RoboFAPE: Failure-Aware Progress Estimation for Robot Manipulation

Yingxi Lu[1,3] **Kairan Dou**[2] **Bu He**[1] **Mengdi Xu**[1,3†]
[1]Tsinghua University [2]Carnegie Mellon University [3]Shanghai Qizhi Institution
yingxilu512@gmail.com xumd0105@gmail.com

Abstract: Task-progress estimates can support reward modeling, failure detection, and trajectory filtering in robot learning. However, temporal labels derived from successful demonstrations provide limited supervision for executions that stall or regress. We present RoboFAPE, a framework for incorporating naturally occurring policy failures into progress estimation. Its labeling pipeline decomposes tasks into stages and aligns failure states with successful references using task-specific distances and segment-aware matching. We curate 6,400 simulated trajectories and evaluate a RoboMeter-based estimator on progress prediction and three downstream applications. Natural-failure supervision improves progress estimation on in-distribution failure trajectories and achieves the highest in-distribution failure-detection and filtering scores among the evaluated training variants. These advantages do not transfer uniformly to out-of-distribution settings: failure detection remains below the rewind-augmentation baseline, and filtering remains below success-only training. A single-task reinforcement-learning experiment provides preliminary evidence of downstream utility. Our results motivate direct supervision from natural failures while highlighting the need for independent label validation and stronger generalization tests. Project page: https://luyingxi35.github.io/RoboFAPE/.

1 Introduction

Task progress describes how far an execution has advanced toward a goal. A learned progress signal can serve several purposes in robot learning: providing dense rewards when task-specific reward design is difficult [1, 2], identifying stalled or regressing executions [3, 4], and ranking demonstrations for data curation [5, 6]. These uses become increasingly relevant as robot policies learn from large and heterogeneous datasets [7, 8].

A common supervision strategy assigns progress according to temporal position in successful demonstrations [9, 10, 6]. This provides a scalable proxy when execution advances toward the goal, but it does not directly describe what happens after a missed grasp, a dropped object, or a prolonged stall. Such states may occur frequently during policy deployment, even when they are poorly represented in successful demonstrations. Learning to assess them is therefore an important part of making progress estimators useful beyond expert behavior.

Existing approaches incorporate imperfect execution through rewinding, counterfactual relabeling, synthetic failures, trajectory preferences, and retry supervision [11, 12, 13, 14, 15]. We study a specific mechanism: assigning frame-level progress labels to naturally occurring policy failures by relating their states to successful references, while retaining an existing estimator architecture.

RoboFAPE combines VLM-based task decomposition with state-based reference alignment. Segment-aware matching addresses cases in which the same subgoal distance occurs at multiple

†Corresponding author.

CoRL 2026 Workshop on Continually Self-Improving Robots (CSIR).

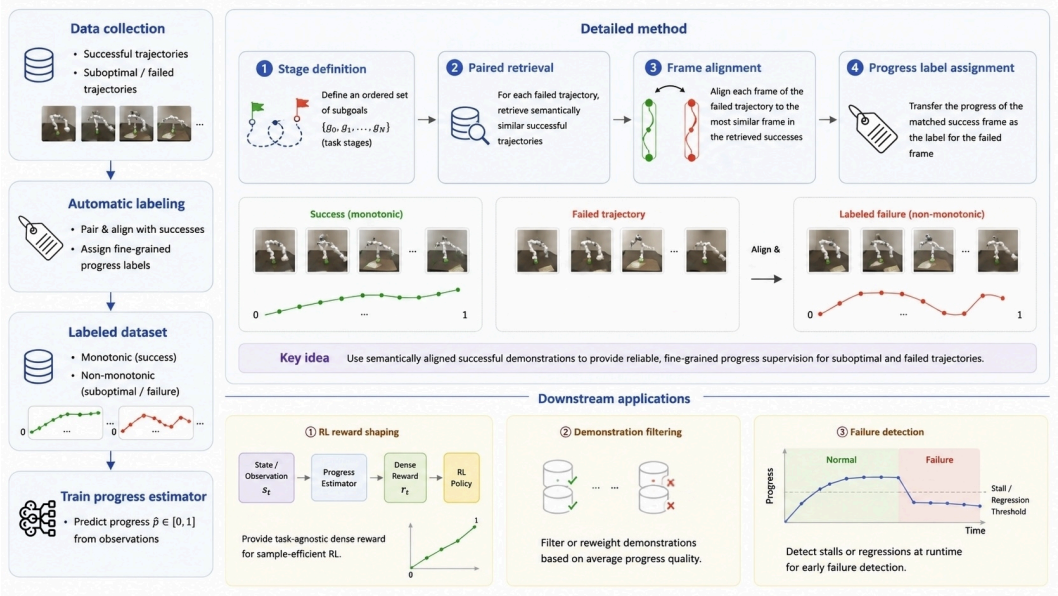


Figure 1: RoboFAPE overview. Successful references and policy failures are organized by task stage. State-based matching transfers progress supervision to stalled or regressing failure frames. The trained estimator supports progress prediction, reward shaping, filtering, and failure detection. Task stages and distance functions used for labeling are task-specific.

reference frames, allowing labels to stall or decrease. Annotation relies on task-specific subgoals and object-state information; it is not task-independent.

1. **A dataset of successful and failed executions.** We curate 6,400 simulated manipulation trajectories, including naturally occurring policy failures, with frame-level progress annotations.
2. **A failure-labeling procedure.** We describe stage-based reference selection and segment-aware alignment for assigning progress supervision to failed executions.
3. **An empirical comparison of training data.** Using a shared RoboMeter-based architecture, we compare natural failures with success-only training and rewind augmentation on progress estimation, failure detection, trajectory filtering, and a preliminary RL application.

2 Related Work

Progress and process reward estimation. Progress estimators use temporal supervision, self-supervised refinement, stage decomposition, or reference information to assess task completion [9, 6, 16, 10, 17, 18]. Related approaches obtain reward or value signals from visual-language feedback, relative progress, video models, and feature matching [2, 19, 20, 21, 22, 23, 24]. These signals serve related purposes, but progress labels, preferences, success predictions, and return estimates are different supervision targets. Our focus is the construction of frame-level progress labels for naturally failed executions.

Learning from failed and suboptimal trajectories. ReWiND uses rewinding-based reward supervision [11]; RoboReward constructs negative and near-miss examples through counterfactual relabeling and temporal clipping [12]; and DenseReward synthesizes failures in simulation [13]. ReTVL exploits retry events [15], while success–failure visitation matching provides process-reward supervision [25]. RoboMeter combines frame-level supervision with trajectory comparisons to learn from executions of different quality [14]. Together, these studies show that using failure information is not unique to our framework.

RoboFAPE investigates a particular way to use that information: transferring progress labels from successful references to naturally occurring failure states through task-stage and distance-based alignment. Our estimator retains the RoboMeter architecture. Accordingly, the experiments isolate training-data variants within that architecture; the rewind-augmentation variant is not a reproduction of the complete ReWiND method, and the experiments do not establish superiority over all of the approaches above.

3 Methodology

3.1 Problem Formulation

For a target task $\mathcal{T}$, let $\mathcal{D}_S$ and $\mathcal{D}_F$ contain successful and naturally failed trajectories, respectively. A trajectory $\tau = (o_1, \ldots, o_T)$ contains observation frames. During labeling, we assume access to task-relevant object states $\mathbf{x}_t \in \mathbb{R}^D$, obtained from the simulator in our experiments. We seek frame-level labels $p_t \in [0, 1]$ for failed executions and use the combined labeled dataset $\mathcal{D} = \mathcal{D}_S \cup \mathcal{D}_F$ to train a progress estimator. We denote its prediction by $\widehat{p}_t = \Phi_\theta(o_t)$, suppressing other conditioning in the notation.

For successful demonstrations, temporal interpolation $p_t = t/T$ supplies a progress proxy. This convention assumes that later frames generally represent greater task advancement; it is not a guarantee about physical progress in every successful execution. For failures, forcing the same temporal rule would label stalls and regressions as continued advancement. RoboFAPE instead establishes correspondences with successful references. The task-specific state information is used to construct supervision, whereas the learned estimator predicts progress from observations.

3.2 Stage-Based Reference Alignment

Task stages and distance functions. A vision-language model (VLM) decomposes $\mathcal{T}$ into ordered stages $\phi_1, \ldots, \phi_K$ and identifies stage boundaries from visual evidence of subgoal achievement. For each stage k, it specifies a subgoal g_k and a distance function $d_k(\mathbf{x})$. For example, a pick-and-place task may use end-effector-to-object distance during approach and object-to-target distance during placement. These distances are task-dependent correspondence signals; a small distance alone does not certify a valid grasp or successful completion.

Reference selection. Write a failure stage as $\tau_F^{(k)}$ and the available successful stages as $\mathcal{D}_S^{(k)}$. We select a reference whose initial distance to the subgoal is closest to that of the failure stage:

$$\tau_S^{(k)*} = \arg \min_{\tau_S^{(k)} \in \mathcal{D}_S^{(k)}} \left| d_k(\mathbf{x}_{F,\text{start}}) - d_k(\mathbf{x}_{S,\text{start}}) \right|. \tag{1}$$

This matches the starting distance, rather than the full semantic or geometric state. For a reference stage with m frames, let $u_j^S = j/m$ denote its local temporal progress. A direct nearest-distance mapping of failure frame q would be

$$j^* = \arg \min_{1 \le j \le m} \left| d_k(\mathbf{x}_q^F) - d_k(\mathbf{x}_j^S) \right|, \qquad u_q^F = u_{j^*}^S. \tag{2}$$

Local stage progress u is distinguished here from task-level progress p. The labeling pipeline uses stage-specific correspondences to provide frame-level supervision across the task.

Why segment-aware matching is needed. The reference distance may decrease and then increase as objects interact. A failure frame can therefore have a similar distance to several reference frames with different progress values. We partition the reference distance profile into monotone subsegments $\mathcal{S}_1, \ldots, \mathcal{S}_J$ and use the preceding failure label to choose among plausible matches. Here k indexes task stages and j indexes reference frames; J denotes the number of subsegments.

For each subsegment i, define its best distance match and corresponding local progress:

$$\begin{aligned} j_i &\in \arg \min_{j \in \mathcal{S}_i} \left| d_k(\mathbf{x}_q^F) - d_k(\mathbf{x}_j^S) \right|, \\ \delta_i &= \left| d_k(\mathbf{x}_q^F) - d_k(\mathbf{x}_{j_i}^S) \right|, \qquad u_i^{\text{match}} = u_{j_i}^S. \end{aligned} \tag{3}$$

Retain candidates in $\mathcal{C}_q = \{i : \delta_i - \min_h \delta_h \le \epsilon\}$, using $\epsilon = 0.001$ in the reported procedure. Then select

$$i_q^* = \arg\min_{i \in \mathcal{C}_q} \left(\frac{\delta_i}{\sigma_d} + \frac{|u_i^{\text{match}} - u_{q-1}^F|}{\sigma_p} \right), \qquad u_q^F = u_{i_q^*}^{\text{match}}. \tag{4}$$

The reference-derived scales σ_d and σ_p normalize per-step distance and progress changes. The first term favors a close state-distance match; the second discourages large changes relative to the preceding label. It penalizes the magnitude of change, not its sign, so progress can decrease as well as increase.

Equation 4 is a sequential selection rule using the previously assigned label. Although it combines local matching and transition costs, it does not express full-sequence Viterbi decoding [26] or guarantee a globally optimal correspondence. Nor does monotonicity alone guarantee a unique nearest frame in sampled data: ties, plateaus, and noisy extrema require implementation choices. These distinctions matter when interpreting the alignment as a labeling heuristic rather than an independent measure of task completion.

3.3 Progress Estimation Training

We retain the RoboMeter-based estimator and compare supervision sources under a shared training setup. Following the twelve-task configuration, the progress head predicts a distribution q_{btk} over $N_{\text{bin}} = 10$ equally spaced bins with centers $c_k = k/(N_{\text{bin}} - 1)$, where $k = 0, \ldots, N_{\text{bin}} - 1$. Each continuous target is represented by linear interpolation between its two adjacent bin centers. For target distribution y_{btk}, the per-frame loss and scalar prediction are

$$\ell_{bt}^{\text{prog}} = -\sum_{k=0}^{N_{\text{bin}}-1} y_{btk} \log q_{btk}, \qquad \widehat{p}_{bt} = \sum_{k=0}^{N_{\text{bin}}-1} c_k q_{btk}. \tag{5}$$

Progress cross-entropy is trained jointly with preference and success objectives. Appendix C specifies their masking and reduction. Successful frames use temporal labels; failed frames use reference-alignment labels. MSE is an evaluation metric, not the training loss.

4 Experiments

4.1 Experimental Setup

Data and splits. The dataset contains 6,400 simulated trajectories: 4,800 for training and 1,600 for evaluation. The training suite includes 12 manipulation tasks in ManiSkill2 [27]; approximately half of the collected data build on RoboFAC [28]. Within each domain, successful and naturally failed executions are equally represented. IND evaluation uses held-out trajectories from the training tasks. OOD evaluation covers changed environments and objects, as well as unseen tasks. Appendix A provides the task list and split statistics, and Appendix B describes policy-failure collection.

Training variants. All variants initialize from the released RoboMeter-4B checkpoint [14], with the same architecture, total number of training trajectories, and optimization budget. We compare *Success Only*, *Success + Rewind Failure*, and *Success + Natural Failure* (RoboFAPE). The second variant uses rewind-video augmentation; the third uses labeled policy failures. This comparison assesses alternative training-data compositions within a shared estimator, not architectural novelty or a full comparison of reward-learning systems.

Evaluation labels and reporting. For intrinsic evaluation, labels are generated by the proposed pipeline and manually calibrated in ambiguous cases. We call them *reference labels*: agreement with them is not independent evidence that the labeling pipeline recovers true physical progress. For detection and filtering, the reported tables use the best observed scores on each evaluation split across evaluated checkpoints, including step 0, rather than a checkpoint selected exclusively on a separate validation set. Such selection can be optimistic and can choose different checkpoints for different splits or metrics.

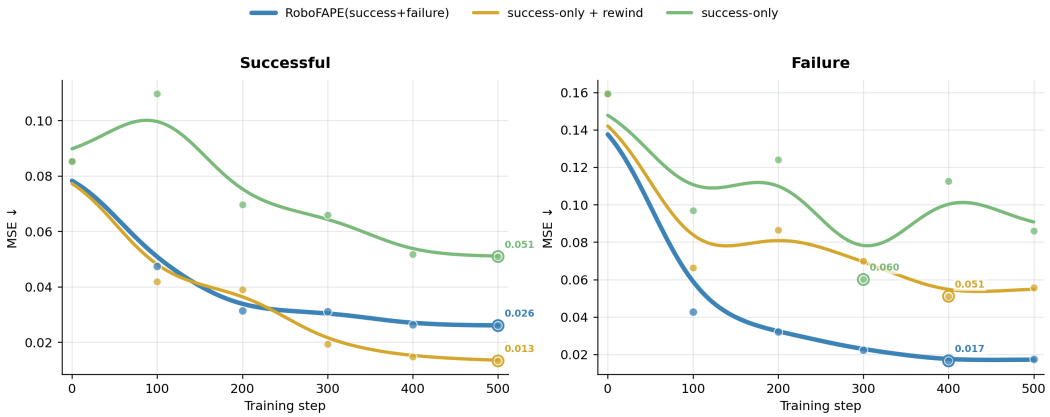


Figure 2: IND MSE versus estimator training step (lower is better), for successful and failed trajectories. Dots are measurements; smooth lines are visual guides. Ringed best observed values need not share a checkpoint.

Reading the figures. Scatter markers represent measurements; smooth lines are visual guides, not additional evaluations or multi-seed uncertainty estimates. Our interpretation concerns measured points, not extrema or crossings of fitted lines.

4.2 Progress Estimation

For a trajectory with T evaluated frames, prediction error against reference labels is

$$\text{MSE} = \frac{1}{T}\sum_{t=1}^{T}(\widehat{p}_t - p_t)^2. \tag{6}$$

We also use Spearman correlation to measure agreement between predicted and reference *progress rankings*. It is the Pearson correlation of their ranks. When no ranks are tied, this reduces to

$$\rho = 1 - \frac{6\sum_{t=1}^{T} a_t^2}{T(T^2-1)}, \tag{7}$$

where a_t is the difference between the two ranks of frame t. This metric does not require progress to increase with time; both rankings may reflect a regression. It should not be confused with the progress–time correlation used for downstream heuristics.

IND results. Figures 2 and 3 show the clearest benefit on failed trajectories: natural-failure supervision yields lower MSE at the displayed trained checkpoints and higher rank correlation at the later checkpoints. Successful-trajectory results involve a tradeoff. RoboFAPE reaches a higher observed peak rank correlation, but rewind augmentation achieves lower MSE at several checkpoints. Thus, the results support improved assessment of IND failure states rather than uniform improvement on both outcomes and both metrics.

OOD results. Figures 4 and 5 show that the IND advantage does not transfer uniformly. RoboFAPE attains higher observed peak Spearman values, but these peaks occur at earlier checkpoints and are not sustained at later evaluations. Rewind augmentation achieves lower successful-trajectory MSE and competitive failure-trajectory MSE. Selecting each method's peak on OOD data should not be interpreted as performance available without OOD-informed selection. These results motivate studying generalization separately from IND label agreement.

4.3 Dense Reward for Reinforcement Learning

We conduct a preliminary experiment on *PushCube* using proximal policy optimization (PPO). Following the reward-shaping protocol described for RoboMeter [14], the reward is $r_t = \widehat{p}_t$ when the

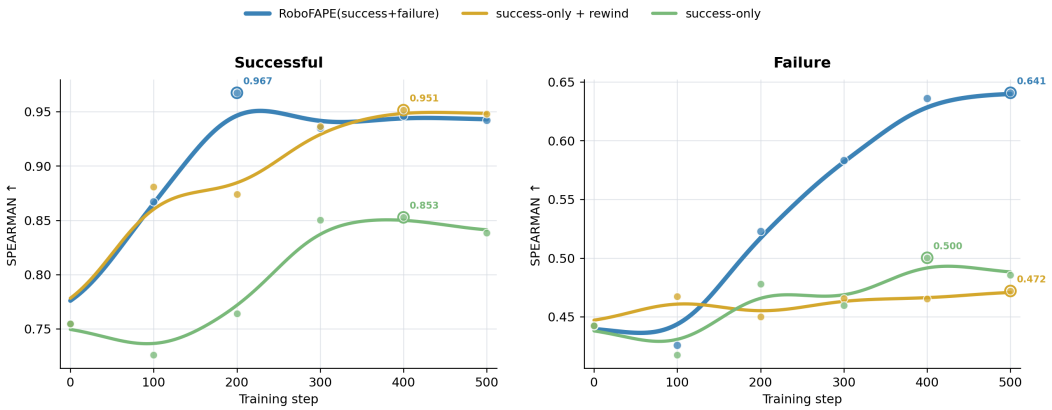


Figure 3: IND Spearman correlation with reference progress labels (higher is better), separated by outcome. Dots are measurements; lines are visual guides. Rank agreement does not require monotonic advancement with time.

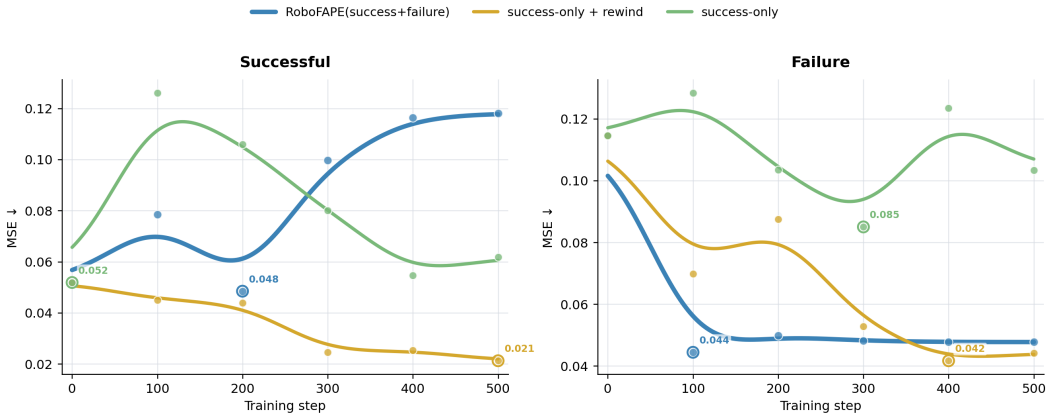


Figure 4: OOD MSE versus estimator training step (lower is better), separated by outcome. OOD combines shifted environments or objects and unseen tasks. Dots are measurements; lines are visual guides.

task success condition is satisfied at step t and $r_t = \widehat{p}_t - 1$ otherwise. The reward therefore uses both predicted progress and a task-success signal; it is not obtained from the progress estimator alone.

We examine policy success rate and critic explained variance,

$$\mathrm{EV} = 1 - \frac{\mathrm{Var}(V_{\text{target}} - V_{\text{pred}})}{\mathrm{Var}(V_{\text{target}})}, \tag{8}$$

where V_{target} is the empirical return and V_{pred} is the critic prediction. Higher EV indicates a smaller residual variance relative to the variance of the return target under that training condition.

Figure 6 shows modest success-rate differences and higher reported EV for RoboFAPE than for success-only training. Because each reward model defines its own return targets, higher EV does not by itself demonstrate more accurate task rewards or explain a causal improvement in policy learning. With one task and no reported multi-seed uncertainty, this experiment illustrates a possible application rather than establishing general RL gains.

4.4 Failure Detection

Progress sequences are downsampled to at most 30 frames. A trajectory is classified as failed if the Pearson correlation between predicted progress and time in any 12-frame window is below -0.2, or

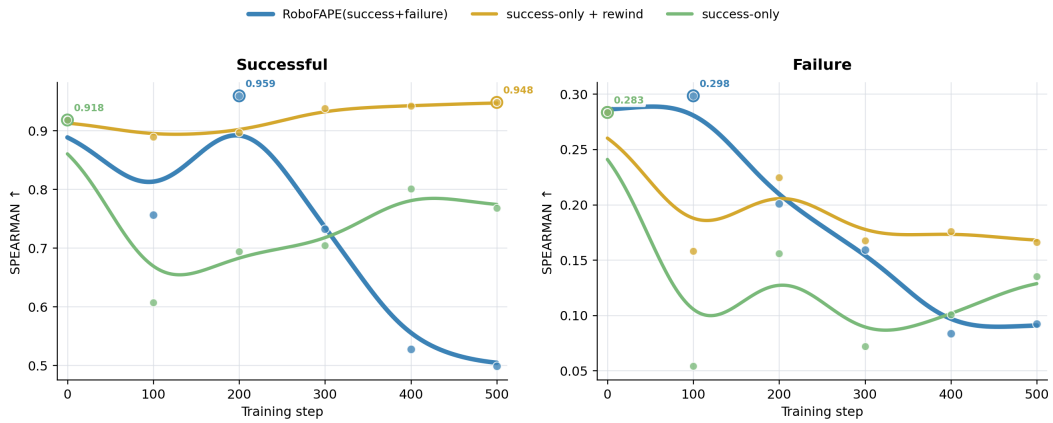


Figure 5: OOD Spearman correlation (higher is better). RoboFAPE's marked peaks occur at different checkpoints for successes and failures. Dots are measurements; lines are visual guides. Peak scores do not establish a sustained advantage.

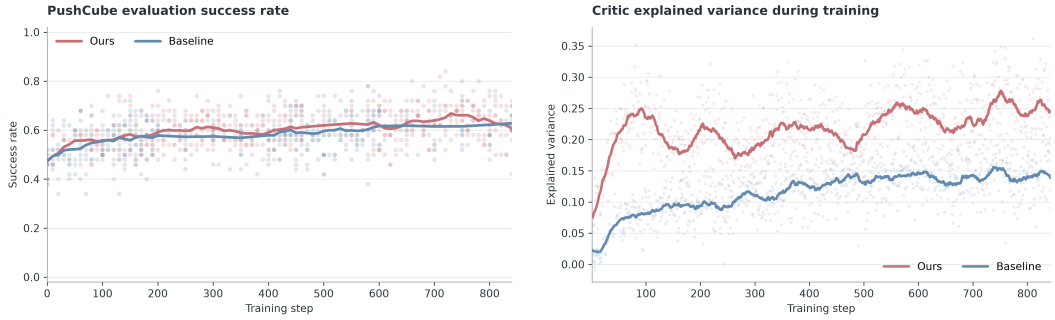


Figure 6: PushCube PPO: success rate (left) and critic explained variance (right) versus training step. "Ours" is RoboFAPE; "Baseline" is success-only training. Dots are measurements; smoothed traces are visual guides, not confidence intervals.

if the final predicted progress is below 0.5; otherwise, it is classified as successful. This trajectory-level test uses the observed sequence; it measures neither online detection latency nor whether a regression is unrecoverable.

We report Failure F1 averaged equally over task set $\mathcal{Q}$, taking failure as the positive class:

$$\mathrm{F1}_{\text{failure}} = \frac{1}{|\mathcal{Q}|} \sum_{a \in \mathcal{Q}} \frac{2\mathrm{TP}_a}{2\mathrm{TP}_a + \mathrm{FP}_a + \mathrm{FN}_a}. \tag{9}$$

Table 1 shows an IND advantage for RoboFAPE (0.812 versus 0.649 for rewind augmentation). On OOD tasks, the ordering reverses: RoboFAPE scores 0.714, below 0.797 for rewind augmentation, although above 0.566 for success-only training. The higher reported average does not remove this OOD limitation. Appendix D includes the existing checkpoint plots.

4.5 Trajectory Filtering

For each query task $\mathcal{T}$, the candidate pool contains all test trajectories across tasks and quality levels. We score each candidate i under the query using three signals: progress–time Pearson correlation r_i, net progress change $\Delta_i = \widehat{p}_{i,L} - \widehat{p}_{i,1}$, and final progress $f_i = \widehat{p}_{i,L}$. Each signal is normalized within the query-task candidate set and combined as

$$s_i = \mathrm{Norm}_{\mathcal{T}}(r_i) + \mathrm{Norm}_{\mathcal{T}}(\Delta_i) + \mathrm{Norm}_{\mathcal{T}}(f_i). \tag{10}$$

Table 1: Reported Failure F1 (higher is better), using best-on-split selection (Section 4.1). Bold indicates the highest value in each column.

Training data	IND	OOD	Average
RoboFAPE	**0.812**	0.714	**0.763**
Success-only + rewind	0.649	**0.797**	0.723
Success-only	0.314	0.566	0.440

Table 2: Reported top-50 filtering correctness (higher is better), using Equation 10 and best-on-split checkpoint selection. Bold indicates the highest value in each column.

Training data	IND	OOD	Average
RoboFAPE	**0.958**	0.770	0.864
Success-only + rewind	0.197	0.695	0.446
Success-only	0.836	**0.910**	**0.873**

We retrieve the top $K = 50$ trajectories. A retrieval is correct if it belongs to the query task and is labeled successful. This evaluates retrieval correctness, not the performance of a policy subsequently trained on the retrieved data.

RoboFAPE achieves the highest IND correctness (0.958), but success-only training performs better on OOD queries (0.910 versus 0.770) and has a slightly higher reported average (0.873 versus 0.864). Rewind augmentation performs worse in this filtering setup, particularly on IND queries. These results support the value of natural-failure supervision for IND retrieval without establishing a general advantage under distribution shift or over other filtering methods.

5 Conclusion

RoboFAPE incorporates naturally occurring policy failures into progress estimation through stage-based alignment with successful references. In the evaluated RoboMeter-based training setup, this supervision improves progress estimation on in-distribution failure trajectories and yields stronger in-distribution failure detection and filtering. OOD results remain mixed, and the PushCube RL experiment provides only preliminary downstream evidence. These findings support further investigation of natural-failure supervision while leaving independent label validation, component attribution, and broader generalization as open questions.

6 Limitations and Future Work

Label validity and component attribution. Intrinsic evaluation uses labels produced by our pipeline and manually calibrated in ambiguous cases. It therefore measures agreement with that annotation convention, not independent validation of physical task progress. Task-specific distance functions can assign similar values to semantically different states, and successful-reference timing is itself a proxy for progress. The current training-data comparison does not isolate the effects of stage decomposition, segment-aware matching, and temporal coherence. Independent annotations and component ablations are needed to evaluate these choices.

Generalization and downstream evidence. All data are simulated, and labeling assumes access to task-relevant object states and subgoals. Perception errors, alternative solution paths, and long-horizon recovery are not evaluated. OOD detection and filtering do not consistently improve, and reporting the best checkpoint on each evaluation split can overestimate performance available under validation-only selection. The single-task RL result and the absence of reported multi-seed uncertainty do not establish general or statistically reliable policy-learning gains. Future work should address these limitations before using progress estimates as standalone signals for autonomous robot improvement.

References

[1] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu. Text2reward: Reward shaping with language models for reinforcement learning, 2024. URL https://arxiv.org/abs/2309.11489.

[2] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback, 2024. URL https://arxiv.org/abs/2402.03681.

[3] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. Ambrus, H. Nishimura, and M. Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies, 2025. URL https://arxiv.org/abs/2503.08558.

[4] R. Römer, A. Kobras, L. Worbis, and A. P. Schoellig. Failure prediction at runtime for generative robot policies, 2025. URL https://arxiv.org/abs/2510.09459.

[5] K. Wu, N. Liu, Z. Zhao, D. Qiu, J. Li, Z. Che, Z. Xu, and J. Tang. Learning from imperfect demonstrations with self-supervision for robotic manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, page 16899–16906. IEEE, May 2025. doi:10.1109/icra55743.2025.11127918. URL http://dx.doi.org/10.1109/ICRA55743.2025.11127918.

[6] Q. Chen, J. Yu, M. Schwager, P. Abbeel, Y. Shentu, and P. Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation, 2026. URL https://arxiv.org/abs/2509.25358.

[7] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation, 2026. URL https://arxiv.org/abs/2410.18647.

[8] E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, M. Z. Irshad, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song,

S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, V. Guizilini, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2025. URL https://arxiv.org/abs/2310.08864.

[9] T. Ayalew, X. Zhang, K. Y. Wu, T. Jiang, M. Maire, and M. R. Walter. Progressor: A perceptually guided reward estimator with self-supervised online refinement, 2024. URL https://arxiv.org/abs/2411.17764.

[10] Y. Feng, H. Shi, H. Li, X. Guo, Y. Wang, C. Zhang, J. Zhang, X. Zhang, J. Tang, and J. Zhang. Procvlm: Learning procedure-grounded progress rewards for robotic manipulation, 2026. URL https://arxiv.org/abs/2605.08774.

[11] J. Zhang, Y. Luo, A. Anwar, S. A. Sontakke, J. J. Lim, J. Thomason, E. Biyik, and J. Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations, 2025. URL https://arxiv.org/abs/2505.10911.

[12] T. Lee, A. Wagenmaker, K. Pertsch, P. Liang, S. Levine, and C. Finn. Roboreward: General-purpose vision-language reward models for robotics, 2026. URL https://arxiv.org/abs/2601.00675.

[13] Y. Fang, W. Dong, J. Liu, Y. Yang, M. Huo, Y. Mu, H. Yao, L. E. Li, D. Szafir, and M. Ding. Densereward: Dense reward learning via failure synthesis for robotic manipulation, 2026. URL https://arxiv.org/abs/2607.13033.

[14] A. Liang, Y. Korkmaz, J. Zhang, M. Hwang, A. Anwar, S. Kaushik, A. Shah, A. S. Huang, L. Zettlemoyer, D. Fox, Y. Xiang, A. Li, A. Bobu, A. Gupta, S. Tu, E. Biyik, and J. Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons, 2026. URL https://arxiv.org/abs/2603.02115.

[15] X. Qin, J. Lu, K. Wang, C. Zhang, S. Kang, K. Lee, M. Xu, B. Liang, J. Yang, and L. Zhao. Beyond monotonic progress: Retry-supervised value learning for robot imitation, 2026. URL https://arxiv.org/abs/2606.24633.

[16] Q. Chen, H. Zheng, J. Yu, S. Huang, J. Sun, K. Goldberg, C. Wen, P. Abbeel, Y. Shentu, P. Wu, and M. Schwager. Sarm2: Multi-task stage aware reward modeling for self improving robotic manipulation, 2026. URL https://arxiv.org/abs/2606.10305.

[17] P. Yang, X. Wang, P. Jing, K. Wen, Y. Qu, Z. Huang, M. Fu, X. Liu, Y. Shen, and F. Shi. Rarm: Confidence-gated progress reward modeling for rl in manipulation, 2026. URL https://arxiv.org/abs/2606.22027.

[18] H. Tan, S. Chen, Y. Xu, Z. Wang, Y. Ji, C. Chi, Y. Lyu, Z. Zhao, X. Chen, P. Co, S. Xie, G. Yao, P. Wang, Z. Wang, and S. Zhang. Robo-dopamine: General process reward modeling for high-precision robotic manipulation, 2025. URL https://arxiv.org/abs/2512.23703.

[19] Y. J. Ma, J. Hejna, A. Wahid, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, J. Tompson, O. Bastani, D. Jayaraman, W. Yu, T. Zhang, D. Sadigh, and F. Xia. Vision language models are in-context value learners, 2024. URL https://arxiv.org/abs/2411.04549.

[20] S. Zhai, Q. Zhang, T. Zhang, F. Huang, H. Zhang, M. Zhou, S. Zhang, L. Liu, S. Lin, and J. Pang. A vision-language-action-critic model for robotic real-world reinforcement learning, 2025. URL https://arxiv.org/abs/2509.15937.

[21] Y. Wu, W. Yuan, A. Qi, V. Guizilini, J. Mao, and Y. Wang. Large reward models: Generalizable online robot reward generation with vision-language models, 2026. URL https://arxiv.org/abs/2603.16065.

[22] J. Lv, H. Li, J. Li, F. Kong, Y. Wang, P. Yi, Y. Nie, X. Wang, Z. Zhu, C. Ni, Q. Deng, H. Li, J. Lv, and G. Huang. Viva: A video-generative value model for robot reinforcement learning, 2026. URL https://arxiv.org/abs/2604.08168.

[23] Z. Wang, J. Li, Y. Cui, Y. Gao, X. Zhan, J. Yu, and X. Ma. World value models for robotic manipulation, 2026. URL https://arxiv.org/abs/2606.24742.

[24] J. Shu, Z. Lin, B. Wei, and Y. Wang. Feat2go: Visual feature-grounded value estimation for embodied reinforcement learning, 2026. URL https://arxiv.org/abs/2605.30795.

[25] R. Tsao, A. Wagenmaker, and S. Levine. Learning process rewards via success visitation matching for efficient rl, 2026. URL https://arxiv.org/abs/2606.23640.

[26] G. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. doi:10.1109/PROC.1973.9030.

[27] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

[28] W. Lu, M. Ye, Z. Ye, R. Tao, S. Yang, and B. Zhao. Robofac: A comprehensive framework for robotic failure analysis and correction, 2025. URL https://arxiv.org/abs/2505.12224.

[29] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

A Dataset Composition

The training set contains 4,800 trajectories across 12 tasks. The evaluation set contains 1,200 IND trajectories and 400 OOD trajectories. Figure 7 lists the training tasks and distinguishes the changed environments or objects from the two unseen tasks in the OOD set.

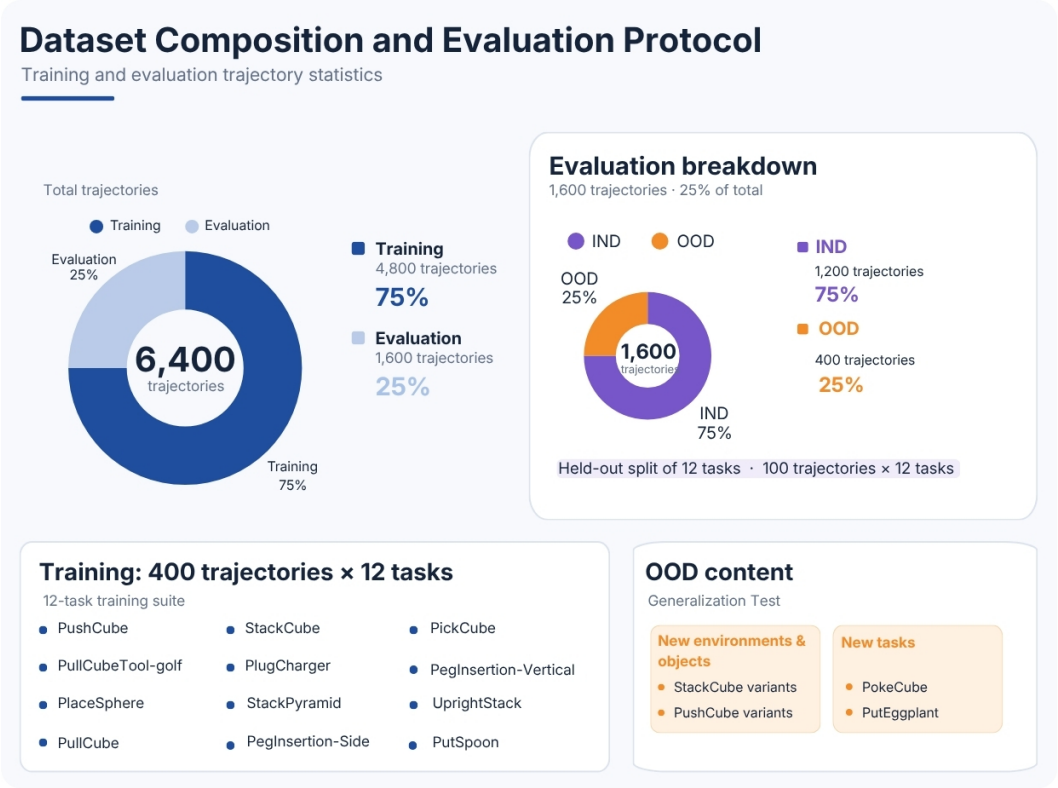


Figure 7: Dataset composition and evaluation split. Training uses 400 trajectories for each of 12 tasks. IND evaluation uses 100 held-out trajectories per training task; OOD evaluation includes StackCube and PushCube variants, PokeCube, and PutEggplant. These are the dataset counts shown in the reported experiment, distinct from the trajectories used to fine-tune the failure-generating policy.

B Natural Failure Collection

For each manipulation task, we first use task-specific motion plans to collect 3,200 successful trajectories. These demonstrations are used only to fine-tune $\pi_{0.5}$ [29] and are separate from the progress-estimator training and evaluation trajectories. The resulting policy is then rolled out to collect unsuccessful executions.

On tasks where the policy exhibits limited failure diversity, a motion plan first advances the task to one of several predefined stages. Control is then handed to the learned policy for the remainder of the rollout, and unsuccessful executions are retained. Varying the takeover stage broadens the states from which the policy acts. Failures are generated by the learned policy after takeover rather than injected as scripted errors. However, these rollouts are conditioned on motion-plan-generated starting states, so "natural failure" does not imply an unbiased sample of deployment failures.

C Model and Training Setup

Architecture and objectives. The estimator follows RoboMeter [14] with the Qwen3-VL-4B-Instruct backbone and separate progress, preference, and success heads. The task configuration enables all three objectives and uses ten progress bins. Continuous progress targets are distributed between adjacent bin centers, with all probability assigned to a center when the target equals it. The progress head minimizes the cross-entropy in Equation 5; its probability-weighted bin centers yield a scalar in $[0, 1]$.

Training uses preference pairs and jointly supervises progress and success in the same forward pass. The total loss is the sum of preference BCE, progress cross-entropy, and success BCE, with unit outer coefficients. The success term uses class-balancing weights internally. A preference-only sample-type ratio therefore does not disable progress or success supervision. When a pair's rejected trajectory is a labeled failure, the configuration selects it for progress supervision with probability 0.5; otherwise the chosen trajectory is selected for that eligible pair. This selection is mapped to the pair's actual A/B ordering.

Progress reduction. For an active A- or B-side batch with B trajectories, T predicted temporal positions, and supervision mask m_{bt}, the trainer computes

$$\mathcal{R}(\ell, m) = \frac{\sum_{b=1}^{B} \frac{1}{T} \sum_{t=1}^{T} m_{bt} \ell_{bt}^{\text{prog}}}{\sum_{b=1}^{B} \sum_{t=1}^{T} m_{bt} + 10^{-8}}. \tag{11}$$

The progress term sums the active A/B contributions. Masks and targets are aligned with the model's temporal downsampling; T counts predicted positions rather than raw video frames. This is the implemented masked reduction, rather than a sum over all frames averaged only over trajectories.

Preference and success supervision. Preference BCE is averaged over sampled pairs. The independent success head predicts a sigmoid probability and uses class-balanced BCE on eligible frames of trajectory A. Labeled failed trajectories receive zero success targets; successful trajectories use the configured progress cutoff, including equality at 1.0. This success objective is distinct from the downstream progress-sequence detection heuristic in Section 4.4.

Reference-label calibration. For evaluations requiring frame-level labels, the automatic pipeline is followed by manual calibration of ambiguous cases. This process can correct individual errors but does not make the resulting annotations independent of the pipeline. Intrinsic MSE and rank correlation should be interpreted accordingly. No independent annotation agreement or component-level label-validation result is reported here.

D Additional Experimental Results

The following plots show the existing checkpoint evaluations for detection and filtering. Dots represent reported measurements; smooth curves provide visual guidance only. Ringed values identify selected observed scores, which can occur at different checkpoints. These curves are not multi-seed confidence intervals and should not be used to infer unmeasured behavior between points.

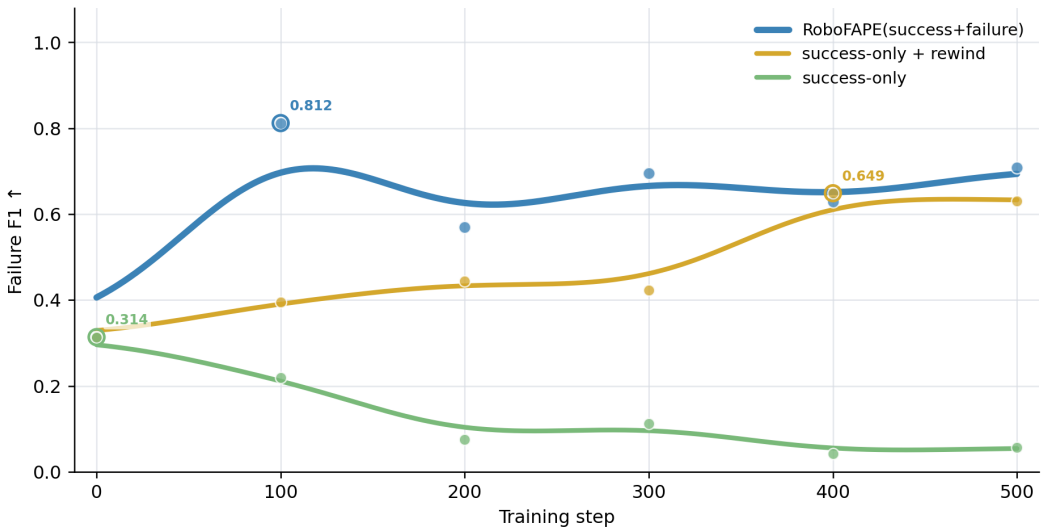


Figure 8: IND Failure F1 versus estimator training step. Dots show reported measurements and ringed values identify selected best observed scores. Smooth lines only guide the eye and do not represent additional evaluations or uncertainty estimates.

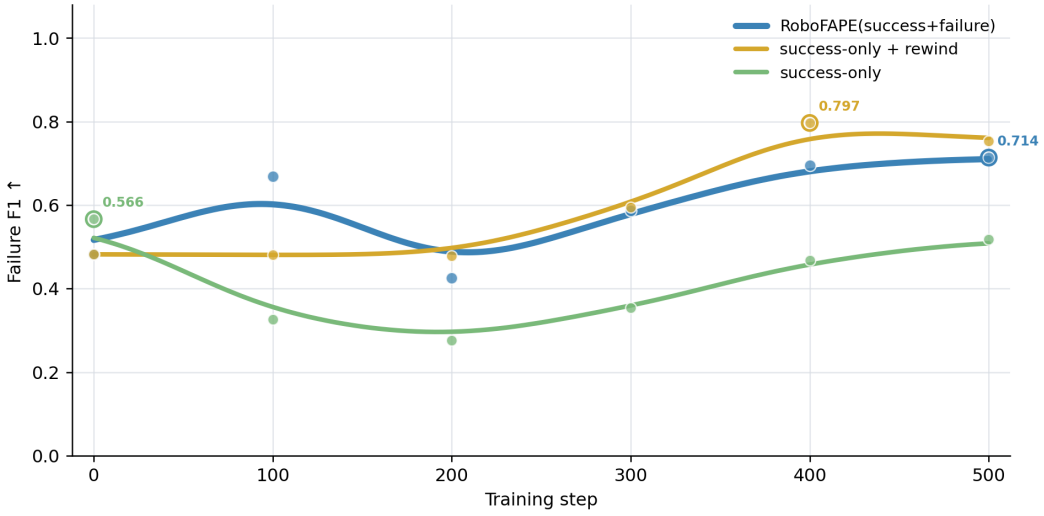


Figure 9: OOD Failure F1 versus estimator training step. As in Figure 8, dots are measurements and ringed scores may occur at different checkpoints. Smooth lines are visual guides; model comparisons should use measured points.

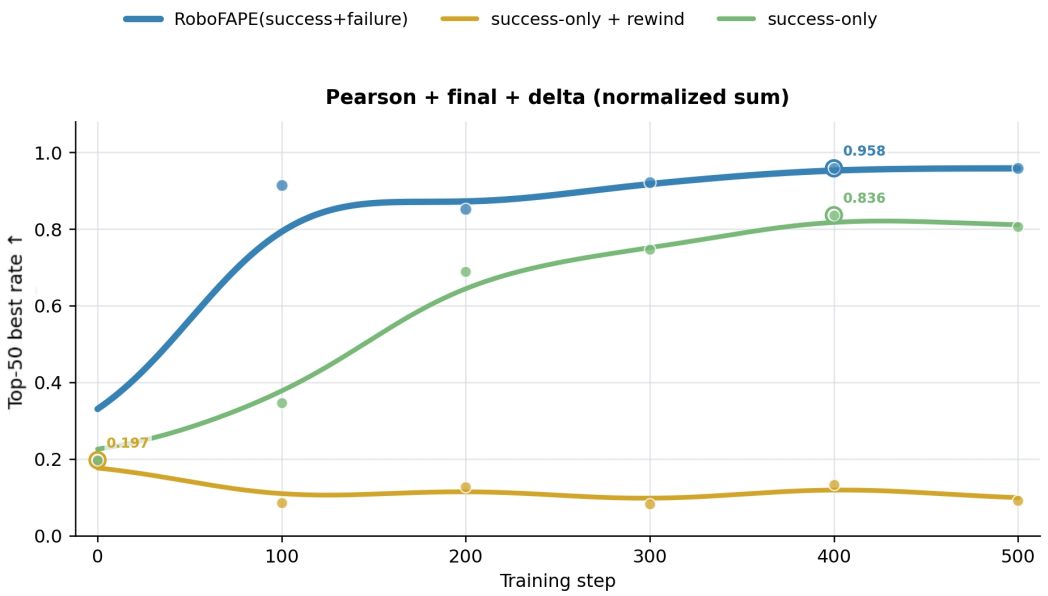


Figure 10: IND top-50 retrieval correctness using normalized progress–time correlation, final progress, and net progress change. Higher is better. Dots are measurements; fitted lines are visual guides and do not represent additional evaluation checkpoints.

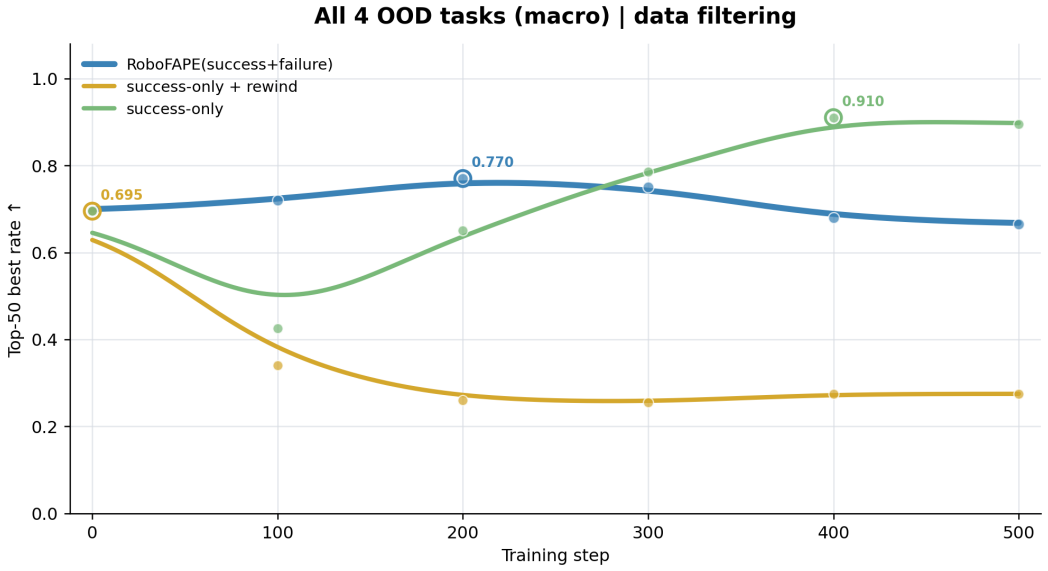


Figure 11: OOD top-50 retrieval correctness under the same scoring rule as Figure 10. The ringed scores are best observed values selected separately for each variant; they are not results at a shared checkpoint.