ReWiND: Language-Guided Rewards Teach Robot Policies without New Demonstrations

Jiahui Zhang[⋆1], **Yusen Luo**[⋆1], **Abrar Anwar**[⋆1],
Sumedh A. Sontakke[2], **Joseph J. Lim**[3], **Jesse Thomason**[1], **Erdem Bıyık**[1], **Jesse Zhang**[1]
[1]Thomas Lord Department of Computer Science, University of Southern California
[2]Amazon Robotics, [3]Kim Jaechul School of Artificial Intelligence, KAIST

Abstract: We introduce ReWiND, a framework for learning robot manipulation tasks solely from language instructions *without per-task demonstrations*. Standard reinforcement learning (RL) and imitation learning methods require expert supervision through human-designed reward functions or demonstrations for every new task. In contrast, ReWiND starts from a small demonstration dataset to learn: (1) a data-efficient, language-conditioned reward function that labels the dataset with rewards, and (2) a language-conditioned policy pre-trained with offline RL using these rewards. Given an unseen task variation, ReWiND fine-tunes the pre-trained policy using the learned reward function, requiring minimal online interaction. We show that ReWiND's reward model generalizes effectively to unseen tasks, outperforming baselines by up to $2.4\times$ in reward generalization and policy alignment metrics. Finally, we demonstrate that ReWiND enables sample-efficient adaptation to new tasks in both simulation and on a real bimanual manipulation platform, taking a step towards scalable, real-world robot learning. See website here.

Keywords: Reinforcement Learning, Reward Modeling, Language

1 Introduction

A great teacher does not just tell you if you are right or wrong. Instead, they guide you by providing feedback when you make mistakes, highlighting progress as you learn something new, and adapting to how you learn best. For deployed robots to learn new tasks in the wild, they need similarly intelligent teachers. These teachers—in the form of robust reward models—should: (1) offer *dense, informative feedback*, especially during failures; (2) *generalize* their guidance to unseen tasks; and (3) remain *robust* to diverse robot behaviors during its learning process. Our paper leverages these insights to develop reward models capable of teaching robots unseen tasks.

In this work, we introduce ReWiND (**Re**wards **Wi**thout **N**ew **D**emonstrations), a framework designed to teach robots unseen tasks in a sample-efficient manner using only a few grounding human demonstrations for training tasks (see Figure 1). Typically, teaching robots involves large-scale imitation learning [1, 2, 3, 4], where human experts provide demonstrations for each new task. However, collecting task-specific demonstrations is expensive and time-consuming. Reinforcement learning (RL) offers a more autonomous alternative by using reward functions as teachers, allowing robots to learn through interaction. Yet, manually designing these reward functions demands substantial manual effort and domain-specific expertise [5]. Recent progress in language-conditioned reward learning [6, 7, 8, 9, 10, 11, 12, 13, 14, 15] has aimed at addressing these challenges, but often assumes unrealistic conditions such as availability of ground-truth states [7, 8, 9, 10, 11, 12], thousands of demonstrations [13], or online training of reward models from scratch [14, 15], limiting their practical applicability.

⋆Equal Contribution

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

Figure 1: **Overview.** We pre-train a policy and reward model from a small set of language-labeled demos. Then, we solve unseen task variations via language-guided RL *without additional demos*.

ReWiND overcomes these challenges by instead assuming only a handful of demonstrations—e.g., five per task—to enable real-world robot learning of unseen task variations. ReWiND first trains a language-conditioned reward model from these demonstrations, then uses it to pre-train a language-conditioned policy via offline RL. When deployed, ReWiND efficiently fine-tunes the policy on new task variations by reward-labeling *online* interaction episodes.

Our core contribution is in designing ReWiND's reward model to capture three key properties outlined earlier: **dense feedback**, **generalization**, and **robustness**. First, to provide *dense, informative feedback*, we design a *cross-modal sequential aggregator* that predicts *progress* within demonstration videos. Progress prediction offers a densely supervised training signal that naturally translates into rewards. Importantly, we also introduce *video rewind* to automatically generate failure trajectories from successful ones, allowing ReWiND to provide dense feedback even when the policy is making mistakes. Then, to encourage *generalization* to unseen tasks and *robustness* to diverse behaviors, we train the sequential aggregator with pre-trained vision and language encoders, selectively applied positional embeddings, and diverse robotics data from Open-X [16]. Focusing on the above three properties enables ReWiND rewards to extrapolate to novel visual and language inputs.

We introduce reward metrics measuring the above properties on which ReWiND achieves **23-74%** relative improvements over reward learning baselines. Further, comprehensive success rate evaluations on MetaWorld manipulation tasks and a real-world bimanual robot setup demonstrate ReWiND beats baselines by **2x** in simulation and improves real-world pre-trained policies by **5x**.

2 Related Works

Learning Reward Functions. Prior work in reinforcement learning has proposed various methods for *learning* reward functions. Examples include inverse RL [17, 18, 19, 20], where reward functions are learned from demonstrations, or methods where rewards are implicitly learned from expert or goal state distributions [21, 22, 23]. However, these works require new target-task demonstrations to reward unseen tasks. ReWiND instead trains a general, language-conditioned reward function from an initial demonstration set to reward unseen task variations without further demos.

Another line of work learns reward functions directly from human feedback in the form of comparisons [24, 25, 26, 27, 28], reward sketches [29], preference rankings [30], scaled preferences [31], critiques [32], corrections [33], interventions [34], and language [15]. While these feedback types may require less human effort than demonstrations or manually written reward functions, these works still require humans to provide extensive feedback for each unseen task.

Reward Generation with Pre-trained Models. Prior work has also explored using large pre-trained models to generate reward functions instead of learning them from scratch. Some approaches use LLMs to generate language-conditioned rewards [7, 8, 9, 10, 11, 12], but they typically rely on ground-truth state information that is difficult to obtain in real-world settings. In contrast, ReWiND generates rewards from just a task description and a policy execution video.

Other approaches use pre-trained vision models to derive rewards from visual observations [35, 36, 37, 38, 39, 6, 14, 13, 40, 41, 42, 43, 44, 45, 46]. Among these, RoboCLIP [6], LIV [10], VLC [13], VICTOR [46], and GVL [11]—like ReWiND—reward unseen manipulation tasks directly from language without sub-task annotations or target-task demos. We show in Section 4.1 that these baselines underperform ReWiND in rewarding policies in our limited-data setting. Similar to ReWiND,

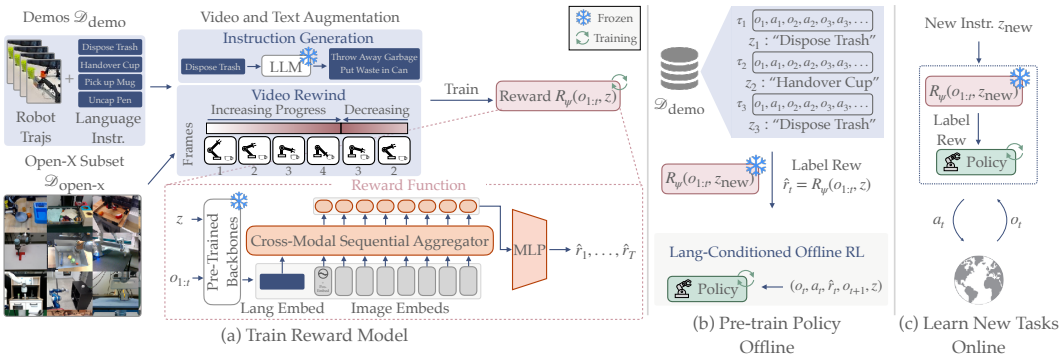


Figure 2: **(a):** We train a reward model $R_\psi(o_{1:t}, z)$ on a small demonstration dataset $\mathcal{D}_{\text{demos}}$ and a curated subset of Open-X, $\mathcal{D}_{\text{open-x}}$, augmented with LLM-generated instructions and video rewinding. $R_\psi(o_{1:t}, z)$ predicts video *progress* rewards $\hat{r}_{1:T}$ from pre-trained embeddings of image observations $o_{1:T}$ and language instructions z, and assigns 0 progress to misaligned video-language pairs. **(b):** We use the trained $R_\psi(o_{1:t}, z)$ to label $\mathcal{D}_{\text{demos}}$ with rewards and pre-train a language-conditioned policy using offline RL. **(c):** For an unseen task specified by z_{new}, we fine-tune the policy with online rollouts and reward labels from $R_\psi(o_{1:t}, z_{\text{new}})$.

Foundation Actor-Critic [47] enables efficient RL from language via potential-based shaping rewards from a pre-trained VLM. However, it depends on predefined policy priors (e.g., code-based primitives from LLMs), whereas ReWiND learns them through offline RL on non-target tasks.

3 ReWiND: Learning Rewards Without New Demonstrations

We study the problem of learning unseen, language-specified tasks in a *target environment*, formulated as a Markov decision process (MDP). The *target environment* refers to the deployment scene (e.g., a robot tabletop). We train a policy $\pi_\theta(a_t \mid o_t, z)$ that selects actions a_t based on images o_t and language instructions z. The policy is optimized to maximize rewards predicted by a learned reward function $R_\psi(o_{1:t}, z)$, which conditions on the frame sequence $o_{1:t}$ and instruction z to output per-timestep estimated rewards $\hat{r}_t$. We assume access to a small demonstration dataset $\mathcal{D}_{\text{demos}}$ in the target environment containing 15–20 tasks with $\sim$5 demonstrations each. Following prior definitions of generalization [48, 49], we define a task as unseen if it requires a novel *action sequence*, its distribution of *image observations* has changed, or needs a new *language instruction*.

ReWiND consists of 3 phases (see Figure 2): (1) learning a reward function from limited target environment demos, then (2) pre-training π with learned rewards on the demos, and finally (3) using the reward function and pre-trained policy to learn a new language-specified task online.

3.1 Learning a Reward Function

Our primary objective for reward prediction is regressing directly to per-frame *progress* within an observation sequence $o_{1:T}$ conditioned on instruction z. Unlike prior methods using relative targets [41, 13], our progress-based objective provides stable, fixed targets, and translates directly into a dense, $[0, 1]$-normalized reward for policy training. To ensure robustness against mismatched observations and instructions, we also sample unrelated observation sequences $o^{\text{other}}_{1:T}$ and train $R_\psi(o_{1:t}, z)$ to predict zero progress. Our reward prediction loss is:

$$\mathcal{L}_{\text{progress}}(o_{1:T}, z, o^{\text{other}}_{1:T}) = \sum_{t=1}^{T}(R_\psi(o_{1:t}, z) - \underbrace{t/T}_{\text{matched seq. progress}})^2 + \sum_{t=1}^{T} \underbrace{R_\psi(o^{\text{other}}_{1:t}, z)^2}_{\text{mismatched seq. 0 progress}} . \quad (1)$$

However, simply training a neural network $R_\psi(o_{1:t}, z)$ on $\mathcal{L}_{\text{progress}}(o_{1:T}, z, o^{\text{other}}_{1:T})$ with a small set of demonstrations is unlikely to ensure that it can train a policy on unseen tasks. $R_\psi(o_{1:t}, z)$ should:

D1 Generalize to new tasks, i.e., new policy execution videos and instructions not in $\mathcal{D}_{\text{demos}}$.
D2 Produce rewards aligned with *policy rollouts*, not just successful demonstration videos.

D3 Be robust to input variations, i.e., different ways to solve or specify the task.

To this end, we introduce a set of design choices spanning the training dataset, model architecture, and video and language augmentations that address all three desiderata. Specifically, we curate diverse off-the-shelf data from the Open-X dataset [16] to promote generalization (**D1**) and robustness (**D3**); apply targeted video and language augmentations for better reward prediction and language input robustness (**D2**, **D3**); and adopt specific network architectural modifications aimed at improving generalization (**D1**). For a visual overview, see Figure 2a.

Incorporating Diverse Data (D1, D3). To help $R_\psi(o_{1:t}, z)$ generalize to tasks unseen in $\mathcal{D}_{\text{demos}}$ (**D1**) and make it robust to diverse ways of executing and specifying tasks (**D3**), we subsample the Open-X Dataset [16], denoted $\mathcal{D}_{\text{open-x}}$. We specifically select Open-X trajectories with object-centric language instructions, e.g., "*pick coke can from fridge*," or directional instructions, e.g., "*drag the circle to the left of the star*," to help $R_\psi(o_{1:t}, z)$ generalize to objects and directions not contained in $\mathcal{D}_{\text{demos}}$. This dataset contains ∼356k trajectories with ∼59k unique task strings. For detailed dataset information, see Appendix A.1.1.

3.1.1 Video and Language Augmentation (D2, D3)

Given our datasets $\mathcal{D}_{\text{demos}}$ and $\mathcal{D}_{\text{open-x}}$, we perform both video and language augmentations that help the reward function accurately predict rewards for unsuccessful *policy execution* videos (**D2**) and be robust to varied ways of specifying the task instructions z (**D3**). We call the video augmentation **video rewind** and our text augmentation **instruction generation**.

Video Rewind. Both $\mathcal{D}_{\text{demos}}$ and $\mathcal{D}_{\text{open-x}}$ contain human demonstrations, which are assumed to be successful and of high-quality. Training $R_\psi(o_{1:t}, z)$ on $\mathcal{L}_{\text{progress}}(o_{1:T}, z, o^{\text{other}}_{1:T})$ only using these successful demonstrations, may result in $R_\psi(o_{1:t}, z)$ overfitting to these successful trajectories. However, during online deployment, $R_\psi(o_{1:t}, z)$ will likely encounter failure trajectories (unseen during training) which such an overfit model may reward highly. This is undesirable and prior works attempt to address this issue by explicitly training their reward model on failed trajectories [13], but these trajectories add a great additional burden on demonstrators to collect and must be added post-hoc to any existing dataset, making it harder to scale.

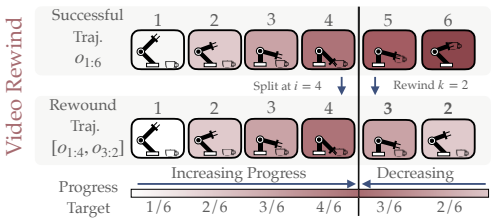


Figure 3: **Video rewind.** We split a demo at intermediate timestep i into forward/reverse sections. Here, the forward section shows the robot approaching the cup; the reverse section $(o_{i-1}, o_{i-2}, \ldots)$ resembles dropping it.

Instead, we address this problem in a scalable manner by randomly *rewinding videos*. Consider a video of a robot picking up a cup. If we rewind the video for a few frames right when the robot grabs the cup, it now looks like one in which the robot attempted to grasp the cup and then dropped it.[1] By training $R_\psi(o_{1:t}, z)$ to predict rewards corresponding to *reverse progress* on the rewound subsequence, it (1) is trained on observation sequences mimicking failed policy rollouts that will occur during online RL, and (2) learns to *decrease* reward when necessary. Thus rewinding helps $R_\psi(o_{1:t}, z)$ reward a *policy's* failures which will help with online RL (**D2**). See Figure 3 for a visual example. Formally, rewinding means sampling a random split point i within an observation sequence $o_1 ... o_T$, rewinding k (k is also sampled) frames, then concatenating those k frames to the end of the original sequence to become $o_1 ... o_i, o_{i-1}, ..., o_{i-k}$. The remaining frames from $i+1$ to T are then unused. Our video rewind training objective follows:

$$\mathcal{L}_{\text{rewind}}(o_{1:T}, z) = \sum_{t=1}^{i} \underbrace{(R_\psi(o_{1:t}, z) - \frac{t}{T})^2}_{\text{Loss for original trajectory until } i} + \sum_{t=1}^{k} \underbrace{(R_\psi([o_{1:i}, o_{i-1:i-t}], z) - \frac{i-t}{T})^2}_{\text{Rewound video for } k \text{ frames from } i-1}. \quad (2)$$

[1] Random rewinding may result in some physically implausible sequences. However, since they won't appear during inference, the rewards produced by $R_\psi(o_{1:t}, z)$ for such sequences should not affect online RL.

Instruction Generation. We also generate 5-10 additional language instructions for each task in $\mathcal{D}_{\text{demos}}$ by prompting an LLM. This augmentation helps $R_\psi(o_{1:t}, z)$ with input robustness to possible new task instructions (**D3**). While training $R_\psi(o_{1:t}, z)$, any time we sample an observation sequence $o_{1:T}$, its instruction z is uniformly randomly sampled from all available matching instructions, generated or original. We did not augment $\mathcal{D}_{\text{open-x}}$ due to its instruction diversity.

3.1.2 Architecture (D1)

Due to the limited size of $\mathcal{D}_{\text{demos}}$, we carefully design the architecture for $R_\psi(o_{1:t}, z)$ to maximize generalization to new tasks (**D1**) while retaining the ability to optimize $\mathcal{L}_{\text{progress}}(o_{1:T}, z, o^{\text{other}}_{1:T})$ well.

Frozen Input Encoders. We use frozen image and language encoders as the backbone of $R_\psi(o_{1:t}, z)$: we use DINOv2 [50] for image encoding due to its strong object-centric representations and `all-MiniLM-L12-v2` [51] for instruction encoding due to its small embedding size ($= 384$). In $R_\psi(o_{1:t}, z)$, we first encode images and instructions: $o^{\text{embed}}_{1:t} = \text{DINO}(o_{1:t})$, $z^{\text{embed}} = \text{MiniLM}(z)$. Then, we train a small *cross-modal sequential aggregator* transformer conditioned on $(o^{\text{embed}}_{1:t}, z^{\text{embed}})$ that learns to aggregate frozen language and image embeddings to generate progress rewards $\hat{r}_t$ directly (see Figure 2(a) in the "Reward Function" box).

Positional Embeddings. Finally, the cross-modal sequential aggregator's transformer requires positional information about the frames to properly predict rewards (e.g., for distinguishing "pull" vs. "push"). However, if we naïvely add positional embeddings to each image, it can "cheat" by predicting progress using the positional embeddings. Therefore, similar to how Ma et al. [43] prompt an LLM with the position of the first video frame, we add a positional embedding to the *first* image.

Reward Model Summary. In summary, ReWiND trains a reward function $R_\psi(o_{1:t}, z)$ to predict task progress, using data augmentation (video rewinding and instruction generation) and additional Open-X data ($\mathcal{D}_{\text{open-x}}$) to improve generalization. $R_\psi(o_{1:t}, z)$ combines pretrained vision and language encoders with a lightweight cross-modal sequential aggregator that uses only first-frame positional embeddings. For full implementation details, see Appendix A.1.2. The final objective is:

$$\min_\psi \mathbb{E}_{(o_{1:T}, z, o^{\text{other}}_{1:T}) \sim \mathcal{D}_{\text{demos}}, \mathcal{D}_{\text{open-x}}} \left[\mathcal{L}_{\text{progress}}(o_{1:T}, z, o^{\text{other}}_{1:T}) + \mathcal{L}_{\text{rewind}}(o_{1:T}, z) \right]. \tag{3}$$

3.2 Policy Learning

Pre-training. After training $R_\psi(o_{1:t}, z)$, we pre-train $\pi_\theta(a_t \mid o_t, z)$ on demonstrations $\mathcal{D}_{\text{demos}}$ labeled with rewards. This pre-training guides $\pi_\theta(a_t \mid o_t, z)$ toward reasonable behaviors during exploration, even if downstream tasks differ from those in $\mathcal{D}_{\text{demos}}$. Given a trajectory with instruction z, $\{(o_t, a_t)\}_1^T$, we assign rewards $\hat{r}_t = R_\psi(o_{1:t}, z)$ at each timestep and add a success bonus to the final reward to encourage reaching the goal despite possibly noisy reward signals:

$$\hat{r}^{\text{off}}_t = R_\psi(o_{1:t}, z) + r_{\text{success}} \cdot \mathbb{1}[t = T]. \tag{4}$$

We then train $\pi_\theta(a_t \mid o_t, z)$ via offline RL using tuples $(o_t, a_t, \hat{r}_t, o_{t+1}, z)$. We use IQL [52] as prior work has demonstrated it works on real robots [53, 54, 55]. See Figure 2(b) for an overview.

Learning Online. To learn a new task online, ReWiND only requires a language description of the task, z_{new}. ReWiND rolls out $\pi(a \mid o_t, z_{\text{new}})$ and fine-tunes it on rewards coming from $R_\psi(o_{1:t}, z_{\text{new}})$. Like prior work [13, 41], we assume access to a success signal during online RL. We use this signal to give r_{success} bonuses similar to pre-training.[2] Our online rewards $\hat{r}^{\text{on}}$ are:

$$\hat{r}^{\text{on}}_t = R_\psi(o_{1:t}, z) + r_{\text{success}} \cdot \mathbb{1}[\text{success at } t]. \tag{5}$$

See full implementation details in Appendix A.1 and pseudocode in Algorithm 1.

[2] Success bonuses can come from a human supervisor [56], learned function [23], or LLM [47]. Our experiments assume a human supervisor because manual resets are required regardless. While we could threshold $R_\psi(o_{1:t}, z)$ outputs to automatically determine success, unseen evaluation task reward ranges can vary, rendering this approach ineffective. Future work could integrate ReWiND with methods reducing human resets [42, 57] and automatic success detectors for truly autonomous RL.

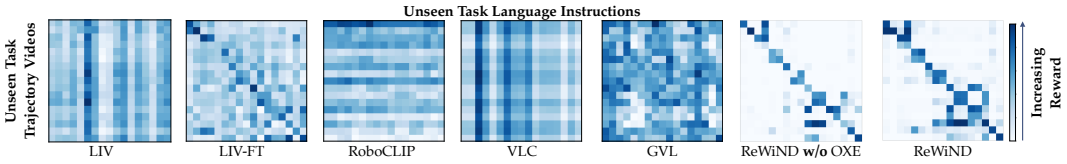


Figure 4: **Video-Language Reward Confusion Matrix.** For each unseen MetaWorld task, we compute rewards for all combinations of demonstration videos and language descriptions. ReWiND produces the most diagonal-heavy confusion matrix, indicating strong alignment between unseen demos and instructions. See Appendix D.1 for train task results, Appendix D.3 for real-world results.

4 Experiments

Our experiments aim to study the efficacy of ReWiND as a reward learning pipeline, evaluate its ability to train robots to learn new tasks efficiently, and analyze its design choices and limitations. To this end, we organize our experiments to answer the following empirical questions, in order:

(**Q1**) **Rewards**: How well do ReWiND rewards correlate with task progress and success?
(**Q2**) **Policy Learning**: Can ReWiND quickly train policies for new tasks?
(**Q3**) **Ablations and Analysis**: Which ReWiND design decisions are most significant?

4.1 Q1: What Makes a Good Reward Function?

We repeat the desiderata from Section 3.1 that we set out to achieve with ReWiND: (1) generalization to new tasks, (2) rewards aligned with videos from *policy rollouts*, and (3) robustness to diverse inputs. We structure this section to demonstrate ReWiND's ability to satisfy these criteria.

We compare ReWiND-learned rewards against all relevant reward learning baselines from Section 2: **LIV** [58] is a robotics reward model pre-trained on EpicKitchens [59], we also fine-tune LIV on $\mathcal{D}_{\text{demos}}$ (**LIV-FT**); **RoboCLIP** [6] uses a pre-trained video language model, S3D [60] trained on HowTo100M [61], to reward agents for language specified tasks; **Video-Language Critic (VLC)** [13] fine-tunes a VLM with a sequential ranking objective to encourage frames later in the video to have higher rewards. We train it on $\mathcal{D}_{\text{demos}}$; **Generative Value Learning (GVL)** [43] prompts a pre-trained Gemini LLM [62] with shuffled frames to predict per-frame progress.

We conduct our primary reward analysis using the simulated MetaWorld benchmark [63] because it enables efficient collection of exemplar failed and partially successful rollout videos for analysis. Smaller-scale real-world analyses, **strongly aligned** with simulation, are in Appendix D.3. $\mathcal{D}_{\text{demos}}$ here consists of 20 tasks with 5 expert demos each. For fair comparison, we include a variant of ReWiND trained without $\mathcal{D}_{\text{open-x}}$ (**ReWiND w/o OXE**). Results are evaluated on 17 unseen MetaWorld tasks. These tasks are visually similar to training tasks but require new motions to solve (e.g., `Door-Open` $\rightarrow$ `Door-Close`). We average metrics across 5 demos per task.

Generalization. We first evaluate how effectively each reward model distinguishes unseen tasks using confusion matrices of unseen task videos versus language instructions (Figure 4). Ideally, a clear blue diagonal indicates correct video-instruction pairs, with low (white) values elsewhere. ReWiND produces clearest disparity between the diagonal and off-diagonal elements, excelling even without OXE due to architectural choices aimed at generalization, i.e., first frame positional encodings and frozen pre-trained input embeddings.

Next, we evaluate how consistently rewards reflect progress over time in successful, unseen demonstrations. We report Pearson correlation (r) of each model's per-frame reward against time and also Spearman's rank correlation (ρ), which, unlike r, captures monotonicity regardless of linearity. As shown in Table 1**(a)**, ReWiND again outperforms all baselines—achieving a **30%** relative improvement in r and **27%** in ρ over the best alternative (VLC).

Policy Rollout Reward Alignment. We also find that ReWiND can properly reward *failed* policy rollouts, which is important for rewarding RL policies on unseen tasks. For each task, we train

Table 1: **Combined Evaluation Metrics.** Comparison of reward models across three axes: (1) Demo Video Reward Alignment, (2) Policy Rollout Reward Ranking, and (3) Input Robustness.

Category	Metric	LIV	LIV-FT	RoboCLIP	VLC	GVL	ReWiND w/o OXE	ReWiND w/ OXE
(a) Demo Reward Alignment	$r \uparrow$	-0.03	0.55	0.01	0.64	0.52	0.67	**0.83**
	$\rho \uparrow$	-0.04	0.55	-0.01	0.62	0.57	0.64	**0.79**
(b) Policy Rollout Ranking	Rew. Order $\rho \uparrow$	-0.32	0.47	0.00	-0.18	0.32	0.76	**0.82**
	Rew. Diff. $\uparrow$	-0.16	0.26	0.06	-0.15	0.17	0.39	**0.41**
(c) Input Robustness	Avg. $\rho \uparrow$	0.03	0.27	0.00	0.60	0.58	0.55	**0.74**
	ρ Variance $\downarrow$	0.08	0.28	**0.00**	**0.00**	0.01	0.03	0.04

an SAC [64] policy from scratch and use trajectories collected from various points of training to construct three evaluation video datasets: `failure`, `near-success`, and `success` containing failed trajectories, trajectories where the policy was close to the goal state but did not succeed, and successful trajectories, respectively. Each task has 2 trajectories of each type.

We evaluate each dataset's relative alignment *ranking* (measured by Spearman's ρ) with each reward model. For example, for a given task, if the average reward for a `failure` video is 0.1, a `near-success` video is 0.5, and `success` video is 0.9, then the rankings would be 1, 2, 3, respectively, where 3 corresponds to the best ranking. Thus, ρ over rankings tells us how often the videos are correctly ranked. We report the ranking in Table 1**(b)**. We also report the average *difference* between rewards for `success` with `near-success` and `near-success` with `failure` videos. Overall, due to *video rewind*, ReWiND has a relative **74%** improvement in reward order and **58%** improvement in reward difference over the best baseline, LIV-FT. Additionally, we qualitatively demonstrate how these rankings translate into policy rollout rewards in Appendix Figure 8 by plotting reward predictions of ReWiND against reward baselines for an unsuccessful policy rollout.

Robustness to Varied Inputs. Finally, we demonstrate ReWiND's robustness to diverse instructions. For each evaluation task, we manually create three additional language instructions (without prior knowledge of ReWiND's performance), resulting in four total instructions per task. For example, "close the door" is an original instruction, and we add "shut the door." Each set of instructions is paired with a single demonstration video, and we compare the reward models by measuring their average Spearman's rank correlation (ρ) and output variance across these instructions in Table 1**(c)**. Higher variance indicates lower robustnes. Again, ReWiND outperforms baselines, achieving the highest average correlation (0.74), **23%** better than VLC, and near-zero variance, even without OXE training—likely aided by our instruction augmentation approach (Section 3.1.1). RoboCLIP and VLC show near-zero variance but achieve significantly lower correlation scores.

So far, our results demonstrate that **ReWiND significantly outperforms all image-language-conditioned reward baselines** in terms of **generalization**, rewarding **policy rollouts**, and input **robustness**. We next demonstrate how these results translate into sample-efficient policy learning.

4.2 Q2: Learning New Tasks with RL

Simulation. We use the MetaWorld simulation benchmark [63], where we pre-train reward models and policies on 20 tasks, each with 5 per-task demos collected from a scripted policy. We evaluate on 8 unseen tasks in MetaWorld, chosen for reasonable initial policy rollout behaviors, across 3 seeds each. We compare ReWiND against the 2 language-conditioned reward model baselines that performed best in reward alignment (**VLC**) and policy rollout rankings (**LIV-FT**) from the reward analysis in Section 4.1. We also compare against **Sparse**, which pre-trains and fine-tunes on only the sparse success reward bonus, and **Pre-train**, which pre-trains on sparse reward and is evaluated zero-shot on new tasks. All baselines are image, proprio (x, y, z, gripper),

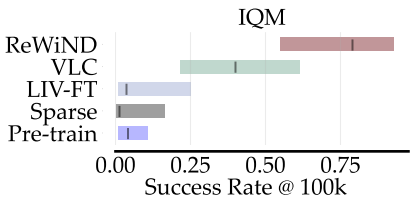


Figure 5: **MetaWorld final performance.** We plot inter-quartile means (IQMs) of success rates after 100k environment steps on 8 unseen tasks in MetaWorld. ReWiND achieves 79%.

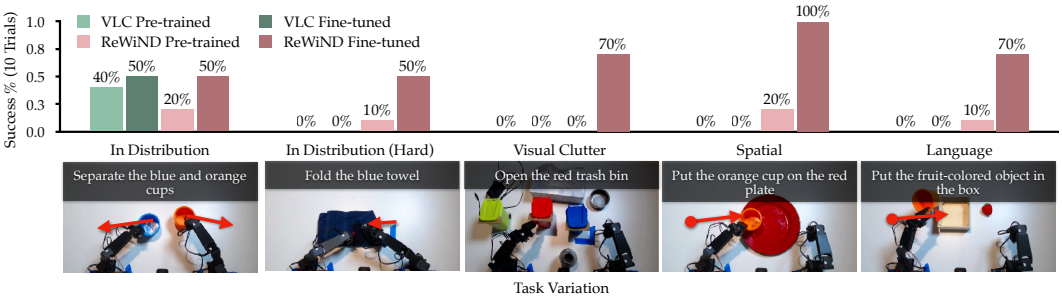


Figure 6: **Real-robot RL.** We present results on the Koch bimanual arms across in-distribution tasks and visual, spatial, and linguistic generalization tasks. Online RL with ReWiND improves a pre-trained policy by an absolute 56% across all five tasks.

and language conditioned. Each method uses the same pre-training and RL procedure as ReWiND (Section 3.2), and is trained online for 100k timesteps. See Appendix B for more experiment details.

As recommended by Agarwal et al. [65], we report the interquartile mean (IQM) and 95% confidence intervals computed over all task success rates at 100k environment steps in Figure 5. Sparse reward fine-tuning and Pre-train (no fine-tuning) result in near-zero success rates, highlighting the difficulty of image-based new task learning under limited data. In fact, Sparse reward fine-tuning, which relies purely on a sparse success bonus, performs worse than Pre-train after fine-tuning. Meanwhile, ReWiND achieves an IQM success rate of **79%**, a **97.5%** improvement over the best baseline, VLC, demonstrating that ReWiND effectively enables the policy to learn new tasks in MetaWorld. These results are well-aligned with our reward analysis in Section 4.1, demonstrating how they correlate with policy learning performance. ReWiND is also more sample-efficient at timesteps less than 100k; see extended discussion in Appendix D.2 and sample efficiency curves in Figure 10.

Real-World Robot Learning. We conduct real-world tabletop manipulation experiments with a bimanual Koch v1.1 robot arm setup [66]. We use 5 demos to train the reward function, but 10 for the policy, as we found policy learning to be a bottleneck on this difficult robot embodiment. Across five tasks, we demonstrate in Figure 6 that an hour of real-world RL with ReWiND improves the success rate over the base pre-trained policy from an average 12% success rate to 68%, a **5**× improvement. Meanwhile, VLC only improves from 8% to 10%—ReWiND outperforms VLC, the best simulation baseline, by **6.7**×. RL for an hour of real-world experiment time corresponds to 50k environment steps with our parallelized codebase that trains the policy while an older checkpoint gathers data in the environment to avoid any training wait time. We select diverse tasks that demonstrate real-world improvement based on generalization metrics defined in prior work [48, 49] on: an in-distribution task, `separate the blue and orange cups`; an in-distribution *difficult* task, `fold the blue towel`; an unseen task in terms of large amounts of *visual* clutter, `open the red trash bin`; an unseen task in terms of spatial relationships between objects requiring *new action sequences*, `put the orange cup on the red plate`; and an unseen task in terms of *language* input, `put the fruit-colored object in the box`. Overall, ReWiND enables real-world reinforcement learning on unseen tasks without requiring new demonstrations, improving over the pre-trained policy, and outperforms the best baseline from simulation, VLC. See Appendix C.1 for real-world experiment details and Figure 12 for policy rollout examples.

Q3: Ablations and Analysis. ReWiND's ability to teach policies unseen tasks comes from achieving the three desiderata listed earlier, namely generalization, providing accurate rewards for policy rollout failures, and input robustness. We demonstrate that each component of ReWiND contributes to at least one of each desiderata in Appendix E where we ablate instruction augmentation, video rewinding, the use of $\mathcal{D}_{\text{demos}}$, and first-frame positional embeddings.

Concluding Statement. In conclusion, our experiments demonstrated ReWiND's effectiveness as a reward function for policy learning through detailed reward analyses and its effectiveness as a framework for sample-efficient robot learning of unseen tasks, both in simulation and on a real bimanual robot. Finally, we thoroughly discuss limitations and failure cases of ReWiND in Section 5.

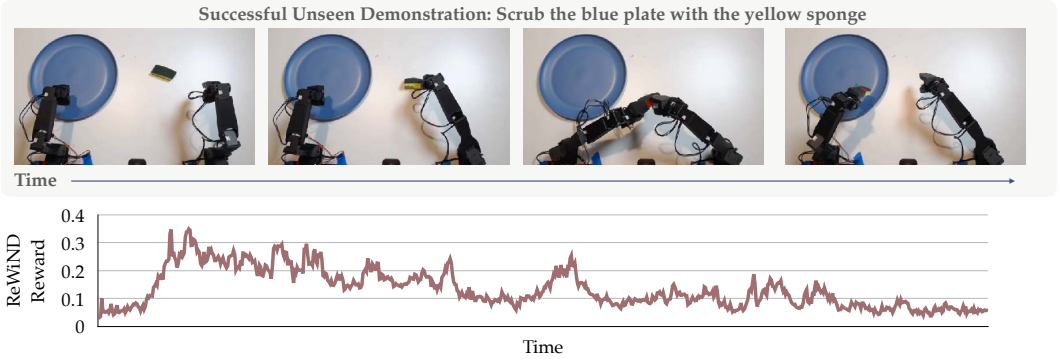


Figure 7: **ReWiND Failure Example.** We collect a demonstration of the Koch arms picking up a sponge, handing it over, and scrubbing a plate. We find that there is poor reward alignment to this successful demonstration, likely due to the lack of bimanual data in Open-X, occlusion of the sponge, and poor camera viewpoints.

5 Limitations

Here we discuss limitations of ReWiND and directions for future work.

Initial Policy Performance. Firstly, we use a relatively simple policy architecture that is trained from scratch for each of our domains. We expect better performance by combining ReWiND with stronger policy architectures capable of ingesting more data (e.g., pre-trained vision-language-action models [16, 67, 2, 68, 4]) that have better *zero-shot performance* on new tasks to enable even more sample-efficient learning irrespective of reward function.

In fact, we confirmed experimentally that initial zero-shot performance was strongly indicative of how well the policy will learn a new task in our real-world experiments. For instance, we did not find ReWiND to help a policy that confidently performs the wrong task due to a combination of the KL-constrained objective we use for offline to online learning in real world experiments (see Appendix C.2 for details) and the fact that unlearning poor behaviors takes a significant amount of time. If the ReWiND reward function could be combined with stronger policies that are easy to learn online in the loop, we hope it will enable learning of many more difficult new tasks.

However, the best way to fine-tune these models with online rewards remains an open challenge [69, 70]. One bottleneck is simply that, even with low-rank adaptation techniques that prior work found to help train large policy architectures more efficiently [71, 70, 67], fine-tuning these models takes a lot of compute and real-world time that makes real-world online learning with reward difficult, albeit feasible in simulation [72]. Recent work has also explored how to use RL to improve more expressive policy classes [73, 74, 75]. We show preliminary experiments with Diffusion Steering RL [73] with ReWiND, along with discussions on potential drawbacks to such methods for learning *new* tasks. We plan to investigate blending ReWiND approaches with such policy architectures and efficient fine-tuning techniques, such as large model steering approaches [73], in the future.

Reward Objective. One limitation of using video progress as a way to label training data, whether one uses relative targets [13, 41] or fixed targets such as with ReWiND in Equation (1), is that we cannot explicitly incorporate collected *failure trajectories* into Equation (1) without labels indicating what the final task progress for the trajectory should be. Alakuijala et al. [13] incorporates failure trajectories by using them as contrastive negatives while disabling their sequential ranking loss which relies on frame progress. Similarly, ReWiND can naïvely be trained with collected failure trajectories by only using them as negatives with a progress target of 0 in Equation (1). However, future work can incorporate other objectives, use accurate VLMs, or assume human labelers to label these videos with approximate progress targets.

Reward Analysis. One of the limitations of ReWiND lies in its inherent tradeoff using pre-trained vision and language embeddings. We do not fine-tune these embeddings because our assumed demonstration dataset $\mathcal{D}_{\text{demos}}$ is very small, and in early experiments, we found that fine-tuning sometimes hurts generalization performance. Not fine-tuning the frozen embeddings may result in the reward model generally underfitting certain tasks, particular to robotics, on which the pre-trained vision and language models were not trained. In Figure 7, we visualize an example of dish scrubbing which ReWiND does not perform well on even though similar linguistic tasks exist in the Open-X dataset. This poor result is likely due to Open-X not containing any bimanual data or partial occlusion due to the camera viewpoint. Future work that pre-trains with even more robotics data or incorporates intermediate representations or objectives with large-scale pre-training on internet data (e.g., Li et al. [4], Team et al. [68]) could allow fine-tuning the input embeddings to ensure they can better fit the $\mathcal{D}_{\text{demos}}$.

Resets. ReWiND, in its current form, requires a human operator to perform resets of the environment. This assumption prevents ReWiND from being fully autonomous. However, recent reset-free RL works [42, 57, 76, 47] demonstrate promising solutions to address the need for humans to supervise learning. Regardless, human resets remain a roadblock to autonomous learning that is difficult to address in the real world [77].

Success Detection. Another limitation comes from requiring success detection for the reward bonus and terminating policy rollouts upon success. We add a success bonus (detailed in Section 3.2) to account for potential noisy rewards and imperfect success detection by the reward model, given that a human is already monitoring to reset the environment, and we subsequently terminate the rollout upon success. Methods such as those introduced by Ye et al. [47], Zhou et al. [78], Yang et al. [42], which utilize VLMs as success detectors, can remove the need for human supervision during the online phase of ReWiND when combined with reset-free RL. In future work, we plan to investigate ReWiND with reset-free approaches, guided by task performance estimates [79] and automatic success detection for truly autonomous learning.

Acknowledgments

We thank Sid Kaushik for feedback on the final draft of the paper. This work was supported in part by a grant from the Army Research Lab (ARL) Army AI Innovations Institute (A2I2), award number W911NF-23-2-0010. The claims and findings of this work do not necessarily represent the views of the ARL. This work was also supported in part by a grant from the DARPA Friction and Accountability in Conversational Transactions (FACT) program. Additionally, this work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant (No.RS2019-II190075, Artificial Intelligence Graduate School Program, KAIST) and National Research Foundation of Korea (NRF) grant (NRF-2021H1D3A2A03103683, Brain Pool Research Program), funded by the Korean government (MSIT).

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arxiv:2410.24164*, 2024.

[4] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, C. R. Garrett, F. Ramos, D. Fox, A. Li, A. Gupta, and A. Goyal. Hamster: Hierarchical action models for open-world robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2025.

[5] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[6] 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 *NeurIPS*, 2023.

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

[8] H. Hu and D. Sadigh. Language instructed reinforcement learning for human-ai coordination. In *International Conference on Machine Learning (ICML)*, 2023.

[9] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. Gonzalez Arenas, H.-T. Lewis Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia. Language to rewards for robotic skill synthesis. In *Conference on Robot Learning (CoRL)*, 2023.

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

[11] Y. J. Ma, W. Liang, H. Wang, S. Wang, Y. Zhu, L. Fan, O. Bastani, and D. Jayaraman. Dreureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)*, 2024.

[12] W. Liang, S. Wang, H.-J. Wang, O. Bastani, D. Jayaraman, and Y. J. Ma. Environment curriculum generation via large language models. In *Conference on Robot Learning (CoRL)*, 2024.

[13] M. Alakuijala, R. McLean, I. Woungang, N. Farsad, S. Kaski, P. Marttinen, and K. Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. In *Transactions on Machine Learning Research (TMLR)*, 2025.

[14] 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.

[15] 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.

[16] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. In *International Conference on Robotics and Automation (ICRA)*, 2024.

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

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

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

[20] 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.

[21] J. Ho and S. Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.

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

[23] 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 S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *NeurIPS*, 2018.

[24] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017.

[25] 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.

[26] 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.

[27] 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.

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

[29] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2020.

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

[31] N. Wilde, E. Biyik, D. Sadigh, and S. L. Smith. Learning reward functions from scale feedback. In *Conference on Robot Learning (CoRL)*, 2021.

[32] Y. Cui and S. Niekum. Active reward learning from critiques. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[33] 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.

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

[35] A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. In *Robotics: Science and Systems (RSS)*, 2021.

[36] 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.

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

[38] 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.

[39] 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.

[40] N. Nguyen, M. N. Vu, T. D. Ta, B. Huang, T. Vo, N. Le, and A. Nguyen. Robotic-clip: Fine-tuning clip on action data for robotic applications. In *International Conference on Robotics and Automation (ICRA)*, 2025.

[41] 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.

[42] J. Yang, M. S. Mark, B. Vu, A. Sharma, J. Bohg, and C. Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 2024.

[43] Y. J. Ma, J. Hejna, A. Wahid, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, J. Tompson, O. Bastani, D. Jayaraman, W. Yu, T. Zhang, D. Sadigh, and F. Xia. Vision language models are in-context value learners. In *International Conference on Learning Representations (ICLR)*, 2025.

[44] 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.

[45] C. Kim, M. Heo, D. Lee, J. Shin, H. Lee, J. J. Lim, and K. Lee. Subtask-aware visual reward learning from segmented demonstrations. In *International Conference on Learning Representations (ICLR)*, 2025.

[46] 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.

[47] W. Ye, Y. Zhang, H. Weng, X. Gu, S. Wang, T. Zhang, M. Wang, P. Abbeel, and Y. Gao. Reinforcement learning with foundation priors: Let embodied agent efficiently learn on its own. In *Conference on Robot Learning (CoRL)*, 2024.

[48] A. Anwar, R. Gupta, and J. Thomason. Contrast sets for evaluating language-guided robot policies. In *Conference on Robot Learning (CoRL)*, 2024.

[49] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh. A taxonomy for evaluating generalist robot policies. *arXiv preprint arXiv:2503.01238*, 2025.

[50] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

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

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

[53] S. Venkataraman, Y. Wang, Z. Wang, Z. Erickson, and D. Held. Real-world offline reinforcement learning from vision language model feedback. In *arXiv preprint arXiv:2411.05273*, 2024.

[54] J. Zhang, J. Zhang, K. Pertsch, Z. Liu, X. Ren, M. Chang, S.-H. Sun, and J. J. Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. In *Conference on Robot Learning (CoRL)*, 2023.

[55] J. Zhang, K. Pertsch, J. Zhang, and J. J. Lim. Sprint: Scalable policy pre-training via language instruction relabeling. In *International Conference on Robotics and Automation (ICRA)*, 2024.

[56] J. Luo, C. Xu, J. Wu, and S. Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. In *arXiv preprint arXiv:2410.21845*, 2024.

[57] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *International Conference on Robotics and Automation (ICRA)*, 2021.

[58] 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.

[59] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, J. Ma, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130: 33–55, 2022.

[60] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *European Conference on Computer Vision (ECCV)*, 2018.

[61] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *International Conference on Computer Vision (ICCV)*, 2019.

[62] G. Team. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2024.

[63] T. Yu, D. Quillen, Z. He, R. Julian, A. Narayan, H. Shively, A. Bellathur, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

[64] 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.

[65] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021.

[66] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, and T. Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

[67] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, 2024.

[68] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, S. Bohez, K. Bousmalis, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, O. Chang, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, D. D'Ambrosio, S. Dasari, T. Davchev, C. Devin, N. D. Palo, T. Ding, A. Dostmohamed, D. Driess, Y. Du, D. Dwibedi, M. Elabd, C. Fantacci, C. Fong, E. Frey, C. Fu, M. Giustina, K. Gopalakrishnan, L. Graesser, L. Hasenclever, N. Heess, B. Hernaez, A. Herzog, R. A. Hofer, J. Humplik, A. Iscen, M. G. Jacob, D. Jain, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, C. Kew, J. Kirkland, S. Kirmani, Y. Kuang, T. Lampe, A. Laurens, I. Leal, A. X. Lee, T.-W. E. Lee, J. Liang, Y. Lin, S. Maddineni, A. Majumdar, A. H. Michaely, R. Moreno, M. Neunert, F. Nori, C. Parada, E. Parisotto, P. Pastor, A. Pooley, K. Rao, K. Reymann, D. Sadigh, S. Saliceti, P. Sanketi, P. Sermanet, D. Shah, M. Sharma, K. Shea, C. Shu, V. Sindhwani, S. Singh, R. Soricut, J. T. Springenberg, R. Sterneck, R. Surdulescu, J. Tan, J. Tompson, V. Vanhoucke, J. Varley, G. Vesom, G. Vezzani, O. Vinyals, A. Wahid, S. Welker, P. Wohlhart, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, Y. Yang, R. Yao, S. Yaroshenko, W. Yu, W. Yuan, J. Zhang, T. Zhang, A. Zhou, and Y. Zhou. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[69] M. Nakamoto, O. Mees, A. Kumar, and S. Levine. Steering your generalists: Improving robotic foundation models via value guidance. *Conference on Robot Learning (CoRL)*, 2024.

[70] Y. Guo, J. Zhang, X. Chen, X. Ji, Y.-J. Wang, Y. Hu, and J. Chen. Improving vision-language-action model with online reinforcement learning. *arXiv preprint arXiv:2501.16664*, 2025.

[71] 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.

[72] G. Lu, W. Guo, C. Zhang, Y. Zhou, H. Jiang, Z. Gao, Y. Tang, and Z. Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. In *arXiv preprint arXiv:2505.18719*, 2025.

[73] 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 *arXiv preprint arXiv:2506.15799*, 2025.

[74] P. Dong, Q. Li, D. Sadigh, and C. Finn. Expo: Stable reinforcement learning with expressive policies. In *arXiv preprint arXiv:2507.07986*, 2025.

[75] Y. Chen, S. Tian, S. Liu, Y. Zhou, H. Li, and D. Zhao. ConRFT: A reinforced fine-tuning method for vla models via consistency policy. In *Robotics: Science and Systems (RSS)*, 2025.

[76] W. Montgomery, A. Ajay, C. Finn, P. Abbeel, and S. Levine. Reset-free guided policy search: Efficient deep reinforcement learning with stochastic initial states. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[77] S. Mirchandani, S. Belkhale, J. Hejna, E. Choi, M. S. Islam, and D. Sadigh. So you think you can scale up autonomous robot data collection? In *Conference on Robot Learning (CoRL)*, 2024.

[78] 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.

[79] A. Anwar, R. Gupta, Z. Merchant, S. Ghosh, W. Neiswanger, and J. Thomason. Efficient evaluation of multi-task robot policies with active experiment selection. In *Conference on Robot Learning (CoRL)*, 2025.

[80] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, A. Lee, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[81] 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.

[82] S. Dass, J. Yapeter, J. Zhang, J. Zhang, K. Pertsch, S. Nikolaidis, and J. J. Lim. Clvr jaco play dataset, 2023. URL https://github.com/clvrai/clvr_jaco_play_dataset.

[83] L. Y. Chen, S. Adebola, and K. Goldberg. Berkeley UR5 demonstration dataset. https://sites.google.com/view/berkeley-ur5/home, 2023.

[84] 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. https://sites.google.com/berkeley.edu/fanuc-manipulation, 2023.

[85] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from human videos as a versatile representation for robotics. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[86] R. Mendonca, S. Bahl, and D. Pathak. Structured world models from human videos. *Conference on Robot Learning (CoRL)*, 2023.

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

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

[89] 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.

[90] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

[91] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[92] J. Wu, S. Park, Z. Lin, J. Luo, and S. Levine. V-former: Offline RL with temporally-extended actions, 2024.

[93] 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.

[94] Z. Zhou, A. Peng, Q. Li, S. Levine, and A. Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *International Conference on Learning Representations (ICLR)*, 2025.

[95] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, 2023.

[96] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao, S. Levine, and K. Hausman. Jump-start reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

[97] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b. *arXiv preprint arXiv:2401.04088*, 2023.

[98] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[99] K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

[100] Q. Li, Z. Zhou, and S. Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

[101] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

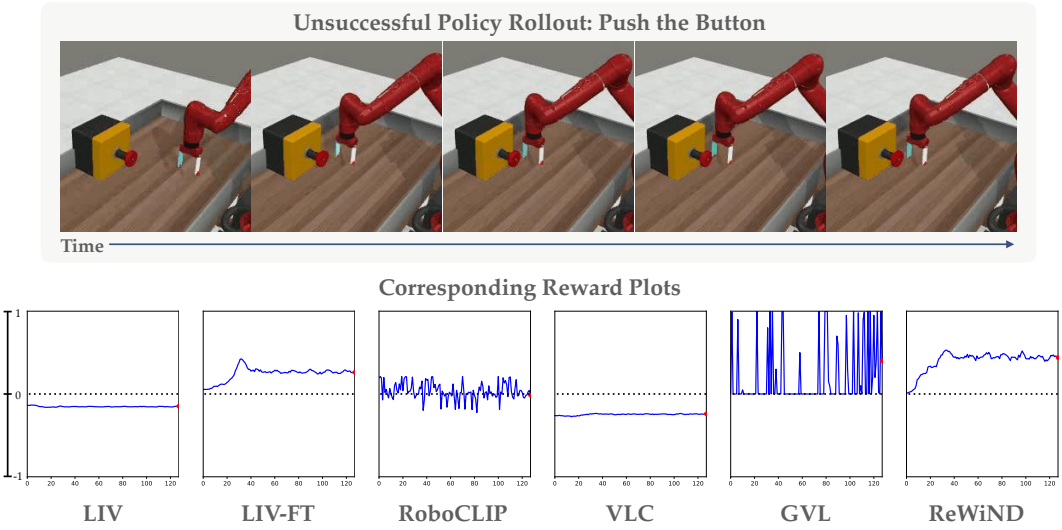


Figure 8: Unsuccessful policy rollout for the "Push the Button" task in Meta-World and its corresponding rewards below it. ReWiND predicts calibrated rewards that reflect better partial progress when the policy gets stuck near the button.

A Implementation Details

This section introduces implementation details for ReWiND in terms of the datasets, reward model, policy training, and online RL.

A.1 ReWiND Implementation

Full pseudocode for ReWiND is listed in Algorithm 1. Individual implementation details follow.

A.1.1 Open-X Dataset

Below we list details of the OXE subset, $\mathcal{D}_{\text{open-x}}$, used for training the reward model $R_\psi(o_{1:t}, z)$ (mentioned in Section 3.1).

We select a subset of datasets from the Open-X Dataset [16]. The subset includes Bridge-V2 [80], BC-Z [81], Fractal [1], CLVR Jaco Play [82], Berkeley Autolab UR5 [83], Berkeley Fanuc Manipulation [84], CMU Stretch [85, 86], Stanford Hydra [87], UCSD Kitchen [88], Austin BUDS [89], Austin Sirius [90], and DROID [91]. These datasets were selected for their high-quality, task-oriented manipulation trajectories (i.e., no play data or extremely high-level annotations). These datasets provide around 350k trajectories and 58k total unique task annotations. To ensure meaningful trajectories for training the ReWiND reward model, we postprocess the data to remove trajectories with less than 5 timesteps. We subsample the videos in the datasets to 16 frames for reward model training, as we did not see a noticeable benefit from training it with longer videos.

A.1.2 Reward Function

We picture the overall architecture of the reward function in Figure 9. We encode input images with the pre-trained DINO-V2 base model (86M params) with 768 embedding size. Similarly, we encode language with the pre-trained ALL-MINILM-L12-V2 model with a 384 embedding size. We project image and language embeddings to 512 dimensions with a single linear layer. We treat the language embedding as a single input token and we evenly downsample DINO-v2 image embeddings for every observation sequence to 16 frames.

The cross-modal sequential aggregator takes these tokens as input and produces a per-image embedding used by an MLP to produce per-timestep rewards. The cross-modal sequential aggregator

Algorithm 1 ReWiND Algorithm, Section 3.

Require: Demo dataset $\mathcal{D}_{\text{demos}}$, Pre-trained LLM, Open-X subset $\mathcal{D}_{\text{open-x}}$, Reward Model $R_\psi(o_{1:t}, z)$, Policy π. $\mathcal{D}_{\text{demos}}$ includes video trajectories $o_{1:t}$ and language embedding z.

/ Train the Reward Model* Section 3.1 */*
REWARDMODELTRAINING($R_\psi(o_{1:t}, z)$, $\mathcal{D}_{\text{demos}}$, $\mathcal{D}_{\text{open-x}}$)
/ Policy Pretraining* Section 3.2 */*
OFFLINEPOLICYPRETRAINING($R_\psi(o_{1:t}, z)$, $\mathcal{D}_{\text{demos}}$, π)
/ Learn New Task Online* Section 3.2 */*
ONLINERL(z_{new}, $R_\psi(o_{1:t}, z)$, π)

procedure REWARDMODELTRAINING($R_\psi(o_{1:t}, z)$, $\mathcal{D}_{\text{demos}}$, $\mathcal{D}_{\text{open-x}}$)
    Augment instruction labels with LLM
    Sample a video clip and annotation $o_{t_1:t_2}, z$ from $\mathcal{D}_{\text{demos}}$ or $\mathcal{D}_{\text{open-x}}$.
    Choose to keep the original video or perform REWINDAUGMENTATION.
    if perform REWINDAUGMENTATION **then**
        $o^{\text{rewound}} \leftarrow$ REWINDAUGMENTATION($o_{t_1:t_2}$)
        Optimize $R_\psi(o_{1:t}, z)$ with $\mathcal{L}_{\text{rewind}}(o^{\text{rewound}}, z)$ ▷ Equation (2)
    else
        Sample a different video clip $o^{\text{other}}_{t_1':t_2'}$
        Optimize $R_\psi(o_{1:t}, z)$ with $\mathcal{L}_{\text{progress}}(o_{t_1:t_2}, z, o^{\text{other}}_{t_1',t_2'})$ ▷ Equation (1)
    end if
end procedure

procedure OFFLINEPOLICYPRETRAINING($R_\psi(o_{1:t}, z)$, $\mathcal{D}_{\text{demos}}$, π)
    Relabel $\mathcal{D}_{\text{demos}}$ with $\hat{r}^{\text{off}}$ coming from $R_\psi(o_{1:t}, z)$. ▷ Equation (4)
    Train π with offline RL on relabeled $\mathcal{D}_{\text{demos}}$.
end procedure

procedure ONLINERL($R_\psi(o_{1:t}, z)$, π)
    For every rollout label the trajectories with $\hat{r}^{\text{on}}$ from $R_\psi(o_{1:t}, z)$. ▷ Equation (5)
    Optimize π with online RL Algorithm
end procedure

procedure REWINDAUGMENTATION($o_{t_1:t_2}$) ▷ Section 3.1.1
    Sample random split point i between t_1 and t_2.
    Sample # frames to rewind for, k
    Reverse $o_{i-k:i}$ and concat with $o_{t_1:i}$
    Return $[o_{t_1:i-1}, o_{i:i-k}]$
end procedure

is a causally masked transformer (`PyTorch nn.TransformerEncoder`) composed of 4 layers, each with 8 heads with a combined hidden dimension of 2048. We add a learnable positional embedding to only the first frame of the video sequence embedding. In the ReWind reward function training phase, we trained 2k steps for Meta-World and 10k steps for Real-World robot experiments, with a batch size of 1024. Each batch includes 80% data from $\mathcal{D}_{\text{open-x}}$ and 20% target environment data from $\mathcal{D}_{\text{demos}}$. Each video in the batch has an 80% probability of having video rewind augmentation, and independently, a 20% percent probability of having a mismatched video-language pairing with 0 progress target (see Section 3.1). For the rewind augmentation, the split point is uniformly sampled from the video clip, and the rewound sequence is obtained by taking all frames from the split point to the end of the clip. In order to better simulate policy execution videos that look *almost successful*, 10% of the rewound videos will only have their last 3 frames rewound. No extensive tuning was performed on these per-sample rewind and mismatch probabilities; they were heuristically chosen during initial small-scale experimentation and then fixed for all experiments.

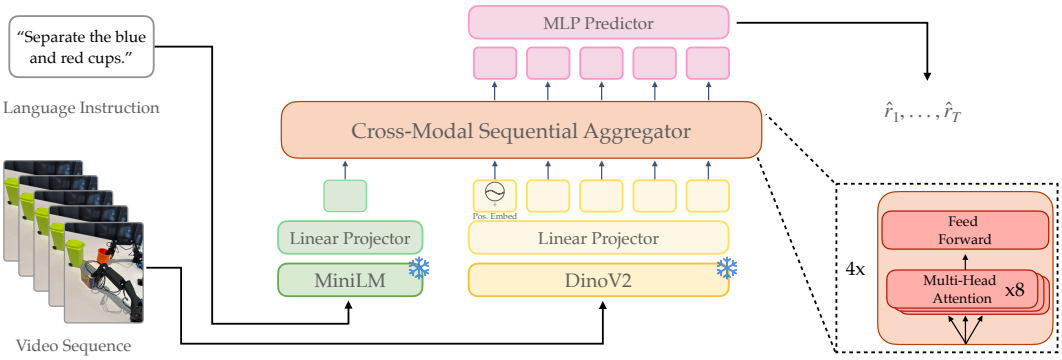


Figure 9: **ReWiND's Reward Model Architecture**. It's composed of frozen language and image input embeddings projected to a shared hidden dimension of 512. These embeddings are treated as input tokens to the cross-modal sequential aggregator transformer composed of 4 causally masked transformer layers composed of 8 multi-head attention blocks each. Per-timestep embeddings for each input observation are fed into an MLP to predict rewards for each timestep.

A.1.3 Policy Training

Specific architectural and training details are discussed per-environment in the corresponding sections Appendix B.2 and Appendix C.2. Below we talk about high-level algorithmic details for policy training along with shared implementation details across environments.

Policy Input. Similar to the reward model, we condition the policy on frozen pre-trained image and language embeddings: DINO-v2-base image embeddings (86M params, 768 embedding size) [50] along with ALL-MINILM-L12-V2 language embeddings of size 384 from the Sentence Transformers Python package [51]. We also include proprioceptive information in both of our envrionments.

Offline RL. We use Implicit Q Learning (IQL) [52] as prior work found it performant and easy to tune for robot manipulation with action-chunked policies [54, 55, 92]. IQL trains on in-distribution (s, a, s', r, a') tuples from the dataset, avoiding using next actions a' sampled from a policy, to ensure the critic functions accurately reflect returns restricted to dataset actions. The value function is optimized with expectile regression, controlled by a hyperparameter τ: $\tau = 0.5$ recovers mean squared error, while $\tau \to 1$ yields a more optimistic estimate, helping the value function "stitch" together distant rewards in sparse settings. The policy is trained via advantage-weighted regression [93], maximizing

$$e^{\beta(Q(s,a)-V(s))} \log \pi(a|s),$$

where β is a temperature hyperparameter controlling how "spiky" the policy loss is. To prevent numerical instability, the exponential term is capped at a maximum value in practice (for us, this is 100).

Online RL. We use a custom soft-actor critic (SAC) [64] implementation initialized with the pre-trained policy from offline RL along with the Q and target Q functions. We follow best practices from recent offline-online RL fine-tuning work [94, 95], namely:

- 5-10 critics instead of 2, with random sampling of critics
- LayerNorm in the critic and possibly LayerNorm in the policy
- A higher update-to-data ratio in the critics
- "Warm-starting" online RL by running with the frozen pre-trained policy for the first few thousand environment steps [94, 96]

- Possibly sampling offline pre-training data at a 50% ratio during online RL
- Removing the SAC entropy term from the target critic

We found that by default, efficient offline-online learning algorithms did not work very well "out of the box" for learning *new* tasks on our real robot. This is perhaps because they focus specifically on offline-online fine-tuning on the same task while we are trying to learn new tasks, or perhaps due to additional challenges of real-robot RL. Therefore, we make some per-environment design decisions for online RL detailed in the respective environment training sections.

B MetaWorld Experiments

B.1 Simulation Setup

Training/Eval Task Selection. We manually select 20 training tasks from MT50 benchmark in the MetaWorld environment. These tasks are used for both reward model training and policy pre-training. The training tasks include: `Button-Press`, `Button-Press-Topdown-Wall`, `Coffee-Pull`, `Dial-Turn`, `Door-Open`, `Door-Unlock`, `Drawer-Close`, `Faucet-Open`, `Handle-Press`, `Handle-Pull-Side`, `Peg-Insert-Side`, `Pick-Place`, `Plate-Slide`, `Plate-Slide-Back-Side`, `Push`, `Reach`, `Stick-Push`, `Stick-Pull`, `Window-Open`, `Hand-Insert`.

We also choose another 17 tasks from the MT50 benchmark for reward model evaluation and 8 of tasks are selected for downstream policy finetuning.[3] The evaluation tasks include `Window-Close`, `Sweep-Into`, `Soccer`, `Reach-Wall`, `Push-Back`, `Plate-Slide-Side`, `Plate-Slide-Back`, `Pick-Place-Wall`, `Handle-Pull`, `Handle-Press-Side`, `Faucet-Close`, `Door-Lock`, `Door-Close`, `Coffee-Push`, `Coffee-Button`, `Button-Press-Wall`, `Button-Press-Topdown`. We select these training tasks and target evaluation tasks to test directional and spatial (e.g., door open $\rightarrow$ close, button press $\rightarrow$ + wall) generalization scenarios where the policy is unlikely to be able to solve the task, due to requiring significantly different action sequences from the most visually similar training task, while a good reward function should be able to guide the policy to solve it. The 8 tasks used for downstream policy training are `Window-Close`, `Reach-Wall`, `Faucet-Close`, `Coffee-Button`, `Button-Press-Wall`, `Door-Lock`, `Handle-Press-Side`, `Sweep-into`.

Environment Details. We use MetaWorld [63] with the default 3rd-person camera viewpoint, pictured in Figure 10, and also 4-dimension proprioception input (x, y, z, gripper). The policy action space is the default one from MetaWorld represented as a 4-dimensional relative action space for ($\Delta x, \Delta y, \Delta z$, gripper). Unlike the MetaWorld environment setups in prior reward learning papers, we do not include goal/ground truth state information. We also terminate the environment on success. Both of these choices were made to mimic a real-world robot learning setup. The time horizon of each episode is limited to 128 steps. The success bonus for online and offline RL used in Equation (4) and Equation (5) is 200 for ReWiND and all baselines.

Figure 10: Example camera viewpoint in MetaWorld.

B.2 Training Details

For $\mathcal{D}_{\text{demos}}$, we select 20 tasks from the MT-50 benchmark. Each task consists of one human-labeled annotation, four augmented annotations (Section 3.1.1), and five optimal demonstrations

[3] These 17 tasks were chosen for sharing at least some characteristic with a training task.

Figure 11: **Real World Bimanual Robot Setup.** Our real-world setup consists of a top-down and side camera mounted to a table where two Koch v1.1 low-cost arms are mounted. This setup allows us to perform bimanual tasks and easily collect data with another pair of low-cost "leader" arms mounted to the same table.

produced by the MetaWorld built-in planner. We render images at the default resolution of 640x480, centercrop to 224x224 and embed the image with DINOv2 encoder.

We pre-train the policy with IQL [52] for 100K steps with learning rate 0.001, discount $\gamma = 0.99$. We use a three-layer MLP of size $[768, 512, 256]$ for both the policy and value function network. The general training procedure is described in Appendix A.1.3

For the various hyperparameters for online policy learning we used in MetaWorld as described in Section A.1.3. We use 10 critics and sample 2 of them during training, LayerNorm in both the critic and policy, and an update-to-data ratio of 4 for the critics. We are not sampling from offline pre-training data during online training nor are we training the target critic with the entropy term, so the implementation is identical to Warm-Start RL [94]. We warm-start online RL for 4000 steps.

C Real Robot Experiments

C.1 Robot Experiment Setup

We use the Koch1.1 bimanual arm setup for data collection and learning [66].[4] Altogether, four total arms (2 for data collection) cost ∼$1000, letting us demonstrate ReWiND enables real-world online RL of new tasks even with very low-cost hardware and noisy control. The observations consist of RGB images from a Logitech C930e top camera and side camera (pictured in Figure 11). We control the robot with absolute joint position control at a frequency of 30Hz. We collect a small dataset of 10 demonstrations over 20 tasks, and then use 5 demos per-task for the reward function. We found the offline-trained policy to be the primary bottleneck to optimizing rewards in unseen tasks, so we used 10 demos per-task for offline policy training. We have an episode timeout of 250 steps and provide a success bonus of 125 upon success (from Equation (4) and Equation (5)). Proprioceptive information in this environment includes 12 robot joint states, 6 for each arm. These represent the rotation of each joint and gripper.

C.2 Real Robot Training Details

We use a small, instruction-tuned, open-source LLM, `Mistral-7B-Instruct-v0.3` [97], to generate 9 additional instructions for each task for instruction augmentation.

For the small dataset in real robot experiments, we manually choose 15 tasks in the Koch tabletop setting, and each task includes 5 trajectories and 10 annotations. The evaluation set is 5 other random tasks, which are irrelevant with the tasks in the small dataset. We use this evaluation set for offline metrics and validating various design choices.

[4] https://github.com/jess-moss/koch-v1-1

Unlike the MetaWorld experiments that use an MLP-based policy, we use an action-chunked policy with temporal ensembling for the real robot. We found chunking to lead to more stable bimanual manipulation on the Koch arms. We implement the action chunking with a Transformer policy that predicts 60 actions at each timestep corresponding to 2 seconds of actions. We also implement a Transformer-based critic. During rollouts, we then use temporal ensembling [98]. Here, the current action is ensembled with the last 60 timesteps' predictions according to an exponential weighting scheme $w_i = \exp(-m * i)$, where we use $m = 0.01$ or $m = 0.1$ depending on the task. We found $m = 0.1$ to work well for tasks requiring grasping solid objects as it weights recent actions more heavily, necessary for ensuring the policy actually commits to the grasp, and $m = 0.01$ to work well for non-grasping tasks as it results in a smoother policy.

The policy is a Transformer decoder with 1 layer and 8 heads with 1.5M params. The critic is a Transformer encoder with 8 heads and 1 layer. We train each policy for 20k steps offline on our offline dataset using IQL with AWR for policy extraction. We train using a batch size of 256, use 5 critics, and subsample 2 critics at each training step. We use LayerNorm only in the critics as we found that LayerNorm in the action-chunked policy could potentially hurt RL performance. We also warm-start online RL for 3000 steps. We do not sample actions during policy rollouts as we found action sampling to conflict with temporal ensembling.

Then, we train the policy online as described in Section 3.2. We train online for 50k environment steps, which takes approximately 1 hour as there is minimal waiting time for policy training due to a threaded implementation that trains the policy while the last iteration's policy checkpoint is used for rollouts. This parallelization nearly doubles the rate at which we are able to collect policy rollouts. Specifically, during online training, we collect a single rollout corresponding to 250 environment steps while simultaneously training the policy for 75 gradient steps. We keep a relatively low policy to environment update ratio in order to ensure that we do not have to wait for offline training to finish in order to start the next online rollout. At each gradient step, we sample our buffer such that 50% is the offline training data, 25% is online failure trajectories, and 25% is online successful trajectories. This sampling approach helps upsample successful online trajectories. For every actor gradient step, we do 5 critic update steps to more quickly train the critic online.

During real-world policy rollouts, it is important for the robot to take safe actions that will not crash into other objects or the table. However, we found that when regularizing the policy's KL divergence against a max-entropy prior as is the case in the entropy maximization objective in standard SAC [64], the growing entropy term would cause the policy to produce largely random actions. Therefore, we regularize against the pretrained policy's distribution to encourage reasonable behaviors throughout the process of learning, similar to the SAC update rules from Pertsch et al. [99]. Thus the π and Q updates follow:

$$\pi \leftarrow \max_{\pi} \mathbb{E}_{\pi}\Big[Q(o,a,z) - \alpha \underbrace{\mathrm{KL}(\pi(\cdot|o,z) \;||\; \pi_{\text{pretrained}}(\cdot|o,z)}_{\text{pre-trained policy guidance}}\Big] \tag{6}$$

$$Q \leftarrow \min_{Q} Q(o,a,z) = r + \gamma\Big[Q(o',a,z) - \alpha \underbrace{\mathrm{KL}\left(\pi(\cdot|o,z) \;||\; \pi_{\text{pretrained}}(\cdot|o,z)\right)}_{\text{pre-trained policy guidance}}\Big] \tag{7}$$

We set α in both equations to a fixed value of 10.0 on tasks where grasping solid objects is not required. For others, we set it to 20.0 to ensure the policy doesn't degenerate from its grasping action early in training. We found that lower KL penalties could result in the policy falling into locally optimal but globally suboptimal behaviors, such as moving a cup with the arm instead of actually picking it up. Work released after our paper further explores RL in chunked action spaces with similar KL-regularized behavior constraints, and find that such constraints and chunking helps improve exploration during RL [100]

C.3 Real Robot Tasks

We collected 10 demos per-task over 20 tasks on the Koch arms. We train the reward function on 5 demos per-task and the policy on 10 demos per-task. We selected these training and evaluation

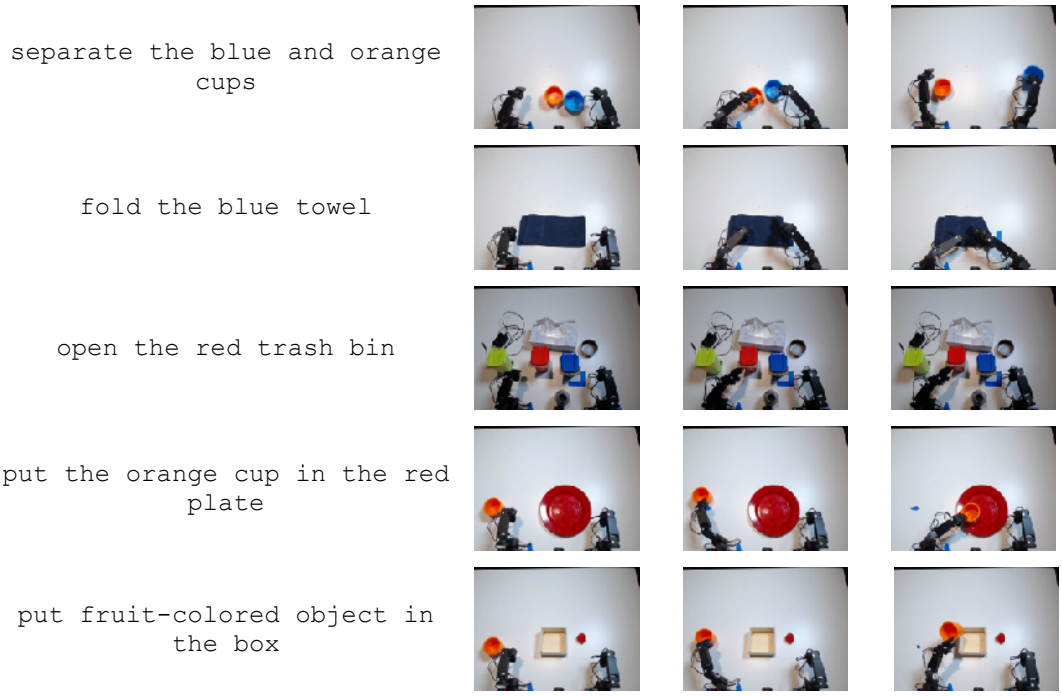


Figure 12: We present rollouts for the 5 tasks we use for online RL. The first two tasks are in-distribution to the policy, while the latter 3 tasks are out-of-distribution with respect to visual, spatial, or semantic generalization.

tasks specifically to evaluate spatial, visual, and language generalization. We list these training tasks below.

`Move the orange cup from the left to the right`, `Move the orange cup from the right to the left`, `Put the orange cup on the red plate`, `Put the red cup on the red plate`, `Separate the blue and red cups`, `Fold the blue towel`, `Open the green trash bin`, `Open the blue trash bin`, `Throw the banana away in the green trash bin`, `Throw the banana away in the blue trash bin`, `Put the red marker in the red trash can`, `Put the pink marker in the green trash can`, `Put the blue tape in the box on the left`, `Put the banana in the box`, `Put the orange cup in the box`, `Put the blue cup on the red plate`, `Separate the orange and blue cups`, `Open the red trash bin`, `Throw the banana away in the red trash bin`, `Put the red tape in the box on the right`.

In addition, we present rollouts of the five online tasks in Figure 12. We also provide additional descriptions of these tasks below:

- `Separate the blue and orange cups`: the robot must separate the two cups in the middle
- `Fold the blue towel`: the robot must fold the towel in half.
- `Open the red trash bin`: the robot is surrounded by clutter compared to the training data above and must open the trash bin
- `Put the orange cup in the red plate`: the robot picks an orange cup and must place it on a plate that is further away from the training data distribution
- `Put fruit-colored object in the box`: we refer to a "fruit-colored" object to test the robot's ability to handle semantic generalization.

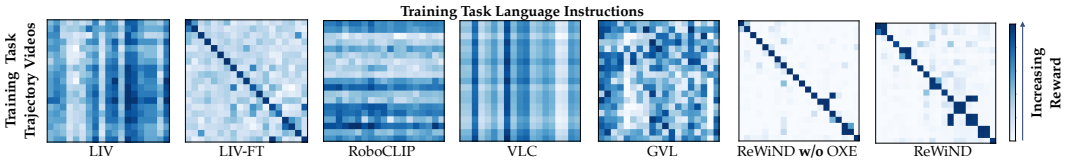


Figure 13: **MetaWorld Reward Confusion Matrix on 20 Training Tasks.** For each training task, we compute rewards for all combinations of demonstration videos and language descriptions. ReWiND produces the most diagonal-heavy confusion matrix, indicating strong alignment between unseen demos and instructions.

D Additional Results

D.1 Additional MetaWorld Reward Analysis

In Figure 13 we plot the confusion matrices of different reward models on training tasks in addition to the evaluation task plots of MetaWorld in Figure 4. LIV, RoboCLIP and GVL are not pretrained or fine-tuned on the etraining tasks while VLC, LIV-FT and ReWiND are. We can see both ReWiND w/ OXE data $\mathcal{D}_{\text{open-x}}$ and ReWiND w/o OXE data $\mathcal{D}_{\text{open-x}}$ are the best, having the clearest disparity between the diagonal and off-diagonal elements. LIV-FT also works well with a diagonal-heavy matrix. However, its disparity is not as clear as ReWiND.

D.2 MetaWorld Sample Efficiency Results

In this section, we analyze the sample efficiency of ReWiND against baselines in MetaWorld. Figure 14 plots the learning curves for all downstream policy training tasks. Each panel corresponds to one specific task. And Figure 14i displays the average of all 8 downstream tasks we used for policy fine-tuning. We can see from the average IQM plot that ReWiND achieves higher success rate than other baselines with the same number of timesteps and ReWiND is generally more sample-efficient at any timestep.

D.3 Real-World Reward Analysis

We evaluated the performance of ReWiND in MetaWorld in Section 4.1. In this section, we analyze how ReWiND works with real-world data. For the real-world setup, we use both views of each trajectory, treated as separate videos (but from the same demonstration) to train and evaluate all models. It can be seen from Table 2 that ReWiND has the highest Spearman's rank correlation (ρ) and Pearson's rank correlation (r) among all reward models. Also, in Figure 15 and Figure 16, ReWiND has the best alignment between true-paired video and language instruction in both training tasks and unseen tasks, displaying strong generalization in new tasks. Note that LIV, GVL, and RoboCLIP are not trained on these training tasks as they are zero-shot models.

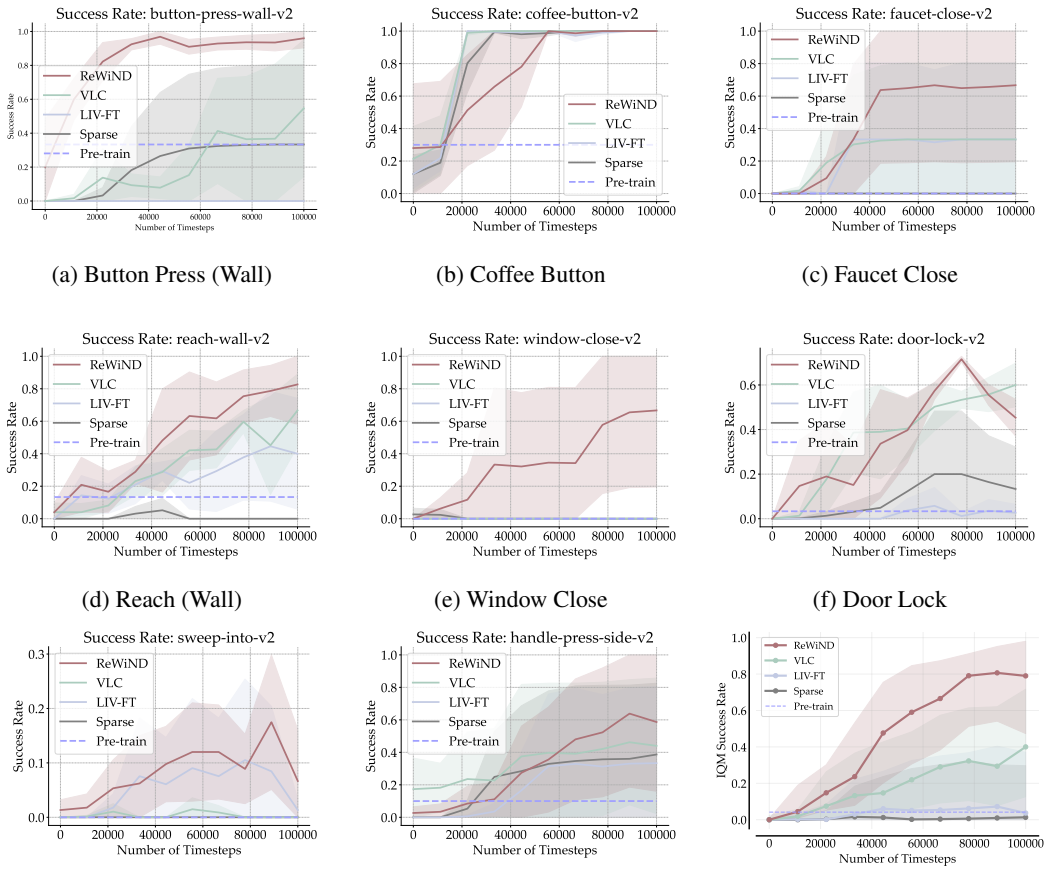
Table 2: **Evaluation Metrics on Real-world Unseen Tasks:** Comparsion between reward models in real-world unseen tasks with rank correlation ρ and r.

Model	LIV	LIV-FT	RoboCLIP	VLC	GVL	ReWiND
$\rho \uparrow$	0.22	-0.18	0.04	0.20	0.57	**0.91**
$r \uparrow$	0.23	-0.13	0.04	0.19	0.52	**0.91**

E Ablation Study

In this section, we perform an ablation study of various ReWiND reward components.

We perform a thorough ablation study of ReWiND regarding how specific design choices influence demonstration reward alignment, policy rollout ranking, and input robustness metrics introduced in Section 4.1. We ablate: instruction generation and video rewinding (Section 3.1.1); using OXE data;



(a) Button Press (Wall) (b) Coffee Button (c) Faucet Close

(d) Reach (Wall) (e) Window Close (f) Door Lock

(g) Sweep Into (h) Handle Press (Side) (i) All Tasks IQM and 95% CI

Figure 14: **MetaWorld success curves.** Task-level success rate learning curves plotting mean and shaded standard deviations. The bottom right figure plots the overall average across all tasks in terms of IQM and 95% confidence intervals.

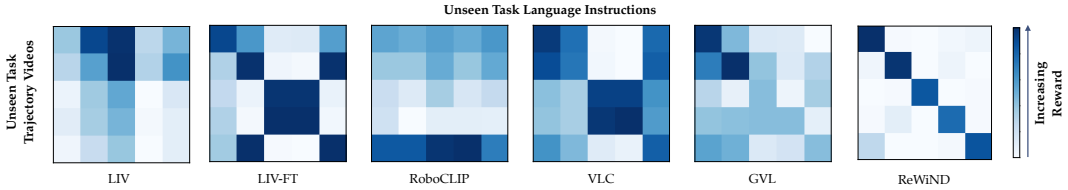


Figure 15: **Real-world Koch Reward Confusion Matrix on 5 Unseen Tasks.** For each unseen task, we compute rewards for all combinations of demonstration videos and language descriptions. ReWiND produces the most diagonal-heavy confusion matrix, indicating strong alignment between unseen demos and instructions.

the need for target environment data $\mathcal{D}_{\text{demos}}$; and finally, the use of first frame vs. full frame positional embeddings on the input observation sequence $o_{1:T}$ in the cross-modal sequential aggregator (Section 3.1.2). Overall, the original ReWiND model performs best across most metrics. Below, we analyze the impact of each ablation:

Datasets. Removing target environment data (**−Targ. Env Data**)—i.e., using only $\mathcal{D}_{\text{open-x}}$ data without $\mathcal{D}_{\text{demos}}$—leads to poor alignment with training demonstrations (Table 3a) and fails to distinguish between failed, near-successful, and successful policy rollouts (Table 3b). However, it retains strong input robustness due to the diversity of OXE data. Meanwhile, removing the Open-X subset

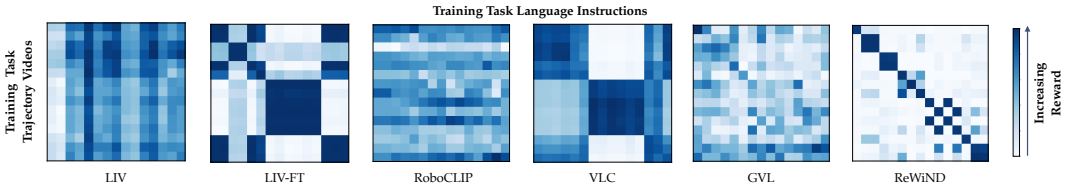


Figure 16: **Real-world Koch Reward Confusion Matrix on 15 Training Tasks.** For each training task, we compute rewards for all combinations of demonstration videos and language descriptions. LIV-FT, VLC and ReWiND are pretrained or fine-tuned with these training tasks while LIV , GVL and RoboCLIP are not

Table 3: **Ablation Study:** subtracting (−) and adding + various ReWiND components on Meta-World training and evaluation task (a) demo reward alignment, evaluation task (b) policy rollout ranking order and reward difference, and evaluation task (c) input robustness.

Model	(a) Demo Reward Alignment		(b) Policy Rollout Ranking		(c) Input Robustness	
	Train Demos $\rho \uparrow$	Unseen Demo $\rho \uparrow$	Rew. Order $\rho \uparrow$	Rew. Diff. $\uparrow$	Avg. $\rho \uparrow$	ρ Variance $\downarrow$
Original ReWiND	**1.00**	0.79	**0.82**	**0.41**	0.74	0.04
− **Targ. Env Data**	0.55	0.77	0.18	0.08	**0.78**	0.04
− **Open-X Subset**	**1.00**	0.64	0.76	0.39	0.55	0.03
− **Video Rewind**	**1.00**	0.69	0.56	0.27	0.66	**0.02**
− **Instr. Generation**	**1.00**	0.66	0.62	0.30	0.52	0.07
+ **Full Pos. Embeds**	0.99	**0.85**	0.71	0.33	**0.78**	0.06

(−**Open-X Subset**) harms unseen task reward alignment (Table 3a) and input robustness (Table 3c), highlighting the importance of OXE data for generalizing across varied language instructions.

Augmentation. Eliminating video rewinding (−**Video Rewind**) degrades rollout ranking performance (Table 3b), showing that rewinding helps distinguish failed rollouts as intended. This variant performs similarly to the single-image LIV-FT baseline in Table 1, indicating that video rewinding more effectively captures the temporal information in the videos. It also reduces policy learning performance by 33% as seen in Table 4. Similarly, removing instruction generation (−**Instruction Generation**) reduces performance on language input robustness (Table 3c), confirming that LLM-generated instructions enhance robustness to diverse inputs.

Architecture. Adding full positional embeddings (+**Full Pos. Embeds**) improves unseen demo alignment (Table 3a) but worsens rollout ranking (Table 3b), likely due to overfitting—where the model learns to predict increasing rewards regardless of input. This overfitting results in 21% inferior relative policy success rate, as seen in Table 4. To avoid overfitting, the main ReWiND model uses only first-frame positional embeddings (Section 3.1.2).

Less Training Tasks. Finally, we additionally check the performance of ReWiND with less training tasks: 5 and 10 instead of the original 20. We can see in Table 4 that the reward model still performs well even with just 5 tasks of training data instead of 20, but policy success rates understandably drop with fewer tasks.

Table 4: **Ablation Policy Performance results**: Select reward metrics and policy success rates for ablations, normalized out of 1.00 by ReWiND's original performance.

ReWiND Variant	**Rank Order Corr.** $\uparrow$	**Avg. Robustness Corr.** $\uparrow$	**Policy Succ. Rate** $\uparrow$
5 Task	0.76	0.86	0.52
10 Task	0.90	0.88	0.47
+ Full PE	0.71	**1.05**	0.69
- Video Rewind	0.87	0.89	0.67
ReWiND	**1.00**	1.00	**1.00**

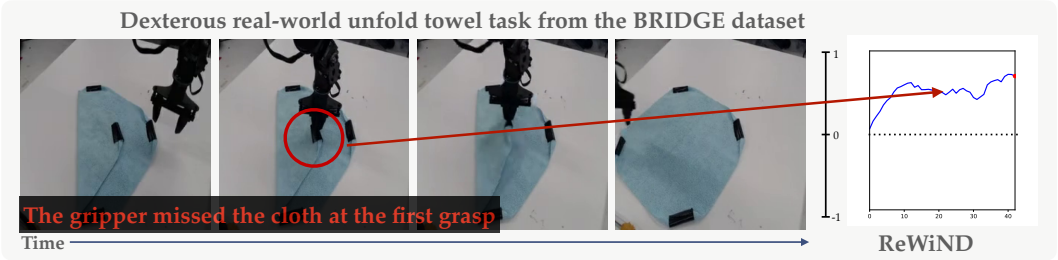


Figure 17: ReWiND reward for a contact-rich BRIDGE task. Rewards stagnate when the gripper misses, then increase as the task succeeds.

F Additional Analysis and Results

Contact Rich BRIDGE Example. We additionally plot an example of a ReWiND's predictions for a contact-rich unfolding towel task from the BRIDGE-v2 dataset [80] contained in the OXE [16] training data used to train ReWiNDin Figure 17. Even though the ReWiND reward function was not trained for many training steps overall in order to prevent overfitting (2k in MetaWorld and 10K in the Real-World, see Appendix A.1.2), it can perform well on this BRIDGE example. Notice specifically that ReWiND rewards stagnate when the gripper first misses the cloth grasp, and then later increase again as it succeeds on the second retry.

Diffusion Steering. We conducted preliminary experiments with ReWiND on more expressive policy classes, namely diffusion policies (DPs) [101]. We use Diffusion Steering RL (DSRL) [73] for online RL of DPs. Rather than optimizing the weights of the DP, DSRL runs RL over the latent-noise space of a diffusion policy. This latent space encodes an action prior that RL can refine to improve DPs. The RL policy, conditioned on image and instruction, was trained offline with ReWiND rewards and finetuned online in MetaWorld (Appendix B.2). For seen tasks, this approach improved base DPs: offline RL learned useful priors that accelerated online RL. However, when we attempted to learn *new* tasks, we found that the action priors learned from offline RL was not representative of the new task as the instruction (OOD and data-sparse regime). We thus trained the latent policy only on images (while keeping the DP instruction-conditioned) As a preliminary result, this DSRL+ReWiND implemented improved DP's performance on an unseen task **0%** $\rightarrow$ **20%**. As such, ReWiND is agnostic to policy architectures: single-action-output MLP architectures in MetaWorld, action-chunked transformers in the real world, and now also DPs via diffusion steering.