ReWiND: Learning New Tasks from Language 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]
{jzhang96,yusenluo,abraranw,jessetho,biyik,jessez}@usc.edu,
Sumedhso@amazon.com, joe.lim@kaist.ac.kr

[1]**University of Southern California**
[2]**Amazon Robotics**
[3]**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, beating baselines by $2\times$ in simulation and improving real-world pretrained bimanual policies by $5\times$, taking a step towards scalable, real-world robot learning. See website at https://rewind-reward.github.io/.

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 that can teach 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). 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 (Sutton & Barto, 2018). Recent progress in language-conditioned reward learning (Sontakke et al., 2023; Kwon et al., 2023; Hu & Sadigh, 2023; Yu et al., 2023; Ma et al., 2024a;b; Liang et al., 2024; Alakuijala et al., 2025; Wang et al., 2024; Yang et al., 2024c) has aimed at addressing these chal-

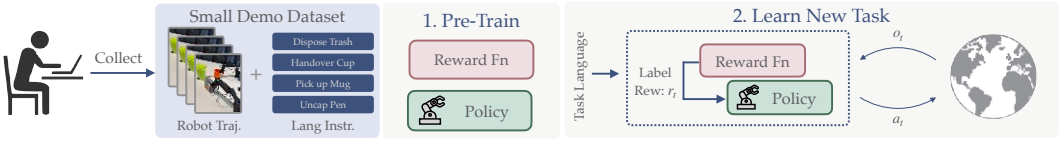


Figure 1: **Overview.** We pre-train a policy and reward model from a small set of language-labeled demos. Then, we solve unseen task variations via language-guided RL *without additional demos*.

lenges, but often assumes unrealistic conditions such as availability of ground-truth states (Kwon et al., 2023; Hu & Sadigh, 2023; Yu et al., 2023; Ma et al., 2024a;b; Liang et al., 2024), thousands of demonstrations (Alakuijala et al., 2025), or online training of reward models from scratch (Wang et al., 2024; Yang et al., 2024c), limiting their practical applicability.

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. We also introduce *video rewinding* to automatically generate failure trajectories from successful demos, allowing ReWiND to provide dense feedback even when the policy is making mistakes. Then, to encourage *generalization* across unseen tasks and *robustness* to diverse behaviors, we train the cross-modal sequential aggregator with pre-trained vision and language embeddings, selectively applied positional embeddings, and diverse robotics data from Open-X dataset (Collaboration et al., 2024) . Focusing on these 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 (Ng & Russell, 2000; Abbeel & Ng, 2004; Ziebart et al., 2008; Finn et al., 2016), where reward functions are learned from demonstrations, or methods where rewards are implicitly learned from expert or goal state distributions (Ho & Ermon, 2016; Fu et al., 2018a;b). 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 Christiano et al. (2017); Sadigh et al. (2017); Biyik et al. (2020); Lee et al. (2021); Hejna & Sadigh (2022), reward sketches Cabi et al. (2020), preference rankings Myers et al. (2021), scaled preferences Wilde et al. (2021), critiques Cui & Niekum (2018), corrections Bajcsy et al. (2018), interventions Korkmaz & Bıyık (2025), and language Yang et al. (2024c). While these feedbacks may require less human effort, but still require humans to provide extensive feedback for new tasks.

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 (Kwon et al., 2023; Hu & Sadigh, 2023; Yu et al., 2023; Ma et al., 2024a;b; Liang et al., 2024), but they typically rely on ground-truth state

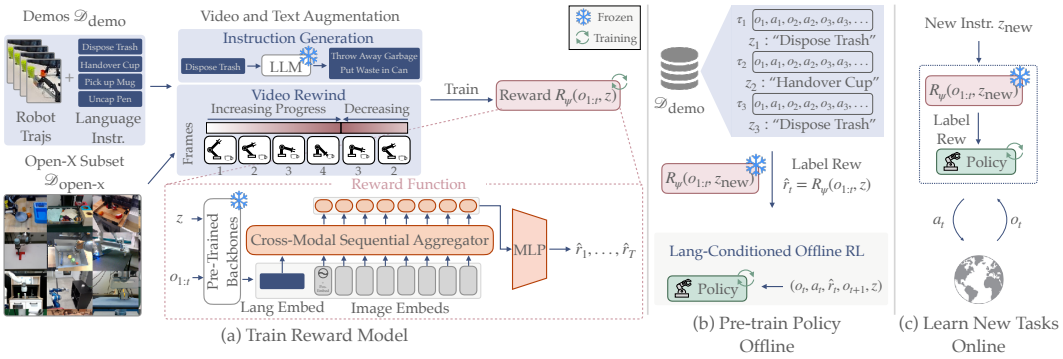


Figure 2: **(a):** We train a reward model $R_\psi(o_{1:t}, z)$ on a small demonstration dataset $\mathcal{D}_{\text{demos}}$ and a curated subset of Open-X, $\mathcal{D}_{\text{open-x}}$, augmented with LLM-generated instructions and video rewinding. $R_\psi(o_{1:t}, z)$ predicts video *progress* rewards $\hat{r}_{1:T}$ 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 π with online rollouts and reward labels from $R_\psi(o_{1:t}, z_{\text{new}})$.

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 (Chen et al., 2021; Cui et al., 2022; Fan et al., 2022; Nottingham et al., 2023; Nam et al., 2023; Sontakke et al., 2023; Wang et al., 2024; Alakuijala et al., 2025; Nguyen et al., 2025; Yang et al., 2024a;b; Ma et al., 2025; Rocamonde et al., 2024; Kim et al., 2025). Among these, RoboCLIP (Sontakke et al., 2023), LIV (Ma et al., 2024a), VLC (Alakuijala et al., 2025), and GVL (Ma et al., 2024b)—like ReWiND—reward unseen robot manipulation tasks directly from language without additional target-task demos or online tuning. We show in Section 4.1 that these baselines underperform ReWiND in rewarding policies in our limited-data setting. Most similar to ReWiND, Foundation Actor-Critic (FAC) (Ye et al., 2024) enables efficient RL from language via potential-based shaping rewards from a pre-trained VLM. However, FAC 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 (Anwar et al., 2024; Gao et al., 2025), 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 (Yang et al., 2024a; Alakuijala et al., 2025), our progress-based objective provides fixed targets that are more stable to train on, 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 (Collaboration et al., 2024) 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 (Collaboration et al., 2024), 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 (Alakuijala et al., 2025), 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.

Instead, we address this problem in a scalable manner by randomly *rewinding videos*. [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:

[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.

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

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.

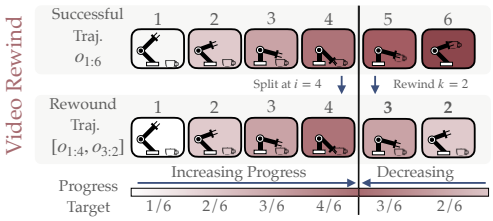


Figure 3: **Video rewind.** We split a demo at intermediate timestep i into forward/reverse sections. Here, the forward section shows the robot approaching the cup; the reverse section $(o_{i-1}, o_{i-2}, \ldots)$ resembles dropping it.

Frozen Input Encoders. We use frozen image and language encoders as the backbone of $R_\psi(o_{1:t}, z)$: we use DINOv2 (Oquab et al., 2024) for image encoding due to its strong object-centric representations and `all-MiniLM-L12-v2` (Reimers & Gurevych, 2019) 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. (2025) 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. For full implementation details, see Appendix A.1.2. The final objective is:

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

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:

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

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 (Kostrikov et al., 2022) as prior work has demonstrated it works on real robots (Venkataraman et al., 2024; Zhang et al., 2023; 2024). See Figure 2(b) for an overview.

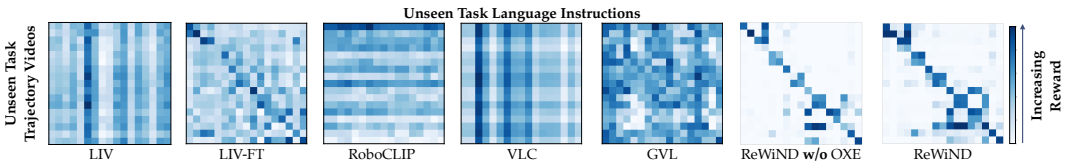


Figure 4: **Video-Language Reward Confusion Matrix.** For each unseen Metaworld task, we compute rewards for all combinations of demonstration videos and language descriptions. ReWiND produces the most diagonal-heavy confusion matrix, indicating strong alignment between unseen demos and instructions.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

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 (Alakuijala et al., 2025; Yang et al., 2024a), we assume access to a success signal during online RL. We use this signal to give r_{success} bonuses similar to in pre-training. [2] Our online rewards $\hat{r}^{\text{on}}$ are:

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

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

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: **LIV** (Ma et al., 2023) , we also fine-tune LIV on $\mathcal{D}_{\text{demos}}$ (**LIV-FT**); **RoboCLIP** (Sontakke et al., 2023); **Video-Language Critic (VLC)** (Alakuijala et al., 2025) We train its reward model on $\mathcal{D}_{\text{demos}}$; **Generative Value Learning (GVL)** (Ma et al., 2025).

We conduct our primary reward analysis using the simulated Metaworld benchmark (Yu et al., 2019). Smaller-scale real-world analyses, **strongly aligned** with simulation, are in Appendix D.3. 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` → `Door-Close`). We average metrics across 5 demos per task.

Generalization. Next, we evaluate how consistently rewards reflect progress over time in successful, unseen demonstrations. We report Pearson correlation (r) of each model's reward against time, and 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).

[2] Success bonuses are provided by a human supervisor. Our experiments assume a human supervisor because manual resets are required regardless.

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

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 an SAC (Haarnoja et al., 2018) 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 the rankings tells us how often the videos are correctly ranked. We report the ranking ρ in Table 1**(b)**. We also report the average *difference* between rewards for `success` with `near-success` and `near-success` with `failure` videos. Overall, likely due to *video rewinding*, ReWiND has a relative **74%** improvement in reward order and **58%** improvement in reward differences over the best baseline, LIV-FT. Additionally, we qualitatively demonstrate how these rankings translate into policy rollout rewards in Appendix Figure 7 by plotting per-frame reward curve 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 Meta-World simulation benchmark (Yu et al., 2019), 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 Meta-World 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, proprioception (x, y, z, gripper), and language conditioned. Each method uses the same policy pre-training and RL procedure as ReWiND , and is trained online for 100k timesteps.

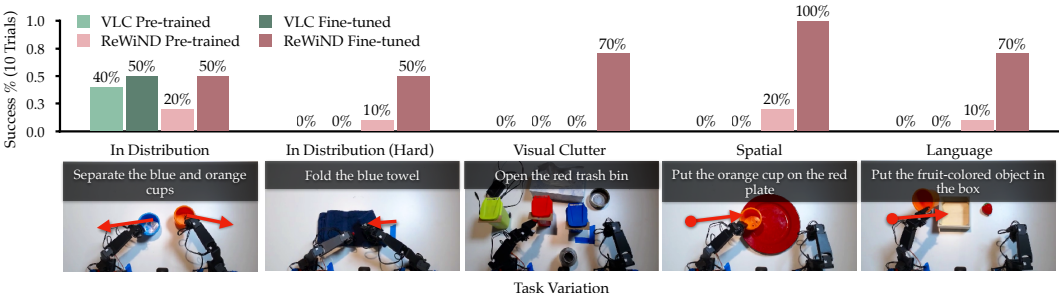


Figure 6: **Real-robot RL.** We present results on the Koch bimanual arms across in-distribution tasks and visual, spatial, and linguistic generalization tasks. Online RL with ReWiND improves a pre-trained policy by an absolute 56% across all five tasks.

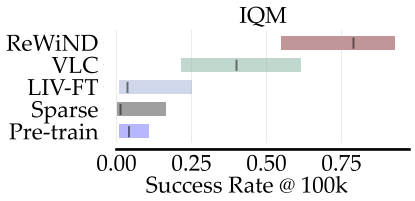


Figure 5: **Meta-World final performance.** We plot inter-quartile means (IQMs) of success rates after 100k environment steps on 8 unseen tasks.

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. In fact, Sparse reward fine-tuning, which relies purely on a sparse success bonus, performs worse than Pre-train after fine-tuning. While, 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 Meta-World. 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 13i.

Real-World Robot Learning. We conduct real-world tabletop manipulation experiments with a bimanual Koch v1.1 robot arm (Cadene et al., 2024). 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 (Anwar et al., 2024; Gao et al., 2025) 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 pretrained policy, and outperforms the best baseline from simulation, VLC.

Q3: Ablations and Analysis. ReWiND's ability to teach policies unseen tasks comes from achieving the three desiderata listed earlier. We demonstrate that each component of ReWiND contributes to at least one of each desiderata in Appendix D.4 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, in simulation and a bimanual robot.

References

Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004.

Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. Video-language critic: Transferable reward functions for language-conditioned robotics. In *Transactions on Machine Learning Research (TMLR)*, 2025.

Abrar Anwar, Rohan Gupta, and Jesse Thomason. Contrast sets for evaluating language-guided robot policies. In *Conference on Robot Learning (CoRL)*, 2024.

Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. Learning from physical human corrections, one feature at a time. In *International Conference on Human-Robot Interaction (HRI)*, 2018.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, 2023.

Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Hydra: Hybrid robot actions for imitation learning. In *Conference on Robot Learning (CoRL)*, 2023.

Erdem Biyik, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. Active preference-based gaussian process regression for reward learning. In *Robotics: Science and Systems (RSS)*, 2020.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2020.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. In *Robotics: Science and Systems (RSS)*, 2021.

Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 demonstration dataset. https://sites.google.com/view/berkeley-ur5/home, 2023.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. In *International Conference on Robotics and Automation (ICRA)*, 2024.

Yuchen Cui and Scott Niekum. Active reward learning from critiques. In *International Conference on Robotics and Automation (ICRA)*, 2018.

Yuchen Cui, Scott Niekum, Abhinav Gupta, Vikash Kumar, and Aravind Rajeswaran. Can foundation models perform zero-shot task specification for robot manipulation? In *Learning for*

Dynamics and Control Conference (L4DC), 2022.

Shivin Dass, Jullian Yapeter, Jesse Zhang, Jiahui Zhang, Karl Pertsch, Stefanos Nikolaidis, and Joseph J. Lim. Clvr jaco play dataset, 2023. URL https://github.com/clvrai/clvr_jaco_play_dataset.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima 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.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning (ICML)*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018a.

Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey 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 (eds.), *NeurIPS*, 2018b.

Jensen Gao, Suneel Belkhale, Sudeep Dasari, Ashwin Balakrishna, Dhruv Shah, and Dorsa Sadigh. A taxonomy for evaluating generalist robot policies. *arXiv preprint arXiv:2503.01238*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Joey Hejna and Dorsa Sadigh. Few-shot preference learning for human-in-the-loop rl. In *Conference on Robot Learning (CoRL)*, 2022.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.

Hengyuan Hu and Dorsa Sadigh. Language instructed reinforcement learning for human-ai coordination. In *International Conference on Machine Learning (ICML)*, 2023.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning (CoRL)*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2401.04088*, 2023.

Changyeon Kim, Minho Heo, Doohyun Lee, Jinwoo Shin, Honglak Lee, Joseph J Lim, and Kimin Lee. Subtask-aware visual reward learning from segmented demonstrations. In *International Conference on Learning Representations (ICLR)*, 2025.

Yigit Korkmaz and Erdem Bıyık. Mile: Model-based intervention learning. In *International Conference on Robotics and Automation (ICRA)*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning (ICML)*, 2021.

William Liang, Sam Wang, Hung-Ju Wang, Osbert Bastani, Dinesh Jayaraman, and Yecheng Jason Ma. Environment curriculum generation via large language models. In *Conference on Robot Learning (CoRL)*, 2024.

Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning (ICML)*, 2023.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations (ICLR)*, 2024a.

Yecheng Jason Ma, William Liang, Hungju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. Dreureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)*, 2024b.

Yecheng Jason Ma, Joey Hejna, Ayzaan Wahid, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Jonathan Tompson, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *International Conference on Learning Representations (ICLR)*, 2025.

Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *Conference on Robot Learning (CoRL)*, 2023.

Vivek Myers, Erdem Biyik, Nima Anari, and Dorsa Sadigh. Learning multimodal rewards from rankings. In *Conference on Robot Learning (CoRL)*, 2021.

Taewook Nam, Juyong Lee, Jesse Zhang, Sung Ju Hwang, Joseph J. Lim, and Karl Pertsch. Lift: Unsupervised reinforcement learning with foundation models as teachers. *arXiv preprint arXiv:2312.08958*, 2023.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.

Nghia Nguyen, Minh Nhat Vu, Tung D Ta, Baoru Huang, Thieu Vo, Ngan Le, and Anh Nguyen. Robotic-clip: Fine-tuning clip on action data for robotic applications. In *International Conference on Robotics and Automation (ICRA)*, 2025.

Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. In *International Conference on Machine Learning (ICML)*, 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. In *arXiv preprint arXiv:1910.00177*, 2019.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.

Sumedh Anand Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. Roboclip: One demonstration is enough to learn robot policies. In *NeurIPS*, 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Sreyas Venkataraman, Yufei Wang, Ziyu Wang, Zackory Erickson, and David Held. Real-world offline reinforcement learning from vision language model feedback. In *arXiv preprint arXiv:2411.05273*, 2024.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. In *International Conference on Machine Learning (ICML)*, 2024.

Nils Wilde, Erdem Biyik, Dorsa Sadigh, and Stephen L. Smith. Learning reward functions from scale feedback. In *Conference on Robot Learning (CoRL)*, 2021.

Jeffrey Wu, Seohong Park, Zipeng Lin, Jianlan Luo, and Sergey Levine. V-former: Offline RL with temporally-extended actions, 2024.

Ge Yan, Kris Wu, and Xiaolong Wang. Ucsd kitchens dataset, August 2023.

Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. Rank2reward: Learning shaped reward functions from passive video. In *International Conference on Robotics and Automation (ICRA)*, 2024a.

Jingyun Yang, Max Sobol Mark, Brandon Vu, Archit Sharma, Jeannette Bohg, and Chelsea 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)*, 2024b.

Zhaojing Yang, Miru Jun, Jeremy Tien, Stuart J. Russell, Anca Dragan, and Erdem Bıyık. Trajectory improvement and reward learning from comparative language feedback. In *Conference on Robot Learning (CoRL)*, 2024c.

Weirui Ye, Yunsheng Zhang, Haoyang Weng, Xianfan Gu, Shengjie Wang, Tong Zhang, Mengchen Wang, Pieter Abbeel, and Yang Gao. Reinforcement learning with foundation priors: Let embodied agent efficiently learn on its own. In *Conference on Robot Learning (CoRL)*, 2024.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, Brian Ichter, Ted Xiao, Peng Xu, Andy Zeng, Tingnan Zhang, Nicolas Heess, Dorsa Sadigh, Jie Tan, Yuval Tassa, and Fei Xia. Language to rewards for robotic skill synthesis. In *Conference on Robot Learning (CoRL)*, 2023.

Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. In *Conference on Robot Learning (CoRL)*, 2023.

Jesse Zhang, Karl Pertsch, Jiahui Zhang, and Joseph J. Lim. Sprint: Scalable policy pre-training via language instruction relabeling. In *International Conference on Robotics and Automation (ICRA)*, 2024.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. In *International Conference on Learning Representations (ICLR)*, 2025.

Xinghao Zhu, Ran Tian, Chenfeng Xu, Mingyu Ding, Wei Zhan, and Masayoshi Tomizuka. Fanuc manipulation: A dataset for learning-based manipulation with fanuc mate 200id robot. https://sites.google.com/berkeley.edu/fanuc-manipulation, 2023.

Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 2022.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

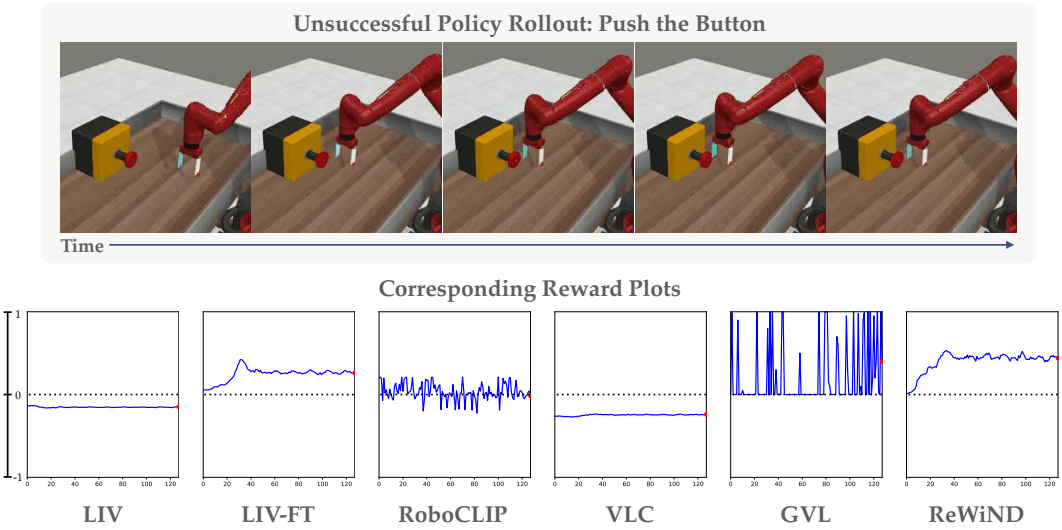


Figure 7: Unsuccessful policy rollout for the "Push the Button" task in Meta-World and its corresponding rewards below it. ReWiND predicts calibrated rewards that reflect better partial progress when the policy gets stuck near the button.

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 (Collaboration et al., 2024). The subset includes Bridge-V2 (Walke et al., 2023), BC-Z (Jang et al., 2021), Fractal Brohan et al. (2023), CLVR Jaco Play (Dass et al., 2023), Berkeley Autolab UR5 (Chen et al., 2023), Berkeley Fanuc Manipulation (Zhu et al., 2023), CMU Stretch (Bahl et al., 2023; Mendonca et al., 2023), Stanford Hydra (Belkhale et al., 2023), UCSD Kitchen (Yan et al., 2023), Austin BUDS (Zhu et al., 2022), and Austin Sirius (Liu et al., 2023). 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 8. 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

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

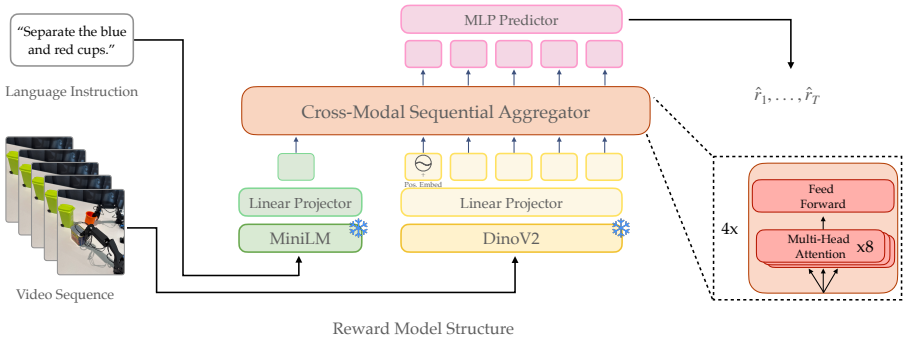


Figure 8: **ReWiND's Reward Model Architecture**. It's composed of frozen language and image input embeddings projected to a shared hidden dimension of 512. These embeddings are treated as input tokens to the cross-modal sequential aggregator transformer composed of 4 causally masked transformer layers 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.

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 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). In order to better policy execution videos that look *close to success*, 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.

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) (Oquab et al., 2024) along with ALL-MINILM-L12-V2 language embeddings of size 384 from the Sentence Transformers python package (Reimers & Gurevych, 2019). We also include proprioceptive information in both of our envrionments.

Offline RL. We use Implicit Q Learning (IQL) (Kostrikov et al., 2022) as prior work found it performant and easy to tune for robot manipulation with action-chunked policies (Zhang et al., 2023; 2024; Wu et al., 2024). 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 (Peng et al., 2019), 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) (Haarnoja et al., 2018) 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 (Zhou et al., 2025; Ball et al., 2023), 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 (Zhou et al., 2025; Uchendu et al., 2023)
- 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`. These tasks are visually similar to the training tasks, but the tasks are different. The 8 tasks used for downstream policy training are `Window-Close`, `Reach-Wall`, `Handle-Pull`, `Coffee-Button`, `Button-Press-Wall`, `Door-Lock`, `Handle-Press-Side`, `Sweep-into`.

Environment Details. We use Metaworld (Yu et al., 2019) with the default 3rd-person camera viewpoint, pictured in Figure 9, 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

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

Figure 10: **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.

space for $(\Delta x, \Delta y, \Delta z, \text{gripper})$. 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.

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 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.

Figure 9: Example camera viewpoint in Metaworld.

We pre-train the policy with IQL (Kostrikov et al., 2022) for 100K steps with learning rate 0.001, 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 Appendix 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 (Zhou et al., 2025). 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 (Cadene et al., 2024).[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

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

observations consist of RGB images from a Logitech C930e top camera and side camera (pictured in Figure 10). 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` (Jiang et al., 2023), 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.

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 timesteps corresponding to 2 seconds of actions. We also implement a Transformer-based critic. During rollouts, we then use temporal ensembling Zhao et al. (2023). Here, the current action is ensembles 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 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 (Haarnoja et al., 2018), 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. (2020). Thus the π and Q updates follow:

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

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

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.

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 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 Figure 11. 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.

D Additional Results

D.1 Additional Metaworld Reward Analysis

In Figure 12 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.

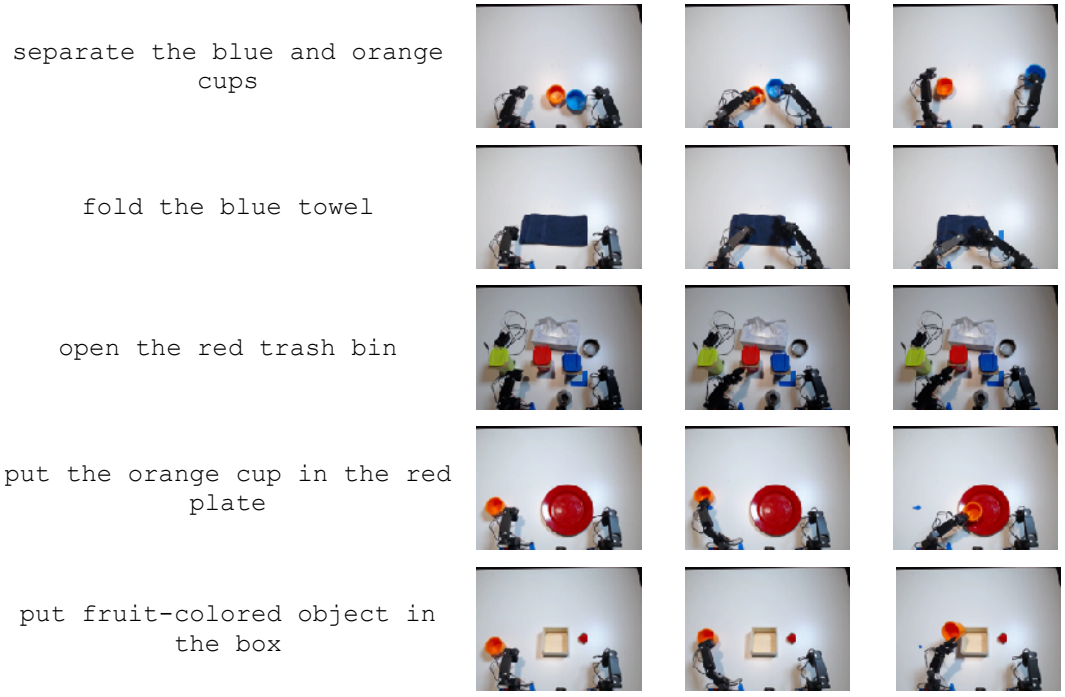


Figure 11: We present rollouts for the 5 tasks we use for online RL. The first two tasks are in-distribution to the policy, while the latter 3 tasks are out-of-distribution with respect to visual, spatial, or semantic generalization.

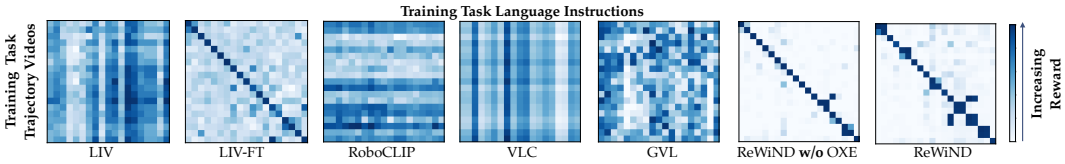


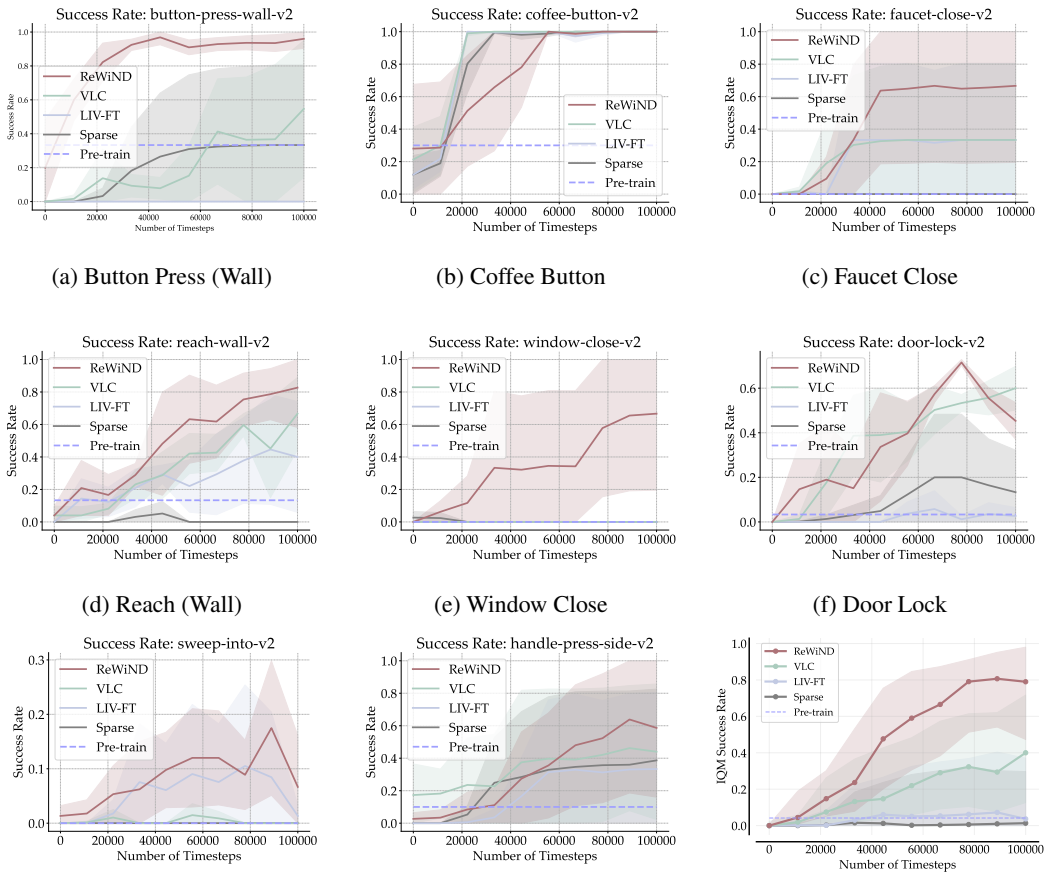
Figure 12: **Metaworld Reward Confusion Matrix on 20 Training Tasks.** For each training task, we compute rewards for all combinations of demonstration videos and language descriptions. ReWiND produces the most diagonal-heavy confusion matrix, indicating strong alignment between unseen demos and instructions.

D.2 MetaWorld Sample Efficiency Results

In this section, we analyze the sample efficiency of ReWiND against baselines in Metaworld. Figure 13 plots the learning curves for all downstream policy training tasks. Each panel corresponds to one specific task. And Figure 13i 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 14 and Figure 15,



(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 13: **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.

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.

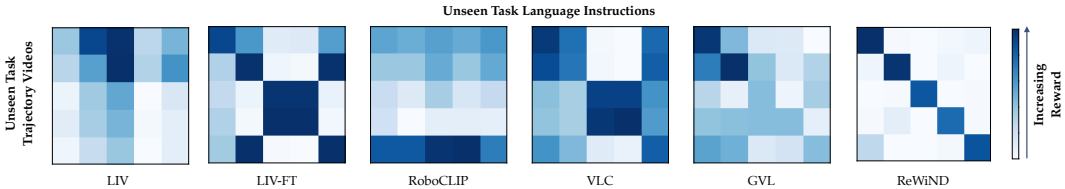


Figure 14: **Real-world Koch Reward Confusion Matrix on 5 Unseen Tasks.** For each unseen task, we compute rewards for all combinations of demonstration videos and language descriptions. ReWiND produces the most diagonal-heavy confusion matrix, indicating strong alignment between unseen demos and instructions.

D.4 Q3: Ablation Study and Additional Analysis

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

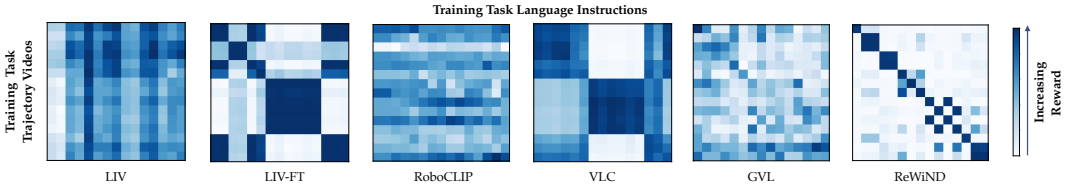


Figure 15: **Real-world Koch Reward Confusion Matrix on 15 Training Tasks.** For each training task, we compute rewards for all combinations of demonstration videos and language descriptions. LIV-FT, VLC and ReWiND are pretrained or fine-tuned with these training task while LIV , GVL and RoboCLIP are 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

in 4.1. We ablate: instruction generation and video rewinding (3.1.1); using OXE data; 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 (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 (3a) and fails to distinguish between failed, near-successful, and successful policy rollouts (3b). However, it retains strong input robustness due to the diversity of OXE data. Meanwhile, removing the Open-X subset (−**Open-X Subset**) harms unseen task reward alignment (3a) and input robustness (3c), highlighting the importance of OXE data for generalizing across varied language instructions.

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**

Augmentation. Eliminating video rewinding (−**Video Rewind**) degrades rollout ranking performance (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. Similarly, removing instruction generation (−**Instruction Generation**) reduces performance on language input robustness (3c), confirming that LLM-generated instructions enhance robustness to diverse inputs.

Architecture. Adding full positional embeddings (+**Full Pos. Embeds**) improves unseen demo alignment (3a) but worsens rollout ranking (3b), likely due to overfitting—where the model learns to predict increasing rewards regardless of input. To avoid overfitting, the main ReWiND model uses only first-frame positional embeddings (3.1.2).