When Learning Is Out of Reach, Reset: Generalization in Autonomous Visuomotor Reinforcement Learning

Zichen Zhang[†], **Luca Weihs**[†]
[†]PRIOR @ Allen Institute for AI
https://zcczhang.github.io/rmrl

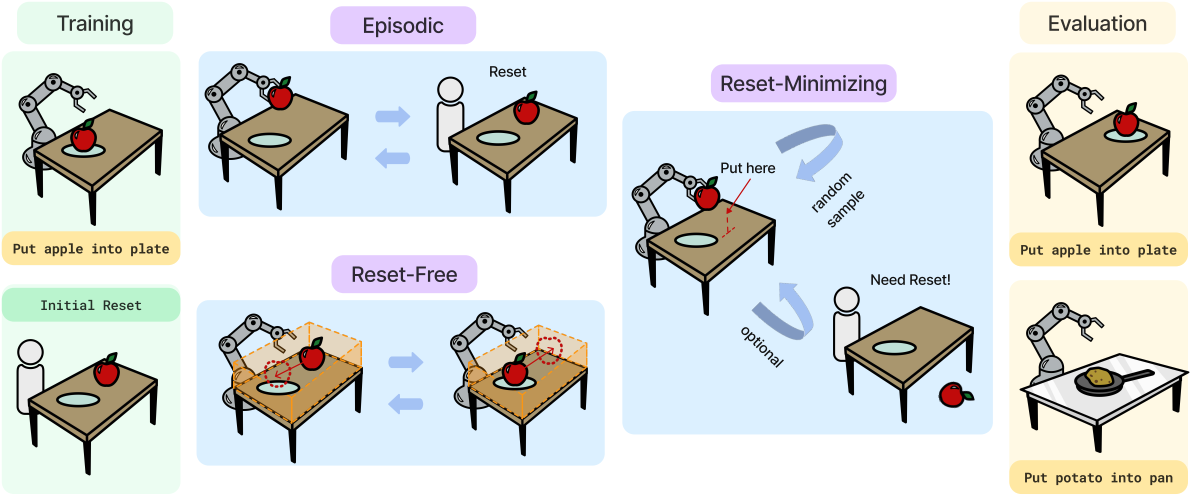

Figure 1: **Episodic, Reset-Free, and Reset-Minimizing RL.** In standard (*i.e.* episodic) reinforcement learning (RL) agents have their environments reset after every success or failure, an expensive operation in the real world. In Reset-Free RL (RF-RL), researchers have designed "reset games" which allow for learning so long as special care is taken to avoid irreversible transitions (*e.g.* an apple falling out of reach). We consider Reset-Minimizing RL (RM-RL) where in realistic and dynamic environments agents may request human interventions but should minimize these requests.

Abstract

*Episodic training, where an agent's environment is reset to some initial condition after every success or failure, is the de facto standard when training embodied reinforcement learning (RL) agents. The underlying assumption that the environment can be easily reset is limiting both practically, as resets generally require human effort in the real world and can be computationally expensive in simulation, and philosophically, as we'd expect intelligent agents to be able to continuously learn without external intervention. Work in learning without any resets, i.e. Reset-Free RL (RF-RL), is very promising but is plagued by the problem of irreversible transitions (e.g. an object breaking or falling out of reach) which halt learning. Moreover, the limited state diversity and instrument setup encountered during RF-RL means that works studying RF-RL largely do not require their models to generalize to new environments. In this work, we instead look to minimize, rather than completely eliminate, resets while building visual agents that can meaningfully generalize. As studying generalization has previously not been a focus of benchmarks designed for RF-RL, we propose a new Stretch Pick-and-Place (*STRETCH-P&P*) benchmark designed for evaluating generalizations across goals, cosmetic variations, and structural changes. Moreover, towards building performant reset-minimizing RL agents, we propose unsupervised metrics to detect irreversible transitions and a single-policy training mechanism to enable generalization. Our proposed approach significantly outperforms prior episodic, reset-free, and reset-minimizing approaches achieving higher success rates with fewer resets in *STRETCH-P&P* and another popular RF-RL benchmark. Finally, we find that our proposed approach can dramatically reduce the number of resets required for training other embodied tasks, in particular for RoboTHOR ObjectNav we obtain higher success rates than episodic approaches using 99.97% fewer resets.*

1. Introduction

A common assumption made when training embodied agents using reinforcement learning (RL) is that the agent's environment can be easily reset after every success or failure: *i.e.*, agents are trained *episodically*. For instance, an agent trained to perform visual navigation that finds itself stuck or lost will be helpfully teleported to a new position (or even placed into a new home) [38, 6, 7] and a mobile manipulation agent that throws all objects onto the floor will find those objects back in their original positions when asked to perform its next task [10, 9, 32, 25, 49]. While this assumption may not induce too high a cost when training agents in simulated environments,[1] resetting an agent's environment in the real world can be extraordinarily expensive, frequently requiring human intervention. This combination of a need for frequent human intervention and data-hungry modern reinforcement learning algorithms has inspired researchers, largely within the robotics community, to pursue *Reset-Free Reinforcement Learning* (RF-RL) [11, 43, 63, 59, 16, 44, 58, 41, 15, 57]. In RF-RL, an agent is placed into an environment in some initial configuration and must, as "reset-free" suggests, learn to perform its task without additional external intervention. An ideal RF-RL algorithm would both save significant expense during initial training and would even allow for agents to continually learn during deployment. Unfortunately, we argue that learning in the reset-free setting can be tremendously difficult due to the presence of irreversible transitions, which can halt learning entirely, and limited state diversity, which harms generalization.

First, for an agent to have any hope of learning in a reset-free setting it is critical that the agent avoids all *irreversible transitions*, namely all state transitions that cannot be undone by taking further actions. For instance, a car whose wheel becomes stuck in a pothole has undergone an irreversible transition as it can no longer, without external intervention, escape that pothole. Similarly, a pick-and-place robotic agent that drops an object beyond its grasp can no longer hope to learn how to interact with that object. Such transitions can simply make RF-RL impossible: an agent can't learn without experience. To avoid these irreversible transitions, many existing works frequently either (1) carefully construct their agent's environment so that irreversible transitions are unlikely or impossible (*e.g.* putting boundaries around a table so that a pick-and-place agent cannot drop an object onto the floor) [16, 44, 58], or (2) provide a set of human demonstrations used to pretrain the agent so that the agent is biased against taking irreversible transitions [15, 57]. Recent works employ these strategies simultaneously [42]. Both of these approaches have clear disadvantages and are not guaranteed to solved the problem of irreversible transitions. Indeed, even when carefully constructing the environment and using additional supervision, it is a common for works studying RF-RL to not be completely "reset-free" as they may reset the environment periodically (albeit after long time horizons) [44, 41, 57] or use heuristics to return at agent to a pre-defined state [42].

Second, while irreversible transitions can be fatal to reset-free learning, a more pernicious problem is that of *limited state diversity*. In the embodied-AI computer vision community, a large emphasis is placed on an agent's ability to generalize. Indeed, existing popular embodied AI benchmarks require that an agent trained in one set of homes is able to successfully complete its task when placed into unseen testing homes containing unseen object instances [38, 49, 54, 10, 6, 25]. When anticipating such significant generalization ability, resets become a critical tool in ensuring that an agent sees a wide variety of unique states. Each reset is an opportunity to place the agent into a room that it has not previously visited or, even, into an entirely new environment. When learning reset-free, however, an agent is constrained to a single environment which it may never exhaustively explore. Moreover, as discussed above, the need to carefully construct environments to avoid irreversible transitions inherently means that the domains in which reset-free agents are trained will be characteristically different than those in which they are deployed. In part due to the above, the problem of reset-free learning has been largely studied in visually simple simulated environments using low-dimensional observations [44, 57] or in real-world robotics settings where generalization across environments is not required [16, 58, 42].

We look to study how one may build visual agents which can learn reset-free and can generalize to new environments. Due to the problems of irreversible transitions and limited state diversity, however, we believe that the "no resets" requirement is simply too strict. Instead, we propose to study *Reset-Minimizing Reinforcement Learning* (RM-RL) where, during training, an agent can request a reset at any point but must attempt to minimize these requests. Similarly as to how competing computer vision models are compared conditional on their parameter counts, this suggests comparing competing RM-RL algorithms conditional on their reset rates. An RM-RL algorithm that achieves a higher success rate than all competitors that use more resets than itself is optimal for that number of resets.

As prior work in RF-RL is primarily set in simple $2D$ or fixed environments where the need for generalization is limited, we create a new *Stretch Pick-and-Place* (STRETCH-P&P) benchmark built in the AI2-THOR simulator [22]. In STRETCH-P&P a mobile manipulation robot (the Stretch

[1] Even in simulation, resetting can be a computationally expensive operation, especially when done frequently (see, *e.g.*, [22, 38, 30, 12, 29]). While rarely documented explicitly, many existing reinforcement learning frameworks for embodied agents (*e.g.* Habitat [38] and AllenAct [56]) employ tricks to so as to minimize the cost of resets during training.

RE1[2]) is placed before a table during training and must move a given object to various target locations, specified by a language prompt, on and around the table. During testing, we evaluate the agent's ability to generalize when faced with positional, cosmetic (*e.g*., colors and materials), and structural (*e.g*., new object instances and furnished scenes) augmentations.

Towards enabling RM-RL, we make two methodological contributions. As our first, and main, methodological contribution we propose, in Sec. 4.1, a collection of well-motivated unsupervised metrics for characterizing when an agent has experienced an irreversible transition. Using these metrics we show, in our experiments, that we can dramatically improve training efficiency suggesting a general framework where an agent requests resets only when necessary. Then, in Sec. 4.2, we describe how, in contrast to the popular, carefully-designed, forward-backward and task-decomposition methodologies used in RF-RL, using a single policy and a random goal sampling approach results in high training performance and enables generalization.

In summary, our contributions include: (1) the STRETCH-P&P benchmark for studying reset minimizing reinforcement learning in visual embodied environments, (2) general metrics designed to characterize when agents have undergone (near-)irreversible transitions during autonomous training, (3) a single-policy, random-goal conditioned, learning strategy for autonomous visuomotor RL control, and (4) extensive experimental results across (mobile, continuous) manipulation and navigation domains in which we ablate our proposed approach and show that our learning framework is efficient and enables generalization. Our benchmark and training code are open-sourced at https://zcczhang.github.io/rmrl.

2. Related work

Reset-free learning. Reset-free learning has been largely studied in visually simplistic grid and MuJoCo [52] style environments [11, 59, 27, 43, 41, 57], and in well-controlled real-world robotics settings carefully designed to avoid irreversible transitions [63, 16, 58, 15, 44, 42]. One popular approach for reset-free RL introduced by Eysenbach *et al.* [11] (see also [17, 37]) involves the joint learning of "forward" and "reset" (or "backward") policies; the forward policy is trained to complete the task of interest (*e.g*. placing a peg into a hole) while the reset policy is trained to reset the environment to some initial state (*e.g*. a peg laying on a table). This forward-reset approach has been further improved, such improvements include the use of curriculum learning to learn harder tasks (VaPRL) [43], encouraging the development of diverse skills via a discriminator-based approach (LSR) [59], and the use of small amounts of expert demonstration data to better inform the reset policy as to which distribution of states it should reset (MEDAL) [41]. Unlike these works, we argue that, when attempting to train mobile, generalizable, agents in high-dimensional visual environments, the problems of irreversible transitions and limited state diversity necessitate developing techniques that minimize resets rather than fully eliminating them.

Avoiding irreversible transitions: safe and reversibility-aware RL. Even in settings where resets are permitted, avoiding irreversible transitions may be preferred for safety or efficiency reasons. Indeed, irreversibility is intimately tied to safety and multiple works have proposed methods to encourage agents to avoid undesirable behaviors via constrained optimization or penalization [5, 1, 51, 36, 48, 33]. A few other works have studied how explicit knowledge of reversibility can improve agent behavior [23, 14]. Xie *et al.* [57] consider the problem of irreversible transitions for reset-free RL and propose "proactive agent interventions" (PAINT) [57] a method which extends MEDAL by training a classifier to predict that it has entered irreversible states (using some ground truth labels for such states). This classifier is then used to allow the agent to request a reset when required. PAINT also penalizes agents for entering irreversible states. Unlike this work, we are most concerned with studying the generalization of reset-minimizing agents within high-dimensional embodied environments. Moreover, as we discuss in Section 4.1, we argue that "ground truth" irreversibility labels frequently suffer from false negatives and so we propose metrics to label such states using unsupervised methods.

Embodied benchmarks. In recent years there have been a significant number of benchmarks proposed for the study of training embodied AI models. Among these, we highlight a few related works that study visual navigation [38, 6, 4, 53, 35, 45], visual mobile object manipulation and rearrangement [49, 25, 13, 55, 10, 9], fine-grained grasping and articulated object manipulation [30, 29], continuous (robotic) control [2, 8, 60], and safety [36]. Most related to our work are the Environments For Autonomous Reinforcement Learning (EARL) [44] and ArmPointNav [10] benchmarks. The EARL benchmark was designed explicitly to study reset-free learning but, unlike our work, does not emphasize reset-minimization, visual observations, or generalization. The ArmPointNav benchmark is, like our work, a mobile manipulation benchmark set in AI2-THOR but is designed for episodic agent training.

3. The Stretch Pick-and-Place Benchmark

Existing benchmarks built to study RF-RL and RM-RL focus primarily on visually simplistic environments with low-dimensional state spaces. Moreover, these benchmarks are designed to evaluate only in-domain performance: an agent is trained in precisely the same environment, and with

[2] https://hello-robot.com/product

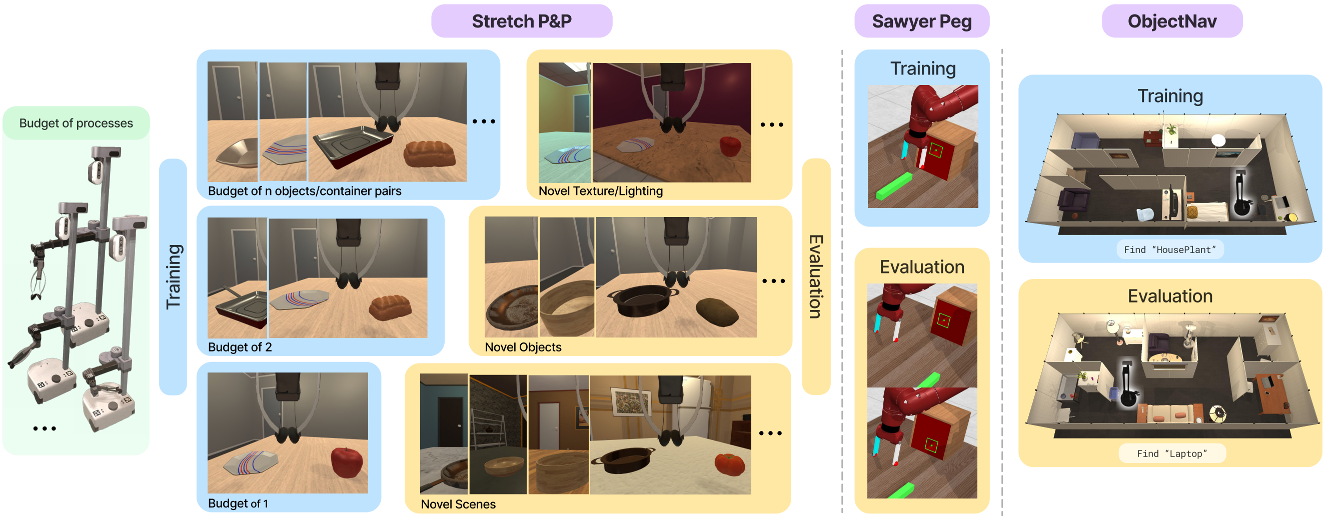

Figure 2: **Overview of proposed Stretch-P&P and other experiment environments.** Here we show the training and evaluation configurations for the Stretch-P&P, Sawyer Peg, and RoboTHOR ObjectNav tasks (from left to right). During training (blue panels), the agent observes: (Stretch-P&P) as few one household object and one container depending on allowed object budget, (Sawyer Peg) exactly one type of stationary box with a goal hole at its upper center, and (ObjectNav) a limited set of house structures. During evaluation (yellow panels), we require the agent to generalize to: (Stretch-P&P) novel cosmetic changes, novel object instances, and a combination of the above alongside with other cosmetic and structural background changes, (Sawyer Peg) novel box and hole positions (the hole position is highlight with green here only for visualization purposes), and (ObjectNav) fully unseen house structures.

the same objects in the same configurations, as in evaluation. As we are interested in studying how RM-RL agents trained with rich, visual, observations are able to generalize when faced with novel objects and visual diversity, we design a new benchmark to evaluate agents in this context.

As we wish to evaluate agents' ability to generalize to both purely cosmetic (*e.g.*, changing the color of materials or lighting of a scene) and structural (*e.g.*, novel objects or scene layouts) changes, we chose to build our benchmark within AI2-THOR [22], a high visual fidelity simulator of indoor environments. While several other alternative visually rich simulators exist, *e.g.* iGibson 2.0 [25] and Habitat 2.0 [49], AI2-THOR offered both a large set of realistic household object instances and a rich set of tools, see *e.g.* ProcTHOR [7], for applying cosmetic and structural augmentations to scenes.

We now define our task. As is highlighted by the struggle of the community to solve even visually simple environments using RF/RM-RL [44, 41, 57]: learning when reset-limited is quite challenging. Given this, an overly complex visual RM-RL benchmark would almost certainly be too difficult to be meaningful in the near term. For this reason, we aim to design a task that, while realistic, does not require excessively long-horizon planning. To this end we design the Stretch Pick&Place (Stretch-P&P) task.

Evaluation. During evaluation in Stretch-P&P, a Stretch RE1 Robot, see Fig. 2, is placed before a table within a room. On this table are two objects, a container (*e.g.* a bowl, plate, *etc*.) and a small household item (*e.g.* an apple, sponge, *etc*.). The agent is given a text description of a task involving how the household item should be moved where this instruction can be semantic *e.g.*, "Put the apple into the plate", or point-based, "Put the apple at location X" where X encodes the relative position between the goal coordinate and the agent's gripper. To study generalization, we consider four separate evaluation settings (see Fig. 2 for visualizations). (1) Positional out-of-domain (Pos-OoD): the environment and objects that must be manipulated are identical to those used during training but the objects' positions and goals are randomized to be much more diverse than those seen in training. (2) Visually out-of-domain (Vis-OoD): object instances are the same as in training but the lighting and the materials/colors of background objects will be varied. (3) Novel objects (Obj-OoD): none of the above visual augmentations will be applied but the container and household object instances will be distinct from those seen during training. (4) All out-of-domain (All-OoD): the agent experiences the visual augmentations from (2), novel object instances as in (3), and the addition of new background distractor objects simultaneously. For more details on these evaluation settings, please see Appendix. B.1.

Training. As in evaluation, the agent is placed before a table with a container and a household object. The table, lighting, and object materials are all kept constant during training. We do not place any constraints on the types of tasks that can be used to train the agent, indeed finding good training tasks is a critical area of study for reset-minimizing learning that we wish to encourage. Upon requesting a reset, the agent's position, as well as the position of the two objects, may be placed into any initial configuration. During training, the agent is intentionally limited to a single environment so as to encourage building tools to enable generalization even in this highly constrained setting. As achieving this generalization may be challenging, we do consider allowing more diversity to be introduced during training by allocating a budget of more than one seen object or container (see Fig. 2, blue areas for STRETCH-P&P). For instance, with a budget of 2, the agent would be allowed to see 2 household objects and 2 containers during training with one of each object being selected at every reset for the agent to manipulate. We provide full descriptions of simulator metadata, success criteria, and object partitions in Appendix. B.1.

Observations and action space. As we show in Fig. 2, the agent's observation is a 224×224 RGB image corresponding to a camera attached to the agent's wrist. We also include proprioceptive sensors corresponding to the agent's arm position. The arm of a Stretch RE1 agent uses a telescoping mechanism to move forward and back, may move up and down, and the gripper has one rotational degree of freedom allowing for changes in yaw, see Figures 2 and 3 for 3rd person views of the stretch robot. The robotic arm of the Stretch RE1 robot is orthogonal to the agent's forward and backward movement and so, to move the arm laterally, the agent must move its body in the forward and backward direction. To highlight the study of irreversible transitions in our benchmark, and as to not add additional complexity to STRETCH-P&P, we restrict the robot body to not rotate in training, although the wrist of the agent may do so. The maximum rotation for the wrist is 2° per step, and horizontal/vertical arm movement is limited to 5cm per step. With perfect execution, success can generally be achieved within 50 steps, thisis similar to other short-horizon, continuous-space, manipulation tasks [60]. See Appendix. B.1 and Table. B.1 for further details regarding the observation and action spaces.

4. Methods

As discussed in Section 1, two fundamental problems when attempting to build generalizable agents in the reset-free setting are irreversible transitions and limited state diversity. As it is often impractical or impossible to guarantee that an agent does not undergo any irreversible transitions during training we will present, in Sec. 4.1, measures that we use to quantify when an agent has undergone such a transition. As we show in our experiments, these measures can be used by the agent to only request resets when it is no longer learning thereby minimizing the total number of resets required to train a performant model. Next, in Sec. 4.2, we take a first step towards building generalizable RM-RL agents; in particular, we propose to do away with the learned forward-reset policies popular in prior RF-RF work and, instead, learn a *single* policy which is presented with randomly generated goals during training.

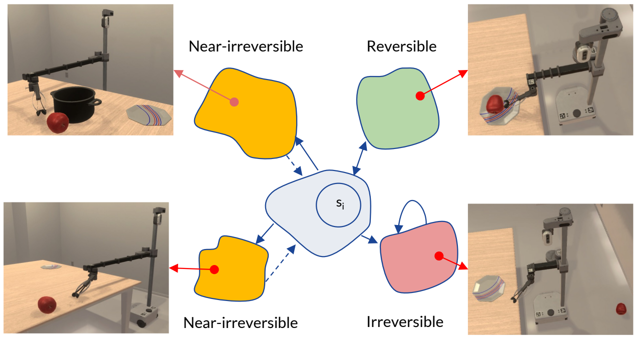

Figure 3: **Reversible and (Near) Irreversible States.** Examples of reversible, irreversible, and NI states for our STRETCH-P&P task. Reversible (top right): the target apple is within easy reaching distance. Irreversible (bottom right): the apple has fallen off the table, as the agent cannot rotate in STRETCH-P&P the apple can no longer be reached. Near-irreversible (left): the apple is in tricky-to-reach locations being behind other objects or at the extreme limits of the arm's reaching capabilities.

4.1. Quantifying Irreversibility

Measures of Irreversibility. Some irreversible transitions are explicit, *e.g.* a glass is dropped and shatters. However, in a more complex real-world environment, they may be more subtle. For instance, when a robot is tasked with cleaning a room, it may encounter situations where some trash is accidentally pushed or blown into hard-to-reach locations, such as under a sofa or in the corners of the room. In such cases, the robot may find it challenging, but not strictly impossible, to pick up or sweep debris back using its regular cleaning tools. We refer to these states that are difficult, but not impossible, to recover from as near-irreversible (NI) states. See Fig. 3 for examples of reversible, irreversible, and near-irreversible states in our STRETCH-P&P benchmark. Explicitly labeling NI and irreversible states can be challenging, as it is practically infeasible to hard-code all possible types of irreversibilities that may occur in different rooms or with different robots in the real world. Complicating this problem further, the set of near-irreversible states also depends on the agent's policy π: which states should be considered near-irreversible can, and should, change during training. For instance, an agent near the end of training may be able to recover after pushing trash beneath a sofa but a near-random agent at the start of training would have little hope of doing so. For this reason, we wish to build measures that, by inspecting the recent behavior of the agent, detect when the agent has undergone a (near-)irreversible transition.

As usual in reinforcement learning for embodied agents, we formalize our setting as a Partially Observed Markov Decision Process (POMDP) $(\mathcal{S}, \mathcal{O}, \mathcal{A}, P_T, A, \mu, \gamma)$ with state space $\mathcal{S}$, partial observation space $\mathcal{O}$, action space $\mathcal{A}$, transition probability measure P_T, reward structure $R : S \times A \to \mathbb{R}$, initial state distribution $\mu \in \Pi(S)$, and discount factor $\gamma \in (0, 1]$. For simplicity of presentation, we will assume that $\mathcal{S}$ and $\mathcal{A}$ are discrete. The goal is to learn a policy π, *i.e.*, a function that maps partial observations to distributions over actions, which maximizes the expected future γ-discounted expected return $\mathbb{E}_{\mu,P_T,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. Let $\tau_\pi(i) \in \mathcal{S}$ for $0 \leq i$ be a random variable representing the state of the environment after an agent has executed its policy π up to timestep i. Here $\tau_\pi(0)$ represents the start state of the agent after a reset (*i.e.*, $\tau_\pi(0) \in \mathcal{I}$ where $\mathcal{I} \subset \mathcal{S}$ is some set of initial starting states). For space, we provide more formal definition and analysis of what we mean by a NI transition in Appendix C.1.

Suppose that agent has taken t steps producing the trajectory $\mathcal{T}_t = \{\tau_\pi(0), \dots, \tau_\pi(t)\}$. Intuitively, undergoing an NI transition should correspond to a decrease in the degrees of freedom available to the agent to manipulate its environment: that is, if an agent underwent an NI transition at timestep i then the diversity of states $\tau_\pi(i+1), \dots, \tau_\pi(t)$ should be small compared to the diversity before undergoing the irreversible transition. This intuition suggests that we may define a collection of irreversibility decision functions, *i.e.* functions that return 1 when predicting that the agent has undergone a near-irreversible transition, using a partitioning approach. In particular, we want to ask: for some width $W > 0$, how many disjoint, continuous, blocks of width W are there in $\mathcal{T}_t$ where the diversity of states within the block is below some threshold $\alpha > 0$. To formalize this, let $P(t) = \{(i_0, i_1, \dots, i_m) : 0 = i_0 < i_1 < ... < i_m = t + 1,\ m > 0\}$. Then we can compute the above count, which we call $\varphi_{W,\alpha,d}(\mathcal{T}_t)$, as

$$\max_{(i_0,\dots,i_m)\in P(t)} \sum_{j=0}^{m-1} 1_{[i_{j+1}-i_j \geq N]} \cdot 1_{\{d(\tau_\pi(i_j),\dots,\tau_\pi(i_{j+1}-1))<\alpha\}}$$

where $d : \mathcal{S}^H \to \mathbb{R}_{\geq 0}$ is some non-negative measure of diversity among W states s. As $\varphi_{W,\alpha,d}$ is a counting function, we can turn it into a decision function simply by picking some count $N > 0$ and deciding to reset when $\varphi_{W,\alpha,d} \geq N$. In particular, we will let $\Phi_{W,N,\alpha,d}$ be the function that equals 1 if and only if $\varphi_{W,\alpha,d} \geq N$. In our experiments we evaluate several diversity measures $d(s_1, \dots, s_H)$ including: (1) an empirical measure of entropy, (2) the mean standard deviation of the s_i, and (3) a euclidean distance-based measure, and (4) a distance measure using dynamic time warping (DTW). Details and algorithm pseudocode of these measures are provided in Appendix C.1. While we find surprisingly robust performance when varying d, we expect that there is no single best choice of diversity measure for all tasks.

The decision function $\Phi_{W,N,\alpha,d}$ can be interpreted as an unsupervised approach for producing labels of (near-)irreversibility. Taking this perspective, it is clear that these labels can be used alongside ground truth labels to supervise the learned irreversibility predictor from PAINT [57]. Indeed the value of these labels is largely orthogonal to contributions from prior work and may be of value to work in RL safety and even in learning to ask for help [47].

4.2. RM-RF with a Single Policy

Instead of the multi-policy forward-backward (FB) approaches frequently employed for RF-RL [16, 1, 44, 41, 57] we propose to train single policy for RM-RL with randomly generated goals that are periodically changed during training. The intuition behind this is straightforward: to enable generalization we allow the agent to practice manipulating objects to many diverse goal positions, as these goals can be periodically changed during training without requiring a reset, this approach encourages the agent to experience explore the state space, improving generalization. This periodic goal switching is analogous to traditional, episodic, RL but with the key distinction that we we do not reset the agent's, or environment's, state when switching goals. Note that, perhaps surprisingly, not resetting the environment may actually result in the agent encountering a more diverse set of "initial" states (*i.e.* states seen when given a new goal) than in the episodic setting as, in the episodic setting, the agent is frequently reset to some fixed initial position.

We will now give a more formal comparison between our approach, FB-RL, and episodic methods. Specifically, recall that in traditional episodic RL, the objective to learn a policy that maximizes the discounted return

$$\pi^\star = \arg\max_\pi \mathbb{E}[J(\pi \mid g)] = \arg\max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t \mid g)\right]$$

where, in comparison to the POMDP formulation from Sec. 4.1, we have, in the above, made the dependence on the randomly sampled goal $g \in \mathcal{G}$ where $\mathcal{G}$ is the set of all goals used during training. In FB-RL, the "forward" goal space is normally defined as a singleton $\mathcal{G}_f = \{g^\star\}$ for the target task goal $g^\star$ (*e.g.* the apple is on the plate, the peg is inserted into the hole, *etc.*). The goal space for "backward" phase is then the (generally limited) initial state space $\mathcal{G}_b = \mathcal{I} \subset \mathcal{S}$ such that $\mathcal{G}_f \cap \mathcal{G}_b = \emptyset$. As the goal spaces in FB-RL are disjoint and asymmetric, it is standard for separate forward/backward policies (with separate parameters) and even different learning objectives to be used when training FB-RL agents. In our setting, however, there is only a single goal space which, in principle, equals the entire state space excluding the states we detect as being NI (*i.e.*, $\mathcal{G} = \mathcal{S} \setminus \{s_t \mid s_t \in \tau_\pi(t) \in \mathcal{T}_\pi, \Phi_{W,N,\alpha,d}(\mathcal{T}_\pi) = 1\}$). In our training setting, we call each period between goal switches a *phase* and, when formulating our learning objectives, treat these phases as separate "episodes" in episodic approaches.

We provide extra details, pseudocode, and comparisons in Appendix C.3.

5. Experiments

In our experiments, we look to answer a number of questions related to: (1) the importance of resets for learning, (2) how challenging STRETCH-P&P is for existing episodic and reset-minimizing approaches, (3) the efficacy of our proposed methodological contributions (unsupervised irreversibility detection and single-policy RM-RF) in reducing the number of resets required for learning and enabling out-of-distribution generalization, and (4) how our methods may be applied more generally to existing embodied tasks. To answer these questions, we run our experiments on three tasks in different embodied settings: our language-conditioned mobile manipulation STRETCH-P&P task, the Sawyer Peg task [60, 43], and the RoboTHOR Object Navigation (ObjectNav) task [6]. The Sawyer-Peg task, described in further detail below, is a popular stationary pick-and-place task used by prior work studying RF-RL and RM-RL. Our Sawyer-Peg experiments primarily use RGB observations rather than the low-dimensional observations used in most prior work. We include this environment to both better compare to prior work and to show the generality of our proposed contributions. Furthermore, to the best of our knowledge, we are the first to utilize methods for autonomous reinforcement learning for ObjectNav, and demonstrate encouraging results. Before moving to our experimental results, we will first provide additional details of our evaluation environments, agent model architectures, and competing baselines.

5.1. Environments

STRETCH-P&P: See Section 3.

Sawyer-Peg: In the Sawyer-Peg task, a Sawyer robot arm is attached to a table and must move a peg object sitting on a table into a hole within a box. As we are interested in the RM rather than RF learning, we remove the barriers from the table during training. So as to allow for evaluating generalization, during testing we consider a setting where *both* the box and target hole are moved into novel positions (see Fig. 2, center). This evaluation setting can be seen as being POS-OOD with and (weakly) OBJ-OOD. Visual observations include wrist-centric and third-person RGB images as in [18, 42]. See Appendix B.2 for more further details.

ObjectNav: To highlight the general applicability of our proposed RM-RL approach for embodied AI tasks, we show initial results on the RoboTHOR Object Navigation (ObjectNav) task [6]. In ObjectNav, an agent (a LoCoBot robot) is placed within a simulated household environment and given a goal object category (*i.e.* TV, chair, table, *etc*.), see Fig. D.9. The agent must then navigate to an object of this category using *only* visual observations. The agent is considered to have successfully completed the task if it executes a special END action and an object of the given category is both visible and within 1m of the agent. During evaluation, must perform the same task, with the same set of object categories, but within unseen household environments (see Fig. 2, right). This can be viewed as scene out-of-domain (SCENE-OOD) evaluation. See Appendix. D for more details.

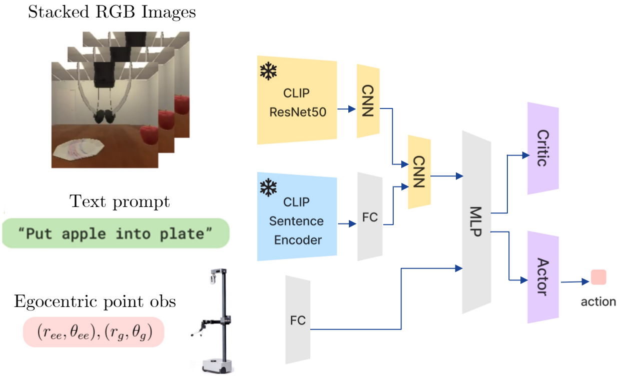

Figure 4: **STRETCH-P&P Model Arch.**

5.2. Training Details

All models are trained using the Proximal Policy Optimization [40] RL algorithm implemented within the AllenAct [56] RL training framework. When training models in Sawyer-Peg (visual), we use 8 parallel processes and a rollout length of 300 for each, training for 1M steps requires approximately 20 minutes with checkpointing every 100k steps. When training models in STRETCH-P&P, we use 16 parallel processes with a rollout length of 200 for each, with 1M steps taking approximately 1.5 hours with checkpointing every 125k steps. Additional training details can be found in Appendix. C.6.

5.3. Model Architecture

As we train our models using the PPO algorithm, our agent architectures are of the actor-critic variety. For fair comparison across training strategies, we use the same model architectures across all baseline models (though this we use different architectures across tasks). A diagrammatic representation of our model architecture used in STRETCH-P&P can be found in Fig. 4. Note that we use frozen CLIP [34] models to encourage visual generalization and language understanding. The model used for Sawyer Peg is similar to that for STRETCH-P&P but does not use CLIP. Unlike separately encoding two views of images in [18, 42], we only use *single* CNN visual encoder that digests both views for parameter-efficiency. We find that single visual encoder is sufficient for solving this task. In ObjectNav, we use the same ResNet50 CLIP architecture with only egocentric visual observation input as proposed in [20]. See Appendix C.4 for more details and hyperparameters.

5.4. Baselines

We consider the following three classes of baseline training strategies.

Periodic resets (+ random goals). Perhaps the simplest strategy for deciding when to request resets is simply to do so after every fixed number of steps. Our periodic resets baselines do precisely this and are labeled simply as "N steps/reset" where N is some positive integer; here N is set to generally be somewhat (or much) larger than in the standard episodic setting. Note that, in principle, there are no irreversible states in ObjectNav as the task merely involves navigating around an, otherwise static, environment. For this reason, we also include a baseline trained without any resets beyond those used to initialize the environment, *i.e.* $N = \infty$, and a baseline trained in the episodic setting just as in prior work [20]. As we show below, choosing when to reset carefully can result in significant improvements in efficiency.

FBRL+GT. Here we implement the popular two-policy forward-backward training strategy from existing work. Inspired by PAINT [57], which learns a classifier trained on ground-truth irreversibility labels to request resets, we will use an *oracle* version of this method and reset the environment whenever the agent enters one of a fixed collection of hand-labeled irreversible states (*e.g.*, target object has fallen off the table). As was discussed in Sec. 4, that this collection may not be exhaustive is a limitation of requiring ground truth labels.

Ours (Random+NI Measure). We report results using our single-policy random-target training strategy with resets being requested based on our unsupervised irreversibility measures, recall Sec. 4.1. As different irreversibility measures may lead to different behavior and performance, we report multiple variants of our approach, one for each of our different irreversibility measures: STD, ENT, DTW, and L2. We describe the details of these irreversibility measures, which differ only based on the state diversity function used, in Appendix. C.1.

Further details of baselines for each task can be found in Appendices A.1, A.2, and D.

5.5. Results

We will now describe our experimental results in the context of the questions they are designed to address.

How important are resets for learning? In Fig. 7 we show training curves for various periodic reset baselines trained for the Sawyer-Peg task. In terms of training steps, the models which reset the most frequently are the most sample efficient with the models resetting less frequently taking millions of steps more to reach high performance. When plotting training performance against the number of resets requested, however, we find that the training curves begin to look very similar with surprisingly consistent trends despite noise inherent in RL training. This suggests that resets are of critical importance: in many cases, many training steps are effectively wasted, we hypothesize that the agent has undergone near-irreversible transitions, as the agent waits for a reset in the future that will enable it to begin learning again. We provide further evidence and point cloud visualizations for this hypothesis in Appendix A.2.3. Thus, as discussed in Sec. 4.2, in the RM-RL setting it is important to evaluate agents considering both their sample and reset efficiency jointly.

Can our unsupervised NI be used to reduce the number resets needed to train performant models? Figure 8 shows the training performance of our method versus other competing baselines for our STRETCH-P&P task with an object budget of 1. We find that our proposed approach is far more efficient in its use of resets: it achieves high success rates more consistently and with far fewer resets than the periodic reset baselines. Surprisingly our method is also *more efficient in terms of training steps*. This suggests that our measures of NI transitions can consistently and accurately identify time-points where a reset will be of *high value for learning*. Despite FB-RL having a set of possible goal states that is more constrained than our method (thus, intuitively, being easier to learn) we find that this constrained goal space appears to have little impact: even when controlling for resets, our method converges to high training performance as, or more quickly, than the FB-RL baseline (see Fig. 8). As we will discuss in more detail below, despite FB-RL having a similar (if somewhat slower) rate of training convergence, our method generalizes far more effectively. Similar trends hold for Sawyer Peg task, see the first row of Fig. 9.

How well do RM-RL agents generalize? We now provide evaluation results for testing the different facets of agent generalization proposed in Sec. 3 for STRETCH-P&P: POS-OOD, VIS-OOD, OBJ-OOD, and ALL-OOD. We also include results for novel box and hole positions for Sawyer Peg. We find, see Fig. 5, that our methods handily outperform competing baselines across all evaluation settings. All evaluations are done for 200 randomly sampled tasks for models saved throughout training (for RoboTHOR ObjectNav we evaluate across the entire validation set which includes 1800 tasks from 15 unseen houses [6]). Notably, for POS-OOD, the easiest evaluation setting in STRETCH-

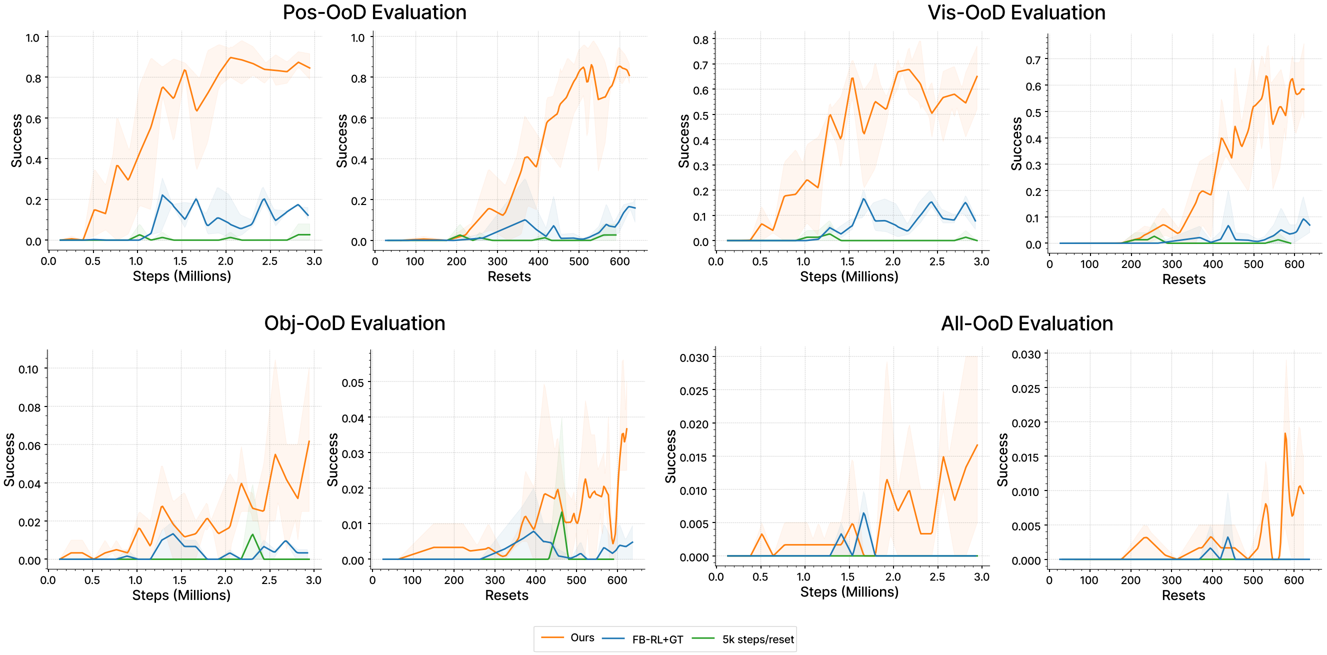

Figure 5: **STRETCH-P&P evaluation results for various methods across at different stages of training.**.

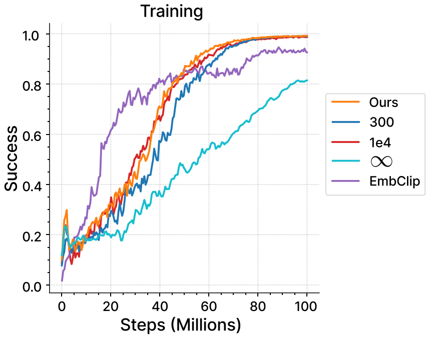

Figure 6: **ObjectNav Training Curve**.

	Success (50M)	SPL (50M)	Resets (50M)	Success (100M)	SPL (100M)	Resets (100M)
Ours	0.216	0.131	**592**	**0.551**	**0.275**	**635**
N=300	0.334	0.166	24k	0.355	0.167	1M
N=10k	0.246	0.134	5k	0.418	0.218	10k
N=∞	0.206	0.141	**60**	0.339	0.178	**60**
[20]	**0.431**	**0.204**	1M	0.504	0.234	2M

Table 1: **ObjectNav results for 50M and 100M steps.**

P&P, our method experiences little to no drop in performance. We attribute this success largely to our random target strategy which results the agent experiencing a diverse set of potential goals. Competing baselines, *e.g.* FB-RL+GT and agents using periodic resets, on the other hand show significantly worse performance on POS-OOD than during training. This is despite the fact that, in POS-OOD, the position of the physical receptacle remains the *same for all baselines*. In Sawyer Peg, we observe similar trends, see Fig. 9, but with somewhat smaller drops in performance for the competing baselines; we attribute this to the smaller state space of Sawyer Peg which makes generalization somewhat easier. Surprisingly the generalization performance given cosmetic augmentations (VIS-OOD) is substantially better than the performance with novel object instances (OBJ-OOD, ALL-OOD). We suspect that this is, in part, due to our use of a frozen CLIP backbone encoder which is largely invariant to such cosmetic differences. Further discussion regarding these results can be found in Appendix. A.1.3.

How does performance vary using different NI measures and object budgets? For brevity and clarity, we provide ablation results for different choices for our proposed NI measures, and for varying object budgets (recall Sec. 3 and Fig. 2) for STRETCH-P&P in Appendix. A.1.4 and Fig. A.2, and for Sawyer Peg in Appendix A.2.2 and Fig A.4. In general, we find that our results are quite robust across diversity measures. We also find that the evaluation results for the OBJ-OOD and ALL-OOD settings in STRETCH-P&P gradually increase when using larger object budgets but, perhaps surprisingly, larger budgets result in lower performance for POS-OOD and VIS-OOD. We suspect that this

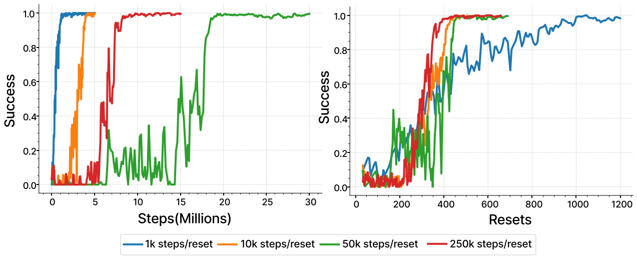

Figure 7: **When controlling the number of resets, training curves converge.** Here we show the success of episodic baselines for the Sawyer-Peg task during training when plotting success against the number of training steps and, alternatively, against the total number of resets taken. In the left plot (Success *v.s.* Steps) we see that methods with more frequent resets appear to substantially outperform, succeeding with far fewer total training steps. The right plot, however, tells a different story: when controlling for the number of resets, all methods learn at roughly the same rate. This suggests that resets are a fundamental part of what drives learning.

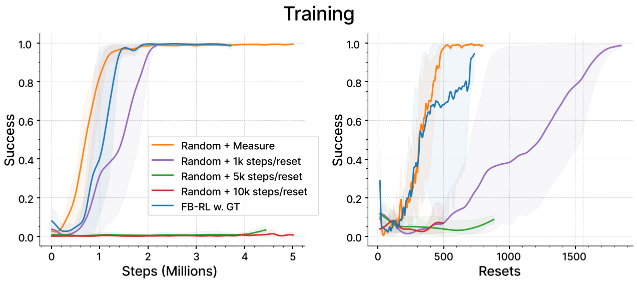

Figure 8: **STRETCH-P&P training performance.**

may be because learning how to manipulate a single object is easier and allows for the agent to learn more complex manipulation behavior within the limited training time. See Appendix A.1 and A.1.4 for further discussions.

Is our RM-RL pipeline general and applicable to different embodied domains? We show our results of training RoboTHOR ObjectNav agents in Fig. 6 with evaluation results in Table 1. The results are very promising: after 100M steps with only 635 resets we are able to achieve success rates **higher** than all competing baselines despite the next best performing baseline using 2M resets. Note also that our agent takes the vast majority (592 of 635) of its resets within the first 50M steps of training showing that our model continues to learn (going from a success rate of 0.216 at 50M training steps to 0.551 at 100M training steps) using very few resets. See Appendix. D for more details.

6. Conclusion

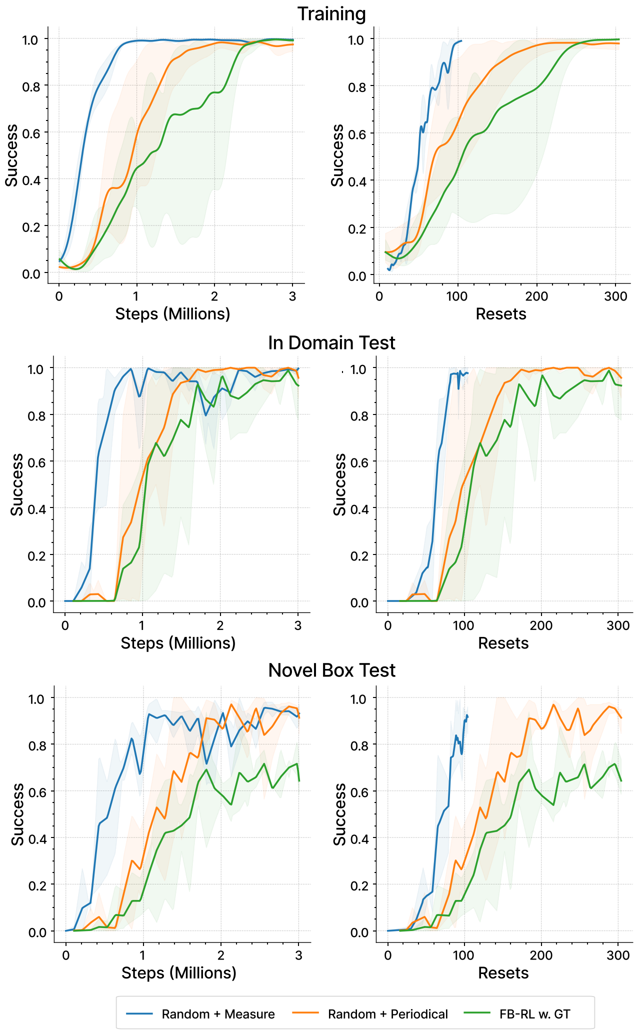

Figure 9: **Sawyer-Peg Results.**

In this work we study the problem of training Reset-Minimizing Reinforcement Learning (RM-RL) agents within visually complex environments which can generalize to novel cosmetic and structural changes during evaluation. We design the STRETCH-P&P benchmark to study this problem and find that two methodological contributions, unsupervised irreversible transition detection and a single-policy random-goal training strategy, allow agents to learn with fewer resets and better generalize than competing baselines. In future work we look to further explore the implications of our irreversible transition detection methods for improving RM-RL methods and for building models that can ask for help during evaluation. We also leave the space for design and balancing of how to penalize visits to unexpected NI states (with labels provided by our method), which may potentially conflict with encouraging exploration, as future work.

Acknowledgements

We thank Winson Han, Eli VanderBilt, Alvaro Herrasti, and Kiana Ehsani for their assistance in the use of the AI2-THOR Stretch RE1 agent and in designing our experimental environment. We also thank Aniruddha Kembhavi and David Forsyth for helpful discussions during this project.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 2017. 3, 6

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. 3

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 23

[4] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, volume 12351 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2020. 3

[5] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *J. Mach. Learn. Res.*, 18:167:1–167:51, 2017. 3

[6] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. Robothor: An open simulation-to-real embodied ai platform. In *CVPR*, 2020. 2, 3, 7, 8, 24

[7] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *NeurIPS*, 2022. Outstanding Paper Award. 2, 4, 24

[8] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1329–1338. JMLR.org, 2016. 3

[9] Kiana Ehsani, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Object manipulation via visual target localization. *arXiv*, 2022. 2, 3

[10] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ManipulaTHOR: A Framework for Visual Object Manipulation. In *CVPR*, 2021. 2, 3, 18

[11] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017. 2, 3

[12] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. 2

[13] Chuang Gan, Siyuan Zhou, Jeremy Schwartz, Seth Alter, Abhishek Bhandwaldar, Dan Gutfreund, Daniel L. K. Yamins, James J. DiCarlo, Josh H. McDermott, Antonio Torralba, and Joshua B. Tenenbaum. The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied AI. *CoRR*, abs/2103.14025, 2021. 3

[14] Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. There is no turning back: A self-supervised approach for reversibility-aware reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1898–1911, 2021. 3

[15] Abhishek Gupta, Corey Lynch, Brandon Kinman, Garrett Peake, Sergey Levine, and Karol Hausman. Demonstration-bootstrapped autonomous practicing via multi-task reinforcement learning. *arXiv preprint arXiv:2203.15755*, 2022. 2, 3

[16] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021. 2, 3, 6

[17] Weiqiao Han, Sergey Levine, and Pieter Abbeel. Learning compound multi-step controllers under unknown dynamics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pages 6435–6442. IEEE, 2015. 3

[18] Kyle Hsu, Moo Jin Kim, Rafael Rafailov, Jiajun Wu, and Chelsea Finn. Vision-based manipulators need to also see from their hands. *arXiv preprint arXiv:2203.12677*, 2022. 7, 8, 19, 23

[19] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot

manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022. 15

[20] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: CLIP embeddings for embodied AI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 14809–14818. IEEE, 2022. 8, 9, 24

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 25

[22] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Kumar Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *ArXiv*, abs/1712.05474, 2017. 2, 4

[23] Maarja Kruusmaa, Yuri Gavshin, and Adam Eppendahl. Don't do things you can't undo: Reversibility models for generating safe behaviours. In *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*, pages 1134–1139. IEEE, 2007. 3

[24] Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J Lim. Generalizable imitation learning from observation via inferring goal proximity. *Advances in neural information processing systems*, 34:16118–16130, 2021. 24

[25] Chengshu Li, Fei Xia, Roberto Mart'in-Mart'in, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In *CoRL*, 2021. 2, 3, 4

[26] Yunfei Li, Tian Gao, Jiaqi Yang, Huazhe Xu, and Yi Wu. Phasic self-imitative reduction for sparse-reward goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pages 12765–12781. PMLR, 2022. 24

[27] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Reset-free lifelong learning with skill-space planning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 3

[28] 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. 24

[29] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Pooria Poorsarvi Tehrani, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments, 2023. 2, 3

[30] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Cathera Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 2, 3

[31] 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. Citeseer, 1999. 24

[32] Tianwei Ni, Kiana Ehsani, Luca Weihs, and Jordi Salvador. Towards disturbance-free visual mobile manipulation. *arXiv*, 2021. 2

[33] Tianwei Ni, Kiana Ehsani, Luca Weihs, and Jordi Salvador. Towards disturbance-free visual mobile manipulation. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023*, pages 5208–5220. IEEE, 2023. 3

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. 7

[35] Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel Xuan Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv*, 2021. 3

[36] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019. 3

[37] Charles Richter and Nicholas Roy. Safe visual navigation via deep learning and novelty detection. In Nancy M. Amato, Siddhartha S. Srinivasa, Nora Ayanian, and Scott Kuindersma, editors, *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, 2017. 3

[38] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 2, 3

[39] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 24

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. 7, 21

[41] Archit Sharma, Rehaan Ahmad, and Chelsea Finn. A state-distribution matching approach to non-episodic reinforcement learning. *arXiv preprint arXiv:2205.05212*, 2022. 2, 3, 4, 6

[42] Archit Sharma, Ahmed M. Ahmed, Rehaan Ahmad, and Chelsea Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning, 2023. 2, 3, 7, 8, 23

[43] Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34:18474–18486, 2021. 2, 3, 7

[44] Archit Sharma, Kelvin Xu, Nikhil Sardana, Abhishek Gupta, Karol Hausman, Sergey Levine, and Chelsea Finn. Autonomous reinforcement learning: Formalism and benchmarking. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 2, 3, 4, 6, 16, 19, 22

[45] Bokui Shen, Fei Xia, Chengshu Li, Roberto Mart'in-Mart'in, Linxi (Jim) Fan, Guanzhi Wang, S. Buch, Claudia. Pérez D'Arpino, Sanjana Srivastava, Lyne P. Tchapmi, Micael Edmond Tchapmi, Kent Vainio, Li Fei-Fei, and Silvio Savarese. igibson, a simulation environment for interactive tasks in large realistic scenes. In *IROS*, 2021. 3

[46] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022. 23

[47] Kunal Pratap Singh, Luca Weihs, Alvaro Herrasti, Aniruddha Kembhavi, and Roozbeh Mottaghi. Ask4help: Learning to leverage an expert for embodied tasks. In *NeurIPS*, 2022. 6

[48] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep RL with a safety critic. *CoRR*, abs/2010.14603, 2020. 3

[49] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Xuan Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021. 2, 3, 4

[50] Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pages 5981–5988, 2020. 23

[51] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 3

[52] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 3

[53] Saim Wani, Shivansh Patel, Unnat Jain, Angel X. Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 3

[54] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *CVPR*, 2021. 2

[55] Luca Weihs, Matt Deitke, Aniruddha Kembhavi, and Roozbeh Mottaghi. Visual room rearrangement. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5922–5931. Computer Vision Foundation / IEEE, 2021. 3

[56] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. AllenAct: A framework for embodied AI research. *arXiv*, 2020. 2, 7, 24

[57] Annie Xie, Fahim Tajwar, Archit Sharma, and Chelsea Finn. When to ask for help: Proactive interventions in autonomous reinforcement learning. *arXiv preprint arXiv:2210.10765*, 2022. 2, 3, 4, 6, 8, 14, 16, 17, 18, 22

[58] Kelvin Xu, Zheyuan Hu, Ria Doshi, Aaron Rovinsky, Vikash Kumar, Abhishek Gupta, and Sergey Levine. Dexterous manipulation from images: Autonomous real-world rl via substep guidance. *arXiv preprint arXiv:2212.09902*, 2022. 2, 3

[59] Kelvin Xu, Siddharth Verma, Chelsea Finn, and Sergey Levine. Continual learning of control primitives: Skill discovery via reset-games. *Advances in Neural Information Processing Systems*, 33:4999–5010, 2020. 2, 3

[60] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Metaworld: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. 3, 5, 7, 19

[61] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Dee M, Jodilyn Peralta, Brian Ichter, Karol Hausman, and Fei Xia. Scaling robot learning with semantically imagined experience. In *arXiv preprint arXiv:2302.11550*, 2023. 15

[62] Yunchu Zhang, Liyiming Ke, Abhay Deshpande, Abhishek Gupta, and Siddhartha Srinivasa. Cherry-picking with reinforcement learning. *arXiv preprint arXiv:2303.05508*, 2023. 15

[63] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real world robotic reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 2, 3

Summary of Appendices

These appendices include:

A Additional experimental results, analysis, and details.

B Low-level details of our experimental environments.

C Implementation details for our irreversibility measures, model architectures, and training pipelines.

D Here we present preliminary but highly promising results when training Object Navigation agents in a reset-minimizing setting using our proposed approach.

A. Extended Experiments

A.1. Additional STRETCH-P&P Details & Analysis

A.1.1 Baseline details

We first provide additional details regarding the differences between the baselines included in our experiments.

• **Ours: random targets + measurement-lead interventions**: Trained using random targets (see Sec. C.3 for algorithm details) and using either a (1) dispersion-based measure (STD, ENT), or a (2) Distance-based measure (L2, DTW). See Sec. C.1 for the details.

• **Random targets + periodic interventions**: Trained using random targets with periodic resets taken every (1) 1k, and (2) 5k steps. No further resets are given.

• **FB-RL + GT**: FB-RL with both the periodic interventions and oracle explicit irreversible interventions (*e.g.* the apple is dropped off the tabletop). This can be considered a “ground-truth“ variant of PAINT [57] as the intervention predicted by trained classifier is replaced with the oracle reset immediately after an object is dropped off the table. We also tried to use similarly scaled discrete penalties for visiting irreversible states illustrated in [57] and found no significant difference.

We use an object budget of 1, *i.e.* the single task with the prompt `“Put red apple into stripe plate.”` for consistent comparison across baselines, and budget of $1, 2, 4$ for ablation of our method. Therefore, when using random targets during training the agent is told to manipulate the red apple to *any* point goal *over* the table, whereas FB-RL aims to move the apple back and forth from the plate to *pre-defined* initial states *on* the table. The targets, both randomly sampled and from pre-defined forward-backward states, are switched every 300 steps during training.

We now provide detailed discussions of our experimental results during training and evaluation.

A.1.2 STRETCH-P&P Training Performance

We first discuss additional results regarding the training-time performance and efficiency of our various baselines.

• **Reset and sample efficiency**: as shown in Fig. 8, our method achieves both high sample, and reset, training efficiency when compared to other baseline models.

• **Toward resetting only when necessary**: We provide an additional ablation when using more, or less, frequent periodic resets during training. As shown in Fig. 8 simply increasing or decreasing the frequency of periodic resets does little to bridge the gap between our approach and these baselines. Resetting every 1k steps results in similar efficiency in terms of the training steps but requires 3× times more resets. Alternatively, resetting every 5k steps results in similar a similar number of total resets after 5M training steps as our method but, unlike our method, results in a very poor success rate.

• **Random targets introduce little additional difficulty over FB-RL**: Intuitively the forward-backward gameplay of FB-RL models should be easier to learn than when using random targets as the space of goal states of FB-RL is a small subset of those used when randomizing targets. However, even when we attempt to control for resets (similar reset rate for convergence in Fig. 8), this appears not to be the case. The runs trained with FB-RL, even with the immediate ground truth interventions for explicit irreversible states, exhibit slightly slower training convergence than our method (see Fig. 8). Therefore, constraining the goal space during training appears to have little impact on simplifying the autonomous learning process. Moreover as, during deployment, we only care about the forward policy trained by FB-RL, it seems intuitively wasteful to spend half of the training time learning to autonomously reset instead of practicing more relevant tasks. Using a single policy with random targets makes every phase equivalent and thereby may result in more efficient use of training time.

A.1.3 STRETCH-P&P Evaluation Results

We now discuss additional evaluation-time results of our trained agents. In particular, we provide additional insight into the generalization abilities of our trained models. Recall, Sec. 3, that we have four evaluation settings testing different facets of agent generalization: POS-OOD, VIS-OOD, OBJ-OOD, and ALL-OOD.

• **POS-OOD**: In POS-OOD, the picking household objects and target receptacles are placed randomly within reach on the table. Due to limited visibility, one or both objects may not be visible to the agent initially. Additionally, the Stretch robot’s unique characteristics, which allows for vertical and horizontal movement but no change in arm

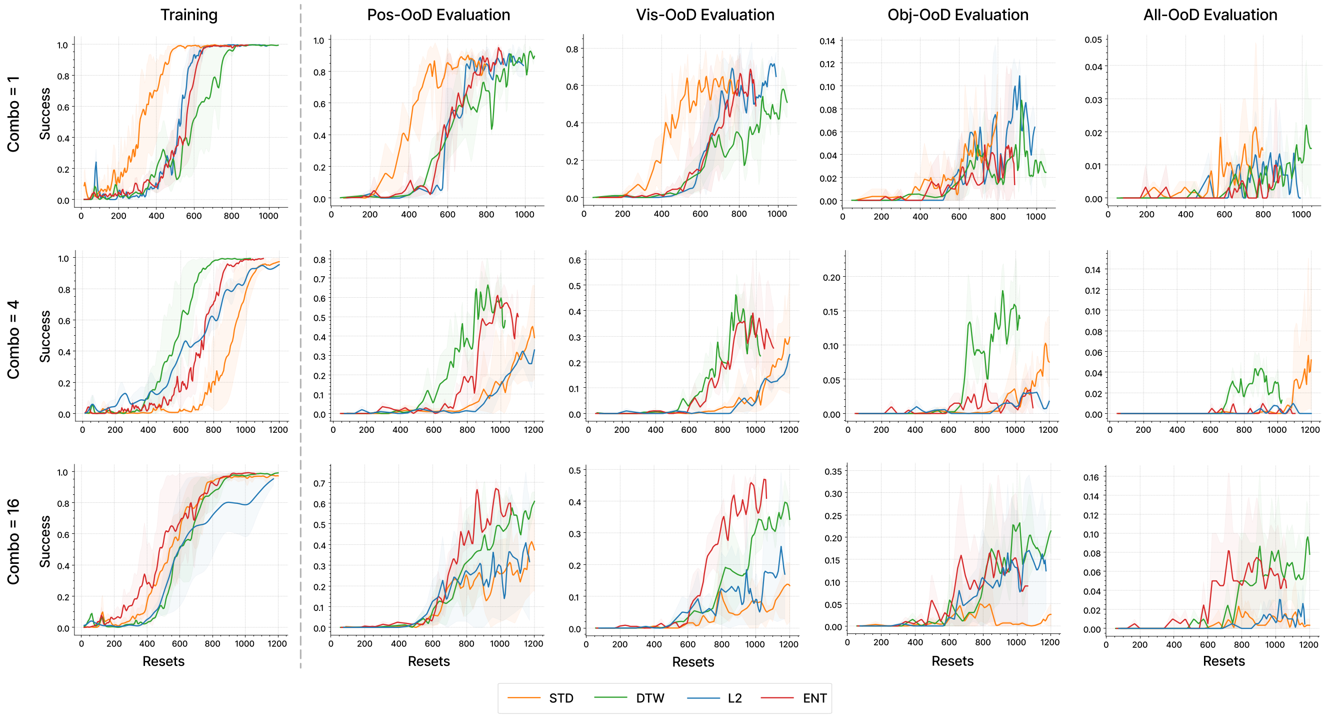

Figure A.2: **Stretch-P&P Irreversibility measures & object budget ablations**. Our measurement-determined irreversibility reset approach is relatively robust to the selection of diversity measure. Note that all models are trained for 5M environment steps but the total number of resets taken by each method may differ.

pitch, make it difficult for the agent to differentiate between larger objects further away from the gripper and smaller objects closer to it. Our method achieves the highest degree of positional-invariance among all the methods, see Fig. 5 (top left). We suspect that next best baseline, FB-RL+GT, method was not able to generalize well due to the constrained space of goal states it observes during training. Interestingly, using an object budget 2 or 4, see Fig. A.2, results in lower performance compared to when using a budget of 1. We suspect that this may be because learning how to manipulate a single object is easier and allows for the agent to learn more complex manipulation behavior within the limited training time.

- **Vis-OoD**: We found, see Fig. 5 (top right), that our method can generalize to the unseen cosmetic changes, *i.e*. novel textures (*e.g*. texture and materials of the table, floor, wall, ceiling) and lighting (random light color and intensity). See Sec. B for randomization details. We attribute this generalization ability in part to the pretrained CLIP backbone used in our model (see Appendix C.4 for architecture details). It is interesting that the performance drops while the object budget increases (see Fig. A.2); we suspect this is due the increase in budgets making training more challenging.

- **Obj-OoD**: All baselines performed relatively poorly in this setting, Fig. 5 (bottom left), though the models with the largest object budget (a budget of 4) during training perform the best, see Fig. A.2. This is not surprising as the language backbone is completely frozen and the visual inputs are only raw RGBs. We suspect that it will be possible to achieve higher success if a perception module with more input streams or augmentations is used. For example, we might include depth images, object center positions estimated in pixel space (as in [62]), detected object bounding boxes (as in [19]), or generative augmentations (as in [61]). As a baseline model, we kept the input as the raw RGB image without augmentation and we leave potential improvements as future work.

- **All-OoD**: For our most challenging evaluation setting the room structure (*e.g*. furniture at the background), texture and lightning in Vis-OoD, and objects in Obj-OoD, are all unseen. Our results, see Figures 5 (bottom right) and A.2, show that this setting is indeed more challenging than either Vis-OoD or Obj-OoD alone. As all methods struggle in the challenging setting there is still significant room for novel approaches to be designed to solve this task.

A.1.4 Ablating Measures of Irreversibility on Stretch-P&P

We now show that our measurement-determined irreversibility intervention method is relatively robust to the selection of diversity measure (recall the discussion in Sec. 4.1). To illustrate this, we used two classes of measures: (1) dispersion-based (ENT, STD) and (2) distance-based (L2, DTW). We provide details and formulations for our proposed measurements in Appendix C.1. As shown in Fig. A.2, no single measure outperforms the others across all evaluation settings. Note that objects occupy a diverse range of states during training, especially when the apple is dropped from the hand in the air. We visualize 10k such steps sampled through training in 3D space in Fig. A.3. The visualization shows that near-irreversible states, and not just explicitly irreversible states (*e.g*. the apple falling from the table), bottleneck the autonomous training process. As our experiments show, our proposed irreversibility measures are able to characterize the near-irreversible states and allow for interventions, namely resets, to be taken effectively.

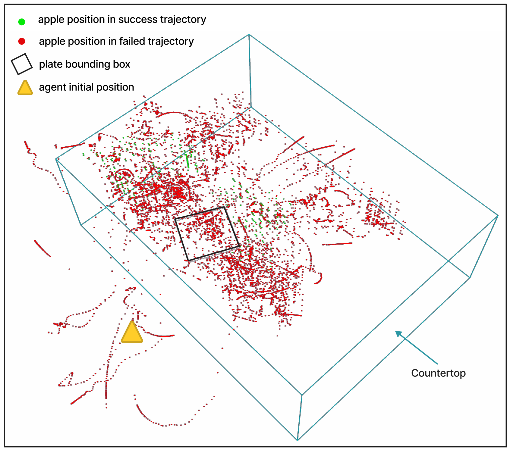

Figure A.3: **Visualizing successful and failed object trajectories in STRETCH-P&P during training.** Notice that the object occupies many diverse states and can fall off of the table or roll away from the agent.

A.2. Additional Sawyer-Peg Details and Analysis

We now provide additional results and analysis for our experiments within the Sawyer-Peg environment.

A.2.1 Experiments with RGB Observations

Unlike previous work with only experiments on low-dimensional observations, we extend the task into the visual domain and found our method could easily solve the problem and make good generalizations. We show our experiment results at Fig. 9. We compare our method (which uses random targets and measurement-determined resets) with two categories of baselines: (1) random targets with the periodic resets (1e4), and (2) FB-RL with the periodic resets as well as the ground-truth resets if the peg is dropped off the table. Similarly as the results for STRETCH-P&P, our method achieves significantly more efficient training in terms of training steps and total resets. When evaluating on novel box goals, *i.e*. where the target hole of the peg box has been randomized relative to the box (see Fig. 2 and Sec. B.2 for details), we find that the FB-RL agents cannot generalize as well as agents trained with random targets. Therefore, the our proposed approach both makes training more efficient (by providing resets when necessary) and allows for more significant generalization (through the use of random targets during training).

A.2.2 Ablating Measures of Irreversibility on Sawyer-Peg

Similarly as in Section A.1.4, we provide additional ablations in the Sawyer-Peg environment showing the relative performance of baselines trained using different irreversibility measures in Fig. A.4. Note that **all** of our proposed baselines achieve consistently high performance in the training, in-domain evaluation, and novel box evaluation. Moreover, they achieve this high performance within **≈100 resets** in total and converge in around **1M steps**. All experiments are run with three random seeds and shaded areas represent the min/max range across different seeds. Note that previous benchmark EARL [44] shows almost 0 success in a variant of this task using FB-RL with periodic resets over 3 to 5 million steps even when: (1) the evaluation environment is identical to that during training, (2) low-dimensional observations containing the true position of the peg are given to the agent, and (3) table boundaries are present to prevent the peg from dropping from the table. More recent work introduced from [57] use a much narrower, boundaryless, table for this task and achieve approximately 60% success in 3M steps with 120 hard resets on average. We reproduce the result at Fig. A.7.

A.2.3 Existence of Near-Irreversible States in Sawyer-Peg

Here we provide further evidence suggesting that prior, reset-free, works' relatively low performance in the Sawyer-Peg task is largely due to the existence of near-irreversible states.

RF Fine-tuning Failure To this end, we train an episodic agent in the random targets setting and then fine-tune this agent **in the same setting and environment** but with using much less frequent resets and a small learning rate. Note that, by the end of the initial stage of training (*i.e*. before finetuning) the agent achieves a near 100% success rate. Beyond periodic resets, during finetuning we provide additional resets immediately after the peg dropped off the table. We provide these resets so we do not conflate the impact of “true“ irreversible states with near-irreversible states. As shown in Fig A.5, where blue curve shows the success rate,

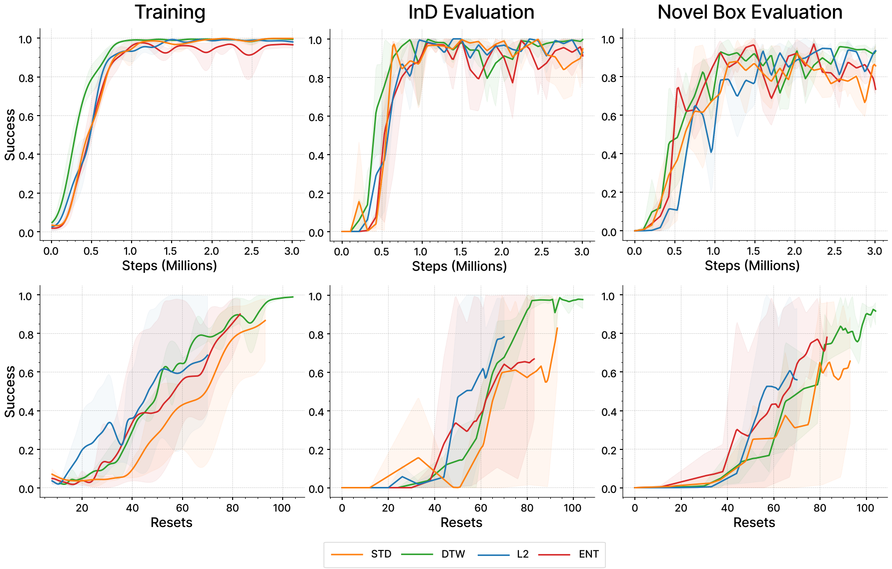

Figure A.4: **Irreversibility measure ablation within the Sawyer-Peg environment.**

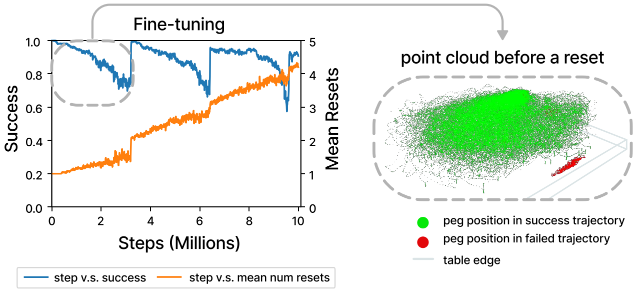

Figure A.5: **Finetuning a Sawyer-Peg agent in a low-reset setting.** A well-trained Sawyer-Peg agent is finetuned in the same environment but with periodic resets happening much less frequently than in the episodic setting (approximately every 3.5M steps). Note that, beyond periodic resets, we reset the environment whenever the agent drops the peg off of the table so the mean number of resets per process will steadily increase during training. Agent performance appears to steadily decrease until a periodic reset occurs after which it returns to a success rate near 100%. Note that periodic resets occur across all processes simultaneously.

and the orange curve shows the number of resets *on average* for each process during fine-tuning, the agent achieves a near 100% success rate at the start of fine-tuning. However, this success rate *continuously decreases* until a periodic resets occurs after $4M$ training steps after which the success rate suddenly recovers back to near 100%. Note that we have already provided "oracle" resets after peg dropped to exclude the "true" irreversible states and yet, the agent's success rate still decreases. This suggests that the agent manages to enter environment states where the peg is still on the table but from which obtaining success is difficult, *i.e*. the agent enters near-irreversible states. To provide a qualitative visualization of where the agent fails during finetuning, we track all positions of the peg in one environment before the first reset and label them red and green for failed and successful trajectories respectively (see Fig A.5). As showing in the 3D point cloud on the right, the success and failure has a clear dividing line near the edge of the table. Thanks to the random targets training setting, the agent can succeed from a diverse range of states but those states near the edge of the table appear to be very difficult (perhaps due to physical limitations on the degrees of freedom of the arm) and thereby represent a set of near-irreversible states.

Evaluation with Narrower Table Besides, we provide further evaluations on the narrower table use in [57]. We surprisingly found that the result is exactly the same as evaluating with the normal-sized table. We illustrate the point clouds of successful and failed trajectories evaluated intermediately and finally in Fig. A.6, where red points indicate the object (peg) head positions for failed trajectories while greens show the successful ones. Evaluations on two settings have the exact the same performance and rollouts trajectories for the final checkpoint (last two figures), but the same policy is leading to different consequences during the evaluation at 300k steps (first two figures): the peg always drops off the narrower table but mostly is still on the edge of the normal-size table. We consider the second case as near-irreversible where we are not able to expect the agent to grasp the peg back given the current imperfect policy.

The above results motivate a need for the measures of irreversibility we introduce in this work: it is a *practical im-*

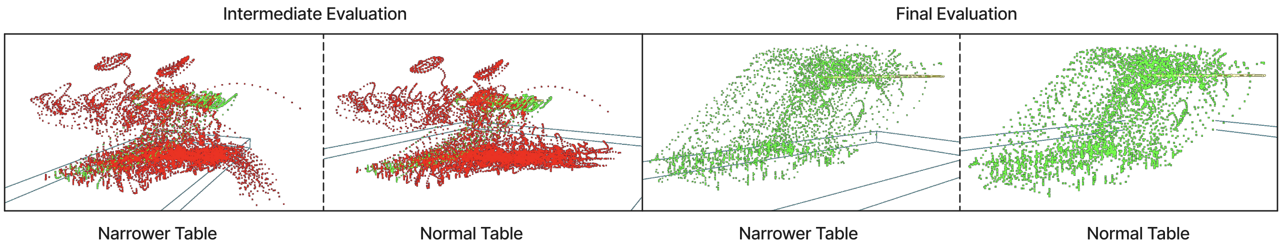

Figure A.6: Point cloud visualizations for evaluations on the narrower (same as [57]) and the normal-sized table, where red points indicate the object (peg) head positions for failed trajectories while greens show the successful ones. Evaluations on two settings have the exact the same performance and rollouts trajectories for the final checkpoint (last two figures), but the same policy is leading to different consequences during the evaluation at 300k steps (first two figures): the peg always drops off the narrower table but mostly is still on the edge of the normal-size table.

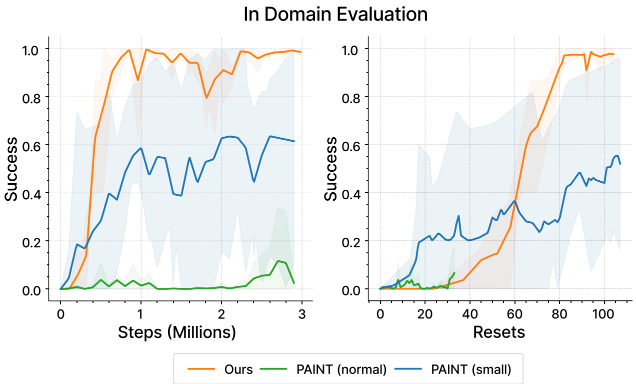

Figure A.7: **Evaluation with supervised method PAINT [57] trained in narrower table and the normal size table (low-dim)**. The significant performance decrease shows for training with the normal size table, mainly due to the false negative labels of near-irreversible states used for training classifiers in PAINT. Due to high oscillations, PAINT experiments are trained with five random seeds and all methods are evaluated with 200 tasks and shadowed area represent the min/max range

possibility to label all (near-) irreversible states before training.

B. Environment Details.

Besides the introduction in Sec. 3 and Appendix. A.1, we further provide low-level details. Code made available at https://zcczhang.github.io/rmrl.

B.1. STRETCH-P&P

Observations. In experiments of STRETCH-P&P, the observation space consists of the visual observation from the wrist-centric camera with dimension 224×224 and field of view (FOV) of 75, the text prompts, and the low-dimensional observation including the egocentric coordinates of the Stretch gripper and the target (4-dimensional in total). The example is show in Fig. C.4. The language instructions are in the format of ”`Put {obj_1} into {obj_2}.`” for object goal and ”`Put {obj_1} into` (r, θ) `.`” for point goal in the polar form r, θ. In STRETCH-P&P, the target task can be further decomposed as the picking task where the prompt is only ”`Pick {obj_1}.`” if there is no object in agent’s hand.

Action Space. The action space is discrete and consists of 10 actions. The robot base has forward and backward movement in parallel to a side of the table along x axis relatively (see Fig. 3). The robot arm can lift or lower along the relative y axis (in AI2-Thor, the y-axis corresponds to the height, which is usually the vertical z-axis in Cartesian coordinate space); the arm can also extend or retract horizontally to move the gripper further or closer relative to the agent base in the z-axis; and the gripper can rotate in positive and negative yaw directions. For object interactions, we have a `PickUp` action where only the specified object with a unique object ID and within a sphere with radius r centered at the gripper (similar to ManipulaTHOR [10]) will be grasped with its previous position and orientation unchanged in hand. The release action simply drops the object if there is one in hand, and we add simulation steps to wait until the object stabilizes for a realistic setting. Table B.1 provides details about the action names, descriptions, and scales used in STRETCH-P&P. For comparison, we also provide the action space for the RoboTHOR object navigation task in Table B.2.

Success Criteria. In STRETCH-P&P tasks, success is determined based on two criteria. Firstly, the picking object’s bounding box should intersect with the receptacle trigger box (which is different from the bounding box and only includes the area of the receptacle, *e.g.* internal rectangular area of a pan without the handle) when both objects static. Secondly, the distance between the picking object and the center of the receptacle trigger box must be within a threshold to avoid edge cases or large receptacles. In the case of

Action	**Description**	**Scale**
`MoveAhead`	Move robot base in $+x$ axis	5 cm
`MoveBack`	Move robot base in $-x$ axis	5 cm
`MoveArmHeightP`	Increase the arm height $(+y)$	5 cm
`MoveArmHeightM`	Decrease the arm height $(-y)$	5 cm
`MoveArmP`	Extend the arm horizontally $(+z)$	5 cm
`MoveArmM`	Retract the arm horizontally $(-z)$	5 cm
`MoveWristP`	Rotate the gripper in $+x$ yaw direction	2°
`MoveWristM`	Rotate the gripper in $-x$ yaw direction	2°
`PickUp(object_id)`	Pick up object with specified unique `object_id` if object within the sphere with radius r centered at gripper	$r = 0.06$
`Release`	Release object with simulation steps until object is relatively stable	–

Table B.1: **Action space in STRETCH-P&P.** All directions are relative to the Stretch robot.

Action	**Description**	**Scale**
`MoveAhead`	Move robot base in $+x$ axis	0.25 m
`MoveBack`	Move robot base in $-x$ axis	0.25 m
`RotateLeft`	Rotate the robot base leftward (yaw)	30°
`RotateRight`	Rotate the robot base rightward (yaw)	30°
`LookUp`	Rotate the camera upward (pitch)	30°
`LookDown`	Rotate the camera downward (pitch)	30°
`End`	Terminate the episode. Success if the goal object within d visibility.	$d = 1$

Table B.2: **Action space in RoboThor ObjectNav.** All directions are relative to the LoCoBot.

random targets or point goals, only the second criterion is used.

Objects Partitions. We illustrate the objects to be picking household objects and target receptacles for each object budget level and unseen tests (OBJ-OOD, ALL-OOD) in Table. B.3.

Levels	**Picking object**	**Receptacle**
Budget $= 1$	red apple	striped plate
Budget $= 2$	red apple, bread	striped plate, baking pan
Budget $= 4$	red apple, bread, sponge, blue cube	striped plate, baking pan, metal bowl, saucer
Unseen objects	green apple, tomato, potato, mug	wooden bowl, bamboo box, rusty pan, pot

Table B.3: **Objects partitions for different object budget levels and unseen tests in STRETCH-P&P.**

Texture and materials randomizations. Further, for texture and lightning randomization evaluations, we randomize the RGB color of materials in range $[0, 0, 0]$ to $[255, 255, 255]$ and over 5 different materials for the table-top (table surface and legs are randomized separately), surrounding walls, and floor. The color of the lights are randomized similarly and the intensity is randomly sampled from $[0.5, 2]$, where during training the light intensity is 1.

Unseen room structures. During the training, there is no extra furniture other than the table in the room, where during the ALL-OOD evaluation, we added fridges, shelves, flowers, *etc.* as distractors in the background of agent's view. Visualizations can be found in Fig. 2

B.2. Sawyer Peg

Description. We use the same Sawyer Peg simulation environment proposed in [60, 44]. This task requires the robot arm to grasp the peg and then insert it into a hole of the box. The success criteria is defined by $||s-g|| \leq \epsilon$ where s, g denotes the state and goal, and ϵ is the small tolerance. We use the distance between the position of the peg and the center of the hole with $\epsilon = 0.05$ for all Sawyer Peg experiments same as [60, 44]. Action space consists of 3D end-effector delta control and 1D gripper open/close control. For visual task, the visual input includes the wrist-centric camera view and the third-person view with 84×84 dimension for each, adapted from [18]. We also include the 3D end-effector position relative to the robot base, 1D gripper width, and randomly sampled goal as low-dimensional observations. No ground-truth position of the object peg is exposed to the agent.

Training and Evaluation Protocol Recall Fig. 9 and Sec. A.2.1. Agents are trained in the usual Sawyer-Peg setting (with one box and hole position) and then tested with both novel hole locations and with novel box positions *and* novel hole positions. Visualizations of the novel box hole can be found at Fig. 2, where the hole is bounded with the green square for demonstration purpose, but will not be in agent's visual observations.

Simulation Stability. We found that when the simulation is not reset over a long time horizon, there are unexpected aberrant outcomes in the Sawyer Peg environment[3]. Specifically, after collisions with the gripper or the peg box, the target peg object may no longer be capable of laying flat along the table as expected. To address this issue we regularly reset the position of the peg to a "flat" state without otherwise changing the position of the agent's arm or the peg. This reset is different than those used when training episodic agents and is only done to maintain simulator stability when not executing a full environment reset for many timesteps. We also encountered floating errors and collisions, which are expected and, in some cases, inevitable. Most of these issues occurred on a reasonably small scale (i.e., less than 1e$-$7) and can be considered as noise. However, some collisions with the gripper and the stationary peg box were more significant and observable. We note that these collisions can lead to (near-)irreversible states which may be challenging, or even impossible, to recover from. The existence of these issues within a simulated environment emphasizes the need for methods, like those proposed in this work, that automatically allow for detecting near-irreversible transitions even when those transitions are impossible to anticipate a priori.

C. Implementation Details

C.1. Measures of Irreversibility

NI Transition. Here we describe more formally what we mean by a near-irreversible (NI) transition. As usual in reinforcement learning for embodied agents, we formalize our setting as a Partially Observed Markov Decision Process (POMDP) with state space $\mathcal{S}$, partial observation space $\mathcal{O}$, action space $\mathcal{A}$, transition probability measure P_T, and reward structure R. For simplicity of presentation we will assume that $\mathcal{S}$ and $\mathcal{A}$ are discrete. The goal is to learn a policy π, *i.e.*, a function that maps partial observations to distributions over actions, which maximizes the expected future γ-discounted rewards ($\gamma \in [0, 1]$). We will say that there is a *path* from $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ if there exists some $n{\geq}0$, $s_1, ..., s_{n-1} \in \mathcal{S}$, and $a_0, ..., a_{n-1} \in \mathcal{A}$ such that $P_T(s_0, a_0, s_1), ..., P_T(s_{n-1}, a_{n-1}, s_n) > 0$ with $s_0 = s$ and $s_n = s'$. We say that s, s' are connected if there exist paths from s to s' and from s' to s. Similarly we will say that a set $S \subset \mathcal{S}$ is connected if all $s, s' \in A$ are connected. Note that connectedness an equivalence relationship and thus partitions $\mathcal{S}$ into equivalence classes, we call these classes *connected components*.

We assume that, during training, we never give the agent a goal that requires it to undergo an irreversible transition. More formally, if we let $\mathcal{G} \subset \mathcal{S}$ be the set of goal states and $\mathcal{I} \subset \mathcal{S}$ be the set of all possible states to which the agent may be reset, then we assume that $\mathcal{G} {\cup} \mathcal{I}$ is connected. Finally we are in a place to define what we mean by an irreversible transition. Let $\tau_\pi(i) \in \mathcal{S}$ for $0 \leq i$ be a random variable representing the trajectory of an agent with policy π where $\tau_\pi(0)$ represents the start state of the agent after a reset (*i.e.*, $\tau_\pi(0) \in \mathcal{I}$). Moreover, let $\mathcal{U} \subset \mathcal{S}$ be the connected component containing $\mathcal{I} {\cup} \mathcal{G}$. Then we say that the agent has undergone an *irreversible transition* at step i if $\tau_\pi(i) \in \mathcal{U}$ and $\tau_\pi(i+1) \notin \mathcal{U}$. We will also call every state $s \in \mathcal{S} \setminus \mathcal{U}$ an *irreversible state*. See Fig. 3 for examples of reversible and irreversible states in STRETCH-P&P.

While this definition of irreversibility reflects the capabilities of the environment, it falls short in two ways. First, during training, we should not care if a state s is truly irreversible, we instead need to consider whether or not the agent's current policy π has any hope of ever moving from s to a goal as, otherwise, it cannot hope to succeed. Secondly, with our current definitions, a state s is considered reversible even when the probability of the agent ever reaching a state in $\mathcal{I} {\cup} \mathcal{G}$ from s is arbitrarily small. For instance, in Fig. 3 we see examples of what we call "near-irreverable" states where the agent must execute a long sequence of precise steps to retrieve the object. To formalize, we will say that a state s is (π, ϵ, N) *near-irreversible* if, when following π, the probability of reaching a goal state from s within N steps is $< \epsilon$.

Computing precisely whether or not a state s is (π, ϵ, N) near-irreversible, or even irreversible, is computationally intractable even with full knowledge of the environment. Instead, we use the above formalization to build intuition for the behavior we expect to see when an agent has experienced a near-irreversible transition.

Measures of reversibility. We now introduce the details of our reversibility measures. We propose two domains of measures: dispersion-based and distance-based. Intuitively, the dispersion-based approaches measure the dynamics of the trajectories with metrics like variation/standard deviations or entropy, where the distance-based methods measure measure the distance from recent states to previous states to characterize how far and how often the agent visits new states or whether it is trapped within a few states.

Dispersion-based measures. We illustrated the dispersion-based method in Algorithm. 1. The standard deviation and entropy metrics are used as the baseline for our dispersion-based measurements. To elaborate, we denote the policy π, dispersion metrics φ, threshold ϵ, trajectory buffer τ, and horizon for measurement check N. Then, we simply define the dispersion measure over the trajectory $\tau = \{s_0, s_1, \cdots, s_N\}$ sampled from on-policy π as $\varphi(\tau)$. And once $\varphi(\tau)$ consistently smaller than the threshold ϵ,

[3] https://github.com/Farama-Foundation/Metaworld/issues/373

then we say the current measure stage as (π, ϵ, N) is near-irreversible (NI). We directly use $\varphi(\tau) = std(\tau)$ for our method STD. And we calculate the entropy over a discretized state space as our ENT method. In practice, to characterize the consistency of near-irreversible behaviors, we introduce the hyperparameters n_{tol} for the number of consecutive phases that the measurement is considered near-irreversible states. In all of our experiments, we use the $N = 300$ and $n_{tol} = 2$.

```
Algorithm 1: Dispersion-Based Measure
Init: policy $\pi$
Init: $\varphi, \epsilon, \tau, N$, optional $n_{tol}$ and $n_{irr} = 0$
while not done do
    # randomly sample goal from the goal space
    sample $g \in \mathcal{G}$;
    rollouts in environment with $\pi$ and update $\tau$;
    # near-irreversible formulation in Appendix. C.1
    $NI(\pi, \epsilon, N) =$ False;
    if len($\tau$) $\geq N$ then
        # dispersion mesure
        $\rho = \varphi(\tau)$ ;
        if $\rho < \epsilon$ then
            $n_{irr} += 1$;
            # consecutive near-irreversible check
            if $n_{irr} \leq n_{tol}$ then
                $NI(\pi, \epsilon, N) =$ True;
                $n_{irr} = 0$;
            end
        else
            $n_{irr} = 0$;
        end
        clear $\tau$;
    end
    update $\pi$ with PPO [40];
end
```

Distance-based measures. Similarly as the dispersion-based measures, our distance-based measures are also calculated over the trajectory. We illustrated the distance-based method in Algorithm. 2. We propose L2 for euclidean distance metric, and DTW for dynamic time wrapping metric as d in the algorithm. The measure is calculated as the distance from a sliding window of the trajectory with total length M, to the previous histories, which is also a sliding window with the maximum length $N - 2M$. Besides the traditional euclidean distance metrics, our setting is intuitively and naturally allows for using a DTW measures as the distance measure M, trajectory horizon N, and sliding windows have different length, makes the DTW efficiently measure the distance between states. We use $M = 100, N = 600$ (which shares the same history length as dispersion-based method but without the consecutive phases check) for all experiments in all simulations.

```
Algorithm 2: Distance-Based Measure
Init: policy $\pi$
Init: $d, \epsilon, \tau, N$, measure steps $M < N/2$
while not done do
    # randomly sample goal from the goal space
    sample $g \in \mathcal{G}$;
    rollouts in environment with $\pi$ and update $\tau$;
    # near-irreversible formulation in Appendix. C.1
    $NI(\pi, \epsilon, N) =$ False;
    if len($\tau$) $\geq N$ then
        # distance from slide-window to past
        distances = [] ;
        for $i \leftarrow 0$ to $M$ do
            $s = \tau[N - M + i]$;
            $past = \tau[: N - 2M + i]$;
            $d_{min} = \min(d(s, past))$;
            Add $d_{min}$ to distances;
        end
        $d_{maxmin} = \max$(distances);
        if $d_{maxmin} < \epsilon$ then
            $NI(\pi, \epsilon, N) =$ True;
        end
        clear $\tau$;
    end
    update $\pi$ with PPO [40];
end
```

C.2. Irreversibility Threshold

In Fig. C.8, green and blue curves used the same periodic reset setting (resetting every 1e4) steps but, due to randomness in RL training, converge at different timesteps. The orange curve (ours) learns faster at the beginning of the training and the red curve (1e5 steps/reset) learns little in 5M steps. We found that both our dispersion-based and distance-based measures effective characterize the learning process and capture those timesteps where periodic resets occur (measures of reversibility become large after resets). In particular, those increases in the reversibility measures capture that the agent can attempt to reach the peg after a reset but, but (as shown the the gradual decline of the reversibility measures) unfortunately pushes the peg into a position that is difficult to reach due to the sub-optimal policy early in training. Selecting a threshold (recall the α parameter from Sec. 4.1 of the main paper) for deciding when to reset based on the reversibility measure is trivial: we select it as some value between the those values obtained by a random policy soon after a reset and those values obtained by doing nothing. Empirically we found that reasonable

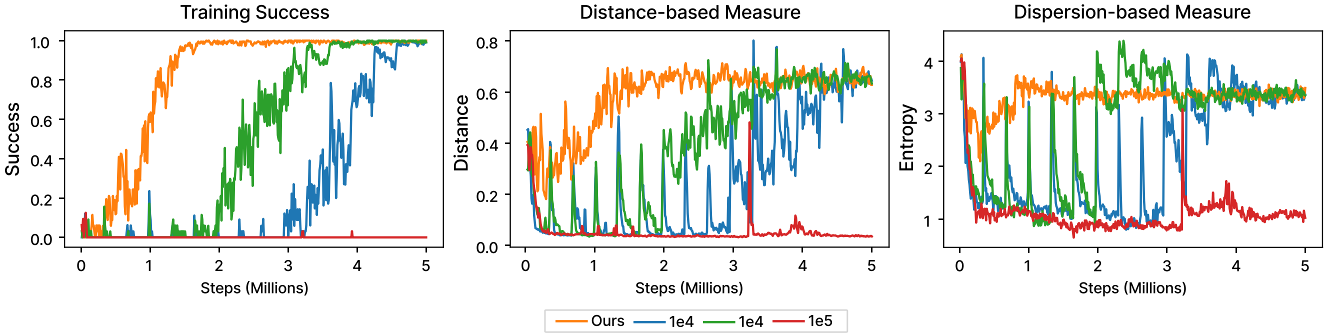

Figure C.8: **Visualizing reversibility measures during Sawyer-Peg training.** Notice that the periodic methods (1e4-green, 1e4-blue, and 1e5-red) methods have low reversibility values during the majority of training (right two figures) with periodic jumps in these values. These periodic jumps correspond to resets and suggest that these agents spend a large portion of their training time in near-irreversible states waiting for a reset to begin learning again. Unlike these methods, our baseline has a consistently high reversibility values across training (as the agent will reset when such values become too low) and learns quickly. For demonstration purpose, all models here are trained using 32 parallel processes.

thresholds are also robust and can be adjusted to fit one's preferences for reset frequency: with a smaller value, we may expect the agent to require fewer resets but a larger number of total training steps to solve the task, and vice-versa. The horizon for irreversible check is even more robust in general. We consistently use 600 steps across all experiments (which corresponds to twice the maximum number of steps in a training episode). Meanwhile, instead of learning being hampered primarily by the challenge of exploration (as argued in [44]), we claim that the main challenge for autonomous RL is actually the existence of near-irreversible states: the agent is able to explore but only so long as it hasn't entered such a state.

```
def episodic_reset():
    reset_environment(
        agent_initial_state,
        arm_initial_state,
        object_initial_states
    )
    # similar start every episode
    agent_start = agent_initial_state
    arm_start = arm_initail_state
    object_start = object_initial_states
    goal = target
    return get_obs()
```

Pseudocode 1: Reset function in episodic setting

C.3. Algorithms

To clearly distinguish between different training methods for traditional episodic or autonomous RL, we now provide pseudocode for the reset function in episodic RL (Pseudocode. 1), our single policy with measurement-determined resets with random goals (Pseudocode. 2), and previous FB-RL with periodic and supervised explicit irreversible interventions (Psudocode. 3). In our experiments, for the FB-RL+GT baseline, we simply replace irreversibility prediction network proposed in [57] with ground-truth values, this should represent an upper bound on the performance of this method. Nevertheless, as described in previous sections, we found that the FB-RL+GT method underperformed our proposed approach, likely due to the existence of near-irreversible states that cannot be easily captured as GT labels.

```
def reset():
    if measure_check_near_irreversible():
        # measurement-led intervention
        return episodic_reset()
    else:
        # continued from previous phase
        agent_start = agent_prev_state
        arm_start = arm_prev_state
        object_start = object_prev_state
        # any state exclude object state
        # prevent success immediately
        goal = random.sample(
            goal_space \ object_start
        )
        return get_obs()
```

Pseudocode 2: Reset function in our random targets with measurement-lead intervention setting

```
def reset():
    # switch phases
    phase = 1 - phase
    if cumulative_steps >= reset_frequency:
        # periodic intervention
        return episodic_reset()
    elif is_explicit_irreversible():
        # Either supervised or ground truth
        # by `object_height < table_height`
        return episodic_reset()
    else:
        # continued from previous phase
        agent_start = agent_prev_state
        arm_start = arm_prev_state
        object_start = object_prev_state
        if phase == 0:
            # forwad phase
            goal = target
        else:
            # backward phase
            goal = random.sample(
                initial_states
            )
        return get_obs()
```

Pseudocode 3: Reset function in previous FB-RL with periodic intervention setting.

With goal-conditioned POMDP defined in Sec. 4.2, in FB-RL, in FB-RL setting, the forward goal space is normally defined as the singleton $\mathcal{G}_f = \{g^\star\}$ for the target task goal $g^\star$ (*e.g*. the apple is on the plate, the peg is inserted into the hole, etc), where the goal space for backward phase is exactly the (limited set of) initial state space $\mathcal{G}_b = \mathcal{I} \subset \mathcal{S}$. Empirically, those initial states are predefined and discriminated in previous work such that the deployed $\mathcal{G}_f$ and $\mathcal{G}_b$ are disjoint. Then it is reasonable to use separate policies and objectives to optimize alternatively during the training. However, as in our setting, by simply given $\mathcal{G} = \mathcal{S} \setminus \{s_t \mid s_t \in \tau_\pi(t) \in \mathcal{T}_\pi, \Phi_{W,N,\alpha,d,\pi}(\mathcal{T}_\pi) = 1\}$ which is represented as the entire state space excluding the NI states we formalized above given the current π during training, we can make the analogue of each *phase* in our autonomous RM-RL setting and using the same objective as an episode noted in episodic RL, where by denoting a *phase*$_i$ with states trajectory $\{s_0^i, \cdots, s_H^i\}$ as an i^{th} episode with state $s_t^i \in \mathcal{S}$ at timestep t, finite horizon H_i, and goal $g_i \in \mathcal{G}$ such that $s_H^i = s_0^{i+1}$, *i.e*. the last state of the i^{th} phase is identical to the initial state of the $(i+1)^{th}$ phase. Therefore, we can directly use the same objective with switching goals without a reset, where a reset thereby can be simply defined as an intervention such that $s_H^i \neq s_0^{i+1}$. Empirically, we define a reset or an intervention in embodied tasks at the beginning of the episode or phase, entirely or partially from 1) teleport the agent, 2) teleport the arm, and/or 3) put objects in environment to some initial configurations. Unlike some previous reset-free work, we still want to highlight that even a hard-code teleportation without human intervention in real world or simulation is also restrictively counted as a reset in our work.

Except the goal updates, which can be done automatically, all other resets require human intervention (including the hard-coded programming for agent or arm teleportation in the real world or simulation) in practice. As shown in Pseudocode. 2, by eliminating the "reset" gameplay from prior work, our method is similar to the episodic setting except that the start state of an phase is exactly the same as the last state of previous phase. Note that the goal space for each phase is symmetric by sampling randomly (exclude the state that the object is exactly at the goal, *i.e*. succeed immediately)

C.4. Model Architecture

Baseline Model for STRETCH-P&P As show in Fig. 4, our baseline model conditioned on visual observations, language instruction, and low-dimensional proprioception and goal state. Specifically, we used the frozen CLIP ResNet50 as the visual backbone followed with a CNN compressor consisting two convolutional layers with 128 and 32 channels respectively. Each convolutional layer had a kernel size of 1×1 and was activated with ReLU. In addition, we discarded the final average pooling and linear layers from the CLIP ResNet50, and only kept the last spatial map before the trainable CNN compressor. And we replace the batch norm with group norm. For the language stream, tokenized texts are passed to the frozen CLIP language encoder and projected with one 1024 linear layer and expanded as the same dimension of the output spatial map like [46], and passed to a CNN combiner with the output of the CNN compression from the visual stream. The CNN combiner has the same channels and kernels as the compressor. Finally, for the low-dimensional point observation, we simply encode it with a linear layer followed with the layer normalization and fuse it with the vision and language embedding and then pass to the actor and critic for on-policy updates. The actor and critic are defined as linear layers with output dimensions of 10 and 1 respectively.

Sawyer Peg For a fair comparison, we modify our model in Fig. 4 by replacing the pretrained vision and language backbone with the CNN backbone from scratch and with random crop and shift augmentations like [18, 42]. However, unlike [18, 42] which use separate vision encoders, we use only *single* CNN encoder (just like our idea of single policy) that digest both views of image for parameter-efficiency. We ended with finding that single visual encoder is enough for solving this task. The low-dimensional observation is encoded as the same way as for STRETCH-P&P. The output is then fused with the visual representation and passed into actor and critic networks with two hidden layers of 512 and 256 unit each. We further follow the practice from [50, 3] that discretizes the continuous action space into multi-discrete action space with 7 bins of each dimension (*i.e.* $4 \times 7 = 28$ in total) and use multi-categorical distribution for our policy. We find slightly better empirically in this task and efficient than using parameterized scale for Gaussian distribution.

ObjectNav Same as our baseline model but without the frozen language backbone and include extra GRU state encoder for consistent comparisons with previous work [20].

C.5. Reward Shaping

Instead of costly process of collecting demonstrations (even from the oracle agent trained in episodic settings), for *all* autonomous training in STRETCH-P&P, Sawyer Peg, and ObjectNav, we adopt a *unified* distance-difference-based reward R given state transition $s_{t-1}, s_t \in \mathcal{S}$ and goal $g \in \mathcal{G}$ as:

$$R(s_t|g) := \alpha \cdot (d(s_{t-1}, g) - d(s_t, g)) + \beta \cdot \mathbb{1}_{success}$$

where d is the non-negative distance function, $\mathbb{1}_{success}$ is the indicator function for the task success, and α, β are weighted multipliers. We simply set $\alpha = 1, \beta = 10$ for experiments except in Sawyer Peg where $\alpha = 100$ due to its smaller state space. The first term incentives the agent to continuously make progress towards the goal while the second terms is the terminal reward for completing the task. This reward formalism is first proposed in [31] and widely used in navigation tasks. Recently it has also been used as an implicit reward for goal-conditioned pretraining in embedding space [24, 26, 28]. In practice, we calculate the Euclidean distance from the gripper to the object and from the object to the target for manipulation experiments, and the distance from the agent to the goal object for object navigation experiments. Note that neither the object positions nor the agent base GPS positions is included in the observations during the training and inference. We believe that a *single* reward shaping that can be easily applied to tasks in different embodied space does not require tedious engineering, and it is applicable and practical for the general formulation of the autonomous RL.

C.6. Training Hyperparameters

To train the policy with RL, we use PPO with Generalized Advantage Estimation (GAE) and normalized advantages [39]. For paralleling training, we adopt DD-PPO in AllenAct [56] with 1 worker on each GPU. We detail the shared (left) and different (right) default hyperparameters for training in each task domain.

D. Initial Results in Reset-Minimization for Object Navigation

To highlight the general applicability of our proposed RM-RL approach for embodied AI tasks, we show preliminary results on the RoboTHOR Object Navigation (ObjectNav) task [6]. In ObjectNav, an agent is placed within a simulated household environment and given a goal object category (*i.e.* TV, chair, table, *etc.*), see Fig. D.9. The agent must then navigate to an object of this category using only visual observations. The agent is considered to have successfully completed the task if it executes a special END action and an object of the given category is both visible and within 1m of the agent. Regardless of whether or not the success criteria are met, the END action always ends the episode. Note that, in principle, there are no irreversible states in ObjectNav: the task merely involves navigating around an, otherwise static, environment. Nevertheless, as we show below, choosing when to reset carefully can result in significant improvements in efficiency.

D.1. Training and Evaluation Protocol

We extend the RoboTHOR ObjectNav training code available in AllenAct [56]. This training code contains the EmbCLIP model introduced by Khandelwal *et al.* [20] which is SoTA among models trained purely upon the RoboTHOR ObjectNav dataset (there exist more performant models, *e.g.* those pretrained upon ProcTHOR [7], but these use extensive additional pretraining data in the form of procedurally generated environments). The training set contains 60 different house plans with 12 goal object types. We use 60 separate training processes for all experiments.

In order to train in a reset-minimizing setting, we create a variant of the RoboTHOR ObjectNav task where the agent is not automatically reset in the usual episodic fashion. In particular, we turn the END into a soft version where, if the agent executes the END action without success criteria being met, then the agent is penalized and the goal object remains the same. If success criteria are met or reach 300 steps for the current goal, however, then the agent is given a new goal object category within the same environment. As usual, we allow the agent to request a full environment reset and we use our usual measures of irreversibility to determine when these resets should be executed. We call training variant RM-ObjectNav.

All models are evaluated on the RoboTHOR validation set as usual, *i.e.* with an END action that immediately ends the episode regardless of success. In particular, the agent is evaluated with 1800 tasks from 15 unseen houses but with the same target object types as training.

D.2. Results

We show our training results in Fig. 6 and evaluation results in Table 1. In particular, we show validation set performance after 50M and 100M training steps. Our baselines include four models trained in the RM-ObjectNav setting: (Ours) an EmbCLIP baseline agent with resets taken when using our STD irreversibility measure and ($N = 300$, $N = 10k$, $N = \infty$) three models with periodic resets taken after every 300, 10k, and ∞ steps. We also include a [20]

Hyperparameter	Value
GPU instance	`g4dn.12xlarge`
Worker per GPU	1
Optimizer	Adam [21]
Discount factor γ	0.99
GAE τ	0.95
Value loss coefficient	0.5
Normalized advantages	True
Max gradient norm	0.5
Entropy coefficient	0.01

Hyperparameter/Value	Stretch-P&P	Sawyer Peg	ObjectNav
Number of GPUs	2	2	4
Environments per GPU	8	4	15
Learning rate	3e-4	5e-4	3e-4
Rollout length	200	1024	128
PPO epochs	10	20	4
Number of mini-batches	4	1	1
PPO Clip	0.1	0.2	0.1

Table C.4: Shared (left) and separate (right) hyperparameters for Stretch-P&P, Sawyer Peg, and ObjectNav experiments

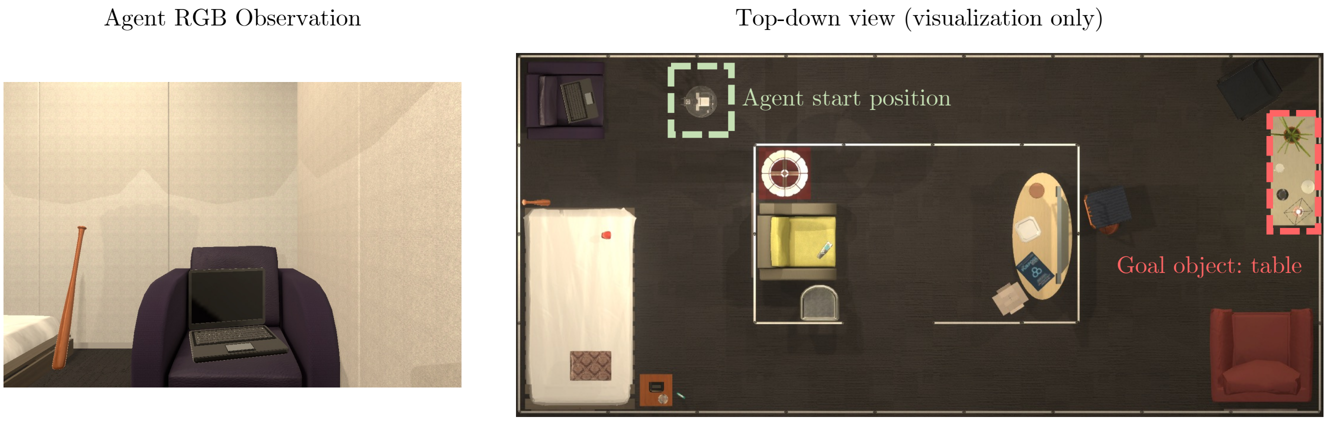

Figure D.9: **RoboTHOR Object Navigation Task.** An example of a the start of a RoboTHOR ObjectNav episode. The agent is given an RGB observation (left) and must navigate to an object of a goal category (table). On the right we show a top down map view of the environment with the agent's position and goal object highlighted. Note that this top down view is for visualization only and is not available to the agent at training or inference time.

baseline trained in the usual ObjectNav setting (*i.e.* without the "soft" End action). To emphasize: all models are evaluated with hard End actions.

The results from Table 1 are very promising: after 100M steps with only 635 resets we are able to achieve success rates **higher** than all competing baselines despite the next best performing baseline using 2M resets. Note also that our agent takes the vast majority (592 of 635) of its resets within the first 50M steps of training showing that our model continues to learn (going from a success rate of 0.216 at 50M training steps to 0.551 at 100M training steps) using very few resets. Recall that, technically, there are no explicitly irreversible states in ObjectNav. Nevertheless, we show that pure Reset-Free training results in slower training convergence and lower evaluation results. In comparison with other periodic reset competitors trained with the same soft End action, we show that all of them provide lower evaluation results than ours and EmbClip, mainly due to the false positive that early stopped during the evaluations. Therefore, we believe providing a very few resets at proper time-point determined by our unsupervised method will give more efficient performance in general.