FLaRe: Achieving Masterful and Adaptive Robot Policies with Large-Scale Reinforcement Learning Fine-Tuning

Jiaheng Hu[1,2], Rose Hendrix[1], Ali Farhadi[1,3], Aniruddha Kembhavi[1,3], Roberto Martín-Martín[2], Peter Stone[2,4], Kuo-Hao Zeng[1,†], and Kiana Ehsani[1,†]

***Abstract*— In recent years, the Robotics field has initiated several efforts toward building generalist robot policies through large-scale multi-task Behavior Cloning. However, direct deployments of these policies have led to unsatisfactory performance, where the policy struggles with unseen states and tasks. How can we break through the performance plateau of these models and elevate their capabilities to new heights? In this paper, we propose FLaRe, a large-scale Reinforcement Learning fine-tuning framework that integrates robust pre-trained representations, large-scale training, and gradient stabilization techniques. Our method aligns pre-trained policies towards task completion, achieving state-of-the-art (SoTA) performance both on previously demonstrated and on entirely novel tasks and embodiments. Specifically, on a set of long-horizon mobile manipulation tasks, FLaRe achieves an average success rate of 79.5% in unseen environments, with absolute improvements of +23.6% in simulation and +30.7% on real robots over prior SoTA methods. By utilizing only sparse rewards, our approach can enable generalizing to new capabilities beyond the pretraining data with minimal human effort. Moreover, we demonstrate rapid adaptation to new embodiments and behaviors with less than a day of fine-tuning. Videos can be found on the project website at `robot-flare.github.io`**

I. INTRODUCTION

Foundation models in computer vision and natural language processing have recently achieved groundbreaking successes. Large-scale transformer models, such as GPT [1] and SAM [2], have demonstrated the ability to perform an extensive range of tasks. Inspired by these advances, the robotics community has set its sights on training high-capacity, multi-task transformers for robotic applications.

One of the prominent methods in this pursuit is large-scale behavior cloning (BC) [3], which leverages large datasets of real-world and simulated demonstrations (e.g., RT-1 [4], RT-2 [5], RT-X [6], and SPOC [7]) to train high-capacity policies that can perform many different tasks. While BC policies have shown promise, they remain fundamentally limited when directly deployed in the real world: models are constrained to the states observed during training, making it difficult to generalize beyond expert trajectories. Consequently, these policies often struggle when faced with unfamiliar states, and fail to recover from errors effectively.

On the other hand, reinforcement learning (RL) [8] offers a complementary approach that directly optimizes the performance of the robot through trial-and-error learning, and

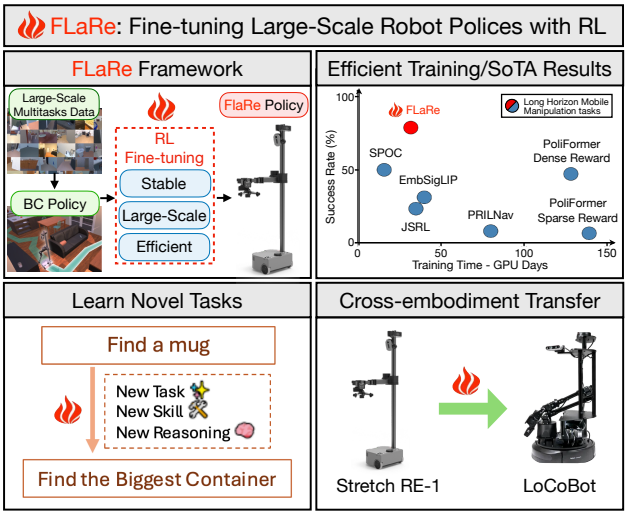


Fig. 1: FLaRe is a simple but effective approach for large-scale fine-tuning of robotic policies. FLaRe achieves SoTA performance on simulation (+23.6%) and real-world (+30.7%) benchmarks, can generalize to unseen tasks, and adapts to new behaviors and embodiments.

RL algorithms have achieved many successes when a well-defined reward function is available [9]–[11]. However, many RL algorithms are notoriously sample inefficient, requiring extensive training time. As the task horizon increases and the action space grows, RL policies struggle to get off the ground due to the large search space. Moreover, RL's reliance on hand-crafted reward functions severely limits its scalability.

Although insufficient for direct deployment, the policies trained through large-scale multi-task Behavior Cloning already possess extremely valuable features and behavior priors. How can we break through the performance plateau of these models and elevate their capabilities to new heights? Our key insight is that, through RL, we can align the behavior of these policies towards true objectives such as task completion (instead of the BC objective), thereby achieving masterful performance not only on tasks seen during BC training, but also on novel tasks and embodiments never seen by the pre-trained policy.

While attempts have been made to fine-tune BC policies with RL [12]–[15], these works are only verified with small-scale networks and in single-task domains. Empirically, we find these methods ineffective as the capacity of the pre-trained policy scales up, where the abrupt shift from BC to RL often results in destructive gradient updates, leading to oscillations or even collapse during RL training.

[1] Allen Institute for Aritifical Intelligence (Ai2)
[2] University of Texas, Austin [3] University of Washington [4] Sony AI
[†] Equal Supervision.

In FLaRe, we introduce an effective, scalable, and robust solution for fine-tuning large-scale robot policies with Reinforcement Learning. Illustrated in Fig. 1 top-left, FLaRe starts from a **multi-task** robotics policy, and fine-tunes it with **large-scale** RL through extensive use of simulation. To ensure the RL fine-tuning is **stable**, FLaRe introduces a set of simple yet highly effective techniques, detailed in Sec. IV-C, that drastically improve performance and reduce training time compared to previous methods.

FLaRe achieves SoTA performance on household mobile manipulation tasks. In established simulation benchmark [7], it achieves an average 79.5% success rate, +23.6% absolute improvements over the best baseline. In the real world, FLaRe achieves excellent results (80.7% SR on average), outperforming the best prior work by +30.7%. Furthermore, FLaRe offers several key advantages:

1) FLaRe enables efficient training with a 15x reduction in training time compared to the previous SoTA method, using a simple sparse reward without the need for handcrafted reward functions (Fig 1 top-right).
2) FLaRe allows for generalization beyond the tasks seen during BC. Even for new tasks without expert trajectories or shaped rewards, FLaRe can be fine-tuned to achieve state-of-the-art performance (Fig 1 bottom-left).
3) FLaRe facilitates rapid adaptation to new embodiments and behaviors, significantly enhancing the base model's flexibility and applicability (Fig 1 bottom-right).

We find that FLaRe marks a promising achievement towards developing highly generalizable robotic systems that can handle a wide range of tasks in diverse environments.

II. Related Work

A. Foundation model for robotics

Following the successes of foundation models in vision [2] and language [1], there has been a recent trend towards training robotics-specific foundation models [16, 17]. While these models focus on different robot applications, such as manipulation (e.g. RT-1 [4], RT-2 [5], RT-X [6], Octo [18], RoboCat [19], OpenVLA [20]), navigation (e.g. ViNT [21]), and mobile manipulation (e.g. SPOC [7]), they share a similar recipe of training a high-capacity transformer model through multi-task behavior cloning [3]. As a result, they generate the same end-product: a multi-task transformer policy, which FLaRe can use as a base model for fine-tuning.

B. RL training and fine-tuning of robotics models

While RL has achieved many successes in robotics [9], directly applying RL from scratch often requires extensive reward engineering and long training time [10, 11, 22, 23]. Hence, previous works have extensively explored leveraging pretrained models to facilitate RL [12]–[15, 24]–[38].

However, many of these approaches focus on fine-tuning models that have been pre-trained using either online RL [24]–[26] or offline RL [39]–[42], which limits their applicability. This makes them unsuitable for fine-tuning most existing robotics foundation models, which are typically trained using Behavior Cloning. Many previous works also require access to the entire offline dataset during fine-tuning [14, 15, 27]–[33], which may be feasible for small-scale data and low-dimensional observations but is unlikely to be computationally feasible for large-scale data and image observations, as also noted by Ramrakhya *et al* [13].

In addition, the techniques proposed in many of these works are only evaluated on simple domains, with low-dimensional state spaces [14, 27, 29], small-scale network architecture (e.g. MLP) [12, 14, 15], single-task pretraining and fine-tuning [32, 43], and often no real robot experiments [15, 27, 29, 34]. PIRLNav [13] and JSRL [12] are two works that are closest to our setting, where only a pretrained policy is required for the fine-tuning phase. However, both of them focus on single-task setting with small-scale networks and no real robot experiments. In contrast, FLaRe explores fine-tuning from large robotics models, where both scalability and applicability to real robots are of critical concern.

III. Problem Formulation

We consider each robotics task $T \in \mathcal{T}$ as a language-conditioned Partially Observable Markov Decision Process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \mathcal{O}, \mathcal{L}, P(s_0), \gamma)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{O}$ is an observation space, $\mathcal{P}$ is a Markovian transition model, $\mathcal{L}$ is a set of natural language instruction, γ is a discount factor, $P(s_0)$ is the initial state distribution, and R is a **sparse** reward function that takes in a natural language instruction $l \in \mathcal{L}$ and a state $s \in \mathcal{S}$ and outputs a binary value indicating whether a given instruction is successfully completed. For the purpose of this paper, we assume that all tasks have the same action space (the actuators of the robot) and observation space (the robot's sensors). Each task $T \in \mathcal{T}$ defines a set of natural language instructions $\mathcal{L}_T$ (e.g., for the task of Object Navigation, potential instructions can be "go to an apple", "find a houseplant", and more). At the start of every episode, an instruction $l_T \in \mathcal{L}_T$ and an initial state $s_0 \sim P(s_0)$ will be sampled. Every time a specific task $T \in \mathcal{T}$ is given, our goal is to train a policy π_θ^T that maximizes the expected return (i.e. success rate) $\mathbb{E}_{\mathcal{L}_T, \pi} \sum_t R(s_t, l)$ for the given task over the possible language instructions $\mathcal{L}_T$.

IV. Method

Considerable effort has been devoted to optimizing performance on robotics tasks via training high-capacity models π_θ with large-scale, multi-task imitation learning [4]–[7]. In practice, these efforts lead to unsatisfactory performance due to compounding errors [44], where small action prediction error leads to state distribution drift. Furthermore, for novel tasks and scenarios where no demonstration data is available, these models have shown limited generalization capabilities, likely due to the limited task coverage of the training data.

FLaRe addresses both problems by fine-tuning the pre-trained model π_θ with RL for each given task $T \in \mathcal{T}$. The key idea of FLaRe is to achieve stable and effective RL fine-tuning through a series of design choices, including 1) utilize a large-scale multi-task model as the base model, 2) achieve large-scale fine-tuning through extensive use of simulations, and 3) a series of algorithmic design to stabilize

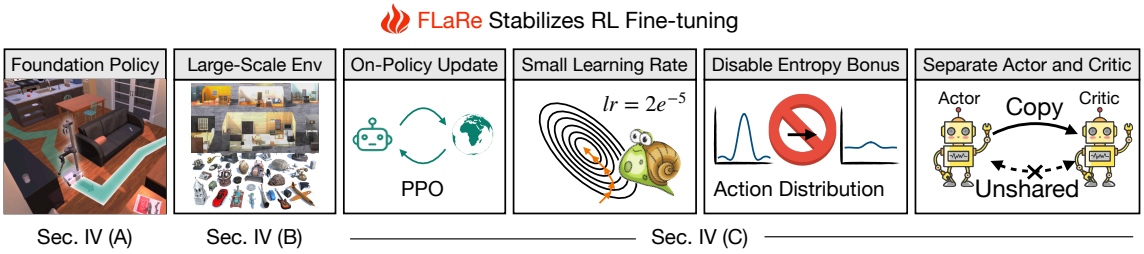


Fig. 2: FLaRe introduces a series of design choices that help stabilize the RL training process, including 1) fine-tuning from a multi-task robotics policy, 2) large-scale fine-tuning in simulation, 3) using an on-policy algorithm as opposed to off-policy methods, 4) utilizing smaller learning rate than when performing RL from scratch, 5) disabling the entropy bonus objective that can potentially distort the policy at the start of the training, and 6) separating the actor and the critic network, so that the critic update will not influence the policy prediction.

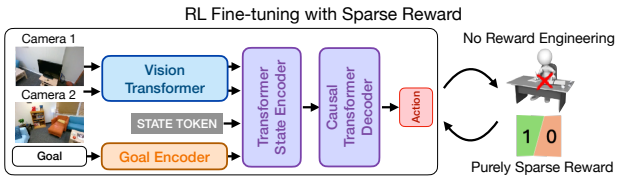


Fig. 3: FLaRe can efficiently fine-tune large transformer policies through large-scale Reinforcement Learning, using a sparse reward function that requires minimal human effort.

the RL fine-tuning. Together, these design choices enable FLaRe to effectively learn from **sparse** reward and achieve good performances. We elaborate in detail on each of these decisions in the following sections (Fig. 2).

A. *Fine-tune from a multi-task robotics model*

The first key design choice of FLaRe is to start from a **multi-task** pre-trained large model (i.e. a foundational robotics model). Compared to fine-tuning from a single-task, small-scale network (as is often the case in previous works [12]–[14, 43]), starting from a robotics foundation model brings three key benefits. First, models pre-trained on diverse tasks can master more robust representations and more versatile behavior priors [45], which will benefit the fine-tuning process. Second, the highly capable network architecture (e.g. large transformer models) that comes with these foundational robotics models brings good inductive bias that can facilitate generalization [46], which is crucial to fine-tuning. Most importantly, the multi-task capability of these models allows us to reuse the same model for fine-tuning for many different tasks. In fact, as we will show in the experiments in Sec. V-B, we can even fine-tune for tasks and embodiments that have never been seen by the pre-trained policy and still achieve good performance.

While our method can in principle work on any foundational robotics model, in this specific work, we focus on fine-tuning the SPOC model (Fig. 3) [7] — a multi-task transformer model for mobile manipulation tasks, trained on large-scale shortest path expert trajectories collected in Objaverse-Populated ProcTHOR houses [47]–[49]. Please find more details regarding the SPOC model in App. VI-E.

B. *Large-scale fine-tuning in simulation*

The second key design choice of FLaRe is to perform large-scale fine-tuning through extensive use of simulation. Recent advancements in robotics and embodied AI have given us a set of tools for simulating robotics tasks [47, 50]–[54]. In this work, we utilize AI2THOR [47] to perform large-scale simulated fine-tuning with diverse objects and scenes, which includes 150k procedurally generated PROC-THOR houses [49] and 800K+ annotated 3D objects [48].

When using simulation in robotics, addressing the sim-to-real gap [55] becomes a critical challenge. In FLaRe, similar to Ehsani *et al.* [7], we employed two techniques to facilitate sim-to-real transfer. First, we perform extensive domain randomization, including color augmentation, applying random crops, and posterizing the images. Second, we extract visual features through DinoV2 [56], a pre-trained foundational vision model, which captures useful features that can generalize across simulation and the real world.

To ensure large-scale training of the transformer policy and value networks, we utilize the KV-cache technique [57] to reduce the computational costs during network inference, similar to Zeng *et al.* [10]. The KV-cache technique caches and reuses the keys and values of earlier observations within an episode. This reduces the inference complexity of the transformer network from quadratic to linear, which is crucial for affordable large-scale RL fine-tuning.

C. *Stabilize RL fine-tuning*

Finally, we introduce a set of simple but very critical algorithmic choices to ensure the stability of RL fine-tuning. While these techniques are relatively simple, as we will show in the ablation studies in Sec. V-E, each choice is very important to ensure stable training and to obtain good performances.

Using On-policy Algorithms. Off-policy RL methods [58, 59] can utilize off-policy data during training, and thus bring the promise of sample-efficient RL. However, compared to on-policy methods, off-policy RL is often less stable and more sensitive to hyperparameters, both in theory and in practice, due to problems associated with the "deadly triad" [8]. In this work, since we perform fine-tuning entirely in

TABLE I: Success and Episode-length weighted Success (SEL) on the CHORES [7] benchmark (in-distribution tasks). Baselines with privileged information are *marked in blue*. FLaRe significantly outperforms the previous SoTA methods.

Success (SEL) ↑	IL+RL: Sparse Reward			IL Only	RL Only		
	FLaRe [Ours]	PIRLNav	JSRL	SPOC	Poliformer - Sparse	Poliformer - Dense	EmbSigLIP - Dense
ObjectNav	85.0 **(67.6)**	20.0 (7.0)	21.0 (15.6)	55.0 (42.2)	14.5 (10.4)	**85.5** (61.2)	36.5 (24.5)
Fetch	**66.9 (54.7)**	0.0 (0.0)	2.9 (2.8)	14.0 (10.5)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
PickUp	**91.8 (90.4)**	0.0 (0.0)	50.9 (47.7)	90.1 (86.9)	0.0 (0.0)	90.1 (88.7)	71.9 (52.9)
RoomVisit	**70.4 (67.1)**	12.5 (11.0)	19.0 (18.6)	40.5 (35.7)	12.5 (12.5)	12.5 (10.9)	16.5 (11.9)

simulation, we are less constrained by the sample efficiency of our RL algorithms, and therefore choose to use on-policy algorithms for stable fine-tuning. Specifically, we use PPO [60], a state-of-the-art on-policy policy gradient method.

Taking Small Update Steps. When setting the learning rate for RL, it is common practice to reuse a learning rate that has previously achieved success in the same/similar domains. However, what we found in FLaRe is that fine-tuning from an existing policy requires significantly lower learning rates than when starting from scratch. For example, in the object navigation task, the previous state-of-the-art result is achieved with PPO from scratch using a learning rate of $2e-4$. In FLaRe, when fine-tuning on the exact same task, we have to reduce the learning rate by an order of magnitude to achieve stable learning. It is important to notice that we do not perform additional LR tuning in FLaRe - the same learning rate is used for all experiments and tasks.

Disabling Entropy Bonus. The PPO objective [60] contains an entropy bonus, which promotes the entropy of the action distribution predicted by the policy network to ensure sufficient exploration. However, we found that when fine-tuning from a pre-trained policy network, this additional entropy term can quickly distort the policy gradient update at the start of the training, leading to unlearning of the pre-trained policy. Hence, we remove this entropy bonus term from our PPO update in FLaRe.

Disabling Feature Sharing. When applying RL to high-dimensional observations such as images, a standard practice is to have a shared feature extractor between the actor and the critic network, which can facilitate the learning of useful features. However, we found that feature sharing during RL fine-tuning can actually hurt the performance since the gradient from the critic loss will change the pre-trained features and lead to the deterioration of the action prediction. Furthermore, during RL fine-tuning, since the pre-trained foundation model should already capture good representations, there is no need for the actor and the critic network to share the same feature extractor. Therefore, in FLaRe, we initialize the policy and the critic network as independent networks, both using the weight and architecture of the pre-trained transformer policy. The policy head of the critic network is replaced by a randomly initialized values.

We found that all four training components are important, and in Section V-E, we show that removing any one of them results in training collapse.

V. Results

We evaluate FLaRe on a set of navigation and manipulation tasks both in simulation and in the real world. Through our experiments, we seek to answer the following questions: **Q1**: Can FLaRe achieve state-of-the-art performance on tasks both within and outside the training data of the pre-trained policy? **Q2**: Can FLaRe learn new capabilities never seen during pre-training and generalize to unseen tasks? **Q3**: Can the policies learned by FLaRe transfer to the real world? **Q4**: Can FLaRe enable efficient adaptation to new robot embodiments and new behavior? **Q5**: Are the stabilization techniques in FLaRe necessary to ensure stable fine-tuning?

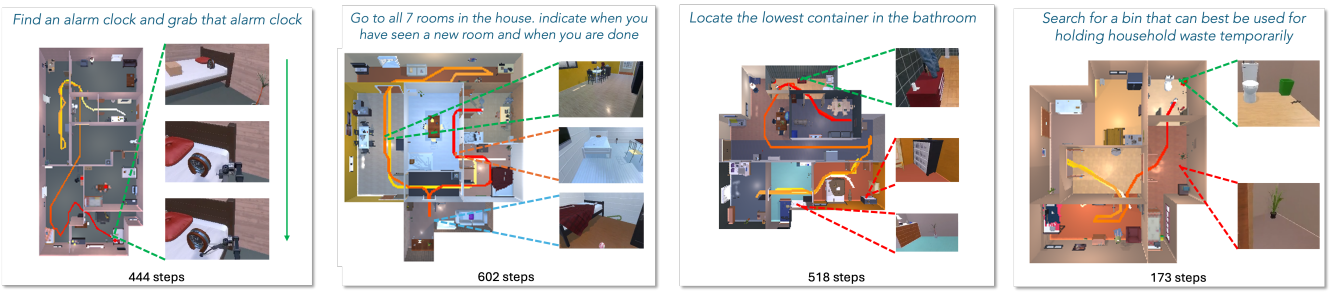
All of the experiments use the same hyperparameters, specified in App. VI-C. Unless stated otherwise, results for FLaRe are obtained using sparse rewards that correspond to task completion. Visualizations of the robot trajectories are shown in Fig. 4 and on our project website.

A. FLaRe on seen capabilities

First, we evaluate the performance of FLaRe in comparison to prior behavior cloning (BC) and reinforcement learning (RL) baselines. Specifically, we test FLaRe on CHORES-$\mathbb{S}$ [7], a recently introduced simulation benchmark designed for household robot tasks. CHORES-$\mathbb{S}$ encompasses four task types that require various skills, including navigation, object recognition, object manipulation, and environment exploration. Similar to [7, 10], the policies use the agent's RGB observations as input to predict discrete actions, which represent movements of the base, arm, gripper, and an *END* action to signify task completion. For further details on the action space, observation space, and task definitions, please refer to App. VI-D.2, VI-D.1, VI-D.3.

CHORES tasks are very challenging due to their long-horizon nature, partial observability, RGB-only observation space, and diverse scenes and objects. Therefore, previous methods struggle to complete these tasks. Since CHORES tasks are contained in the training data of the SPOC model that FLaRe fine-tunes upon, our goal is to utilize FLaRe to improve performance on these in-distribution capabilities.

Baselines. Our baselines consist of prior works in imitation learning, reinforcement learning from scratch, and reinforcement learning fine-tuning from pre-trained policies. Aside from **SPOC** [7], the robot foundation model that we fine-tune upon, we additionally compare against **Poliformer** [10], a transformer-based RL-from-scratch method that achieved SOTA performance on object navigation; **EmbSigLIP** [61], a GRU-based RL-from-scratch method; **PIRL-**



(a) Fetch Task (b) RoomVisit Task (c) ObjNavRelAttr Task (d) ObjNavAfford Task

Fig. 4: We evaluate FLaRe on mobile manipulation tasks. (a, b) In-distribution tasks, in unseen environments. (c, d) Novel tasks that require unseen capabilities from pretraining, in unseen environments. FLaRe excels in long-horizon tasks, showing strong object recognition, relational reasoning, and exploration abilities.

Nav [13], an RL fine-tuning method that employs learning rate scheduling to warm-start the value function; and **JSRL** [12], an off-policy RL fine-tuning method that gradually "roll in" experiences with the prior policy.

We compare against baselines in two settings: (1) a fair-comparison setting, where the baseline methods use the same sparse reward as FLaRe, and (2) an unfair-comparison setting, where the baseline methods use a privileged, task-specific dense reward that has been hand-coded by human experts. It is important to note that each new task demands significant researcher effort to design and curate a dense reward function that avoids collapsing during training and is not scalable to new tasks.

To demonstrate the superiority of FLaRe, all baseline methods are trained for more steps than FLaRe. Specifically, the fair-comparison baselines are trained for $3x$ more steps on ObjectNav and RoomVisit, and $2x$ more steps on Fetch and Pickup. The unfair-comparison baselines are trained until convergence to obtain the best possible result. Notice that this often means significantly longer training time. For example, for the Poliformer (Dense) on ObjectNav, the result is obtained after training for 300M steps - over $15x$ as many training steps that FLaRe uses on ObjectNav.

Results are shown in Table I and Fig. 4 (a, b), where we evaluate on **unseen** simulated houses and report Success rate as well as Episode-length weighted Success (SEL [62]) which measures the efficiency of the policies. As shown by the table, FLaRe not only significantly outperforms the fair-comparison baselines, but outperforms the unfair baseline on three out of the four tasks despite using significantly fewer training steps (**Q1**). Please find training curves in Fig. 6(a).

B. FLaRe on novel capabilities

A well-trained robotics foundation model should learn features useful for all robotics tasks, not only applicable to in-distribution tasks appearing in its original training data. To investigate if FLaRe can take advantage of these pre-trained features, we examine the performance of FLaRe on a set of novel capabilities never seen by the foundation model. Specifically, we evaluate FLaRe on three navigation tasks that specify target objects/locations in different ways and require distinct types of explorations and skills, including

TABLE II: FLaRe can fine-tune for tasks that are never seen by the base model, and achieve state-of-the-art performance. Baselines with privileged information are *marked in blue*.

Success (SEL) ↑	FLaRe [ours]	Poliformer - Sp	SPOC++	Poliformer - De
ObjNavRelAttr	**71.0 (63.6)**	6.7 (6.7)	54.5 (44.6)	36.1 (32.4)
RoomNav	**91.6 (85.6)**	57.0 (51.8)	74.5 (59.9)	75.0 (62.4)
ObjNavAfford	**79.7 (70.6)**	35.5 (29.4)	62.4 (50.6)	53.8 (43.1)

1) ObjNavRelAttr, which identifies target objects through relative object attributes comparison (e.g. "find the largest apple"); 2) RoomNav, which requires the robot to navigate to room types instead of objects (e.g. "go to the kitchen"); and 3) ObjNavAfford, which requires object affordance understanding (e.g. "find something I can sit on"). Note that new reasoning skills are required for these unseen tasks; for example, in ObjNavRelAttr, the agent must search the environment for all objects of the specified type, reason about their properties, and issue a completion action when it identifies the correct instance.

We compare against the Poliformer [10] baseline described in Sec. V-A, as well as SPOC++, a BC baseline that has the same network architecture as SPOC but uses additional expert demonstrations (1M frames per aforementioned task). Note that these demonstrations are not available to FLaRe, nor to the SPOC model that FLaRe fine-tunes.

We show the results in Table II and Fig. 4 (c, d). On these out-of-distribution tasks that require novel capabilities, FLaRe achieves state-of-the-art performance without any additional hyperparameter tuning (**Q2**), even where the baselines have unfair advantages. It is worth noting that, since specifying each of these new tasks T_n is as simple as specifying a success criteria R_n and the associated language instructions L_n, these results imply that we can apply FLaRe to on-the-fly tasks without much engineering effort. This suggests a path towards continual adaptation.

C. FLaRe on real robots

To examine the performance of FLaRe on real robots, we evaluate the policies learned by FLaRe in a real-world apartment on a Stretch RE-1 [63]. This layout (Fig. 5) is never seen by the robot during training. We directly deploy policies without any adaptation or real-world fine-tuning,

TABLE III: Real-world results (total of 46 tasks). For manipulation tasks, we report both full success (policy and heuristic grasping) and policy success (proximity) following [7].

Success Rate ↑	FLaRe [ours]	SPOC	Poliformer - Dense
ObjectNav	**94.4**	50.0	83.3
Fetch	**66.7** (55.6)	33.3 (11.1)	X
PickUp	**86.7** (66.7)	66.7 (46.7)	X
RoomVisit	**75.0**	50.0	X

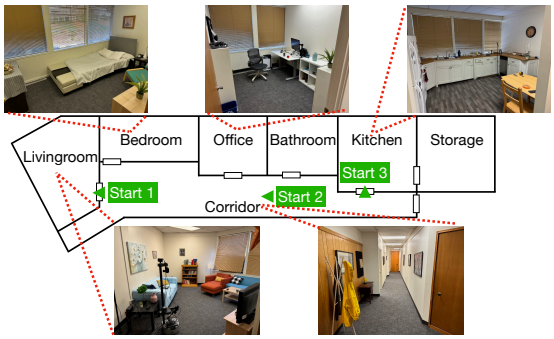


Fig. 5: The real-world layout that we tested upon

and leverage a heuristic object grasping model following SPOC [7]. We compare against SPOC and Poliformer[1] with dense reward, and report the results in Table III. Sim-to-real approaches introduced in Sec. IV-B enable the successes of FLaRe in simulation to directly transfer to the real world, achieving state-of-the-art performances on a set of real world navigation and mobile manipulation tasks (**Q3**).

D. FLaRe for adaptation

FLaRe opens up the possibilities for learning behaviors not captured by the demonstration data (and thus the foundation robotics model). We examine this in two setups, cross-objective and cross-embodiment capabilities of FLaRe (**Q4**).

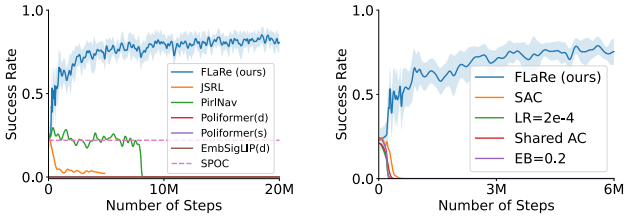
1) Adaptation to New Embodiment: We use FLaRe to fine-tune SPOC (which is trained only on Stretch-RE1) to adapt to Locobot [64]. Locobot has different action space and camera parameters: it lacks the manipulation degrees-of-freedom that Stretch possesses, but has a rotatable, narrower field-of-view camera mounted lower. To facilitate cross-embodiment transition, we simply mask out the invalid actions output by the policy, and repurpose two of the invalid actions to control the camera. FLaRe effectively utilizes the pre-trained policy to adapt to the new embodiment, as shown by the table below on the ObjectNav Task:

New Embodiment	Success Rate ↑	SEL ↑
FLaRe	**72.0**	**47.2**
Poliformer zero-shot[2]	57.5	30.1
Poliformer (Sparse Reward)	44.0	29.7

2) Adaptation to New Behavior: We investigate whether FLaRe can be used to shape a robot's behavior after the policy is trained, using only a few training steps. We test two new behaviors: 1) encouraging the agent to be more efficient

[1]Poliformer reported real-world results only for the ObjectNav Task.

[2]We zero-shot evaluate Poliformer [10] 400M ckpt trained with LoCoBot. This baseline was trained in ProcTHOR-10k, instead of ObjaTHOR houses.



(a) Baselines (b) Ablations

Fig. 6: Baseline performances and ablation studies on the Fetch task. FLaRe is the only method that can achieve good performance on this challenging task.

(+step penalty -0.01/step), and 2) reducing the number of unwanted collisions with the environment (+collision penalty -0.5/collision). By adding a reward term tailored to each behavior, the policy adapts to these new behaviors after just 6 hours of training, with minimal impact on the success rate. The following table presents the results for the Fetch task:

New Behaviors	Success Rate ↑	Episode Length ↓	# of Collisions ↓
FLaRe	**66.9**	258.2	10.0
+ Step Pen.	65.7	**222.8**	10.0
+ Coll. Pen.	66.7	251.2	**3.1**

E. Ablation studies

To evaluate whether the techniques proposed in Sec. IV-C are necessary for the performance of FLaRe, we evaluate four ablation variants of our method. To evaluate whether using on-policy methods is important, we tested switching the PPO algorithm by Soft Actor-Critic [59] (**SAC**). To evaluate whether a small learning rate is necessary, we tested setting the learning rate to $2e-4$, 10 times our original learning rate. To evaluate the importance of having separated actor and critic, we tested **Shared AC**, where the actor and critic share the transformer encoder and decoder trunk. Finally, we tested **EB=0.2**, which set the coefficient of the entropy bonus in PPO to 0.2. We show the training curves in Fig. 6(b).

Perhaps surprisingly, removing any single one of the stabilizing techniques in FLaRe results in the success rate quickly collapsing to 0 on the fetch task, while FLaRe learns very robustly with the same set of hyperparameters across variety of tasks and experiment setups (**Q5**). This showcases the importance of all of the techniques introduced in FLaRe.

VI. CONCLUSIONS

FLaRe is an efficient and scalable approach for fine-tuning large-scale robot policies using RL. It enables effective adaptation to unseen tasks and achieves state-of-the-art performance in both simulation and real-world settings. FLaRe's adaptability to new embodiments and behaviors unlocks the potential for flexible deployment across a wide range of robotic platforms. FLaRe's main limitation lies in its reliance on simulation environments for fine-tuning. While leveraging recent work in simulation generation [65, 66] offers a promising direction, tackling tasks where robust simulations are unavailable—such as those involving liquids or soft objects—remains challenging and may require fine-tuning directly in the real world.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[2] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[3] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[6] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[7] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti *et al.*, "Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 238–16 250.

[8] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[9] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," *arXiv preprint arXiv:2408.03539*, 2024.

[10] K.-H. Zeng, Z. Zhang, K. Ehsani, R. Hendrix, J. Salvador, A. Herrasti, R. Girshick, A. Kembhavi, and L. Weihs, "Poliformer: Scaling on-policy rl with transformers results in masterful navigators," *arXiv preprint arXiv:2406.20083*, 2024.

[11] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," *arXiv preprint arXiv:2004.12570*, 2020.

[12] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao *et al.*, "Jump-start reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 34 556–34 583.

[13] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das, "Pirlnav: Pretraining with imitation and rl finetuning for objectnav," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 896–17 906.

[14] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[15] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Reincarnating reinforcement learning: Reusing prior computation to accelerate progress," *Advances in neural information processing systems*, vol. 35, pp. 28 955–28 971, 2022.

[16] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *arXiv preprint arXiv:2312.07843*, 2023.

[17] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, Z. Zhao *et al.*, "Toward general-purpose robots via foundation models: A survey and meta-analysis," *arXiv preprint arXiv:2312.08782*, 2023.

[18] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[19] K. Bousmalis, G. Vezzani, D. Rao, C. M. Devin, A. X. Lee, M. B. Villalonga, T. Davchev, Y. Zhou, A. Gupta, A. Raju *et al.*, "Robocat: A self-improving generalist agent for robotic manipulation," *Transactions on Machine Learning Research*, 2023.

[20] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[21] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "Vint: A foundation model for visual navigation," *arXiv preprint arXiv:2306.14846*, 2023.

[22] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[23] J. Hu, P. Stone, and R. Martín-Martín, "Causal policy gradient for whole-body mobile manipulation," *arXiv preprint arXiv:2305.04866*, 2023.

[24] A. A. Taiga, R. Agarwal, J. Farebrother, A. Courville, and M. G. Bellemare, "Investigating multi-task pretraining and generalization in reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[25] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *Journal of Artificial Intelligence Research*, vol. 75, pp. 1401–1476, 2022.

[26] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[27] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[28] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[29] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," *arXiv preprint arXiv:1910.11956*, 2019.

[30] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," *arXiv preprint arXiv:2004.10190*, 2020.

[31] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[32] J. Kober, B. Mohler, and J. Peters, "Imitation and reinforcement learning for motor primitives with perceptual coupling," in *From motor learning to interaction learning in robots*. Springer, 2010, pp. 209–225.

[33] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari, D. Kalashnikov *et al.*, "Aw-opt: Learning robotic skills with imitation and reinforcement at scale," *arXiv preprint arXiv:2111.05424*, 2021.

[34] B. Baker, I. Akkaya, P. Zhokov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune, "Video pretraining (vpt): Learning to act by watching unlabeled online videos," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 639–24 654, 2022.

[35] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[36] M. Wołczyk, B. Cupiał, M. Ostaszewski, M. Bortkiewicz, M. Zając, R. Pascanu, Łukasz Kuciński, and P. Miłoś, "Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem," 2024. [Online]. Available: https://arxiv.org/abs/2402.02868

[37] H. Hu, S. Mirchandani, and D. Sadigh, "Imitation bootstrapped reinforcement learning," *arXiv preprint arXiv:2311.02198*, 2023.

[38] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping reinforcement learning with imitation for vision-based agile flight," *arXiv preprint arXiv:2403.12203*, 2024.

[39] M. Nakamoto, S. Zhai, A. Singh, M. Sobol Mark, Y. Ma, C. Finn, A. Kumar, and S. Levine, "Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

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

[41] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online rein-

forcement learning via balanced replay and pessimistic q-ensemble," in *Conference on Robot Learning*. PMLR, 2022, pp. 1702–1712.

[42] A. Kumar, A. Singh, F. Ebert, M. Nakamoto, Y. Yang, C. Finn, and S. Levine, "Pre-training for robots: Offline rl enables learning new tasks from a handful of trials," *arXiv preprint arXiv:2210.05178*, 2022.

[43] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[44] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[45] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE transactions on knowledge and data engineering*, vol. 34, no. 12, pp. 5586–5609, 2021.

[46] B. L. Edelman, S. Goel, S. Kakade, and C. Zhang, "Inductive biases and variable creation in self-attention mechanisms," in *International Conference on Machine Learning*. PMLR, 2022, pp. 5793–5831.

[47] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu *et al.*, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[48] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 142–13 153.

[49] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi, "Procthor: Large-scale embodied ai using procedural generation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5982–5994, 2022.

[50] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 80–93.

[51] E. Jin, J. Hu, Z. Huang, R. Zhang, J. Wu, L. Fei-Fei, and R. Martín-Martín, "Mini-behavior: A procedurally generated benchmark for long-horizon decision-making in embodied ai," *arXiv preprint arXiv:2310.01824*, 2023.

[52] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. W. Clegg, M. Hlavac, S. Y. Min *et al.*, "Habitat 3.0: A co-habitat for humans, avatars and robots," *arXiv preprint arXiv:2310.13724*, 2023.

[53] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.

[54] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 097–11 107.

[55] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[56] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[57] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently scaling transformer inference," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 606–624, 2023.

[58] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[59] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[60] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[61] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: Clip embeddings for embodied ai," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 829–14 838.

[62] A. Eftekhar, K.-H. Zeng, J. Duan, A. Farhadi, A. Kembhavi, and R. Krishna, "Selective visual representations improve convergence and generalization for embodied ai," *arXiv preprint arXiv:2311.04193*, 2023.

[63] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich, "The design of stretch: A compact, lightweight mobile manipulator for indoor human environments," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3150–3157.

[64] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto, "Robot learning in homes: Improving generalization and reducing dataset bias," *Advances in neural information processing systems*, vol. 31, 2018.

[65] M. Deitke, R. Hendrix, A. Farhadi, K. Ehsani, and A. Kembhavi, "Phone2proc: Bringing robust robots into our chaotic world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9665–9675.

[66] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal, "Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation," *arXiv preprint arXiv:2403.03949*, 2024.

[67] L. Weihs, J. Salvador, K. Kotar, U. Jain, K.-H. Zeng, R. Mottaghi, and A. Kembhavi, "Allenact: A framework for embodied ai research," in *arXiv preprint arXiv:2008.12760*, 2020.

[68] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[69] J. Ni, G. H. Abrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang, "Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models," *arXiv preprint arXiv:2108.08877*, 2021.

[70] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford *et al.*, "Robothor: An open simulation-to-real embodied ai platform," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3164–3174.

APPENDIX

A. Results Visualization

We encourage the reader to visit our website (`robot-flare.github.io`) for visualizations of trajectories generated by FLaRe both in simulation and in the real world, including performances visualization, behavior analysis, and failure mode analysis.

B. Hyperparameter

Training and Model Details	
Parameter	**Value**
Total Rollouts	32
Learning Rate	0.0002
Mini Batch per Update	1
Update Repeats	4
Max Gradient Norm	0.5
Discount Value Factor γ	0.99
GAE λ	0.95
PPO Surrogate Objective Clipping	0.1
Value Loss Weight	0.5
Entropy Loss Weight	0.0
Steps for PPO Update	128
Transformer State Encoder Layers	3
Transformer State Encoder Hidden Dims	512
Transformer State Encoder Heads	8
Causal Transformer Deocder Layers	3
Causal Transformer Deocder Hidden Dims	512
Causal Transformer Deocder Heads	8

TABLE IV: Hyperparameters for training and model architecture. We use AllenAct [67] to implement our models and conduct experiments.

C. Number of Training Steps

The base SPOC model that we evaluted and fine-tuned upon is trained for 50k gradient update steps on a total of 100k episodes of demonstrations across the CHORES tasks, where the training hyperparameter and training data is exactly the same as in the original SPOC paper.

For navigation tasks that do not involve manipulating objects (i.e. ObjectNav and RoomVisit), FLaRe performs RL fine-tuning for 20M steps, while all other fair-comparison baseline methods perform RL training for 60M steps. For mobile manipulation tasks (i.e. Fetch and Pickup), FLaRe performs RL fine-tuning for 50M steps, while all other fair-comparison baseline methods perform RL training for 100M steps. For adaptation tasks, we run FLaRe fine-tuning for 50M steps on ObjNavRelAttr and ObjNavAfford, and 20M steps on RoomNav. For cross-embodiment, we run FLaRe for 30M steps.

All of the aforementioned experiments use the same base SPOC mode, with exactly the same set of hyperparameters.

D. CHORES Benchmark

A big portion of FLaRe's evaluation is carried out on the CHORES benchmark. We provided detailed information about this benchmark, including the observation space, action space, and task specifications.

1) Observation Space: The observation space of CHORES consists of two ego-centric 384 × 224 RGB camera pointing towards orthogonal directions, where one points towards the direction of navigation and the other points at the arm. Additionally, a natural language text instruction is re-sampled at the start of each episode and appended to the observation to specify what the robot should be doing.

2) Action Space: The action space of CHORES consists of 20 discrete actions: Move Base (± 20 cm), Rotate Base (±6°, ±30°), Move Arm (x, z) (±2 cm, ±10 cm), Rotate Grasper (±10°), pickup, dropoff, done with subtask, and terminate.

3) Tasks Specifications: We describe the CHORES tasks in Table. V. For each task, if the robot exceeds the maximum steps, the episode is terminated and marked as failed.

For each task, we splited a total of 191,568 houses from ProcThor [49] into training and testing sets with a ratio of 10:1, to ensure that the test evaluation is conducted in unseen houses.

E. The SPOC Model

In this work, we use a slightly modified version of the SPOC model [7] inspired by Poliformer [10], where the transformer decoder block in SPOC is replaced by the decoder from Llama 2 LLM [68] to speed up training and inference. At each step, the SPOC model takes in the new observations consisting of two RGB images and a text instruction. Each of these images are separately passed through a frozen **vision transformer model** (DinoV2 [56]) to extract a set of visual tokens. These tokens, along with an embedding of the natural language instructions using a pre-train text encoder T5 [69], are summarized by a **transformer state encoder** to produce the observation representation. A **causal transformer decoder** then decodes the observations feature across all steps within the current episode into a belief vector that is passed through an actor head to generate the action prediction. We provide a visualization of our model in Fig. 7, and explain each of these components in detail below.

1) Vision Transformer Model: We use DINOv2 as the visual foundation backbone because of its remarkable ability to make dense predictions that generalize across sim and real. Our input to the visual backbone are two RGB observations i_a and i_b. $i_a \in \mathbb{R}^{H\times W\times 3}$ is captured by the navigation camera and $i_b \in \mathbb{R}^{H\times W\times 3}$ is captured by manipulation camera, where H and W are the height and width of the image. The visual backbone then produces a patch-wise representation $r \in \mathbb{R}^{\frac{H}{14}\times\frac{W}{14}\times h}$, where h is the hidden dimensions of the visual representations. r is then reshaped and projected to generate visual tokens $v_{\text{raw}} \in \mathbb{R}^{n_{\text{patch}}\times d_{\text{encoder}}}$. A learnable camera-type embedding is then added to this visual tokens to ensure the model can differentiate between the navigation and the

Task	Description & Example	Max Steps
ObjectNav	Locate an object category: "find a mug"	600
PickUp	Pick up a specified object in agent line of sight: "pick up a mug"	600
Fetch	Find and pick up an object: "locate a mug and pick up that mug"	600
RoomVisit	Traverse the house. "Visit every room in this 5-room house. Indicate when you have seen a new room and when you are done."	1000

TABLE V: CHORES tasks.

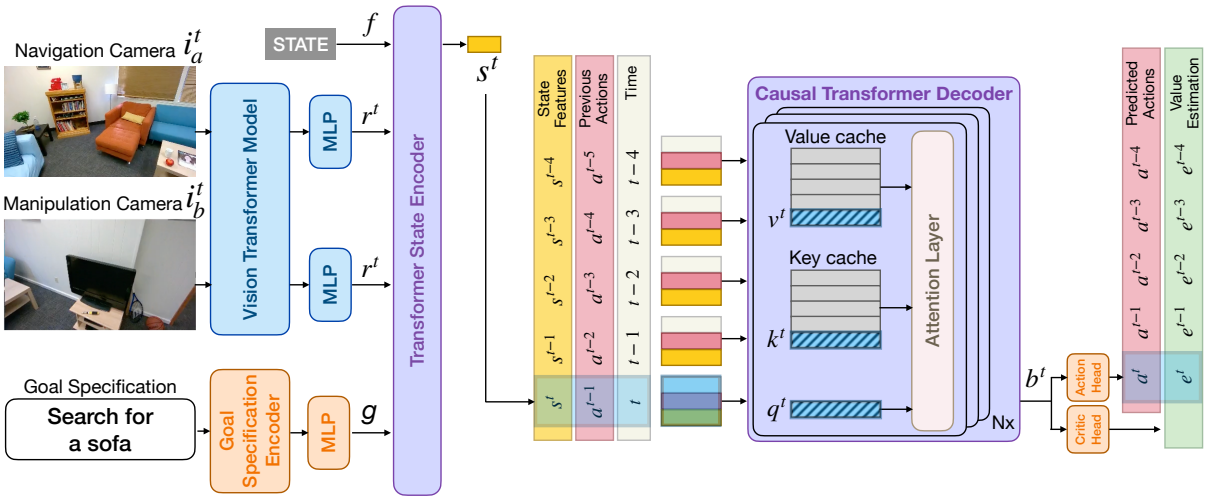


Fig. 7: A visualization of the network architecture of the transformer-based SPOC model that FLaRe fine-tunes upon.

manipulation cameras, resulting in the final visual features v. To ensure sim-to-real transfer, we freeze the DinoV2 weight throughout training.

2) Transformer State Encoder: This module summarizes the observations at each timestep as a vector $s \in \mathbb{R}^d$. The input to this encoder includes the visual representation v, the text feature g, and a learnable STATE token f. We concatenate these features together and feed them to a non-causal transformer encoder. This encoder then returns the output corresponding to the STATE token as the state feature vector. The transformer state encoder digests both visual and text features, and can thus be seen as generating a text-conditioned visual state representation.

3) Causal Transformer Decoder: To deal with partial observability and handle long-horizon tasks, SPOC uses a causal transformer decoder to perform explicit memory modeling over time. The causal transformer decoder consumes the visual representations generated by the transformer state encoder, additively combines them with sinusoidal temporal position encodings and learned previous time step action embeddings, and generates the belief vector used for action generation.

F. Real Robot Setup

Following SPOC [7], we equipped our Stretch RE-1 robot with two identical Intel RealSense 455 fixed cameras, namely the navigation and the manipulation camera. These cameras have a vertical field of view of 59° and are capable of capturing 1280×720 RGB-D images. Both of these cameras point slightly down, with the horizon at a nominal 30°, to optimize the agent's perspective of its functional workspace. The images returned by these cameras are first resized to 396 × 224, and the cropped to 384 × 224, to match the image observations during training.

Same as SPOC, we assess the performance of our models on ObjectNav and Fetch in a 6-room apartment also used in Phone2Proc [65], Pickup in RoboThor [70], and RoomVisit in both environments. The 6-room apartment contains environment variations wholly unseen at train time, including a new configuration (multiple rooms off a long corridor), two new room types (office and corridor), rooms with non-orthogonal wall alignment, and many unseen object instances. For each object in ObjectNav and Fetch, we tested three starting positions: once from the living room, once from the middle of the corridor, and once from the kitchen. We visualize these starting locations in Fig. 5. Below, we provide objects that we tested upon in the real world for each tasks.

1) ObjectNav: Target objects are Sofa, Bed, Chair, Apple, Vase, and Houseplant, each from three starting positions.

2) Fetch: Target objects are Apple, Vase, and Houseplant from the same three starting positions. In one small change from ObjectNav episodes, object instances are replaced with instances which better fit into Stretch's grasping envelope and in some cases at a better height for interaction, but availability and placement are nearly identical.

3) PickUp: Objects are placed on three different surfaces (coffee table, desk, and nightstand) at three different heights. Objects are Apple, Houseplant, Spray Bottle, Mug, and Vase.

4) RoomVisit: The full 6-room apartment is explored, and then partitioned into two 3-room apartments to evaluate the ability of SPOC to explore large and small spaces. We additionally explore a section of RoboTHOR and attached workroom as a novel 3-room apartment.