ROBOMETER: Scaling General-Purpose Robotic Reward Models via Trajectory Comparisons

Anonymous Submission

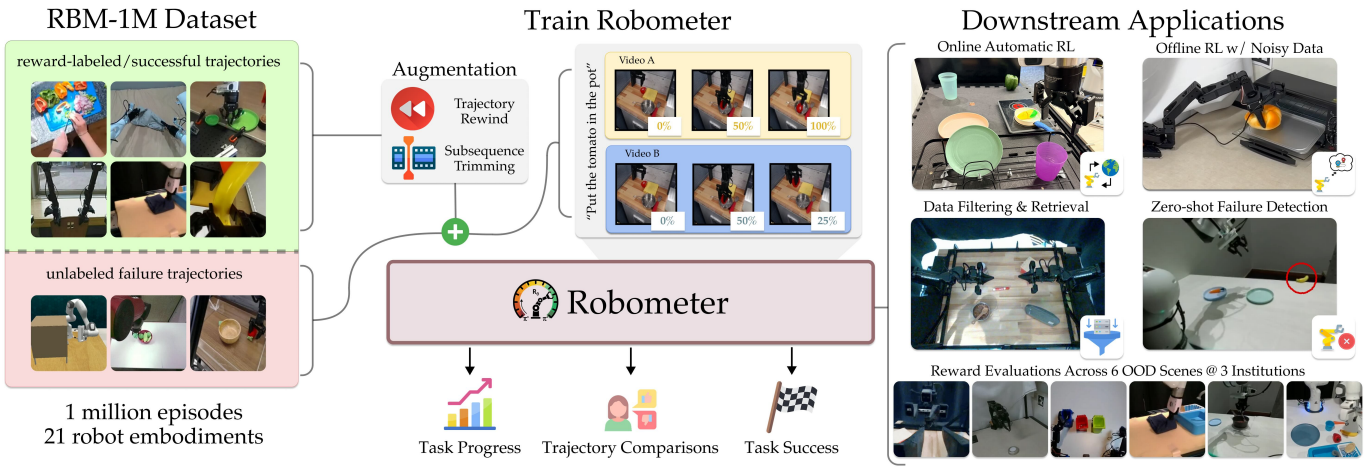


Fig. 1: **ROBOMETER Overview.** ROBOMETER is trained on RBM-1M, a 1M-trajectory dataset spanning 21 robot embodiments, containing both reward-labeled/expert trajectories and reward-unlabeled, failed trajectories. The model is supervised with a dual objective: predicting frame-level task progress (reward) and learning trajectory-level preferences from pairwise comparisons. To help with downstream RL, it is also trained to predict per-frame task success. This training recipe enables scalable reward learning, is validated on reward model evaluations from 6 out-of-distribution scenes collected at 3 institutions, and supports diverse downstream applications such as offline & online RL, imitation learning data filtering and retrieval, and automated failure detection.

***Abstract*—Current general-purpose robot reward models rely on frame-level progress labels from expert demonstrations. This approach scales poorly to large datasets, where suboptimal or failed trajectories are abundant and absolute progress is ambiguous. To address this, we introduce ROBOMETER, a scalable reward modeling framework combining intra-trajectory progress supervision with inter-trajectory preference supervision. ROBOMETER utilizes a dual objective: a frame-level loss that anchors reward magnitude to expert data, and a trajectory-comparison preference loss that imposes global ordering constraints. This enables effective learning from both successful and failed trajectories. To support this formulation at scale, we curate RBM-1M, a dataset of over one million multi-embodiment trajectories containing extensive suboptimal and failure data. Across benchmarks and real-world evaluations, ROBOMETER learns highly generalizable reward functions and improves downstream robot learning performance. Code, model weights, and videos at https://anon-robometer.github.io/.**

I. INTRODUCTION

In human cognition, comparative judgments are a core mechanism for internalizing calibrated scales [1, 2, 3], enabling reasoning about *relative* progress and outcomes rather than isolated states. Analogously, the supervision signals used to train robotic reward models determine how well they internalize notions of task progress, enabling downstream applications such as online reinforcement learning (RL) [4, 5], imitation learning (IL) from noisy data [6, 7], automated failure detection [8], and offline RL [9]. However, current general-purpose reward models rely exclusively on *absolute* progress labels derived from expert or reward-labeled demonstrations, providing pointwise, *trajectory-local* supervision [5, 10, 11, 12].

Such reward labels are easy to obtain for expert trajectories—for example, by linearly interpolating progress from 0 to 1—but become ill-defined and costly to annotate for failed attempts, where progress may fluctuate over time. As a result, large amounts of suboptimal data—ubiquitous in real-world robot learning—cannot be effectively leveraged [13]. This reliance on trajectory-*local* progress supervision limits both scalability and generalization. In this work, we address this limitation by training reward models with an additional *global* supervision signal that improves generalization across embodiments, scenes, and varying trajectory quality.

Our key insight is that *preference prediction* over trajectory pairs provides a complementary form of supervision. While progress labels anchor reward values along individual trajectories, pairwise comparisons impose ordering constraints across diverse trajectories, tasks, robots, and viewpoints. This formulation enables learning from previously unusable suboptimal data by requiring only relative comparisons—curated without additional human annotation—rather than absolute scores. Specifically, trajectory comparison supervision (1) enforces consistent ordering across trajectories, providing global grounding beyond individual rollouts, and (2) scales naturally to unlabeled failed trajectories where absolute progress is ambiguous, resulting in better-calibrated rewards.

To instantiate our key insight, we propose ROBOMETER, a general-purpose, video-language-input, dense reward model

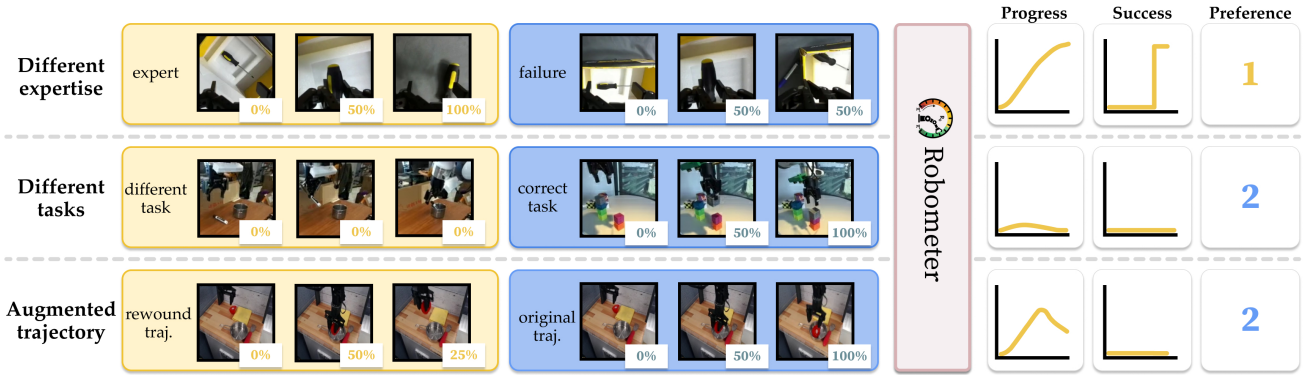


Fig. 2: **ROBOMETER** is a VLM-based reward model, that predicts dense, per-frame progress-based rewards and success labels for the first of two video trajectories. To be able to train with failed, non-expert data, we also predict which of the two video trajectories better completes the task. We use three strategies for curating training examples from our given datasets, which are further detailed in Section II-D.

trained with a dual reward-prediction objective: a frame-level progress loss on expert data and a preference-prediction loss over trajectory comparisons (see Figure 3). We train ROBOMETER on RBM-1M, a large-scale reward-learning dataset which we curate, that contains over one million trajectories collected from 21 robot platforms, including bimanual, single-arm, and mobile manipulators, as well as human demonstrations. Importantly, RBM-1M is intentionally constructed to include a substantial number of suboptimal and failed trajectories that naturally arise during real-world data collection but are difficult to exploit with absolute progress-based supervision. In addition to training with real failure data, we generate preference pairs using a suite of augmentations—including video rewinding [5], sequence trimming, and cross-task comparisons—that expose the model to diverse successful and suboptimal behaviors. Across external benchmarks and our own evaluation trajectories collected from six out-of-distribution scenes from three institutions, ROBOMETER outperforms state-of-the-art baselines by an average of 14% in reward rank correlation and 32% relative improvement in distinguishing suboptimal from successful trajectories.

Finally, we show that ROBOMETER outperforms relevant baselines in real-world robot learning applications across a diverse set of downstream applications that span different learning paradigms: (1) automatic online RL, (2) offline RL with noisy and expert trajectories, (3) dataset filtering for imitation learning, and (4) zero-shot failure detection across multiple robot embodiments and institutions. Overall, policy learning experiments with ROBOMETER demonstrate $\mathbf{2.4-4.5}\times$ higher success rates than the best baseline in each category. We publicly release the ROBOMETER model, the RBM-1M dataset, and code at https://anon-robometer.github.io.

II. ROBOMETER

We propose ROBOMETER, a large-scale reward model that provides dense reward feedback for robot learning. Our approach rests on three pillars: a diverse 1M-trajectory dataset (RBM-1M) which includes unlabeled failure trajectories, a pre-trained VLM backbone for cross-task generalization, and a hybrid training objective that combines **dense, per-frame progress** with **global trajectory preferences**. For brevity, we omit a related works section.

A. RBM-1M Dataset

Notation. We define the dataset $\mathcal{D} = \{\tau_i\}$ of trajectories, where each $\tau = \{o_{1:T}, l, p\}$ contains image observations o, a language instruction l, and a scalar progress label $p \in [0, 1]$ corresponding to the progress at the end of the trajectory.

Data Composition. Rather than maximizing trajectory quantity, RBM-1M focuses on viewpoint, scene, and embodiment diversity. We aggregate 1 million trajectories from: (1) **Expert robot data** from diverse, multi-robot sources such as Open-X [14] and subsets of high-quality, single-robot data such as AGIBotWorld [15]; (2) **Human videos** from datasets such as Epic-Kitchens [16] for scene diversity or human-robot paired datasets like RH20T [17] to promote embodiment-invariant representations; (3) **Simulation** data from sources like LIBERO [18]; and (4) **Failed trajectories** from automated policy rollouts [19] and failure-detection datasets [20]. Our dataset overall includes 21 robot embodiments and over 1 million trajectories, hence RBM-1M.

B. ROBOMETER Architecture and Tokenization

ROBOMETER instantiates a causally masked VLM, QWEN3-VL-4B-INSTRUCT, to process either one video (for reward inference) or a pair of videos (for preference training).

Hidden Embedding Extraction. To extract rewards without disrupting the VLM's pre-trained internal representations, we insert new, learned tokens into the sequence. We interleave **progress tokens** ($\langle|\text{prog\_token}|\rangle$) within the first video sequence and a single **preference token** ($\langle|\text{pref\_token}|\rangle$) at the end of the multi-video prompt:

$$\begin{aligned}\text{Tok}(l, o^1, o^2) \rightarrow \text{Tok}(l)\langle|\text{video\_start}|\rangle \left[\text{Tok}(o_t^1)\langle|\text{prog\_token}|\rangle\right]_{t=1}^{T} \\ \langle|\text{split\_token}|\rangle \left[\text{Tok}(o_t^2)\right]_{t=1}^{T} \langle|\text{pref\_token}|\rangle,\end{aligned} \quad (1)$$

where $\langle|\text{video\_start}|\rangle$ is the model's default image-start delimiter and $\langle|\text{split\_token}|\rangle$ is a separator. The causal mask ensures that $\langle|\text{prog\_token}|\rangle$ tokens attend only to the current and previous frames of o^1, producing dense, frame-level progress estimates for online reward inference, while $\langle|\text{pref\_token}|\rangle$ attends to both trajectories to make a relative judgment. We fix

both trajectories to length T to avoid preference predictions that rely on trajectory length as a proxy for quality. Progress tokens are inserted only for o^1 since at inference time, progress is predicted for a single trajectory; furthermore, if we insert progress tokens between o^2 frames, they would attend to o^1.

C. Training Objectives

We optimize ROBOMETER using a composite loss: $\mathcal{L} = \mathcal{L}_{\text{pref}} + \mathcal{L}_{\text{prog}} + \mathcal{L}_{\text{succ}}$. This allows the model to anchor rewards to absolute progress while learning to distinguish subtle quality differences through trajectory comparisons across the dataset.

Preference Prediction. We train a binary classifier, MLP_{pref}, on the hidden state $h_{\langle|\text{pref\_token}|\rangle}$ of the $\langle|\text{pref\_token}|\rangle$ to predict which trajectory better satisfies l:

$$\mathcal{L}_{\text{pref}} = -\Big[\mathbb{I}_{y=1}\ \log \sigma\big(\text{MLP}_{\text{pref}}(h_{\langle|\text{pref\_token}|\rangle})\big) + \mathbb{I}_{y=2}\ \log\big(1-\sigma\big(\text{MLP}_{\text{pref}}(h_{\langle|\text{pref\_token}|\rangle})\big)\big)\Big], \tag{2}$$

where y is the ground-truth preferred trajectory.

Progress and Success. For the first trajectory o^1, we attach an MLP head to each $h_{\langle|\text{prog\_token}|\rangle,t}$ to predict continuous progress p_t and binary success s_t. Similar to prior work [5, 7, 10], we define per-frame progress targets $p_{1:T}$ for expert demonstration data, where the final target progress $p = 1$. To better model multi-modal reward/progress distributions than continuous progress prediction, we discretize progress into $N = 10$ uniformly spaced bins over $[0, 1]$ and model progress prediction as a categorical distribution, following the C51 formulation [21]. For a trajectory of length T, the ground-truth continuous progress target at frame t is defined as $p_t = t/T$ for $t \in \{1, \dots, T\}$. This scalar target is projected onto a categorical distribution over N bins using linear interpolation between neighboring bin centers. The progress head $\text{MLP}_{\text{progress}}$ outputs a categorical distribution $\hat{p}_t \in \Delta^N$, and the progress loss is computed using cross-entropy:

$$\mathcal{L}_{\text{prog}} = \frac{1}{T}\sum_{t=1}^{T} \text{CE}\big(\text{Proj}(p_t),\ \text{MLP}_{\text{progress}}(h_{\langle|\text{prog\_token}|\rangle,t})\big)\,.$$

At inference time, a continuous progress estimate is recovered by taking the expectation over the bin centers, $\hat{p}_t = \sum_{i=1}^{N} z_i\, \hat{p}_{t,i}$, where $\{z_i\}_{i=1}^{N}$ denote the fixed bin centers. Per-frame success targets are defined such that $s_t = 0$ for $t < T$ and $s_t = 1$ for $t = T$. We train success prediction with binary cross-entropy on s, with balanced class weights adjusted per-batch to account for negative sample imbalance:

$$\mathcal{L}_{\text{succ}} = \text{BalancedBCE}(s_{1:T}, [\text{MLP}_{\text{success}}(h_{\langle|\text{prog\_token}|\rangle,t})]_{1:T}).$$

D. Data Sampling and Augmentation

Given these losses, the ideal training regime for ROBOMETER would rely on large-scale, preference-labeled robot trajectory datasets containing explicit progress-labeled failures. Such failures are particularly important because, at deployment time, reward models are frequently queried on out-of-distribution trajectories. In practice, however, preference annotations over robot trajectories are limited, and dense per-frame progress labels for failed executions are difficult to obtain. We address this limitation by constructing training inputs (l, o^1, o^2) and targets y dynamically from `RBM-1M` using three complementary strategies displayed in Figure 2:

1) **Progress-Based Comparisons (Different Expertise).** To teach the model to distinguish execution quality, we sample two trajectories τ_1, τ_2 sharing an instruction l but differing in outcome (e.g., an expert demonstration p=1 vs. an unlabeled failure p=`None`) or progress ($p^1 \neq p^2$). We set the preference target y=1 if $p^1 > p^2$ (or if τ_1 is the expert), and $y = 2$ otherwise. This allows ROBOMETER to leverage unlabeled failures by contrasting them against successful demonstrations.
2) **Instruction Negatives (Different Tasks).** To ensure rewards are grounded in the language command, we sample τ_1 and τ_2 with distinct instructions $l^1 \neq l^2$. We randomly select one instruction as the conditioning text l, set the preference label y to the trajectory corresponding to the selected instruction, and set the progress target p_t=0 for the other, enforcing that correct behavior for the wrong task yields no reward.
3) **Video Rewind (Augmented Failures).** To explicitly model "undoing" progress—a common failure mode in RL—we generate synthetic preferences from a single expert trajectory τ by reversing a segment of time. Prior work denotes this type of augmentation as *video rewind* [5, 7, 22]. We sample indices $1 \leq t_1 < t_2 < t_3 \leq T$ to form a *Chosen* forward sequence $o^c = o_{t_1:t_3}$ and a *Rejected* rewound sequence, either $o^r = [o_{t_1:t_3}, o_{t_3-1:t_2}]$ or $[o_{t_3:t_1}]$. The Chosen and Rejected sequences are randomly assigned to o^1 and o^2 to construct the preference label. While o^c targets linear forward progress, we explicitly penalize the reversal in o^r by assigning decreasing progress targets matched to their frame indices.

III. EXPERIMENTS

Our experiments aim to study ROBOMETER's effectiveness in producing rewards for robot learning. Specifically, we organize our experiments to answer the following questions:
(**Q1**) **Reward Evaluation**: How well do ROBOMETER rewards reflect task progress on *unseen* tasks and embodiments?
(**Q2**) **Ablation + Analysis**: How much does each component of ROBOMETER contribute to reward performance?
(**Q3**) **Policy Learning**: How does ROBOMETER compare against baselines in enabling downstream robot learning?

Baselines. We compare ROBOMETER against a strong set of video-language input, zero-shot-capable, and open-sourced or API-accessible reward baselines.

- **VLAC-8B** [10] trains a VLA that predicts actions and rewards on a dataset of 300k human and robot trajectories. We compare against their larger 8B parameter checkpoint.
- **GVL** [6] prompts a pre-trained closed-source LLM with shuffled video frames to predict task progress for subsampled frames across the video sequence. We use GPT-5 mini

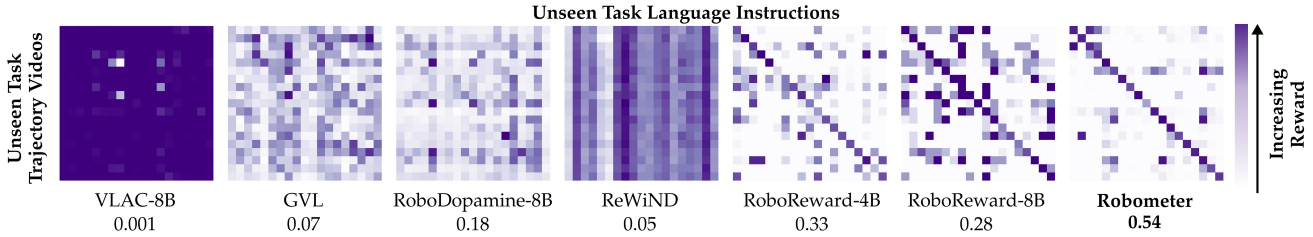


Fig. 3: **Video-Language Reward Confusion Matrix.** For each task sampled at random from *self-collected, unseen* data from RBM-EVAL-OOD, we compute rewards for all combinations of demonstration videos and language descriptions. ROBOMETER produces the most diagonal-heavy confusion matrix, indicating strong alignment between unseen demos and instructions. We also report the column-normalized diagonal mean under each model, which represents the fraction of the model's total reward for aligned task and video pairs.

		Baselines			w/ RoboReward Training Data			w/ our RBM-1M data	
		GVL	VLAC	RoboDopamine-8B	RoboReward-4B	RoboReward-8B	ROBOMETER	ReWiND	ROBOMETER
(a) VOC $r \uparrow$	RBM-EVAL-ID	0.16	0.16	0.75	0.77	0.82	0.84	0.46	**0.92**
	RBM-EVAL-OOD	0.21	0.17	0.74	0.88	0.88	0.93	0.51	**0.95**
(b) Kendall $\tau_a \uparrow$	RBM-EVAL-OOD	0.19	0.08	0.39	0.50	0.47	0.55	0.01	**0.66**

TABLE I: (a) Reward alignment (VOC Pearson r) and (b) trajectory ranking (Kendall τ_a) on RBM-EVAL datasets. We compare ROBOMETER against RoboReward-4B/8B with their own training data, and we also evaluate ReWiND and ROBOMETER trained with the full RBM-1M dataset. Kendall τ_a is not calculated for RBM-EVAL-ID due to it only having simulation failure data.

as it is the best-performing closed-source model on the RoboRewardBench reward evaluation benchmark [11].

- **ReWiND** [5] trains a small transformer-based network with a direct progress prediction objective along with video rewinding to simulate failed policy rollouts. We train ReWiND with RBM-1M to maximize its zero-shot capability.
- **RoboDopamine-8B** [12] fine-tunes a VLM for reward prediction via "frame hops" comparing forward and rewound frames. Although it is designed for reward prediction conditioned on a *goal image* and instruction, we evaluate in our zero-shot setting without a goal image for fair comparison.
- **RoboReward-4B/8B** [11]: Fine-tunes a Qwen-3-VL 4B/8B VLM for discrete (1-5) progress prediction on a dataset consisting of data from OXE [14] and RoboArena evaluations [23]. Generates *counterfactual* instructions via closed-source VLMs to simulate failed trajectories.

Custom Evaluation Datasets. We train ROBOMETER, and certain baselines when applicable, on the aforementioned RBM-1M dataset. We collect our own evaluation dataset, RBM-EVAL-OOD, consisting of 976 trajectories collected from 3 academic institutions that are guaranteed not to be in the training data of *any* baseline. We also aggregate an in-distribution, unseen validation split of unseen trajectories collected from datasets in RBM-1M, denoted RBM-EVAL-ID.

Q1: Reward Evaluation

As detailed in Section II-B, we aim to train a reward model that (1) generalizes across tasks, embodiments, and domains, and (2) provides reward signals useful for *policy learning*.

Trajectory Task Alignment. ROBOMETER accurately distinguishes among tasks in RBM-EVAL-OOD, indicating that its rewards remain aligned with task semantics under unseen embodiments, viewpoints, and scenes. Figure 3 shows confusion matrices between successful OOD trajectory videos and language instructions. An ideal model has a strong **purple diagonal** and white off-diagonal entries. ROBOMETER shows the clearest separation, discriminating the *correct* tasks.

Model Type	Model	RoboRewardBench MAE (↓)
Qwen3-4B Models	ROBOMETER	**0.72**
	ROBOMETER (only RoboReward data)	0.75
	RoboReward-4B	0.85
	Qwen3-VL-4B-Instr.	1.03
Closed / Larger	RoboReward-8B	**0.67**
	GPT-5-mini	0.69
	Qwen3-VL-8B-Instr.	0.89
	Gemini-2.5-pro	0.90

TABLE II: Evaluation on the RoboRewardBench benchmark [11].

Reward Alignment. We next evaluate whether models assign increasing rewards along *successful* trajectories from RBM-EVAL-OOD and RBM-EVAL-ID in Table I(a). We report Value Order Correlation (VOC) [6] $\in [-1, 1]$, the Pearson correlation between predicted frame rewards and ground-truth timestep values. ROBOMETER performs best on both benchmarks, with especially large gains on RBM-EVAL-OOD.

RoboRewardBench Evaluation. We also evaluate on the external RoboRewardBench benchmark [11], reporting Mean Absolute Error (MAE) on discretized 1–5 reward scores; results are shown in Table II. For a fair comparison, we train a ROBOMETER variant using only RoboReward data with matching 5-bin outputs.

This RoboReward-only variant achieves 0.75 MAE, outperforming RoboReward-4B, while the full model improves further to 0.72, trailing only the much larger RoboReward-8B and GPT-5-mini. Since RoboReward-8B and 4B perform similarly on our OOD evaluations in Table I, while GPT-5-mini performs much worse, we attribute stronger RoboRewardBench results for the larger models primarily to discrete 5-way labeling and final-frame-only evaluation.

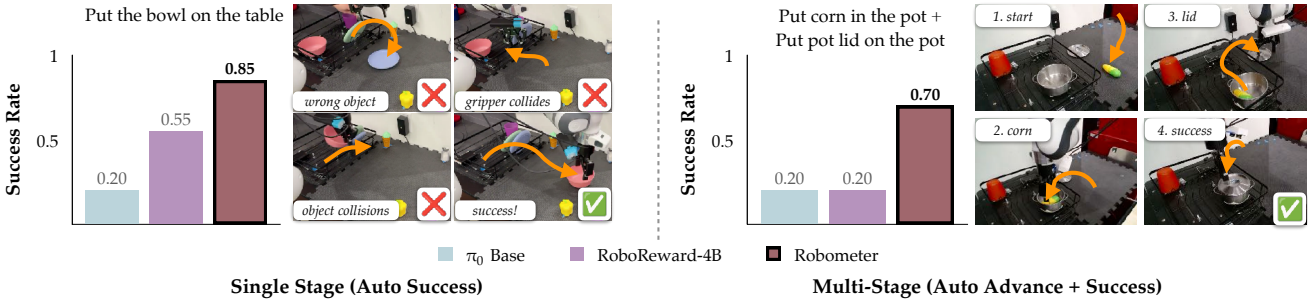


Fig. 4: **Automatic online RL** with DSRL on a DROID setup with ROBOMETER improves π_0 from 20% to 85% on a single-stage task and 20% to 70% on a two-stage task, outperforming RoboReward's overall success rate by 2.5×. The setup is "automatic" because success detection and stage advancement are handled automatically by the reward model, requiring human intervention only for physical scene resets.

	(a) LIBERO-90			(b) RBM-EVAL-OOD		
Ablation	VOC r	Kendall τ	Suc − Fail	VOC r	Kendall τ	Suc − Fail
H1 **Prog. Only**	0.96	0.63	0.11	0.93	0.31	0.08
H1 **+Preference**	0.90	0.74	0.22	**0.95**	0.54	0.24
H2 **+Failed Data**	**0.98**	**0.92**	**0.46**	**0.95**	**0.66**	**0.33**
H3 **ReWiND Arch.**	0.48	-0.14	-0.02	0.51	0.01	0.02

TABLE III: Reward alignment (VOC Pearson r), policy ranking (Kendall τ), and average reward difference between successful and failed trajectories on LIBERO-90 and RBM-EVAL-OOD.

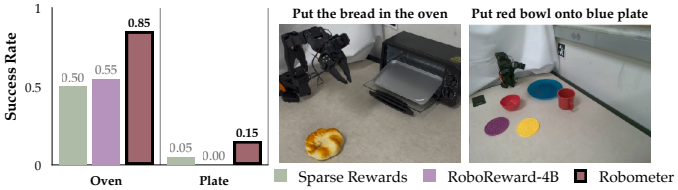


Fig. 5: **Offline RL** results using IQL on a mixture of Noisy and Expert trajectories. ROBOMETER rewards consistently outperform both RoboReward and sparse rewards: 2.4× average success rate improvement over the best baseline for each task.

Relative Trajectory Rankings for Mixed Expertise Data. Next, we quantitatively demonstrate that ROBOMETER is more effective than baselines at providing rewards useful for *policy learning*. For a robot policy to learn with rewards, the rewards should not only be high when performing the correct task, but also be low for incorrect execution. We measure this using the Kendall-τ_a coefficient [24], an ordering metric $\in [-1, 1]$ robust to ties. We calculate the alignment between model-assigned final rewards and the ground-truth ordering between failed, suboptimal, and successful trajectories for the same task. A higher τ_a value demonstrates that the reward model more accurately distinguishes between levels of policy performance and thus can provide proper reward signals to the policy for both low- and high-quality behaviors. We report results in Table I(b). On RBM-EVAL-OOD, ROBOMETER achieves a Kendall-τ_a of 0.66, substantially outperforming RoboReward-4B (0.50) and RoboReward-8B (0.47), indicating that ROBOMETER more reliably recovers the correct ordering among failed, suboptimal, and successful trajectories.

Q2: Ablations: Why does ROBOMETER *Perform so Well?*

Here, we investigate individual components of ROBOMETER to evaluate specific hypotheses about reward model training and its effects on downstream RL performance.

- **H1** Predicting **preferences** (Equation (2)), even without paired failure trajectories, improves reward performance.
- **H2** Scaling preference prediction with additional **failure data** leads to improved reward model performance.
- **H3** Fine-tuning from **pre-trained VLMs** helps with reward predictions on unseen tasks.

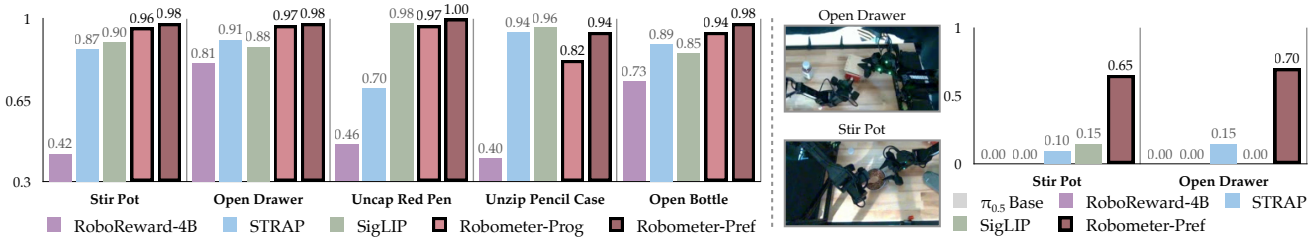
Reward Model Ablations. To test H1, we train ROBOMETER with only progress prediction and with both progress and preference prediction on the 1,709-demo dataset without failed trajectories. To test H2, we add 1,929 generated failed LIBERO trajectories and train ROBOMETER with the full objective. Finally, to test H3, we replace the pretrained VLM with a 500M-parameter variant of ReWiND's transformer, trained with the same objectives on the same LIBERO dataset.

We report results on LIBERO and, separately, models trained on the full RBM-1M and evaluated on RBM-EVAL-OOD (with failed data removed for +Preference and +Failed Data) in Table III. Compared to **H1 Prog. Only**, adding preference supervision (**H1 +Preference**) improves ranking, raising Kendall-τ_a from 0.63 to 0.74 on LIBERO-90 and from 0.31 to 0.54 on RBM-EVAL-OOD. Adding failed trajectories (**H2 +Failed Data**) yields the largest gains: on LIBERO-90, Kendall-τ rises to 0.92 and the final-reward gap between successful and failed trajectories increases by over 4× relative to progress-only training. Similar trends hold on RBM-EVAL-OOD. In contrast, replacing the pretrained VLM backbone with a scaled ReWiND architecture (**H3 ReWiND Arch.**) sharply degrades performance across all metrics, highlighting the importance of large-scale multimodal pretraining. We omit RL experiment details on these LIBERO ablations for brevity, but ROBOMETER-LIBERO demonstrates **2-4**× better sample efficiency than sparse rewards, with better ablation results directly translating into better sample efficiency.

Q3: Accelerating Robot Learning with Generalizable Rewards

We evaluate whether ROBOMETER's dense and generalizable reward signals can be used **zero-shot** into improved downstream robot learning across four settings. Across all experiments, we compare against RoboReward-4B—the strongest baseline reward model in our offline evaluations.

Automatic Online RL. First, we evaluate ROBOMETER in an *automated* online RL setting by training DSRL [25] from



(a) Proportion of Task-Relevant Subtrajectories Retrieved **(b) Policy Success Rates**

Fig. 6: **(a): Proportion of task-relevant subtrajectories** out of 100 retrieval queries. Our method consistently retrieves a high number of relevant subtrajectories using either the preference or progress objective. **(b): Success rates** of LoRA-finetuned $\pi_{0.5}$ policies using the retrieved trajectories from each method. ROBOMETER-retrieval attains an average $4.5\times$ success rate improvement over the best baseline.

scratch on a π_0 base policy [26] pre-trained on DROID [27]. ROBOMETER enables autonomous RL by providing dense rewards and explicit *success predictions*, which we use to automate episode termination; manual human intervention is required only for scene resets. As shown in Figure 4 (left), DSRL+ROBOMETER improves success from 20% to 85% in ≤ 45 **minutes** (10k timesteps), outperforming RoboReward's 55%. RoboReward frequently assigns maximum rewards for unrelated tasks (e.g., picking up the wrong object), leading to premature resets and reinforcing incorrect behaviors.

Next, we evaluate a *longer-horizon multi-stage RL* setting in Figure 4 (right), where success predictions trigger progression between stages. Unlike methods that explicitly train with multi-stage rewards and thus require stage labels (e.g., REDS [28] or SARM [7]), we simply decompose tasks into stages at inference time using a pre-trained VLM and use ROBOMETER to advance stages automatically. In this setting, DSRL+ROBOMETER improves π_0's success from 20% to 70% over 10k timesteps, outperforming RoboReward's 20%.

Combining Noisy and Expert Data via Offline RL. We consider an offline RL setting with mixed-expertise data for two tasks on an SO-101 robot (SO-101 is not in `RBM-1M`), combining expert and noisy, suboptimal demos, as shown in Figure 5. We train policies with Implicit Q-Learning (IQL) [29] to study how dense rewards from ROBOMETER improve learning stability and policy extraction in offline RL.

Accurate, dense reward signals can provide informative intermediate feedback, reducing reliance on long-horizon credit assignment and enabling trajectory "stitching" with smaller discount factors γ, thereby reducing value function variance. For each of sparse reward, RoboReward, and ROBOMETER, we sweep $\gamma \in \{0.90, 0.95, 0.99\}$ and report the best-performing checkpoint. We observe that ROBOMETER, which provides dense, temporally aligned rewards, performs best at a lower discount factor $\gamma = 0.9$ and outperforms both RoboReward and sparse rewards across both tasks with a $\mathbf{2.4}\times$ success rate improvement over the best baseline in each.

Data Filtering & Retrieval. We next evaluate ROBOMETER as a mechanism for unsupervised data filtering and retrieval. Using a bimanual "play" dataset [30] of unannotated, multi-task trajectories collected on a Trossen AI setup (not in `RBM-1M`), we retrieve the top 100 subtrajectories for a given task instruction. We compare retrieval relevance against RoboReward, pre-trained SigLIP [31], and a retrieval-specific baseline, STRAP [32]. For ROBOMETER we retrieve subtrajectories using (i) the preference objective via pairwise trajectory comparisons, or (ii) the progress objective by computing per-timestep progress values and each trajectory's value–order correlation. As shown in Figure 6(a), ROBOMETER consistently achieves higher retrieval relevance across five tasks. Finally, we LoRA-finetune $\pi_{0.5}$ [33, 34, 35] on these retrieved segments. Policies trained on ROBOMETER-filtered data averages a $\mathbf{4.5}\times$ higher success rate than those using baseline-retrieved data on `Stir the Pot` and `Open the Red Drawer` (Figure 6(b)).

Task	T.U.	VLAC	GPT-5-mini	RoboReward-4B	ROBOMETER
move banana	0.53	0.45	0.48	0.91	**0.94**
move mouse	0.50	0.00	0.89	0.80	**0.91**
pour pebble	0.32	0.00	0.25	0.73	**0.83**
fold towel	**0.58**	0.16	0.27	0.40	**0.58**
pull tissue	0.43	0.00	0.00	0.57	**0.76**
put spoon	0.22	0.00	0.25	**0.73**	**0.73**
stir pot	0.47	0.00	0.17	**0.95**	0.90
Average	0.48	0.16	0.33	0.74	**0.81**

TABLE IV: **Failure detection performance**. Our method achieves the highest average F1 score across tasks. T.U. stands for the token-uncertainty baseline.

Failure Detection. We evaluate ROBOMETER's zero-shot failure detection on 100 manipulation trajectories from a Franka Panda DROID robot (30 successful, 70 failed) spanning seven tasks collected in scenes unseen in `RBM-1M`. We compare our method against: the token-uncertainty [8] of π_0-FAST-DROID [36] as proposed by Gu et al. [8] for zero-shot failure detection, VLAC, which reports failure detection results in prior work, GPT-5-mini, and RoboReward-4B.

As shown in Table IV, ROBOMETER achieves the highest average F1 score, effectively balancing true positive and true negative rates (TPR and TNR). VLAC frequently flags trajectories as failures, achieving high TPR but low TNR. RoboReward-4B performs competitively but underperforms ROBOMETER on tasks with subtle failure modes such as *fold towel* and *pull tissue*. ROBOMETER robustly detects irreversible, insufficient-progress, and non-terminal failures (e.g., hovering, oscillation, or partial completion), fully zero-shot across tasks unlike prior methods that require task-specific thresholds, calibration, or test-time interaction [8, 37, 38].

References

[1] D. R. J. Laming, "The relativity of 'absolute' judgements," *British Journal of Mathematical and Statistical Psychology*, vol. 37, pp. 152–183, 1984.

[2] N. Stewart, G. D. A. Brown, and N. Chater, "Absolute identification by relative judgment." *Psychological review*, vol. 112 4, pp. 881–911, 2005.

[3] M. A. Sharif and D. M. Oppenheimer, "The effect of relative encoding on memory-based judgments," *Psychological Science*, vol. 27, no. 8, pp. 1136–1145, 2016.

[4] D. Yang, D. Tjia, J. Berg, D. Damen, P. Agrawal, and A. Gupta, "Rank2reward: Learning shaped reward functions from passive video," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[5] 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," in *Conference on Robot Learning (CoRL)*, 2025.

[6] Y. J. Ma, J. Hejna, A. Wahid, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani *et al.*, "Vision language models are in-context value learners," in *International Conference on Learning Representations (ICLR)*, 2025.

[7] Q. Chen, J. Yu, M. Schwager, P. Abbeel, F. Shentu, and P. Wu, "Sarm: Stage-aware reward modeling for long horizon robot manipulation," *arXiv preprint arXiv:2509.25358*, 2025.

[8] Q. Gu, Y. Ju, S. Sun, I. Gilitschenski, H. Nishimura, M. Itkina, and F. Shkurti, "SAFE: Multitask failure detection for vision-language-action models," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[9] S. Venkataraman, Y. Wang, Z. Wang, N. S. Ravie, Z. Erickson, and D. Held, "Real-world offline reinforcement learning from vision language model feedback," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[10] S. Zhai, Q. Zhang, T. Zhang, F. Huang, H. Zhang, M. Zhou, S. Zhang, L. Liu *et al.*, "A vision-language-action-critic model for robotic real-world reinforcement learning," *arXiv preprint arXiv:2509.15937*, 2025.

[11] T. Lee, A. Wagenmaker, K. Pertsch, P. Liang, S. Levine, and C. Finn, "Roboreward: General-purpose vision-language reward models for robotics," *arXiv preprint arXiv:2601.00675*, 2026.

[12] H. Tan, S. Chen, Y. Xu, Z. Wang, Y. Ji, C. Chi, Y. Lyu, Z. Zhao *et al.*, "Robo-dopamine: General process reward modeling for high-precision robotic manipulation," *arXiv preprint arXiv:2512.23703*, 2025.

[13] R. Tian, Y. Wu, and A. Bacjsy, "Position: Good embodied reward models need bad behavior data," Carnegie Mellon University, Tech. Rep., 2026. [Online]. Available: https://cmu-intentlab.github.io/pdf/tian_icml_26_position.pdf

[14] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee *et al.*, "Open X-Embodiment: Robotic learning datasets and RT-X models," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[15] A. W. C. contributors, "Agibot world colosseum," 2024.

[16] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, J. Ma, E. Kazakos, D. Moltisanti, J. Munro *et al.*, "Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100," *International Journal of Computer Vision (IJCV)*, vol. 130, p. 33–55, 2022.

[17] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu, "Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot," *arXiv preprint arXiv:2307.00595*, 2023.

[18] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *arXiv preprint arXiv:2306.03310*, 2023.

[19] Z. Zhou, P. Atreya, A. Lee, H. R. Walke, O. Mees, and S. Levine, "Autonomous improvement of instruction following skills via foundation models," in *Conference on Robot Learning (CoRL)*, 2024.

[20] Z. Lin, J. Duan, H. Fang, D. Fox, R. Krishna, C. Tan, and B. Wen, "Failsafe: Reasoning and recovery from failures in vision-language-action models," *arXiv preprint arXiv:2510.01642*, 2025.

[21] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2017.

[22] Y. Huang, S. Zou, J. Zhang, X. Liu, R. Hu, and K. Xu, "Adapower: Specializing world foundation models for predictive manipulation," *arXiv preprint arXiv:2512.035358*, 2025.

[23] P. Atreya, K. Pertsch, T. Lee, M. J. Kim, A. Jain, A. Kuramshin, C. Eppner, C. Neary *et al.*, "Roboarena: Distributed real-world evaluation of generalist robot policies," in *Conference on Robot Learning (CoRL)*, 2025.

[24] C. Muslimani, K. Johnstonbaugh, S. Chandramouli, S. Booth, W. B. Knox, and M. E. Taylor, "Towards improving reward design in RL: A reward alignment metric for RL practitioners," in *Reinforcement Learning Conference (RLC)*, 2025.

[25] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, "Steering your diffusion policy with latent space reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2025.

[26] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arxiv:2410.24164*, 2024.

[27] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," *arXiv preprint arXiv:2403.12945*, 2024.

[28] C. Kim, M. Heo, D. Lee, H. Lee, J. Shin, J. J. Lim,

and K. Lee, "Subtask-aware visual reward learning from segmented demonstrations," in *International Conference on Learning Representations (ICLR)*, 2025.

[29] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[30] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," *Conference on Robot Learning (CoRL)*, 2019.

[31] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *International Conference on Computer Vision (ICCV)*, 2023.

[32] M. Memmel, J. Berg, B. Chen, A. Gupta, and J. Francis, "Strap: Robot sub-trajectory retrieval for augmented policy learning," in *International Conference on Learning Representations (ICLR)*, 2025.

[33] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[34] Z. Liu, J. Zhang, K. Asadi, Y. Liu, D. Zhao, S. Sabach, and R. Fakoor, "TAIL: Task-specific adapters for imitation learning with large pretrained models," in *International Conference on Learning Representations (ICLR)*, 2024.

[35] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.

[36] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn *et al.*, "Fast: Efficient action tokenization for vision-language-action models," *arXiv preprint arXiv:2501.09747*, 2025.

[37] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. Ambrus *et al.*, "Can we detect failures without failure data? Uncertainty-aware runtime failure detection for imitation learning policies," in *Robotics: Science and Systems (RSS)*, 2025.

[38] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg, "Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress," in *Conference on Robot Learning (CoRL)*, 2025.