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

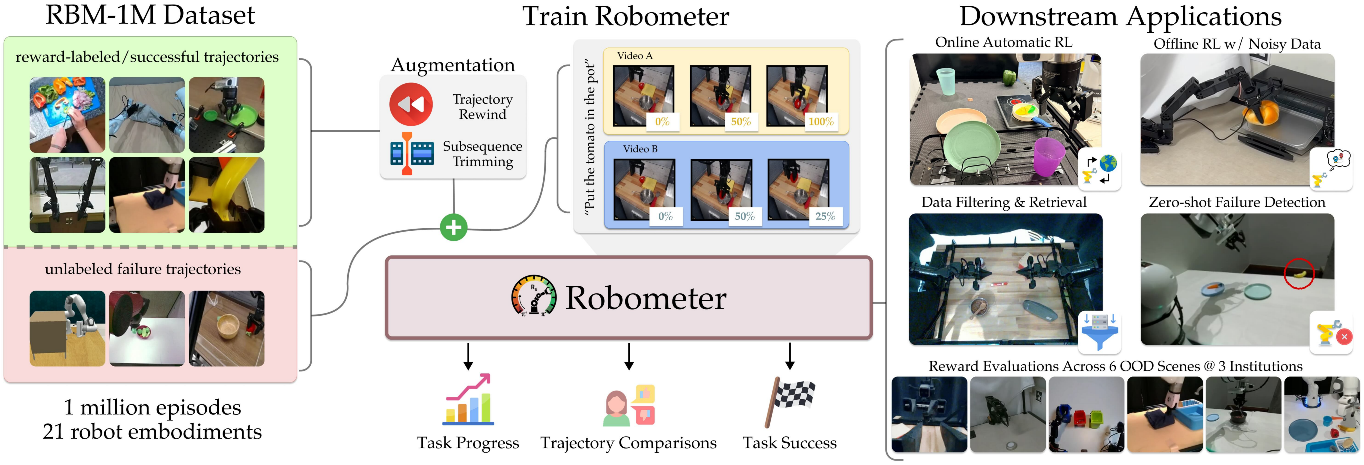


Fig. 1: **Robometer Overview.** Robometer is trained on `RBM-1M`, a 1M-trajectory dataset spanning 21 robot embodiments, containing both reward-labeled/expert trajectories and reward-unlabeled, failed trajectories. The model is supervised with a dual objective: predicting frame-level task progress (reward) 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*—General-purpose robot reward models are typically trained to predict absolute task progress from expert demonstrations, providing only local, frame-level supervision. While effective for expert demonstrations, this paradigm scales poorly to large-scale robotics datasets where failed and suboptimal trajectories are abundant and assigning dense progress labels is ambiguous. We introduce Robometer, a scalable reward modeling framework that combines intra-trajectory progress supervision with inter-trajectory preference supervision. Robometer is trained with a dual objective: a frame-level progress loss that anchors reward magnitude on expert data, and a trajectory-comparison preference loss that imposes global ordering constraints across trajectories of the same task, enabling effective learning from both real and augmented failed trajectories. To support this formulation at scale, we curate `RBM-1M`, a reward-learning dataset comprising over one million trajectories spanning diverse robot embodiments and tasks, including substantial suboptimal and failure data. Across benchmarks and real-world evaluations, Robometer learns more generalizable reward functions than prior methods and improves robot learning performance across a diverse set of downstream applications. Code, model weights, and videos at https://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]. 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 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 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 1). To support downstream RL, ROBOMETER also predicts a frame-level task success. Our ablations reveal a mutual reinforcement effect: preference supervision improves ROBOMETER's ability to distinguish suboptimal from successful trajectories even when trained solely on expert demonstrations, suggesting that global comparative constraints induce a better-structured internal reward representation. Furthermore, as additional unlabeled suboptimal data is introduced, ROBOMETER scales naturally to further improve performance.

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. The scale and diversity of `RBM-1M` are therefore essential for learning globally consistent preference relations across embodiments, tasks, and viewpoints, and for fully leveraging failure data that would otherwise be discarded. 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://robometer.github.io.

II. Related Works

Learning reward functions is a central problem in reinforcement learning and robotics, and prior work has explored a wide range of approaches that differ in both the form of supervision and the scope of generalization.

Reward Learning from Demonstrations. Learning reward functions from human supervision is a long-studied topic in inverse RL (IRL), where reward functions are inferred from human demonstrations [14, 15, 16, 17, 18] or from expert and goal-state distributions [19, 20, 21, 22, 23, 24]. However, most IRL methods require task-specific expert demonstrations, making it necessary to collect new demonstrations whenever the task or reward specification changes. In contrast, ROBOMETER trains a general-purpose vision-language-input reward model which can generalize effectively to new tasks.

Preference-Based Reward Learning. Prior work in psychology has observed that humans often use relative comparisons over absolute numerical scales when making judgments [1, 2, 3]. This insight has been central to modern reinforcement learning from human feedback (RLHF), where preference supervision from humans is used to learn reward models that are only identifiable up to monotone transformations, yet sufficient for effective policy optimization [25, 26, 27, 28, 29, 30, 31, 32, 33]. In contrast to these works, which treat preferences as the primary supervision signal to train domain-specific reward models, we generate preference supervision from both synthetic and real trajectories for which assigning progress labels is difficult. We use this preference signal as an *auxiliary* objective that complements direct progress prediction, enabling ROBOMETER to learn from large, heterogeneous datasets without additional human preference supervision. Closest in spirit, Kwok et al. [34] generate synthetic preference labels from action mean-squared-error to train a robot action verifier, and Venkataraman et al. [9], Wang et al. [35] use vision-language models (VLMs) to generate frame-level preferences as the primary supervision signal for task-specific reward models. In contrast, we generate preference supervision by comparing entire trajectories and use it as an auxiliary signal to learn a general-purpose reward function across tasks and embodiments.

Rewards from Foundation Models. Finally, recent work has sought to construct more general reward functions that operate directly on images, videos, and language. Large language models (LLMs) and VLMs have been applied to reward design, for example by generating executable reward code or shaping functions from natural language descriptions [36, 37, 38], directly producing reward functions [39], or guiding reward computation via language-conditioned state masking [40]. However, most of these approaches assume access to privileged state information, which is often unavailable or difficult to obtain in real-world robotic deployment settings.

To overcome this challenge, some approaches use task progress as a proxy reward, either by applying pre-trained

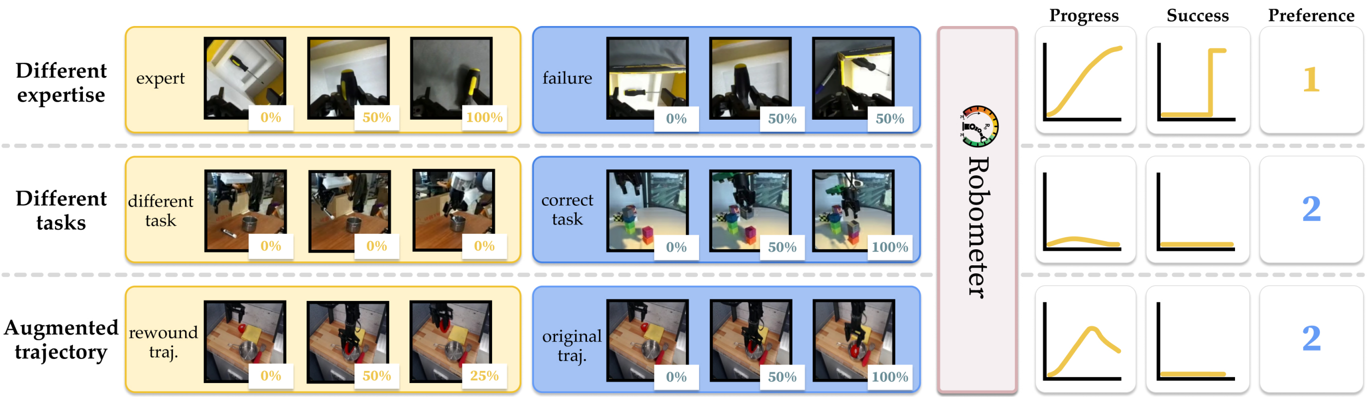


Fig. 2: **ROBOMETER** is a VLM-based reward model, that predicts dense, per-frame progress-based rewards and success labels for the first of two video trajectories. To be able to train with failed, non-expert data, we also predict which of the two video trajectories better completes the task. We use three strategies for curating training examples from our given datasets, which are further detailed in Section III-D with model architecture shown in Appendix Figure 11.

VLMs as zero-shot progress or success estimators [6, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50], or by training domain-specific models with progress-prediction objectives [4, 5, 7, 41, 51, 52, 53, 54, 55, 56]. However, directly using pre-trained VLMs for zero-shot video-language reward prediction often yields noisy or inconsistent signals [5, 11, 57], while smaller per-task models tend to overfit to domain-specific visual and semantic features, limiting generalization.

More recent methods address these limitations by fine-tuning large pretrained VLMs or VLAs, enabling progress prediction to leverage visual–semantic representations learned from diverse data [10, 11, 12, 58, 59, 60] and demonstrating improved sample efficiency for real-world RL. VLAC [10], for example, co-trains a VLA with relative progress difference targets, while $\pi^*_{0.6}$ [59] and Ghasemipour et al. [58] train distance-to-goal value functions that encode task progress. RoboReward [11] fine-tunes a VLM to predict discretized (1-5) progress labels generated via counterfactual instruction labeling by closed-source VLMs, and RoboDopamine [12] similarly fine-tunes a VLM for progress prediction but requires a goal image in addition to the language instruction and task-specific one-shot fine-tuning. In contrast, ROBOMETER introduces an explicit auxiliary *preference prediction* objective, enabling us to scale to datasets containing suboptimal or failed trajectories without reward labels. Using such failed trajectories, even as just in-context learning examples for pre-trained VLMs, has been demonstrated to help reward models align more closely with human-specified rewards [13].

III. ROBOMETER

We propose ROBOMETER, a large-scale reward model designed to provide 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**.

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. For expert demonstrations, $p = 1.0$; for datasets with partial progress labels (e.g., RoboArena [61]), 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 [62] and subsets of high-quality, single-robot data such as AGIBotWorld [63]; (2) **Human videos** from datasets such as Epic-Kitchens [64] for scene diversity or human-robot paired datasets like RH20T [65] to promote embodiment-invariant representations; (3) **Simulation** data from sources like LIBERO [66]; and (4) **Failed trajectories** from automated policy rollouts [67] and failure-detection datasets [68].

Our dataset overall includes 21 robot embodiments and over 1 million trajectories, hence `RBM-1M`. We also construct two evaluation datasets, `RBM-EVAL-ID` and `RBM-EVAL-OOD`, detailed in Section A-3. For further details on dataset filtering and aggregation, see Appendix A. We also list all dataset categories and trajectory counts in Appendix Figure 10, Table IX.

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** (⟨|prog_token|⟩) within the first video sequence and a single **preference token** (⟨|pref_token|⟩) at the end of the multi-video prompt:

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

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 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:

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

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 $p = 1$. Rather than directly regressing a scalar progress value, we discretize progress into N uniformly spaced bins over $[0, 1]$ and model progress prediction as a categorical distribution, following the C51 formulation [69]. 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 $t = T$. In some datasets, the human operator stops recording a trajectory several frames after the task has already been completed; to improve the accuracy of both progress and success prediction, we sample 10 trajectories per data source to determine the point at which the task actually ends (Appendix Section A-1). 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, 70]. 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.

Subsequence Trimming. We discourage overfitting to fixed trajectory lengths by randomly sampling T frames with uniform start/end indices from the full video.

Summary. Robometer builds upon a base VLM, Qwen3-VL-4B-Instruct, and inserts additional learnable tokens to predict preference, progress, and success. We train

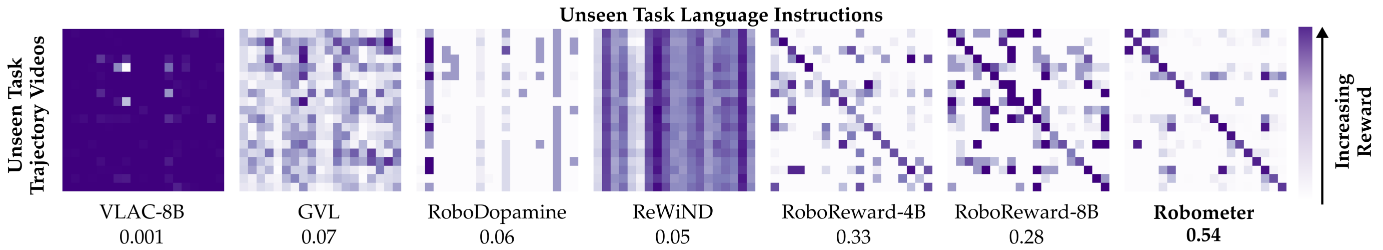


Fig. 3: **Video-Language Reward Confusion Matrix.** For each task sampled at random from *self-collected, unseen* data from RBM-EVAL-OOD, we compute rewards for all combinations of demonstration videos and language descriptions. ROBOMETER produces the most diagonal-heavy confusion matrix, indicating 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	RoboReward-4B	RoboReward-8B	ROBOMETER	ReWiND	ROBOMETER
(a) VOC $r \uparrow$	RBM-EVAL-ID	0.16	0.16	0.13	0.77	0.82	0.84	0.46	**0.92**
	RBM-EVAL-OOD	0.21	0.17	0.08	0.88	0.88	0.93	0.51	**0.95**
(b) Kendall $\tau_a \uparrow$	RBM-EVAL-OOD	0.19	0.08	0.11	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. Baselines are split into categories based on training data: GVL's training data (w/ GPT-5-mini) 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.

ROBOMETER on RBM-1M, a diverse reward modeling dataset consisting of both progress-labeled and progress-unlabeled data, by sampling trajectory comparisons across the entire dataset. These comparisons come from unlabeled failure trajectories, comparisons across different tasks, and generated pseudo-failures via video rewind. For additional model implementation and training details, see Appendix B.

IV. 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 described in Section II. We list a dataset size comparison table for baselines and related papers in Appendix Table VI.

- **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 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-3B** [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 [62] and RoboArena evaluations [61]. 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. Prior large-scale reward modeling baselines mainly evaluate on validation or test set versions of the datasets they train on [5, 10, 11, 12], which contain in-distribution arms, camera angles, or scenes. Instead, 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, consisting of 6 embodiments (including human hands), 3 of which are not in RBM-1M, collected across diverse camera angles. We also aggregate an in-distribution test split of unseen trajectories collected from datasets in RBM-1M, denoted RBM-EVAL-ID. See more evaluation dataset details in Section A-3.

Q1: Reward Evaluation

As detailed in Section III-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 *policy learning*. We structure this subsection to highlight

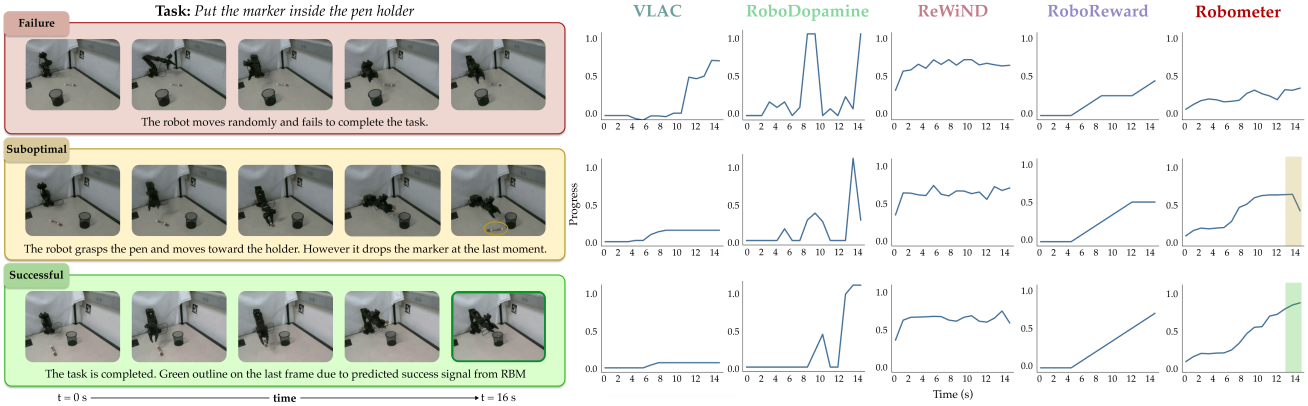


Fig. 4: **Qualitative Analysis of Failure, Suboptimal and Successful Trajectories.** We visualize the progress predictions for three trajectories of different quality for the same task. Notably, for the suboptimal trajectory, Robometer predicts steadily increasing progress as the robot approaches the pen holder, but sharply reduces its progress estimate when the marker is dropped, correctly reflecting regression in task completion. In contrast, RoboReward continues to assign high progress despite the task failure. Finally, Robometer is the only model that correctly predicts task success for the successful trajectory (i.e., high final progress value and explicit success prediction).

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].

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 III-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. We break down per-dataset and per-subset results for both test sets in Appendix C.

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

This RoboReward-only Robometer variant attains an MAE of 0.75, outperforming RoboReward-4B. Our full model improves to 0.72, trailing only the substantially larger RoboReward-8B and GPT-5-mini. The strong performance of the RoboReward-only variant highlights the benefit of our dual-objective training and augmentations even on narrower datasets. Finally, note that in our OOD evaluations in Table I, RoboReward-8B and 4B perform similarly, and GPT-5-mini with GVL performs markedly worse. We therefore attribute the stronger results of larger models on RoboRewardBench to its 5 discrete labels and its final-frame-only evaluation protocol.

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 [71], 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. Notably, even when trained on the same data

Method	VOC r ↑	Kendall τ ↑	Succ-Fail Diff ↑
Robometer-4B (Zero-shot)	0.652	0.436	0.141
Qwen3-VL (LoRA)	0.701	0.067	0.005
Qwen3-VL (Full FT)	0.727	0.102	0.008
Robometer-4B (LoRA)	0.875	0.786	0.271
Robometer-4B (FFT)	**0.884**	**0.802**	**0.302**

TABLE III: **Finetuning Robometer on RoboFAC dataset.** Zero-shot performance is strong, and fine-tuning the base Qwen3 VLM performs worse on Kendall τ and success - fail difference compared to zero-shot Robometer. Meanwhile, Robometer fine-tuned performs best, either with LoRA or FFT.

as RoboReward, Robometer outperforms RoboReward in both policy ranking and reward alignment, highlighting the effectiveness of our data augmentation strategies.

To further illustrate this behavior, we visualize reward predictions over time for failed, suboptimal, and successful trajectories over time in Figure 4. Robometer exhibits sharper separation in rewards between different levels of execution and accurately reflects regression in task progress. We also point to additional results in Appendix C evaluating preference prediction accuracy.

Reward Fine-tuning. Finally, we demonstrate that Robometer serves as a good initialization for domain-specific fine-tuning. We fine-tune on RoboFAC [72], a dataset of robotic failures and corrections spanning 16 tasks and 53 scenes (11k trajectories), including both simulated and real-world successes and failures. We adapt Robometer via LoRA [73] adapters and full fine-tuning (FFT). We compare against fine-tuning the base VLM `Qwen/Qwen3-VL-4B-Instruct` in the same way.

In addition to the previous VOC r and Ranking Kendall τ_a, we also compare a *success - fail* metric measuring the difference in final reward between successful and failed trajectories of the same task. Qwen3 fine-tuned still performs worse than Robometer zero-shot on 2 out of 3 metrics; fine-tuning from Robometer yields substantially better reward evaluation results than training Qwen3-VL from scratch across all of our ranking metrics (Table III). Importantly, LoRA and FFT perform similarly, demonstrating that Robometer can be effectively fine-tuned with just 1 GPU. See further experiment details in Appendix F.

Overall, our results demonstrate that **Robometer outperforms reward baselines** in both **generalization** and **distinguishing** successful / failed trajectories, and that it also serves as a **strong initialization for further fine-tuning**. We next analyze *why*.

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.

	(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 IV: 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.

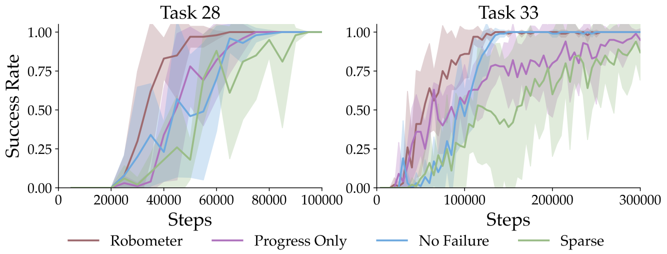


Fig. 5: **RL w/ Ablation Models** in LIBERO-90 tasks from scratch, corresponding to ablations trained only on LIBERO-10/Object/Goal/Spatial data from Table IV. We report the average success rate ± standard deviation across 5 seeds.

H3 Fine-tuning from **pre-trained VLMs** helps with reward predictions on unseen tasks.

Our main analysis is performed via a controlled setting with data from the LIBERO [66] robot manipulation simulated benchmark. We train models with 1,709 successful demos from `LIBERO-{10, Object, Goal, Spatial}` and evaluate performance on a sample of the 8,262 unseen, paired, successful and failed trajectories from `LIBERO-90`.

Reward Model Ablations. To measure **H1**, we train Robometer with *only* progress prediction and also Robometer with both progress and preference prediction on the 1,709 demo dataset containing no failed trajectories. We then measure **H2**—about scaling with failure data for preference training—by adding in 1,929 generated, failed LIBERO trajectories and train Robometer with the full Robometer training objective of progress and preference prediction on the larger dataset. These LIBERO ablations are trained with LoRA [73] due to the small dataset size. Finally, we verify the importance of a pre-trained VLM (**H3**) by training a larger, 500M-parameter version of ReWiND's transformer model (originally designed for low-data regimes) with our preference and progress objectives on the paired-failure LIBERO dataset.

We depict results on LIBERO, and separately, trained on the full `RBM-1M` and evaluated on `RBM-EVAL-OOD` (with failed data removed for +Preference and +Failed Data), in Table IV. First, comparing **H1 Prog. Only** to **H1 +Preference**, adding preference supervision consistently improves policy ranking performance, increasing Kendall-τ_a from 0.63 to 0.74 on LIBERO-90 and from 0.31 to 0.54 on `RBM-EVAL-OOD`. Second, **incorporating failed trajectories** for preference training (**H2 +Failed Data**) yields the **largest gains** across all ranking-based metrics. On LIBERO-90, Kendall-τ improves to 0.92 and the average difference in final rewards between

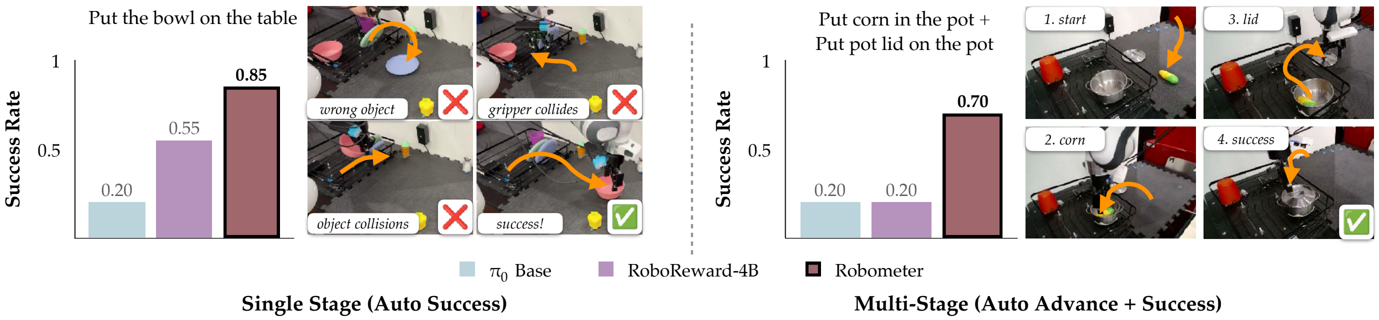


Fig. 6: **Automatic online RL** with DSRL on a DROID setup with ROBOMETER improves π_0 from 20% to 85% on a single-stage task and 20% to 70% on a two-stage task, outperforming RoboReward's overall success rate by 2.5×. DSRL with ROBOMETER learns to avoid base π_0 errors such as collisions or moving the 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.

success–failure increases more than 4× relative to progress-only training. Similar trends hold on RBM-EVAL-OOD, where Kendall-τ increases from 0.54 to 0.66 and the success–failure separation improves from 0.24 to 0.33. Finally, replacing the pretrained VLM backbone with a scaled variant of ReWiND's architecture (**H3 ReWiND Arch.**) results in a severe degradation across all metrics, confirming that large-scale multimodal pretrained backbone is essential for learning generalizable reward representations. See Appendix D for additional ablations.

Ablation Results on RL Performance. Before moving to our full policy learning experiments, we verify that the reward evaluation metric trends observed in our ablation studies hold for policy learning. We train policies via online RL using the ablated reward models on two tasks from the unseen LIBERO-90 suite. These two tasks were specifically selected because sparse-reward RL stably learns to a near-100% success rate, allowing us to directly compare sample efficiency.

Results in Figure 5 demonstrate that improvements in reward evaluation metrics consistently transfer to policy success rates. For both tasks, policies trained using ROBOMETER (the **H2** LIBERO model) demonstrate better sample efficiency than ablations (**H1** LIBERO models) and sparse reward, highlighting the importance of dense, well-calibrated reward signals for efficient policy optimization. Overall, ROBOMETER trained only on LIBERO achieves $\mathbf{2-4\times}$ **better sample efficiency** than sparse reward on these unseen tasks. Additional details of the RL training setup are provided in Appendix E.

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. Additional details and finer-grained results on each experiment can be found in Appendix E.

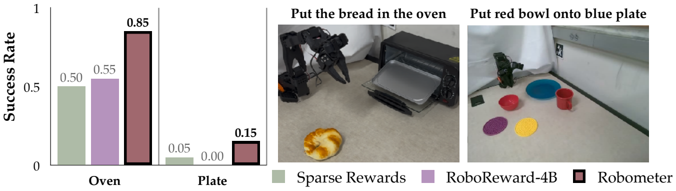
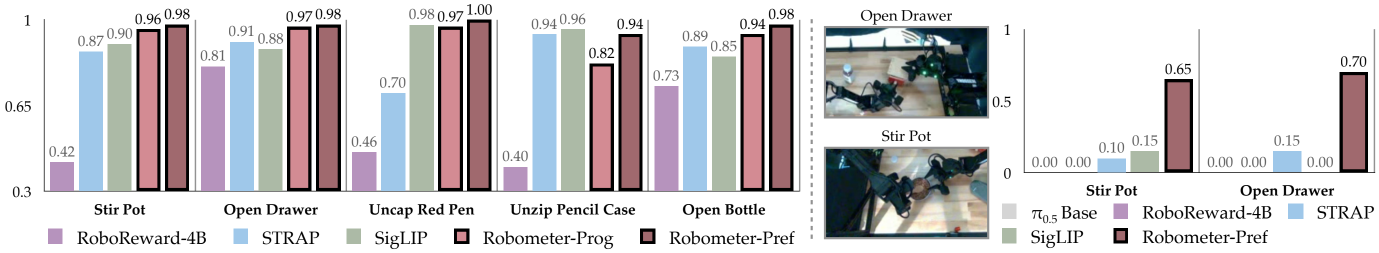


Fig. 7: **Offline RL** results using IQL on a mixture of Noisy and Expert trajectories. ROBOMETER rewards consistently outperform both RoboReward and sparse rewards: 2.4× average success rate improvement over the best baseline for each task.

Automatic Online RL. First, we evaluate ROBOMETER in an *automated* online RL setting by training DSRL [74] from scratch on a π_0 base policy [75] pre-trained on DROID [76]. 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 6 (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 multi-stage RL* setting in Figure 6 (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 [56] 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%, which suffers from inaccurate rewards and unreliable stage transitions. Across both setups, ROBOMETER outperforms RoboReward's overall success rate by an average of $\mathbf{2.5\times}$.

Finally, we perform an additional online RL



(a) Proportion of Task-Relevant Subtrajectories Retrieved

(b) Policy Success Rates

Fig. 8: **(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. Small amounts of suboptimal & unrelated data retrieved by other baselines degrade policy-learning performance: ROBOMETER-retrieval attains an average 4.5× success rate improvement over the best baseline.

experiment—*model-based RL* integrating ROBOMETER into DreamZero [77]—where ROBOMETER improves DreamZero's success rate from 20% to 70%. See details and results in Appendix G.

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 7. We train policies with Implicit Q-Learning (IQL) [78] 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 over 30,000 offline training steps. 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. RoboReward performs similarly to sparse reward across the γ sweep due to its categorical 1-5 rewards providing less dense guidance than ROBOMETER's dense rewards.

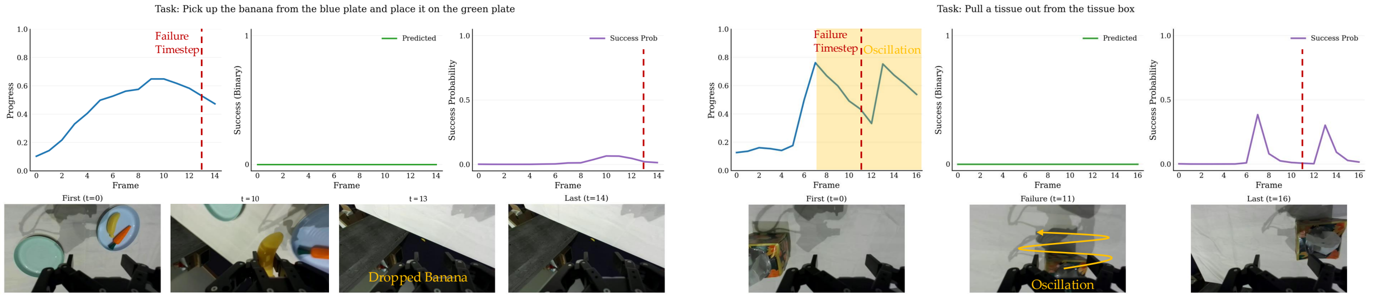
Data Filtering & Retrieval. We next evaluate ROBOMETER as a mechanism for unsupervised data filtering and retrieval. Using a bimanual "play" dataset [79] of unannotated, multi-task trajectories collected on a Trossen AI setup (not in `RBM-1M`), we retrieve the top 100 subtrajectories for a given task instruction. We compare retrieval relevance against RoboReward, pre-trained SigLIP [80], and a retrieval-specific baseline, STRAP [81]. 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 8(a), ROBOMETER consistently achieves higher retrieval relevance across five tasks. Finally, we LoRA-finetune $\pi_{0.5}$ [73, 82, 83] on these retrieved segments. Policies trained on ROBOMETER-filtered data vastly outperform those using baseline-retrieved data on `Stir the Pot` and `Open the Red Drawer` (Figure 8(b)), demonstrating its efficacy for targeted imitation learning. Low baseline success rates despite high retrieval rates stem from their retrieval of more failed and suboptimal, yet task-relevant, subtrajectories. Overall, ROBOMETER-retrieval averages a $\mathbf{4.5}\times$ higher success rate than the best baseline in each task.

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

Failure Detection. Detecting failures during online deployment is critical for safe robotic operation. Thus, 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`. Failures are evenly split between irreversible failures (e.g., drops or spills) and insufficient-progress failures, where execution stalls or terminates prematurely. We compare our method against: the token-uncertainty [8] of π_0-FAST-DROID [84] 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. Failures are detected via temporal inconsistencies in predicted per-frame rewards.

As shown in Table V, ROBOMETER achieves the highest average F1 score, effectively balancing true positive and true negative rates (TPR and TNR); the full breakdown with TPR and TNR is provided in Appendix Table XXIII. VLAC frequently flags trajectories as failures, achieving high TPR but low TNR, resulting in lower F1 scores. RoboReward-4B performs competitively but underperforms ROBOMETER, particularly on tasks with subtle failure modes such as *fold towel* and *pull tissue*. Figure 9 illustrates representative failure



(a) Object dropped during transport.

(b) Oscillatory behavior without task completion.

Fig. 9: **Failure Detection Examples. (a):** Terminal events such as drops cause a sharp regression in predicted task progress, which ROBOMETER flags shortly after the event. **(b):** Non-terminal failures correctly exhibit oscillatory progress with ROBOMETER.

cases. Irreversible failures such as object drops induce sharp regressions in predicted task progress, which ROBOMETER flags shortly after the event, while non-terminal failures exhibit stagnating or oscillatory progress without convergence to success. As detailed in Section E-3, ROBOMETER robustly detects irreversible, insufficient-progress, and non-terminal failures (e.g., hovering, oscillation, or partial completion), fully zero-shot across tasks and environments—unlike prior methods that require task-specific thresholds, calibration, or test-time interaction [8, 85, 86].

V. Limitations and Future Work

ROBOMETER operates as a frame-based reward model over temporally subsampled video inputs (e.g., 8 frames per trajectory), which enables scalable training but limits its ability to capture fine-grained temporal dynamics and long-horizon structure. In addition, real-world robot executions exhibit a wide diversity of failure modes, many of which are rare, subtle, or task-specific, and the current training data may not fully capture this breadth. As a result, ROBOMETER may fail to recognize or correctly reward certain failure cases that fall outside the dominant patterns seen during training. As a vision-language-based model, ROBOMETER also lacks direct access to latent physical state such as contact forces, grasp stability, or compliance, and may fail to recognize or correctly reward failure cases driven by these factors until they become visually observable. Future work could address these limitations by incorporating denser temporal modeling, VQA-style supervision to reason about task structure and completion criteria, and off-domain data for better generalization [87, 88], as well as by developing more systematically curated failure datasets that better reflect the diversity of real-world failure modes [13].

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. 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] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2000.

[15] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2004.

[16] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2008.

[17] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning (ICML)*, 2016.

[18] A. Bobu, M. Wiggert, C. Tomlin, and A. D. Dragan, "Feature expansive reward learning: Rethinking human input," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2021.

[19] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[20] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *International Conference on Robotics and Automation (ICRA)*, 2018.

[21] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2018.

[22] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine, "Variational inverse control with events: A general framework for data-driven reward definition," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[23] Y. Fu, H. Zhang, D. Wu, W. Xu, and B. Boulet, "Robot policy learning with temporal optimal transport reward," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[24] A. K. Jain, V. Mohta, S. Kim, A. Bhardwaj, J. Ren, Y. Feng, S. Choudhury, and G. Swamy, "A smooth sea never made a skilled SAILOR: Robust imitation via learning to search," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[25] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[26] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Robotics: Science and Systems (RSS)*, 2017.

[27] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time," in *International Conference on Human-Robot Interaction (HRI)*, 2018.

[28] E. Biyik, N. Huynh, M. J. Kochenderfer, and D. Sadigh, "Active preference-based gaussian process regression for reward learning," in *Robotics: Science and Systems*

(RSS), 2020.

[29] K. Lee, L. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," in *International Conference on Machine Learning (ICML)*, 2021.

[30] J. Hejna and D. Sadigh, "Few-shot preference learning for human-in-the-loop rl," in *Conference on Robot Learning (CoRL)*, 2022.

[31] V. Myers, E. Biyik, N. Anari, and D. Sadigh, "Learning multimodal rewards from rankings," in *Conference on Robot Learning (CoRL)*, 2021.

[32] Z. Yang, M. Jun, J. Tien, S. J. Russell, A. Dragan, and E. Bıyık, "Trajectory improvement and reward learning from comparative language feedback," in *Conference on Robot Learning (CoRL)*, 2024.

[33] Y. Korkmaz and E. Bıyık, "Mile: Model-based intervention learning," in *International Conference on Robotics and Automation (ICRA)*, 2025.

[34] J. Kwok, C. Agia, R. Sinha, M. Foutter, S. Li, I. Stoica, A. Mirhoseini, and M. Pavone, "Robomonkey: Scaling test-time sampling and verification for vision-language-action models," in *Conference on Robot Learning (CoRL)*, 2025.

[35] 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," in *International Conference on Machine Learning (ICML)*, 2024.

[36] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan *et al.*, "Eureka: Human-level reward design via coding large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[37] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. Gonzalez Arenas, H.-T. Lewis Chiang, T. Erez *et al.*, "Language to rewards for robotic skill synthesis," in *Conference on Robot Learning (CoRL)*, 2023.

[38] 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," in *International Conference on Learning Representations (ICLR)*, 2024.

[39] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[40] M. Hwang, A. Forsey-Smerek, N. Dennler, and A. Bobu, "Masked irl: Llm-guided reward disambiguation from demonstrations and language," *arXiv preprint arXiv:2511.14565*, 2025.

[41] Y. Cui, S. Niekum, A. Gupta, V. Kumar, and A. Rajeswaran, "Can foundation models perform zero-shot task specification for robot manipulation?" in *Learning for Dynamics and Control Conference (L4DC)*, 2022.

[42] P. Mahmoudieh, D. Pathak, and T. Darrell, "Zero-shot reward specification via grounded natural language," in *International Conference on Machine Learning (ICML)*, 2022.

[43] S. A. Sontakke, J. Zhang, S. Arnold, K. Pertsch, E. Biyik, D. Sadigh, C. Finn, and L. Itti, "Roboclip: One demonstration is enough to learn robot policies," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[44] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi, "Vision-language models as success detectors," in *Conference on Lifelong Learning Agents*, 2023.

[45] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang, "Vip: Towards universal visual reward and representation via value-implicit pre-training," in *International Conference on Learning Representations (ICLR)*, 2023.

[46] A. Adeniji, A. Xie, C. Sferrazza, Y. Seo, S. James, and P. Abbeel, "Language reward modulation for pretraining reinforcement learning," *arXiv preprint arXiv:2308.12270*, 2024.

[47] Y. Fu, H. Zhang, D. Wu, W. Xu, and B. Boulet, "FuRL: Visual-language models as fuzzy rewards for reinforcement learning," in *Internatonal Conference on Machine Learning (ICML)*, 2024.

[48] L. Guan, Y. Zhou, D. Liu, Y. Zha, H. B. Amor, and S. Kambhampati, "Task success is not enough: Investigating the use of video-language models as behavior critics for catching undesirable agent behaviors," in *Conference on Language Modeling (COLM)*, 2024.

[49] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner, "Vision-language models are zero-shot reward models for reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2024.

[50] 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.

[51] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang *et al.*, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," in *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[52] K. Nottingham, P. Ammanabrolu, A. Suhr, Y. Choi, H. Hajishirzi, S. Singh, and R. Fox, "Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling," in *International Conference on Machine Learning (ICML)*, 2023.

[53] T. Nam, J. Lee, J. Zhang, S. J. Hwang, J. J. Lim, and K. Pertsch, "Lift: Unsupervised reinforcement learning with foundation models as teachers," *arXiv preprint arXiv:2312.08958*, 2023.

[54] Y. J. Ma, W. Liang, V. Som, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman, "Liv: Language-image

representations and rewards for robotic control," in *International Conference on Machine Learning (ICML)*, 2023.

[55] K.-H. Hung, P.-C. Lo, J.-F. Yeh, H.-Y. Hsu, Y.-T. Chen, and W. H. Hsu, "VICtor: Learning hierarchical vision-instruction correlation rewards for long-horizon manipulation," in *International Conference on Learning Representations (ICLR)*, 2025.

[56] 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.

[57] P. Budzianowski, E. Wiśnios, G. Góral, I. Kulakov, V. Petrenko, and K. Walas, "Opengvl: Benchmarking visual temporal progress for data curation," *arXiv preprint arXiv:2509.17321*, 2025.

[58] S. K. S. Ghasemipour, A. Wahid, J. Tompson, P. R. Sanketi, and I. Mordatch, "Self-improving embodied foundation models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[59] 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.

[60] J. Zhang, C. Qian, H. Sun, H. Lu, D. Wang, L. Xue, and H. Liu, "Progresslm: Towards progress reasoning in vision-language models," *arXiv preprint arXiv:2601.15224*, 2026.

[61] 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.

[62] 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.

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

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

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

[70] 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.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

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

[79] 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.

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

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

[88] 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.

[89] Y. Dai, J. Lee, N. Fazeli, and J. Chai, "Racer: Rich language-guided failure recovery policies for imitation learning," in *International Conference on Robotics and Automation (ICRA)*, 2025.

[90] T. Yu, D. Quillen, Z. He, R. Julian, A. Narayan, H. Shively, A. Bellathur, K. Hausman *et al.*, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2019.

[91] T. Jiang, T. Yuan, Y. Liu, C. Lu, J. Cui, X. Liu, S. Cheng, J. Gao *et al.*, "Galaxea open-world dataset and g0 dual-system vla model," *arXiv preprint arXiv:2509.00576*, 2025.

[92] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang *et al.*, "Molmoact: Action reasoning models that can reason in space," *arXiv preprint arXiv:2508.07917*, 2025.

[93] Z. Zhao, H. Jing, X. Liu, J. Mao, A. Jha, H. Yang, R. Xue, S. Zakharor *et al.*, "Humanoid everyday: A comprehensive robotic dataset for open-world humanoid manipulation," *arXiv preprint arXiv:2510.08807*, 2025.

[94] M. Hwang, J. Hejna, D. Sadigh, and Y. Bisk, "Motif: Motion instruction fine-tuning," *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[95] R.-Z. Qiu, S. Yang, X. Cheng, C. Chawla, J. Li, T. He, G. Yan, D. J. Yoon *et al.*, "Humanoid policy˜ human policy," *arXiv preprint arXiv:2503.13441*, 2025.

[96] S. Xie, H. Cao, Z. Weng, Z. Xing, H. Chen, S. Shen, J. Leng, Z. Wu *et al.*, "Human2robot: Learning robot actions from paired human-robot videos," *arXiv preprint arXiv:2502.16587*, 2025.

[97] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin *et al.*, "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," in *Robotics: Science and Systems (RSS)*, 2025.

[98] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[99] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine, "Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world," *arXiv preprint arXiv:2503.24278*, 2025.

[100] A. Inceoglu, E. E. Aksoy, A. C. Ak, and S. Sariel, "Fino-net: A deep multimodal sensor fusion framework for manipulation failure detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[101] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman *et al.*, "Rt-1: Robotics transformer for real-world control at scale," in *Robotics: Science and Systems (RSS)*, 2023.

[102] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers *et al.*, "Bridgedata v2: A dataset for robot learning at scale," in *Conference on Robot Learning (CoRL)*, 2023.

[103] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[104] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "BC-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning (CoRL)*, 2021.

[105] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar, "Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking," *arXiv preprint arXiv:2309.01918*, 2023.

[106] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation," in *Robotics: Science and Systems (RSS)*, 2023.

[107] R. Shah, R. Martín-Martín, and Y. Zhu, "MUTEX: Learning unified policies from multimodal task specifications," in *Conference on Robot Learning (CoRL)*, 2023.

[108] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine, "Multi-stage cable routing through hierarchical imitation learning," *arXiv preprint arXiv:2307.08927*, 2023.

[109] S. Dass, J. Yapeter, J. Zhang, J. Zhang, K. Pertsch, S. Nikolaidis, and J. J. Lim, "Clvr jaco play dataset," 2023.

[110] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik, "Robot learning with sensorimotor pre-training," *arXiv preprint arXiv:2306.10007*, 2023.

[111] G. Zhou, V. Dean, M. K. Srirama, A. Rajeswaran, J. Pari, K. Hatch, A. Jain, T. Yu *et al.*, "Train offline, test online: A real robot learning benchmark," in *International Conference on Robotics and Automation (ICRA)*, 2023.

[112] S. Saxena, M. Sharma, and O. Kroemer, "Multi-resolution sensing for real-time control with vision-language models," in *Conference on Robot Learning (CoRL)*, 2023.

[113] S. Belkhale, Y. Cui, and D. Sadigh, "Hydra: Hybrid robot actions for imitation learning," in *Conference on Robot Learning (CoRL)*, 2023.

[114] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, "Real-world robot learning with masked visual pre-training," in *Conference on Robot Learning (CoRL)*, 2022.

[115] X. Zhu, R. Tian, C. Xu, M. Ding, W. Zhan, and M. Tomizuka, "Fanuc manipulation: A dataset for learning-based manipulation with fanuc mate 200id robot," 2023.

[116] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," in *Conference on Robot Learning (CoRL)*, 2024.

[117] G. Yan, K. Wu, and X. Wang, "Ucsd kitchens dataset," August 2023.

[118] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[119] J. Vogel, A. Hagengruber, M. Iskandar, G. Quere, U. Leipscher, S. Bustamante, A. Dietrich, H. Hoeppner *et al.*, "Edan - an emg-controlled daily assistant to help people with physical disabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[120] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel, "Shared Control Templates for Assistive Robotics," in *International Conference on Robotics and Automation (ICRA)*, 2020.

[121] T. Osa, "Motion planning by learning the solution manifold in trajectory optimization," *The International Journal of Robotics Research*, 2022.

[122] S. Haldar, V. Mathur, D. Yarats, and L. Pinto, "Watch and match: Supercharging imitation with regularized optimal transport," in *Conference on Robot Learning (CoRL)*, 2023.

[123] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster *et al.*, "OpenVLA: An open-source vision-language-action model," in *Conference on Robot Learning (CoRL)*, 2024.

[124] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding *et al.*, "Qwen3-vl technical report," *arXiv preprint arXiv:2511.21631*, 2025.

[125] R. Hoque, P. Huang, D. J. Yoon, M. Sivapurapu, and J. Zhang, "Egodex: Learning dexterous manipulation from large-scale egocentric video," *arXiv preprint arXiv:2505.11709*, 2025.

[126] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang *et al.*, "Qwen2. 5-vl technical report," *arXiv preprint arXiv:2502.13923*, 2025.

[127] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, p. 324, 1952.

[128] W. Monroe, R. X. Hawkins, N. Goodman, and C. Potts, "Colors in context: A pragmatic neural model for grounded language understanding," *Transactions of the Association for Computational Linguistics (TACL)*, 2017.

[129] C. Mitra, A. Anwar, R. Corona, D. Klein, T. Darrell, and J. Thomason, "Which one? leveraging context between objects and multiple views for language grounding," in *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

[130] Y. Bao, S. Ghosh, and J. Chai, "Learning to mediate disparities towards pragmatic communication," *arXiv preprint arXiv:2203.13685*, 2022.

[131] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang *et al.*, "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[132] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2024.

[133] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[134] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018.

[135] Y. Huang, J. Song, Z. Wang, S. Zhao, H. Chen, F. Juefei-Xu, and L. Ma, "Look before you leap: An exploratory study of uncertainty measurement for large language models," *arXiv preprint arXiv:2307.10236*, 2023.

[136] M. Hong, A. Liang, K. Kim, H. Rajaprakash, J. Thomason, E. Bıyık, and J. Zhang, "Hand me the data: Fast

robot adaptation via hand path retrieval," in *International Conference on Robotics and Automation (ICRA)*, 2026.

[137] C. Agia, R. Sinha, J. Yang, R. Antonova, M. Pavone, H. Nishimura, M. Itkina, and J. Bohg, "Cupid: Curating data your robot loves with influence functions," in *Conference on Robot Learning (CoRL)*, 2025.

[138] A. Xie, R. Chand, D. Sadigh, and J. Hejna, "Data retrieval with importance weights for few-shot imitation learning," in *Conference on Robot Learning (CoRL)*, 2025.

[139] Y. Zhang, Y. Xie, H. Liu, R. Shah, M. Wan, L. Fan, and Y. Zhu, "Scizor: Self-supervised data curation for large-scale imitation learning," in *International Conference on Robotics and Automation (ICRA)*, 2026.

[140] J. Hejna, S. Mirchandani, A. Balakrishna, A. Xie, A. Wahid, J. Tompson, P. Sanketi, D. Shah *et al.*, "Robot data curation with mutual information estimators," in *Robotics: Science and Systems (RSS)*, 2025.

[141] M. Du, S. Nair, D. Sadigh, and C. Finn, "Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets," in *Robotics: Science and Systems (RSS)*, 2023.

[142] L.-H. Lin, Y. Cui, A. Xie, T. Hua, and D. Sadigh, "Flowretrieval: Flow-guided data retrieval for few-shot imitation learning," in *Conference on Robot Learning (CoRL)*, 2024.

[143] A. Liang, I. Singh, K. Pertsch, and J. Thomason, "Transformer adapters for robot learning," in *CoRL 2022 Workshop on Pre-training Robot Learning*, 2022.

[144] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations (ICLR)*, 2022.

[145] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," in *arXiv preprint arXiv:1910.00177*, 2019.

Appendix Table of Contents

Appendix A
Dataset Details

Paper	**# Trajectories**
RoboReward [11]	45k
RoboDopamine [12]	100k
VLAC [10]	300k
RBM-1M (ours)	1M

TABLE VI: Comparison of known approximate trajectory counts across recent general-purpose reward modeling papers.

1. *Individual RBM-1M Training Dataset Details*

Here, we list dataset curation details for every source in RBM-1M.

Preference Only Data. Some data sources vary significantly in when the teleoperator marks the end of the trajectory. As such, using these data sources for progress prediction and end-state success prediction negatively affects prediction accuracy. To avoid this issue, progress-only generalist robotic reward models would have to either manually label end states or simply forgo using these data sources. However, due to Robometer's trajectory comparison-based preference prediction objective, we can still use these noisier datasets for preference prediction; we list which sources are used only for preference in each subsection.

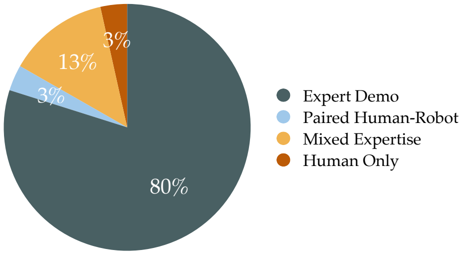


Fig. 10: Pie chart of RBM-1M dataset types. Full table with individual dataset details in Table IX.

Image Resolution and Frame Downsampling. We first perform data pre-processing before using the data for training. For storage efficiency, we downsample all trajectories to a maximum of 32 frames. Furthermore, we downsize image resolution such that the shortest edge (either height or width) of each image is 240 pixels. This procedure allows us to maintain the aspect ratio while ensuring the dataset does not consume excessive storage space. In total, the dataset size is around 6 TB.

Task End Thresholds. For predicting both progress and success (Section III-C), it's important that the end-frame of the trajectory corresponds to the timestep at which the task is actually finished. However, in most real-world datasets, this is not the case due to human teleoperator delay or different notions of when tasks are finished. We address this issue by manually setting this "task finished" threshold for each data source, which takes about 2 minutes per data source. See Section A-2 for further details.

1) Expert Demonstration Datasets: We start with expert demonstration datasets, i.e., datasets that contain only successful trajectories demonstrated by competent teleoperators. For all of these datasets, the progress target we use in Section III-C is 1.0.

a) Open-X Embodiment Mix: We select a subset of datasets from the Open-X Embodiment (OXE) Dataset [62]. This subset was chosen from those selected for reward learning from prior work [5]. The subset includes DROID [76], Fractal [101], BRIDGE-v2 [102], Language Table [103], BC-Z [104], RoboSet [105], FurnitureBench [106], UT Austin Mutex [107], Berkeley Cable Routing [108], CLVR Jaco Play [109], Berkeley RPT [110], Toto [111], CMU Franka Pick-Insert [112], Stanford Hydra [113], Berkeley MVP [114], Berkeley Fanuc Manipulation [115], Mobile ALOHA [116], Imperial College Sawyer Wrist Cam, UCSD Kitchen [117], Austin BUDS [118], DLR Edan [119, 120], UTokyo LSMO [121], and NYU Rot [122].

We use wrist camera and external camera viewpoints from DROID due to the wide-angle of the DROID wrist cameras.

Dataset	Embodiment	# Trajectories	Citation
OXE Eval Suite	Franka, Google Robot, Jaco 2, WidowX	14,399	[62]
RoboReward Test	12 mixed embodiments	9,063	[11]
RACER (Eval)	Franka Panda (simulation)	7,227	[89]
Metaworld Eval	Sawyer	151	[90]
LIBERO Failure Eval	Franka Panda (simulation)	1,927	–
Total	13 unique robot embodiments	32,767	–

TABLE VII: **RBM-EVAL-ID** - In distribution evaluation datasets overview. # Trajectories is computed by aggregating all evaluation splits per dataset, counting unique annotated trajectories.

Dataset	Embodiment	# Trajectories
USC Franka	Franka Panda	24
USC Koch	Bimanual Kochv1.1	407
USC Trossen	Trossen	27
USC xArm	xArm	36
MIT Franka	Franka Panda	152
UTD;' SO101 Cluttered	SO101	60
UTD;' SO101 Clean (Top)	SO101	30
UTD;' SO101 Clean (Wrist)	SO101	30
Total	5 unique robot embodiments	976

TABLE VIII: **RBM-EVAL-OOD** - Out-of-distribution evaluation datasets overview. # Trajectories is computed by aggregating all evaluation splits per dataset, counting unique annotated trajectories.

For most other datasets, we only use external cameras unless wrist is the only viewpoint available.

Some of these datasets have validation or test-set splits; we use these splits for RBM-EVAL-ID.

For BC-Z and DLR Edan, we use these datasets only for preference prediction due to highly varied trajectory termination times relative to when the task was actually completed.

b) AGIBot World: The AgiBotWorld-Alpha dataset [63] consists of 100k+ long-horizon trajectories on the AgiBot G1 bimanual mobile manipulator. We randomly select a 34,098 trajectory subset of the 100k trajectories to form our dataset. Each long-horizon trajectory is annotated with each shorter-horizon subskill that is required to accomplish the task. In total, this leads 216,911 long-horizon and short-horizon trajectories. These short and long horizon skills include higher-level tasks such as "put all the oranges in the basket" and the lower-level skills for each "pick orange" and "place orange." Although the dataset includes wrist cameras and multiple fisheye camera angles, we only include the egocentric head camera in our dataset.

c) Galaxea Open-World: The Galaxea Open-World Dataset [91] is a large-scale humanoid dataset with 108,118 trajectories across 150 task categories. The dataset includes diverse tasks from pick-and-place to whole-body manipulation on a Galaxea R1 Lite bimanual mobile manipulator. Due to highly varied trajectory termination times relative to when the task was actually completed, we use this dataset only for preference prediction.

d) RoboReward OXE + Roboarena Mix: In addition to our own OXE mix, we incorporate the 45k trajectory OXE + Roboarena [61] training subset from RoboReward. This dataset is labeled using RoboReward's VLM-based counterfactual instruction labeling technique to generate pseudo-failure instructions for successful trajectories. This technique complements Robometer's training objectives — by directly incorporating all data into Robometer's training mix and objectives. Final rewards for each trajectory are discrete numbers ranging from 1 to 5. We normalize their rewards to the range of $[0, 1]$, making this data suitable for all of our prediction objectives. We refer readers to Lee et al. [11] for further details. Similar to the OXE dataset mix, the DLR Edan and BC-Z subsets of the RoboReward data is also only used for preference prediction due to highly varied trajectory termination times relative to when the task was actually completed.

e) MolmoACT: We use all external-view data from MolmoACT [92], collected on a Franka Panda arm, excluding trajectories with corrupted videos. We do not use wrist-cam data from MolmoACT because the camera angle is too narrow and often does not show the object being manipulated. We selected this dataset because it includes a diverse set of trajectories collected in clutter with good camera visibility.

f) Humanoid Everyday: We use all data from [93], which contains Unitree G1 & H1 bimanual humanoid data from an egocentric viewpoint. This data was selected for diversity as it includes bimanual mobile data.

g) LIBERO: LIBERO [66] provides a diverse set of simulated household manipulation tasks across 5 task suites. We use LIBERO-{10, Object, Spatial, Goal} for RBM-1M and LIBERO-90 for evaluation in RBM-EVAL-ID. We follow OpenVLA's [123] dataset re-generation scheme by re-rendering at 256x256, removing no-ops, and removing demonstrations which are not successful upon replay. We selected LIBERO for its use as a popular VLA benchmark and for the ease with which we can generate our own failed trajectories across a diverse set of tasks. Thus, we also include a corresponding set of failed trajectories for all 5 task suites, constructed by replaying demonstration trajectories with added

Dataset	Embodiment	# Trajectories	Citation
Expert Demonstration Datasets			
OXE Mix	11 mixed embodiments	449,475	[62]
AGIBotWorld-Alpha Subset	AgiBot G1 Bimanual Mobile Manipulator	216,911	[63]
Galaxea Open World	Galaxea R1 Lite Bimanual Mobile Manipulator	108,118	[91]
RoboReward OXE Mix	13 mixed embodiments	45,072	[11]
MolmoACT	Franka Panda	15,546	[92]
Humanoid Everyday	Unitree G1 & H1 Bimanual Humanoids	9,208	[93]
LIBERO	Franka Panda (simulation)	1,709	[66]
MetaWorld (ReWiND)	Sawyer (simulation)	100	[5, 90]
Paired Human–Robot Datasets			
MotIF	Stretch Mobile Manipulator	83	[94]
RH-20T	4 embodiments	29,969	[65]
PH2D	H1 (partly in MuJoCo simulation)	3,596	[95]
H2R	XArm	2,254	[96]
Mixed Expertise Datasets			
RoboArena Pairwise Comparisons	Franka Panda	12,379	[61]
SOAR Paired Success/Fail	WidowX	16,812	[67]
FAILSafe	Franka Panda (ManiSkill [97] simulation)	71,614	[68]
RACER Failure Dataset	Franka Panda (RLBench [98] simulation)	29,115	[89]
AutoEval Failed Trajectories	WidowX250	8,677	[99]
LIBERO Failure Dataset	Franka Panda (simulation)	1,473	[66]
Fino-Net Paired Failure (Baxter)	Baxter	229	[100]
Human-Only Datasets			
Epic-Kitchens	Human	37,030	[64]
Total	21 unique robot embodiments	1,059,370	–

TABLE IX: **RBM-1M** Training datasets overview. # Trajectories is determined by counting every unique video-language annotation (possibly across multiple views when available).

Gaussian noise to each action, mimicking policy execution error that results in failure.

h) Metaworld: MetaWorld [90] is a multi-task simulated manipulation benchmark with a Sawyer arm. We use the 20-task training split consisting of 5 demonstrations each from Zhang et al. [5] for RBM-1M. Correspondingly, we use the 17-task evaluation dataset from Zhang et al. [5] for RBM-EVAL-ID. MetaWorld was selected early on for basic testing and ensuring that we can reproduce the results from ReWiND [5] in our own implementation of it. We kept it in RBM-1M for visual feature diversity.

2) Paired Human-Robot Datasets:

a) MotIF: A human-robot paired dataset with a Stretch Mobile Manipulator containing tasks involving motion-counting such as "stir 3 times" or shaking boba tea [94]. This dataset was selected for these counting tasks to encourage learning to track repetitive motions.

b) RH-20T: A paired human-robot dataset for table-top manipulation spanning four robot embodiments including Flexiv, UR5, Franka Panda, and Kuka robots [65]. This dataset was selected for its diversity in tasks and embodiments. The RH20T dataset consists of 7 configurations each with its own robot, table and camera setup. For each configuration, we select 1–2 camera views which both 1) capture the full scene and robot motion and 2) are consistent with the language instruction in terms of spatial relationships, e.g., left, right, top, bottom, etc. We removed null robot demonstrations without any arm movements, as well as demonstrations which seem to be the concatenation of demonstrations for multiple tasks.

c) PH2D: A human-robot, real and simulation dataset containing Unitree H1 trajectories collected from an egocentric viewpoint [95]. This dataset was selected because it pairs simulation, real, and human trajectories; it contains many pouring tasks and many very detailed task descriptions.

d) H2R: A human-robot paired dataset with a UFACTORY XARM robot [96]. It contains pick-and-place and pushing tasks. We selected it because many tasks have multiple objects in the scene (e.g., a lighter tray and a darker tray to place a cube on), helping the reward model learn to better distinguish *correctly* manipulated objects.

3) Mixed Expertise Datasets: The data here contains paired successful and failed trajectories. We incorporate these mixed expertise data to encourage ROBOMETER to effectively reward failed trajectories, which can help in a variety of domains (e.g., all downstream experiments in Section IV-Q3).

a) RoboArena: Roboarena [61] data is from a set of human-performed evaluations of various generalist VLA policies on the DROID setup with a Franka Panda arm. Each evaluation has a partial progress score $\in [0, 1]$ with the vast majority of evaluations being failures. We use these progress scores solely to construct trajectory comparisons to predict over as trajectory termination times are highly varied and can essentially undo progress made in the middle of the trajectory, for which the human gave a partial progress score. We save videos from all available camera viewpoints, including wrist camera.

b) SOAR: SOAR [67] data comes from autonomous policy rollouts guided by a VLM on a WidowX250 robot. Success/fail labels, generated by a VLM, are also provided. Due to automatic task generation and success/fail labeling, the dataset labels are quite noisy, and many trajectories contain tasks that are not possible in the scene, tasks that have already started from the first frame, and incorrect success/fail classifications. Therefore, we use a pre-trained Qwen-3-VL-4B [124] to filter out incorrectly labeled samples, including ones that were infeasible or unrelated to the task description. We do this by using the first, middle, and last frame to establish a general flow of the trajectory and ask the model to critique the positioning of relevant items. This is separated into a stage prompt that empirically improved filtering quality using a small, manually verified set and ultimately filters out 45% of trajectories from the original dataset.

We save all filtered successful trajectories, and we also save all failed trajectories that have the same language description as at least one successful trajectory. Because this data does not contain progress labels and because trajectory end frames are highly variable with respect to when the task was actually completed in successful trajectories, we use the entire dataset only for generating trajectory comparisons for preference prediction.

c) FAILSafe: FAILSafe [68] contains successful and failed trajectories from a Franka Panda collected in the Maniskill simulator [97]. Each task has many example failures collected from both wrist and external cameras. Tasks are also segmented into sub-tasks, e.g., reaching a cube → grasping the cube → ...

d) RACER: RACER contains paired failed and successful trajectories on a Franka Panda in the RLBench simulator [89, 98]. This dataset was picked for its non-prehensile tasks, such as opening/closing drawers and sweeping.

e) AutoEval: AutoEval contains data from automatic policy evaluations collected on 2 different WidowX250 setups [99]. While task diversity is limited, we used this dataset because it contains diverse strategies coming from the evaluation of arbitrary policies.

This dataset is used only for preference prediction due to some noisy automatic success/fail detection.

f) LIBERO: We self-generated a failure dataset in LIBERO [66] by adding Gaussian noise to successful demonstration trajectories to re-generate paired failure trajectories for every task. This data is paired with the original success-only LIBERO dataset mentioned in Section A-11.

g) Fino-Net: Fino-Net data contains egocentric, paired success/fail data from a Baxter robot [100]. The data consists mainly of pick-and-place tasks, but it was selected for its use of a unique robot not present in the other datasets.

4) Human only Datasets: Finally, we also include human-only data. Our final dataset contains just one dataset, Epic-Kitchens, but early on we also experimented with EgoDex [125]. We found that training solely on Epic-Kitchens helped predict rewards for robot data, but this was not the case with EgoDex, perhaps due to Epic-Kitchen's background scene diversity and clutter.

a) Epic-Kitchens: We include a subset of data from Epic-Kitchens [64]. Due to some difficulties we encountered in downloading the entire dataset, we picked a subset of Epic-Kitchens 100 uploaded to HuggingFace Datasets.[1]

Furthermore, because language annotations and trajectory end times are quite noisy, we use the Epic-Kitchens dataset only for preference prediction.

2. Dataset Filtering and Task End-State Adjustment

To determine task-finished thresholds for each dataset, we designed a lightweight UI to visualize randomly sampled trajectories from each data source. With this visualizer, we sample 10 trajectories from each data source and manually mark the frame at which we deem the task to be complete. We define this threshold as the point at which the task description is satisfied. Then, we use the 90th percentile of end-frame thresholds (i.e., when 90% of the visualized trajectories are complete) as our threshold. Most teleoperators collecting data in `RBM-1M` define the trajectory as complete when the robot has performed a partial or full reset to neutral. As such, our end-state thresholds are typically set to around 80-95% of the trajectory length.

This process takes no more than 2 minutes per data source, and the thresholds are used to appropriately adjust target progress and success thresholds for training, detailed further in Section B-2.

3. Individual Evaluation Dataset Details

We summarize the number of trajectories in each of our in-distribution evaluation dataset in Table VII and the out-of-distribution evaluation dataset in Table VIII and describe them in detail below.

a) USC Franka: USC Franka is a dataset of mixed expertise trajectories collected between four different tabletop tasks such as "fold towel" and "put the plate on the sink." For each quality label, we collected at least two trajectories each.

b) USC Trossen: USC Trossen comprises of mixed expertise trajectories collected between five different tabletop tasks using a bimanual Trossen. Some tasks are more articulated and dexterous such as "unzip the pencil case" and "stir the pot". For each quality label, we collected at least two trajectories each.

c) USC Koch Arms: USC Koch Arms replicates the real-world data collected in ReWiND [5] using bimanual Koch Arms. This dataset consists of 10 demos per-task over 20 tasks. We also collect suboptimal and failure examples for each task.

d) MIT Franka: MIT Franka is a dataset composed of diverse tasks, including pick and place ("pick up the banana from X and place it on Y") and dexterous tasks such as "fold the towel in half", "pick up the spatula and stir the beans in the pot". We also include a task "pick up the mouse and place it on X while avoiding Y" that requires semantic scene understanding. We collect trajectories of different levels of expertise for each task.

e) UTD;' SO-101: Univ3 SO-101 is a real-world dataset of manipulation trajectories collected using a single-arm SO-101 robot. The dataset comprises two mixed-quality settings, each containing both successful and failure trajectories. The first setting is a clean pick-and-place environment centered on a single household task, "put the bread in the oven."

[1] https://huggingface.co/datasets/awsaf49/epic_kitchens_100

TABLE X: **Training Configuration for ROBOMETER**

Parameter	**Value**
Base Model	`Qwen/Qwen3-VL-4B-Instruct`
Number of frames	8
Per-device batch size	16
Learning rate	2×10^{-5}
Weight decay	0.01
Total training steps	6500
Max sequence length	1024
LR scheduler	Cosine
Warmup ratio	0.1
Min frames per trajectory	5
Progress loss type	Discrete (C51 style)
Number of discrete bins	10
MLP head num hidden layers	1
MLP head dropout	0.1
MLP head hidden dim	2048

This setting includes 30 successful demonstrations and 45 failure trajectories. The second setting is a cluttered multitask environment consisting of three pick-and-place tasks, such as "put the red bowl on the blue plate." For each task in this setting, we collect 20 successful demonstrations and 15 failure trajectories.

APPENDIX B
MODEL DETAILS

1. *Model Architecture and Training Parameters*

a) Architecture: We illustrate the overall architecture of ROBOMETER in Figure 11. ROBOMETER instantiates a causally masked vision–language model (VLM) backbone, QWEN3-VL-4B-INSTRUCT, a unified transformer that processes interleaved text and visual tokens using a single autoregressive decoder. Natural language instructions are tokenized using Qwen's SentencePiece-based tokenizer.

Each video trajectory is first subsampled into a fixed number of frames. Each frame is independently encoded by a ViT-style visual encoder and projected into a sequence of visual tokens. A special ⟨|video_start|⟩ token marks the beginning of visual input, after which visual tokens are appended sequentially and assigned unique positional indices in the unified token sequence. The decoder jointly attends over language tokens, visual tokens, and special tokens using a single causal attention mask, supporting unified multimodal reasoning.

To enable dense, frame-level reward estimation, a learned progress token ⟨|prog_token|⟩ is inserted after each frame o_t^1 in the first trajectory. Under causal masking, the hidden state of ⟨|prog_token|⟩ can attend only to the instruction and visual tokens from frames $o_{1:t}^1$.

We introduce a dedicated separator token ⟨|split_token|⟩ to demarcate the boundary between two video trajectories. The second trajectory o^2 is appended after ⟨|split_token|⟩ and processed jointly with o^1 in the same causal sequence. A learned preference token ⟨|pref_token|⟩ is appended at the end of the prompt; its hidden state aggregates information from the instruction and both trajectories and is used to predict pairwise preference.

We attach lightweight MLP heads to the shared VLM backbone. Specifically, the progress, preference, and success heads each consist of a two-layer MLP followed by LayerNorm, GELU activation, and dropout, and a final linear projection that outputs a scalar logit. The progress head is applied to the hidden states of the interleaved ⟨|prog_token|⟩ tokens to produce frame-level progress logits, while the success head operates on the corresponding per-frame hidden states to predict frame-level success logits. The preference head is applied to the hidden state of ⟨|pref_token|⟩ to produce a single logit indicating whether the first trajectory is preferred over the second.

For preference supervision, we construct a single multimodal prompt that contains two trajectories, serialized into a single causal sequence and separated by a split token. This is a more detailed, expanded version of Equation (1).

$$\begin{aligned}\mathrm{Tok}(l, o^A, o^B) \rightarrow\ & \mathrm{Tok}(l)\ \text{This is Trajectory A.''}\ \langle|\text{video\_start}|\rangle\ \mathrm{Tok}(o^A_{1:T})\\ & \langle|\text{split\_token}|\rangle\ \text{This is Trajectory B.''}\\ & \langle|\text{video\_start}|\rangle\ \mathrm{Tok}(o^B_{1:T})\ \langle|\text{pref\_token}|\rangle.\end{aligned}$$

Prompt. Since we train the model on all 3 objectives (progress, success, preference) simultaneously, we always sample a preference prompt. Thus, we always condition the model on the following natural-language prompt.

> Given these two trajectories for the task "{`task`}", evaluate which one makes more progress towards the task. Return A for the first trajectory and B for the second trajectory. Additionally, predict the task progress at each frame of the first trajectory as a float between 0 and 1, where 0 corresponds to the initial state and 1 corresponds to task completion. If the robot is not performing the specified task, predict 0 progress.

b) Model and Training Params: We list overall hyperparameters in Table X. We did not extensively sweep these hyperparameters—we followed best practices and parameters from prior work [11, 124, 126]. For the preference, success, and progress prediction MLPs, we heuristically select a hidden dimension of 2048, which is half the input size (Qwen's hidden embedding size) of 4096.

2. *Training Objectives*

a) Trajectory Cutoffs and Success Supervision: In several teleoperated datasets, episode termination does not coincide with task completion. Operators typically complete the task and then manually stop the recording after a short delay, resulting in trailing frames in which the robot remains static or performs incidental motions unrelated to task execution. These frames do not reflect additional task progress and introduce noise when used for frame-level progress supervision. To mitigate this issue, we manually annotate a dataset-specific success cutoff corresponding to the frame at which the task objective is first achieved and apply this cutoff uniformly to all trajectories within the dataset (see Table XI).

For frames occurring after the dataset-specific success cutoff, we assign target progress and success labels of 1.0, reflecting that the task has already been completed. Success

Dataset	Success Cutoff
Open-X Embodiment (OXE)	
Aloha Mobile	0.95
Austin BUDS	0.95
Berkeley Cable Routing	0.95
Berkeley FANUC Manipulation	0.98
Bridgev2	0.95
DLR EDAN Shared Control	0.95
CMU Pickup Insert	0.95
UCSD Kitchen	0.95
UT Austin Mutex	0.95
BC-Z	0.95
Berkeley MVP	1.00
Berkeley RPT	0.76
Fractal	1.00
Furniture Bench	1.00
DROID	0.95
Imperial College Sawyer Wrist Cam	0.90
Language Table	1.00
NYU ROT	0.70
RoboSet	0.85
Stanford HYDRA	1.00
Tokyo-U LSMO	1.00
TOTO	1.00
Other Datasets	
MolmoAct Household	0.94
MolmoAct Tabletop	0.94
AgibotWorld	0.95
PH2D	0.95
RH20T (Human)	0.92
RH20T (Robot)	0.94
RoboArena	0.90
H2R	0.90
SOAR	0.95
AutoEval	0.94
Galaxea	0.80
FINO-Net	0.75
Humanoid Everyday	0.80
Motif	0.95

TABLE XI: **Dataset-specific success cutoffs** used to correct for delayed episode termination in teleoperated data. Datasets not contained here use a default success cutoff of 1.0.

supervision is applied selectively to avoid ambiguous or conflicting learning signals. Specifically, we predict success only for frames whose target progress is either strictly below a minimum success threshold τ_{succ}, corresponding to clearly pre-completion states, or exactly equal to 1.0, corresponding to completed states. Frames with intermediate progress values near completion are excluded from success supervision, as they often correspond to visually ambiguous transitional or stabilization phases.

b) Progress Supervision in Preference Samples: For preference samples, we apply progress supervision only to the first trajectory (Trajectory A), reflecting the fact that progress prediction is used at inference time for a single video. To avoid introducing noisy or ill-defined targets, we compute progress loss for Trajectory A only when it corresponds to a successful trajectory. When Trajectory A corresponds to a failed or suboptimal execution, we do not apply progress supervision.

For datasets that provide continuous partial completion annotations, such as RoboArena, we supervise progress on the final frame of Trajectory A using the ground-truth `partial_success` value, even when the trajectory is suboptimal. This allows the model to learn calibrated progress estimates from human-annotated partial completion labels.

c) Data Source and Strategy Sampling: Detailing Section III-D further, we construct pairs of trajectory comparisons for preference supervision (and also progress/success supervision with the first trajectory input to the model) by first sampling a *preference sampling strategy* from *different-task*, *rewind augmentation*, or *different-expertise*. Conditioned on the selected strategy, we restrict sampling to datasets from which the corresponding type of preference pair can be constructed.

- For *different-expertise* strategy, samples are constructed from mixed expertise datasets, such as RACER or FINO-Net, where trajectories are labeled as successful, suboptimal, or failed. We use these annotations to form preference pairs that rank trajectories by execution quality. In datasets such as RoboArena that additionally provide continuous `partial_success` annotations, we sample two trajectories from the same task and assign the trajectory with higher partial success as the preferred one.
- *Rewind augmentation* can be applied to trajectories from any dataset and does not require additional annotations.
- For *different-task pairing*, we sample a trajectory that is successful for one task and pair it with a trajectory executing a different task. When constructing different-task pairs, we control whether both trajectories are drawn from the same data source or from different sources. With probability ρ_{same}, we sample different-task pairs from the same dataset to discourage reliance on dataset-specific visual cues, while with probability $1 - \rho_{\text{same}}$ the trajectories are drawn from different datasets to encourage robustness to domain shift. We set $\rho_{\text{same}} = 0.5$, yielding an equal mix of same-source and cross-source different-task comparisons.

d) Preference Prediction Loss: Bradley-Terry vs BCE: Our preference prediction loss in Equation (2) uses a binary cross-entropy loss coming from an MLP on top of the $\langle|\text{pref\_token}|\rangle$ embedding in Equation (1). Using this token with *two* input videos allows ROBOMETER to simultaneously attend to *both* videos to predict this loss, resulting in one forward and backward pass to train ROBOMETER on preference prediction.

A common alternative in both reward modeling from pseudo-preferences [4] and general RLHF reward function training on real human preferences [25] is to instead use the Bradley-Terry loss [25, 127], where a single preference score is computed over an entire individual trajectory at a time, and then the loss is backpropagated across batch comparisons.

Ignoring computational efficiency differences, our main reason for not implementing the Bradley–Terry loss into ROBOMETER is that it does not leverage the pre-trained attention mechanism as effectively: our two-video formulation

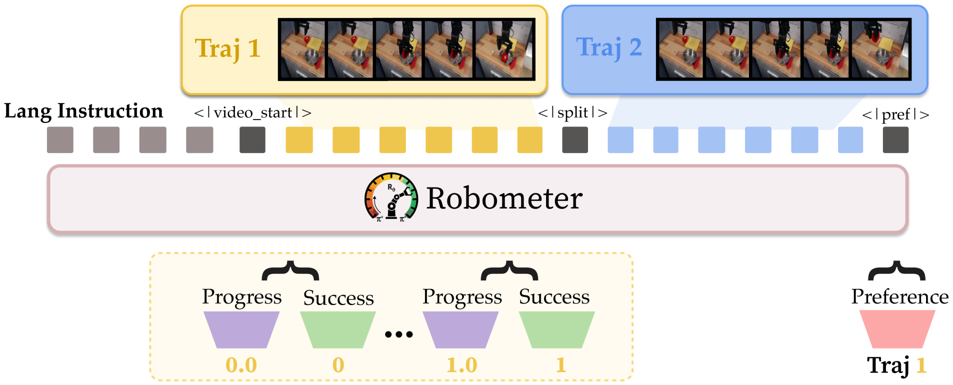


Fig. 11: **Robometer model architecture.** For a given task, our VLM based model takes in language description and two trajectories 1 and 2 separated by split tokens. The vlm output for trajectory 1 is fed into two MLP heads : progress - task completion percent and success - task completion probability. Finally the full output is passed into a preference MLP to choose which trajectory best completes the provided task.

Pref. Loss	VOC $r\uparrow$	Kendall $\tau\uparrow$	Succ–Fail Diff. ↑
BT	0.862	0.325	0.242
BCE	**0.948**	**0.655**	**0.320**

TABLE XII: **Bradley–Terry (BT) versus preference label** from a dedicated ⟨|pref_token|⟩ (BCE) given both videos in a single forward pass, evaluated on `RBM-EVAL-OOD`.

allows tokens from one trajectory to explicitly attend to tokens from the other, enabling direct cross-video comparison during preference prediction. This strategy is common for comparative reasoning tasks in language reference games [128, 129, 130]. In contrast, Bradley–Terry computes independent scalar scores per trajectory and only couples them through the loss, which can make learning more sensitive to score calibration across batches. We compare these objectives in Appendix D. As shown in Table XII, predicting a preference label from a dedicated ⟨|pref_token|⟩ given both videos (BCE) improves reward alignment (from 0.862 to 0.948) and trajectory ranking over Bradley–Terry (from 0.325 to 0.655).

3. *Computational Resources*

We train Robometer using a per-GPU batch size of 16 across 4 GPUs, resulting in an effective batch size of 64. All experiments are run on a server with jobs requesting 4 NVIDIA H200 GPUs and 32 CPU threads for 6.5k training steps, corresponding to approximately 2 days of wall-clock training time. Unless otherwise specified, we use the same training configuration across all experiments.

Appendix C
Additional Reward Evaluation Results

We first list baseline implementation details for main paper baselines before discussing additional reward evaluation results.

a) VLAC: VLAC released 2 models built on the InternVL [131] VLM, with 2B and 8B parameters. VLAC takes 2 frames as input and predicts the relative increase in progress between the first and second frames; thus, its output range is $[-1, 1]$. We directly perform inference using the publicly available GitHub code and test both the 2B and 8B models—we found the 8B model to perform slightly better on our evaluation results and thus use VLAC-8B as our baseline. The pretrained VLAC checkpoints are obtained from the authors' public release.[2] For relevant reward comparisons, we normalize its outputs to make them directly comparable to those of all other models that output only positive rewards.

b) Robo-Dopamine: trains a step-aware *general process reward model* (GRM) to estimate fine-grained manipulation progress from *multi-view* observations by predicting discretized progress "hops" between states and aggregating these signals into a dense reward. We use the authors' released inference code and GRM-3B checkpoint.[34] Robo-Dopamine relies on richer task-specific context at inference time (e.g. such as multi-view and goal images), which may not be consistently available across datasets or deployment settings.

c) GVL: We compare against GVL, which queries a closed-source vision-language model to estimate task progress directly from video frames. GVL does not involve model finetuning; instead, progress is inferred solely through prompting at inference time. In our experiments, we use `GPT-5-mini`,[5] which is the strongest-performing closed-source model reported in the RoboRewardBench [11] benchmark. To discourage the model from exploiting trivial temporal cues and to reduce correlations with frame index, GVL shuffles video frames during inference and prompts the model to predict progress values that are subsequently reordered chronologically.

d) ReWiND: We implement ReWiND [5], which operates on precomputed visual and language embeddings rather than raw pixels. Each video frame is encoded using a frozen DINOv2 vision encoder [132], and task instructions are embedded using the frozen Sentence-Transformers MiniLM-L6-v2 model (`all-MiniLM-L6-v2` [133]). The resulting visual and text embeddings are linearly projected into a shared latent space and processed by a Transformer encoder.

[2] https://huggingface.co/InternRobotics/VLAC
[3] https://github.com/FlagOpen/Robo-Dopamine
[4] https://huggingface.co/tanhuajie2001/Robo-Dopamine-GRM-3B
[5] `GPT-5-mini-2025-08-27` the latest version as of writing.

To enable a direct architectural comparison with Robometer, we predict per-frame progress and success directly from the hidden states corresponding to visual frame embeddings using lightweight MLP heads, and train progress using a discrete bin formulation. In our ablation study in Table IV, we further extend the ReWiND baseline with a preference objective by concatenating two trajectories into a single sequence and predicting a binary preference label via a learned preference token, and we scale up its original architecture (4 layers, 8 attention heads) to a hidden dimension of 1024, 32 transformer layers, and 16 attention heads.

e) RoboReward: is trained to predict a discrete 1-5 progress target on OXE and RoboArena. RoboReward is trained using trajectory-level supervision and does not model intermediate task progress within a trajectory. As a result, the model is primarily designed to provide a sparse, terminal reward. For fair evaluation and comparison with our dense reward formulation, we obtain frame-level rewards from RoboReward by running inference independently on each frame of a trajectory and treating the resulting predictions as per-frame rewards. We use the publicly released RoboReward-8B pretrained checkpoint provided by the authors.[6][7]

1. Preference Prediction

a) RL-VLM-F: RL-VLM-F [35] predicts trajectory preferences by prompting a closed-source vision–language model to compare the *final frame* of two trajectories conditioned on a task description. In our experiments, we instantiate RL-VLM-F using the same OpenAI `GPT-5-mini` model as for GVL. RL-VLM-F uses the following fixed prompt for preference prediction.

> Each frame comes from a robot trajectory. (Think causally and use image comparison to verify any confusion between the base of the robot and the end effector.)
>
> 1) What is shown in the first image (Image A)?
> 2) What is shown in the second image (Image B)?
> 3) For this question, here is the Goal Text: `GOAL_TEXT`
>
> Is the goal being better achieved in Image A or Image B? Reply with a single line containing `0` if the goal is better achieved in Image A, or `1` if the goal is better achieved in Image B. Reply `-1` if there is no discernible difference or progress.

For RL-VLM-F, preferences are inferred using only the final frame of each trajectory, whereas Robometer applies a learned preference head that directly compares full video sequences. Preference accuracy is computed as the fraction of pairwise comparisons in which the predicted ordering matches the ground truth. To evaluate preference quality, we construct trajectory pairs from the `RBM-EVAL-OOD` split along two axes: (i) differing task instructions and (ii) differing trajectory quality labels. For each dataset, we randomly sample 500 pairwise trajectory comparisons and evaluate predicted preferences against ground-truth ordering labels. As shown in Tables XIII and XIV, Robometer consistently outperforms RL-VLM-F, improving average preference accuracy by 27.0% on different-quality pairs and by 32.4% on different-task pairs.

Dataset	**RL-VLM-F (%)**	**Robometer (%)**
USC Franka	52.1	**75.0**
USC Koch	54.4	**79.4**
USC Trossen	66.7	**76.2**
USC xArm	48.6	**88.9**
MIT Franka	54.4	**85.4**
UTD;’ SO-101	56.7	**90.0**
Average	55.5	**82.5**

TABLE XIII: **RL-VLM-F vs Robometer**. Different quality trajectory pairwise preference accuracy on 500 comparisons for each of the `RBM-EVAL-OOD` datasets.

Dataset	**RL-VLM-F (%)**	**Robometer (%)**
USC Franka	70.7	**100.0**
USC Koch	54.7	**89.8**
USC Trossen	64.0	**99.0**
USC xArm	73.3	**98.2**
MIT Franka	55.3	**98.4**
UTD;’ SO-101	72.7	**100.0**
Average	65.1	**97.6**

TABLE XIV: **RL-VLM-F vs Robometer**. Different task trajectory pairwise preference accuracy on 500 comparisons for each of the `RBM-EVAL-OOD` datasets.

Appendix D
Additional Ablations

1. Preference pair sampling ablations

As shown in Table XVIII, we ablate our preference pair sampling strategies to isolate which parts of the data construction are responsible for *generalization*. We evaluate on held-out `RBM-EVAL-OOD` and the out-of-distribution LIBERO-90 benchmark, neither of which is used for training. We remove one strategy at a time from Section III-D to measure its contribution to held-out ranking and reward separation.

Across both held-out benchmarks, removing any component hurts performance, with the largest drop coming from disabling **trajectory rewinding**, which most strongly reduces success–failure separation and degrades trajectory ranking. **Different-task negatives** primarily affect instruction grounding: dropping them yields a consistent (though smaller) degradation in OOD ranking. Finally, **suboptimal trajectory** pairs help calibrate rewards across mixed-quality behavior; removing them reduces ranking quality and narrows the margin between successful and failed trajectories. Overall, each preference construction contributes complementary signal, and combining them yields the strongest held-out ranking and reward separation.

2. `RBM-EVAL-OOD` Pre-Training Objective Ablations

As shown in Table XV, incorporating trajectory-level preference supervision consistently improves reward-model behavior

[6] https://huggingface.co/teetone/RoboReward-4B

[7] https://huggingface.co/teetone/RoboReward-8B

	VOC $r \uparrow$			Kendall $\tau \uparrow$			Succ–Fail Diff. ↑		
Dataset	**Prog. Only**	**+Preference**	**Robometer**	**Prog. Only**	**+Preference**	**Robometer**	**Prog. Only**	**+Preference**	**Robometer**
USC Franka	0.913	**0.974**	0.959	0.083	0.542	**0.646**	0.039	**0.428**	0.326
USC Koch Arm	0.933	0.932	**0.950**	0.231	0.357	**0.471**	0.081	0.142	**0.191**
USC Trossen	0.199	0.902	**0.911**	0.333	0.423	**0.653**	0.052	0.231	**0.312**
USC xArm	0.890	**0.973**	0.961	0.389	0.597	**0.694**	0.079	0.154	**0.345**
MIT Franka	0.936	0.942	**0.954**	0.183	0.458	**0.601**	0.063	0.223	**0.310**
UTD;' SO101	**0.964**	0.899	0.952	0.533	0.667	**0.867**	0.134	0.244	**0.438**
Average	0.806	0.939	**0.948**	0.292	0.507	**0.655**	0.075	0.237	**0.320**

TABLE XV: **Per-dataset model ablation results.** Reward alignment, trajectory ranking, and final reward difference between successful and failed trajectories on `RBM-EVAL-OOD`

		Baselines			w/ RoboReward Training Data			w/ our `RBM-1M` data	
Split	Dataset	GVL	VLAC-8B	RoboDopamine	RoboReward-4B	RoboReward-8B	Robometer	ReWiND	Robometer
`RBM-EVAL-ID`	RACER (Val)	0.131	0.156	0.197	0.491	0.652	0.937	0.561	**0.943**
	OXE (BC-Z Eval)	0.142	-0.150	-0.109	0.643	0.809	0.683	0.442	**0.922**
	OXE (Berkeley Cable Routing Eval)	0.075	-0.425	-0.384	0.657	0.801	**0.900**	0.491	0.887
	OXE (Bridge V2 Eval)	0.143	-0.875	-0.834	0.891	0.898	0.633	0.526	**0.920**
	OXE (Jaco Play Eval)	0.114	-0.151	-0.110	0.793	0.785	0.861	0.550	**0.872**
	OXE (Toto Eval)	0.254	-0.416	-0.375	0.886	0.939	**0.947**	0.340	0.930
	OXE (Viola Eval)	0.264	0.479	0.520	0.915	0.896	**0.967**	-0.014	0.947
	Metaworld (Eval)	0.134	0.211	0.252	0.746	0.779	0.737	0.630	**0.900**
	Libero (90)	0.254	0.217	0.258	0.874	0.846	0.912	0.592	**0.967**
	Average	0.168	0.089	0.130	0.766	0.823	0.842	0.458	**0.921**
`RBM-EVAL-OOD`	USC Franka	0.102	0.356	0.061	0.909	0.909	**0.959**	0.772	**0.959**
	USC Koch	0.176	0.074	-0.221	0.866	0.916	**0.969**	0.585	0.950
	USC Trossen	0.542	0.256	-0.039	0.781	0.726	**0.925**	0.226	0.911
	USC xArm	0.282	0.454	0.159	0.896	0.914	0.951	0.435	**0.961**
	MIT Franka	0.268	0.601	0.306	0.896	0.899	0.868	0.531	**0.954**
	UTD;' SO101	0.203	0.511	0.216	0.926	0.930	0.888	0.497	**0.952**
	Average	0.262	0.375	0.080	0.879	0.882	0.927	0.508	**0.948**

TABLE XVI: **Per-dataset reward alignment results.** Pearson correlation results across individual datasets for `RBM-EVAL-ID` and `RBM-EVAL-OOD`.

	Baselines				w/ RoboReward Training Data			w/ our `RBM-1M` data	
Dataset	GVL	VLAC-2B	VLAC-8B	RoboDopamine	RoboReward-4B	RoboReward-8B	Robometer	ReWiND	Robometer
USC Franka	0.250	0.292	0.271	0.167	0.625	0.625	0.583	-0.125	**0.646**
USC Koch	-0.008	0.167	0.064	0.175	0.332	0.264	**0.533**	0.336	0.471
USC Trossen	0.292	-0.111	-0.417	0.000	0.333	0.389	0.646	0.028	**0.653**
USC xArm	0.056	0.167	0.139	0.014	0.528	0.347	0.403	-0.167	**0.694**
MIT Franka	0.306	-0.017	0.072	0.220	0.494	0.396	0.479	0.080	**0.601**
UTD;' SO101	0.300	-0.033	0.167	0.067	0.700	0.767	0.667	-0.067	**0.867**
Average	0.199	0.077	0.049	0.107	0.502	0.465	0.552	0.014	**0.655**

TABLE XVII: **Per-dataset trajectory ranking results.** Trajectory ranking results on individual `RBM-EVAL-OOD` datasets.

compared to training on progress labels alone. On average, adding preferences boosts reward alignment from 0.806 to 0.939 and improves trajectory ranking from 0.292 to 0.507, indicating that pairwise comparisons provide a strong signal for ordering failed, suboptimal, and successful behaviors. Robometer achieves the best average Kendall score of 0.655 and the largest success–failure final reward difference (0.320), while maintaining high alignment (0.948). Notably, Robometer yields consistent gains in across all OOD datasets, suggesting that jointly leveraging dense progress targets with preference learning produces a reward function that both tracks task completion and better discriminates overall trajectory quality in OOD settings.

We report detailed evaluation results for each dataset in `RBM-EVAL-OOD` in Table XVI (reward alignment) and Table XVII (trajectory ranking).

3. Training loss weight ablations

Table XIX sweeps the relative weighting between the preference and progress losses. We find that the best performance is achieved when the two objectives are weighted uniformly, with $(\lambda_{\text{pref}}, \lambda_{\text{prog}}) = (1, 1)$ outperforming settings that upweight either preference or progress on LIBERO-90 across alignment, ranking, and success–failure separation.

Model	VOC $r \uparrow$		Kendall $\tau \uparrow$		Succ–Fail Diff. $\uparrow$	
	RBM-EVAL-OOD	LIBERO (90)	RBM-EVAL-OOD	LIBERO (90)	RBM-EVAL-OOD	LIBERO (90)
No Different Task	0.930	0.966	0.560	0.903	0.260	0.362
No Rewind	0.860	0.818	0.480	0.815	0.150	0.241
No Subopt	0.915	0.910	0.585	0.890	0.235	0.413
Ours	**0.950**	**0.976**	**0.660**	**0.919**	**0.330**	**0.455**

TABLE XVIII: **Data sampling strategy ablation results.** Reward alignment, trajectory ranking, and final reward difference between successful and failure trajectories on RBM-EVAL-OOD and LIBERO-90. Models are trained without using RBM-EVAL-OOD scenes; RBM-EVAL-OOD and LIBERO-90 are held out for evaluation.

Weights $(\lambda_{\text{pref}}, \lambda_{\text{prog}})$	VOC $r \uparrow$		Kendall $\tau \uparrow$		Succ–Fail Diff. $\uparrow$	
	LIBERO (10)	LIBERO (90)	LIBERO (10)	LIBERO (90)	LIBERO (10)	LIBERO (90)
(2, 1)	0.983	0.964	0.981	0.898	0.347	0.352
(1, 2)	0.986	0.946	0.982	0.875	0.423	0.367
$(1, 1)$	**0.994**	**0.976**	**0.986**	**0.919**	**0.483**	**0.455**

TABLE XIX: **Loss weight ablation results.** Hyperparameter sweep over preference/progress loss weights $(\lambda_{\text{pref}}, \lambda_{\text{prog}})$ on LIBERO-10 and LIBERO-90. Each model is trained on LIBERO-(10, Spatial, Object, Goal) and LIBERO-90 is heldout for evaluation only.

Appendix E
Policy Learning Experiment Details

1. *RL with Ablated Reward Models*

Experiment Setup. For each ablated reward model described in Section IV-Q2, we train an RL policy from scratch using SAC [134]. We evaluate performance on two tasks from the LIBERO-90 suite: Task 28 (*close the top drawer*) and Task 33 (*close the microwave*). Both reward estimation and RL training use only the external camera view. The policy observations consist of a DINO-v2-small [132]-featurized external image concatenated with proprioceptive state inputs. During training, we perform 25 evaluation episodes every 5000 training steps and report the average success rate ± standard deviation. For each reward model, we train 5 random seeds. The SAC hyperparameters are provided in Table XX.

Hyperparameter	Value
Batch size	128
Target update rate τ	0.005
Discount factor γ	0.99
Target update interval	1
Number of critics	5
Critics sampled per update	2
Pooled critic features	True
Actor updates per train step	1
Critic updates per actor update	1
Actor learning rate	1×10^{-5}
Critic learning rate	1×10^{-5}
Entropy coeff learning rate	3×10^{-4}
Target entropy	0
Learning starts	5000 steps

TABLE XX: **RL with Ablated Reward Models.** SAC hyperparameters.

2. *Automatic Online RL*

a) Environment Setup: We setup a DROID [76]-style Franka Panda environment on a cluttered tabletop as depicted in Figure 6. Following DROID, we have a Robotiq gripper, an exterior Zed camera mounted to the left of the robot arm, and a Zed wrist camera.

- **Single Stage**: The task is to put the bowl onto the table, where the difficulty lies in the bowl starting in a tall dish rack which physically blocks the robot gripper if it approaches the bowl from too low of an angle. The clutter also makes it difficult for π_0 to perform this task with high success rates zero-shot (20% initial success rate), making it well-suited for dense reward RL. The task instruction given to π_0 is "put the bowl on the table."
- **Multi-Stage**: The first task is to put the corn in the pot, and the second is to then put the lid on the corn, emulating a "steam corn" task. The pot being in the dish rack confuses π_0, so the base policy tends to either miss putting the corn in the pot or collide with the dish rack and get stuck. The exact instructions for π_0 are "put the corn in the pot located in the dish rack" and "put the lid on the pot located in the dish rack."

b) RL Algorithm: We train an RL policy from scratch with Diffusion Steering (DSRL) [74] to steer a frozen π_0 policy pre-trained on the DROID [76] dataset. The DSRL policy trains an SAC [134]-style algorithm that operates over the *noise space* of the π_0 flow matching head. Specifically, the inputs to the RL policy are:

- DINO-v2-small [132]-featurized wrist image (384-dim)
- π_0 VLM hidden embedding (2048-dim)
- Proprioceptive joint and gripper positions (8-dim)
- For multi-stage only: the index of the current stage the policy is in (1-dim), concatenated with the proprioception.

The output is an action chunk of length 8, where each action is of dimension 32 (π_0's flow matching head noise dimension) and the action bounds are $[-2.0, 2.0]$ for single-stage and $[-1.5, 1.5]$ for multi-stage. The original DSRL paper proposed an MLP policy that outputs a single output noise action, which is copied 10 times (π_0-DROID has a default action length of 10). Unlike the original DSRL paper, we parameterize the policy with a transformer backbone so that it can output a unique noise embedding for each action in the 8-length action chunk—we found this action chunked version to perform more meaningful exploration and thus learn quicker. We copy the first 2 noise actions of the action chunk to fill the remaining 2 noise timesteps for π_0-DROID, but we only execute 8 actions out of 10.

Similarly, the Q function takes as input an 8-length action chunk but otherwise follows a standard Q-function formulation. The network architectures follow the same one used in the Offline RL experiments, whose details are in Table XXVI. SAC algorithm hyperparameters are detailed below.

Hyperparameter	**Value**
Batch size	64
Target update rate τ	0.005
Discount factor γ	0.995
Target update interval	1
Number of critics	4
Critics sampled per update	2
Pooled critic features	True
Actor updates per train step	10 (single), 12 (multi)
Critic updates per actor update	4 (single), 2 (multi)
Actor learning rate	5×10^{-5}
Critic learning rate	1×10^{-4}
Entropy coeff learning rate	1×10^{-4}
Target entropy	0
Learning starts	1200 steps
Training steps	10000 steps

TABLE XXI: **Automatic Online RL.** DSRL SAC hyperparameters.

We run RL for 10000 environment steps, corresponding to about 40 minutes of total real-world experiment time. We train after each episode rather than after each step, and we design an *asynchronous* reward relabeling pipeline that relabels rewards in the replay buffer using reward models after they've been added, to prevent reward-related latency.

Success Detection. Success detection and episode termination come from the reward models. With ROBOMETER, we use the model's success prediction probability and threshold it so that if the last timestep's success detection probability is > 0.6, the episode is marked as a success and terminated. With RoboReward, we use its discrete reward predictions where we mark the episode as successful if it predicts 5/5 reward. If no success is detected by 240 timesteps, the episode terminates and resets.

For the multi-stage experiment, we run for 200 timesteps per stage. If the first stage (corn in pot) fails after 200 timesteps, we reset. If the reward model detects success (as in single-stage) in the first stage, we advance to the second stage, where the episode timeout is reset back to 200 timesteps. If there's a failure in the second stage, we still reset the entire scene back to before the first stage was completed. Each stage advance simply updates the language instruction for π_0 and the task index for the DSRL policy/Q function.

Environment Reward. For the single-stage setup, the base reward for the task is $-1/0$, where -1 is given at every step except success, where 0 is given. This base reward is added to the reward predictions $\in [0, 1]$, thus bounding the per-step reward to be $[-1, 0]$.

In the multi-stage setup, the base reward when the policy is in the first stage is -2 at each timestep, added to the reward predictions from the reward model. When the policy enters the second stage, the base reward is $-1/0$ just like in the single-stage setup. This simple reward design always encourages the policy to advance to the next stage, even under possibly suboptimal rewards.

c) Results: Our single-stage results in Figure 6 show a 55% improvement in performance of ROBOMETER over RoboReward after 10k online RL steps. A key failure mode of RoboReward is its tendency to predict high rewards even when the robot is executing the wrong task. In cluttered environments, RoboReward frequently assigns a maximum reward (5/5) when the robot manipulates objects that are not the target bowl. This leads to a large number of false positives, as quantified in Table XXII.

Method	True Positives	False Positives
RoboReward	6	45
ROBOMETER	18	0

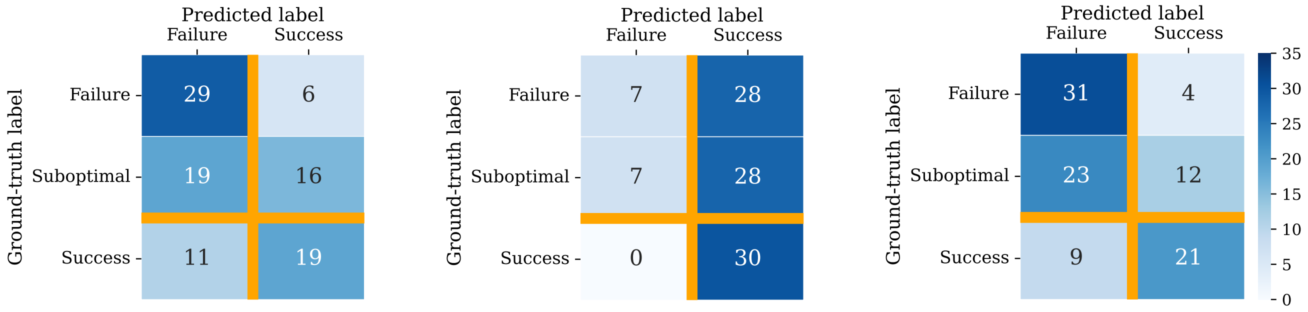
TABLE XXII: **Automatic Online RL reward relabeling.** True positives (TP) and false positives (FP) for reward predictions during online RL.

Finally, in our multi-stage results in Figure 6, RoboReward does not improve the π_0 policy at all. This failure to improve stems from how the π_0 policy tends to fail the first subtask, putting the corn in the pot, by dropping it *near* but not in the pot. RoboReward often falsely assigns a 5/5 score to these failed states, allowing the policy to move onto the next stage. As a result, during evaluation, the RoboReward-trained DSRL policy often drops the corn into various parts of the dish rack rather than into the pot. ROBOMETER, on the other hand, rarely assigns these states as successful, enabling the policy to learn more.

3. *Out of Distribution Failure Detection*

For out-of-distribution failure detection, we evaluate our model on MIT Franka dataset. For quantitative analysis in Table XXIII and Figure 12, we mark a trajectory as success if success probability is higher than or equal to 0.5, and failure otherwise.

a) Baselines: In addition to RoboReward-4B, VLAC, and GPT-5-mini, we compare against an uncertainty-based baseline, following [8, 135]. *Token-Uncertainty* baseline



(a) RoboReward-4B (b) GPT-5-mini (c) ROBOMETER(Ours)

Fig. 12: **Failure Detection OOD confusion matrices** with ternary ground truth and binary prediction. Rows indicate ground-truth execution outcomes (*failure*, *suboptimal*, *success*), while columns indicate binary predictions (*predicted failure* vs. *predicted success*). *Suboptimal* trajectories correspond to executions that make partial progress but do not complete the task. Suboptimal trajectories are treated as failures in our quantitative evaluation, as emphasized with the horizontal orange divider in the above confusion matrices. Color intensity reflects the number of trajectories.

Task	Token-Unc.			VLAC			GPT-5-mini			RoboReward-4B			ROBOMETER		
	TPR	**TNR**	**F1**	**TPR**	**TNR**	**F1**	**TPR**	**TNR**	**F1**	**TPR**	**TNR**	**F1**	**TPR**	**TNR**	**F1**
move banana	0.00	1.00	0.53	0.89	0.35	0.45	0.31	1.00	0.48	1.00	0.62	0.91	1.00	0.75	**0.94**
move mouse	0.00	1.00	0.50	1.00	0.00	0.00	0.80	1.00	0.89	0.80	0.75	0.80	1.00	0.75	**0.91**
pour pebble	0.00	1.00	0.32	0.88	0.00	0.00	0.14	1.00	0.25	0.57	1.00	0.73	0.71	1.00	**0.83**
fold towel	0.00	1.00	0.58	0.95	0.10	0.16	0.15	1.00	0.27	0.31	0.67	0.40	0.54	0.56	**0.58**
pull tissue	0.00	1.00	0.43	1.00	0.00	0.00	0.00	1.00	0.00	0.55	0.00	0.57	0.73	0.50	**0.76**
put spoon	0.00	1.00	0.22	1.00	0.00	0.00	0.14	1.00	0.25	0.57	1.00	**0.73**	0.57	1.00	**0.73**
stir pot	0.00	1.00	0.47	0.94	0.00	0.00	0.09	1.00	0.17	0.91	1.00	**0.95**	0.82	1.00	0.90
Average	0.00	1.00	0.48	0.95	0.10	0.16	0.20	1.00	0.33	0.69	0.63	0.74	0.77	0.70	**0.81**

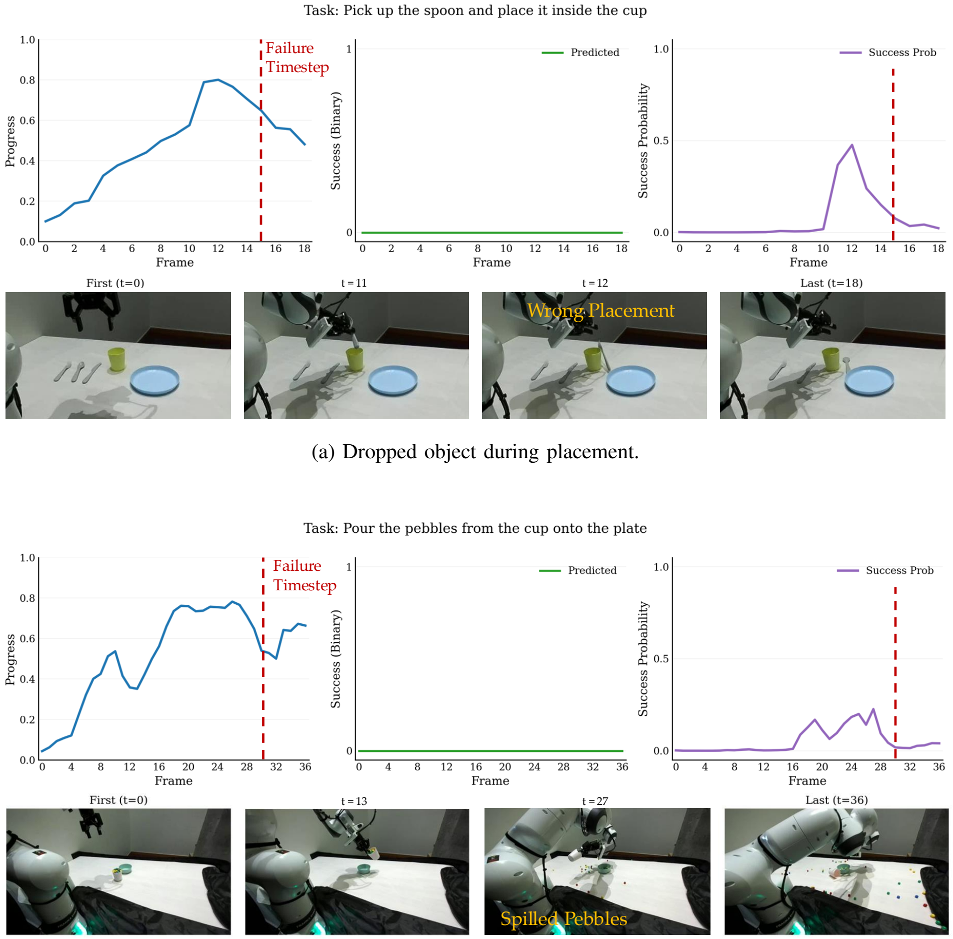
TABLE XXIII: **Failure detection performance**. We report true positive rate (TPR; correctly detecting failures), true negative rate (TNR; correctly identifying successful executions), and F1 score.

estimates predictive uncertainty by computing entropy over generated language tokens. We compute token uncertainty on evaluation trajectories using the π_0-FAST DROID policy [84].

b) Discussion: Figure 12 visualizes out-of-distribution failure detection using an asymmetric 3×2 confusion matrix: ground-truth labels include *failure*, *suboptimal*, and *success*, while predictions are binary (*failure* vs. *success*). This view separates two practically distinct error types: (i) missed failures/suboptimal executions (mass in the right column for the top two rows), and (ii) false alarms on successful executions (mass in the left column for the bottom row). Across methods, ROBOMETER allocates more mass to correctly flagging both *failure* and *suboptimal* trajectories as failures, while maintaining relatively few false alarms on *success* trajectories, consistent with its higher average F1 in Table V. In contrast, GPT-5-mini concentrates mass in the *predicted success* column for *failure* and *suboptimal* rows, indicating a conservative bias that avoids false positives but misses many non-success outcomes, whereas RoboReward-4B is intermediate with improved failure sensitivity but still more confusion between *suboptimal* and *success* than ROBOMETER.

c) Failure Detection with Progress Monitoring: Table XXIII compares our model against baselines on three evaluation metrics: true positive rate (TPR; correctly detecting failures), true negative rate (TNR; correctly identifying successful executions), and F1 score. ROBOMETER consistently outperforms baselines, resulting in higher average F1 score. Token-Uncertainty and GPT-5-mini baselines attain perfect TNR but extremely low TPR, indicating a strong bias toward predicting success and missing failures. VLAC exhibits the opposite behavior: it frequently flags trajectories as failures, achieving high true positive rates but low true negative rates due to many false positives on successful executions, which results in lower F1 scores. RoboReward-4B lies between these extremes, with improved failure sensitivity over GPT-5-mini but more confusion between *suboptimal* and *success* trajectories than ROBOMETER.

We provide qualitative examples of failure detection using progress monitoring across different failure categories in Figures 13–15. Specifically, we compute the Pearson correlation between progress values and time over a sliding window, and flag a failure at the first timestep where this correlation becomes lower than a threshold. This captures both irreversible failures, where rewards sharply decrease after an error (e.g., object drops), and insufficient-progress failures, where rewards stagnate or regress over time. If no such failure is detected or if our model predicts success, the trajectory is classified as successful. Our failure evaluation dataset include both *irreversible failures* (e.g., object drops or spills) and *insufficient-progress failures*, where the robot



(a) Dropped object during placement.

(b) Spilled contents during pouring.

Fig. 13: **Irreversible failures.** Terminal events such as drops or spills cause a sharp regression in predicted task progress, which our model reliably flags as failures shortly after the event.

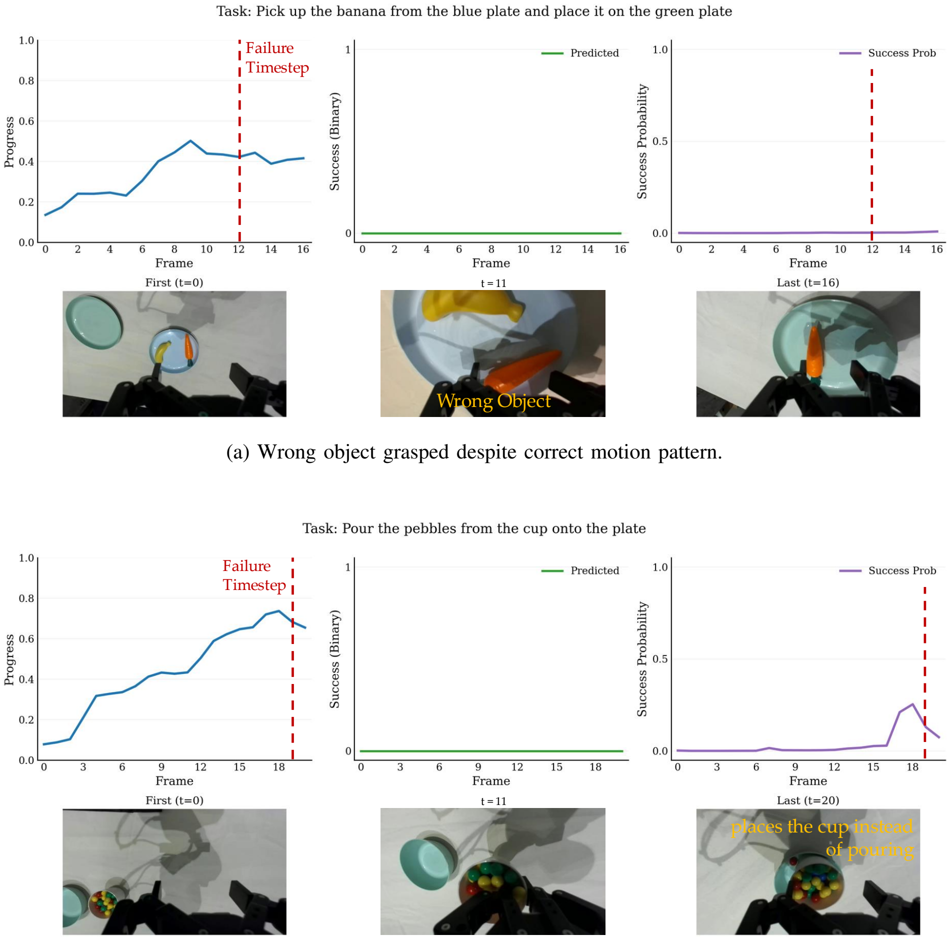
stalls, oscillates, or terminates execution before completing the task. We additionally highlight *semantic failures*, where the robot executes a physically plausible behavior that violates the task instruction. All qualitative examples use window size of 5 besides Figure 13 (c) which uses window size of 9. We use correlation threshold of -0.5 in all examples.

Across all cases, our reward model identifies failures by detecting regressions or stagnation in predicted task progress. For irreversible failures (see Figure 13), the predicted progress increases initially but drops sharply after the terminal event (e.g., dropping or spilling), leading to early failure detection. In semantic failures (see Figure 14), progress remains consistently low despite smooth execution, reflecting instruction-level mismatch. For insufficient-progress failures (see Figure 15), the progress signal plateaus or oscillates without converging to success, allowing the model to flag failure even in the absence of abrupt terminal events.

4. Data Filtering & Retrieval

a) Procedure and Discussion: We have 50 trajectories in our play dataset, where each trajectory consists of five tasks executed in random order: *uncap the red pen*, *open the bottle*, *open the red drawer*, *stir the pot*, and *unzip the pencil case*, collected using the Trossen Stationary AI bimanual setup. The dataset includes three camera views: one top-down view and one wrist-mounted camera for each end effector. Example images from all camera views are shown in Figure 16. Following prior work [81, 136], we first segment each trajectory into subtrajectories based on end-effector velocities and changes in gripper states. In practice, we found that explicitly incorporating gripper state changes produces cleaner segmentation of different tasks. After segmentation, the play dataset yields a total of 615 subtrajectories. If subtrajectories were uniformly distributed across tasks, this would correspond to 123 relevant segments per task; to ensure conservative evaluation, we retrieve 100 subtrajectories per task.

We compare our method against three baselines: RoboReward, pre-trained SigLIP [80], a vision–language model trained with contrastive image–language supervision, and a retrieval-specific baseline, STRAP [81]. We chose to include STRAP as it is one of the few viable approaches for zero-shot



(a) Wrong object grasped despite correct motion pattern.

(b) Instruction-inconsistent action (placing the cup instead of pouring).

Fig. 14: **Semantic failures.** The robot executes smooth and physically plausible trajectories but violates the task instruction, resulting in persistently low predicted progress and failure detection without an abrupt terminal event.

Fig. 15: **Insufficient-progress failures.** Although the robot continues to move, predicted progress stagnates or oscillates without converging to success, enabling detection of failures caused by stalling or premature termination. Execution stalls with plateaued progress in this example.

data filtering and retrieval; alternative methods either require policy rollouts [137], larger amounts of task-specific data [138], or training additional task-specific retrieval modules [139, 140, 141, 142].

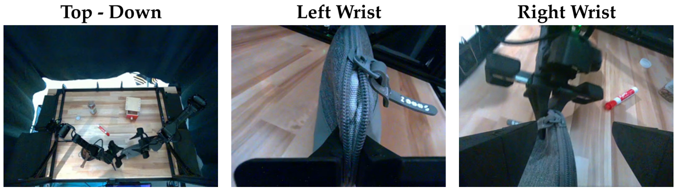


Fig. 16: **Data filtering & retrieval scene configuration** from all cameras.

Given a task language instruction, we perform retrieval using different reward models as follows. For RoboReward and ROBOMETER-Prog, we first compute per-timestep reward predictions for each subtrajectory conditioned on the instruction. We then calculate the value-order correlation of each subtrajectory using the predicted rewards and select the top 100 subtrajectories with the highest VOC scores. When using the preference-based variant of our method (ROBOMETER-Pref), we construct pairwise comparisons between all subtrajectories, aggregate the results into a win matrix, and rank subtrajectories according to estimated pairwise preferences. The top 100-ranked subtrajectories are then selected. For SigLIP, we compute the average vision–language similarity between each subtrajectory and the instruction and select the top 100 subtrajectories with the highest average similarity scores. We use wrist-camera images for retrieval in all methods, as they yield the highest-quality retrievals for all methods.

For $\pi_{0.5}$ policy fine-tuning via LoRA [82, 143], we directly use the retrieved 100 subtrajectories together with the corresponding task instruction. During fine-tuning, we include observations from all camera views, as well as states and actions. Results reported in Figure 8(b) use a strict success

Task	Method	#Succ	#Subopt	#Fail	#Unrel
Open the red drawer	SigLIP	2	2	1	0
	RoboReward	2	1	2	0
	STRAP	2	2	1	0
	Robometer–Prog	**3**	**2**	0	0
	Robometer–Pref	**3**	**2**	0	0
Unzip the pencil case	SigLIP	2	1	2	0
	RoboReward	0	0	2	3
	STRAP	2	2	1	0
	Robometer–Prog	**3**	**2**	0	0
	Robometer–Pref	2	2	1	0

TABLE XXIV: **Trajectory retrieval analysis.** Top-5 retrieval quality summary per task. Counts indicate how many of the top-5 retrieved trajectories are labeled as success (Succ), suboptimal (Subopt), failure (Fail), or unrelated (Unrel).

metric: a trial is counted as successful only if the robot fully completes the task. This criterion partially explains the low success rates observed for the baselines. Qualitative inspection of learned behaviors reveals clear differences between retrieval methods. Policies trained on SigLIP-retrieved data often learn to approach task-relevant objects (e.g., reaching the drawer) but fail to complete the task. In contrast, policies trained on RoboReward-retrieved data frequently exhibit random or unstable behaviors, consistent with a higher proportion of unrelated subtrajectories in the retrieved set.

b) Additional Experiments: We additionally evaluate retrieval quality in a controlled setting where ground-truth trajectory labels are available. Instead of using the play dataset, we consider the Trossen subset of `RBM-EVAL-OOD`, which contains six tasks with known numbers of successful, suboptimal, and failed trajectories. We perform retrieval for two representative tasks—*open the red drawer* and *unzip the pencil case*—each of which contains three successful, two suboptimal, and two failure trajectories. Using the same retrieval procedures described above, we retrieve five trajectories per task and assess their quality. Results are shown in Table XXIV. Robometer consistently retrieves higher-quality trajectories, while baselines either fail to retrieve task-relevant trajectories or select failure cases. These results highlight Robometer's ability to both distinguish between tasks and differentiate levels of execution quality.

5. Combining Noisy and Expert Trajectories via Offline RL

a) Procedure and Data Collection: We study how different reward formulations affect downstream policy learning when training on a mixture of expert and noisy trajectories using offline reinforcement learning. Specifically, we compare three reward settings: (i) a sparse terminal reward, (ii) rewards predicted by RoboReward, and (iii) rewards predicted by Robometer.

We evaluate offline RL on the SO-101 robot platform under two settings across two manipulation tasks: *Put the bread in the oven* and *Put the red bowl on the blue plate*. **Setting 1** is a clean single-task setting that evaluates *Put the bread in the oven*; **setting 2** is a cluttered multi-task setting that evaluates *Put the red bowl on the blue plate*. We visualize these settings in Figure 7.

For each task, we collect a mixed-expertise offline dataset containing both expert demonstrations and failed executions. In **Setting 1**, we collect 30 successful trajectories and 45 failed trajectories for *Put the bread in the oven*. In **Setting 2**, we collect 20 successful trajectories and 15 failed trajectories each for 3 separate tasks: *Put the marker in the pen cup*, and *Put the red cup on the purple coaster*, and the evaluation task *Put the red bowl on the blue plate*. With offline RL, transitions from other tasks, whether the data is suboptimal or expert, should be useful for the *Put the red bowl on the blue plate* task as they share a similar cluttered scene.

Observations include images from both an external camera and a wrist-mounted camera, as well as proprioceptive states. During offline RL training, we use images from both camera views, while reward estimation is performed using the external camera view only.

b) Offline RL Setup and IQL Objectives: We follow an offline RL training setup similar to ReWiND [5]. For all methods, we train policies using Implicit Q-Learning (IQL) [144] with identical actor, critic, and value network architectures, optimization hyperparameters, and datasets. IQL learns a value function $V_\psi(s)$, an ensemble of action-value functions $\{Q_{\theta_i}(s,a)\}_{i=1}^N$, and a policy $\pi_\phi(a \mid s)$ from an offline dataset $\mathcal{D} = \{(s,a,r,s')\}$. Each critic is trained by minimizing the temporal-difference loss $\mathcal{L}_Q(\theta_i) = \mathbb{E}_{\mathcal{D}}\big[(Q_{\theta_i}(s,a) - (r + \gamma V_\psi(s')))^2\big]$. The value network is trained via expectile regression toward the critic ensemble using $\mathcal{L}_V(\psi) = \mathbb{E}_{\mathcal{D}}[\rho_\tau(\min_i Q_{\theta_i}(s,a) - V_\psi(s))]$, where $\rho_\tau(u) = |\tau - \mathbb{I}(u < 0)|u^2$. The policy is learned via advantage-weighted regression with objective $\mathcal{L}_\pi(\phi) = \mathbb{E}_{\mathcal{D}}[\exp(A(s,a)/\beta)\log\pi_\phi(a \mid s)]$, where the advantage is defined as $A(s,a) = \min_i Q_{\theta_i}(s,a) - V_\psi(s)$ and β is the advantage temperature.

To account for differences in reward sparsity and temporal structure, we sweep the discount factor $\gamma \in \{0.9, 0.95, 0.99\}$ for each reward setting. All other IQL hyperparameters and actor–critic architecture details are held fixed across methods and summarized in Tables XXV and XXVI. For each method, we select the best-performing checkpoint based on validation performance.

c) Reward Specification: RoboReward provides a sparse, trajectory-level progress signal. To enable its use in IQL, we convert RoboReward outputs into a per-frame dense reward by querying the model on partial trajectory prefixes $o_{1:t}$ and assigning the resulting prediction as the reward at timestep t. In contrast, our reward model directly produces dense, temporally aligned rewards over video sequences, which we use without additional post-processing. Aside from this difference in reward construction, all other aspects of offline RL training are kept identical across methods.

Appendix F
Low-rank Fine-tuning Robometer-4B

RoboFAC dataset. We fine-tune and evaluate on RoboFAC, a video-based dataset designed for robotic failure analysis and correction. RoboFAC spans 16 tasks across 53 scenes

Hyperparameter	Value
Batch size	256
Target update rate τ	0.005
Discount factor γ	$\{0.9, 0.95, 0.99\}$
Target update interval	1
Advantage temperature	2
Expectile	0.7
Advantage clipping	100.0
Policy extraction method	AWR [145]
Number of critics	5
Critics sampled per update	2
Pooled critic features	True
Updates per train step	1
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Value network learning rate	3×10^{-4}
Weight decay	0.0

TABLE XXV: **SO-101 offline RL experiment.** Hyperparameters used for Implicit Q-Learning (IQL) in offline policy training.

Component	Actor	Critic
Transformer Encoder		
Model dimension (d_{model})	256	256
Number of heads	8	8
Encoder layers	6	6
Transformer dropout	0.0	0.0
Transformer activation	GELU	GELU
Layer normalization	False	True
Pooling strategy	–	First token
Feature Processing MLP		
Hidden dimensions	[512, 512]	[768, 512]
Activation	ReLU	ReLU
Dropout	0.0	0.0
Output Head		
Output hidden dims	None	None
Action squashing	Tanh	–
Deterministic policy	False	–
Log-std init	0	–
Log-std range	$[-20, 2]$	–

TABLE XXVI: **SO-101 offline RL model architectures.** Transformer-based actor and critic architectures used for offline IQL training. Both networks share a lightweight Transformer encoder, followed by task-specific MLP heads.

and provides 78K video QA pairs annotated over 10,722 trajectories. The dataset contains both failures and successes: 9,440 failure trajectories and 1,282 successful trajectories. Failures are collected in both simulation and the real world, with 8,960 simulated failures and 480 real-world failures; additionally, the dataset includes 1,160 simulated successes and 122 real-world successes.

LoRA fine-tuning Robometer-4B. We keep our pre-trained Robometer-4B reward model backbone fixed and train LLM LoRA adapters (76M parameters) while also fine-tuning the MLP prediction heads (progress, preference, and success). We train for 500 steps with a batch size of 8 on a single NVIDIA RTX A6000 for approximately 8 hours of wall clock time.

Baselines. To isolate the benefit of initializing from our pre-trained ROBOMETER-4B checkpoint versus starting from scratch, we include two Qwen3-VL baselines fine-tuned on the same dataset. First, we perform LoRA fine-tuning starting from `Qwen/Qwen3-VL-4B-Instruct` and train the prediction heads from random initialization. We also fully fine-tune all parameters of `Qwen/Qwen3-VL-4B-Instruct`), again training the prediction heads from random initialization.

Results. Table III reports offline reward evaluation on RoboFAC. In the zero-shot setting, ROBOMETER-4B already achieves strong correlation with ground-truth progress (VOC r=0.652, Kendall τ=0.436), indicating meaningful transfer without any RoboFAC supervision. After adaptation, ROBOMETER-4B provides a substantially better initialization than training the base VLM from scratch. Full fine-tuning from our checkpoint reaches VOC r=0.884 and Kendall τ=0.802, outperforming the strongest from-scratch baseline by $+21.6\%$ in VOC and nearly an order of magnitude in Kendall τ. Notably, LoRA on ROBOMETER-4B attains essentially the same performance, indicating that the gains primarily come from our pre-trained initialization rather than full-parameter adaptation.

APPENDIX G
ROBOMETER WITH MODEL-BASED RL

Setup. For this proof-of-concept experiment, we integrate ROBOMETER into a model-based algorithm by using it to rank candidate trajectories sampled from the world model.

We adopt the DreamZero [77] world model checkpoint trained on DROID to use in an identical DROID setup to the online RL experiments, albeit with a different task.

Given 3 camera observations, proprioceptive states, and a language instruction, DreamZero generates candidate future observation sequences (multiple video frames spanning 1.6 seconds) and associated action chunks of length 24.

We compare DreamZero with and without ROBOMETER ranking candidate trajectories. DreamZero without ROBOMETER directly generates a single candidate future observation sequence from which we extract the action chunk. With ROBOMETER, we generate 6 candidate observation sequences at each inference step, rank them with the progress output from ROBOMETER, and then execute the action sequence extracted from the highest-ranked generated observation sequence.

In total, generating a single action chunk of length 24, including the world model forward pass for the 6 candidate observation sequences, takes approximately 28 seconds on 1 H200, resulting in a single trajectory taking around 3 minutes to execute including robot execution time. The vast majority of this inference time is from the world model; ROBOMETER's forward pass only takes around 0.6-1 seconds.

Results. We evaluate on a "put the ice cream in the pink plate" task in an extremely cluttered scene with many possible receptacle plates of different colors. DreamZero typically

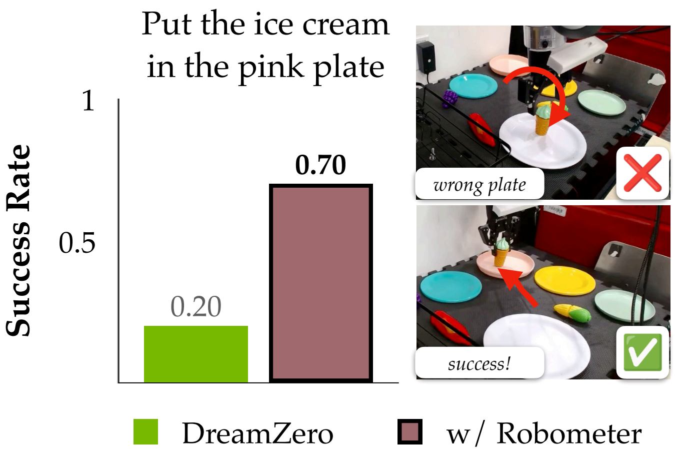


Fig. 17: **Model-Based RL** with ROBOMETER integrated into DreamZero [77]. In this cluttered scene, ROBOMETER improves DreamZero's performance from 20% success rate to 70%.

places the ice cream cone in the wrong plate, while integrating ROBOMETER corrects for this mistake. Overall, results in Figure 17 demonstrate a 3.5× improvement in success rate evaluated over 10 trials with ROBOMETER.