
Self-Improving Embodied Foundation Models

Anonymous Author(s)

Affiliation

Address

email

Abstract

Foundation models trained on web-scale data have revolutionized robotics, but their application to low-level control remains largely limited to behavioral cloning. Drawing inspiration from the success of the reinforcement learning stage in fine-tuning large language models, we propose a two-stage post-training approach for robotics. The first stage, Supervised Fine-Tuning (SFT), fine-tunes pretrained foundation models using both: a) behavioral cloning, and b) steps-to-go prediction objectives. In the second stage, Self-Improvement, steps-to-go prediction enables the extraction of a well-shaped reward function and a robust success detector, enabling a fleet of robots to autonomously practice downstream tasks with minimal human supervision. Through extensive experiments on real-world and simulated robot embodiments, our novel post-training recipe unveils significant results on Embodied Foundation Models. First, we demonstrate that the combination of SFT and Self-Improvement is significantly more sample-efficient than scaling imitation data collection for supervised learning, and that it leads to policies with significantly higher success rates. Further ablations highlight that the combination of web-scale pretraining and Self-Improvement is the key to this sample-efficiency. Next, we demonstrate that our proposed combination uniquely unlocks a capability not possible by current methods: autonomously practicing and acquiring novel skills that generalize far beyond the behaviors observed in the imitation learning datasets used during training. These findings highlight the transformative potential of combining pretrained foundation models with online Self-Improvement to enable autonomous skill acquisition in robotics.

1 Introduction

Recent works have demonstrated that foundation models can be effectively fine-tuned to directly act as low-level robot policies [9, 40, 44, 38, 30, 20, 7], and that they inherit significant generalization and robustness capabilities due to the web-scale pretraining of the foundation models from which they were derived. Thus far the training regime for Embodied Foundation Models (EFMs) has been limited to behavioral cloning (i.e. supervised learning) [9, 40, 44, 38, 30, 20, 7]. In contrast, from the literature on large language models (LLM) we observe that after the initial pretraining, post-training for downstream tasks is typically divided into two stages: 1) Supervised Fine-Tuning (SFT), followed by 2) Reinforcement Learning (RL). RL-tuning of LLMs has been shown to markedly, and rapidly, improve downstream task performance beyond SFT [46, 39], and has become a critical stage in the training recipe of foundation models [1, 49, 19].

Despite the unique algorithmic and engineering challenges of investigating RL-tuning of foundation models in the context of real-world robotics, the aforementioned sample-efficiency and performance gains from the LLM literature strongly motivate its investigation. In this work we directly tackle these challenges and design a two-stage framework inspired by LLM post-training processes: In Stage 1 “Supervised Fine-Tuning” (SFT), given an imitation learning dataset we fine-tune EFMs using

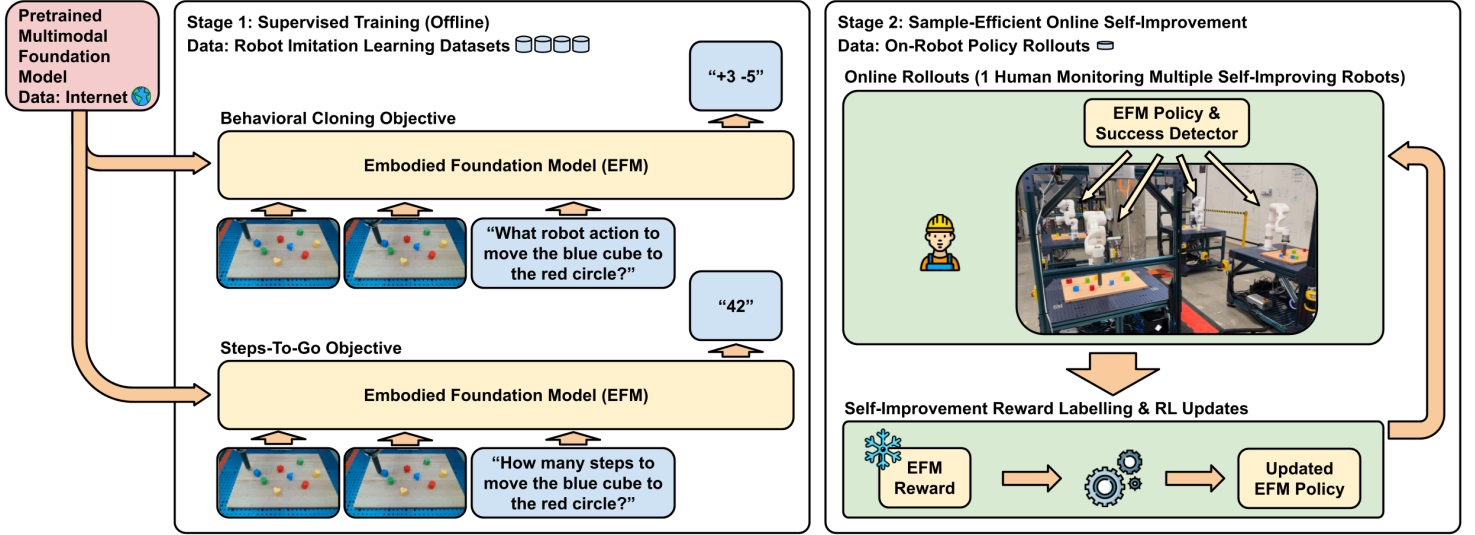


Figure 1: **Overview of our proposed two-stage fine-tuning approach. Stage 1 (Supervised Fine-Tuning):** Starting from a pretrained multimodal foundation model, using robot imitation learning datasets with fine-tune EFM with Behavioral Cloning and steps-to-go prediction objectives. **Stage 2 (Self-Improvement):** With minimal human supervision, online Reinforcement Learning using self-predicted rewards rapidly improves robot policies and enables the learning of novel out-of-distribution tasks.

two objectives: a) behavioral cloning, and b) predicting the number of “steps-to-go” to accomplish desired goals. In Stage 2 “Self-Improvement”, we leverage the model’s steps-to-go predictions to extract a well-shaped reward function as well as a robust success detector. These key components enable 1 human operator to monitor multiple robots as they autonomously practice downstream tasks. Critically, our data-driven reward design eliminates the need for ground-truth rewards, and leverages the robustness and generalization properties of the underlying foundation models.

Through extensive experiments on two robot embodiments, LanguageTable [33] and Aloha [56, 3], in the real-world and simulations, we demonstrate the surprising efficacy of our novel post-training framework. First, we demonstrate that not only does Self-Improvement robustly improve policy performance beyond behavioral cloning, but the combination of SFT and Self-Improvement is significantly more sample-efficient than scaling imitation data collection for supervised learning alone. As an example, on the LanguageTable domain [33], 10% additional robot time in the form of Self-Improvement increases policy success rates from 45% \rightarrow 75%. In contrast, increasing the amount of robot imitation data by 8x leads to a meager 45% \rightarrow 60% improvement. Further ablations highlight the key role of foundation model pretraining in enabling this sample-efficiency.

Excitingly, our novel combination of online Self-Improvement and web-scale pretraining also unlocks a unique capability not afforded by prior methods: enabling robots to autonomously practice and acquire new skills. In contrast to prior works that have demonstrated semantic generalization – such as executing the same pick-and-place motions in new contexts [9] – we show that this combination enables behavioral generalization that extends far beyond the imitation data used in Stage 1. Our work highlights the transformative potential of combining pretrained foundation models with online Self-Improvement to unlock autonomous skill acquisition in robotics. Our project website can be found at: <https://sites.google.com/view/mfa-self-improvement/home> In our supplementary zip file we include a version of this pdf that also contains the appendix.

2 Methodology

Our focus in this work is to investigate the efficacy of RL post-training for Embodied Foundation Models in the context of robotics. However, a critical challenge of reinforcement learning for robotics, and in particular for manipulation tasks, is the problem of reward engineering. Designing effective reward functions requires repeated trial-and-error iterations of training policies and patching reward definitions to mitigate unintended outcomes. Furthermore, even with a perfect definition,

measuring rewards in the real-world requires significant engineering effort. Thus, as we move towards a future where we train robots to accomplish increasingly broad sets of tasks, manual reward design becomes untenable for real-world robotics. We overcome this obstacle via learning data-driven reward functions that also inherit robustness and generalization properties from the web-scale pretraining of the underlying foundation models.

2.1 Stage 1: Supervised Fine-Tuning (SFT)

The first stage of our framework is the Supervised Fine-Tuning (SFT) stage. We assume access to a goal-conditioned imitation learning dataset \mathcal{D} consisting of a collection of episodes $\tau = \{(o_t, a_t, g_\tau)\}_{t=0}^T$, where o_t and a_t denote observation and action at timestep t respectively, and g_τ denotes the goal for episode τ ¹. We assume that all episodes in the dataset end in a state where the episode goal is accomplished. Given \mathcal{D} , we initialize the EFM using a pre-trained foundation model and perform supervised fine-tuning using the following objectives:

$$\begin{aligned}\mathcal{L}_{\text{BC}}(\text{EFM}) &= -\mathbb{E}_{(o_t, a_t, g_\tau) \sim \mathcal{D}} \left[\log p_{\text{action}}^{\text{EFM}}(a_t \mid o_t, g_\tau) \right] \\ \mathcal{L}_{\text{steps-to-go}}(\text{EFM}) &= -\mathbb{E}_{(o_t, a_t, g_\tau) \sim \mathcal{D}} \left[\log p_{\text{steps-to-go}}^{\text{EFM}}(\text{length}(\tau) - t \mid o_t, g_\tau) \right]\end{aligned}$$

\mathcal{L}_{BC} denotes a goal conditioned behavioral cloning loss, where we maximize the log-likelihood of a dataset action conditioned on the observation and the goal. The second objective, $\mathcal{L}_{\text{steps-to-go}}$ teaches the EFM to predict how many timesteps away the policy is from accomplishing the intended goal, given the current observations. This objective plays a critical role in enabling the second post-training stage, Self-Improvement².

2.2 Stage 2: Self-Improvement

In Stage 2, our goal is to fine-tune the EFM on downstream tasks using online RL in order to rapidly improve policy performance. As we will see later on in our experiments (Sections 4.3.1 and 4.3.2), downstream tasks may even be significantly different than those that appeared in the dataset \mathcal{D} used for Stage 1 training.

Reward Function Let $d(o, g) := \mathbb{E}_{p_{\text{steps-to-go}}^{\text{EFM}}(\text{steps-to-go} \mid o, g)} [\text{steps-to-go}]$ denote the expected value of “steps-to-go” in order to accomplish goal g given observation o , as predicted by the model. The reward function we use for online RL fine-tuning is defined as $r(o_t, a_t, o_{t+1}, g) := d(o_t, g) - d(o_{t+1}, g)$. Intuitively, this reward function predicts how much closer the robot got towards accomplishing goal g after taking action a_t . As the reward function is derived from $d(o, g)$, which is a function of the EFM itself, we refer to our RL fine-tuning process as “Self-Improvement”. The choice of using the expected value in defining $d(o, g)$ is for simplicity and alignment with the notion of a value function in RL (Section ??). We leave investigations of alternative definitions such as CVaR [4] for risk-aware policies, or distributional RL [5], to future work.

Success Detection It is important for robot episodes to terminate upon successfully accomplishing intended goals. Otherwise, a significant portion of collected data will include the robot resting in a successful state. In settings where we do not have a ground-truth success detector, as in real-world experiments, we use the following success indicator derived from the model, $\text{success}(o, g) := \mathbb{1}[d(o, g) \leq s]$, with s being a very small number of timesteps (we use $s = 3$ unless noted otherwise). We found this formulation of success detection to be very robust even in low data regimes, and significantly more reliable than explicitly including a success detection binary classification objective in Stage 1.

Self-Improvement With the above reward function and success detector in place, we can perform online RL fine-tuning of the EFM on desired downstream tasks. We take a frozen Stage 1 checkpoint for reward function computation and success detection, and initialize the Stage 2 policy from a Stage

¹We treat single-task datasets as goal-conditioned datasets where all episodes share the same goal.

²In Stage 1 we can include additional auxiliary supervised objectives as well. As an example, in our experiments with the LanguageTable domain, conditioned on the first and last image of an episode we ask the model to predict the instruction that was executed in that episode.

Algorithm 1: Self-Improvement

Input: Initialize the policy $p_{\text{action}}^{\text{EFM}}$ from a Stage 1 checkpoint. Initialize and freeze a Stage 1 checkpoint for reward computation and success detection.

while true do

- Initialize empty replay buffer;
- while replay buffer smaller than $N \times B$ do**
 - Sample instruction g ;
 - Execute current policy $p_{\text{action}}^{\text{EFM}}(a_t|o_t, g)$, terminate episode if {success(o_t, g) == 1 OR max episode length is reached OR operator manually terminates episode};
 - Compute Monte Carlo returns: $R_t \leftarrow \sum_{i=t}^T \gamma^{i-t} \cdot r(o_t, a_t, o_{t+1}, g)$;
 - Place (o_t, a_t, g, R_t) tuples in the replay buffer;
- end**
- Perform N policy updates using REINFORCE loss $\left[-c \cdot R_t \cdot \log p_{\text{action}}^{\text{EFM}}(a_t|o_t, g) \right]$;

end

111 1 checkpoint as well ³. Each iteration of our Self-Improvement loop proceeds as follows: Using
112 the current policy we collect a set of robot trajectories by sampling an instruction g , executing the
113 robot policy, and terminating the episode when either a) the success detector indicates success, b) a
114 pre-specified maximum episode length is reached, or c) a human operator manually terminates an
115 episode (e.g. if the robot station gets into a bad configuration). Subsequently, for each timestep in the
116 collected trajectories we compute the Monte Carlo returns $R_t \leftarrow \sum_{i=t}^T \gamma^{i-t} \cdot r(o_t, a_t, o_{t+1}, g)$ and
117 place elements (o_t, a_t, g, R_t) in a replay buffer. Once enough data has been collected, we perform N
118 policy updates using the REINFORCE loss $\left[-c \cdot R_t \cdot \log p_{\text{action}}^{\text{EFA}}(a_t|o_t, g) \right]$, sampling minibatches
119 from the replay buffer without replacement⁴. After N updates, remaining items in the replay buffer
120 are cleared out and the next iteration begins. Algorithm 1 above presents pseudocode of our Stage
121 2 Self-Improvement procedure. Although sample-efficiency is a key consideration of our work,
122 we chose to perform on-policy RL without data reuse. On-policy methods enjoy better training
123 stability [50], and using REINFORCE specifically eliminates the need for training value functions.
124 In Appendix F we discuss how our choice of reward function leads to a well-shaped objective that
125 reduces the need for baselines in the REINFORCE estimator. We leave the investigation of alternative
126 RL algorithms, including off-policy methods, to future work.

127 3 Intuition on Reward Function

128 **Visual Intuition** We can begin to build our intuition regarding the efficacy of steps-to-go prediction
129 by visualizing model predictions on domains of interest. Figure 2 visualizes an example trajectory
130 on the Aloha Single Insertion task, where the left arm must first pick up a blue socket, after which
131 the right arm must pick up the red peg and fully insert it into the socket. The caption in Figure 2
132 walks the reader through the level of intricate details that be can learned during Stage 1 training. We
133 encourage readers to visit our supplementary material website to view additional visualizations in the
134 form of videos, including on the LanguageTable domain.

135 Due to space limitations, we continue our discussion in Appendix F. First we discuss the mathe-
136 matical intuition behind why our algorithmic choices above lead to a well-shaped reward function.
137 Subsequently, we discuss the python notebook in our supplementary material, and demonstrate the
138 efficacy of Self-Improvement on a pedagogical 2D pointmass domain.

139 4 Experiments

140 In our experiments we seek to validate Self-Improvement as an effective post-training framework for
141 Embodied Foundation Models, and answer the following questions:

- 142 • **Q1:** Does Self-Improvement boost performance on downstream tasks beyond SFT?

³Note that these checkpoints are not necessarily identical. For a discussion on checkpoint selection process we refer the interested reader to Appendix B.

⁴We use $\gamma = 0.9$, $c = 5\text{e-}2$. Please refer to Appendix C for further discussion.

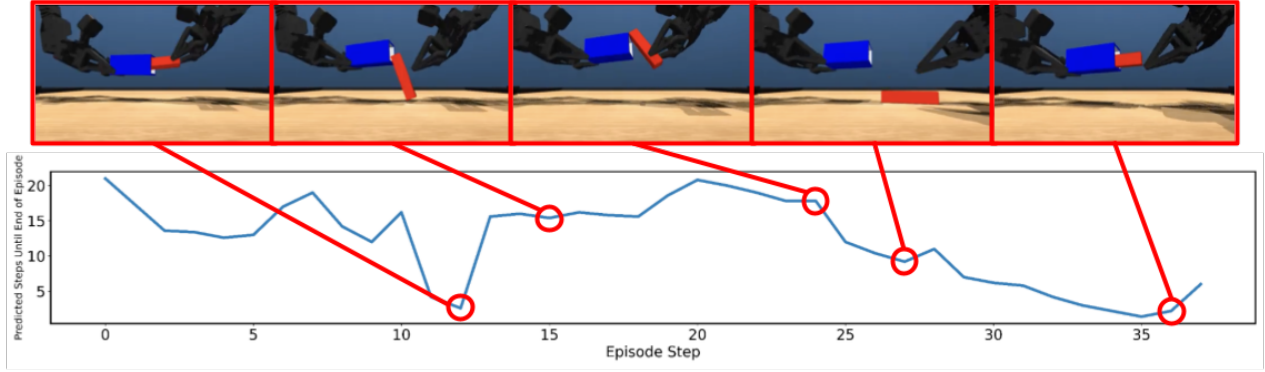


Figure 2: An example trajectory from the Aloha Single Insertion Task and a plot of model predictions for steps-to-go, $d(o, g)$. Key moments: 1) Model believes episode is about to complete successfully, 2) Policy accidentally drops the peg and $d(o, g)$ increases, 3) Policy regrasps the peg from a bad angle not suitable for insertion so $d(o, g)$ remains high, 4) Policy drops the peg, providing an opportunity to regrasp correctly which reduces $d(o, g)$, 5) Policy is pushing the peg inside and $d(o, g)$ marks that episode is about to succeed.

- **Q2:** Is the combination of supervised learning and Self-Improvement more sample-efficient than supervised learning alone?
- **Q3:** Is Self-Improvement, which depends on RL, reliable and reproducible enough to be employed in real-world robotics?
- **Q4:** What is the contribution of pretraining to our Self-Improvement procedure?
- **Q5:** Does web-scale foundation model pretraining enable Self-Improvement on tasks that generalize beyond what was seen in the imitation dataset?

We study these questions using the LanguageTable [33] and Aloha [56, 3] robot embodiments, with experiments in both simulation and the real-world. Throughout this work we use the PaLI 3 billion parameter vision-language model [12, 13] as our base pretrained foundation model. The inputs to our PaLI EFM are images alongside a text sequence representing relevant information such as instruction, auxiliary information (e.g. joint positions), and whether to predict actions or steps-to-go. The output is a sequence of tokens. To employ PaLI models as policies, we follow the RT-2 [9] policy parameterization and predict tokenized actions. Thus, our Stage 1 behavioral cloning policies are exactly equivalent RT-2 policies which will serve as key baselines. For full details regarding models, environments, tokenization, and training we refer readers to Appendix A.

4.1 Self-Improvement is Effective, Robust, and More Efficient Than SFT Alone

4.1.1 Simulated LanguageTable

The dataset we use to train Stage 1 policies for the simulated LanguageTable domain is the one provided by the original work [33]. This dataset consists of 181,020 human-generated trajectories, with 78,623 unique instructions describing the goals of the trajectories. We subsample this dataset to create 3 new datasets 10%, 20%, and 80% of the original size. For each dataset size, following Stage 1 training we perform Stage 2 fine-tuning with 3 seeds to validate the reliability of our Self-Improvement procedure. We perform Stage 2 fine-tuning on the Block2Block subset of tasks (e.g. "move the blue moon to the red pentagon")⁵ We stop Stage 2 training when policy success rates appear to plateau.

Results Figure 3 (first plot) presents our results on the simulated LanguageTable domain, where orange markers represent BC policy performance after Stage 1 (equivalent to RT-2), and blue markers represent policy performance after Stage 2 Self-Improvement. As can be observed, across all dataset sizes (10%, 20%, 80%), our proposed Self-Improvement procedure leads to very significant increase in success rates (minimum 1.5x performance boost), with incredible sample-efficiency in terms of number of episodes (less than 2% extra episodes collected in the Self-Improvement stage). Of particular note, Self-Improvement with 1% additional episodes on top of the 10% dataset size leads to policies that significantly outperform BC policies trained on 20% and 80% dataset sizes. In Appendix H we also show that Self-Improvement is robust and reproducible across random seeds.

⁵Block2Block instructions make up $\sim 47\%$ and $\sim 49\%$ of sim and real LanguageTable datasets (Appendix I).

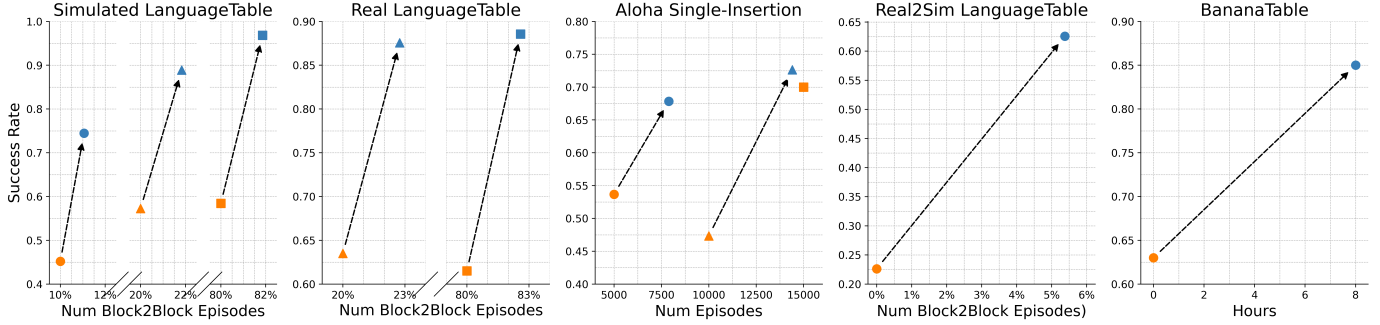


Figure 3: **Stage 2 Self-Improvement Results.** Orange: Stage 1 (equivalent to RT-2 Brohan et al. [9] baseline). Blue: Stage 2 Self-Improvement. Our results in simulated and real LanguageTable, and well as Aloha domain, demonstrate that our proposed two-stage approach achieves higher success rates significantly more sample-efficiently than supervised learning alone. Our Real2Sim LanguageTable and in particular BananaTable results demonstrate that the combination of Stage 2 and web-scale pre-training enables policies to acquire novel skills far outside the Stage 1 imitation learning dataset. Variations across random seeds are small, highlighting the robustness of our approach. Values above are averaged across 3 seeds (unaggregated results in Figure 9, Figure 10). While Stage 1 LanguageTable datasets contain varied tasks, for fairness, the x-axes in the plots above count number of Block2Block episodes (normalized by number of Block2Block episodes in full dataset).

4.1.2 Real-World LanguageTable

The significant sample-efficiency and robustness of our results above suggest that our Self-Improvement procedure may indeed be applicable for real-world robotics. We apply our proposed approach to the real-world LanguageTable domain, in two settings of using 20% and 80% of the imitation learning dataset [33]⁶. As in the simulated setting, we perform Stage 2 fine-tuning on the Block2Block subset of tasks. Given that instruction sampling, reward labeling, and success detection are entirely automated processes, during Self-Improvement *1 human operator is able to monitor our full fleet of LanguageTable robot stations*. The sole responsibility of the human operator is to reset a station if a block falls off the table, or if a station has not been shuffled for 5 minutes. Each experiments is run for approximately 20 hours. For details on the real-world LanguageTable experimentation protocol we refer the interested reader to Appendix G.

Results Figure 3 (second plot) presents our results. For both the 20% and 80% data settings, our Stage 2 Self-Improvement procedure improves policy success rate from ~62-63% up to ~87%-88%, with only ~3% additional Block2Block episodes collected. To put this into perspective, this means that with a total amount of experience equivalent to 20% (imitation dataset size) + 3% (Self-Improvement episodes), we obtain policies that far exceed BC (RT-2) policies trained with 80% imitation dataset size! Furthermore, as opposed to the 1-to-1 human-to-robot ratio needed for teleop imitation data collection, Self-Improvement requires only a fraction of the human effort due to the 1-to-many human-to-robot ratio enabled by our proposed approach.

4.1.3 Simulated Aloha Single Insertion Task

We also validate our proposed fine-tuning framework on a second robot embodiment, the bimanual Aloha manipulation platform [56, 3]. We design and collect data for a bimanual insertion task, where the left gripper must pick up a socket, and the right gripper must pick up a peg and insert it into the socket. Figure ?? presents a visualization of this task, with videos available on our supplementary materials website. Due to the much more complex observations, 70-dim action space, and much smaller imitation datasets, this presents a challenging setting for further validation of our proposed approach. For details on the environment and data collection process, we refer readers to Appendix E.3. We create 3 imitation dataset sizes of 5K, 10K, and 15K episodes. We apply our two-stage fine-tuning on 5K and 10K dataset sizes, and report results for supervised learning on the 15K dataset as well to better situate the numbers. The differences in methodology compared to LanguageTable domain are the following: 1) Stage 2 checkpoint initialization (Appendix B) 2) Adding a small

⁶We run the 80% data experiment once using 3 robot stations, and run the 20% data experiment twice, once with 3 and once with 4 robot stations.

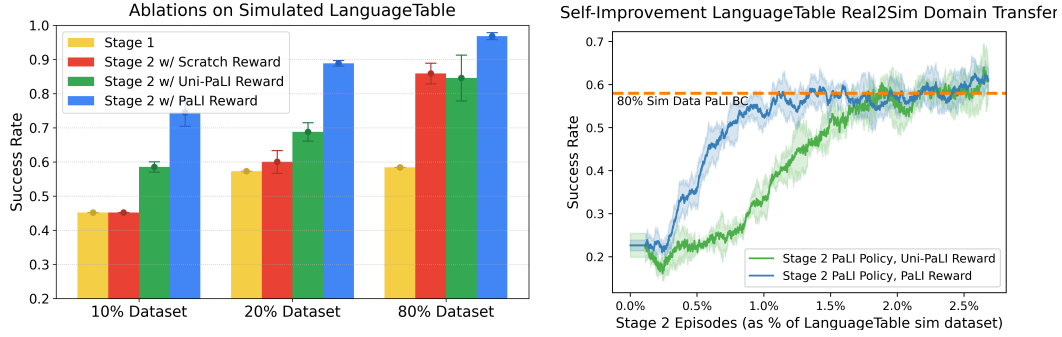


Figure 4: **Left** Our ablation results demonstrate the critical role of the web-scale pre-training of foundation models for enabling effective Stage 2 training, in particular in the small dataset size regime. **Right** LanguageTable Real2Sim domain transfer results.

positive constant to the reward function when the peg fully reaches the end of the socket, since we noticed this event is not observable from the cameras.

Results Figure 3 (middle) presents our results. As can be seen, for both dataset sizes Self-Improvement significantly improves policy success rates. As before, we also notice significant sample-efficiency gains where policies trained with 5K (imitation) + 2.5K (Self-Improvement) episodes outperform policies trained with 10K imitation episodes (i.e. RT-2), and rival the success rate of those trained with 15K imitation episodes.

- A1 Self-Improvement significantly improves policy performance beyond SFT.
- A2 The combination of supervised learning + Self-Improvement is more sample-efficient than supervised learning alone.
- A3 Self-Improvement is robust and effective for real-world robot learning.

217

4.2 Importance of Foundation Model Pretraining

It is critical to study to what extent the success of our proposed Self-Improvement procedure is afforded by the webscale pretraining of the PaLI [12, 13] vision-language foundation model we start from. To ablate the effect of the multimodal knowledge embedded into PaLI, we run our proposed two-stage fine-tuning process starting from alternative variations of the PaLI model:

- **Scratch:** PaLI architecture but with randomly initialized parameters.
- **Uni-PaLI:** PaLI parameters initialized from a vision model and language model, each pretrained unimodally without any joint multimodal vision-language fine-tuning (Appendix A.1).

We compare these variations using an identical setup as Section 4.1.1 on the Simulated LanguageTable domain. Despite our best efforts and very long training runs, we observed that Stage 1 BC policies derived from the Scratch and Uni-PaLI variations very significantly underperformed PaLI BC policies. Hence, we focus our ablations on the Self-Improvement stage, where we initialize policies from PaLI Stage 1 checkpoints, and use Scratch or Uni-PaLI checkpoints for reward computation.

Results Figure 4 (left) presents our results. There is a clear ordering in performance: PaLI reward models are best, followed by Uni-PaLI, and then Scratch. Scratch reward models lead to high variance results across random seeds, and struggle to provide any meaningful improvements in low-data (10% & 20%) regimes. While better than Scratch, Uni-PaLI reward models perform significantly worse than PaLI reward models across the board, with the gaps more pronounced at lower data settings. In fact, Self-Improvement with the PaLI reward model in the 20% regime leads to better policies than Self-Improvement with the Uni-PaLI reward model in the 80% regime! These results clearly demonstrate the immense value of multimodal pretraining for Self-Improvement.

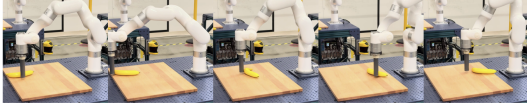
- A4: Multimodal pretraining leads to significantly better Self-Improved policies, and is a key enabler of sample-efficiency.

239

Before Stage 2 Fine-Tuning (Instruction: Move the banana to the left center of the board)



After Stage 2 Fine-Tuning (Instruction: Move the banana to the top center of the board)



LanguageTable Robot View



BananaTable Robot View



Figure 5: **Strong Generalization to BananaTable.** **Top** Before Stage 2 fine-tuning on the BananaTable domain, the policy struggles to effectively maneuver a banana across the table due to the difficult geometry. **Bottom Left** After Stage 2 fine-tuning policies are visibly more proficient at the BananaTable task (videos on our supplementary website). **Bottom Right** Prior to Stage 2 BananaTable fine-tuning, the policy and reward models have never seen the BananaTable task, creating a very challenging generalization problem.

4.3 Generalization

The combination of our proposed online Self-Improvement process and the use of pretrained multi-modal foundation models unlocks a unique capability: enabling policies to practice novel tasks that generalize beyond what was covered by the Stage 1 imitation learning datasets. In this section we present results for two increasingly difficult forms of generalization.

4.3.1 Domain Transfer Between Simulation and Real

Starting with a simpler form of generalization, in this section we investigate domain transfer between simulation and real. Sim2Real is an important class of approaches that can significantly reduce the amount of real-world experience needed to train performant robot policies, and has been successfully applied in many settings [42, 47, 2, 43, 29]. To make experimentation simpler, in this section we investigate the inverse problem, Real2Sim transfer, on the LanguageTable domain. We train Stage 1 models using 80% of the *real-world* LanguageTable dataset, and perform Stage 2 Self-Improvement in the *simulated* LanguageTable environment. Similar to our ablation in Section 4.2, we also train Stage 2 models using the Uni-PaLI reward model variant to highlight the role of foundation model pretraining in enabling domain transfer.

Results Figure 4 (right) presents our results. With 3% extra episodes in the target domain (simulated LanguageTable), our Self-Improvement procedure improves policy performance from $\sim 22\%$ to $\sim 59\%$. This performance is equivalent to BC policies trained with 80% of target domain’s imitation dataset. Additionally, Figure 4 (right) demonstrates that the Uni-PaLI reward model leads to a significantly slower Self-Improvement procedure, highlighting the key advantage of pretraining. Given our strong real-world LanguageTable results in section 4.1.2, we expect our Real2Sim results to be strongly indicative of Sim2Real transfer as well.

4.3.2 Strong Generalization to Learning Novel Skills

Moving towards a stronger form of generalization, we investigate whether Self-Improvement with pretrained foundation models enables policies to practice and acquire novel behavioral skills beyond those observed in the imitation learning datasets used in the SFT stage. Starting from a policy and reward model trained with the real-world LanguageTable dataset⁷, we perform Self-Improvement on a new task we dub “BananaTable” (Figure 5). In this task we replace the LanguageTable blocks with a single prosthetic banana and request policies to push the banana to various locations on the board (e.g. “move the banana to the left center of the board”). The LanguageTable dataset contains no bananas, nor any episodes where the blocks are not on the table. Thus we are solely relying on the generalization abilities of the underlying PaLI foundation model from which the policy, reward function, and success detector are derived.

In contrast to prior works that have demonstrated semantic generalization abilities of robot foundation models (e.g. executing the same pick and place motions in novel contexts in RT-2 [9]), transfer to the BananaTable task requires behavioral generalization, necessitating policies to learn new skills. As an example, due to its elongated geometry, inaccurate pushing of the banana results in it rotating around itself instead of moving in the intended direction (Figure 5, top).

⁷We initialize the BananaTable Self-Improvement procedure using the reward model and Self-Improved policy from Section 4.1.2 in the 80% data setting.

Results Within ~ 8 hours of Self-Improvement using 2 robot stations, the policy success rate improves from $\sim 63\%$ to $\sim 85\%$. Beyond the quantitative results, videos on our supplementary website (as well as Figure 5) demonstrate that the policy becomes visibly more proficient at accomplishing the BananaTable tasks, as it picks up on effective strategies for moving the banana around the table.

A5: The combination of our proposed online Self-Improvement procedure and webscale pretrained foundation models enables policies to rapidly acquire new skills that generalize beyond the tasks covered by the imitation learning datasets they are provided.

5 Related Works

Embodied Foundation Models A number of prior works have leveraged pretrained multimodal foundation models as robot policies, by incorporating action prediction heads. Brohan et al. [9] discretize continuous robot actions and map them onto language model token spaces. Other examples architectures include diffusion models [38, 53] (building on diffusion policies [14, 27]), flow matching [7, 28] (building on [32]), and regression [30]. Critically, these prior works only leverage supervised learning, and the delineation between pretraining and post-training is the task and data mixture used. To the best of our knowledge, our work is the first to move past supervised learning for training robot foundation models. The key contribution of our work is to present a general-purpose, reward-engineering-free, online Self-Improvement procedure that not only leads to rapid policy improvement, but enables acquisition of novel behaviors outside of what the models have seen in their training data. This form of behavioral generalization is only achievable through online post-training.

Improving Robot Policies Without Ground-Truth Rewards A key obstacle of general-purpose Self-Improvement through RL is that commonly we do not have access to ground-truth reward functions and success metrics, either due to the challenge of designing one (reward engineering), or difficulty in measuring them in the real world (reward instrumentation). A line of prior work, dubbed "Code-As-Rewards", leverages LLMs to write code for reward engineering [35, 55, 52]. Such approaches have a number of downsides that make them impractical for general-purpose robot learning in the real world (discussion in Appendix L). Aside from Code-As-Rewards, there exists a rich literature on obtaining data-driven reward functions. An important class of works [6, 34, 45, 11] learn latent observation representations on top of which rewards and value functions can be defined (e.g. via L2 distance in latent space to target goals). Other approaches include creative relabeling techniques [31, 10, 8] and RL objectives [21, 10]. In comparison, the key advantage of our proposed approach is the straightforward integration with webscale pretrained foundation models. The closest related works to ours are those based on learning distances in timesteps [25, 26]. Aside from key differences in settings, our focus in this work is on post-training Embodied Foundation Models and generalization, demonstrating that steps-to-go policy improvement is a viable path towards general-purpose Self-Improving robot policies. Additionally, we present steps-to-go thresholding as a novel path towards obtaining robust open-ended success detectors, which have typically been trained via binary classification [18]. Lastly, Yang et al. [54] extend our approach to real-world simulators built on generative video foundation models, and Ma et al. [36] present an extension to an in-context learning formulation. For an extended discussion on related works, please refer to Appendix L.

6 Future Work and Limitations

Our work has clearly demonstrated the immense potential of the novel combination of pretrained multimodal foundation models with online Self-Improvement. This combination not only enables very efficient policy improvement on real-world robots, but also unlocks strong generalization capabilities and autonomous acquisition of new skills. There still exist, however, many important avenues for future work. **Algorithmic:** In this work we leveraged on-policy REINFORCE for simplicity which does not reuse any collected data during Self-Improvement. Off-policy methods have the potential to even more substantially boost sample-efficiency. **Skill-Chaining:** How can Self-Improvement and our proposed success detector be extended to enable skill-chaining and solving long-horizon tasks? **Robustness:** Since imitation datasets typically contain expert behavior, they do not contain many forms of failure cases and recovery behaviors. How can our post-training approach be extended to ensure the reliability of reward models and success detectors on out-of-distribution failure cases? We hope that the strong results presented in this work motivate investigation of these fruitful research avenues. For an extended discussion on future works and limitations, please refer to Appendix ??.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik’s cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
- [3] Jorge Aldaco, Travis Armstrong, Robert Baruch, Jeff Bingham, Sanky Chan, Kenneth Draper, Debidatta Dwibedi, Chelsea Finn, Pete Florence, Spencer Goodrich, et al. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation. *arXiv preprint arXiv:2405.02292*, 2024.
- [4] Gordon J Alexander and Alexandre M Baptista. A comparison of var and cvar constraints on portfolio selection with the mean-variance model. *Management science*, 50(9):1261–1273, 2004.
- [5] Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023. <http://www.distributional-rl.org>.
- [6] Chethan Bhateja, Derek Guo, Dibya Ghosh, Anikait Singh, Manan Tomar, Quan Vuong, Yevgen Chebotar, Sergey Levine, and Aviral Kumar. Robotic offline rl from internet videos via value-function pre-training. *arXiv preprint arXiv:2309.13041*, 2023.
- [7] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [8] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.
- [9] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.
- [10] Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021.
- [11] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.
- [12] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.
- [13] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023.
- [14] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.
- [15] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

- [16] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, 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 Buechler, 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, 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, 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, Kanishk 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, Minh 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 Bajjal, 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, Shuran Song, Sichun Xu, Siddhant Halder, 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, 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. <https://arxiv.org/abs/2310.08864>, 2023.
- [17] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.
- [18] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.
- [19] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd

- of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [20] Zane Durante, Bidipta Sarkar, Ran Gong, Rohan Taori, Yusuke Noda, Paul Tang, Ehsan Adeli, Shrinidhi Kowshika Lakshmikanth, Kevin Schulman, Arnold Milstein, et al. An interactive agent foundation model. *arXiv preprint arXiv:2402.05929*, 2024.
- [21] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.
- [22] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.
- [23] Raj Ghugare, Matthieu Geist, Glen Berseth, and Benjamin Eysenbach. Closing the gap between td learning and supervised learning—a generalisation point of view. *arXiv preprint arXiv:2401.11237*, 2024.
- [24] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [25] Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.
- [26] Joey Hejna, Jensen Gao, and Dorsa Sadigh. Distance weighted supervised learning for offline interaction data. In *International Conference on Machine Learning*, pages 12882–12906. PMLR, 2023.
- [27] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [28] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.
- [29] Satoshi Kataoka, Youngseog Chung, Seyed Kamyar Seyed Ghasemipour, Pannag Sanketi, Shixiang Shane Gu, and Igor Mordatch. Bi-manual block assembly via sim-to-real reinforcement learning. *arXiv preprint arXiv:2303.14870*, 2023.
- [30] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- [31] Aviral Kumar, Anikait Singh, Frederik Ebert, Mitsuhiro Nakamoto, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. *arXiv preprint arXiv:2210.05178*, 2022.
- [32] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- [33] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.
- [34] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.
- [35] 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. *arXiv preprint arXiv:2310.12931*, 2023.

- [36] Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, et al. Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations*, 2024.
- [37] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.
- [38] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.
- [39] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [40] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.
- [41] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018.
- [42] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.
- [43] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.
- [44] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.
- [45] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.
- [46] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.
- [47] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.
- [48] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.
- [49] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.
- [50] Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

- 530 [51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,
531 Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information*
532 *processing systems*, 30, 2017.
- 533 [52] David Venuto, Sami Nur Islam, Martin Klissarov, Doina Precup, Sherry Yang, and Ankit
534 Anand. Code as reward: Empowering reinforcement learning with vlms. *arXiv preprint*
535 *arXiv:2402.04764*, 2024.
- 536 [53] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng,
537 Chaomin Shen, Yaxin Peng, et al. Tinyvla: Towards fast, data-efficient vision-language-action
538 models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.
- 539 [54] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and
540 Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*,
541 2023.
- 542 [55] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez
543 Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language
544 to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.
- 545 [56] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual
546 manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.
- 547 [57] Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea
548 Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint*
549 *arXiv:2410.13126*, 2024.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [NA].
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes]" is generally preferable to "[No]", it is perfectly acceptable to answer "[No]" provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No]" or "[NA]" is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS Paper Checklist”,**
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers.**

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes]

Justification: We provide extensive experiments for each claim made, and answer all of the key questions laid out in Section 4.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Please refer to Section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA] .

Justification: Our work does not provide any major theoretical results. Our discussion on Mathematical Intuition discusses simple algebraic manipulations.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We present our method in full detail, with further experimental details in the Appendix. We also include a Colab notebook as a pedagogical implementation of our algorithm.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We are unable to open-source our code. However, we include a Colab notebook as a pedagogical implementation of our algorithm. The datasets we use for imitation learning are existing open-sourced datasets.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: We provide extensive details in the main text as well as Appendix sections.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: All experiments in the simulated domains were run with a minimum of 3 random seeds. Aside from Figure 3, all plots (including the ones in the Appendix) contain either raw data or error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We provide a discussion in Appendix A.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines?>

Answer: [Yes]

Justification: We have read the NeurIPS Code of Ethics and conform to the code.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We include a discussion in Appendix N.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not release any models or datasets.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: We provide citations for data, model, and simulation sources.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: In our supplementary materials we include a Colab notebook with a pedagogical implementation of our proposed method. We discuss the setup in our manuscript.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We do not have crowdsourcing or experiments with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not have experiments requiring such approvals.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Pretrained multimodal foundation models are a key component of our work and we provide detailed discussions in our manuscript.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Implementation Details

A.1 Background

PaLI Vision-Language Foundation Model While our investigations in this work are independent of the choice of underlying foundation model used, throughout this work we use the 3 billion parameter PaLI-3B [12, 13] vision-language model as the base pretrained foundation model that we fine-tune for robotics tasks. A PaLI model receives as input one or more images alongside text, and provides text as output. At a high level, the PaLI architecture is comprised of two components: 1) a Vision Transformer (ViT) [41], and 2) an encoder-decoder Transformer [51]. Input images are processed by the ViT into a sequence of “visual tokens”. The sequence of visual tokens is concatenated with the tokenized text input and fed into the Transformer which outputs text tokens. The weights of the PaLI model are initialized from a Transformer language model and ViT vision model that are pretrained separately in a unimodal fashion. Following this initialization, the model is fine-tuned with a variety of vision-language training objectives to obtain a multimodal foundation model. For further details regarding PaLI model, we refer the interested reader to [12, 13]. We emphasize that our framework is independent of the choice of underlying multimodal foundation model used.

RT-2 Brohan et al. [9] introduce a model family, dubbed RT-2, that enables vision-language foundation models (VLMs) to directly produce low-level robot actions for closed-loop control. The two VLMs considered in that work are PaLI [12, 13] and PaLM-E [17], both of which take images alongside text as input, and provide output in the form of text tokens. To enable these VLMs to act as robot policies, continuous robot actions are discretized and mapped onto the text token space of the VLMs. Given image and text inputs, the VLMs are fine-tuned via behavioral cloning (i.e. supervised learning) to predict the tokenized robot actions. While the methods we present in this work are independent of the choice of underlying model, throughout this work our robot policy architectures are equivalent to RT-2 using the PaLI VLM.

As we use the RT-2 policy representation, we also decided to model steps-to-go predictions by discretizing range of possible number of steps, and mapping them onto the PaLI VLM token space.

A.2 Environments, Tasks, and Tokenization

For details about the environments and tasks used in this work, please refer to Appendix E.

For details about the tokenization approach for domain and task, please refer to Appendix D.

A.3 Training Details

Stage 1 (SFT) During the supervised training stage we uniformly distribute each training batch amongst the objectives used in Stage 1. For all domains considered in this work this includes the a) behavioral cloning, and b) steps-to-go prediction objectives. In the real-world and simulated LanguageTable domain experiments, we have an additional objective c) predicting the instruction given the first and last frame of an episode. We did not ablate the value of incorporating this objective during training. We used batch size 128 during this stage, used the AdamW optimizer, and trained the entire PaLI model (i.e. kept no component frozen).

Stage 2 (Self-Improvement) During this stage we used batch size 64 to require less real-world rollouts for a given number of desired training steps. We kept the ViT portion of the model frozen, intuitively believing that the model has already learned visual features for the task, and that freezing the ViT may potentially help with model stability. We did not ablate this decision. We used the same AdamW optimizer as in Stage 1.

A.4 Compute Resources

Stage 1 (SFT) training was done using one of the following configurations, interchangeably:

- 64 TPUv4 (2x4x4)
- 128 TPUv3

953 For Stage 2 (Self-Improvement) we used:

- 954 • Half of SFT stage resources for the learner job (since we used half batch size)
- 955 • 4 TPUv4 (2x2x1) or for the reward model
- 956 • 4 TPUv4 (2x2x1) for the success detector

957 B Checkpoint Selection for Stage 2 Initialization

958 The frozen checkpoint used for reward computation and success detection is not necessarily identical
959 to the checkpoint used for policy initialization since the best performance for steps-to-go and BC
960 objectives can happen at different points over the course of Stage 1 training. For the most part
961 throughout this work we took the checkpoints at the best validation loss for the corresponding
962 objective. An exception to this was how we chose the policy initialization checkpoint in the Aloha
963 domain. We observed that at the best validation loss ($\sim 5\text{K}$ - 10K steps into training) the BC policy did
964 not have a reasonable success rate. By allowing the model to continue training for much longer and
965 overfitting ($\sim 100\text{K}$ - 300K steps into training) the policy success rate improved significantly.

966 C REINFORCE Multiplicative Constant

967 We perform policy updates using the REINFORCE loss,

$$-c \cdot R_t \cdot \log p_{\text{action}}^{\text{EFA}}(a_t | o_t, g)$$

968 In simulation experiments we found that using a small positive multiplicative factor c in the RE-
969 INFORCE loss plays a significant role in ensuring the model trains stably. Note that this is not
970 equivalent to scaling the learning rate due to interactions with regularizers such as weight decay.
971 Throughout this work we use $c = 5\text{e-}2$. We did not perform any careful tuning of c , and chose its
972 approximate scale using the following intuition: Let γ denote the discount factor being used. If we
973 assume the policy gets N steps closer to the goal after every timestep, we have,

$$R_t = \sum_{i=t}^T \gamma^{i-t} \cdot N \simeq \frac{N}{1-\gamma}$$

974 Intuitively, we would like to make the weights on the log probability fall approximately into the range
975 -1 to 1 (i.e. $-c \cdot R_t \in [-1, 1]$). Thus we have $c = \frac{1-\gamma}{N}$. We use $\gamma = 0.9$ throughout this work,
976 and hypothetically assume that the range of N is approximately $[-2, 2]$ (e.g. we believe the Stage 2
977 policy can become twice as efficient as the BC policy)⁸. This results in our choice of $c = 5\text{e-}2$.

978 D Tokenization

979 D.1 Real/Sim LanguageTable & BananaTable

980 We use the same tokenization approach for the real-world and simulated LanguageTable, as well as
981 the BananaTable domains.

982 **Action Tokenization** As noted in Appendix E.1, the above domains have a 2D continuous action
983 space. We represent LanguageTable actions via a sequence of 4 tokens: a) token for $+/-$, b) token
984 representing a number in the range $[0, 10]$, c) token for $+/-$, d) token representing a number in the
985 range $[0, 10]$. The continuous 2D actions are binned to fall into this representation.

986 **Steps-to-go Tokenization** We represent timesteps until end of episode using one token, which
987 represents a number between 0-50.

⁸Note that this is an approximate intuitive guess and does not need to be precise. A poor guess simply affects the scale of the loss and does not constrain policy learning in any manner.



Figure 6: **LanguageTable Environments.** **Left** The four LanguageTable robot stations used for our real-world experiments. **Right, Top** Camera view of the real-world LanguageTable robot station. **Right, Bottom** Camera view of the simulated LanguageTable robot station.

D.2 Aloha

Action Tokenization As discussed in E.3 our action space is 5×14 dimensions. We represent each dimension with 1 token, meaning the model outputs 70 tokens. Each token represents a number from 0-255. The continuous Aloha actions are discretized and binned into these 256 bins.

Steps-to-go Tokenization We represent timesteps until end of episode using one token, which represents a number between 0-300. Since we use an action-chunk of 5, this means that we can support episodes that are up to 1500 timesteps long.

Joints Tokenization In the Aloha domain, as input we also provide the model with the current joint positions, i.e. we append 14 tokens to the input text instructions, where each token represents a number from 0-255. The continuous Aloha joints are discretized and binned into these 256 bins, in an identical manner as the actions.

E Environments and Tasks

E.1 LanguageTable

Figure 6 shows the real-world and simulated LanguageTable environments used in this work. The LanguageTable domain [33] has a 2D action space representing delta movement in the x-y plane. The dataset we used in Stage 1 (SFT) are the ones provided by the original work [33] introducing this domain. This dataset consists of 181,020 human-generated trajectories, with 78,623 unique instructions describing the goals of the trajectories. The tasks we perform Stage 2 (Self-Improvement) on are the Block2Block subset of tasks which contain instructions of the form ‘move the blue cube to the green star’. As noted in Appendix I, for the simulated and real dataset respectively, 47% and 49% of the instructions fall in the Block2Block grouping. The two images given to PaLI represent the current and previous frame as viewed by the LanguageTable robot camera.

E.2 BananaTable

In the BananaTable task we remove all blocks from the LanguageTable stations and replace them with a single banana. The instructions for the BananaTable task have the form, ‘X the banana to the Y of the table.’, where X is a set of verbs synonymous with pushing, and Y is one of left, top left, top center, top right, right, bottom right, bottom, bottom left, center.

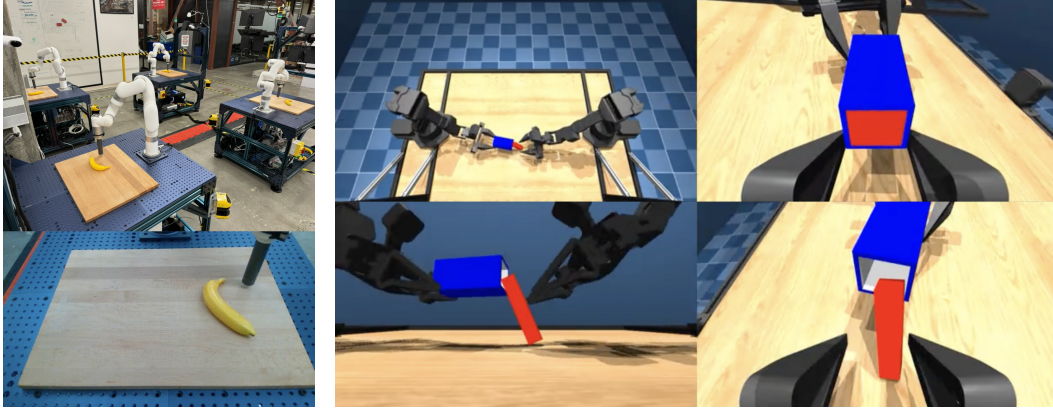


Figure 7: **Left, Top** BananaTable robot stations used in our experiments. **Left, Bottom** Camera view of the BananaTable robot station. **Right** The four camera views in the simulated Aloha Single-Insertion task.

1016 E.3 Aloha

1017 The Aloha domain [56] is 14 degree of freedom joint-space controlled robot. As opposed to the
 1018 default of predicting 50Hz actions, we predict 10Hz actions. A common design choice in the Aloha
 1019 domain is to train policies to predict N actions into the future. This is commonly referred to as
 1020 action-chunking [56], or action horizon [14]. We use $N = 5$ which results in an action space that is
 1021 70-dimensional (14×5). In the Aloha domain, as input we also provide the model with the current
 1022 joint positions, i.e. we append 14 tokens to the input text instructions, where each token represents a
 1023 number from 0-255.

1024 The Aloha environment has 4 cameras (Figure 7, right). To turn them into two images to pass to
 1025 our PaLI models, we stack two images into one image with a black buffer in between. We stack the
 1026 top and table view images into one image, and stack the left and right wrist view into one image as
 1027 well. We add a small black band between the stacked views inside each image in hopes of better
 1028 delineating them. Since we pass 224×224 images to PaLI, this means that each Aloha camera view
 1029 appears with an effective resolution of about 100×100 . This is significantly less resolution than the
 1030 typical Aloha resolution of 480×640 [57].

1031 We designed and collected data for a bimanual insertion task, where the left gripper must pick up a
 1032 socket, and the right gripper must pick up a peg and insert that peg into the socket. We collected 800
 1033 demonstrations using a VR headset to view the Mujoco simulation, and using the real-world Aloha
 1034 leader robots to control the virtual robots. We then trained a small diffusion policy [14] on the 800
 1035 demonstrations and used the model to generate 3 datasets of size, 5K, 10K, and 15K. Note that these
 1036 datasets only contain successful rollouts, and max episode lengths was chosen generously to allow
 1037 for recovery from mistakes.

1038 Critical to successful PaLI policies was to employ semi-global action representations as in [15], as
 1039 well as training Stage 1 (SFT) far beyond the point at which the best validation loss was obtained for
 1040 the behavioral cloning loss (Appendix B).

1041 F Intuition On Reward Function (Continued)

1042 **Mathematical Intuition** Let μ denote the policy corresponding to the imitation learning dataset \mathcal{D}
 1043 (e.g. if the dataset was collected via teleoperation, μ would represent the “human policy”). Consider
 1044 the reward function $-\mathbb{1}[o_t \text{ satisfies } g]$ that is 0 when the goal is satisfied, and -1 elsewhere. Let V^μ
 1045 denote the undiscounted value function of policy μ for this reward function. We have,

$$V^\mu(o_t, g) = \mathbb{E}_\mu \left[\sum_{i=t}^T -\mathbb{1}[o_i \text{ satisfies } g] \right] = \mathbb{E}_\mu \left[-1 \cdot \text{steps-to-go} \right] =: -d(o_t, g)$$

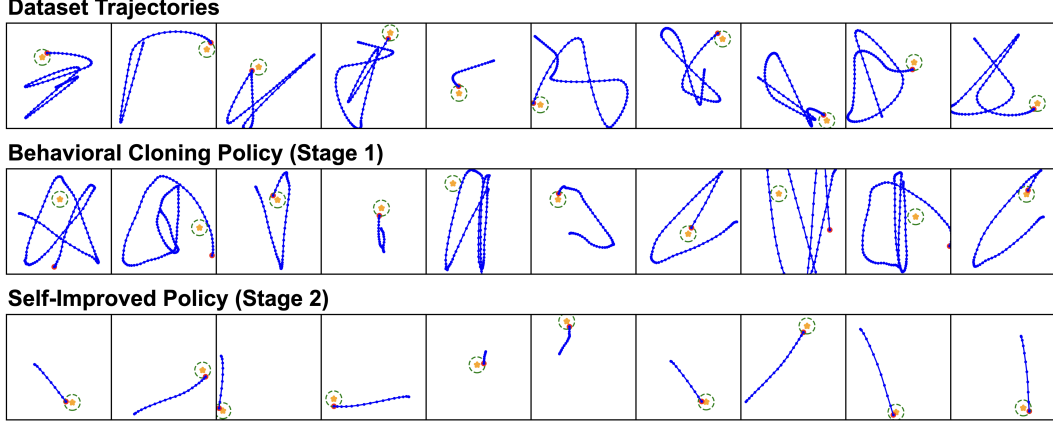


Figure 8: **Pointmass Navigation Domain.** Sample trajectories from the imitation learning dataset, as well as BC (Stage 1) and Self-Improved (Stage 2) policies.

1046 Substituting V^μ we obtain,

$$r(o_t, a_t, o_{t+1}, g) = V^\mu(o_{t+1}, g) - V^\mu(o_t, g) = \underbrace{(1 - \gamma) \cdot V^\mu(o_{t+1}, g)}_{\text{core reward}} + \underbrace{\left[\gamma \cdot V^\mu(o_{t+1}, g) - V^\mu(o_t, g) \right]}_{\text{reward shaping}} \quad (1)$$

1047 where γ is the discount factor used in the Stage 2 RL updates. We see that $r(o_t, a_t, o_{t+1}, g)$ is
 1048 implicitly a shaped reward function [37] that provides higher rewards in states where μ knows how
 1049 to perform well (i.e. core reward $(1 - \gamma) \cdot V^\mu(o_{t+1}, g)$ is high). Thus, *Self-Improvement leads to*
 1050 *policies that achieve intended goals more efficiently than the dataset policy μ , while being implicitly*
 1051 *regularized to stay close to regions of the state space where μ is proficient!*

1052 Using Equation 1 to simplify the telescoping sum in the Monte Carlo returns we have,

$$R_t = \sum_{i=t}^T \gamma^{i-t} \cdot r(o_i, a_i, o_{i+1}, g) = \left[(1 - \gamma) \cdot \sum_{i=t}^T \gamma^{i-t} \cdot V^\mu(o_{i+1}, g) \right] - \underbrace{V^\mu(o_t, g)}_{\text{baseline}}$$

1053 The baseline $V^\mu(o_t, g)$ leads to lower variance estimates that are particularly useful in our case of
 1054 using the REINFORCE estimator. When γ is close to 0, we have $R_t = V^\mu(o_{t+1}, g) - V^\mu(o_t, g)$
 1055 which is closely similar to a single-step policy improvement for the $-\mathbb{1}[o_t \text{ satisfies } g]$ reward. As
 1056 $\gamma \rightarrow 1$, R_t encourages policies to traverse trajectories along which the states have high value under
 1057 the dataset policy μ (i.e. high V^μ).

1058 **Pointmass Navigation Domain** In our supplementary materials website we include a self-contained
 1059 python notebook implementing Self-Improvement on a pointmass navigation domain. In each episode
 1060 the pointmass starts in a random position, and the goal is for the pointmass to reach a different
 1061 randomly sampled target position. We create a purposely sub-optimal imitation learning dataset
 1062 for this task, where using a PD-controller we navigate to 5 waypoints before heading to the goal
 1063 position. We then execute our proposed fine-tuning procedure on this imitation dataset using MLP
 1064 policy and steps-to-go prediction models. Figure 8 shows sample trajectories from the dataset, as
 1065 well as BC (Stage 1) and Self-Improved (Stage 2) policies. As anticipated, BC policies mimic the
 1066 sub-optimality of the dataset. However, in the second stage, and without access to ground-truth
 1067 rewards, our proposed Self-Improvement procedure very rapidly brings policies close to optimality.
 1068 For reproduction using our self-contained Colab notebook, as well as videos visualizing trajectories
 1069 and steps-to-go predictions, please refer our supplementary materials website.

1070 G Real-World LanguageTable Experimentation Procedure

1071 For all real-world experiments, 1 human was responsible for monitoring all robots and performing
 1072 resets. They did not provide any form of labels or success indicators to the models. Operators were
 1073 instructed to perform resets either when a block drops off the table, or if a station has not been
 1074 shuffled and reset in the past 3-5 minutes of operation.

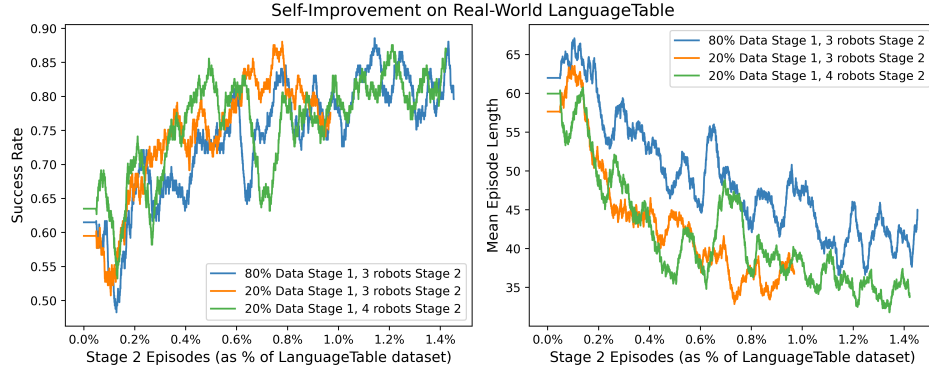


Figure 9: Self-Improvement results on real-world LanguageTable domain. We conducted real-world experiment 3 times: 1) 80% data in Stage 1, Stage 2 fine-tuned on 3 robots simultaneously, 2) 20% data in Stage 1, Stage 2 with 3 robots, 3) 20% data in Stage 1, Stage 2 with 4 robots. In all Stage 2 experiments 1 human monitored and performed period resets for all robots. Each experiment took approximately 20 hours (4 hours \times 5 days).

1075 H Additional Plots

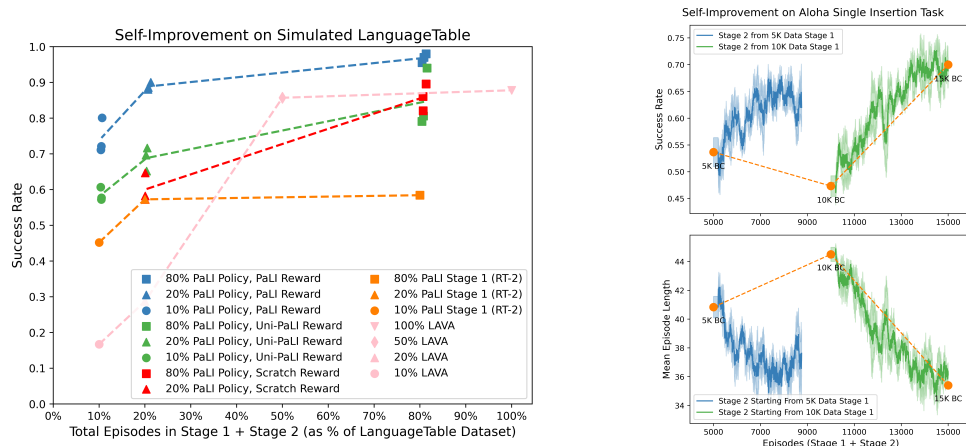


Figure 10: (left) Results and ablations on the simulated LanguageTable domain. We emphasize to the reader that while it appears that the Stage 1 and Stage 2 plots have identical x-axis values, there is no bug in the plot and they are in fact different. The Stage 2 process is simply sample-efficient to the point that the difference in x -axis is negligible. (right) Plots demonstrating the efficacy of the Self-Improvement Process on Aloha Single Insertion Task in the 5K and 10K data settings (3 random seeds each setting). The blue plots demonstrate that despite the much smaller datasets compared to LanguageTable, distributing environment interaction budget between Stage 1 and Stage 2 is a more sample-efficient approach towards obtaining performant policies, as opposed to allocating the full budget to Stage 1 (yellow markers).

1076 I Computing Percentage of Block2Block Instruction in LanguageTable

1077 We used Gemini 1.5 Pro and the structured outputs feature to label LanguageTable instructions as
 1078 being Block2Block or not. For both the simulation and real datasets we randomly sample $N = 5000$
 1079 of the instructions in the dataset, and use the following prompt to classify them as Block2Block.
 1080 Using the structured outputs feature of Gemini we can enforce LLM responses to be either Yes or No.

You are a language model with expertise in determining the structure and type of robotic instructions. Your task is to identify whether a given instruction is a “block to block” type instruction or not. A “block to block” instruction involves moving or pushing one block towards another specific block on a board. This does not include separating two blocks, putting one block in between two blocks, or putting a block near a group of blocks.

Please analyze the following instruction and respond with either “yes” or “no” based on whether it fits the definition of a “block to block” instruction:

Examples where the answer is “yes”:

- “push the red circle towards the blue triangle”
- “push blue cube to the right of green cube”
- “move the red moon towards the bottom left side of the red pentagon”
- “push red pentagon into the green cube”
- “place the yellow star to the left of the red moon”
- “push the green cube vertically below the yellow pentagon”
- “drag green star into blue cube”
- “slide the red star at the bottom right of the green star”

Examples where the answer is “no”:

- “push the blue cube in between yellow star and green star”
- “push the red crescent away from the blue crescent”
- “place the arm to the left of the yellow star”
- “move the blue crescent to the center of the board”
- “adjust the group of blocks to form a circle”
- “separate the green star and the blue cube”
- “push blue moon along with yellow star to the left side”
- “move yellow star and blue moon together slightly to the top side of the board”
- “place the blue cube at the top center”
- “slide blue cube a bit right”

The instruction for you to label as “yes” or “no” is:

1081

1082 For the simulated and real dataset respectively, 47% and 49% of the instructions were labeled as
1083 Block2Block. Using Hoeffding’s Inequality we can see that with $N = 5000$, these estimates are
1084 within 2.8% error margin with 99.9% confidence.

1085 J Interesting Observations & Incomplete Experiments

- 1086 • improving block2block performance make all tasks work better
- 1087 • all instructions experiments
- 1088 • real-world Aloha

1089 K Infrastructure Overview

1090 K.1 Version 1: Non-Local Policy

1091 K.2 Version 2: Local Policy

1092 L Extended Related Works

1093 **Embodied Foundation Models** A number of prior works have leveraged pretrained multimodal
1094 foundation models as robot policies. Driess et al. [17] demonstrate how separately pretrained vision

and language foundation models can be co-trained to create multimodal foundation models. They highlight how these multimodal models can learn to query low-level robot controllers towards accomplishing high-level objectives. Building on this direction Brohan et al. [9] discretize continuous robot actions and map them onto language model token spaces. This method, dubbed RT-2, enables pretrained vision-language foundation models (VLMs) to be fine-tuned as robot policies. This approach was further validated by applying to the Open X Embodiment [16] dataset containing over 1M robot trajectories from 21 institutions, and is the policy architecture we base our work off of. Since RT-2, a variety of works have extended pretrained VLMs by incorporating action prediction heads. Example action head architectures include diffusion models [38, 53] (building on diffusion policies [14, 27]), flow matching [7, 28] (building on [32]), and regression [30]. Critically, in prior works the delineation between pretraining and post-training is the task and data mixture used for offline supervised learning. To the best of our knowledge, our work is the first to move past supervised learning for training robot foundation models. The key contribution of our work is to present a general-purpose, reward-engineering-free, online Self-Improvement procedure that not only leads to rapid policy improvement, but enables acquisition of novel behaviors outside of what the models have seen in their training data. This form of behavioral generalization is only possible through an online post-training mechanism.

Improving Robot Policies Without Ground-Truth Rewards Our goal is to design methods that enable generalist robot foundation models to autonomously become proficient on any downstream task. A key obstacle of general-purpose Self-Improvement through reinforcement learning is that commonly we do not have access to ground-truth reward functions and success metrics, either due to the challenge of designing one (reward engineering), or difficulty in measuring them in the real world (reward instrumentation). A line of prior work, that we dub "Code-As-Rewards", leverages LLMs to write code for reward engineering [35, 55, 52]. Policies are then trained with the designed reward, and the success rates and other feedback are provided to the LLM for improving the reward function. Such approaches have a number of downsides that make them impractical for general-purpose robot learning in the real world: 1) It is notoriously difficult to arrive at intended policies through reward engineering, in particular for dexterous manipulation tasks, 2) The iteration loop of training a policy with a given reward and patching the reward function based on outcomes is impractical for real world robotics, 3) The variables needed in such reward functions require bespoke instrumentation to be measurable outside of simulation domains, 4) We still require a ground-truth success detector to provide feedback to the LLM designing the rewards. Aside from Code-As-Rewards, there exists a rich literature on obtaining data-driven reward functions. Such approaches forego manual reward engineering and instead design expressive representations that scale with increasing data. An important class of works [6, 34, 45, 11] learn latent observation representations on top of which rewards and value functions can be defined (e.g. via L2 distance in latent space to target goals). Kumar et al. [31] use a heuristic of labeling imitation learning datasets with +1 rewards near successful states, and 0 elsewhere. They demonstrate that offline and online RL, using the combination of target task and pre-existing data, can be used to sample-efficiently improve robot policy performance. [21] design a contrastive learning objective that corresponds to a form of goal-conditioned Q-Learning. [10] demonstrate that offline goal-conditioned RL with relabeled goals can lead to sample-efficient downstream fine-tuning for new tasks. In comparison to the above, the key advantage of our proposed approach is the straightforward integration with webscale pretrained foundation models. RobotCat [8] trains a large behavioral cloning Transformer with a similar architecture as Gato [44] on a diverse set of robotics tasks. They demonstrate that policy performance can be improved by rolling out BC policies, hindsight relabeling episodes with accomplished goals, and adding the trajectories back into the imitation dataset. However, it is important to note that using hindsight relabeled supervised learning as a policy improvement procedure can have important failure cases [23]. The closest related works to ours are those based on learning distances in timesteps. Hartikainen et al. [25] present an iterative procedure where 1) policies are rolled out to collect trajectories, 2) steps-to-go prediction between states is updated through supervised learning, and 3) negative distance reward function is used for updating policies through RL. In an offline setting, Hejna et al. [26] also learn steps-to-go between states using supervised learning, estimate shortest paths between states and goals, and use weighted behavioral cloning to obtain improved policies. Aside from important differences in settings, our focus in this work is on foundation models and generalization, demonstrating that steps-to-go policy improvement is a viable path towards general-purpose Self-Improving robot policies. In addition to providing a dense reward signal, our work presents steps-to-go thresholding as a novel path towards obtaining robust open-ended success detectors, which have typically been

1153 trained via binary classification [18]. Lastly, we highlight two extensions of our work. Yang et al.
 1154 [54] demonstrate that our approach is effective in real-world simulators built on generative video
 1155 foundation models. Ma et al. [36] present an extension our work to an in-context learning formulation.
 1156 They demonstrate that the long context capabilities of state-of-the-art foundation models enables
 1157 policy improvement (offline setting), success detection, and dataset filtering on a diverse array of
 1158 tasks without necessitating model fine-tuning.

1159 M Future Work and Limitations

1160 **Episode Boundaries & Skill-Chaining.** The *steps-to-go* auxiliary that underpins our *Self-*
 1161 *Improvement* stage naturally lends itself to hierarchical control. By explicitly annotating the start
 1162 and termination of *sub-skills*, the same progress estimator can be reused to furnish dense sub-task
 1163 rewards, enabling long-horizon skill-chaining reminiscent of option-based RL. At inference time, a
 1164 high-level planner—whether a finite-state machine, a human teleoperator, or a reasoning foundation
 1165 model—can invoke the learned sub-policies and rely on the steps-to-go head to decide when to
 1166 transition to the next skill. The chief obstacle is scalable boundary annotation: manual labeling is
 1167 prohibitively expensive, calling for creative weak- or self-supervised strategies that recover consistent
 1168 boundaries from raw interaction logs. Exploring such automated segmentation is an exciting avenue
 1169 for future research.

1170 **Reward Models.** Because reward inference is performed *offline* in our framework, latency con-
 1171 straints are minimal; we can therefore allocate far larger models—or even iterative, chain-of-thought
 1172 style reasoning [24]—to obtain higher-fidelity labels. Richer reward models may mitigate a key
 1173 failure mode of imitation-based datasets: the absence of mistake-and-recovery trajectories. A more
 1174 expressive steps-to-go estimator could recognise these out-of-distribution states and either assign
 1175 appropriate shaped rewards or trigger a switch to a recovery skill, thereby improving robustness at
 1176 deployment.

1177 **Robotics Foundation Models.** Our study fine-tunes general-purpose vision–language backbones
 1178 that were never exposed to robotics data during pretraining. As larger multimodal corpora of robot
 1179 experience become available, it will be crucial to design pretraining curricula that endow *Robotics*
 1180 *Foundation Models* with strong priors for physical reasoning while preserving their broad visual–
 1181 semantic knowledge [28, 48]. Moreover, a post-training stage analogous to RLHF for language
 1182 models could render the resulting policies effective in a purely *zero-shot* manner, reducing or even
 1183 eliminating the need for downstream fine-tuning. Preliminary evidence from our LanguageTable
 1184 experiments hints that strengthening one task (Block2Block) via Self-Improvement can spill over
 1185 to other instructions—an encouraging sign that such post-training could confer wide generalisation
 1186 benefits.

1187 **RL Algorithms.** For simplicity and stability we elected to use on-policy REINFORCE; however,
 1188 this choice forgoes the data-reuse benefits of modern off-policy algorithms. Investigating off-policy
 1189 variants that scale to large action vocabularies [22] may further curb robot-hour requirements. Another
 1190 promising direction is to keep updating the steps-to-go head during Self-Improvement so that it serves
 1191 as a learned, rapidly-adapting value baseline. Finally, we observe that pushing Self-Improvement
 1192 *beyond* its performance peak can degrade success rates—suggesting that better stopping criteria or
 1193 adaptive regularisers are required to prevent over-optimisation of the shaped reward.

1194 **Summary.** While our two-stage recipe already unlocks substantial gains—with minimal reward
 1195 engineering and a single human supervising multiple robots—addressing the challenges above is
 1196 essential for scaling towards lifelong, autonomous skill acquisition. We look forward to community
 1197 efforts that extend our groundwork to long-horizon tasks, richer reward inference, domain-aligned
 1198 pretraining, and more sample-efficient reinforcement learning.

1199 N Broader Impacts

1200 In this work we present a novel Self-Improvement approach that enables Embodied Foundation
 1201 Models to sample-efficiently improve their task performance, as well as autonomously practice and

1202 acquire novel skills. Enabling autonomous Self-Improvement may have significant positive and
1203 negative societal impacts that should be carefully considered.

1204 **Potential Positive Impacts** Our approach fundamentally improves policy success rates and can
1205 unlock new capabilities. This enables robotics to be applicable in new domains and industries
1206 that previously lacked robotic solutions. We also demonstrated that the combination of SFT and
1207 Self-Improvement is significantly more sample-efficient than supervised learning alone. This sample-
1208 efficiency directly translates to more effective use of resources.

1209 **Potential Negative Impacts** More effective robotic solutions may lead to job loss. Thus, careful
1210 attention to opportunity creation is key for the long term stability of society. Enabling robots to more
1211 efficiently acquire a broader skillset may also enable unacceptable applications of robots by bad
1212 actors.