Step Rejection Fine-Tuning: A Practical Distillation Recipe

Anonymous Authors[1]

Abstract

Rejection sampling Fine-Tuning (RFT) is a standard method for training LLM agents, where unsuccessful trajectories are discarded from the training set. In the context of `SWE-bench` tasks, this corresponds to filtering out runs where the submitted patch does not pass the tests. However, this approach discards unresolved trajectories, even though they form a large portion of all trajectories for hard tasks and even then may be partially correct. In this work, we propose Step Rejection Fine-Tuning (SRFT)—a practical way to leverage these unresolved trajectories. For this, we employ a critic LLM to assess the correctness of each step in a trajectory. Consequently, during training, we mask the loss for erroneous steps while retaining them in the context window. This way we ensure the model learns to recover from errors without reproducing them. Evaluation on `SWE-bench Verified` shows that while RFT improves the resolution rate by 2.4% by excluding unresolved trajectories, SRFT improves it by 3.7% by filtering them instead of discarding completely, reaching the total resolution rate of 32.2%.

1. Introduction

LLM-based agents are systems designed to autonomously perceive and interact with environments to achieve complex goals (Xi et al., 2023; Wang et al., 2023), with the ability to reason, plan, and use tools to solve open-ended problems. LLM-based agents often employ paradigms like ReAct (Yao et al., 2023) or Reflexion (Shinn et al., 2023) that interleave reasoning and acting phases, constituting the so-called trajectory that consists of multiple such steps.

Rejection sampling Fine-Tuning (RFT) (Yuan et al., 2023) has emerged as a standard paradigm for training LLM agents (Pan et al., 2025; Jain et al., 2025; Yang et al., 2025). This process involves generating multiple trajectories for a given task, filtering for those that successfully resolve the task, and performing supervised fine-tuning on this filtered set. Filtering is needed, since incorporating unresolved trajectories into the training set degrades performance by teaching the model to imitate erroneous actions. A major limitation of RFT is the inability to leverage unresolved trajectories. For instance, SWE-smith (Yang et al., 2025) introduces a large-scale dataset of agent trajectories obtained by solving synthesized software engineering tasks and a model trained on this dataset. However, due to RFT, they discard approximately 61% of the collected runs, therefore losing a lot of potentially informative data.

Our key insight is that unresolved trajectories are not entirely erroneous; rather, they often consist of correct and useful steps interspersed with errors. Our manual analysis of 20 trajectories indicates that even in unresolved trajectories, only up to 24% of steps can be classified as mistakes. To capitalize on this finding, we introduce Step Rejection Fine-Tuning (SRFT). In this method, we utilize a critic Large Language Model (LLM) to discriminate trajectories on a lower level, marking singular steps of trajectories as either worthy or unworthy to train on. This method allows the model to learn from the valid portions of unresolved trajectories without internalizing mistakes.

The contributions of this paper are as follows. We present a practical light-weight approach that allows utilizing the data routinely considered noise without changing the overall pipeline. The approach is detailed in Section 3. Then, we conduct an experiment that shows that on the challenging `SWE-bench Verified` benchmark, our method successfully improves over the naïve RFT baseline. The experiments are described and analyzed in Section 4.

2. Related Work

Research has increasingly moved beyond standard Rejection sampling Fine-Tuning—which treats all actions in a resolved trajectory as equally valid—towards extracting granular supervision from suboptimal or failed attempts.

Learn-by-interact (Su et al., 2025), an instance of a broader Hindsight Experience Replay (HER) approach (Andrychow-

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

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

icz et al., 2017), addresses instruction–trajectory misalignment in long trajectories. They propose a backward construction method that decomposes trajectories into shorter segments and synthesizes specific tasks for each, thereby creating valid demonstrations for these synthesized tasks. While this and other HER-based methods effectively utilize available data, they rely on generating synthetic instructions. In contrast, our approach focuses on filtering steps within the original task context, avoiding the need for task synthesis. Also, our method alleviates the risk of drifting off of the ground truth task.

SWEET-RL (Zhou et al., 2025) tackles the credit assignment problem by transforming trajectory-level feedback into step-wise signals. It trains a step-wise critic by comparing pairs of trajectories and constraining the score to be a sum of per-step contributions, forcing an implicit decomposition of preferences. This critic then guides policy optimization via Direct Preference Optimization (DPO; Rafailov et al., 2023). While effective, this introduces the complexity of Reinforcement Learning. In contrast, our method avoids training a critic with complex objectives, offering a simpler alternative within the supervised learning paradigm.

Finally, *STeP* (Chen et al., 2025) employs a teacher-in-the-loop to actively synthesize "self-reflected" trajectories where errors are immediately followed by teacher-generated reflections and corrections. By applying partial masking to error steps, they enable the model to learn recovery strategies. While our work shares the core mechanism of partial masking, we propose a simple practical approach that can retroactively fit into existing pipelines. Specifically, instead of requiring expensive real-time teacher intervention to synthesize new correction steps, we use an offline critic to salvage valid signals from standard unresolved trajectories.

3. Method

3.1. Preliminaries

We consider the problem of distilling the agentic behavior from a strong Teacher model to a weaker Student model. The distillation process consists of gathering the outputs of the Teacher and training the Student to mimic them. We denote the set of the collected Teacher outputs as $\mathcal{D}$. We denote the Student model as a function f_θ parametrized by weights θ, that estimates the probability of the next token given the previous tokens.

For agentic behavior in particular, we need to sample coherent ReAct trajectories rather than one-shot predictions, which complicates the dataset collection. Each trajectory $\tau_i \in \mathcal{D}$ is defined as $\tau_i = (s, u_i, (a_{i,0}, o_{i,0}), (a_{i,1}, o_{i,1}), \ldots, (a_{i,T_i}, o_{i,T_i}))$, comprising a system message s, a task description u_i, and a sequence of assistant actions and corresponding environment observations $\{(a_t, o_t)\}_{t=0}^{T_i}$, where T_i is the length of the i-th trajectory in steps. For the sake of simplicity, we use $\tau_{i,[0:t]}$ to denote the concatenation of the trajectory τ_i up to but not including the t-th step, and $a_{t,[0:j]}$ to denote the string representation of the action a_t up to but not including j-th token.

3.2. Methods Investigated in Current Work

The **naïve distillation** is performed by minimizing the following loss $\mathcal{L}(\theta, \mathcal{D})$, where NLL is the negative log-likelihood loss.

$$\mathcal{L}(\theta, \mathcal{D}) = \sum_{\tau_i \in \mathcal{D}} \sum_{t=0}^{T_i} \sum_{j=0}^{|a_t|} \mathrm{NLL}\big(f_\theta(\tau_{i,[0:t]} + a_{t,[0:j]}), a_{t,j}\big)$$

The **RFT** filters the raw teacher trajectories in $\mathcal{D}$ to retain only trajectories satisfying the final success criteria of the environment, resulting in the subset $\mathcal{D}_s$. For example, in `SWE-bench`, it means that after the agent has finished, a pre-defined set of tests should succeed. The Student model is then trained by minimizing the loss $\mathcal{L}(\theta, \mathcal{D}_s)$. However, this approach discards the set of unresolved trajectories $\mathcal{D}_f = \mathcal{D} \setminus \mathcal{D}_s$, which may be large for complex tasks.

To leverage the data rejected under the RFT framework, we introduce a fine-grained supervision mechanism **SRFT**. Instead of discarding or keeping the entire trajectories, we introduce weights $\mathcal{W}$ to keep alongside the trajectories. Weight $w_{i,t} \in \mathcal{W}$ corresponds to the step $(a_{i,t}, o_{i,t})$ in trajectory τ_i. Given the weights, we modify the loss to perform the weighted distillation on the whole dataset with loss $\mathcal{L}(\theta, \mathcal{D}, \mathcal{W})$.

$$\mathcal{L}_{\mathcal{W}}(\theta, \mathcal{D}, \mathcal{W}) = \sum_{\tau_i \in \mathcal{D}'} \sum_{t=0}^{T_i} w_{i,t} \cdot \sum_{j=0}^{|a_t|} \mathrm{NLL}\big(f_\theta(\tau_{i,[0:t]} + a_{t,[0:j]}), a_{t,j}\big)$$

3.3. SRFT Instantiation in Current Work

SRFT supports different weighting schemes, but for our current experiments, we instantiate it as follows. For successful trajectories in $\mathcal{D}_s$, we assign a weight of 1 to each step. For unresolved trajectories in $\mathcal{D}_f$, we employ a critic model to label each action a_t. We classify a step as *good* if it advances the task, *harmful* if it hinders progress (e.g., introduces a bug), and *unnecessary* if it neither helps nor damages. We group *good* and *unnecessary* steps into a single category of *Productive* steps. Accordingly, we assign $w_t = 0$ to harmful steps and $w_t = 1$ otherwise, forming $\mathcal{W}_f$. The total set of weights is $\mathcal{W} = \mathcal{W}_f \cup \mathcal{W}_s$.

Given this weighting scheme, the loss $\mathcal{L}_{\mathcal{W}}$ simply omits the loss calculation for the steps that were marked as harm-

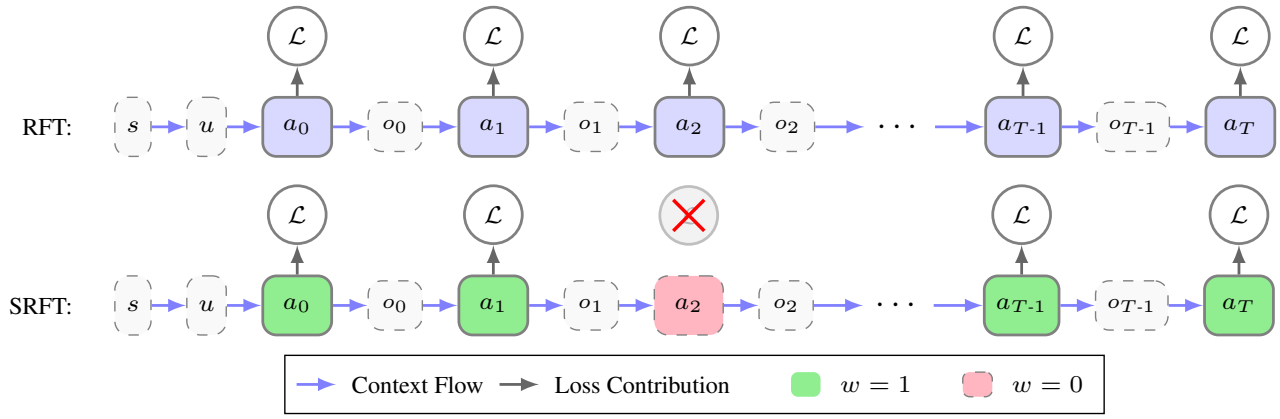

Figure 1. Comparison of RFT and SRFT training approaches.

ful. Conveniently, for binary weights, this corresponds to modifying token-wise masks. This is routinely done for observations, and including masks for some pre-defined steps incurs little engineering overhead. A further example of how this is applied to a trajectory is depicted in Figure 1.

4. Experimental Setup

To experimentally demonstrate that SRFT improves the performance of the Student model, we employ it on a complex Software Engineering task. We fine-tune `Qwen2.5-Coder-32B-Instruct` (Hui et al., 2024) as our Student model, using the SWE-Agent (Yang et al., 2024) framework as the scaffold. For training, we utilize the `SWE-smith-trajectories` dataset (Yang et al., 2025), which consists of approximately 25,000 trajectories generated by `SWE-agent`. The dataset comprises 39% resolved and 61% unresolved trajectories. For our study, we sample a balanced set of 5,000 resolved ($\mathcal{D}_s$) and 5,000 unresolved ($\mathcal{D}_f$) trajectories. We further instantiate datasets for different methods as described in Section 3.

To provide the markup for the SRFT method, we employ Claude 4 Sonnet (snapshot 20250514) (Anthropic, 2025) as the critic model, incurring a total cost of $660 for 5,000 trajectories. The critic prompt was selected by expert labeling of trajectories, and is presented alongside the expert data in Section A. Table 1 details the distribution of step labels. We note that unresolved trajectories contain more steps marked as harmful, but the critic was never provided with the resolution status of the trajectory. This aligns with the intuition that the unresolved trajectories contain potentially adversarial patterns not to be distilled.

We evaluate the performance of the Student model on the `SWE-bench Verified` dataset (OpenAI, 2024). To ensure robustness, each experiment is repeated 7 times. We report the Resolved Rate in percent. The results are presented in Table 2.

Table 1. Label distribution for 5,000 resolved and 5,000 unresolved trajectories.

Label	Resolved	Unresolved
Productive steps	95.9%	93.0%
Harmful steps	4.1%	7.1%

Consistent with our hypothesis, naively including unresolved trajectories leads to a performance degradation compared to the RFT baseline (28.5% vs 30.9%), as the model internalizes errors present in the failed attempts. However, by applying critic-guided masking, we not only mitigate this degradation but achieve a performance gain, outperforming the RFT baseline (32.2% vs 30.9%). This improvement is statistically significant; a bootstrap analysis confirms a gain of 1.3% with a 95% confidence interval of [0.4, 2.3] (refer to Appendix B for detailed statistical analysis).

Table 2. Main results on `SWE-bench Verified`. Each experiment was run 7 times; we report mean ± standard deviation. Base model performance is from the paper R2E-Gym (Jain et al., 2025), which uses the same scaffold and model.

Training Data	Resolved (%)
Base model	7.0 ± 1.3 (-21.5)
Naïve distillation	28.5 ± 1.7
RFT	30.9 ± 1.1 (+2.4)
SRFT	32.2 ± 0.9 (+3.7)

5. Limitations and Future Work

Our approach relies on the accuracy of the critic. Mislabeling valid steps as harmful can reduce the effective training data, while failing to identify subtle errors can allow them to propagate into the student model. We leave a thorough study of different critics and labeling approaches to future work.

This work is an early-stage report, and we did not explore the possibility of the loss weights beyond binary. However, we note that this may be the key to further improving the quality, both on resolved and unresolved trajectories.

While we test SRFT on a challenging SWE-bench task, we acknowledge the lack of a generalization study and leave the investigation of the method's generalizability to future work.

6. Conclusion

We have presented Step Rejection Fine-Tuning, a straightforward yet effective enhancement to the Rejection sampling Fine-Tuning distillation method that unlocks the value of unresolved trajectories. By selectively masking steps, we enable agents to learn from the partial successes of the Teacher model within failed attempts without internalizing its errors. On `SWE-bench Verified`, our method yields statistically significant improvements over the RFT (32.2% vs 30.9% resolved issues), highlighting the potential of step-level supervision in agentic distillation.

Acknowledgements

Acknowledgements will be added in the camera-ready version.

Impact Statement

This paper presents Step Rejection Fine-Tuning, a technique for improving the training of LLM-based software engineering agents by leveraging previously discarded trajectory data. The primary application domain is automated software development, where more capable agents could accelerate development cycles and lower barriers to software creation. We do not foresee direct negative societal consequences specific to this work beyond those generally associated with advances in code-generating AI systems, such as potential misuse for generating malicious code. The critic-guided masking approach we propose can be applied to other agentic domains, which may carry their own domain-specific risks that would need to be assessed separately.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Anthropic. Claude 4 system card. Technical report, Anthropic, 2025. URL https://www.anthropic.com/claude-4-system-card. Model version: claude-4-sonnet-20250514.

Chen, Y., Xu, B., Wang, X., Zhang, Y., and Mao, Z. Training llm-based agents with synthetic self-reflected trajectories and partial masking. *arXiv preprint arXiv:2505.20023*, 2025. URL https://arxiv.org/abs/2505.20023.

Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Dang, K., An, K., Zheng, Y., Xu, J., Lin, Y., and Zhou, J. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Jain, N., Singh, J., Shetty, M., Zheng, L., Sen, K., and Stoica, I. R2E-Gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint arXiv:2504.07164*, 2025. URL https://arxiv.org/abs/2504.07164.

OpenAI. SWE-bench Verified. Web page, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. Accessed: 2025-08-11.

Pan, J., Wang, X., Neubig, G., Jaitly, N., Ji, H., Suhr, A., and Zhang, Y. Training software engineering agents and verifiers with swe-gym. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*, 2025. URL https://arxiv.org/abs/2412.21139. arXiv:2412.21139, accepted at ICML 2025.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023. URL https://arxiv.org/abs/2305.18290.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *arXiv e-prints*, March 2023. doi: 10.48550/arXiv.2303.11366. URL https://arxiv.org/abs/2303.11366.

Su, H., Sun, R., Yoon, J., Yin, P., Yu, T., and Arık, S. O. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. *arXiv preprint arXiv:2501.10893*, 2025. URL https://arxiv.org/abs/2501.10893.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., and Wen, J.-R. A survey on large language model based autonomous agents. *arXiv e-prints*, August 2023. doi: 10.48550/arXiv.2308.11432. URL https://arxiv.org/abs/2308.11432.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., Yin, Z., Dou, S., Weng, R., Cheng, W., Zhang, Q., Qin, W., Zheng, Y., Qiu, X., Huang, X., and Gui, T. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864, 2023. doi: 10.48550/arXiv.2309.07864. URL http://arxiv.org/abs/2309.07864.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K. R., and Press, O. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024. URL https://arxiv.org/abs/2405.15793.

Yang, J., Lieret, K., Jimenez, C. E., Wettig, A., Khandpur, K., Zhang, Y., Hui, B., Press, O., Schmidt, L., and Yang, D. Swe-smith: Scaling data for software engineering agents, 2025. URL https://arxiv.org/abs/2504.21798.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Yuan, Z., Yuan, H., Li, C., Dong, G., Lu, K., Tan, C., Zhou, C., and Zhou, J. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023. URL https://arxiv.org/abs/2308.01825.

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

A. Critic Prompt and Evaluation

We validated the critic prompt by manually labeling a ground truth set of 20 trajectories (comprising 444 steps in total). Table 3 details the performance of the critic against these manual labels. The critic demonstrates robust performance in identifying *good* steps (F1=0.737), which constitutes the majority class. While performance on minority classes (*harmful* and *unnecessary*) is lower, the overall system benefit suggests that even imperfect filtering is preferable to naïve data inclusion or complete data exclusion.

Table 3. Critic prompt evaluation on 444 manually labeled steps.

Label	F1	Precision	Recall
good	0.737	0.660	0.835
unnecessary	0.236	0.253	0.221
harmful	0.384	0.604	0.282
Accuracy		0.586	

The specific prompt used for the critic is provided below:

```
You are evaluating SWE-agent trajectories.
Your goal is to annotate each step of the trajectory.
This annotation will be used to train a model only on good steps.
First, construct a brief "ideal solution path" and patch (in your own reasoning, not
in the output).
Then, for each step, decide if it matches or deviates from this ideal path.

## Mark Each Step
- **"good"**:  Part of or necessary for the ideal shortest path (exploration, correct
fixes).
- **"harmful"**:  Moves away from the path (wrong edits, irrelevant files), harmful
tool usage, etc.
- **"unnecessary"**:  Not harmful but not part of the path (redundant exploration,
repeated actions).

## Rules
- Always compare each step to the shortest correct path you imagined.
- A step is "good" only if it is part of or necessary for that shortest path.
- A step is "harmful" if it deviates from or worsens the trajectory relative to that
path.
- A step is "unnecessary" if it neither helps nor harms but is outside the shortest
path.
```

B. Statistical Significance and Extended Results

Given the inherent high variance in LLM agent evaluation, we conducted a rigorous statistical analysis. We identified two primary sources of noise: rollout variance, where two distinct rollouts of the same model can differ by up to 4.8%, and training variance, where two models trained with different seeds can differ by 1.1% on average across 7 rollouts.

To verify that our method provides a statistically significant improvement, we focused on the unresolved split, which most clearly highlights the difference between using all steps versus masking. We performed 5 training runs: 3 without masking and 2 with masking. Accounting for both sources of noise, we still observe a performance increase (see Table 4). As shown in Figure 2, the bootstrap analysis confirms a statistically significant improvement of 1.1% with a 95% confidence interval of [0.4, 1.8].

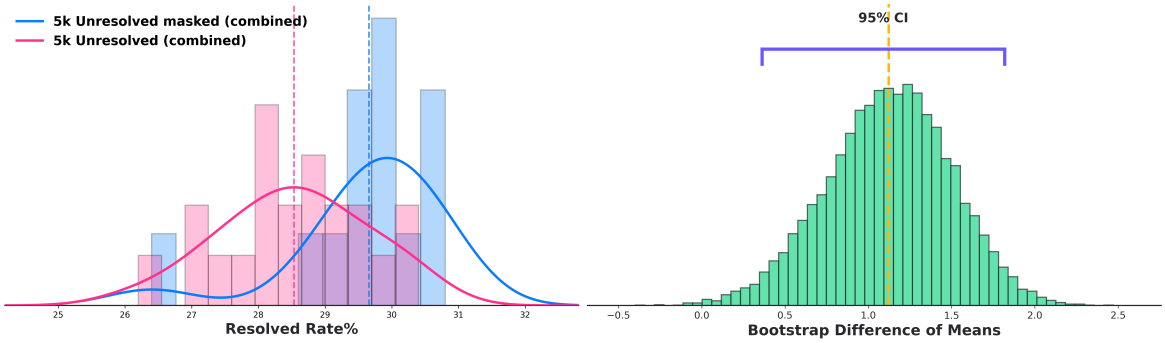

Figure 2. Bootstrap analysis of Resolved Rate% for 5k Unresolved vs 5k Unresolved masked.

Table 4. Results from multiple training runs on 5k Unresolved trajectories. Mean and standard deviation are calculated over 7 rollouts for individual runs; combined results are calculated over 21 and 14 rollouts respectively.

Configuration	Resolved%	pass@7%
5k Unresolved (train #1)	28.8 ± 1.0	42.8
5k Unresolved (train #2)	27.7 ± 0.9	42.2
5k Unresolved (train #3)	29.1 ± 1.0	41.8
5k Unresolved (combined)	28.5 ± 1.1	42.3
5k Unresolved masked (train #1)	29.8 ± 0.6	42.6
5k Unresolved masked (train #2)	29.5 ± 1.5	43.8
5k Unresolved masked (combined)	29.7 ± 1.1	43.2

Table 5. Extended experimental results on `SWE-bench Verified`.

	Training Data	Resolved%	pass@7%
Base model		7.0 ± 1.3	
RFT	5k Resolved	30.9 ± 1.1	45.8
	5k Unresolved	27.7 ± 0.9	42.2
Naïve distillation	5k Resolved + 5k Unresolved	28.5 ± 1.7	43.8
SRFT	5k Resolved + 5k Unresolved (masked)	**32.2** ± 0.9	45.8
	5k Resolved (masked) + 5k Unresolved (masked)	29.4 ± 1.6	41.2