Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning

Abhishek Gupta[*]
Department of EECS
University of California Berkeley
United States
abhigupta@berkeley.edu

Vikash Kumar
Google Brain
United States
vikashplus@gmail.com

Corey Lynch
Google Brain
United States
coreylynch@google.com

Sergey Levine
Department of EECS
University of California Berkeley
United States
svlevine@eecs.berkeley.edu

Karol Hausman
Google Brain
United States
karolhausman@google.com

Abstract: We present relay policy learning, a method for imitation and reinforcement learning that can solve multi-stage, long-horizon robotic tasks. This general and universally-applicable, two-phase approach consists of an imitation learning stage resulting in goal-conditioned hierarchical policies that can be easily improved using fine-tuning via reinforcement learning in the subsequent phase. Our method, while not necessarily perfect at imitation learning, is very amenable to further improvement via environment interaction allowing it to scale to challenging long-horizon tasks. In particular, we simplify the long-horizon policy learning problem by using a novel data-relabeling algorithm for learning goal-conditioned hierarchical policies, where the low-level only acts for a *fixed* number of steps, regardless of the goal achieved. While we rely on demonstration data to bootstrap policy learning, we do not assume access to demonstrations of specific tasks. Instead, our approach can leverage unstructured and unsegmented demonstrations of semantically meaningful behaviors that are not only less burdensome to provide, but also can greatly facilitate further improvement using reinforcement learning. We demonstrate the effectiveness of our method on a number of multi-stage, long-horizon manipulation tasks in a challenging kitchen simulation environment.

Keywords: Hierarchical RL, Multi-task RL, Imitation Learning

1 Introduction

Recent years have seen reinforcement learning (RL) successfully applied to a number of robotics tasks such as in-hand manipulation [1], grasping [2] and door opening [3]. However, these applications have been largely constrained to relatively simple short-horizon skills. Hierarchical reinforcement learning (HRL) [4] has been proposed as a potential solution that should scale to challenging long-horizon problems, by explicitly introducing temporal abstraction. However, HRL methods have traditionally struggled due to various practical challenges such as exploration [5], skill segmentation [6] and reward definition [7]. We can simplify the above-mentioned problems by utilizing extra supervision in the form of unstructured human demonstrations, in which case the question becomes: how should we best use this kind of demonstration data to make it easier to solve long-horizon robotics tasks?

This question is one focus area of hierarchical imitation learning (HIL), where solutions [8, 9] typically try to achieve two goals: i) learn a temporal task abstraction, and ii) discover a meaningful segmentation of the demonstrations into subtasks. These methods have not traditionally been tailored to further RL fine-tuning, making it challenging to apply them to a long-horizon setting, where pure

[*]Work done as an intern at Google Brain

imitation is very likely to fail. To address this need, we devise a simple and universally-applicable two-phase approach that in the first phase pre-trains hierarchical policies using demonstrations such that they can be easily fine-tuned using RL during the second phase. In contrast to HRL methods, our method takes advantage of unstructured demonstrations to bootstrap further fine-tuning, and in contrast to conventional HIL methods, it does not focus on careful subtask segmentation, making the method simple, general and very amenable to further reinforcement fine-tuning. In particular, we show that we can develop an imitation and reinforcement learning approach that while not necessarily perfect at imitation learning, is very amenable to improvement via fine-tuning with reinforcement learning and that can be scaled to challenging long-horizon manipulation tasks.

First, the approach is very general, in that it can be applied to any demonstration data, including easy to provide unsegmented, unstructured and undifferentiated demonstrations of meaningful behaviors. Second, our method does not require any explicit form of skill segmentation or subgoal definition, which otherwise would need to be learned or explicitly provided. Lastly, and most importantly, since our method ensures that every low-level trajectory is goal-conditioned (allowing for a simple reward specification) and of the same, limited length, it is very amenable to reinforcement fine-tuning, which allows for continuous policy improvement. We show that relay policy learning allows us to learn general, hierarchical, goal-conditioned policies that can solve long-horizon manipulation tasks in a challenging kitchen environment in simulation, while significantly outperforming hierarchical RL algorithms and imitation learning algorithms.

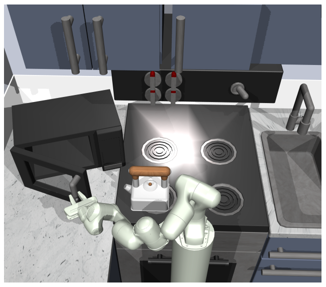
Figure 1: RPL learns complex, long-horizon manipulation tasks in a simulated kitchen environment

2 Related Work

Typical solutions for solving temporally extended tasks have been proposed under the HRL framework [4]. Solutions like the options framework [6, 10], HAM [11], max-Q [12], and feudal networks [13, 14] present promising algorithmic frameworks for HRL. A particularly promising approach was proposed in [15] and [16], using goal conditioned policies at multiple layers of hierarchy for RL. Nevertheless, these algorithms still suffer from challenges in exploration and optimization (as also seen in our experimental comparison with Nachum et al. [15]), which have limited their application to general robotic problems. In this work, we tackle these problems by using additional supervision in the form of unstructured, unsegmented human demonstrations. Our work builds on goal-conditioned RL [17, 18, 19, 20], which has been explored in the context of reward-free learning [21], learning with sparse rewards [18], large scale generalizable imitation learning [22], and hierarchical RL [15]. We build on this principle to devise a general-purpose imitation and RL algorithm that uses data relabeling and bi-level goal conditioned policies to learn complex skills.

There has a been a number of hierarchical imitation learning (HIL) approaches [23, 24, 25, 9, 26] that typically focus on extracting transition segments from the demonstrations. These methods aim to perform imitation learning by learning low-level primitives [9, 26] or latent conditioned policies [23] which meaningfully segment the demonstrations. Traditionally, these approaches do not aim to and are not amenable to improving the learned primitives with subsequent RL, which is necessary as we move towards multi-task, challenging long-horizon problems where pure imitation might be insufficient. This is likely because it is hard to define a clear reward function and objective for continuous improvement of these primitives. In this work, we specifically focus on utilizing both imitation and RL, and devise a method that does not explicitly try to segment out individual primitives into meaningful subtasks, but instead splits the demonstration data into fixed-length segments, amenable to fine-tuning with reinforcement learning. This simplification allows us to leverage the idea of relabelling demonstrations across different goals. Related concepts have shown significant promise with Q-learning algorithms [17, 18]. We introduce a novel form of goal relabeling and demonstrate its efficiency when applied to learning robust bi-level policies. A related idea is presented in Le et al. [27], where the authors assume that an expert provides labelled and segmented demonstrations at both levels of the hierarchy, with an interactive expert for guiding RL. In contrast, we use a pool of unlabelled demonstrations and apply our method to learn a policy to achieve various desired goals, without needing interactive guidance or segmentation. The insight of using imitation learning as a way to bootstrap RL has been previously leveraged by a number of deep RL algorithms [28, 29, 30], where a flat imitation learning initialization is improved using reinforcement learning with additional auxiliary objectives. In this work, we show that we can learn hierarchical policies in a way that can be fine-tuned better than their flat counterparts, owing to temporal decomposition of the problem and the goal conditioned nature of the policy.

3 Preliminaries

Goal-conditioned reinforcement learning: We define $\mathcal{M} = (S, A, P, r)$ to be a finite-horizon Markov decision process (MDP), where S and A are state and action spaces, $P(s_{t+1} \mid s_t, a_t)$ is a transition function, r a reward function. The goal of RL is to find a policy $\pi(a|s)$ that maximizes expected reward over trajectories induced by the policy: $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_i(s_t, a_t)]$. To extend RL to multiple tasks, a goal-conditioned formulation ([17]) can be used to learn a policy $\pi(a|s, s_g)$ which maximizes the expected reward $r(a, s, s_g)$ with respect to a goal distribution $s_g \sim \mathcal{G}$ as follows: $\mathbb{E}_{s_g \sim \mathcal{G}}[\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_i(s_t, a_t, s_g)]]$.

Goal-conditioned imitation learning: In typical imitation learning, instead of knowing the reward r, the agent has access to demonstrations $\mathcal{D}$ containing a set of trajectories $\mathcal{D} = \{\tau^i, \tau^j, \tau^k, ...\}$ of state-action pairs $\tau^i = \{s_0^i, a_0^i, \ldots, s_T^i, a_T^i\}$. The goal is to learn a policy $\pi(a|s)$ that imitates the demonstrations. A common approach is to maximize the likelihood of actions in the demonstration, i.e. $\max E_{(s,a)\sim\mathcal{D}} \log \pi(a|s)$, referred to as behavior cloning (BC). When there are multiple demonstrated tasks, we consider a goal-conditioned imitation learning setup where the dataset of demonstrations $\mathcal{D}$ contains sequences that attempt to reach different goals $s_g^i, s_g^j, s_g^k, \ldots$. The objective is to learn a goal-conditioned policy $\pi(a|s, s_g)$ that is able to reach different goals s_g by imitating the demonstrations.

4 Relay Policy Learning

In this section, we describe our proposed relay policy learning (RPL) algorithm, which leverages unstructured demonstrations and reinforcement learning to solve challenging long-horizon tasks. Our approach consists of two phases: relay imitation learning (RIL), followed by relay reinforcement fine-tuning (RRF) described in Sec. 4.2 and 4.3 respectively. While RIL by itself is not able to solve the most challenging tasks that we consider, it provides a very effective initialization for fine-tuning.

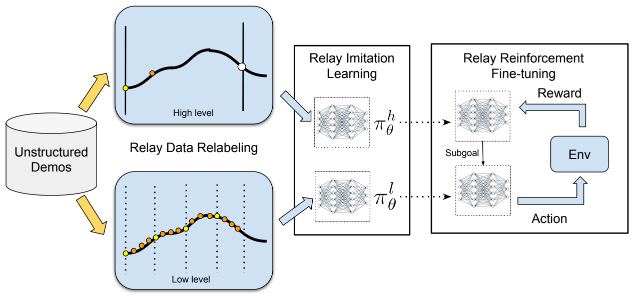

Figure 2: Relay policy learning: the algorithm starts with relabelling unstructured demonstrations at both the high and the low level of the hierarchical policy and then uses them to perform relay imitation learning. This provides a good policy initialization for subsequent relay reinforcement fine-tuning. We demonstrate that learning such simple goal-conditioned policies at both levels from demonstrations using relay data relabeling, combined with relay reinforcement fine-tuning allows us to learn complex manipulation tasks.

4.1 Relay Policy Architecture

We first introduce our bi-level hierarchical policy architecture (shown in Fig 3), which enables us to leverage temporal abstraction. This architecture consists of a high-level goal-setting policy and a low-level subgoal-conditioned policy, which together generate an environment action for a given state. The high-level policy $\pi_\theta^h(s_g^l|s_t, s_g^h)$ takes the current state s_t and a long-term high-level goal s_g^h and produces a subgoal $s_g^l \in \mathcal{S}$ which is then ingested by a low-level policy $\pi_\phi^l(a|s_t, s_g^l)$. The

Algorithm 1 Relay Policy Learning

```
Require: Unstructured pool of demonstrations $D = \{\tau_0, \tau_1, ...\tau_N\}$
1: Relabel goals in demonstration trajectories using Algorithm 2, 3 to extract $D_l$, $D_h$
2: Relay Imitation Learning: Train $\pi_\theta^h$ and $\pi_\phi^l$ using Eqn 1
3: while not done do
4:     Collect on-policy experience with $\pi_\theta^h$ and $\pi_\phi^l$ for high level goals different $s_g^h$
5:     [Optional] Relabel this experience (Sec. 4.3), and add to $D_l$, $D_h$
6:     Update the policy via policy gradient update using Eqn 2, 3.
7: end while
8: Distill fine-tuned policies into a single multi-goal policy
```

Algorithm 2 Relay data relabeling for RIL low level

```
Require: Demonstrations $D = \{\tau_0, \tau_1, ...\tau_N\}$
1: for $n = 1...N$ do
2:     for $t = 1...t_n$ do
3:         for $w = 1...W_l$ do
4:             Add $(s_t^n, a_t^n, s_{t+w}^n)$ to $D_l$
5:         end for
6:     end for
7: end for
```

Algorithm 3 Relay data relabeling for RIL high level

```
Require: Demonstrations $D = \{\tau_0, \tau_1, ...\tau_N\}$
1: for $n = 1...N$ do
2:     for $t = 1...t_n$ do
3:         for $w = 1...W_h$ do
4:             Add $(s_t^n, s_{t+\min(w,W_l)}^n, s_{t+w}^n)$ to $D_h$
5:         end for
6:     end for
7: end for
```

low-level policy takes the current state s_t, and the subgoal s_g^l commanded by the high-level policy and outputs an action a_t, which is executed in the environment.

Importantly, the goal setting policy π_θ^h makes a decision every H time steps (set to 30 in our experiments), with each of its subgoals being kept constant during that period for the low-level policy, while the low-level policy π_ϕ^l operates at every single timestep. This provides temporal abstraction, since the high level policy operates at a coarser resolution than the low-level policy. This policy architecture, while inspired by goal-conditioned HRL algorithms [15], requires a novel learning algorithm to be applicable in the context of imitation learning, which we describe in Sec. 4.2. Given a high-level goal s_g^h, π_θ^h samples a subgoal $s_{g_0}^l$, which is passed to π_θ^l to generate action a_0. For the subsequent H steps, the goal produced by π_θ^h is kept fixed, while π_θ^l generates an action a_t at every time step.

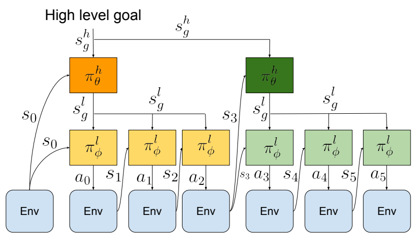

Figure 3: Relay Policy Architecture: A high level goal setter π_θ takes high level goal s_g^h and sets goals s_g^l for a lower level policy π_ϕ, which acts for a fixed time horizon before resampling s_g^l

4.2 Relay Imitation Learning

Our problem setting assumes access to a pool of unstructured, unlabelled demonstrations $\mathcal{D}$ that correspond to meaningful activities provided by the user, without any particular task in mind, e.g. opening cabinet doors, playing with different objects, or simply tidying up the scene. We do not assume that this demonstration data actually accomplishes any of the final task goals that we will need to solve at test-time, though we do need to assume that the test-time goals come from the same distribution of goals as those accomplished in the demonstration data. In order to take the most advantage of such data, we initialize our policy with our proposed relay imitation learning (RIL) algorithm. RIL is a simple imitation learning procedure that uses goal relabeling and hierarchical policy decomposition which result in improved handling of multi-task generalization and compounding error. RIL assumes access to the pool of demonstrations consisting of N trajectories $\mathcal{D} = \{\tau^i, \tau^j, \tau^k, ...\}$, where each trajectory consists of state-action pairs $\tau^i = \{s_0^i, a_0^i, \ldots, s_T^i, a_T^i\}$. Importantly, these demonstrations can be attempting to reach a variety of different high level goals s_g^h, but we do not require these goals to be specified explicitly. To learn the relay policy from these demonstrations, we construct a low-level dataset $\mathcal{D}_l$, and a high-level dataset $\mathcal{D}_h$ from these demonstrations via "relay data relabeling", which is described below, and use them to learn π_θ^h and π_θ^l via supervised learning at multiple levels.

We construct the low-level dataset by iterating through the pool of demonstrations and relabeling them using our relay data relabelling algorithm. First, we choose a window size W_l and generate state-goal-action tuples for $\mathcal{D}_l$, (s, s_g^l, a) by goal-relabeling within a sliding window along the demonstrations, as described in detail below and in Algorithms 2, 3. The key idea behind relay data relabeling is to consider all states that are actually reached along a demonstration trajectory within W_l time steps from any state s_t to be goals reachable from the state s_t by executing action a_t. This allows us to label all states $s_{t+1}, ..., s_{t+W_l}$ along a valid demonstration trajectory as potential goals that are reached from state s_t, when taking action a_t. We repeat this process for all states s_t along all the demonstration trajectories being considered. This procedure ensures that the low-level policy is proficient at reaching a variety of goals from different states, which is crucial when the low-level policy is being commanded potentially different goals generated by the high-level policy.

We employ a similar procedure for the high level, generating the high-level state-goal-action dataset $\mathcal{D}_h$. We start by choosing a high-level window size W_h, which encompasses the high-level goals we would like to eventually reach. We then generate state-goal-action tuples for $\mathcal{D}_h$, via relay data relabeling within the high-level window being considered, as described in Algorithm 2, 3. We also label all states $s_{t+1}, ..., s_{t+W_h}$ along a valid trajectory as potential high-level goals that are reached from state s_t by the high level policy, but we set the high-level action for a goal j steps ahead s_{t+j}, as $s_{t+\min(W_l,j)}$ choosing a sufficiently distant subgoal as the high-level action.

Given these relay-data-relabeled datasets, we train π_θ^l and π_θ^h by maximizing the likelihood of the actions taken given the corresponding states and goals:

$$\max_{\phi,\theta} \mathbb{E}_{(s,a,s_g^l)\sim D_l}[\log \pi_\phi(a|s, s_g^l)] + \mathbb{E}_{(s,s_g^l,s_g^h)\sim D_h}[\log \pi_\theta(s_g^l|s, s_g^h)]. \tag{1}$$

This procedure gives us an initialization for both the low-level and the high-level policies, without the requirement for any explicit goal labeling from a human demonstrator. Relay data relabeling not only allows us to learn hierarchical policies without explicit labels, but also provides algorithmic improvements to imitation learning: (i) it generates more data through the relay-data-relabelling augmentation, and (ii) it improves generalization since it is trained on a large variety of goals.

4.3 Relay Reinforcement Fine-tuning

The procedure described in Sec. 4.2 allows us to extract an effective policy initialization via relay imitation learning. However, this policy is often unable to perform well across all temporally extended tasks, due to the well-known compounding errors stemming from imitation learning [31]. Reinforcement learning provides a solution to this challenge, by enabling continuous improvement of the learned policy directly from experience. We can use RL to improve RIL policies via fine-tuning on different tasks. We employ a goal-conditioned HRL algorithm that is a variant of natural policy gradient (NPG) with adaptive step size [32], where both the high-level and the low-level goal-conditioned policies π_θ^h and π_ϕ^l are being trained with policy gradient in a decoupled optimization.

Given a low-level goal-reaching reward function $r_l(s_t, a_t, s_g^l)$, we can optimize the low-level policy by simply augmenting the state of the agent with the goal commanded by the high-level policy and then optimizing the policy to effectively reach the commanded goals by maximizing the sum of its rewards. For the high-level policy, given a high-level goal-reaching reward function $r_h(s_t, g_t, s_g^h)$, we can optimize it by running a similar goal-conditioned policy gradient optimization to maximize the sum of high-level rewards obtained by commanding the current low-level policy.

To effectively incorporate demonstrations into this reinforcement learning procedure, we leverage our method via: (1) initializing both π_θ^l and π_θ^h with the policies learned via RIL, and (2) encouraging policies at both levels to stay close to the behavior shown in the demonstrations. To incorporate (2), we augment the NPG objective with a max-likelihood objective that ensures that policies at both levels take actions that are consistent with the relabeled demonstration pools D_l and D_h from Section 4.2, as described in Eqn 2 and 3:

$$\nabla_\phi J_l = \mathbb{E}[\nabla_\phi \log \pi_\phi^l(a|s, s_g^l) \sum_t r_l(s_t, a_t, s_g^l)] + \lambda_l \mathbb{E}_{(s,a,s_g^l)\sim\mathcal{D}_l}[\nabla_\phi \log \pi_\phi^l(a|s, s_g^l)] \tag{2}$$

$$\nabla_\theta J_h = \mathbb{E}[\nabla_\theta \log \pi_\theta^h(s_g^l|s, s_g^h) \sum_t r_h(s_t, s_g^l, s_g^h)] + \lambda_h \mathbb{E}_{(s,s_g^l,s_g^h)\sim\mathcal{D}_h}[\nabla_\theta \log \pi_\theta^h(s_g^l|s, s_g^h)]. \tag{3}$$

While a similar objective has been described in [28, 30], it is yet to be explored in the hierarchical, goal-conditioned scenarios, which makes a significant difference as indicated in our experiments.

In addition, since we are learning goal-conditioned policies at both the low and high level, we can leverage relay data relabeling as described in Sec. 4.2 to also enable the use of off-policy data for fine-tuning. Suppose that at a particular iteration i, we sampled N trajectories according to the scheme proposed in Sec. 4.1. While these trajectories did not necessarily reach the goals that were originally commanded, and therefore cannot be considered optimal for those goals, they do end up reaching the *actual* states visited along the trajectory. Thus, they can be considered as optimal when the goals that they were intended for are relabeled to states along the trajectory via relay data relabeling described in Algorithm 2, 3. This scheme generates a low-level dataset $\mathcal{D}_l^i$ and a high level dataset $\mathcal{D}_h^i$ by relabeling the trajectories sampled at iteration i. Since these are considered "optimal" for reaching goals along the trajectory, they can be added to the buffer of demonstrations D_l and D_h, thereby contributing to the objective described in Eqn 2 and Eqn 3 and allowing us to leverage off-policy data during RRF. We experiment with three variants of the fine-tuning update in our experimental evaluation: IRIL-RPL (fine-tuning with Eqn 2, 3 and iterative relay data relabeling to incorporate off-policy data as described above), DAPG-RPL (fine-tuning the policy with the update in Eqn 2, 3 without the off-policy addition) and NPG-RPL (fine-tuning the policy with the update in Eqn 2, 3, without the off-policy addition or the second maximum likelihood term). The overall method is described in Algorithm 1.

As described in Ghosh et al. [33], it is often difficult to learn multiple tasks together with on-policy policy gradient methods, because of high variance and conflicting gradients. To circumvent these challenges, we use RPL to fine-tune on a number of different high level goals individually, and then *distill* all of the learned behaviors into a single policy as described in Rusu et al. [34]. This allows us to learn a single policy capable of achieving multiple high level goals, without dealing with the challenges of multi-task optimization.

5 Experimental Results

Our experiments aim to answer the following questions: (1) Does RIL improve imitation learning with unstructured and unlabelled demonstrations? (2) Is RIL more amenable to RL fine-tuning than its flat, non-hierarchical alternatives? (3) Can we use RPL to accomplish long-horizon manipulation tasks? Videos and further experimental details are available at `https://sites.google.com/view/relay-policy-learning`

Environment Setup To evaluate our algorithm, we utilize a challenging robotic manipulation environment modeled in MuJoCo, shown in Fig. 1. The environment consists of a 9 DoF position-controlled Franka robot interacting with a kitchen scene that includes an openable microwave, four turnable oven burners, an oven light switch, a freely movable kettle, two hinged cabinets, and a sliding cabinet door. We consider reaching different goals in the environment, as shown in Fig. 4, each of which may require manipulating many different components. For instance, in Fig. 4 (a), the robot must open the microwave, move the kettle, turn on the light, and slide open the cabinet. While the goals we consider are temporally extended, the setup is fully general. We collect a set of unstructured and unsegmented human demonstrations described in Sec. 4.2, using the PUPPET MuJoCo VR system [35]. We provide the algorithm with 400 sequences containing various unstructured demonstrations that each manipulate four different elements of the scene in sequence.

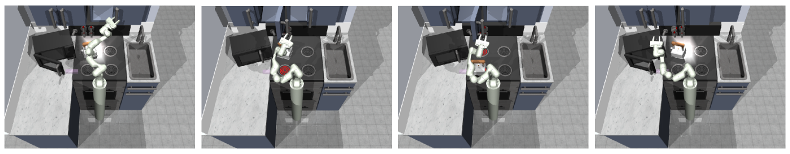

Figure 4: Examples of compound goals in the kitchen environment. Each goal has different elements manipulated, requiring multiple stages to solve: (a) microwave, kettle, light, slider, (b) kettle, burner, slider, cabinet, (c) burner, top burner, slide hinge, (d) kettle, microwave, top burner, lights

Evaluation and Comparisons Since each of our tasks consist of compound goals that involve manipulating four elements in the environment, we evaluate policies based on the number of steps that they complete out of four, which we refer to as step-completion score. A step is completed when the corresponding element in the scene is moved to within ϵ distance of its desired position.

We compare variants of our RPL algorithm to a number of ablations and baselines, including prior algorithms for imitation learning combined with RL and methods that learn from scratch. Among algorithms which utilize imitation learning combined with RL, we compare with several methods that utilize flat behavior cloning with additional finetuning. Specifically, we compare with (1) flat goal-conditioned behavior cloning followed by finetuning (*BC*), (2) flat goal-conditioned behavior cloning trained with data relabeling followed by finetuning (*GCBC*) [22], and variants of these algorithms that augment the *BC* and *GCBC* fine-tuning with losses as described in Rajeswaran et al. [28] - (3) *DAPG-BC* and (4) *DAPG-GCBC*. We also compare RPL to (5) hierarchical imitation learning + finetuning with an oracle segmentation scheme, which performs hierarchical goal conditioned imitation learning by using a hand-specified oracle to segment the demonstrations for imitation learning, followed by RRF style fine-tuning. Details of this scheme can be found in Appendix 3. For comparisons with methods that learn from scratch we compare with (6) an on-policy variant of *HIRO* [15] trained from scratch with natural policy gradient [32] instead of Q-learning and (7) a baseline (*Pre-train low level*) that learns low-level primitives from the demonstration data, and learns the high-level goal-setting policy from scratch with RL. The last baseline is representative of a class of HIL algorithms [23, 24, 26], which are difficult to fine-tune because it is not clear how to provide rewards for improving low-level primitives. Lastly, we compare RPL with a baseline (7) (*Nearest Neighbor*) which uses a nearest neighbor strategy to choose the demonstration which has the achieved goal closest to the commanded goal and subsequently executes actions open-loop.

5.1 Relay Imitation Learning from Unstructured Demonstrations

We start by aiming to understand whether RIL improves imitation learning over standard methods. We compare the step-wise completion scores averaged over 17 different compound goals with RIL as compared to flat BC variants. We find that, while none of the variants are able to achieve near-perfect completion scores via just imitation, the average stepwise completion score is higher for RIL as compared to both flat variants (see Table 1, first row). Additionally, we find that the flat policy with data augmentation via relabeling performs better than without relabeling. When we analyze the proportion of compound goals that are actually fully achieved (see Table 1, bottom row), RIL shows significant improvement over other methods. This indicates that, even for imitation learning, we see benefits from introducing the simple RIL scheme described in Sec. 4.2.

	RIL (ours)	GCBC relabeling	GCBC no relabeling
Success Rate (%)	21.7	8.8	7.6
Average Step Completion (of 4)	2.4 $\pm$1.13	2.2 $\pm$ 0.95	1.78 $\pm$ 1.0

Table 1: Comparison of RIL to goal-conditioned behavior cloning with and without relabeling in terms success and step-completion rate averaged across 17 tasks. RIL outperforms the non-hierarchical methods

5.2 Relay Reinforcement Fine-tuning of Imitation Learning Policies

Although pure RIL does succeed at times, its performance is still relatively poor. In this section, we study the degree to which RIL-based policies are amenable to further reinforcement fine-tuning. Performing reinforcement fine-tuning individually on 17 different compound goals seen in the demonstrations, we observe a significant improvement in the average success rate and stepwise completion scores over all the baselines when using any of the variants of RPL (see Fig. 5). In our experiments, we found that it was sufficient to fine-tune the low-level policy, although we could also fine-tune both levels, at the cost of more non-stationarity. Although the large majority of the benefit is from RRF, we find a slight additional improvement from the DAPG-RPL and IRIL-RPL schemes, indicating that including the effect of the demonstrations throughout the process helps.

When compared with HRL algorithms that learn from scratch (on-policy HIRO [15]), we observe that RPL is able to learn much faster and reach a much higher success rate, showing the benefit of demonstrations. Additionally, we notice better fine-tuning performance when we compare RPL with flat-policy fine-tuning. This can be attributed to the fact that the credit assignment and reward specification problems are much easier for the relay policies, as compared to fine-tuning flat policies,

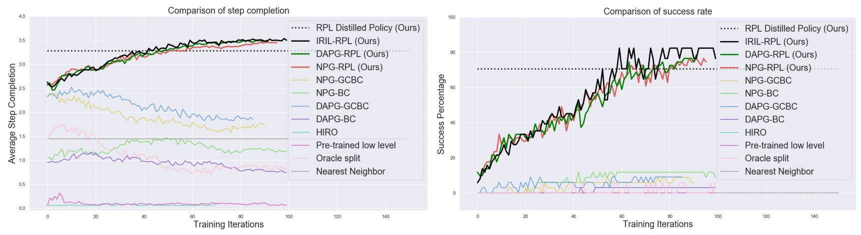

Figure 5: Comparison of the RPL algorithm with a number of baselines averaged over 17 compound goals and 2 (baseline methods) or 3 (our approach) random seeds. Fine-tuning with all three variants of our method outperforms fine-tuning using flat policies. Unsurprisingly, RIL initialization at both levels improves the performance over HIRO [15] or learning only the high-level policy from scratch. If we use policy distillation, we are able to get a successful, multi-task goal-conditioned policy.

where a sparse reward is rarely obtained. The RPL method also outperforms the pre-train-low-level baseline, which we hypothesize is because we are not able to search very effectively in the goal space without further guidance. We also see a significant benefit over using the oracle scheme described in Appendix 3, since the segments become longer making the exploration problem more challenging. The comparison with the nearest neighbor baseline also suggests that there is a significant benefit from actually learning a *closed-loop* policy rather than using an open-loop policy. While plots in Fig. 5 show the average over various goals when fine-tuned individually, we can also distill the fine-tuned policies into a single, multi-task policy, as described in Sec. 5, that is able to solve almost all of the compound goals that were fine-tuned. While the success rate drops slightly, this gives us a single multi-task policy that can achieve multiple temporally-extended goals (Fig 5).

5.3 Ablations and Analysis

To understand design choices, we consider the role of using different window sizes for RPL as well as the role of reward functions during fine-tuning. In Fig 6 (left), we observe that the window size for RPL plays a major role in algorithm performance. As window size increases, both imitation learning and fine-tuning performance decreases since the behaviors are now more temporally extended.

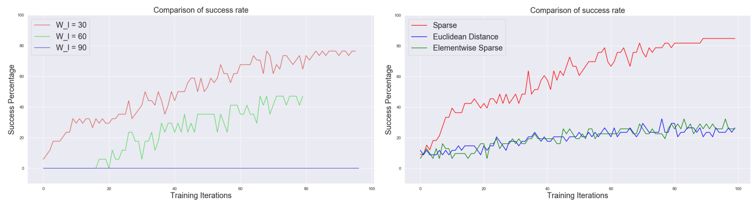

Figure 6: **Left:** Role of low level window size in RPL. As the window size increases, imitation learning and fine-tuning become less effective. **Right:** Role of fine-tuning reward function in RPL. We see that the sparse reward function is most effective once exploration is sufficiently directed.

Next, we consider the role of the chosen reward function in fine-tuning with RRF. We evaluate the relative performance of using different types of rewards for fine-tuning - sparse reward, euclidean distance, element-wise reward (refer to Appendix A for details). When each is used as a goal conditioned reward for fine-tuning the low-level, sparse reward works much better. This indicates that when exploration is sufficient, sparse reward functions are less prone to local optima than alternatives.

6 Conclusion and Future Work

We proposed relay policy learning, a method for solving long-horizon, multi-stage tasks by leveraging unstructured demonstrations to bootstrap a hierarchical learning procedure. We showed that we can learn a single policy capable of achieving multiple compound goals, each requiring temporally extended reasoning. In addition, we demonstrated that RPL significantly outperforms other baselines that utilize hierarchical RL from scratch, as well as imitation learning algorithms.

In future work, we hope to tackle the problem of generalization to longer sequences and study extrapolation beyond the demonstration data. We also hope to extend our method to work with off-policy RL algorithms, so as to further improve data-efficiency and enable real world learning on a physical robot.

References

[1] V. Kumar, E. Todorov, and S. Levine. Optimal control with learned local models: Application to dexterous manipulation. In *ICRA 2016*.

[2] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation.

[3] S. Gu, E. Holly, T. P. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA 2017*.

[4] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 2003.

[5] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS 2016*.

[6] R. S. Sutton, D. Precup, and S. P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 1999.

[7] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *CoRR*, abs/1802.06070, 2018. URL http://arxiv.org/abs/1802.06070.

[8] S. Krishnan, A. Garg, R. Liaw, B. Thananjeyan, L. Miller, F. T. Pokorny, and K. Goldberg. SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *I. J. Robotics Res.*, 38(2-3), 2019. doi:10.1177/0278364918784350. URL https://doi.org/10.1177/0278364918784350.

[9] R. Fox, S. Krishnan, I. Stoica, and K. Goldberg. Multi-level discovery of deep options. *CoRR*, abs/1703.08294, 2017. URL http://arxiv.org/abs/1703.08294.

[10] P. Bacon, J. Harb, and D. Precup. The option-critic architecture. *CoRR*, abs/1609.05140, 2016. URL http://arxiv.org/abs/1609.05140.

[11] R. Parr and S. J. Russell. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*, pages 1043–1049, 1997. URL http://papers.nips.cc/paper/1384-reinforcement-learning-with-hierarchies-of-machines.

[12] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res.*, 13:227–303, 2000. doi:10.1613/jair.639. URL https://doi.org/10.1613/jair.639.

[13] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pages 271–278, 1992. URL http://papers.nips.cc/paper/714-feudal-reinforcement-learning.

[14] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017. URL http://arxiv.org/abs/1703.01161.

[15] O. Nachum, S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 3307–3317, 2018.

[16] A. Levy, R. P. Jr., and K. Saenko. Hierarchical actor-critic. *CoRR*, abs/1712.00948, 2017. URL http://arxiv.org/abs/1712.00948.

[17] L. P. Kaelbling. Learning to achieve goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1094–1099, 1993.

[18] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017. URL http://arxiv.org/abs/1707.01495.

[19] V. Pong, S. Gu, M. Dalal, and S. Levine. Temporal difference models: Model-free deep RL for model-based control. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=Skw0n-W0Z.

[20] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1312–1320, 2015. URL http://proceedings.mlr.press/v37/schaul15.html.

[21] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *NeurIPS 2018*, pages 9209–9220, 2018. URL http://papers.nips.cc/paper/8132-visual-reinforcement-learning-with-imagined-goals.

[22] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. *CoRR*, abs/1903.01973, 2019.

[23] K. Hausman, Y. Chebotar, S. Schaal, G. S. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *NeurIPS 2017*.

[24] P. Henderson, W. Chang, P. Bacon, D. Meger, J. Pineau, and D. Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *AAAI 2018*.

[25] A. Sharma, M. Sharma, N. Rhinehart, and K. M. Kitani. Directed-info GAIL: learning hierarchical policies from unsegmented demonstrations using directed information. *CoRR*, abs/1810.01266, 2018.

[26] T. Kipf, Y. Li, H. Dai, V. F. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. Compile: Compositional imitation learning and execution. In *ICML 2019*.

[27] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. D. III. Hierarchical imitation and reinforcement learning.

[28] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS 2018*.

[29] Y. Zhu, Z. Wang, J. Merel, A. A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and imitation learning for diverse visuomotor skills. In *RSS 2018*.

[30] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *ICRA 2018*.

[31] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS 2011*.

[32] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

[33] D. Ghosh, A. Singh, A. Rajeswaran, V. Kumar, and S. Levine. Divide-and-conquer reinforcement learning. *CoRR*, abs/1711.09874, 2017. URL http://arxiv.org/abs/1711.09874.

[34] A. A. Rusu, S. G. Colmenarejo, Ç. Gülçehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. In *ICLR 2016*.

[35] V. Kumar and E. Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 657–663. IEEE, 2015.

A Experimental Details

We use feed-forward MLPs for all our policies, with two layer neural networks with 256 units each and ReLu nonlinearities used for both the high-level policy π_θ and the low-level policy π_ϕ in all methods. Flat baselines use the same architecture as well, but additional experimentation with the architecture did not yield substantially different results. We train all imitation learning algorithms with the ADAM optimizer using a batchsize of 128 and a learning rate of 0.005. We choose W_l to be 30 in all experiments and W_h to be 260 in the experiments. The larger the window is, the harder the learning problem becomes for both imitation and finetuning.

For reinforcement learning, we utilize a variant of Trust Region Policy Optimization (TRPO). We fine-tune on 17 different compound goals individually, with a path length of 280 for every compound goal, and the low-level horizon set to 30. We use 100 trajectories in each iteration of on-policy finetuning, with a discount of 0.995. When using variants of augmenting the policy gradient objective with demonstrations, we experimented with different weightings λ_h and λ_l, but we found 0.0001 to work well. We use a batch size of a 100 trajectories per iteration, and fairly standard parameters for truncated natural policy gradient based on `https://github.com/aravindr93/mjrl`

The simulation environment has a 30 dimensional state space which consists of positions of the arm, the objects in the scene and the various implements in the kitchen. The action space is 9 dimensional with 7 DoF for the arm and 2 DoF for the gripper, which is velocity controlled in joint space.

B Reward Function Details

For the comparisons detailed in Section 5.3, the reward functions used for sparse, euclidean and elementwise sparse reward functions are detailed below, with ϵ set to 0.3. For all our experimental results in Fig 5, we use the sparse reward variant as the reward function for finetuning.

$$R_{\text{sparse}}(s, g) = \mathbb{1}(\|s - g\|_2 < \epsilon) \tag{4}$$

$$R_{\text{euclidean}}(s, g) = -\|s - g\|_2 \tag{5}$$

$$R_{\text{elementwise sparse}}(s, g) = \sum_{\text{idx} \in \text{element indices}} \mathbb{1}(\|s[\text{idx}] - g[\text{idx}]\|_2 < \epsilon) \tag{6}$$

In the element-wise sparse reward case, idx is selected to be the indices of state corresponding to different distinct elements of the scene such as the microwave, stove burners, light switch, sliding cabinet, hinge cabinets and so on. The robot arm is excluded from these indices.

C Oracle Baseline Details

For the oracle comparison described in Section 5, a hand-designed scheme is used to segment the demonstration into segments corresponding to semantically meaningful components, thereby generating variable sized windows rather than fixed length ones. Specifically, we considered breaking a segment any time one of [microwave, kettle, light switch, burners, slide cabinet, hinge cabinet] is moved more than $\epsilon = 0.3$. This leads to a variable segment generation scheme, which generates splits that is shown in Fig 7.

Segments generated in this fashion can then be used for imitation learning both the low-level and high-level policies. Specifically, the actions for the high level policies are chosen to be the states at which the segments are broken, and the low level is trained via goal conditioned behavior cloning with those states set as goals.

D Visualization of Learned Behaviors

We show example visualizations of several successful learned behaviors for compound tasks, and some failed behaviors to better understand the behavior of the method. These can

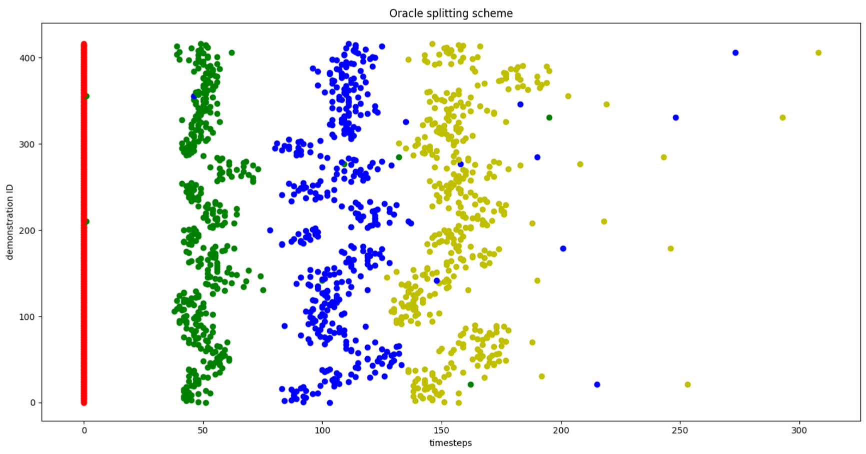

Figure 7: Splits generated by the oracle segmentation scheme. Each color corresponds to a different split and different demonstrtaions as plotted as different rows along the y-axis, with time-steps along the x-axis. We see that the split of demonstrations is fairly variable in time-steps. This makes the imitation learning and finetuning quite challenging.

be best appreciated by viewing the accompanying video on the supplementary website https://sites.google.com/view/relay-policy-learning.

D.1 Successful cases

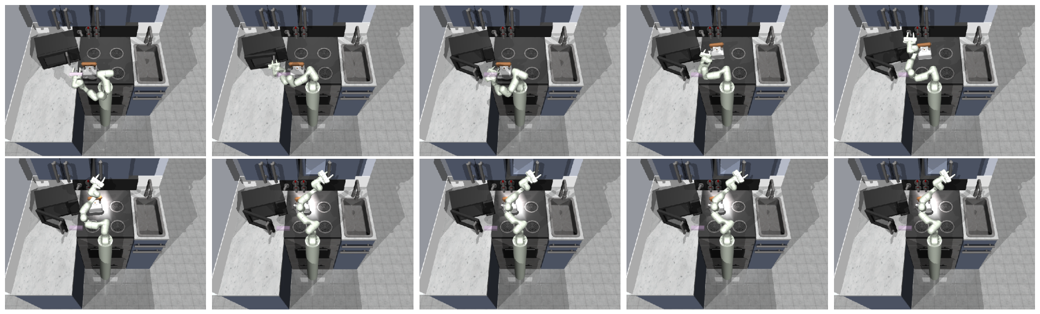

Figure 8: Visualization of successful learned behavior for opening microwave, moving kettle, turning on light switch, sliding the slider

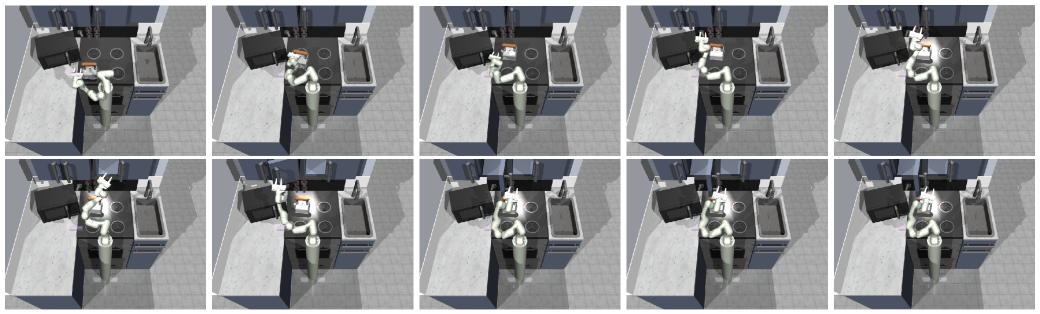

Figure 9: Visualization of successful learned behavior for moving kettle, turning top knob, sliding the slider and opening the hinge cabinet

D.2 Failure Cases

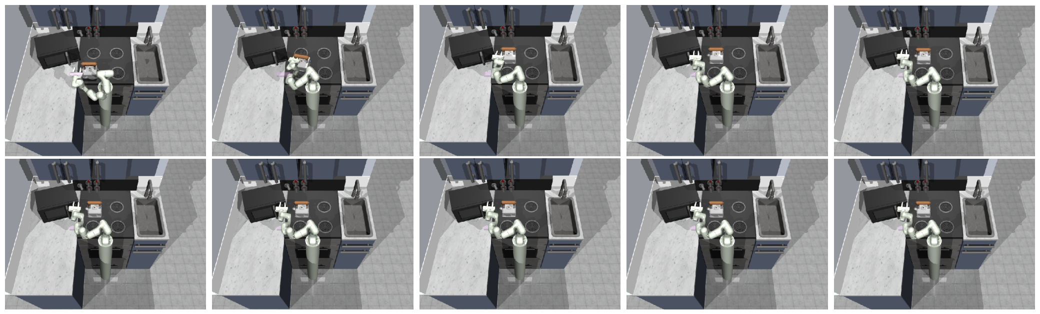

Figure 10: Visualization of failing learned behavior for moving kettle, turning the bottom knob, moving the slider and turning on the oven light