev, an open-source code LLM for SWE tasks. Specifically, we first develop an Agentless training recipe, which includes mid-training, cold-start, reinforcement learning, and test-time self-play. This results in 60.4% accuracy on SWE-bench Verified, the SoTA performance among the workflow-based solutions. Building on this, we show that Agentless training induces skill priors: a minimal SFT cold-start from Kimi-Dev with 5k publicly-available trajectories enables efficient SWE-agent adaptation and reaches 48.6% pass@1 score, similar to that of Claude 3.5 Sonnet (the 20241022 version, Anthropic (2024)). We demonstrate that these induced skills transfer from the non-agentic workflows to the agentic frameworks, and the self-reflection in long Chain-of-Thoughts baked through Agentless training further enable the agentic model to leverage more turns and succeed with a longer horizon. Finally, we also show that the skills from Agentless training generalize beyond SWE-bench Verified to broader benchmarks like SWE-bench-live (Zhang et al., 2025) and SWE-bench Multilingual (Yang et al., 2025c). Together, these results reframe the relationship between Agentless and agentic frameworks: not mutually exclusive, but as complementary stages in building transferable coding LLMs. This shift offers a principled view that training with structural skill priors could scaffold autonomous agentic interaction.

The remainder of this paper is organized as follows. Section 2 reviews the background of the framework dichotomy and outlines the challenges of training SWE-Agents. Section 3 presents our Agentless training recipe and the experimental results. Section 4 demonstrates how these Agentless-induced skill priors enable efficient SWE-Agent adaptation, and evaluates the skill transfer and generalization beyond SWE-bench Verified.

2 BACKGROUND

In this section, we first review the two dominant frameworks for SWE tasks and their dichotomy in Section 2.1. We then summarize the progress and challenges of training SWE-Agents in Section 2.2. The background introduction sets the stage for reinterpreting Agentless training as skill priors for SWE-Agents, a central theme developed throughout the later sections.

2.1 Framework Dichotomy

Two paradigms currently dominate the solutions for automating software engineering tasks. Agentless approaches decompose SWE tasks into modular workflows (Xia et al., 2024; Wei et al., 2025; Ma et al., 2025a;b; Xie et al., 2025). Typical workflows consist of bug localization, bug repair, and test generation. This design provides modularity and stability: each step could be optimized separately as a single-turn problem with verifiable rewards (Wei et al., 2025; He et al., 2025). However, such rigidity comes at the cost of flexibility. When encountering scenarios requiring multiple rounds of incremental updates, the Agentless approaches struggle to adapt.

By contrast, SWE-agents adopt an end-to-end, multi-turn reasoning paradigm (Yang et al., 2024a; Wang et al., 2025a). Rather than following a fixed workflow, they iteratively plan, act, and reflect, resembling how human developers debug complex issues. This design enables greater adaptability, but introduces significant difficulties: trajectories often extend over tens or even hundreds of steps, context windows of the LLMs must span over the entire interaction history, and the model must handle exploration, reasoning, and tool use simultaneously.

The dichotomy between fixed workflows (e.g., Agentless) and agentic frameworks (e.g., SWE-Agent) has shaped much of the community's perspective. The two paradigms are often regarded as mutually exclusive: one trades off flexibility and performance ceiling for modularity and stability, whereas the other makes the reverse compromise. Our work challenges this dichotomy, as we demonstrate that Agentless training induces skill priors that make further SWE-agent training both more stable and more efficient.

2.2 TRAINING SWE-AGENTS

Training SWE-agents relies on acquiring high-quality trajectories through interactions with executable environments. Constructing such large-scale environments and collecting reliable trajecto-

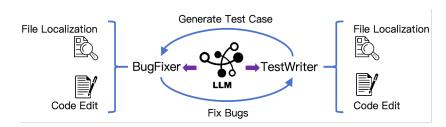


Figure 1: Agentless framework for Kimi-Dev: the duo of BugFixer and TestWriter.

ries, however, requires substantial human labor as well as costly calls to frontier models, making data collection slow and resource-demanding (Pan et al., 2024; Badertdinov et al., 2024b). Recent studies also attempt to scale environment construction by synthesizing bugs for the reverse construction of executable runtime (Jain et al., 2025; Yang et al., 2025c). However, credit assignment across long horizons still remains challenging, as outcome rewards are sparse and often only available when a final patch passes its tests. Reinforcement learning techniques have been proposed, but frequently suffer from instability or collapse when trajectories exceed dozens of steps (Luo et al., 2025; Cao et al., 2025). SWE-agent training is also highly sensitive to initialization: starting from a generic pre-trained model often leads to brittle behaviors, such as failing to use tools effectively or getting stuck in infinite loops of specific action patterns (Pan et al., 2024; Yang et al., 2025c).

These limitations motivate our central hypothesis: instead of training SWE-agents entirely from scratch, one can first induce skill priors through agentless training, enhancing the atomic capabilities like localization, repair, test composition, and self-reflection. These priors lay a foundation that makes subsequent agentic training both more efficient and more generalizable.

3 AGENTLESS TRAINING RECIPE

Instead of training SWE-agents from scratch, we leverage Agentless training to induce skill priors. Skill priors enhanced by Agentless training include but are not limited to bug localization, patch generation, self-reflection and verification, which lay the foundation for end-to-end agentic interaction. In this section, we elaborate our Agentless training recipe: the duo framework design of BugFixer and TestWriter, mid-training and cold-start, reinforcement learning, and test-time self-play. Sections 3.1–3.4 detail these ingredients, and Section 3.5 presents the experimental results for each of them. This training recipe results in Kimi-Dev, an open-source 72B model that achieves 60.4% on SWE-bench Verified, the SoTA performance among the workflow-based solutions.

3.1 Framework: The Duo of Bugfixer and Testwriter

In GitHub issue resolution, we conceptualize the process as the collaboration between two important roles: the BugFixer, who produces patches that correctly address software bugs, and the TestWriter, who creates reproducible unit tests that capture the reported bug. A resolution is considered successful when the BugFixer's patch passes the tests provided for the issue, while a high-quality test from the TestWriter should fail on the pre-fix version of the code and pass once the fix is applied.

Each role relies on two core skills: (i) file localization, the ability to identify the specific files relevant to the bug or test, and (ii) code edit, the ability to implement the necessary modifications. For Bug-Fixer, effective code edits repair the defective program logic, whereas for TestWriter, they update precise unit test functions that reproduce the issue into the test files. As illustrated in Figure 1, these two skills constitute the fundamental abilities underlying GitHub issue resolution. Thus, we enhance these skills through the following training recipes, including mid-training, cold-start, and RL.

3.2 MID-TRAINING & COLD START

To enhance the model's prior as both a BugFixer and a TestWriter, we perform mid-training with \sim 150B tokens in high-quality and real-world data. With the Qwen 2.5-72B-Base (Qwen et al., 2024) model as a starting point, we collect millions of GitHub issues and PR commits to form its mid-training dataset, which consists of (i) \sim 50B tokens in the form of Agentless derived from the

natural diff patch, (ii) \sim 20B tokens of curated PR commit packs, and (iii) \sim 20B tokens of synthetic data with reasoning and agentic interaction patterns (upsampled by a factor of 4 during training). The data recipe is carefully constructed to enable the model to learn how human developers reason with GitHub issues, implement code fixes, and develop unit tests. We also performed strict data decontamination to exclude any repository from the SWE-bench Verified test set. Mid-training sufficiently enhances the knowledge in the model about practical bug fixes and unit tests, making it a better starting point for later stages. The details of the recipe are covered in Appendix A.

To activate the model's long Chain-of-Thought (CoT) capability, we also construct a cold-start dataset with reasoning trajectories based on the SWE-Gym (Pan et al., 2024) and SWE-bench-extra (Badertdinov et al., 2024a) datasets, generated by the DeepSeek R1 model (Luo et al. (2025), the 20250120 version). In this setup, R1 acts the roles of Bugfixer and Testwriter, producing outputs such as file localization and code edits. Through supervised finetuning as a cold start with this dataset, we enable the model to acquire essential reasoning skills, including problem analysis, method sketching, self-refinement, and exploration of alternative solutions.

3.3 REINFORCEMENT LEARNING

After mid-training and cold-start, the model demonstrates strong performance in localization. Therefore, reinforcement learning (RL) focuses solely on the code edit stage. We construct a training set specifically for this stage, where each prompt is equipped with an executable environment. We further employ multiple localization rollouts from the initial model to generate varied file location predictions, which diversifies the prompts used in code-edit RL.

For the RL algorithm, we adopt the policy optimization method proposed by Kimi k1.5 (Team et al., 2025), which has shown promising results on reasoning tasks in both math and coding. Unlike GRPO (Shao et al., 2024) that eliminates the critic by using average rewards for advantage estimation, Kimi k1.5 (Team et al., 2025) adopts a simpler policy gradient approach motivated by path consistency (Nachum et al., 2017). The specific RL objective is provided in Appendix C.1.

When adapting the algorithm in our SWE-bench setting, we highlight the following 3 key desiderata:

- 1. **Outcome-based reward only**: We rely solely on the final execution outcome from the environment as the raw reward (0 or 1), without incorporating any format- or process-based signals. For BugFixer, a positive reward is given if the generated patch passes all ground-truth unittests. For TestWriter, a positive reward is assigned when the predicted test raises a failure in the absence of the ground-truth bugfix patch and, once the patch is applied, the failure is resolved.
- 2. **Adaptive prompt selection**: Prompts with pass@16 = 0 are initially discarded as they do not contribute to the batch loss, effectively enlarging the usable batch size. A curriculum learning scheme is then applied: once the success rate on the current set exceeds a threshold, previously excluded prompts (with initial pass@16 = 0 but improved under RL) are reintroduced to gradually raise task difficulty.
- 3. **Positive example reinforcement**: As performance improvements begin to plateau in later stages of training, we incorporate the positive samples from the recent RL iterations into the training batch of the current iteration. This approach reinforces the model's reliance on successful patterns, thereby accelerating convergence in the final phase.

Robust sandbox infrastructure. We construct the docker environment with Kubernetes (Burns et al., 2016), which provides a secure and scalable sandbox infrastructure and efficient training and rollouts. The infra supports over 10,000 concurrent instances with robust performance, making it ideal for competitive programming and software engineering tasks (see Appendix D).

3.4 TEST-TIME SELF-PLAY

After RL, the model masters the roles of both a BugFixer and a TestWriter. During test time, it adopts a self-play mechanism to coordinate its bug-fixing and test-writing abilities. Following Agentless (Xia et al., 2024), we leverage the model to generate 40 candidate patches and 40 tests for each instance. Each patch generation involves independent runs of the localization and code edit from BugFixer, where the first run uses greedy decoding (temperature 0), and the remaining 39 use

Table 1: Performance comparison for models on SWE-bench Verified under Agentless-like frameworks. All the performances are obtained under the standard 40 patch, 40 test setting (Xia et al., 2024), except that Llama3-SWE-RL uses 500 patches and 30 tests.

Model	#Params	Resolve Rate (%)
Llama3-SWE-RL (Wei et al., 2025)	70B	41.0
Seed1.5-Thinking (Seed et al., 2025)	200B	47.0
OpenAI-o1 (OpenAI, 2024)	-	48.9
DeepSeek-R1-0120 (Guo et al., 2025)	671B	49.2
OpenAI-o3-mini-high (OpenAI, 2025a)	-	49.3
Claude 3.5 Sonnet (241022) (Anthropic, 2024)	-	50.8
MiniMax-M1 (Chen et al., 2025a)	456B	56.0
DeepSeek-R1-0528 (Guo et al., 2025)	671B	57.6
SWE-SWISS (He et al., 2025)	32B	58.2
Kimi-Dev (Ours)	72B	60.4

temperature 1 to ensure diversity. Similarly, 40 tests are generated independently from TestWriter. For the test patch candidates, to guarantee their validity, we first filter out those failing to raise a failure in the original repo without applying any BugFixer patch.

Denote the rest TestWriter patches as set \mathcal{T} , and the BugFixer patches as set \mathcal{B} . For each $b_i \in \mathcal{B}$ and $t_j \in \mathcal{T}$, we execute the test suite over the test file modified by t_j for twice: first without b_i , and then with b_i applied. From the execution log for the first run, we get the count of the failed and the passed tests from t_j , denoted as F(j) and P(j). Comparing the execution logs for the two test suite runs, we get the count of the fail-to-pass and the pass-to-pass tests, denoted as FP(i,j) and PP(i,j), respectively. We then calculate the score for each b_i with

$$S_i = \frac{\sum_j \text{FP}(i,j)}{\sum_j \text{F}(j)} + \frac{\sum_j \text{PP}(i,j)}{\sum_j \text{P}(j)},$$
(1)

where the first part reflects the performance of b_i under reproduction tests, and the second part could be viewed as the characterization of b_i under regression tests (Xia et al., 2024). We select the BugFixer patch b_i with the highest S_i score as the ultimate answer.

3.5 EXPERIMENTS

3.5.1 MAIN RESULTS

We evaluate our approach on SWE-bench Verified (Jimenez et al., 2023), which includes 500 GitHub issues guaranteed to be solved by human programmers. Table 1 shows the overall results. Unlike SWE-RL (Wei et al., 2025), which relies on text-based similarity to the ground-truth patch, we use execution results as the reward signal, reducing shortcuts and improving solution quality. Unlike most Agentless approaches (Xia et al., 2024; Guo et al., 2025; He et al., 2025), which place a single-file reproduction test at the root directory of the repository, we leverage the two-stage TestWriter to better capture repository context and align with human standard workflows (OpenAI, 2025). Kimi-Dev model achieves the state-of-the-art performance among open-source models, resolving 60.4% of the issues in SWE-bench Verified.

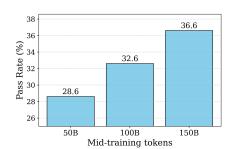


Figure 2: The performance on SWEbench Verified after mid-training with different training token budgets.

3.5.2 MID-TRAINING

In this section, we evaluate the relationship between the amount of data used during mid-training and model performance. Specifically, we finetuned Qwen 2.5-72B-Base with the subset of mid-training data of 50B, 100B, and approximately 150B tokens, and then lightly activated them using the same 2,000 Bugfixer input-output pairs as cold start. We report BugFixer pass@1 without self-

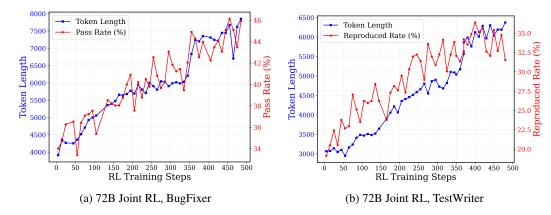


Figure 3: Joint code-edit RL experiments on the model after mid-training and cold-start. The pass rate for BugFixer and the reproduced rate for TestWriter are reported as pass@1 with temperature=1.0. The performance improves consistently as the output becomes increasingly longer.

play for simplicity of evaluation. Figure 2 shows that increasing the number of tokens in mid-training consistently improves model performance, highlighting the effectiveness of this stage.

3.5.3 Reinforcement Learning

Experimental setup We set the training step per RL iteration as 5 and sample 10 rollouts for each of the 1,024 problems from the union of SWE-gym (Pan et al., 2024) and SWE-bench-extra (Badertdinov et al., 2024b). We dynamically adjust the prompt set every 20 iterations to gradually increase task difficulty. We fix the maximum training context length as 64k tokens, since the prompt input contains the contents of the entire files localized by the initial model in advance.

Results Figure 3 shows the performance and response length curves on the test set during RL training. The pass rate and the reproduced rate are calculated from pass@1 and temperature=1. Specifically, we observe that both model performance and response length steadily increase, reflecting the expected benefits of RL scaling. Similar RL scaling curves are also observed in our ablation experiments run on Qwen2.5-14B-Instruct models, proving the effectiveness of the RL training recipe across models of different sizes. The experimental details, as well as the ablation studies on positive example reinforcement in Section 3.3, are listed in Appendix C.3). The lengthy outputs consist of in-depth problem analysis and self-reflection patterns, similar to those in the math and code reasoning tasks (Team et al., 2025; Guo et al., 2025). We have also observed that for TestWriter, occasional false-positive examples take place during RL training due to the lack of reproduction coverage. We leave the case studies in Appendix F and further improvement for future work.

3.5.4 TEST-TIME SELF-PLAY

Following Section 3.4, we evaluate how the final performance on the SWE-bench Verified scales with the number of patches and tests generated. The temperature is fixed at 0 for the initial rollout and set to 1.0 for the subsequent 39 rollouts. As shown on the left of Figure 4, performance improves from 48.0% to 60.4% as the number of patch test pairs increases from 1×1 to 40×40 , and consistently outperforms the results obtained from the majority vote of BugFixer patches. Specifically, the self-play result obtained from 3 patches and 3 tests for each instance has already surpassed the performance with

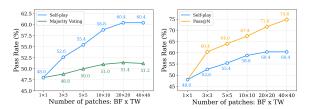


Figure 4: Test-time self-play on SWE-bench Verified. Performance improves with more generated patches and tests. Left: Execution-based self-play consistently surpasses BugFixer majority voting. Right: Self-play performances remain below pass@N where the ground-truth test patch is used, suggesting the room exists for TestWriter to improve.

majority voting from 40 BugFixer patches. This demonstrates the effectiveness of additional in-

Table 2: Single-attempt performance of different models on SWE-bench Verified under end-to-end agentic frameworks, categorized by proprietary or open-weight models, and size over or under 100B. "Internal" denotes results achieved with their in-house agentic frameworks.

Model	System	#Params	Pass Rate (%)		
Proprietary					
Gemini 2.5 Pro (Comanici et al., 2025)	Internal	-	60.3		
OpenAI-o3 (OpenAI, 2025)	Internal	-	69.1		
GPT-5 (OpenAI, 2025c)	Internal	-	74.9		
Claude 3.5 Sonnet (241022) (Anthropic, 2024)	SWE-Agent	-	49.0		
Claude 3.7 Sonnet (Anthropic, 2025a)	SWE-Agent	-	62.3		
Claude 4.0 Sonnet (Anthropic, 2025b)	SWE-Agent	-	72.7		
Open Weight, ≥ 100B					
gpt-oss-120b (High) (OpenAI, 2025b)	Internal	120B	62.4		
DeepSeek-v3.1 (Guo et al., 2025)	Internal	671B	66.0		
Kimi-K2-0905 (Kimi et al., 2025)	SWE-Agent	1T	69.2		
Qwen3-Coder (Yang et al., 2025a)	OpenHands	480B	69.6		
Open Weight, < 100B					
Openhands-LM (Wang et al., 2025b)	OpenHands	32B	37.2		
Skywork-SWE (Zeng et al., 2025)	OpenHands	32B	38.0		
SWE-agent-LM (Yang et al., 2025b)	SWE-Agent	32B	40.2		
DeepSWE (Luo et al., 2025)	OpenHands	32B	42.2		
Devstral-Small-2507 (AI & AI, 2025)	OpenHands	24B	53.6		
gpt-oss-20b (High) (OpenAI, 2025b)	Internal	20B	60.7		
Kimi-Dev (SFTed)	SWE-Agent	72B	48.6		

formation from test-time execution. The room for improvement of TestWriter, though, still exists for more powerful self-play: Shown on Figure 4, self-play performances remain below pass@N, where ground-truth test cases serve as the criterion for issue resolution. This finding aligns with Anthropic (2024), which introduced a final edge-case checking phase to generate a more diverse set of test cases, thereby strengthening the role of the "TestWriter" in their SWE-Agent framework. We also report preliminary observations of a potential parallel scaling phenomenon, which requires no additional training and may enable scalable performance improvements (see Appendix G).

4 INITIALIZING SWE-AGENTS FROM AGENTLESS TRAINING

End-to-end multi-turn frameworks, such as SWE-Agent (Yang et al., 2024a; Anthropic, 2024) and OpenHands (Wang et al., 2025a), enable agents to leverage tools and interact with environments. Specifically, the system prompt employed in the SWE-Agent framework (Anthropic, 2024) outlines a five-stage workflow: (i) repo exploration, (ii) error reproduction via a test script, (iii) code edit for bug repair, (iv) test re-execution for validation, and (v) edge-case generation and checks. Unlike Agentless, the SWE-Agent framework doesn't enforce a strict stage-wise workflow; the agent can reflect, transition, and redo freely until it deems the task complete and submits.

The performance potential is therefore higher without a fixed routine; However, the training for SWE-Agent is more challenging because of the sparsity of the outcome reward for long-horizon credit assignment. Meanwhile, our Kimi-Dev model has undergone Agentless training, with its skills of localization and code edit for BugFixer and TestWriter strengthened elaborately. In this section, we investigate whether it can serve as an effective prior for multi-turn SWE-Agent scenarios.

4.1 Performance after SWE-Agent Fine-tuning

We use the publicly available SWE-Agent trajectories to finetune Kimi-Dev. The finetuning dataset we used is released by SWE-smith (Yang et al., 2025b), consisting of 5,016 SWE-Agent trajectories collected with Claude 3.7 Sonnet (Anthropic, 2025a) in the synthetic environments. We perform supervised fine-tuning over Kimi-Dev, setting the maximum context length as 64K tokens during training, and allowing up to 128K tokens and 100 turns during inference.



Figure 5: Comparing the quality of the raw Base, the Agentless mid-trained (MT), the Agentless mid-trained with reasoning-intensive cold-start (SFT), and the Kimi-Dev model after RL as the prior for SWE-Agent adaptation. The tokens of the SWE-Agent SFT trajectories are swept over different scales, and the SWE-Agent performances are reported up to pass@3 on SWE-bench Verified.

As shown in Table 2, without collecting more trajectory data over realistic environments, or conducting additional multi-turn agentic RL, our finetuned model achieves a pass@1 score of 48.6% on SWE-bench Verified under the agentic framework setup, without additional test-time scaling. Using the same SFT data, our finetuned Kimi-Dev model outperforms the SWE-agent-LM (Yang et al., 2025c), with the performance comparable to that of Claude 3.5 Sonnet (49% by the 241022 version). The pass@10 of our SWE-Agent adapted model is 74.0% and surpasses the pass@30 of our model under Agentless (73.8%), proving the higher potential for the SWE-Agent framework.

4.2 SKILL TRANSFER AND GENERALIZATION

The results shown in Section 4.1 demonstrate that Kimi-Dev, a model with extensive Agentless training, could be adapted to end-to-end SWE-Agents with lightweight supervised finetuning. As the Agentless training recipe consists of mid-training, cold-start (SFT) and RL, we explore the contribution of each part in the recipe to the SWE-Agent capability after adaptation.

To figure this out, we perform SWE-Agent SFT on the original Qwen2.5-72B (Base), the midtrained model (MT), the model then activated with Agentless-formatted long CoT data (SFT), and the (Kimi-Dev) model after finishing RL training (RL). As we are treating the four models as the *prior* for SWE-Agents¹, and a good prior always demonstrates the ability of fast adaptation with a few shots (Finn et al., 2017; Brown et al., 2020), we also sweep the amount of SWE-Agent SFT data to measure the *efficiency* of each prior in SWE-Agent adaptation.

Specifically, we randomly shuffle the 5,016 SWE-Agent trajectories and construct nested subsets of sizes 100, 200, 500, 1,000, and 2,000, where each smaller subset is contained within the larger ones. In addition, we prepend two extreme baselines: (i) zero-shot, where the prior model is directly evaluated under the SWE-Agent framework without finetuning, and (ii) one-step gradient descent, where the model is updated with a single gradient step using the 100-trajectory subset. This yields a range of SFT token budgets spanning $\{0, 2^{21}, 2^{23}, 2^{24}, 1.1 \times 2^{25}, 1.1 \times 2^{26}, 1.1 \times 2^{27}, 1.5 \times 2^{28}\}$. After these lightweight SFT experiments, we evaluate performance in terms of pass@ $\{1,2,3\}$ under the SWE-Agent framework, with evaluations for pass@1 conducted at temperature 0, and those for pass@2 and pass@3 at temperature 1.0.

Figure 5 presents the SWE-Agent performances of each prior (Base, MT, SFT, RL) after being fine-tuned with different amounts of agentic trajectories. We have the following observations:

1. The RL prior is outperformed by all the other models in nearly all the SWE-Agent SFT settings. This demonstrates that the Agentless training recipe indeed strengthens the prior in terms of SWE-Agent adaptation. For example, To achieve the top pass@1 performance

¹We slightly abuse the term "prior" to refer to a model to be finetuned with SWE-Agent trajectories in the following analysis.

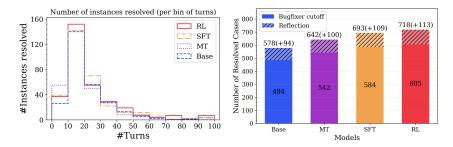


Figure 6: Left: Performance of the four priors under turn limits after SWE-Agent adaptation. Right: The characterization of the BugFixer and the reflection skills for each prior by counting the resolved cases of the 3 runs at Stage-3 cutoff moment, and comparing those with the final success cases.

of the Base prior, the RL prior needs only 2^{23} SWE-Agent SFT tokens, whereas the Base prior consumes 1.5×2^{28} tokens.

- 2. The MT prior is lagged behind the SFT and the RL ones in extremely data-scarce settings (zero-shot (0) and one-step gradient descent (2^{21})), but quickly becomes on par with them after 200 trajectories (2^{24}) are available for finetuning. This indicates that adaptation efficiency remains comparable after the prior is strengthened through Agentless mid-training.
- 3. The performance of the SFT prior is mostly similar to the RL one except for two cases: (i) The SFT prior outperforms the RL one under the zero-shot setting. This is reasonable, as the RL prior might overfit to the Agentless input-output format, while the SFT prior suffers less from this. (ii) The SFT prior exhibits a significant degradation with 200 SWE-Agent trajectories (2²⁴). A potential reason could be that the 200 trajectories collapse onto a single data mode, leading the SFT prior to overfit through memorization (Chu et al., 2025); the RL prior instead embeds stronger transferable skills and thus generalizes better.

From long CoT to extended multi-turn interactions. We hypothesize that reflective behaviors cultivated through long chain-of-thought reasoning may transfer to settings requiring extended multi-turn interactions. To examine this, we evaluate the four priors (Base, MT, SFT, and RL) by finetuning on the 5,016 trajectories and test on SWE-bench Verified, under varying turn limits with pass@3 as the metric (Figure 6, left). The distinct interaction-length profiles show supportive evidence: the RL prior, after finetuning, continues to make progress beyond 70 turns, while the SFT, mid-trained, and raw models show diminishing returns around 70, 60, and 50 turns, respectively.

We further evaluate the efficacy of the Agentless skill priors (**BugFixer** and **reflection**) in the SWE-Agent adapted model. For **BugFixer**, given that the SWE-Agent may autonomously reflect between the five stages, we examine the moment in each trajectory when the bug fix of the third stage is *initially* completed, and the test rerun of the fourth stage has not yet been entered. Heuristically, when the SWE-Agent just completes the third stage, it has not yet obtained the execution feedback from the fourth stage, and thus has not further reflected based on the execution information or refined the bug fix. We therefore calculate the success rate of direct submission at this cutoff moment, which reflects the capability of the BugFixer skill. Regarding **reflection**, we further compare the performance at the cutoff point with the performance after full completion for each problem. The increment in the number of successful problems is used to reflect the capability of the reflection skill. We use Kimi-K2 to annotate the SWE-Agent stage each turn lies in along each trajectory. Figure 6 demonstrates that both skills are strengthened in the RL prior, leading to superior performance.

5 CONCLUSION AND FUTURE WORK

In this work, we reframed Agentless and agentic paradigms for automated software engineering as complementary rather than competing. By introducing Kimi-Dev, we demonstrated that structured Agentless training can induce transferable skill priors, including bug localization, code repair, and self-reflection. As a result, Kimi-Dev not only achieves SoTA results on SWE-bench Verified among the workflow-based approaches, but enables efficient SWE-Agent adaptation as well. These findings establish a novel path toward building more generalizable coding agents through staged training.

ETHICS AND REPRODUCIBILITY STATEMENTS

This work obeys the Code of Ethics required by the ICLR conference. The study does not involve human subjects or animal experimentation. The personally identifiable information from raw data is excluded for privacy consideration (see the mid-training data recipe detailed in Appendix A).

For all of the experiments, we have covered the detailed setups in the appendices: mid-training for Agentless in Appendix A, details of the used dockers in Appendix B, Agentless RL in Appendix C, agent infrastructure in Appendix D, and case studies in Appendix F.

REFERENCES

- Mistral AI and All Hands AI. Devstral-small-2507. https://mistral.ai/news/devstral-2507, July 2025.
- Anthropic. Raising the bar on swe-bench verified with claude 3.5 sonnet. Online; AI model, Oct 2024. URL https://www.anthropic.com/engineering/swe-bench-sonnet.
- Anthropic. Claude 3.7 sonnet: Hybrid reasoning model. https://www.anthropic.com/news/claude-3-7-sonnet, February 2025a.
- Anthropic. Claude sonnet 4. https://www.anthropic.com/news/claude-4, May 2025b.
- Ibragim Badertdinov, Maria Trofimova, Yuri Anapolskiy, Sergey Abramov, Karina Zainullina, Alexander Golubev, Sergey Polezhaev, Daria Litvintseva, Simon Karasik, Filipp Fisin, et al. Scaling data collection for training software engineering agents. *Nebius blog*, 2024a.
- Ibragim Badertdinov, Maria Trofimova, Yury Anapolskiy, Sergey Abramov, Karina Zainullina, Alexander Golubev, Sergey Polezhaev, Daria Litvintseva, Simon Karasik, Filipp Fisin, Sergey Skvortsov, Maxim Nekrashevich, Anton Shevtsov, and Boris Yangel. Scaling data collection for training software engineering agents. *Nebius blog*, 2024b.
- Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents, 2025. URL https://arxiv.org/abs/2505.20411.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. URL https://arxiv.org/abs/2005.14165.
- Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, Omega, and Kubernetes. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, pp. 285–301, 2016.
- Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.
- Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv* preprint arXiv:2506.13585, 2025a.
- Mouxiang Chen, Binyuan Hui, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Jianling Sun, Junyang Lin, and Zhongxin Liu. Parallel scaling law for language models. *arXiv preprint arXiv:2505.10475*, 2025b.

- Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=dYur3yabMj.
 - Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261, 2025.
 - Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.
 - Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
 - Xinran Gu, Kaifeng Lyu, Jiazheng Li, and Jingzhao Zhang. Data mixing can induce phase transitions in knowledge acquisition. *arXiv preprint arXiv:2505.18091*, 2025.
 - Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv* preprint arXiv:2501.12948, 2025.
 - Zhenyu He, Qingping Yang, Wei Sheng, Xiaojian Zhong, Kechi Zhang, Chenxin An, Wenlei Shi, Tianle Cai, Di He, Jiaze Chen, Jingjing Xu, and Mingxuan Wang. Swe-swiss: A multi-task fine-tuning and rl recipe for high-performance issue resolution. Notion page, 2025.
 - Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv* preprint *arXiv*:2504.07164, 2025.
 - Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.
 - Team Kimi, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv* preprint arXiv:2507.20534, 2025.
 - Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Shang Zhu Tarun Venkat, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. DeepSWE: Training a State-of-the-Art Coding Agent from Scratch by Scaling RL. Notion page, 2025. Notion Blog.
 - Yingwei Ma, Rongyu Cao, Yongchang Cao, Yue Zhang, Jue Chen, Yibo Liu, Yuchen Liu, Binhua Li, Fei Huang, and Yongbin Li. Swe-gpt: A process-centric language model for automated software improvement. *Proc. ACM Softw. Eng.*, 2(ISSTA), June 2025a. doi: 10.1145/3728981. URL https://doi.org/10.1145/3728981.
 - Yingwei Ma, Qingping Yang, Rongyu Cao, Binhua Li, Fei Huang, and Yongbin Li. Alibaba ling-maagent: Improving automated issue resolution via comprehensive repository exploration. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 238–249, 2025b.
 - Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/facf9f743b083008a894eee7baa16469-Paper.pdf.
 - OpenAI. Openai o1 system card. Technical report, OpenAI, 2024. URL https://arxiv.org/abs/2412.16720. Includes o1 and o1-mini models, safety and evaluation work.

- OpenAI. Introducing codex. https://openai.com/index/introducing-codex/, May 2025
 - OpenAI. Openai o3 and openai o4-mini system card. Technical report, OpenAI, Apr 2025. URL https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.
 - OpenAI. Openai o3-mini system card. Technical report / system card, OpenAI, 2025a. URL https://cdn.openai.com/o3-mini-system-card-feb10.pdf. Description of o3-mini model, its safety evaluations and reasoning benchmarks.
 - OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025b. URL https://arxiv.org/abs/2508.10925.
 - OpenAI. Gpt-5 system card. OpenAI website, Aug 2025c. URL https://cdn.openai.com/gpt-5-system-card.pdf.
 - Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv* preprint *arXiv*:2412.21139, 2024.
 - Jiayi Pan, Xiuyu Li, Long Lian, Charlie Victor Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=YqwQ7sXPXU.
 - Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.
 - ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.
 - Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
 - Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
 - Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=OJd3ayDDoF.
 - Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=OJd3ayDDoF.
 - Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv* preprint arXiv:2502.18449, 2025.

- Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.
 - Chengxing Xie, Bowen Li, Chang Gao, He Du, Wai Lam, Difan Zou, and Kai Chen. SWE-fixer: Training open-source LLMs for effective and efficient GitHub issue resolution. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 1123–1139, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.62. URL https://aclanthology.org/2025.findings-acl.62/.
 - An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.
 - John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024a.
 - John Yang, Kilian Leret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025b.
 - John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents, 2025c. URL https://arxiv.org/abs/2504.21798.
 - Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. ReAct Meets ActRe: Autonomous Annotation of Agent Trajectories for Contrastive Self-Training. In *First Conference on Language Modeling*, 2024b. URL https://openreview.net/forum?id=0VLBwQGWpA.
 - Liang Zeng, Yongcong Li, Yuzhen Xiao, Changshi Li, Chris Yuhao Liu, Rui Yan, Tianwen Wei, Jujie He, Xuchen Song, Yang Liu, et al. Skywork-swe: Unveiling data scaling laws for software engineering in llms. *arXiv preprint arXiv:2506.19290*, 2025.
 - Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, et al. Swe-bench goes live! *arXiv preprint arXiv:2505.23419*, 2025.

APPENDIX

A DETAILS OF MID-TRAINING

We curate a mid-training data recipe with a focus on enhancing SWE capabilities. Central to this effort is the collection of pull request (PR) data from GitHub, which provides extensive coverage of real-world bug fixes, feature requests, and code enhancements. To ensure data quality, we apply two filters: (i) we only retain repositories that have accumulated at least five GitHub stars, thereby excluding sparsely maintained projects with limited community engagement; and (ii) we remove any repositories overlapping with the SWE-bench benchmark (Jimenez et al., 2023) to prevent potential data leakage. For each candidate repository, we query the GitHub API for all PRs with the state MERGED, while discarding those abandoned, superseded, or left under review. To preserve more context information, we also snapshot the entire codebase at the base commit before the first code change in the PR.

After data crawling, we incorporate two complementary forms for the natural code change data: (i) **natural diff patches** and (ii) **PR commit packs**. A natural diff patch consolidates all commits in a PR into the final code difference, typically expressed as SEARCH–REPLACE blocks. This format aligns with the Agentless paradigm, in which the model must directly output the final patch. In contrast, a commit pack captures the sequence of human-authored commits within a PR, where each commit message (textual reasoning) is paired with the corresponding code modification (action). This structure closely parallels the SWE-Agent setting, where intermediate reasoning steps are interleaved with actions. However, the distinction of the utilities for the two types of data is not absolute: commit messages in a PR commit pack can still inform the model's knowledge and indirectly strengthen its reasoning ability in the Agentless setting.

Natural diff patches. The natural diff patches used in the mid-training data recipe are processed with the following rules:

- Incorporate the agentless prompt template (see Prompts 1,2,3,4; These four prompt templates are also used in the later stages, including cold-start, RL, and test-time self-play), and apply a loss mask to the prompt part. For the localization prompt, the response is set as the files modified in the ground-truth diff patch.
- If a related issue to the PR exists, use its content of the related issue; otherwise, use the PR title as the surrogate of the issue content.
- If a related issue to the PR exists, prepend the issue discussion at the beginning of the output in the code edit response. We aim to strengthen the model's capability of code edit reasoning by leveraging the discussion contents.
- Discard PRs that include modifications to files other than {.py, .md, .rst}.
- For PRs containing {.md, .rst} file modifications, retain only the Python diffs and rewrite them into SEARCH–REPLACE blocks.
- Remove PRs involving file additions or deletions.
- For the code edits with only line insertions or deletions, preserve the original Git diff hunks as the SEARCH content in the SEARCH–REPLACE blocks.
- Ensure that no more than three Python files are modified per PR.
- Apply a filtering script to exclude PRs with non-{.py, .md, .rst} modifications, or PRs modifying more than three Python files.
- Further exclude PRs containing more than five SEARCH-REPLACE blocks.

A total of \sim 50B tokens for natural diff patches are obtained after applying these filtering rules.

```
Please look through the following GitHub problem description and
Repository structure and provide a list of files that one would need
to edit to fix the problem.

### GitHub Problem Description ###
{related issue / PR title content}
```

```
756
     5
757
        ###
     6
758
     7
759
       ### Repository Structure ###
       {file structure induced by the repo snapshot}
760
761
    11
762
       ###
    12
763
    13
764
       Please only provide the full path and return at most 5 files.
765
    15
       The returned files should be separated by new lines ordered by most to
           least important and wrapped with '''
766
       For example:
    16
767
        . . .
    17
768
    18
       file1.py
769
       file2.py
    19
    20
770
771
```

Listing 1: Agentless prompt template: Localization for BugFixer.

```
773
       Please look through the following GitHub problem description and
774
            Repository structure and provide a list of test files that should be
775
            run after applying the patch to fix the issue.
776
        ### GitHub Problem Description ###
    23
777
    24
        {related issue / PR title content}
778
779
        ###
    26
780
    27
781
       ### Repository Structure ###
    28
       {file structure induced by the repo snapshot}
782
    29
    30
783
    31
784
       ###
    32
785
    33
786
       Please only provide the full path and return at most 5 files.
       The returned files should be separated by new lines ordered by most to
787
            least important and wrapped with '''
788
       For example:
789
    37
        . . .
790
    38
       file1.py
791
    39
       file2.py
        • • •
    40
792
793
```

Listing 2: Agentless prompt template: Localization for TestWriter.

```
795
        We are currently solving the following issue within our repository.
796
          Here is the issue text:
797
        --- BEGIN ISSUE ---
798
    43
        {related issue / PR title content}
799
        --- END ISSUE ---
    44
    45
800
    46
801
        Below are some code segments, each from a relevant file. One or more of
    47
802
             these files may contain bugs.
803
         -- BEGIN FILE ---
804
    49
    50
805
        ### {filename1}
    51
806
        {content of filename1}
    52
807
    53
808
        ### {filename2}
809
        {content of filename2}
    56
       { . . . }
```

```
810
811
        . . .
    58
812
       --- END FILE ---
    59
813
    60
       Please first localize the bug based on the issue statement, and then
    61
814
           generate *SEARCH/REPLACE* edits to fix the issue.
815
816
       Every *SEARCH/REPLACE* edit must use this format:
    63
817
       1. The file path
818
       2. The start of search block: <<<< SEARCH
       3. A contiguous chunk of lines to search for in the existing source
819
           code
820
       4. The dividing line: ======
821
       5. The lines to replace into the source code
822
       6. The end of the replace block: >>>>> REPLACE
823
    70
    71
       Here is an example:
824
    72
825
        '''python
    73
826
       ### mathweb/flask/app.py
    74
827
       75
828
    76
       from flask import Flask
829
       import math
830
       from flask import Flask
    79
831
       >>>>> REPLACE
    80
832
833
       Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
    83
834
            If you would like to add the line ^{\prime}
                                                        print(x)', you must
835
           fully write that out, with all those spaces before the code!
836
       Wrap the *SEARCH/REPLACE* edit in blocks '''python...'''.
837
```

Listing 3: Agentless prompt template: Code edit for BugFixer.

```
839
        We are currently solving the following issue within our repository.
    85
840
           Here is the issue text:
841
        --- BEGIN ISSUE ---
842
        {related issue / PR title content}
    87
843
        --- END ISSUE ---
    88
844
845
        Below are some code segments, each from a relevant test file. One or
846
           more of these files may be added some new tests which can reproduce
847
           the issue.
848
    93
        --- BEGIN FILE ---
849
        . . .
    94
850
        ### {filename1}
    95
851
        {content of filename1}
852
    97
853
        ### {filename2}
    98
        {content of filename2}
    99
854
        {...}
    100
855
    101
856
        . . .
    102
857
    103
        --- END FILE ---
858 104
        Please first localize some possible locations in those test files
859 105
            within the repo, and then generate *SEARCH/REPLACE* edit updates to
860
           the **test** files in the repo, so that the erroneous scenario
861
            described in the problem is reproduced.
862 106
863 107
       Every *SEARCH/REPLACE* edit must use this format:
    108 1. The file path
```

```
864
        2. The start of search block: <<<< SEARCH
865
    110
        3. A contiguous chunk of lines to search for in the existing source
866
            code
        4. The dividing line: ======
867 111
        5. The lines to replace into the source code
868
        6. The end of the replace block: >>>>> REPLACE
869
    114
870
    115
        Here is an example:
871
    116
872
    117
        '''python
        ### mathweb/flask/app.py
    118
873
        <<<<< SEARCH
    119
874
        from flask import Flask
    120
875
        _____
876
    122
        import math
        from flask import Flask
877
   123
    124
878
    125
        def test__rules__std_L060_raised() -> None:
879
    126
            try:
880
                sql = "SELECT
                                 IFNULL (NULL, 100),
   127
881
                    NVL(NULL, 100);"
   128
882 129
                result = lint(sql, rules=["L060"])
                assert len(result) == 2
    130
883
            except:
    131
884
                print("Other issues")
    132
885
                return
886
    134
887
    135
                assert result[0]["description"] == "Use 'COALESCE' instead of '
    136
888
                    IFNULL'."
889
                assert result[1]["description"] == "Use 'COALESCE' instead of '
890
                    NVL'."
891 <sub>138</sub>
                print("Issue resolved")
892 139
            except AssertionError:
                print("Issue reproduced")
   140
893
    141
                return
894
    142
895
    143
            return
896
        >>>>> REPLACE
   144
   145
897
    146
898
        Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
    147
899
            If you would like to add the line ' print(x)', you must
900
            fully write that out, with all those spaces before the code!
901
        Wrap the *SEARCH/REPLACE* edit in blocks '''python...'''.
902
```

Listing 4: Agentless prompt template: Code edit for TestWriter.

PR commit packs. The PR commit packs used in the mid-training data recipe are processed with the following rules:

• Discard PRs that include modifications to files other than {.py, .md, .rst}.

- For {.md, .rst} file modifications, retain the "diff –git" signature but remove the actual content changes.
- Ensure that each PR modifies at most five Python files (with at least one required). PRs
 exceeding this limit are discarded.
- Apply a filtering script to exclude PRs containing non-{.py, .md, .rst} file modifications or those modifying more than five Python files.
- Filter out all of the developer signatures and GitHub IDs for ethics considerations.

A total of \sim 20B tokens for PR commit packs are obtained after applying these filtering rules.

In addition, we incorporate synthetic data to further enhance both the reasoning and agentic capabilities of the model. A key observation is that the ground-truth reward for the localization stage in the Agentless setting can be derived directly from the diff patch, since the set of files requiring modification is explicitly indicated.

Synthetic reasoning data. To improve reasoning quality, we perform a lightweight SFT of the Qwen-2.5-72B-Instruct model on 2,000 R1 trajectories. The resulting model is then used to generate large-scale rollouts for the localization stage of both BugFixer and TestWriter. We retain only the rollouts that achieve exactly correct file localizations. This procedure yields approximately $\sim 10B$ tokens of reasoning-intensive data dedicated to Agentless localization in the mid-training recipe.

Synthetic agentic interactions. To strengthen agentic capabilities, we simulate agent—environment interactions with a custom tool set designed to mimic file-system operations without execution. This design is motivated by practical constraints: while repository snapshots from GitHub are available, not all snapshots are equipped with an executable Docker environment. As a result, shell commands are disabled. Instead, we introduce synthetic tools that allow the agent to view file contents and perform keyword-based search for localization, which effectively reproduces the first stage of Agentless but in an agentic manner. The specification of this tool set is covered in the system prompt, which is then used to elicit agentic interaction rollouts from the Qwen-2.5-72B-Instruct model. The complete system prompt is provided in Prompt 5. We apply a loss mask only to the system prompt, and enable the model to simultaneously learn both actions and observations along the trajectory, inspired by Yang et al. (2024b). This approach integrates both policy and world modeling into mid training.

```
938
       Your job is to look through the given GitHub problem description and
939
           Repository structure, and edit updates to the files in the repo to
940
           resolve the problem.
941
    150
       The job is divided into two stages:
        + In Stage 1, you should localize the files the files that you would
942
           need to edit to fix the problem.
943
    152
        + In Stage 2, you should edit the updates to the repo.
944
       Let's begin from Stage 1 to localize the bugs:
    153
945
    154
946
    155
       In Stage 1, besides reading the provided Repository structure, you can
           use the following skills for exploration. The skills are to be
947
           called in an environment wrapped by <execute> and </execute>, listed
948
             in the form of python functions as below:
949
    156
950
       open_file(path: str, is_all | None = False, line_number: int | None =
    157
           1, context_lines: int | None = 100) -> None:
951
            Opens the file at the given path in the editor for exploration.
    158
952
    159
            By default, only the first 100 lines of the file are displayed. To
953
                open the entire file, set 'is_all' to 'True'.
954
                'context_lines' parameter determines the maximum number of
    160
955
                lines to be displayed, with a cap of 100 lines. Use 'scroll_up'
956
                and 'scroll_down' to view more content up or down.
            If a 'line_number' is provided, the window will be moved to include
    161
957
                that line.
958
            Note: When 'is_all' is set to 'True', the 'line_number' and '
    162
959
                context_lines' parameters will not take effect, as the entire
960
                file will be opened and displayed without any line-specific
961
                focus or context limitation.
            Args:
962
    163
            path: str: The path to the file to open. the full path of the
    164
963
                filename should be provided.
964
            is_all: bool | None = False: If set to 'True', the entire file will
    165
965
                be opened. Defaults to 'False'.
            line_number: int | None = 1: The line number to move to. Defaults
966
    166
967
            context_lines: int | None = 100: Only shows this number of lines in
    167
968
                the context window (usually from line 1), with line_number as
969
                the center (if possible). Defaults to 100.
970
971
       goto_line(line_number: int) -> None:
    169
           Moves the window to show the specified line number.
    170
```

```
972
            Args:
973
            line_number: int: The line number to move to.
    172
974 173
975 174
        goto_class_or_func(class_or_func_name: str) -> None:
976 175
            Moves the window to show the specified class or function in the
                current open file.
977
    176
            Aras:
978
            class_or_func_name: str: The name of the given class, function, or
    177
979
                method in a class to move to.
980 178
        scroll_down() -> None:
981 179
            Moves the window down by 100 lines.
    180
982
            Args:
    181
983
            None
    182
984 <sub>183</sub>
        scroll_up() -> None:
985 184
            Moves the window up by 100 lines.
986 185
            Args:
    186
987
    187
            None
988
    188
989 189
        search_dir(search_term: str, dir_path: str | None) -> None:
990 190
            Searches for search_term in all files in dir. If dir is not
                provided, searches in the entire repository. Filename, fine-
991
                grained line number, and the relative class or function it is
992
                located in (if applied) will be shown for each found position.
993
    191
            Aras:
994 192
            search_term: str: The term to search for.
995 193
            dir_path: str: The path to the directory to search. Should be full
                path filename.
996
    194
997
        search_file(search_term: str, file_path: str | None = None) -> None:
    195
998
            Searches for search_term in file. If file is not provided, searches
999
                 in the current open file. Filename, fine-grained line number,
                and the relative class or function it is located in (if applied)
1000
                 will be shown for each found position.
1002 <sup>197</sup>
            Args:
    198
            search_term: str: The term to search for.
1003 199
            file_path: str | None: The path to the file to search. Should be
1004
                full path filename if provided.
1005 200
1006 201
        find_file(file_name: str, dir_path: str | None) -> None:
1007 202
            Finds all files with the given name in the specified directory. If
                dir is not provided, find in the entire repository.
1008 203
            Args:
1009 204
            file_name: str: The name of the file to find.
1010 205
            dir_path: str: The path to the directory to search.
1011 <sup>206</sup>
1012 207
        str_replace(path: str, old_str, new_str)
        old_str=[the old content to be replaced]
1013 <sub>209</sub>
        new_str=[the new content after replacement]
1014 <sub>210</sub>
        -> None:
1015 211
            Replace the old content (old_str) in the file at the given path
                with the new content (new_str). This is the skill that you will
1016
1017 212
                be using to edit the updates.
1018 <sup>-</sup> <sub>213</sub>
            path: str: The path to the file to be updated. The full path of the
1019
                 filename should be provided.
            old_str: str: The old content to be replaced. Note that this
1020 214
                argument should be written in a new line starting with "old_str
1021
                =", and the string content should not be quoted.
1022
            new_str: str: The new content after replacement. Note that this
    215
1023
                argument should be written in a new line starting with "new_str
1024
                =", and the string content should not be quoted.
1025 216
            Example:
```

```
Assuming a call is shown as follows:
1027 219
1028 <sub>220</sub>
        str_replace("filename.py", old_str, new_str)
1029 221
        old str=
1030 <sup>222</sup>
1031 223
        new_str=
    224
        С
1032 225
        . . .
1033 <sub>226</sub>
            Then it will function as replacing the '
                                                          a\n' string with the '
                   b\nc ' string in the 'filename.py' file.
1034
1035 227
1036 228
        insert(path: str, insert_line: int, new_str)
1036
1037 229
230
        new_str=[the new content to be inserted]
        -> None:
1038 231
            Insert the new content (new_str) in the file at the given path.
1039
                When you want to add an entirely new class/function to the file,
                 it would be better to use this method.
1041 232
            Args:
1042
            path: str: The path to the file to be updated. The full path of the
                filename should be provided.
1043 <sub>234</sub>
            insert_line: int: The Line number below which the new content is to
1044
                 be added. This Line number should be within the range of lines
                of the file: [0, Lines_of_the_File]. Specifically, when
1045
                insert_line = 0, the added content starts from the top of the
1046
                file.
            new_str: str: The new content to be inserted. Note that this
1048
                argument should be written in a new line starting with "new_str
1049
                =", and the string content should not be quoted.
1050 236
1051 237
            Example:
            Assuming a call is shown as follows:
1052 239
1053 <sub>240</sub>
        insert("test_filename.py", 5, new_str)
1054 241
        new_str= def test_add():
1055 242
               assert add(1, 2) == 3
1056 243
1057
            Then it will function as inserting the string ' def test_add():\
                          assert add(1, 2) == 3' below the Line 5 of the '
               n
1058
                test_filename.py 'file.
1059 245
1060 <sup>246</sup>
        stop() -> None:
1061 247
            Terminate the editing process.
1061 248
1062 249
            Args:
            None
1063 <sub>250</sub>
        NOTE:
1064 251
        Responses should be concise.
1065 252
        When exploring, you should attempt fewer things at a time: Include ONLY
1066 253
            ONE <execute> per response, and use a SINGLE skill listed above
1067
            within the <execute> environment. DO NOT use other python functions,
1068
            as the environment does not support them.
        You should first reason in the verbal form, then use a skill with <
1069 254
            execute> and </execute>.
1071 255
        You should avoid apologies and thanks in the responses.
    256
1072 <sub>257</sub>
1073 258
        When you finish exploring and analyzing with the provided skills,
           please return at most 3 files with the full path only. Each full
           path should be placed in a single line, INSTEAD OF BROKEN WITH
1075
           MULTIPLE LINES.
1077 <sup>259</sup>
1076
        The returned files should be separated by new lines ordered by most to
           least important, wrapped with ''' and NOTHING ELSE.
1078 <sub>260</sub>
        An example for a full output:
1079 261
   262 | full_path_to_file1.py
```

```
1080
           full_path_to_file2.py
1081 <sub>264</sub>
1082 <sub>265</sub>
1083 266
          Now Let's start!
1084 <sup>267</sup>
1085 268
           ### GitHub Problem Description ###
1086
     270
           {issue content}
1087 271
1088 272
          ### Repository Structure ###
1089 <sup>273</sup>
           {file structure}
1090
     275
1091 <sub>276</sub>
          ###
1092
```

Listing 5: A non-execution set of tools empowering the simulation of agentic interaction trajectories.

After completing the initial localization stage, the agent is guided into the code-editing phase through a follow-up instruction: "Now let's move on to Stage 2 and edit the updates. Remember, you can still decide at any point whether a file actually requires modification." We retain partial rollouts from Stage 1, provided that the localization results include at least one correct file.

In Stage 2, we first simulate the agent's interaction by allowing it to open incorrectly localized files, and we artificially inject agentic reasoning patterns such as "I realize that I do not need to modify this file" after inspecting the file content. This procedure is designed to strengthen the self-reflection ability of the agent by exposing it to false-positive contexts regarding the issue to be solved.

Subsequently, we transcribe the ground-truth PR commit pack into trajectory form: each commit message is treated as the agent's reasoning step, and each code update is represented as the corresponding action, expressed through the "str_replace" or "insert" tools. These interactions are appended to the trajectory, followed by a terminating "stop" call. Due to storage constraints on repository snapshots, this trajectory simulation is applied to only a subset of PRs. Overall, this process contributes approximately $\sim \! 10B$ tokens of agentic interaction data to the mid-training recipe. Future directions for scaling this component in the data recipe include leveraging the idea of environment scaling (Yang et al., 2025c).

Training. We perform mid-training using a standard next token prediction approach, initialized from the Qwen2.5-72B-Base (Qwen et al., 2024) model. We upsample the synthetic part of the data by a factor of 4 during mid-training, inspired by the practice in Grattafiori et al. (2024); Qwen et al. (2024); Gu et al. (2025). A global batch size of 256 with a maximum sequence length of 32K tokens is used, optimizing for long-context capabilities necessary for real-world software engineering tasks. The learning rate is set to 2e-5, with a cosine decay schedule and a minimum learning rate of 2e-6. The warm-up phase covers over approximately 3 billion tokens, followed by learning rate decay until approximately 150 billion tokens are processed. The training was conducted on 256 NVIDIA H100 GPUs and lasted for 213 hours.

B DOCKER ENVIRONMENTS

Table 3: The sources of the docker environments used in the development of Kimi-Dev.

Dataset Name	Dataset Link	Number of Dockers
SWE-Gym (Pan et al. (2024))	https://huggingface.co/datasets/SWE-Gym/	2,356
SWE-bench-extra (Badertdinov et al. (2024a))	SWE-Gym/ https://huggingface.co/ datasets/nebius/ SWE-bench-extra/	3,846
R2E-Gym-Lite (Jain et al. (2025))	https://huggingface.co/ datasets/R2E-Gym/ R2E-Gym-Lite	3,671

Docker environment construction. To validate non-ground-truth patches generated by model rollouts and expand our dataset, we required executable Docker environments. We combined publicly available datasets with custom-configured Docker environments (see Table. 3). Among them, SWE-Gym and R2E-Gym-Lite open-source their dockers that we can directly use. For datasets lacking Docker support (SWE-Bench-Extra), we implemented an automated configuration method:

- 1. Initialize a Docker environment with fixed dependencies.
- 2. Select Python version based on commit year.
- 3. Install dependencies via requirements.txt and "pip install -e .".
- 4. Resolve ModuleNotFound errors during test execution.
- 5. Validate success if a FAIL_TO_PASS test transitions from failing (without gt_patch) to passing (with gt_patch).

Out of 6.38k SWE-bench-extra instances, 3,846 environments are successfully constructed and subsequently used for cold-start and RL training.

C More Details of RL training

C.1 RL TRAINING OBJECTIVE

For the RL algorithm, we adopt the policy optimization method proposed by Kimi k1.5 (Team et al., 2025), which has shown promising results on reasoning tasks in both math and code. Unlike GRPO (Shao et al., 2024), which eliminates the critic by using average rewards for advantage estimation, Kimi k1.5 (Team et al., 2025) adopts a simpler policy gradient approach. Specifically, we optimize the policy π_{θ} using the following objective:

$$\mathcal{J}_{\theta} = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta}(\cdot|q)} \left\{ \sum_{i=1}^G \left[\left(R_i(q, o_i) - \text{mean}(\{R_j\}_{j=1}^G) - \tau \log \frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{ref}}(o_i|q)} \right)^2 \right], \right\}$$
(2)

where q is a prompt sampled from the prompt data distribution P(Q). $\{o_i\}_{i=1}^G$ are G responses generated by the current policy π_θ for prompt q. $R_i(q,o_i)$ is the reward for response o_i . The mean reward across the G responses is used as a baseline to center the reward (similar to advantage estimation). The final term penalizes divergence from a reference policy $\pi_{\theta_{\rm ref}}$, with weight τ . This formulation avoids the need for a critic or value model by using group-wise relative rewards, and supports stable updates through KL regularization.

C.2 PROMPT SET SELECTION

In the main text, we introduce the adaptive prompt selection method for RL training. Specifically, we construct an initial prompt set of 1,200 problems by selecting those with pass@16 > 0 from SWE-Gym (Pan et al., 2024), SWE-bench-extra (Badertdinov et al., 2025), and R2E-gym (Jain et al., 2025). Then, every 100 training steps, we expand the prompt set by adding 500 new problems. These additional problems are randomly sampled from the pool of problems for which the current model has pass@16 = 0, thereby progressively increasing the difficulty.

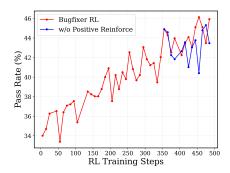


Figure 7: Ablation of positive example reinforcement during 72B Bugfixer RL.

C.3 RL EXPERIMENT ABLATION

Figure 8 shows the performance of the Qwen2.5-14B

model in RL experiments, where both the BugFixer and the TestWriter exhibit clear scaling law behavior.

Furthermore, Figure 7 illustrates the effect of our proposed positive example reinforcement. We incorporated this improvement in the later stage of training and found that the RL of BugFixer

achieved better performance compared to not using this trick. The main reason is that positive examples can reinforce the correct reasoning patterns, enabling the model to converge more quickly, especially when its exploration ability diminishes in the later stages. It is noticed that a similar trick was also applied in the development of the Seed1.5-thinking model (Seed et al., 2025).

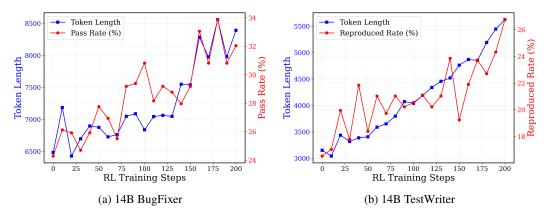


Figure 8: RL scaling experiments on Qwen2.5-14B model.

D AGENT INFRASTRUCTURE

We present a unified sandbox architecture designed for hybrid cloud cluster environments, leveraging Kubernetes orchestration to provide researchers with a flexible, secure, and highly scalable code execution platform. The architecture is accessible through a Python SDK and implements an ephemeral, use-and-destroy paradigm that ensures resource efficiency and security isolation. The system demonstrates exceptional scalability, supporting over 10,000 concurrent sandbox instances while maintaining performance stability. This infrastructure serves as a foundational platform for diverse computational scenarios, including automated code evaluation and RL experiments.

The system employs a sidecar container pattern to efficiently orchestrate and manage a heterogeneous collection of over 25,000 docker images sourced from multiple datasets. This architectural choice enables optimal resource utilization while maintaining isolation between different execution environments. Additionally, we have developed a proprietary Software Engineering (SWE) image-building pipeline that continuously expands the repository of supported images, thereby enhancing the system's adaptability to diverse computational workloads and research requirements.

E SKILL GENERALIZATION

In this section, we study whether the skills incentivized by Agentless training and SWE-Agent adaptation through SWE-Smith trajectories could transfer to other SWE tasks. We use SWE-bench-live and SWE-bench Multilingual as our testbeds.

SWE-bench-live (Zhang et al., 2025) is a benchmark for real-world issue resolution, evaluating AI systems on software engineering tasks. Using an automated curation pipeline, it is updated monthly to ensure fresh tasks and rigorous, contamination-free evaluation. For our experiments, we selected the default set of 300 tasks, with data collected between October 2024 and March 2025. Compared to SWE-bench Verified, SWE-bench-live exhibits a higher degree of distributional shift.

SWE-bench Multilingual (Yang et al., 2025c) introduces 300 curated tasks from 42 GitHub repositories across 9 programming languages, including Rust, Java, PHP, Ruby, JavaScript/TypeScript, Go, and C/C++, covering domains such as web frameworks, data tools, core utilities, and libraries. Compared to SWE-bench Verified, which focuses exclusively on Python, SWE-bench Multilingual exhibits greater linguistic and domain diversity, posing additional challenges in cross-language generalization and transferability of software engineering capabilities.

Similar to previous experiments, we evaluated four model stages as the priors: the original Qwen2.5-72B (Base), the mid-trained model (MT), the model activated with reasoning data through super-

vised finetuning (SFT), and the model after RL training (RL). We still use the open-source SWE-smith trajectories to activate the agentic capabilities of each prior.

Figures 9 and 10 show the performance of the four priors on SWE-bench-Live and SWE-bench Multilingual under varied amounts of agentic trajectories for adaptation (2^{21} as one-step gradient descent, 2^{23} , 1.1×2^{27} , and 1.5×2^{28} as 100, 2,000, and 5,016 training trajectories). Each SWE-Agent adaptation experiment is conducted throught lightweight supervised finetuning, the training time of which ranges from several minutes to two hours at most.

Compared to the Base prior, those specifically enhanced with Agentless skills (SFT and RL) demonstrate stronger task generalization, especially under the data-scarce settings. However, when more SWE-Smith trajectories are used for adaptation, the performances of the Base and the MT priors become closer to those of the SFT and the RL priors. This could be attributed to the gaps between the different SWE tasks. The exploration for recipes that enable stronger out-of-distribution and task-agnostic generalization is left for future work.

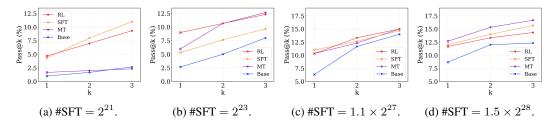


Figure 9: Generalization analysis on SWE-bench-Live.

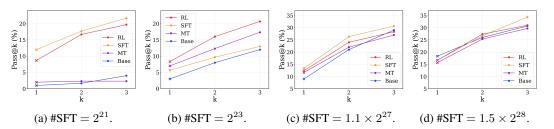


Figure 10: Generalization analysis on SWE-bench Multilingual.

F CASE STUDY

F.1 BUGFIXER TRAJECTORIES

We provide a Bugfixer's response trajectory during the code repair stage (see Listing 6). From this, we can observe that the initial code given was return self[i, j - pos - other.cols], but the model quickly realized its mistake and corrected it to return self[i, j - other.cols]. Although some repetitive self-reflections followed, the increasing response length indeed improved answer accuracy and robustness in our experiments.

```
⊲think⊳
1287
       Okay, let's tackle this problem. The user reported an issue where the
1288
           Matrix.col_insert() method isn't working correctly. The example
           given shows that after inserting a matrix V into M at position 3,
1290
           the resulting matrix has the inserted columns shifted incorrectly.
           The expected result would have the inserted columns at positions 3
1291
           and 4, but the actual output shows them starting at position 3, but
1292
           the original matrix's columns after the insert are shifted upwards
1293
           instead of remaining in their original positions.
1294 <sub>279</sub>
1295 280
       First, I need to understand how col_insert works. Looking at the code
          provided, especially in the MatrixShaping class in sympy/matrices/
```

```
1296
            common.py, the _eval_col_insert method is responsible for inserting
1297
            columns. The current implementation shifts the existing columns to
1298
            the right by the number of columns inserted. However, the bug might
1299
            be in how the original matrix's columns are being shifted.
1300 <sup>281</sup>
1301 <sup>282</sup>
        Let me look at the existing code for _eval_col_insert:
    283
1302
        def _eval_col_insert(self, pos, other):
    284
1303 <sub>285</sub>
            cols = self.cols
1304 286
1305 287
            def entry(i, j):
1306 <sup>288</sup>
                if j < pos:
1307 289
1307 290
                     return self[i, j]
                 elif pos <= j < pos + other.cols:
1308 291
                     return other[i, j - pos]
1309 292
                 return self[i, j - pos - other.cols]
1310 293
1311 294
            return self._new(self.rows, self.cols + other.cols,
    295
                               lambda i, j: entry(i, j))
1312 296
1313 297
        Wait, the problem here is in the last case. When j is greater than or
1314
            equal to pos + other.cols, the code subtracts (pos + other.cols)
            from j. But that's incorrect because the original columns after the
1315
            inserted ones should start at j = pos + other.cols. So for the
1316
            original matrix's columns that come after the inserted ones, their
1317
            new positions should be j + other.cols. But in the current code,
1318
            when j is in the original matrix's columns (after the inserted ones)
1319
            , the code is subtracting (pos + other.cols), which would
            incorrectly shift them.
1320
1321 298
        Wait, let's think again. The original code for the entry function:
1322 <sub>300</sub>
1323 301
        if j < pos:
1324 302
            return self[i, j] # original columns before pos
1325 303
        elif pos <= j < pos + other.cols:
1326 304
        return other[i, j - pos] \# inserted columns return self[i, j - pos - other.cols] \# original columns after the
    305
1327
            inserted ones
1328 306
        Wait, no. Let's say the original matrix has cols = 6. We insert 2
1329 307
            columns at pos=3. The new total cols is 6 + 2 = 8. The original
1330
            columns after pos 3 (columns 3,4,5) should now be at positions
1331
            5,6,7. But according to the code, when j is >=3 +2=5 (since pos +
1332
            other.cols =3+2=5), then the code returns self[i, j -5 -2] = self[i,
1333
             j-7]. Wait, that's not right. For example, if j=5 (the new column
            5), then j - pos - other.cols =5 -3 -2=0. So the original column 0
1334
            would be mapped to the new column 5. That's definitely wrong.
1335
1336 308
        Ah, here's the mistake. The code is subtracting (pos + other.cols) from
1337
            j, but that's incorrect. The original columns after the inserted
1338
            ones should be shifted to the right by other.cols. So, the correct
1339
            calculation should be j - other.cols. For example, original column 3
             becomes 3 + 2 = 5, original column 4 becomes 6, etc.
1340
1341 310
    311
        So the line in the code:
1342 312
1343 313
        return self[i, j - pos - other.cols]
1344 314
1345 315
        should be:
1346 316
1346
1347 317
318
        return self[i, j - other.cols]
1348 319
        Because the inserted columns have taken up 'other.cols' positions
        starting at 'pos', so the original columns after 'pos' are shifted
```

```
1350
            by 'other.cols' to the right. Therefore, to get the correct original
1351
             column index, we subtract 'other.cols' from the current j.
1352 320
1353 321
        So the fix is to change the subtraction from (pos + other.cols) to
            other.cols. Let me check:
1354
1355 322
    323
        Original code:
1356 324
1357 325
        return self[i, j - pos - other.cols]
1358 326
1359 327
        Corrected code:
1360 328
        return self[i, j - other.cols]
    329
1361 330
1362 <sub>331</sub>
        Wait, no. Wait, the original columns after the inserted block (which
1363
           starts at pos and has other.cols columns) would start at j = pos +
            other.cols. So for those j values, the original column index would
1364
           be j - (pos + other.cols). Wait, no. Let's think with numbers.
1365
1366 333
        Original matrix has 6 columns. Insert 2 columns at position 3. The new
1367
           matrix has 8 columns. The original columns 0,1,2 remain in positions
            0,1,2. The inserted 2 columns are at positions 3 and 4. The
1368
            original columns 3,4,5 now become 5,6,7.
1369
1370 334
        So for j in 5,6,7 (original columns 3,4,5), the mapping is:
1371 336
1372 337
        5 $\rightarrow$ 3
1373 338
        6 $\rightarrow$ 4
1374 339
1375 340
        7 $\rightarrow$ 5
1376 342
1377 343
        So the formula should be original_column = j - inserted_columns. In
           this case, inserted_columns is other.cols. So the code should
1378
            subtract other.cols from j when accessing the original matrix's
1379
            columns after the inserted block.
1380
1381 345
        Therefore, the line:
1382 346
1383 347
        return self[i, j - pos - other.cols]
1384 348
1385 349
        should be:
    350
1386 351
        return self[i, j - other.cols]
1387 352
        Because pos is where the insertion starts, but the original columns
1388 353
           after the inserted block are shifted by other.cols. Let's verify
1389
           with the example:
1391 355
        pos =3, other.cols=2. For j=5 (new column 5), original column index is
1392
            5 -2=3, which is correct. For j=6, 6-2=4; j=7, 7-2=5. That's correct
1393
1394 <sup>356</sup>
1395 357
        So the error in the original code was subtracting (pos + other.cols)
            instead of other.cols. The pos is the starting position of the
1396
           insertion, but after inserting other.cols columns, the original
1397
            columns after the inserted block are shifted by other.cols.
1398
           Therefore, the correct subtraction is other.cols.
1399 <sup>358</sup>
1400 359
        Thus, the fix is to change the line in the _eval_col_insert method from
1401 360
1402 361
        return self[i, j - pos - other.cols]
1403 362
       to:
    363
```

```
1404
1405 <sub>365</sub>
        return self[i, j - other.cols]
1406 366
1407 367
        Now, looking back at the code in the _eval_col_insert function:
1408 368
1409 369
        The code for the else clause is:
    370
1410 371
        return self[i, j - pos - other.cols]
1411 <sub>372</sub>
1412 373
        Which is incorrect. Changing that to j - other.cols should fix the
            problem.
1413
1414 374
    375
        So the SEARCH block would look for the lines:
1415 376
1416 377
             def _eval_col_insert(self, pos, other):
1417 378
                cols = self.cols
1418 379
1419 380
                 def entry(i, j):
    381
                     if j < pos:
1420 382
                          return self[i, j]
1421 383
                     elif pos <= j < pos + other.cols:
                         return other[i, j - pos]
1422 384
1423 385
                     return self[i, j - pos - other.cols]
1424 386
                 return self._new(self.rows, self.cols + other.cols,
    387
1425 388
                                    lambda i, j: entry(i, j))
1426 389
1427 390
        And the REPLACE block would change the last part to:
1428 391
1429 392
                     elif pos <= j < pos + other.cols:
                         return other[i, j - pos]
1430 394
                     return self[i, j - other.cols]
1431 395
        So the correction is in the return statement of the entry function. The
1432 396
             original code subtracted (pos + other.cols) from j, but it should
1433
            subtract other.cols instead.
1434
1435 398
        Therefore, the correct SEARCH/REPLACE block would be:
1436 399
1437 400
        ### sympy/matrices/common.py
1438 401
        1439 402
            def _eval_col_insert(self, pos, other):
                cols = self.cols
    403
1440 404
1441 405
                 def entry(i, j):
                     if j < pos:
1442 406
1443 <sup>407</sup>
                         return self[i, j]
1444 408
                     elif pos <= j < pos + other.cols:
                         return other[i, j - pos]
1445 410
                     return self[i, j - pos - other.cols]
1446 411
1447 412
                 return self._new(self.rows, self.cols + other.cols,
1448 413
                                    lambda i, j: entry(i, j))
1449 <sup>414</sup>
             def _eval_col_insert(self, pos, other):
    415
1450 416
                 def entry(i, j):
1451 417
                     if j < pos:
1452 418
                          return self[i, j]
1453 419
                     elif pos <= j < pos + other.cols:
1454 420
                         return other[i, j - pos]
    421
                     return self[i, j - other.cols]
1455 + 422
1456 423
                 return self._new(self.rows, self.cols + other.cols,
                                   lambda i, j: entry(i, j))
1457 424
   425 |>>>>> REPLACE
```

```
1458
    426
1459
        This change corrects the column shifting calculation for the original
    427
1460
            matrix's columns after the inserted block. Instead of subtracting (
1461
            pos + other.cols), we subtract other.cols, which properly shifts the
             original columns to the right by the number of inserted columns.
1462
1463 428

√think

    429
1464
        '''python
    430
1465 431
        ### sympy/matrices/common.py
1466 432
        1467 433
            def _eval_col_insert(self, pos, other):
1468 434
                 cols = self.cols
    435
1469 436
                 def entry(i, j):
1470 437
                     if j < pos:
1471 438
                         return self[i, j]
                     elif pos <= j < pos + other.cols:
1472 439
1473 440
                         return other[i, j - pos]
    441
                     return self[i, j - pos - other.cols]
1474 442
1475 443
                 return self._new(self.rows, self.cols + other.cols,
1476 444
                                   lambda i, j: entry(i, j))
1477 445
1478 446
            def _eval_col_insert(self, pos, other):
                def entry(i, j):
    447
1479 448
                     if j < pos:
1480 449
                         return self[i, j]
1481 450
                     elif pos <= j < pos + other.cols:
                         return other[i, j - pos]
1482 <sup>451</sup>
1483 452
                     return self[i, j - other.cols]
1484 454
                 return self._new(self.rows, self.cols + other.cols,
1485 455
                                   lambda i, j: entry(i, j))
1486 456
        >>>>>> REPLACE
1487 457
```

Listing 6: Reasoning trajectory example for BugFixer.

F.2 TESTWRITER FALSE POSITIVE CASES

1488

1489 1490 1491

1492

1493

1494

1495

1496

1497

1498 1499

1501

1502

1503 1504

1505

1506

1507

1508

During the RL training process, we observed that the Testwriter component exhibits some false positives, which can result in tests that contain certain gaps. This might cause the test patches generated by the model to be mistakenly considered as correct. Consequently, the performance improvement of Testwriter does not accurately reflect the actual performance. In this section, we provide two intuitive examples to illustrate this phenomenon, and leave further enhancement for TestWriter as future work.

Listing 7 shows the ground-truth test patch for instance django_django-16877, with Listing 8 showing the test patch generated by the TestWriter. A detailed comparison reveals that the model-generated test cases are not comprehensive: Two locations should have been tested, but only one was checked, and the effect of mark_safe was not considered. In practice, this allowed the model-predicted patch to pass the TestWriter reward check, but result in a false positive.

The second example instance is sympy_sympy-13974, with Listings 9 and 10 showing the ground-truth and model-generated test patches. In this case, the coverage of the generated test is also insufficient. The generated test does not account for the need to satisfy the associativity property, e.g., assert tensor_product_simp(TP(A, B)**x) == TP(A**x, B**x). As a consequence, it fails to serve as a reliable judge for BugFixer.

```
1509
1510
diff --git a/tests/template_tests/filter_tests/test_escapeseq.py b/
tests/template_tests/filter_tests/test_escapeseq.py
1511
2 new file mode 100644
--- /dev/null
```

```
1512
       +++ b/tests/template_tests/filter_tests/test_escapeseq.py
1513
       @@ -0,0 +1,59 @@
       +from django.test import SimpleTestCase
1515
       +from django.utils.safestring import mark_safe
1516
        +from ..utils import setup
1517
    10
1518
    11
1519
    12
        +class EscapeseqTests(SimpleTestCase):
1520
    13
             The "escapeseq" filter works the same whether autoescape is on or
    14
1521
            off,
1522
             and has no effect on strings already marked as safe.
1523
1524
    17
             @setup(
1525 18
    19
1526
                      "escapeseq_basic": (
    20
1527
                          '{{ a|escapeseq|join:", " }} -- {{ b|escapeseq|join:",
1528
              } } ′
1529 22
                      ),
1530 23
1531 <sup>24</sup>
             def test_basic(self):
1532
    26
                 output = self.engine.render_to_string(
1533
    27
                      "escapeseq_basic",
1534
                      {"a": ["x&y", ""], "b": [mark_safe("x&y"), mark_safe("<
    28
            p>")]},
1535
1536
    29
                 self.assertEqual(output, "x&y, <p&gt; -- x&y, ")
    30
1537
    31
1538
    32
             @setup(
1539 33
                      "escapeseq_autoescape_off": (
1540 34
                          '{% autoescape off %}{{ a|escapeseq|join:", " }}'
1541 35
    36
1542
                          '{{ b|escapeseq|join:", "}}{% endautoescape %}'
    37
1543
    38
                      )
1544 39
1545 40
1546 <sup>41</sup>
             def test_autoescape_off(self):
    42
                 output = self.engine.render_to_string(
1547
                      "escapeseq_autoescape_off",
    43
1548
                      {"a": ["x&y", ""], "b": [mark_safe("x&y"), mark_safe("<
1549
            p>")]},
1550 45
                 self.assertEqual(output, "x&y, <p&gt; -- x&y, ")
1551
    46
    47
1552
             @setup({"escapeseq_join": '{{ a | escapeseq | join: " < br/>" }}'})
1553
    49
             def test_chain_join(self):
1554 50
                 output = self.engine.render_to_string("escapeseq_join", {"a":
            ["x&y", ""]})
1555
                 self.assertEqual(output, "x&y<br/>&lt;p&gt;")
1556 51
    52
1557
    53
             @setup(
1558
    54
1559
    55
                      "escapeseq_join_autoescape_off": (
                          '{% autoescape off %}{{ a|escapeseq|join:"<br/>" }}{%
1560 56
            endautoescape %}'
1561
    57
                      ),
1562
    58
1563
    59
1564
    60
        +
             def test_chain_join_autoescape_off(self):
1565
    61
                 output = self.engine.render_to_string(
                      "escapeseq_join_autoescape_off", {"a": ["x&y", ""]}
    62
```

```
1566
63
1567
64 + )
+ self.assertEqual(output, "x&y<br/>&lt;p&gt;")
```

Listing 7: GT test patch of django_django-16877.

```
1570
1571
        diff --git a/tests/template_tests/filter_tests/test_safeseq.py b/tests/
           template_tests/filter_tests/test_safeseq.py
1572
         -- a/tests/template_tests/filter_tests/test_safeseq.py
1573
        +++ b/tests/template_tests/filter_tests/test_safeseq.py
1574
       @@ -20,3 +20,20 @@
1575
             def test_safeseq02(self):
                 output = self.engine.render_to_string("safeseq02", {"a": ["&",
1576
                      "<" ] } )
1577
                 self.assertEqual(output, "&, < -- &, <")</pre>
1578
1579
             # New tests for escapeseq
     9
1580
    10
             @setup({"escapeseq01": '{{ a | escapeseq | join:", " }}'})
             def test_escapeseq01(self):
1581
    11
                 output = self.engine.render_to_string("escapeseq01", {"a":
    12
1582
            ["&", "<"]})
                 self.assertEqual(output, "&, <")
    13
1584
    14
1585
    15
             @setup(
1586 16
                      "escapeseq02": (
1587 <sup>17</sup>
                          '{% autoescape off %}{{ a|escapeseq|join:", " }}{%
    18
1588
            endautoescape %}'
1589
    19
1590
    20
1591 21
1592 22
             def test_escapeseq02(self):
                output = self.engine.render_to_string("escapeseq02", {"a":
1593
            ["&", "<"]})
1594
                 self.assertEqual(output, "&, <")
    24
1595
```

Listing 8: Model predicted test patch of django_django-16877.

```
1597
1598
        diff --git a/sympy/physics/quantum/tests/test_tensorproduct.py b/sympy/
            physics/quantum/tests/test_tensorproduct.py
1599
          -- a/sympy/physics/quantum/tests/test_tensorproduct.py
1600
        +++ b/sympy/physics/quantum/tests/test_tensorproduct.py
1601
        @@ -10,7 +10,7 @@
1602
        from sympy.physics.quantum.density import Density
1603
         from sympy.core.trace import Tr
1604
        -A, B, C = symbols('A,B,C', commutative=False)
1605
        +A, B, C, D = symbols('A,B,C,D', commutative=False)
1606
         x = symbols('x')
     10
1607
1608 <sub>12</sub>
         mat1 = Matrix([[1, 2*I], [1 + I, 3]])
        @@ -47,6 +47,11 @@ def test_tensor_product_commutator():
1609 13
1610 <sup>14</sup>
    15
         def test_tensor_product_simp():
1611
             assert tensor_product_simp(TP(A, B) * TP(B, C)) == TP(A*B, B*C)
     16
1612
    17
              # tests for Pow-expressions
1613 <sub>18</sub>
             assert tensor_product_simp(TP(A, B)**x) == TP(A**x, B**x)
             assert tensor_product_simp(x \times TP(A, B) \times 2) == x \times TP(A \times 2, B \times 2)
1614 19
1615 <sup>20</sup>
             assert tensor_product_simp(x*(TP(A, B)**2)*TP(C,D)) == x*TP(A**2*C
            , B**2*D)
1616
             assert tensor_product_simp(TP(A,B)-TP(C,D)**x) == TP(A,B)-TP(C**x,
1617
            D**x)
1618 22
1619 23
         def test_issue_5923():
```

1622

1653 1654 1655

1656 1657

1658

1659

1664

1665

1666

1668

1669

1670

1671

Listing 9: GT test patch for sympy_sympy-13974.

```
1623
        diff --qit a/sympy/physics/quantum/tests/test_tensorproduct.py b/sympy/
1624
           physics/quantum/tests/test_tensorproduct.py
1625
          a/sympy/physics/quantum/tests/test_tensorproduct.py
1626
        +++ b/sympy/physics/quantum/tests/test_tensorproduct.py
1627
       @@ -47,6 +47,27 @@
1628
1629
        def test_tensor_product_simp():
             assert tensor_product_simp(TP(A, B) \starTP(B, C)) == TP(A\starB, B\starC)
1630
             # Test tensor product powers
1632
             a = symbols('a', commutative=False)
    11
             t1 = TP(1, 1) * TP(1, 1)
    12
             t1_simp = tensor_product_simp(t1)
1634
             assert t1_{simp} == TP(1, 1), f"Expected 1x1, got {t1_simp}"
    13
1635
    14
1636
    15
             # Test with Pauli matrices
1637
    16
             from sympy.physics.paulialgebra import Pauli
1638
    17
             t2 = TP(1, Pauli(3)) * TP(1, Pauli(3))
             t2_simp = tensor_product_simp(t2)
    18
1639
             assert t2_simp == TP(1, 1), f"Expected 1x1, got {t2_simp}"
    19
1640
    20
1641
    21
             # Test substitution cases from issue
1642
    22
             expr1 = TP(1, 1) * TP(1, a)
1643
             result1 = expr1.subs(a, 1)
             assert tensor_product_simp(result1) == TP(1, 1), f"Substitution 1
1644
           failed: {result1}"
1645
1646
             expr2 = TP(1, Pauli(3)) * TP(1, a)
1647
    27
             result2 = expr2.subs(a, Pauli(3))
1648
    28
             assert tensor_product_simp(result2) == TP(1, 1), f"Substitution 2
           failed: {result2}"
1649
    29
1650
    30
1651
         def test_issue_5923():
    31
1652
```

Listing 10: Model predicted test patch of sympy_sympy-13974.

EMERGENT PARALLEL SCALING

Compared to the self-play method that leverages execution during test time, parallel scaling (Chen et al., 2025b; Pan et al., 2025) is a recently-introduced test-time scaling method that aggregates multiple independent rollouts for additional improvement. While previous works incentivize the ability of parallel scaling through training, or train a separate reward model (Pan et al., 2024; Jain et al., 2025) for patch selection, we observe that the parallel scaling effect emerges in our model without any specific training. Specifically, patch candidates from parallel rollouts of BugFixer are used to prompt the model to summarize their differences and compose a new bugfix patch. Listing 11 is the prompt template of parallel scaling. In the emplate, promblem_statement is the GitHub issue, and trajs_content represents the content of multiple patch candidates.

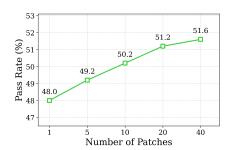


Figure 11: Emergent parallel scaling results on SWE-bench Verified.

```
1672 458
       We are currently solving the following issue within our repository.
1673
           Here is the issue text:
          - BEGIN ISSUE ---
```

```
{problem_statement}
1675 461
        --- END ISSUE ---
1676 462
1677 463
        Below are some thinking trajectories, each from 11m reasoning model.
          Only one trajetory is right.
1678
1679 <sup>464</sup>
        --- BEGIN FILE --
    465
1680
    466
        {trajs_content}
1681 467
        --- END FILE ---
1682 468
1683 469
        Please first summary and analyze the key differences between the
1684 470
            trajectories, and then generate *SEARCH/REPLACE* edits to fix the
1685
            issue.
1686 471
1687 472
        Every *SEARCH/REPLACE* edit must use this format:
        1. The file path
1688 <sup>473</sup>
1689 <sup>474</sup>
        2. The start of search block: <<<< SEARCH
        3. A contiguous chunk of lines to search for in the existing source
1690
            code
1691 476
        4. The dividing line: ======
1692 477
        5. The lines to replace into the source code
1693 478
        6. The end of the replace block: >>>>> REPLACE
1694 479
        Here is an example:
    480
1695 <sub>481</sub>
1696 482
        '''python
        ### mathweb/flask/app.py
1697 483
        <<<<< SEARCH
1698 <sup>484</sup>
1699 485
        from flask import Flask
        -----
1700 487
        import math
1701 488
        from flask import Flask
        >>>>> REPLACE
1702 489
1703 490
1704 491
        Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
    492
1705
             If you would like to add the line '
                                                             print(x)', you must
1706
            fully write that out, with all those spaces before the code!
        Wrap the *SEARCH/REPLACE* edit in blocks '''python...'''.
1707 493
        The summary of the key differences between the trajectories should be
1708 <sup>494</sup>
            in the thinking part.
1709
```

Listing 11: The prompt template for parallel scaling.

1711 1712

1713

1714

1715

1716

1717

The results in Figure 11 show that the performance of the parallel aggregation improves as the number of patch candidates in the prompt increases. The advantage of this scaling paradigm over majority voting lies in its ability to leverage the model's own capacity to analyze multiple candidate patches, thereby surpassing the simplistic approach of weighting answers merely by their frequency of occurrence: Listing 12 covers a full example with 14 different input patch candidates, and how the model successfully recognizes the correct one and generates a novel patch with additional reasoning.

```
1718
1719 495
       Prompt:
1720 496
       We are currently solving the following issue within our repository.
           Here is the issue text:
1721 497
        --- BEGIN ISSUE ---
       Data <@filename isn't relative to the YAML file
1723 499
       The [docs say] (https://gabbi.readthedocs.io/en/latest/format.html#data)
1724
    500
1725 501
       > If the value is a string that begins with <0 then the rest of the
1726
           string is treated as the name of a file to be loaded from the same
1727
           directory as the YAML file.
    502
```

```
But I haven't found it works like this unless I cd into the directory
1729
            containing the yaml file.
1731 505
         '_load_data_file ' says:
        '''python
1732 <sup>506</sup>
1733 507
        path = os.path.join(self.test_directory, os.path.basename(filename))
    508
1734 <sub>509</sub>
        this does a few things:
1735 510
        - makes the path begin with 'self.test_directory' (defaults to '.', the
1736
            current working directory)
1737 511
        - disguards any folders in the <@ 'filename' path
1738 512
        - appends 'filename'
    513
1739 514
        This means, if I'm in '/', and I have a test that says 'data: <@cake.
1740
            jpg', I cannot run my tests as advised in the docs:
         'gabbi-run -- /my/test.yaml /my/other.yaml'
1741 515
1742 516
1743 517
         'FileNotFoundError: [Errno 2] No such file or directory: './cake.jpg'
    518
1744 519
        So in our test running script, we have to start with 'cd tests/'.
1745 520
        My preference, would be to make the path relative to the location of
1746 521
            the yaml file. That way, I can run my tests from whatever directory.
1747
1748 522
1749 <sup>523</sup>
        Obviously this isn't possible when redirecting with '<' as the filename
             is never seen by gabbi. So I'm happy is discuss other ideas.
1750 <sub>524</sub>
1751 525
1752 <sup>526</sup>
1753 527
        Furthermore I can't keep my test data in a child directory, unless I cd
            into that directory like:
1754 <sub>528</sub>
        '''shell
1755 529
        cd yaml_tests/test_data
        gabbi-run -- ../test_things.yaml
1756 530
1757 531
1758 532
1759 533
        So for that reason, I'd like to allow directories like '<@test_data/
            filename.txt' to be included in the path.
1760 <sub>534</sub>
1761 535
1762 536
1763 537
        Happy to write the patch, if we come up with an approach here.
1763 538
1764 539
        --- END ISSUE ---
1765 <sub>540</sub>
       Below are some thinking trajectories, each from 11m reasoning model.
1766 541
1767 542
1768 543
        --- BEGIN FILE ---
        . . .
1769 545
        Answer 1:
1770 546
        '''python
        ### gabbi/runner.py
1771 547
1772 548
        <<<<< SEARCH
1773 549
        import argparse
        from importlib import import_module
    550
1774 551
        import sys
1775 552 import unittest
1776 553
1777 554 | import argparse
1778 555
        from importlib import import_module
1779 556 import os import sys
1780 558 import unittest
1781 559 >>>>> REPLACE
    560
```

```
'''python
1783 <sub>562</sub>
        ### gabbi/runner.py
1784 563
         1785 564
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
             False,
1786
1787 565
                         failfast=False):
             """Run the tests from the YAML in handle."""
    566
1788 <sub>567</sub>
             data = utils.load_yaml(handle)
1789 568
             if force_ssl:
1790 569
                 if 'defaults' in data:
                      data['defaults']['ssl'] = True
1791 570
1792 571 572
                  else:
                      data['defaults'] = {'ssl': True}
1793 573
1794 <sub>574</sub>
             loader = unittest.defaultTestLoader
             test_suite = suitemaker.test_suite_from_dict(
1795 575
                  loader, 'input', data, '.', host, port, None, None, prefix=
1796 <sup>576</sup>
1797 <sub>577</sub>
                      prefix,
                  handlers=handler_objects)
1798 <sub>578</sub>
1799 579
             result = ConciseTestRunner(
1800 580
                  verbosity=2, failfast=failfast).run(test_suite)
             return result.wasSuccessful()
1801 <sup>581</sup>
1802 <sup>582</sup>
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
    583
1803
             False,
1804 <sub>584</sub>
                         failfast=False, test_directory='.'):
             """Run the tests from the YAML in handle."""
1805 585
             data = utils.load_yaml(handle)
1806 <sup>586</sup>
1807 587
             if force ssl:
                 if 'defaults' in data:
1808 <sub>589</sub>
                      data['defaults']['ssl'] = True
1809 <sub>590</sub>
                  else:
1810 591
                      data['defaults'] = {'ssl': True}
1811 592
1812 593
             loader = unittest.defaultTestLoader
    594
             test_suite = suitemaker.test_suite_from_dict(
1813 <sub>595</sub>
                  loader, 'input', data, test_directory, host, port, None, None,
1814 596
                  prefix=prefix, handlers=handler_objects)
1815 597
1816 598
             result = ConciseTestRunner(
1817 599
                 verbosity=2, failfast=failfast).run(test_suite)
             return result.wasSuccessful()
    600
1818 601
         >>>>> REPLACE
1819 602
         '''python
1820 603
         ### gabbi/runner.py
1821 <sup>604</sup>
1822 605
         <<<<< SEARCH
             else:
1823 607
                  for input_file in input_files:
1824 <sub>608</sub>
                       with open(input_file, 'r') as fh:
                           success = run_suite(fh, handler_objects, host, port,
1825 609
                                                   prefix, force_ssl, failfast)
1826 610
1827 611
    612
             else:
1828 613
                  for input_file in input_files:
1829 614
                       test_directory = os.path.dirname(input_file) or '.'
                       with open(input_file, 'r') as fh:
1830 615
1831 616
                           success = run_suite(fh, handler_objects, host, port,
1832 617
                                                   prefix, force_ssl, failfast,
                                                   test_directory=test_directory)
    618
1833 619
        >>>>> REPLACE
1834 <sub>620</sub>
1835 621
    Answer 2:
```

```
'''python
1837 <sub>624</sub>
        ### gabbi/runner.py
1838 <sub>625</sub>
        <<<<< SEARCH
1839 626
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
            False,
1840
1841 627
                         failfast=False):
         _____
    628
1842 629
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1843
            False,
1844 630
                         failfast=False, test_directory='.'):
1845 631
        >>>>> REPLACE
         ...
1846 632
         '''python
1847 633
634
         ### gabbi/runner.py
1848 <sub>635</sub>
         test_suite = suitemaker.test_suite_from_dict(
1849 636
                  loader, 'input', data, '.', host, port, None, None, prefix=
1850 637
1851 638
                      prefix,
                  handlers=handler_objects)
1852 639
         _____
1853 <sub>640</sub>
             test_suite = suitemaker.test_suite_from_dict(
                 loader, 'input', data, test_directory, host, port, None, None,
1854 641
                      prefix=prefix,
1855
1856 642
                 handlers=handler_objects)
1857 643
644
        >>>>> REPLACE
1858 <sub>645</sub>
         '''python
1859 646
        ### gabbi/runner.py
        <><< SEARCH
1860 <sup>647</sup>
                           success = run_suite(fh, handler_objects, host, port,
1861 648
                                                   prefix, force_ssl, failfast)
1862 <sub>650</sub>
         ======
1863 651
                           test_directory = os.path.dirname(input_file) or '.'
                           success = run_suite(fh, handler_objects, host, port,
1864 652
                                                   prefix, force_ssl, failfast,
1865 653
                                                       test_directory=test_directory)
1866
        >>>>> REPLACE
1867 <sub>655</sub>
1868 <sub>656</sub>
1869 657
        Answer 3:
1870 <sup>658</sup>
         '''python
1871 659
        ### gabbi/runner.py
        <<<<< SEARCH
    660
1872 661
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1873
            False,
1874 662
                         failfast=False):
             """Run the tests from the YAML in handle."""
1875 663
1876 664
             data = utils.load_yaml(handle)
1877 665
1877 666
             if force_ssl:
                 if 'defaults' in data:
1878 <sub>667</sub>
                      data['defaults']['ssl'] = True
1879 668
                  else:
                      data['defaults'] = {'ssl': True}
1880 <sup>669</sup>
1881 670
             loader = unittest.defaultTestLoader
    671
1882 <sub>672</sub>
             test_suite = suitemaker.test_suite_from_dict(
1883 <sub>673</sub>
                  loader, 'input', data, '.', host, port, None, None, prefix=
                      prefix,
1884
1885 <sup>674</sup>
                  handlers=handler_objects)
1886 675
             result = ConciseTestRunner(
    676
1887 <sub>677</sub>
              verbosity=2, failfast=failfast).run(test_suite)
1888 <sub>678</sub>
             return result.wasSuccessful()
1889 679 ======
```

```
def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1891
             False,
1892 <sub>681</sub>
                          failfast=False, test_directory='.'):
1893 682
              """Run the tests from the YAML in handle."""
              data = utils.load_yaml(handle)
1894 <sup>683</sup>
1895 <sup>684</sup>
              if force_ssl:
                  if 'defaults' in data:
    685
1896 <sub>686</sub>
                       data['defaults']['ssl'] = True
1897 687
1898 688
                       data['defaults'] = {'ssl': True}
1899 689
1900 <sup>690</sup>
             loader = unittest.defaultTestLoader
             test_suite = suitemaker.test_suite_from_dict(
    691
1901 692
                  loader, 'input', data, test_directory, host, port, None, None,
1902 693
                  prefix=prefix, handlers=handler_objects)
1903 694
             result = ConciseTestRunner(
1904 <sup>695</sup>
1905 <sup>696</sup>
                  verbosity=2, failfast=failfast).run(test_suite)
    697
              return result.wasSuccessful()
1906 698
         >>>>>> REPLACE
1907 699
         . . .
         '''python
1908 700
1909 701
         ### gabbi/runner.py
1910 702
         <<<<< SEARCH
                 for input_file in input_files:
    703
1911 704
                       with open(input_file, 'r') as fh:
1912 705
                            success = run_suite(fh, handler_objects, host, port,
1913 706
                                                    prefix, force_ssl, failfast)
                       if not failure: # once failed, this is considered
1914 <sup>707</sup>
1915 <sub>708</sub>
                           immutable
                            failure = not success
1916 <sub>709</sub>
                       if failure and failfast:
1917 710
                           break
1918 711
         _____
1919 712
                  for input_file in input_files:
1920 713
                       test_directory = os.path.dirname(input_file)
                       with open(input_file, 'r') as fh:
    714
1921 715
                            success = run_suite(
1922 716
                                 fh, handler_objects, host, port, prefix, force_ssl,
                                 failfast, test_directory=test_directory)
1923 717
1924 718
                       if not failure: # once failed, this is considered
1925 719
                            immutable
                            failure = not success
1926 <sub>720</sub>
                       if failure and failfast:
1927 <sub>721</sub>
                            break
         >>>>> REPLACE
1928 722
1929 <sup>723</sup>
1930 <sup>724</sup>
         Answer 4:
1931 <sub>726</sub>
         '''python
1932 <sub>727</sub>
         ### gabbi/runner.py
1933 728
         <<<<< SEARCH
1934 729
         def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1935 730
             False,
                         failfast=False):
1936 <sub>731</sub>
         _____
1937 <sub>732</sub>
         def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1938
1939 733
                          failfast=False, test_directory='.'):
1940 <sup>734</sup>
         >>>>> REPLACE
    735
1941 <sub>736</sub>
         '''python
1942 <sub>737</sub>
         ### gabbi/runner.py
1943 738
         <<<<< SEARCH
            test_suite = suitemaker.test_suite_from_dict(
    739
```

```
1944
                  loader, 'input', data, '.', host, port, None, None, prefix=
1945
                      prefix,
1946 <sub>741</sub>
                  handlers=handler_objects)
1947 742
1948 743
             test_suite = suitemaker.test_suite_from_dict(
1949 <sup>744</sup>
                  loader, 'input', data, test_directory, host, port, None, None,
                  prefix=prefix, handlers=handler_objects)
    745
1950 <sub>746</sub>
        >>>>> REPLACE
1951 747
         '''python
1952 748
1953 749
        ### gabbi/runner.py
1954 750
        <<<<< SEARCH
    751
             for input_file in input_files:
1955 752
                  with open(input_file, 'r') as fh:
1956 753
                       success = run_suite(fh, handler_objects, host, port,
                                              prefix, force_ssl, failfast)
1957 754
1958 <sup>755</sup>
1959 756
             for input_file in input_files:
    757
                  # Determine test directory based on input file's location
1960 <sub>758</sub>
                  test_directory = os.path.dirname(os.path.abspath(input_file))
1961
                      or '.'
1962 759
                  with open(input_file, 'r') as fh:
1963 <sup>760</sup>
                       success = run_suite(fh, handler_objects, host, port,
1964 761
                                              prefix, force_ssl, failfast,
                                              test_directory=test_directory)
    762
1965 763
        >>>>> REPLACE
1966 764
1967 765
        Answer 5:
1968 <sup>766</sup>
1969 <sup>767</sup>
         '''python
        ### gabbi/runner.py
1970 <sub>769</sub>
        <<<<< SEARCH
1971 <sub>770</sub>
        import sys
        import unittest
1972 771
1973 772
        _____
1974 773
        import os
    774
        import sys
1975 <sub>775</sub>
        import unittest
1976 776
        >>>>> REPLACE
1977 777
         '''python
1978 <sup>778</sup>
1979 779
        ### gabbi/runner.py
        780
1980 781
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1981
             False,
1982 782
                         failfast=False):
1983 <sup>783</sup>
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1984 784
            False,
1985 785
                         failfast=False, test_directory='.'):
1986 786
        >>>>> REPLACE
1987 787
         '''python
1988 <sup>788</sup>
1989 <sup>789</sup>
         ### gabbi/runner.py
         <<<<< SEARCH
    790
1990 <sub>791</sub>
             test_suite = suitemaker.test_suite_from_dict(
                  loader, 'input', data, '.', host, port, None, None, prefix=
1991 792
                      prefix,
1992
1993 793
                  handlers=handler_objects)
1994 794
         ======
             test_suite = suitemaker.test_suite_from_dict(
    795
1995 796
                  loader, 'input', data, test_directory, host, port, None, None,
1996 797
                  prefix=prefix, handlers=handler_objects)
        >>>>> REPLACE
1997 798
        111
    799
```

```
'''python
1999 801
        ### gabbi/runner.py
2000 802
        <<<<< SEARCH
2001 803
                for input_file in input_files:
2002 804
                     with open(input_file, 'r') as fh:
2003 805
                         success = run_suite(fh, handler_objects, host, port,
                                               prefix, force_ssl, failfast)
    806
2004 807
2005 808
                for input_file in input_files:
2006 809
                     test_directory = os.path.dirname(input_file) or '.'
                     with open(input_file, 'r') as fh:
2007 810
                         success = run_suite(fh, handler_objects, host, port,
2008 811
                                               prefix, force_ssl, failfast,
2009 813
    812
                                               test_directory=test_directory)
2010 814
        >>>>> REPLACE
2011 815
2012 816
2013 817
        Answer 6:
    818
        '''python
2014 <sub>819</sub>
        ### gabbi/driver.py
2015 820
        2016 821
                file_suite = suitemaker.test_suite_from_dict(
2017 822
                     loader, test_base_name, suite_dict, path, host, port,
2018 823
                     fixture_module, intercept, prefix=prefix,
2019 824
                     test_loader_name=test_loader_name, handlers=handler_objects
2020 825
                     inner_fixtures=inner_fixtures)
2021 826
        ======
                test_file_dir = os.path.dirname(test_file)
2022 827
2023 828
                 file_suite = suitemaker.test_suite_from_dict(
                     loader, test_base_name, suite_dict, test_file_dir, host,
2024
2025 830
                     fixture_module, intercept, prefix=prefix,
2026 831
                     test_loader_name=test_loader_name, handlers=handler_objects
2027
2028 832
                     inner_fixtures=inner_fixtures)
    833
        >>>>> REPLACE
2029 <sub>834</sub>
2030 835
        '''python
2031 836
        ### gabbi/runner.py
2032 837
        2033 838
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
           False,
2034 <sub>839</sub>
                       failfast=False):
2035 <sub>840</sub>
        def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2036 841
          False,
2037
2038 842
                       failfast=False, test_directory='.'):
        >>>>> REPLACE
2039 844
        . . .
2040 845
        '''python
2041 846
        ### gabbi/runner.py
2042 847
        2043 848
            test_suite = suitemaker.test_suite_from_dict(
                 loader, 'input', data, '.', host, port, None, None, prefix=
2044
                    prefix,
2045 850
                handlers=handler_objects)
2046 851
2047 852
            test_suite = suitemaker.test_suite_from_dict(
2048 853
                loader, 'input', data, test_directory, host, port, None, None,
                prefix=prefix, handlers=handler_objects)
    854
2049 855
        >>>>> REPLACE
2050 856
        . . .
        '''python
2051 857
    858 | ### gabbi/runner.py
```

```
<<<<< SEARCH
2053 860
                          success = run_suite(fh, handler_objects, host, port,
2054 861
                                                prefix, force_ssl, failfast)
2055 862
2056 863
                          test_directory = os.path.dirname(input_file)
2057 864
                          success = run_suite(fh, handler_objects, host, port,
                                                prefix, force_ssl, failfast,
2058
                                                    test_directory)
2059 866
        >>>>> REPLACE
2060 867
2061 868
2062 869
        Answer 7:
        '''python
    870
2063 871
        ### gabbi/runner.py
2064 872
        <<<<< SEARCH
2065 873
        import argparse
        from importlib import import_module
2066 874
2067 875
        import sys
    876
        import unittest
2068 877
        ======
2069 878
        import argparse
2070 879 from importlib import import_module
        import os
2071 880
2072 <sup>881</sup>
        import sys
        import unittest
    882
2073 883
        >>>>> REPLACE
2074 <sub>884</sub>
        ```python
2075 885
 ### gabbi/runner.py
2076 886
2077 ⁸⁸⁷
 <<<<< SEARCH
2078 888
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
 False,
2079 889
 failfast=False):
 """Run the tests from the YAML in handle."""
2080 890
 data = utils.load_yaml(handle)
2081 891
2082 892
 if force_ssl:
 if 'defaults' in data:
 893
2083 894
 data['defaults']['ssl'] = True
2084 895
 else:
 data['defaults'] = {'ssl': True}
2085 896
2086 897
2087 898
 loader = unittest.defaultTestLoader
 test_suite = suitemaker.test_suite_from_dict(
 899
2088 900
 loader, 'input', data, '.', host, port, None, None, prefix=
2089
 prefix,
2090 901
 handlers=handler_objects)
2091 902
2092 903
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
 False,
2093 904
 failfast=False, test_directory='.'):
 """Run the tests from the YAML in handle."""
2094 905
 data = utils.load_yaml(handle)
2095 906
2096 907
 if force_ssl:
2097 908
 if 'defaults' in data:
 data['defaults']['ssl'] = True
 909
2098 910
 else:
2099 911
 data['defaults'] = {'ssl': True}
2100 912
 loader = unittest.defaultTestLoader
2101 913
2102 914
 test_suite = suitemaker.test_suite_from_dict(
 loader, 'input', data, test_directory, host, port, None, None,
 915
2103 916
 prefix=prefix, handlers=handler_objects)
2104 917
 >>>>> REPLACE
 . . .
2105 918
 919 '''python
```

```
| ### gabbi/runner.py
2107 921
 <<<<< SEARCH
2108 922
 with open(input_file, 'r') as fh:
2109 923
 success = run_suite(fh, handler_objects, host, port,
 prefix, force_ssl, failfast)
2110 924
 if not failure: # once failed, this is considered
2111 925
 immutable
2112 926
 failure = not success
2113 927
 if failure and failfast:
2114 928
 break
2115 929
 test_dir = os.path.dirname(os.path.abspath(input_file))
2116 930
2117 931 2117 932
 with open(input_file, 'r') as fh:
 success = run_suite(fh, handler_objects, host, port,
2118 933
 prefix, force_ssl, failfast,
2119
 test_directory=test_dir)
 if not failure: # once failed, this is considered
2120 934
2121 935
 immutable
 failure = not success
2122 936
 if failure and failfast:
2123 937
 break
2124 938
 >>>>> REPLACE
2125 939
2126 940
 941
 Answer 8:
2127 942
 '''python
2128 ₉₄₃
 ### gabbi/runner.py
 <<<<< SEARCH
2129 944
 data = utils.load_yaml(handle)
2130 945
2131 946 947
 if force_ssl:
 if 'defaults' in data:
2132 ₉₄₈
 data['defaults']['ssl'] = True
2133 949
 else:
2134 950
 data['defaults'] = {'ssl': True}
2135 951
2136 952
 loader = unittest.defaultTestLoader
 953
 test_suite = suitemaker.test_suite_from_dict(
2137 954
 loader, 'input', data, '.', host, port, None, None, prefix=
2138
 prefix,
2139 955
 handlers=handler_objects)

2140 956
2141 957
 data = utils.load_yaml(handle)
 if force_ssl:
 958
2142 959
 if 'defaults' in data:
2143 ₉₆₀
 data['defaults']['ssl'] = True
2144 961
 else:
 data['defaults'] = {'ssl': True}
2145 962
2146 963
 # Determine test directory based on input handle
2147 965
 if hasattr(handle, 'name'):
2148 966
 test_directory = os.path.dirname(os.path.abspath(handle.name))
 else:
2149 967
 test_directory = '.'
2150 968
2151 969
 loader = unittest.defaultTestLoader
 970
2152 971
 test_suite = suitemaker.test_suite_from_dict(
2153 972
 loader, 'input', data, test_directory, host, port, None, None,
 prefix=prefix, handlers=handler_objects)
2154 973
2155 974
 >>>>> REPLACE
2156 975
 976
2157 ₉₇₇
 Answer 9:
2158 978
 '''python
2159 979 | ### gabbi/driver.py
 980 <<<<< SEARCH
```

```
for test_file in glob.iglob('%s/*.yaml' % path):
2161 982
 if '_' in os.path.basename(test_file):
2162 983
 warnings.warn(exception.GabbiSyntaxWarning(
2163 984
 "'_' in test filename %s. This can break suite grouping
2164
2165 985
 % test_file))
 if intercept:
 986
2166 987
 host = str(uuid.uuid4())
2167 988
 suite_dict = utils.load_yaml(yaml_file=test_file)
2168 989
 test_base_name = os.path.splitext(os.path.basename(test_file))
 [0]
2169
2170 990
 if all_test_base_name:
 test_base_name = '%s_%s' % (all_test_base_name,
 991
2171
 test_base_name)
2172 992
2173 993
 if require_ssl:
 if 'defaults' in suite_dict:
2174 994
2175 995
 suite_dict['defaults']['ssl'] = True
 996
 else:
2176 997
 suite_dict['defaults'] = {'ssl': True}
2177 998
2178 999
 file_suite = suitemaker.test_suite_from_dict(
2179¹⁰⁰⁰
 loader, test_base_name, suite_dict, path, host, port,
2180¹⁰⁰¹
 fixture_module, intercept, prefix=prefix,
 test_loader_name=test_loader_name, handlers=handler_objects
 1002
2181
2182₁₀₀₃
 inner_fixtures=inner_fixtures)
21831004
 for test_file in glob.iglob('%s/*.yaml' % path):
21841005
 if '_' in os.path.basename(test_file):
2185¹⁰⁰⁶
 1007
 warnings.warn(exception.GabbiSyntaxWarning(
2186₁₀₀₈
 "'_' in test filename %s. This can break suite grouping
2187
21881009
 % test_file))
 if intercept:
2189¹⁰¹⁰
2190¹⁰¹¹
 host = str(uuid.uuid4())
 suite_dict = utils.load_yaml(yaml_file=test_file)
 1012
2191₁₀₁₃
 test_base_name = os.path.splitext(os.path.basename(test_file))
2192
 [0]
21931014
 if all_test_base_name:
2194¹⁰¹⁵
 test_base_name = '%s_%s' % (all_test_base_name,
 test_base_name)
2195 1016
2196₁₀₁₇
 if require_ssl:
2197₁₀₁₈
 if 'defaults' in suite_dict:
 suite_dict['defaults']['ssl'] = True
21981019
2199¹⁰²⁰
 else:
2200¹⁰²¹
 suite_dict['defaults'] = {'ssl': True}
 1022
2201₁₀₂₃
 # Use absolute path of the YAML file's directory
22021024
 test_file_dir = os.path.dirname(os.path.abspath(test_file))
 file_suite = suitemaker.test_suite_from_dict(
22031025
2204¹⁰²⁶
 loader, test_base_name, suite_dict, test_file_dir, host,
 port,
2205 1027
 fixture_module, intercept, prefix=prefix,
2206₁₀₂₈
 test_loader_name=test_loader_name, handlers=handler_objects
2207
 inner_fixtures=inner_fixtures)
22081029
2209¹⁰³⁰
 >>>>> REPLACE
2210¹⁰³¹
 1032
2211₁₀₃₃
 Answer 10:
2212₁₀₃₄
 '''python
 ### gabbi/runner.py
22131035
 1036 <
```

```
import argparse
2215₁₀₃₈
 from importlib import import_module
2216₁₀₃₉
 import sys
22171040
 import unittest
2218¹⁰⁴¹
2219¹⁰⁴²
 import argparse
 1043
 from importlib import import_module
2220₁₀₄₄
 import os
2221₁₀₄₅
 import sys
22221046
 import unittest
 >>>>> REPLACE
22231047
 ...
2224¹⁰⁴⁸
 '''python
2224
1049
2225
1050
 ### gabbi/runner.py
2226₁₀₅₁
 <<<<< SEARCH
22271052
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
 False,
2228
2229¹⁰⁵³
 failfast=False):
2229
1054
2230
1055
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2231
 False,
22321056
 failfast=False, test_directory='.'):
22331057
 >>>>> REPLACE
2234¹⁰⁵⁸
 '''python
 1059
2235₁₀₆₀
 ### gabbi/runner.py
2236₁₀₆₁
 22371062
 test_suite = suitemaker.test_suite_from_dict(
 loader, 'input', data, '.', host, port, None, None, prefix=
22381063
 prefix,
2239 1064
 force_ssl=force_ssl, failfast=failfast,
2240₁₀₆₅
 handlers=handler_objects)
2241₁₀₆₆

22421067
 test_suite = suitemaker.test_suite_from_dict(
2243¹⁰⁶⁸
 loader, 'input', data, test_directory, host, port, None, None,
2244¹⁰⁶⁹
 prefix=prefix, force_ssl=force_ssl, failfast=failfast,
2245₁₀₇₁
 1070
 handlers=handler_objects)
 >>>>> REPLACE
2246₁₀₇₂
 . . .
 '''python
22471073
2248¹⁰⁷⁴
 ### gabbi/runner.py
2249¹⁰⁷⁵
 <<<<< SEARCH
 1076
 success = run_suite(fh, handler_objects, host, port,
2250₁₀₇₇
 prefix, force_ssl, failfast)
\bm{2251}_{1078}

 test_dir = os.path.dirname(input_file)
22521079
 success = run_suite(fh, handler_objects, host, port,
22531080
2254¹⁰⁸¹
 prefix, force_ssl, failfast,
 test_directory=test_dir)
2255₁₀₈₃
 1082
 >>>>> REPLACE
2256₁₀₈₄
22571085
 '''python
2258¹⁰⁸⁶
 ### gabbi/case.py
2259¹⁰⁸⁷
 <<<<< SEARCH
2260 1088
 path = os.path.join(self.test_directory, os.path.basename(filename)
2261₁₀₈₉
 path = os.path.join(self.test_directory, filename)
22621090
22631091
 >>>>> REPLACE
2264¹⁰⁹²
 1093
2265₁₀₉₄
 Answer 11:
2266₁₀₉₅
 '''python
22671096 | ### gabbi/runner.py
 1097 <
```

```
def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2269
 False,
2270₁₀₉₉
 failfast=False):
22711100
 """Run the tests from the YAML in handle."""
2272¹¹⁰¹
 data = utils.load_yaml(handle)
 if force_ssl:
2273¹¹⁰²
 if 'defaults' in data:
 1103
2274₁₁₀₄
 data['defaults']['ssl'] = True
2275₁₁₀₅
 else:
22761106
 data['defaults'] = {'ssl': True}
22771107
2278¹¹⁰⁸
 loader = unittest.defaultTestLoader
 1109
 test_suite = suitemaker.test_suite_from_dict(
2279₁₁₁₀
 loader, 'input', data, '.', host, port, None, None, prefix=
2280
 prefix,
22811111
 handlers=handler_objects)
22821112
2283¹¹¹³
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
 False,
2284₁₁₁₄
 failfast=False, test_directory='.'):
2285₁₁₁₅
 """Run the tests from the YAML in handle."""
22861116
 data = utils.load_yaml(handle)
22871117
 if force_ssl:
2288¹¹¹⁸
 if 'defaults' in data:
 data['defaults']['ssl'] = True
2289₁₁₂₀
 1119
 else:
2290₁₁₂₁
 data['defaults'] = {'ssl': True}
22911122
 loader = unittest.defaultTestLoader
22921123
2293¹¹²⁴
 test_suite = suitemaker.test_suite_from_dict(
 loader, 'input', data, test_directory, host, port,
 1125
2294₁₁₂₆
 None, None, prefix=prefix, handlers=handler_objects)
2295₁₁₂₇
 >>>>> REPLACE
22961128
 '''python
2297¹¹²⁹
2298¹¹³⁰
 ### gabbi/runner.py
2299
1131
2299
1132
 <<<<< SEARCH
 success = run_suite(fh, handler_objects, host, port,
2300₁₁₃₃
 prefix, force_ssl, failfast)
23011134
 test_dir = os.path.dirname(input_file)
23021135
2303¹¹³⁶
 success = run_suite(fh, handler_objects, host, port,
 1137
 prefix, force_ssl, failfast,
2304₁₁₃₈
 test_directory=test_dir)
2305₁₁₃₉
 >>>>> REPLACE
23061140
23071141
2308¹¹⁴² Answer 12:
 '''python
 1143
2309₁₁₄₄ | ### gabbi/runner.py
2310₁₁₄₅
 <<<<< SEARCH
23111146
 import argparse
2312¹¹⁴⁷
 from importlib import import_module
2313¹¹⁴⁸
 import sys
 import unittest
 1149
2314₁₁₅₀

2315₁₁₅₁ import argparse
23161152 from importlib import import_module
2317¹¹⁵³ import os
2318¹¹⁵⁴ import sys
 1155
 import unittest
2319₁₁₅₆
 >>>>> REPLACE
23201157
23211158 '`'python
 ### gabbi/runner.py
```

```
<<<<< SEARCH
2323₁₁₆₁
 else:
2324₁₁₆₂
 for input_file in input_files:
23251163
 with open(input_file, 'r') as fh:
 success = run_suite(fh, handler_objects, host, port,
23261164
2327¹¹⁶⁵
 prefix, force_ssl, failfast)
 1166
 ======
2328₁₁₆₇
 else:
2329₁₁₆₈
 for input_file in input_files:
23301169
 test_dir = os.path.dirname(os.path.abspath(input_file))
 with open(input_file, 'r') as fh:
23311170
2332¹¹⁷¹
 success = run_suite(fh, handler_objects, host, port,
 prefix, force_ssl, failfast,
 1172
2333₁₁₇₃
 test_directory=test_dir)
2334₁₁₇₄
 >>>>> REPLACE
23351175
 '''python
2336¹¹⁷⁶
2337¹¹⁷⁷
 ### gabbi/runner.py
2337
1178
2338
1179
 <<<<< SEARCH
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2339
 False,
23401180
 failfast=False):
2341¹¹⁸¹
 """Run the tests from the YAML in handle."""
2342¹¹⁸²
 data = utils.load_yaml(handle)
 1183
 if force_ssl:
2343₁₁₈₄
 if 'defaults' in data:
2344₁₁₈₅
 data['defaults']['ssl'] = True
23451186
 else:
 data['defaults'] = {'ssl': True}
23461187
2347¹¹⁸⁸
 loader = unittest.defaultTestLoader
 1189
2348₁₁₉₀
 test_suite = suitemaker.test_suite_from_dict(
2349₁₁₉₁
 loader, 'input', data, '.', host, port, None, None, prefix=
2350
 prefix,
 handlers=handler_objects)
2351¹¹⁹²
2352¹¹⁹³
2353 1194
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
 False,
2354₁₁₉₅
 failfast=False, test_directory='.'):
 """Run the tests from the YAML in handle."""
23551196
2356¹¹⁹⁷
 data = utils.load_yaml(handle)
2357¹¹⁹⁸
 if force_ssl:
 if 'defaults' in data:
2358₁₂₀₀
 1199
 data['defaults']['ssl'] = True
2359₁₂₀₁
 else:
 data['defaults'] = {'ssl': True}
23601202
2361 1203
2362¹²⁰⁴
 loader = unittest.defaultTestLoader
 test_suite = suitemaker.test_suite_from_dict(
 1205
2363₁₂₀₆
 loader, 'input', data, test_directory, host, port, None, None,
23641207
 prefix=prefix, handlers=handler_objects)
 >>>>> REPLACE
23651208
2366¹²⁰⁹
2367¹²¹⁰
 Answer 13:
 1211
2368₁₂₁₂
 '''python
2369₁₂₁₃ | ### gabbi/runner.py
23701214 <<<<<< SEARCH
2371 1215
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2372 1216
 False,
 failfast=False):
2373
 """Run the tests from the YAML in handle."""
2374₁₂₁₈
 data = utils.load_yaml(handle)
23751219
 if force_ssl:
 if 'defaults' in data:
 1220
```

```
2376
 data['defaults']['ssl'] = True
23771222
 else:
2378₁₂₂₃
 data['defaults'] = {'ssl': True}
23791224
2380¹²²⁵
 loader = unittest.defaultTestLoader
2381 1226
 test_suite = suitemaker.test_suite_from_dict(
2382 1227
 loader, 'input', data, '.', host, port, None, None, prefix=
 prefix,
2383₁₂₂₈
 handlers=handler_objects)
23841229

 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
23851230
2386 1231
 False,
 failfast=False, test_directory='.'):
2387₁₂₃₂
 """Run the tests from the YAML in handle."""
2388₁₂₃₃
 data = utils.load_yaml(handle)
23891234
 if force_ssl:
 if 'defaults' in data:
2390¹²³⁵
2391 1236
 data['defaults']['ssl'] = True
2391
2392
1237
 else:
 data['defaults'] = {'ssl': True}
2393₁₂₃₉
23941240
 loader = unittest.defaultTestLoader
2395¹²⁴¹
 test_suite = suitemaker.test_suite_from_dict(
2396¹²⁴²
 loader, 'input', data, test_directory, host, port, None, None,
2390
1243
2397
1244
 prefix=prefix, handlers=handler_objects)
 >>>>> REPLACE
2398₁₂₄₅
 . . .
 '''python
23991246
 ### gabbi/runner.py
2400¹²⁴⁷
2401 1248
 1249
 for input_file in input_files:
2402₁₂₅₀
 with open(input_file, 'r') as fh:
2403₁₂₅₁
 success = run_suite(fh, handler_objects, host, port,
24041252
 prefix, force_ssl, failfast)
2405¹²⁵³
2406¹²⁵⁴
 for input_file in input_files:
2407
1255
2407
1256
 test_dir = os.path.dirname(os.path.abspath(input_file))
 with open(input_file, 'r') as fh:
2408₁₂₅₇
 success = run_suite(fh, handler_objects, host, port,
24091258
 prefix, force_ssl, failfast,
2410¹²⁵⁹
 test_directory=test_dir)
2411 1260
 >>>>> REPLACE
2412₁₂₆₂
2413₁₂₆₃
 Answer 14:
 '''python
24141264
 ### gabbi/runner.py
2415¹²⁶⁵
2416¹²⁶⁶
 <<<<< SEARCH
2410
1267
2417
1268
 for input_file in input_files:
 with open(input_file, 'r') as fh:
24181269
 success = run_suite(fh, handler_objects, host, port,
24191270
 prefix, force_ssl, failfast)
2420¹²⁷¹
2421 1272
 for input_file in input_files:
 with open(input_file, 'r') as fh:
 1273
2422₁₂₇₄
 success = run_suite(input_file, fh, handler_objects,
2423
 host, port,
 prefix, force_ssl, failfast)
24241275
 >>>>> REPLACE
2425¹²⁷⁶
2426¹²⁷⁷
 '''python
 1278
2427₁₂₇₉
 ### gabbi/runner.py
2428₁₂₈₀
 <<<<< SEARCH
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
24291281
 False,
```

```
2430 1282
 failfast=False):
2431₁₂₈₃
 """Run the tests from the YAML in handle."""
2432₁₂₈₄
 data = utils.load_yaml(handle)
24331285
 if force ssl:
2434¹²⁸⁶
 if 'defaults' in data:
2435¹²⁸⁷
 data['defaults']['ssl'] = True
 else:
 1288
2436₁₂₈₉
 data['defaults'] = {'ssl': True}
2437₁₂₉₀
24381291
 loader = unittest.defaultTestLoader
 test_directory = '.'
2439¹²⁹²
2440¹²⁹³
 test_suite = suitemaker.test_suite_from_dict(
 loader, 'input', data, test_directory, host, port, None, None,
2441 1294
 prefix=prefix,
24421295
 handlers=handler_objects)
24431296
 ======
 def run_suite(input_file, handle, handler_objects, host, port, prefix,
2444¹²⁹⁷
 force_ssl=False,
2445 1298
 failfast=False):
2446₁₂₉₉
 """Run the tests from the YAML in handle."""
2447₁₃₀₀
 data = utils.load_yaml(handle)
24481301
 if force_ssl:
2449¹³⁰²
 if 'defaults' in data:
2450¹³⁰³
 data['defaults']['ssl'] = True
 else:
 1304
2451₁₃₀₅
 data['defaults'] = {'ssl': True}
2452₁₃₀₆
24531307
 loader = unittest.defaultTestLoader
 test_directory = os.path.dirname(input_file) if input_file else '.'
2454¹³⁰⁸
2455¹³⁰⁹
 test_suite = suitemaker.test_suite_from_dict(
 loader, 'input', data, test_directory, host, port, None, None,
 1310
2456
 prefix=prefix,
2457₁₃₁₁
 handlers=handler_objects)
 >>>>> REPLACE
24581312
2459¹³¹³
2460¹³¹⁴
 '''python
 ### gabbi/runner.py
2461₁₃₁₆
 1315
 <<<<< SEARCH
2462₁₃₁₇
 if not input_files:
 success = run_suite(sys.stdin, handler_objects, host, port,
24631318
2464¹³¹⁹
 prefix, force_ssl, failfast)
2465¹³²⁰

 if not input_files:
2466₁₃₂₂
 1321
 success = run_suite(None, sys.stdin, handler_objects, host,
2467
 port,
 prefix, force_ssl, failfast)
24681323
 >>>>>> REPLACE
2469¹³²⁴
2470¹³²⁵
 . . .
 . . .
2470
1326
2471
1327
 --- END FILE ---
2472₁₃₂₈
 Please first summary and analyze the key differences between the
24731329
 trajectories, and then generate *SEARCH/REPLACE* edits to fix the
2474
 issue.
2475
2476₁₃₃₁
 Every *SEARCH/REPLACE* edit must use this format:
2477₁₃₃₂ 1. The file path
24781333 2. The start of search block: <<<< SEARCH
24791334 | 3. A contiguous chunk of lines to search for in the existing source
2480 1335
 code
 4. The dividing line: ======
2481₁₃₃₆
 5. The lines to replace into the source code
2482₁₃₃₇
 6. The end of the replace block: >>>>> REPLACE
24831338
 Here is an example:
```

```
2485₁₃₄₁
 '''python
2486₁₃₄₂
 ### mathweb/flask/app.py
24871343
 2488¹³⁴⁴
 from flask import Flask
2489¹³⁴⁵

 1346
 import math
2490₁₃₄₇
 from flask import Flask
2491₁₃₄₈
 >>>>> REPLACE
24921349
24931350
2494¹³⁵¹
 Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
 If you would like to add the line '
 print(x)', you must
2495
 fully write that out, with all those spaces before the code!
2496₁₃₅₂
 Wrap the *SEARCH/REPLACE* edit in blocks '''python...'''.
24971353
 The summary of the key differences between the trajectories should be
 in the thinking part.
2499¹³⁵⁴
2499
1355
2500
1356
 Response:
2501₁₃₅₇
25021358
 ⊲think⊳
 Okay, let's tackle this problem. The user is having an issue where when
2503¹³⁵⁹
 they specify a data file in their YAML test using the <@ syntax,
2504
 the path isn't resolved relative to the YAML file's location.
2505
 Instead, it's using the current working directory. They want the
2506
 path to be relative to the YAML file so they can run tests from any
2507
 directory.
25081360
2509¹³⁶¹
 First, I need to understand how the current code is handling the data
 file paths. The user mentioned that in _load_data_file, the code
2510
 uses os.path.join(self.test_directory, os.path.basename(filename)).
2511
 This takes the test_directory (which defaults to '.'), discards any
2512
 directories in the filename, and appends the basename. That's why if
 the YAML is in /my/tests/test.yaml and the data is <@data.txt, it
2513
 looks for ./data.txt instead of /my/tests/data.txt.
2514
2515₁₃₆₃
 The key here is to make sure that the test_directory is set to the
2516
 directory where the YAML file is located. Then, when joining with
 the filename (without basename), it would correctly resolve the
2517
 relative path.
2518
2519¹³⁶⁴
 Looking at the provided code snippets, the main changes are in runner.
2520
 py where the test_directory is determined. The user's proposed
2521
 solutions involve modifying the run_suite function to accept a
 test_directory parameter, which is derived from the input file's
2522
 directory.
2523
2524¹³⁶⁶
 In the original code, when running a suite from an input file, the
2525
 test_directory is hardcoded to '.'. The various answers modify this
2526
 by passing the directory of the input file as test_directory. For
 example, in Answer 10, they add the test_directory parameter to
2527
 run_suite and set it to os.path.dirname(input_file). Then, in case.
2528
 py, they change the path joining to use the filename directly
2529
 instead of basename.
2530₁₃₆₈
2531₁₃₆₉
 Wait, the critical part is in case.py's _load_data_file method. The
 original code uses os.path.basename(filename), which strips any
2532
 directories from the filename. So if the YAML has <@test_data/file.
2533
 txt, the basename is file.txt, and it's joined with test_directory.
2534
 But if test_directory is the YAML's directory, then it should be
2535
 YAML_DIR/test_data/file.txt. However, the current code would do
2536
 test_directory + file.txt, which would be YAML_DIR/file.txt. That's
2537
 incorrect. So the problem is that the code is taking the basename,
 thus ignoring any subdirectories in the filename.
```

**2539**<sub>1371</sub> Ah, so the fix here is to not take the basename. Instead, the filename 2540 after <@ should be treated as a relative path from the YAML's 2541 directory. So in case.py's \_load\_data\_file, instead of using os.path .basename(filename), we should use the filename as-is. Then, os.path 2542 .join(self.test\_directory, filename) would correctly resolve the 2543 path relative to the YAML's directory. **2544**<sub>1372</sub>  $\bm{2545}_{1373}$ But wait, the filename could be an absolute path. For example, if the YAML has <@/absolute/path/file.txt, then os.path.join would just use the absolute path. However, the documentation says that the 2547 filename is relative to the YAML file. So absolute paths might not be intended. But the code should handle that. 2549 **2550**<sub>1375</sub> So combining two changes: in runner.py, set test\_directory to the YAML 2551 file's directory, and in case.py, stop using basename and use the full filename. 2552 **2553**<sup>1376</sup> **2554**<sup>1377</sup> Looking at Answer 10, there's a change in case.py from os.path.basename (filename) to filename. That's exactly the fix needed. Then, in 2555 runner.py, when running the suite, pass the test\_directory as the 2556 directory of the input file. **2557**<sup>1378</sup> **2558**<sup>1379</sup> So the key files to modify are: 1380 **2559**<sub>1381</sub> 1. gabbi/runner.py: When calling run\_suite, pass test\_directory as the 2560 directory of the input file. This involves modifying the run\_suite 2561 function to accept test\_directory as a parameter and then passing it when the input file is processed. 2562 **2563**<sup>1382</sup> 1383 2. gabbi/case.py: Change the path construction from using os.path. 2564 basename (filename) to using the filename directly. This allows 2565 subdirectories in the filename to be included in the path. 25661384 **2567**<sup>1385</sup> Looking through the provided code snippets, the one that does both is Answer 10. Let me check: 2568 **2569**<sub>1387</sub> In Answer 10, the runner.py changes add the test\_directory parameter 2570 and set it to os.path.dirname(input\_file). Then, in case.py, the path is changed to os.path.join(self.test\_directory, filename). That 2571 would correctly resolve the path relative to the YAML's directory, 2572 including any subdirectories in the filename. 2573 1388 **2574**<sub>1389</sub> But wait, the original code in case.py uses os.path.basename(filename). 2575 So changing that to filename means that if the input is 'test\_data/ file.txt', it's joined as test\_directory/test\_data/file.txt. If 2576 test\_directory is the YAML's directory, that's correct. 2577 **2578**<sup>1390</sup> **2579**<sub>1392</sub> ----- Omit some reasoning content -----**2580**<sub>1393</sub> But perhaps better to use os.path.abspath and then dirname to handle relative paths correctly. But maybe the existing code already works. 2581 For example, if the input\_file is 'tests/test.yaml', then os.path. 2582 dirname would return 'tests', which when joined with os.getcwd() 2583 would be correct. Wait, no. os.path.dirname just gives the directory 2584 part of the path. So if input\_file is a relative path, like 'tests/ 2585 test.yaml', then test\_directory would be 'tests', and when os.path. join is called, it's relative to the current working directory. But 2586 the code in suitemaker.test\_suite\_from\_dict might expect an absolute 2587 path. However, the original code uses '.' as the test\_directory, 2588 which is relative. So perhaps using the directory of the input file 2589 as test\_directory is sufficient. **2590**<sub>1394</sub> Another thing to consider: when the YAML is loaded from stdin (like via **2591**1395

redirect), the test\_directory remains '.'. That's acceptable, as

```
2592
 the documentation mentions that when using <, the filename isn't
2593
 seen by Gabbi, so it's not possible to resolve the path relative to
2594
 the YAML's original location. So in that case, the test_directory
2595
 defaults to '.'.
2596¹³⁹⁶
2597¹³⁹⁷
 Now, in case.py:
 1398
2598₁₃₉₉
 Original line:
2599_{1400}
26001401
 path = os.path.join(self.test_directory, os.path.basename(filename))
2601 1402
2602¹⁴⁰³
 Modified line:
 1404
2603₁₄₀₅
 path = os.path.join(self.test_directory, filename)
2604₁₄₀₆
26051407
 So the SEARCH/REPLACE block for case.py:
2606¹⁴⁰⁸
2607¹⁴⁰⁹
 ### gabbi/case.py
 1410
 <<<<< SEARCH
2608 1411
 path = os.path.join(self.test_directory, os.path.basename(filename)
2609
26101412
 path = os.path.join(self.test_directory, filename)
2611¹⁴¹³
2612¹⁴¹⁴
 >>>>> REPLACE
 1415
2613₁₄₁₆
 Putting all these together, the final edits are:
2614₁₄₁₇
26151418
 1. Add import os in runner.py.
2616¹⁴¹⁹
2617¹⁴²⁰
 2. Modify run_suite to accept test_directory and pass it to
 test_suite_from_dict.
2618₁₄₂₁
2619₁₄₂₂
 3. In the loop over input_files, compute test_directory as the
 directory of input_file and pass it to run_suite.
2620
2621 1423
2622¹⁴²⁴
 4. Modify case.py's path construction.
 1425
2623₁₄₂₆
 So the final answer would include these four SEARCH/REPLACE blocks.
2624₁₄₂₇
 ⊲/think⊳
26251428
 '''python
26261429
2627¹⁴³⁰ | ### gabbi/runner.py
 <<<<< SEARCH
 1431
2628₁₄₃₂
 import argparse
2629₁₄₃₃
 from importlib import import_module
26301434
 import sys
2631 1435 import unittest
2632¹⁴³⁶

 1437 import argparse
2633₁₄₃₈ from importlib import import_module
2634₁₄₃₉
 import os
26351440
 import sys
26361441
 import unittest
2637¹⁴⁴²
 >>>>> REPLACE
 1443
2638₁₄₄₄
2639₁₄₄₅
 '''python
26401446 | ### gabbi/runner.py
 <<<<< SEARCH
2641 1447
2642¹⁴⁴⁸
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
 False,
2643₁₄₄₉
 failfast=False):
2644₁₄₅₀
 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
26451451
 False,
```

```
2646₁₄₅₂
 failfast=False, test_directory='.'):
2647₁₄₅₃
 >>>>> REPLACE
2648₁₄₅₄
26491455
2650¹⁴⁵⁶
 '''python
2651 1457 1458
 ### gabbi/runner.py
 <<<<< SEARCH
2652₁₄₅₉
 test_suite = suitemaker.test_suite_from_dict(
 loader, 'input', data, '.', host, port, None, None, prefix=
26531460
2654
 prefix,
2655¹⁴⁶¹
 handlers=handler_objects)
2656¹⁴⁶²

 1463
 test_suite = suitemaker.test_suite_from_dict(
2657₁₄₆₄
 loader, 'input', data, test_directory, host, port, None, None,
2658₁₄₆₅
 prefix=prefix, handlers=handler_objects)
 >>>>> REPLACE
26591466
2660¹⁴⁶⁷
2661 1468 1469
 '''python
2662₁₄₇₀
 ### gabbi/runner.py
26631471
 26641472
 else:
2665¹⁴⁷³
 for input_file in input_files:
2666¹⁴⁷⁴
 with open(input_file, 'r') as fh:
 success = run_suite(fh, handler_objects, host, port,
2667
 prefix, force_ssl, failfast)
2668₁₄₇₇
26691478
 else:
2670¹⁴⁷⁹
 for input_file in input_files:
2671 1480 1481
 test_directory = os.path.dirname(input_file) or '.'
 with open(input_file, 'r') as fh:
2672₁₄₈₂
 success = run_suite(fh, handler_objects, host, port,
2673₁₄₈₃
 prefix, force_ssl, failfast,
 test_directory=test_directory)
2674
2675¹⁴⁸⁴
 >>>>> REPLACE
2676¹⁴⁸⁵
 1486
2677₁₄₈₇
 '''python
2678₁₄₈₈
 ### gabbi/case.py
 <<<<< SEARCH
26791489
2680¹⁴⁹⁰
 path = os.path.join(self.test_directory, os.path.basename(filename)
2681 1491
2682₁₄₉₂
 path = os.path.join(self.test_directory, filename)
2683₁₄₉₃
 >>>>> REPLACE
26841494
2685¹⁴⁹⁵
2686 1496 1497
 These changes:
 1. Add 'os' import needed for path handling
2687₁₄₉₈
 2. Modify run_suite to accept a test_directory parameter
26881499
 3. Use the input file's directory as the test_directory when processing
2689
 files
 4. Update data file path resolution to preserve directory structure
2690¹⁵⁰⁰
2691 ₁₅₀₁
 from YAML relative paths
 5. Remove basename handling that was stripping directory information
2692
 from data file paths
 The net effect is that data files specified with '<@filename' in YAML
26941503
 tests will now be resolved relative to their YAML file's directory,
2695
 supporting both flat and nested directory structures for test data.
2696
 When tests are run from stdin (using '<), the default directory
2697
 remains the current working directory as before.
2698
```

Listing 12: Example of emergent parallel scaling (cdent\_gabbi-191 in SWE-bench-extra).

2699

## H USE OF LARGE LANGUAGE MODELS

The initial draft of this paper was written entirely by the authors. A large language model (gpt-5) was used only to aid with polishing the language (e.g., grammar and clarity). All conceptual contributions, experimental designs, analyses, and conclusions are the work of the authors.