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

Anthony Liang*[1], Yigit Korkmaz*[1], Jiahui Zhang[2], Minyoung Hwang[3], Abrar Anwar[1], Sidhant Kaushik[4]
Aditya Shah[5], Alex S. Huang[2], Luke Zettlemoyer[5], Dieter Fox[5,6], Yu Xiang[2], Anqi Li[7]
Andreea Bobu[3], Abhishek Gupta[5], Stephen Tu†[1], Erdem Bıyık†[1], Jesse Zhang†[5]

[1]Univ. of Southern California [2]UT Dallas [3]MIT [4]Indep. Researcher [5]Univ. of Washington [6]Ai2 [7]NVIDIA
*Equal contribution †Equal advising

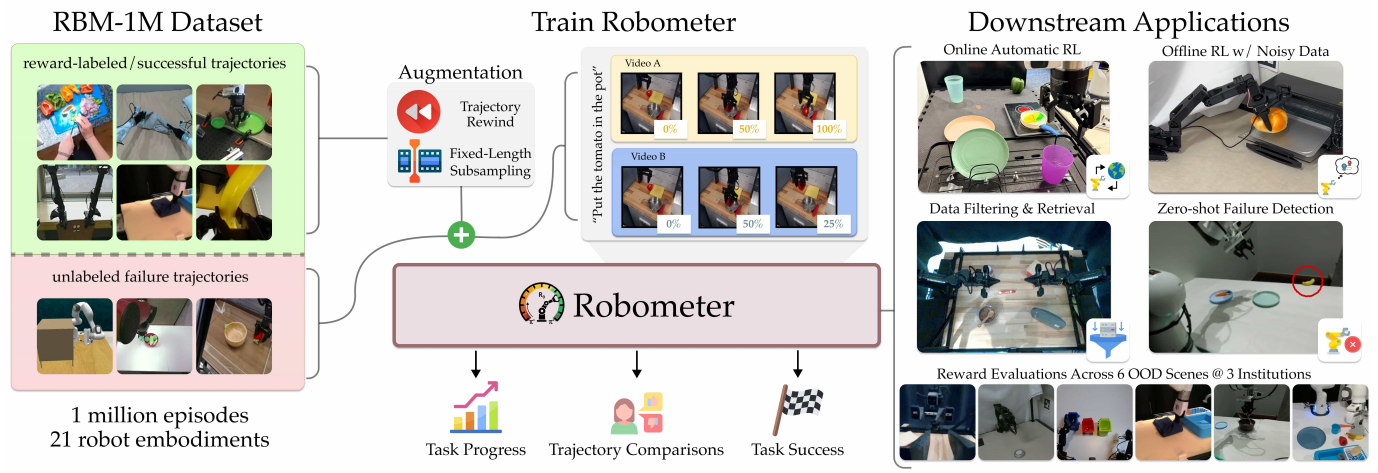

Fig. 1: **ROBOMETER Overview.** ROBOMETER is trained on RBM-1M, a 1M-trajectory dataset spanning 21 robot embodiments, containing both reward-labeled/expert trajectories and reward-unlabeled, failed trajectories. The model is supervised with a dual objective: predicting frame-level task progress (reward) 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://robometer.github.io/.**

I. INTRODUCTION

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]. Current general-purpose robot manipulation 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 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 complementary 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.

To instantiate this insight, we propose a scalable training

recipe for reward modeling based on a dual reward-prediction objective: a frame-level progress loss on expert demonstrations and a preference-prediction loss over trajectory comparisons. Using this recipe, we train ROBOMETER, a manipulation-centric reward model. Even when trained only on expert demonstrations, preference supervision improves ROBOMETER's ability to distinguish successful from suboptimal trajectories. Moreover, as additional unlabeled suboptimal data is introduced, ROBOMETER naturally scales to further improve performance.

To train ROBOMETER, we curate `RBM-1M`, a large-scale reward-learning dataset for robot manipulation containing over one million trajectories collected from 21 robot platforms, including single-arm, bimanual, and mobile manipulators, as well as human demonstrations (see Figure 3). Importantly, `RBM-1M` includes a substantial number of suboptimal and failed trajectories that naturally arise during real-world data collection but are difficult to leverage with purely absolute progress-based supervision. Beyond real-world failures, we further construct preference pairs using augmentations—including video rewinding [5] 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 spanning three institutions, ROBOMETER outperforms state-of-the-art baselines by an average of 14% in reward rank correlation and achieves a 32% relative improvement in distinguishing successful from suboptimal trajectories.

Finally, we show that ROBOMETER outperforms relevant baselines in real-world robot learning applications across a diverse set of downstream applications: (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 ROBOMETER, `RBM-1M`, and the training code at https://robometer.github.io/, enabling the community to train their own models using our recipe or to directly use ROBOMETER for robot manipulation applications.

II. SCALABLE REWARD MODEL TRAINING

We propose a *scalable recipe* for training reward models, along with ROBOMETER, an instantiation of this recipe that provides dense rewards for robot manipulation. Our approach rests on three pillars: a diverse dataset which includes unlabeled failure trajectories, a pre-trained VLM backbone for generalization, and a training objective that combines **per-frame rewards** with **trajectory preferences**.

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 $p \in \{\text{None}, [0, 1]\}$ corresponding to the progress at the end of the trajectory. For expert demos, $p = 1.0$; for datasets with partial progress labels (e.g., RoboArena [14]), we use the provided score. For unlabeled failed trajectories, we set $p = \texttt{None}$.

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 [15] and subsets of high-quality, single-robot datasets such as AGIBotWorld [16]; (2) **Human videos** from datasets such as Epic-Kitchens [17] for scene diversity or human-robot paired datasets like RH20T [18] to promote embodiment-invariant representations; (3) **Simulation** data from sources like LIBERO [19]; and (4) **Failed trajectories** from automated policy rollouts [20] and failure-detection datasets [21].

Our dataset overall includes 21 robot embodiments and over 1 million trajectories, hence `RBM-1M`.

B. ROBOMETER *Architecture and Tokenization*

ROBOMETER builds on 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:

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

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 current/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 $T = 8$ 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:

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

where y is the ground-truth preferred trajectory.

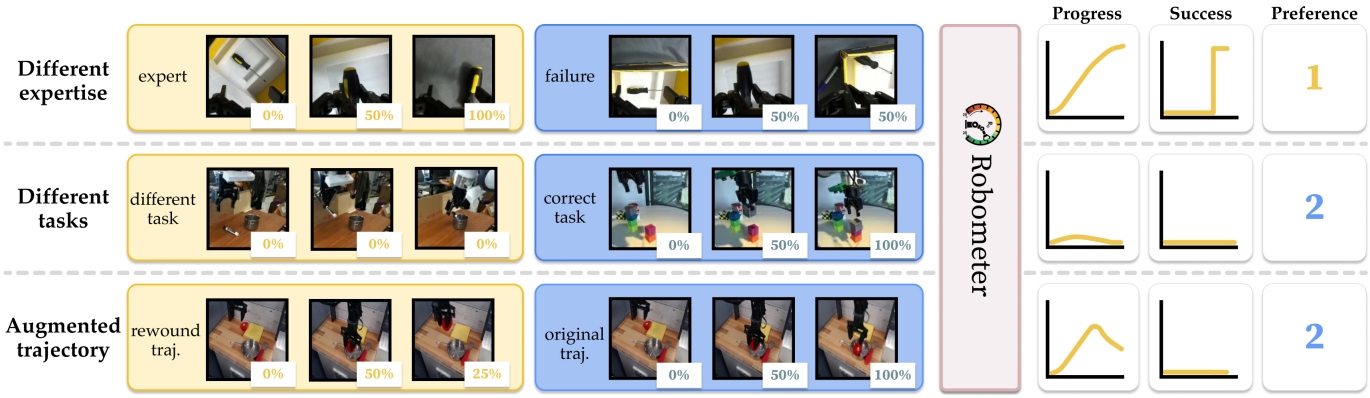

Fig. 2: **ROBOMETER** is a VLM-based reward model that predicts dense, per-frame progress-based rewards and success labels for the first of two video trajectories. To scalably train with failure data, we also predict which of the two video trajectories better completes the task (preference). We use three strategies for curating training examples from our given datasets, which are further detailed in Section II-D. At inference, the model can input either up to two trajectories: with a single trajectory, it outputs per-frame progress & success predictions; with two, it additionally predicts a preference for which trajectory best performs the task.

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 [22, 23, 24]. 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 the 2 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:

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

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—e.g., failures induced by online RL exploration, compounding execution errors, or noisy data collection—that deviate substantially from the training distribution. In practice, however, preference annotations over robot trajectories are limited, and dense per-frame progress labels for failed executions are expensive and 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, 25]. 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.

Fixed-length Subsampling. To avoid biasing toward longer trajectories, we construct fixed-length inputs by randomly selecting start and end indices in the trajectory, and uniformly

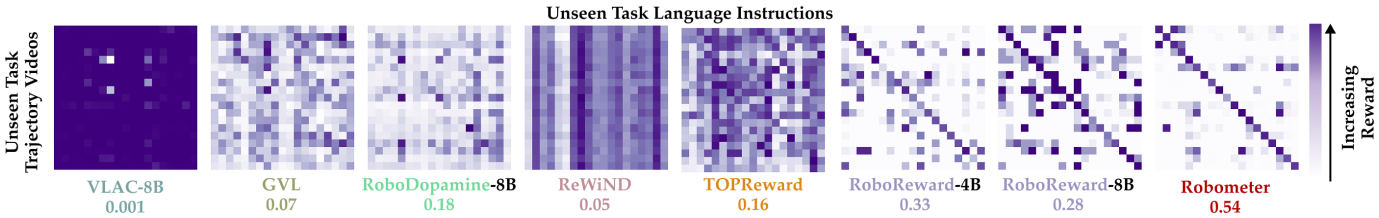

Fig. 3: **Video-Language Reward Confusion Matrix.** For each task sampled at random from *self-collected, unseen* data from RBM-EVAL-OOD, we compute rewards for all combinations of demonstration videos and language descriptions. ROBOMETER produces the most diagonal-heavy confusion matrix, indicating 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.

		w/ Varied Training Data				w/ RoboReward Training Data			w/ our RBM-1M data	
		GVL	VLAC	TOPReward	RoboDopamine-8B	RoboReward-4B	RoboReward-8B	ROBOMETER	ReWiND	ROBOMETER
(a) VOC $r \uparrow$	RBM-EVAL-ID	0.16	0.16	0.42	0.79	0.77	0.82	0.84	0.46	**0.92**
	RBM-EVAL-OOD	0.21	0.17	0.40	0.80	0.88	0.88	0.93	0.51	**0.94**
(b) Kendall $\tau_a \uparrow$	RBM-EVAL-OOD	0.19	0.08	0.13	0.45	0.50	0.47	0.55	0.01	**0.64**

TABLE I: (a) Reward alignment (VOC Pearson r) and (b) trajectory ranking (Kendall τ_a) on RBM-EVAL datasets. Baselines are split into categories based on training data: GVL and TOPReward training data is unknown, VLAC is trained on a 300k-trajectory dataset, and RoboDopamine is trained on a 100k-trajectory dataset. 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.

sampling T frames between them. [1]

Summary. ROBOMETER builds upon a base VLM, QWEN3-VL-4B-INSTRUCT, and inserts additional learnable tokens to predict preference, progress, and success. We train ROBOMETER on RBM-1M, a dataset consisting of both progress-labeled and progress-unlabeled data, by sampling trajectory comparisons across the entire dataset. These comparisons come from failure trajectories, comparisons across different tasks, and rewound videos. At inference, ROBOMETER can operate on either one or two trajectories: with a single trajectory, it outputs per-frame progress and success predictions; with two trajectories, it additionally predicts a preference while still producing per-frame predictions for the first trajectory.

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 the strongest 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.
- **GVL** [6] prompts a pre-trained closed-source LLM with shuffled video frames to predict task progress. We use GPT-5 mini as it is the best-performing closed-source model on the RoboRewardBench benchmark [11].
- **TOPReward** [26] prompts a pre-trained VLM (Qwen-3-VL 8B) with "does the trajectory complete the task?" and uses the log-likelihood of the "true" token as the reward.
- **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. While designed for reward prediction conditioned on a *goal image* and instruction, we evaluate it 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 [15] and RoboArena evaluations [14]. Generates *counterfactual* instructions via closed-source VLMs to simulate failed trajectories.

Custom Evaluation Datasets. We train ROBOMETER on RBM-1M. Unlike prior baselines that evaluate on in-distribution test splits [5, 10, 11, 12], we collect RBM-EVAL-OOD: 976 trajectories from 3 institutions guaranteed not to be in any baseline's training data, spanning 6 embodiments and diverse camera angles. We also use an in-distribution split, RBM-EVAL-ID.

Q1: Reward Evaluation

As detailed in Section II-B, our focus is on training a reward model which: (1) generalizes to new tasks, embodiments, and domains while (2) providing reward feedback useful for

[1] ROBOMETER is still a *dense*, per-frame reward model: during inference, we query over $o_{1:t}$ $\forall t \in [1, ..., T]$ for a trajectory of length T to produce T rewards. Our subsampling always includes the first and current frame.

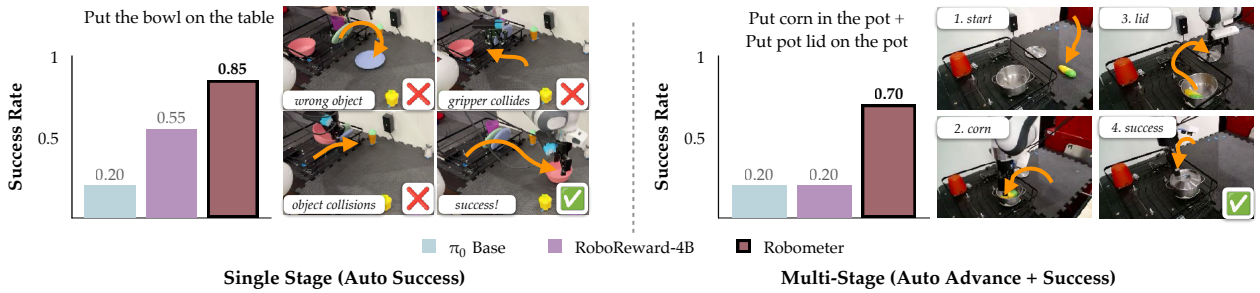

Fig. 4: **Automatic online RL** with DSRL [30] on a DROID [31] setup with ROBOMETER improves π_0 from 20% to 85% on a single-stage task and 20% to 70% on a two-stage task, outperforming RoboReward's overall success rate by 2.5×. DSRL with ROBOMETER learns to avoid base π_0 errors such as collisions or moving the wrong object. The setup is deemed "automatic" because success detection and stage advancement are handled automatically by the reward model, requiring human intervention only for physical scene resets.

policy learning. We structure this subsection to highlight ROBOMETER's strong performance across both criteria.

Trajectory Task Alignment. Our first main result demonstrates that ROBOMETER accurately distinguishes between different tasks in RBM-EVAL-OOD, which directly reflects its ability to assign rewards that align with task semantics, even across unseen robot embodiments, camera viewpoints, and scenes. We plot confusion matrices comparing unseen, successful trajectory videos versus their language instructions in Figure 3. Ideally, a **purple diagonal** indicates correct video-instruction pairs, with low (white) values elsewhere. ROBOMETER clearly produces the strongest disparity between the diagonal and off-diagonal elements, highlighting its superior ability to reward a robot for performing the *correct* task, which is especially important in cluttered, multi-task settings. This ability is in part due to how we sample *different-task* negative preference and progress examples across the entire dataset (cf. Section II-D).

Reward Alignment. Quantitatively, we evaluate the ability of baselines to predict increasing progress for *successful* robot videos from both RBM-EVAL-OOD and RBM-EVAL-ID in Table I(a). We report Value Order Correlation (VOC) [6] $\in [-1, 1]$, which calculates the Pearson correlation of predicted rewards for each trajectory video frame against their ground-truth timestep value. Overall, ROBOMETER performs the best across the board on both test sets, especially on RBM-EVAL-OOD.

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 [27], 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 0.64, 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. Notably, even when trained on the same data as RoboReward, ROBOMETER outperforms RoboReward in both policy ranking and reward alignment, highlighting the effectiveness of our data augmentation strategies.

We additionally demonstrate that ROBOMETER serves as a good initialization for domain-specific fine-tuning on RoboFAC [28]; fine-tuning via LoRA [29] yields better results than training from scratch, demonstrating effective adaptation with just 1 GPU. Overall, ROBOMETER outperforms baselines in both **generalization** and **distinguishing** successful / failed trajectories. We next analyze *why*.

	(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**	0.94	**0.64**	**0.32**
H3 **ReWiND Arch.**	0.48	-0.14	-0.02	0.51	0.01	0.02

TABLE II: 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.

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

We investigate individual components of ROBOMETER via controlled experiments on LIBERO [19], training on 1,709 demos and evaluating on 8,262 unseen trajectories from LIBERO-90.

- **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.

Results in Table II show that adding preference supervision (**H1 +Preference**) consistently improves policy ranking, and incorporating failed data (**H2 +Failed Data**) yields the largest gains—Kendall-τ improves to 0.92 on LIBERO-90. Replacing the VLM backbone (**H3 ReWiND Arch.**) causes severe degradation, confirming that a pre-trained VLM is essential. We

Fig. 5: **Offline RL with mixed-expertise data.** Success rates on two SO-101 tasks using IQL with sparse rewards, RoboReward-4B, and ROBOMETER. ROBOMETER achieves $\mathbf{2.4}\times$ higher success rate than the best baseline in each task.

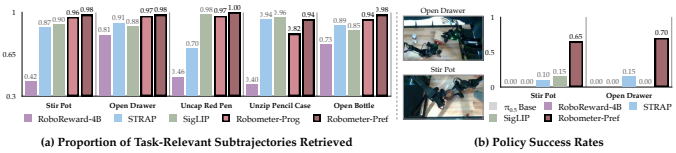

Fig. 6: **Data filtering & retrieval results.** (a) Proportion of task-relevant subtrajectories retrieved from an unannotated bimanual play dataset across five tasks. (b) Policy success rates after LoRA-finetuning $\pi_{0.5}$ on filtered data. ROBOMETER achieves the highest retrieval relevance and downstream policy performance.

also verify these trends hold for RL: ROBOMETER achieves $\mathbf{2-4}\times$ **better sample efficiency** than sparse reward on unseen LIBERO-90 tasks.

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, including both prehensile and non-prehensile tasks: (1) automatic online RL, (2) offline RL with mixed-expertise data, (3) data filtering and retrieval for policy improvement, and (4) out-of-distribution failure detection. Across all experiments, we compare against RoboReward-4B—the strongest baseline reward model in our offline evaluations—to assess how ROBOMETER's dense, instruction-aligned rewards affect learning stability, robustness, and sample efficiency. We also compare against strong, relevant, non-reward-model baselines for each setting where applicable. All policy learning results are averaged over 20 evaluation trials.

Automatic Online RL. First, we evaluate ROBOMETER in an *automated* online RL setting by training DSRL [30] from scratch on a π_0 base policy [32] pre-trained on DROID [31]. 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 physical scene resets. For comparison, RoboReward's discrete scores are also used for both reward shaping and success detection. As shown in Figure 4 (left), DSRL+ROBOMETER improves success from 20% to 85% in ≤ 45 **minutes** (10k timesteps), outperforming RoboReward's 55%. This gap arises from a key failure mode of RoboReward: it frequently assigns maximum rewards for unrelated tasks (e.g., picking up the wrong object), leading to premature resets and reinforcing incorrect behaviors. In contrast, ROBOMETER provides a more reliable learning signal.

Next, we evaluate a **longer-horizon RL** setting in Figure 4 (right), where ROBOMETER's success predictions trigger progression between stages of a multi-stage task. Unlike methods that explicitly train with multi-stage rewards and thus require stage labels [7, 33], 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%, which suffers from inaccurate rewards and unreliable stage transitions. Across both setups, ROBOMETER outperforms RoboReward's overall success rate by an average of $\mathbf{2.5}\times$.

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 III: **Failure detection performance**. ROBOMETER achieves the highest average F1 score. T.U. is the token-uncertainty baseline.

Finally, we perform an additional online RL experiment—*model-based RL* integrating ROBOMETER into DreamZero [34]—where ROBOMETER improves success rate from 20% to 70%.

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. We train policies with IQL [35] and sweep $\gamma \in 0.90, 0.95, 0.99$, reporting the best checkpoint over 30k steps. As shown in Figure 5, ROBOMETER performs best at $\gamma = 0.9$ and outperforms both baselines with a $\mathbf{2.4}\times$ success rate improvement over the best baseline in each task. RoboReward performs similarly to sparse rewards, as its categorical (1–5) outputs provide less dense guidance and often assign high rewards to suboptimal trajectories.

Data Filtering & Retrieval. We evaluate ROBOMETER for unsupervised data filtering using a bimanual "play" dataset [36] of unannotated, multi-task trajectories on a Trossen AI setup (not in `RBM-1M`), retrieving the top 100 subtrajectories per task instruction. We compare against RoboReward, SigLIP [37], and STRAP [38]. As shown in Figure 6, ROBOMETER consistently achieves higher retrieval relevance across five tasks. We LoRA-finetune $\pi_{0.5}$ [29, 39, 40] on retrieved segments; policies trained on ROBOMETER-filtered data average a $\mathbf{4.5}\times$ higher success rate than the best baseline.

Failure Detection. We evaluate zero-shot failure detection on 100 trajectories from a Franka Panda DROID robot (30 successful, 70 failed) spanning 7 tasks not in `RBM-1M`, comparing against token-uncertainty [8] of π_0-FAST-DROID [41], VLAC, GPT-5-mini, and RoboReward-4B. As shown in Table III, ROBOMETER achieves the highest average F1 score, robustly detecting failures fully zero-shot—unlike prior methods requiring task-specific thresholds [8, 42, 43].

ACKNOWLEDGMENTS

We thank Chuning Zhu and Kaiyuan (Kyle) Zheng for testing out ROBOMETER integrated with DreamZero and providing us with starter code for our model-based RL experiment. Additionally, we thank Harine Ravichandiran for helping supervise an online RL run and Matthew Hong for helping us with task selection for our LIBERO RL ablation experiments.

Anthony L, YK, and EB acknowledge funding by the Airbus Institute for Engineering Research (AIER); Jesse Z, LZ, and DF acknowledge support by the Cross-Pacific AI Initiative from Amazon; Jesse Z and AG are partly supported by funding from Amazon FAR through the Amazon Science Hub; Jesse Z and DF acknowledge support by Amazon and Toyota Research Institute; MH and AB acknowledge funding from the Tata Group via the MIT Generative AI Impact Consortium (MGAIC) Award; Jiahui Z and YX acknowledge funding from the National Science Foundation (NSF) under Grant No. 2520553.

Additionally, we thank the UW Hyak and Tillicum high-performance computing clusters and USC Center for Advanced Research Computing (CARC) for providing us with compute resources.

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. Bajcsy, "Position: Good embodied reward models need bad behavior data," in *International Conference on Machine Learning (ICML)*, 2026.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] J. Farebrother, J. Orbay, Q. Vuong, A. A. Taiga, Y. Cheb-

otar, T. Xiao, A. Irpan, S. Levine *et al.*, "Stop regressing: Training value functions via classification for scalable deep RL," in *International Conference on Machine Learning (ICML)*, 2024.

[24] P. Intelligence, A. Amin, R. Aniceto, A. Balakrishna, K. Black, K. Conley, G. Connors, J. Darpinian *et al.*, "$\pi^*_{0.6}$: A vla that learns from experience," *arXiv:2511.14759*, 2025.

[25] 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.

[26] S. Chen, C. Harrison, Y.-C. Lee, A. J. Yang, Z. Ren, L. J. Ratliff, J. Duan, D. Fox *et al.*, "Topreward: Token probabilities as hidden zero-shot rewards for robotics," *arXiv preprint arXiv:2602.19313*, 2026.

[27] 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.

[28] W. Lu, M. Ye, Z. Ye, R. Tao, S. Yang, and B. Zhao, "Robofac: A comprehensive framework for robotic failure analysis and correction," *arXiv preprint arXiv:2505.12224*, 2025.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, G. Kurian, S. Indupuru, Y. L. Tan *et al.*, "World action models are zero-shot policies," *arXiv preprint arXiv:2602.15922*, 2026.

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

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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," in *Robotics: Science and Systems (RSS)*, 2025.

[42] 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.

[43] 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.

[44] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, C. R. Garrett, F. Ramos, D. Fox *et al.*, "Hamster: Hierarchical action models for open-world robot manipulation," in *International Conference on Learning Representations (ICLR)*, 2025.

[45] J. Zhang, M. Memmel, K. Kim, D. Fox, J. Thomason, F. Ramos, E. Bıyık, A. Gupta *et al.*, "Peek: Guiding and minimal image representations for zero-shot generalization of robot manipulation policies," in *International Conference on Robotics and Automation (ICRA)*, 2026.