CRISP: CURRICULUM INDUCING PRIMITIVE INFORMED SUBGOAL PREDICTION FOR HIERARCHICAL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical reinforcement learning is a promising approach that uses temporal abstraction to solve complex long horizon problems. However, simultaneously learning a hierarchy of policies is unstable as it is challenging to train higher-level policy when the lower-level primitive is non-stationary. In this paper, we propose to generate a curriculum of achievable subgoals for evolving lower-level primitives using reinforcement learning and imitation learning. The lower level primitive periodically performs data relabeling on a handful of expert demonstrations using our primitive informed parsing method. We provide expressions to bound the sub-optimality of our method and develop a practical algorithm for hierarchical reinforcement learning. Since our approach uses a handful of expert demonstrations, it is suitable for most real world robotic control tasks. Experimental results on complex maze navigation and robotic manipulation environments show that inducing hierarchical curriculum learning significantly improves sample efficiency, and results in better learning of goal conditioned policies in complex temporally extended tasks.

1 INTRODUCTION

Reinforcement learning (RL) algorithms have made significant progress in solving continuous control tasks like performing robotic arm manipulation Levine et al. (2015); Vecerík et al. (2017) and learning dexterous manipulation Rajeswaran et al. (2017). However, the success of RL algorithms on complex long horizon continuous tasks has been limited by issues like long term credit assignment and inefficient exploration Nachum et al. (2019); Kulkarni et al. (2016), especially in sparse reward scenarios Andrychowicz et al. (2017). Hierarchical reinforcement learning (HRL) Dayan & Hinton (1993); Sutton et al. (1999); Parr & Russell (1998) promises the benefits of temporal abstraction and efficient exploration for solving tasks that require long term planning. In *goal-conditioned* hierarchical framework, the high-level policy predicts subgoals for lower primitive, which in turn performs primitive actions directly on the environment Nachum et al. (2018); Vezhnevets et al. (2017); Levy et al. (2017). However, simultaneously learning multi-level policies has been found to be challenging in practice due to non-stationary higher level state transition and reward functions.

Prior works have leveraged expert demonstrations to bootstrap learning Nair et al. (2017); Rajeswaran et al. (2017); Hester et al. (2017). Some approaches rely on leveraging expert demonstrations via fixed parsing, and consequently bootstrapping multi-level hierarchical RL policy using imitation learning Gupta et al. (2019). Generating an efficient subgoal transition dataset is *crucial* in such tasks. In this work, we propose an adaptive parsing technique for leveraging expert demonstrations and show that it outperforms fixed parsing based approaches on tasks that require long term planning. Ideally, a good subgoal should properly balance the task split between the hierarchical levels according to current goal reaching ability of the lower primitive, thus avoiding degenerate solutions. As the lower primitive improves, the subgoals provided to lower primitive should become progressively more difficult, such that *(i)* the subgoals are always achievable by the current lower level primitive, *(ii)* task split is properly balnced between hierarchical levels, and *(iii)* reasonable progress is made towards achieving the final goal. In this work, we introduce hierarchical curriculum learning to deal with non-stationarity issue. We build upon these ideas and propose a generally applicable HRL approach: *Curriculum inducing primitive informed subgoal prediction* (CRISP).

CRISP parses a handful of expert demonstrations using our novel subgoal relabeling method: primitive informed parsing (PIP). In PIP, current lower primitive is used to perform data relabeling on expert demonstrations dataset to generate efficent subgoal supervision for the higher level policy. Since the lower primitive performs data relabeling, this approach does not require explicit labeling or segmentation of demonstrations by an expert. The periodically generated higher level subgoal dataset is used with an additional imitation learning (IL) objective to provide curriculum based regularization for the higher policy. For imitation learning, we devise inverse reinforcement learning regularizer Ghasemipour et al. (2020); Kostrikov et al. (2018); Ho & Ermon (2016), which constraints the state marginal of the learned policy to be similar to that of the expert demonstrations. The details of CRISP, PIP, and IRL objective are mentioned in Section 3. We also derive sub-optimality bounds in Section 3.2 to theoretically justify the benefits of curriculum learning in hierarchical framework. Finally, we provide a practical approach to perform hierarchical reinforcement learning.

Since our approach uses a handful of expert demonstrations, it is generally applicable on most complex long horizon tasks. We perform experiments on random maze navigation and complex robotic pick and place environments, and empirically verify that the proposed approach clearly outperforms the baseline approaches on long horizon tasks.

2 BACKGROUND

We consider *Universal Markov Decision Process* (UMDP) Schaul et al. (2015) setting, which is a Markov Decision process (MDP) augmented with the goal space G. UMDPs are represented as a 6-tuple (S, A, P, R, γ, G), where S is the state space, A is the action space, $P(s^{'}|s, a) = \mathbb{P}(s_{t+1} = s^{'}|s_t = s, a_t = a)$ is the transition function that describes the probability of reaching state $s^{'}$ when the agent takes action a in the current state s. The reward function R generates rewards r at every timestep, γ is the discount factor, and G is the goal space. In the UMDP setting, a fixed goal g is selected for an episode, and $\pi(a|s, g)$ denotes the goal-conditioned policy. $d^{\pi}(s) = (1-\gamma)\sum_{t=0}^{T}\gamma^t P(s_t = s|\pi)$ represents the discounted future state distribution, and $d_c^{\pi}(s) = (1-\gamma^c)\sum_{t=0}^{T}\gamma^{tc} P(s_{tc} = s|\pi)$ represents the c-step future state distribution for policy π. The overall objective is to learn policy $\pi(a|s, g)$ which maximizes the expected future discounted reward objective $J = (1-\gamma)^{-1}\mathbb{E}_{s\sim d^{\pi}, a\sim\pi(a|s,g), g\sim G}\left[r(s_t, a_t, g)\right]$

Let s be the current state and g be the final goal for the current episode. In our goal-conditioned hierarchical RL setup, the overall policy π is divided into multi-level policies. The higher level policy $\pi^H(s_g|s, g)$ predicts subgoals Dayan & Hinton (1993) s_g for the lower level primitive $\pi^L(a|s, s_g)$, which in turn executes primitive actions a directly on the environment. The lower primitive π^L tries to achieve subgoal s_g within c timesteps by maximizing intrinsic rewards r_{in} provided by the higher level policy. The higher level policy π^H gets extrinsic reward r_{ex} from the environment, and predicts the next subgoal s_g for the lower primitive. The process is continued until either the final goal g is achieved, or the episode terminates. We consider sparse reward setting where the lower primitive is sparsely rewarded intrinsic reward r_{in} if the agent reaches within δ^L distance of the predicted subgoal s_g: $r_{in} = 1(\|s_t - s_g\|_2 \leq \delta^L)$, and the higher level policy is sparsely rewarded extrinsic reward r_{ex} if the achieved goal is within δ^H distance of the final goal g: $r_{ex} = 1(\|s_t - g\|_2 \leq \delta^H)$. We assume access to a handful of expert demonstrations $D = \{e^i\}_{i=1}^N$, where $e^i = (s_0^e, s_1^e, \ldots, s_{T-1}^e)$. We only assume access to demonstration states s_i^e (and not demonstration actions) which can be obtained in most robotic control tasks.

3 METHODOLOGY

In this section, we explain our hierarchical curriculum learning based approach CRISP. First, we formulate our primitive informed parsing method PIP, which periodically performs data relabeling on expert demonstrations to populate subgoal transition dataset. Then, we explain how we use this dataset to learn high level policy using reinforcement learning and additional IRL based regularization objective.

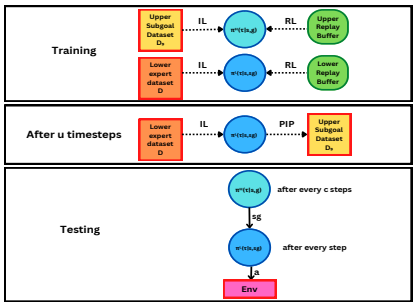

Figure 1: Overview of CRISP: While training, both upper policy(π^H) and lower primitive(π^L) are trained using RL and IL(imitation learning). After every u timesteps, D_g is re-populated using PIP.

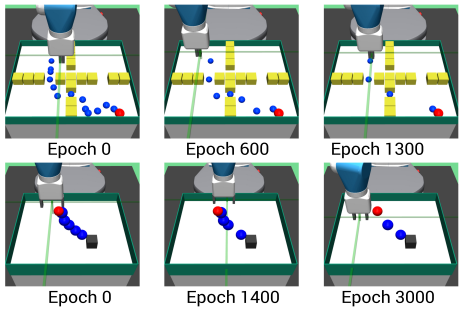

Figure 2: This figure depicts CRISP subgoals progress vs training epochs elapsed. The higher level policy generates a curriculum of subgoals for lower primitive(walls:yellow, final goal:red, subgoals:blue)

3.1 PRIMITIVE INFORMED PARSING: PIP

Primitive informed parsing approach uses the current lower primitive π_L^j to parse expert state demonstrations dataset $\mathcal{D}$, where superscript j in π_L^j is an indicator of the current subgoal reaching capability of the lower primitive. The underlying idea is: PIP should select sequences of maximally temporally separated states from demonstration trajectory e. These maximally temporally separated state sequences constitute the higher level subgoal dataset $\mathcal{D}_g$. We explain below how PIP adaptively parses expert demonstration trajectories from D.

We start with current lower primitive π_L^j and an expert state demonstration trajectory $e = (s_0^e, s_1^e, \ldots, s_{T-1}^e)$. The environment is reset to state s_0^e. Starting at $i = 1$ to $T - 1$, we incrementally provide states s_i^e as subgoals to lower primitive π_L^j. From s_0^e, π_L^j tries to achieve s_i^e within c timesteps. If π_L^j fails to achieve the subgoal s_i^e from the initial state, we add s_i^e to the list of subgoals. The underlying idea is that since s_{i-1}^e was the last subgoal achieved by lower-level primitive, it thus makes a good candidate for maximally reachable subgoal. Once we have added s_{i-1}^e to the list of subgoals, we continue the process after setting s_{i-1}^e as the new initial state until we reach the end of demonstration trajectory e. This subgoal transition sequence thus collected is added to D_g. The method thus populates the subgoal transition dataset with maximally temporally separated achievable subgoals.

Algorithm 1 PIP: Primitive Informed Parsing

Initialize $D_g = \{\}$
for each trajectory $e = (s_0^e, s_1^e, \ldots, s_{T-1}^e)$ in $\mathcal{D}$ **do**
    initial state $\leftarrow s_0^e$
    final goal $\leftarrow g$
    Initialize list of subgoals $D_g^e = \{\}$
    for i = 1 **to** $T - 1$ **do**
        Reset to initial state
        Pass s_i^e as the current goal to π_L^j
        if s_i^e is not achieved by π_L^j in c time-steps **then**
            Add (initial state, s_i^e), final goal) to D_g^e
            initial state $\leftarrow s_{i-1}^e$
    $D_g \leftarrow D_g \cup D_g^e$

The pseudocode for PIP is given in Algorithm 1. PIP maximizes the utility of the lower level primitive while ensuring that it always receives achievable subgoals. However, it assumes that we can reset the environment to any state in $\mathcal{D}$ while collecting subgoal dataset. We discuss different ways to relax this assumption in section 6.

3.2 SUBOPTIMALITY ANALYSIS

In this section, we analyze the suboptimality of our method, and examine how the performance benefits from curriculum learning and imitation learning objective. Let π^* be the unknown optimal policy to be learnt, $\pi^H_{\theta_H}$ is our high level CRISP policy, and $\pi^L_{\theta_L}$ is our lower CRISP primitive policy, where θ_H and θ_L are trainable parameters of higher and lower level policies respectively. s is the current state, g is the final episodic goal, s_g is the subgoal provided by upper level policy and τ are c length sub-trajectories. We assume Π_D is probability distribution generating dataset D, κ is some distribution over states and actions, and G is the goal space. We extend the definition from Ajay et al. (2020) to goal-conditioned policies:

Definition 1. π^* is ϕ_D-common in Π_D, if $\mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,g\sim G}[D_{TV}(\pi^*(\tau|s,g)||\pi_D(\tau|s,g))] \leq \phi_D$

We define the suboptimality of policy π with respect to optimal policy π^* as:

$$Subopt(\theta) = |J(\pi^*) - J(\pi)| \tag{1}$$

Theorem 1. Assuming the optimal policy π^* is ϕ_D common in Π_D, the suboptimality of upper policy $\pi^H_{\theta_H}$, over c length sub-trajectories τ sampled from $d_c^{\pi^*}$ can be bounded as:

$$|J(\pi^*) - J(\pi^H_{\theta_H})| \leq \lambda * \phi_D + \lambda * \mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,g\sim G}[D_{TV}(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))]] \tag{2}$$

where $\lambda = \frac{2}{(1-\gamma)(1-\gamma^c)} R_{max} \| \frac{d_c^{\pi_*}}{\kappa} \|_\infty$

Furthermore, the suboptimality of lower primitive $\pi^L_{\theta_L}$ can be bounded as:

$$|J(\pi^*) - J(\pi^L_{\theta_L})| \leq \lambda * \phi_D + \lambda * \mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,s_g\sim\pi^H_{\theta_H}}[D_{TV}(\pi_D(\tau|s,s_g)||\pi^L_{\theta_L}(\tau|s,s_g))]] \tag{3}$$

where $\lambda = \frac{2}{(1-\gamma)^2} R_{max} \| \frac{d_c^{\pi^*}}{\kappa} \|_\infty$

Proof. We extend the suboptimality bound from Ajay et al. (2020) between goal conditioned policies π^* and $\pi^H_{\theta_H}$ as follows:

$$|J(\pi^*) - J(\pi^H_{\theta_H})| \leq \frac{2}{(1-\gamma)(1-\gamma^c)} R_{max} \mathbb{E}_{s\sim d_c^{\pi^*},g\sim G}[D_{TV}(\pi^*(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))] \tag{4}$$

By applying triangle inequality:

$$D_{TV}(\pi^*(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g)) \leq D_{TV}(\pi^*(\tau|s,g)||\pi_D(\tau|s,g)) + D_{TV}(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g)) \tag{5}$$

Taking expectation wrt $s \sim \kappa$, $g \sim G$ and $\pi_D \sim \Pi_D$,

$$\begin{aligned}\mathbb{E}_{s\sim\kappa,g\sim G}[D_{TV}(\pi^*(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))] \leq \mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,g\sim G}[D_{TV}(\pi^*(\tau|s,g)||\pi_D(\tau|s,g))] + \\ \mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,g\sim G}[D_{TV}(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))]\end{aligned} \tag{6}$$

Since π^* is ϕ_D common in Π_D, we can write 6 as:

$$\mathbb{E}_{s\sim\kappa,g\sim G}[D_{TV}(\pi^*(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))] \leq \phi_D + \mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,g\sim G}[D_{TV}(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))] \tag{7}$$

Substituting the result from Equation 7 in Equation 4, we get

$$|J(\pi^*) - J(\pi^H_{\theta_H})| \leq \lambda * \phi_D + \lambda * \mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,g\sim G}[D_{TV}(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))]] \tag{8}$$

where $\lambda = \frac{2}{(1-\gamma)(1-\gamma^c)} R_{max} \| \frac{d_c^{\pi^*}}{\kappa} \|_\infty$ □

Equation 2 can be rearranged to yield the following form:

$$J(\pi^*) \geq J(\pi^H_{\theta_H}) - \lambda * \phi_D - \lambda * \mathbb{E}_{s\sim\kappa,\pi_D\sim\Pi_D,g\sim G}[d(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))] \tag{9}$$

where, $d(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g)) = D_{TV}(\pi_D(\tau|s,g)||\pi^H_{\theta_H}(\tau|s,g))$

This can be solved as a minorize maximize algorithm which intuitively means: the overall objective can be optimized by (i) maximizing the objective $J(\pi^H_{\theta_H})$ via RL, and (ii) minimizing TV divergence between π^* and $J(\pi^H_{\theta_H})$. We use entropy regularized RL technique Soft Actor Critic Haarnoja et al. (2018a) to maximize $J(\pi^H_{\theta_H})$.

3.3 HIERARCHICAL CURRICULUM LEARNING

In Equation 2, the suboptimality bound is dependent on ϕ_{D_g}, which represents how good is the subgoal dataset D_g populated by PIP. A lower value of ϕ_{D_g} implies that the optimal policy π^* is closely represented by the dataset D_g. Since we use lower primitive to parse expert demonstrations, as the lower primitive gets better, π_{D_g} gets closer to π_*. Hence D_g improves and the value of parameter ϕ_D decreases, which implies that suboptimality bound in Equation 2 gets tighter.

To implement curriculum learning while generating subgoal transition dataset D_g, the dataset D_g is cleared after every u timesteps, and re-populated using PIP, as explained in Algorithm 2. Periodically re-populating buffer D_g after u timesteps generates a natural curriculum for lower primitive, as shown in Figure 2.

3.4 IMITATION LEARNING REGULARIZATION FOR HIGHER-LEVEL POLICY

Different approximations to the distance function d in Equation 9 yield different imitation learning regularizers. If d is replaced by Kullback–Leibler divergence, the imitation learning regularizer becomes supervised MLE objective Nair et al. (2017). If we replace d with Jensen-Shannon divergence, the imitation learning objective takes the form of Inverse RL objective. We consider both behavior cloning and IRL objectives in our experiments. We devise IRL objective as a GAIL Ho & Ermon (2016) like objective implemented using LSGAN Mao et al. (2016). Let $(s^e, g^e, s^e_g) \sim D_g$ be a subgoal transition where s^e is a state in an expert trajectory, g^e is the corresponding final goal and s^e_g is the subgoal. Let s_g be the subgoal predicted by the high level policy $\pi^H_\theta(\cdot|s^e, g^e)$. We bootstrap the learning of higher level policy by optimizing:

$$\max_{\pi^H_\theta} \min_{\epsilon} \frac{1}{2}\mathbb{E}_{(s^e,g^e,s^e_g)\sim D_g}[\mathbb{D}^H_\epsilon(s^e_g) - 1]^2 + \frac{1}{2}\mathbb{E}_{(s^e,g^e)\sim D_g, s_g\sim\pi^H_\theta(\cdot|s^e,g^e)}[\mathbb{D}^H_\epsilon(\pi^H_\theta(\cdot|s^e, g^e)) - 0]^2 \tag{10}$$

This objective forces the higher policy subgoal predictions to be closer to the subgoal predictions given in the dataset $\mathcal{D}_g$. For brevity, we can refer to this IRL objective function as $J_D(\theta_H, \epsilon_H)$. The discriminator $\mathbb{D}^H_\epsilon$ creates a natural curriculum for regularizing higher level policy by assigning the value 1 to the predicted subgoals that are closer to the subgoals from dataset D_g, and 0 otherwise. As the training continues, the discriminator improves, and it regularizes the higher policy to predict achievable subgoals while still making reasonable progress towards achieving the final goal. In next section, we explain how we use this IRL objective as regularization objective with the RL objective.

3.5 POLICY OPTIMIZATION

The higher level policy is trained to produce subgoals, which when fed into the lower level primitive, maximize the sum of future discounted rewards for our task using off-policy reinforcement learning. Here, π^L_θ is the current lower primitive, s_t is the state at time t, T is the task horizon and g is the sampled goal for the current episode. For brevity, we can refer to this objective function as J^H_D and J^L_D for upper and lower policies, which depend on parmeters (θ_H, ϵ_H) and (θ_L, ϵ_L) respectively. To make the high level policy learning more sample efficient, we use the IRL objective, that leverages the primitive-parsed dataset $\mathcal{D}_g$. Therefore, the high level policy is trained by optimizing

$$\max_{\theta_H} J^H_{\theta_H} + \lambda(\min_{\epsilon_H} J^H_D(\theta_H, \epsilon_H)) \tag{11}$$

Whereas, the lower level primitive is trained by optimizing,

$$\max_{\theta_L} J^L_{\theta_L} + \lambda(\min_{\epsilon_L} J^L_D(\theta_L, \epsilon_L)) \tag{12}$$

The lower policy is regularized using primitive expert demonstration dataset, and the upper level is optimized using subgoal transition dataset populated using PIP. λ is the regularization weight hyperparameter for the IRL objective. When $\lambda = 0$, the method reduces to HRL policy with no higher level policy regularization. When λ is too high, the method might overfit to the expert demonstration dataset. We perform ablation analysis to choose λ in our experiments as shown in Appendix A. The CRISP algorithm is shown in Algorithm 0

Algorithm 2 CRISP

Require: Expert state demonstrations D, hyperparameter u

Initialize higher level subgoal transition dataset $D_g = \{\}$
for epoch $i = 1 \dots N$ **do**
    if $i\%u == 0$ **then**
        Clear D_g
        Populate D_g by relabeling D using PIP
    for j = 1 **to** $T - 1$ **do**
        Collect off policy experience using π_H and π_L
    Update lower primitive via Soft Actor Critic (SAC) and IRL(Eq 12)
    Sample transitions from D_g
    Update higher policy via SAC and IRL(Eq 11)

4 Related Work

Learning effective hierarchies of policies has garnered substantial research interest in RL Barto & Mahadevan (2003); Sutton et al. (1999); Parr & Russell (1998); Dietterich (1999). Options framework Sutton et al. (1999); Bacon et al. (2016); Harutyunyan et al. (2017); Harb et al. (2017); Harutyunyan et al. (2019); Klissarov et al. (2017) learns temporally extended macro actions, and a termination function for solving long horizon tasks. However, these approaches run into degenerate solutions in absence of proper regularization, where a sub-policy either terminates after each step, or runs for the entire episode. In goal-conditioned learning, some previous approaches restrict the search space by greedily solving for specific goals Kaelbling (1993); Foster & Dayan (2002). This approach has also been extended to hierarchical RL Wulfmeier et al. (2019; 2020); Ding et al. (2019). HIRO Nachum et al. (2018) and HRL with hindsight Levy et al. (2017) approaches deal with the non-stationarity issue in hierarchical learning by relabeling transition data for training goal-conditioned policies, where the higher level predicts subgoals for the lower primitive. In contrast, our method deals with non-stationarity by regularizing the higher policy with imitation learning to provide a curriculum of achievable subgoals to the lower primitive. Our approach is inspired from curriculum learning Bengio et al. (2009), where the task difficulty for the lower primitive gradually increases in complexity, thereby amortizing non-stationarity.

Previous approaches that leverage expert demonstrations have shown impressive results Nair et al. (2017); Rajeswaran et al. (2017); Hester et al. (2017). Expert demonstrations have also been used to bootstrap option learning Krishnan et al. (2017b); Fox et al. (2017); Shankar & Gupta (2020); Kipf et al. (2019). Other approaches use imitation learning to bootstrap hierarchical approaches in complex task domains Shiarlis et al. (2018); Krishnan et al. (2017a; 2019); Kipf et al. (2019). Relay Policy Learning (RPL) Gupta et al. (2019) parses uses simple fixed window based approach for parsing expert demonstrations to generate subgoal transition dataset for training higher level policy. However, fixed parsing based approaches might predict subgoals that are either too hard for the lower level primitive, in which case the higher level is cursed with ambiguous extrinsic reward signal, or too easy subgoals, in which case the higher level is forced to do most of the heavy-lifting for solving the task. In contrast, our data relabeling technique PIP segments expert demonstration trajectories into *meaningful* subtasks, without requiring an external expert. Our adaptive parsing approach considers the limited goal reaching ability of lower primitive, and is therefore able to produce much better subgoals.

5 Experiments

For experimental analysis, we considered complex long horizon tasks with continuous state and action spaces. We perform experiments on the following two Mujoco Todorov et al. (2012) environments: (i) robotic maze navigation, and (ii) robotic pick and place environment. These environments employ a 7-DoF robotic arm to perform maze navigation and robotic manipulation. We empirically show the performance comparison of our approach with various baseline methods in challenging task domains.

5.1 COMPARATIVE ANALYSIS

Here, we enlist the baseline methods for comparison, and explain the rationale.

- **Relay Policy Learning (RPL)** Gupta et al. (2019) parses subgoals from expert state demonstrations using a fixed window approach. We use this baseline to highlight the advantage of adaptive parsing of subgoals compared to fixed window parsing. We perform extensive search for the window size hyper-parameters in RPL for each environment, which we provide in appendix A.
- **Hierarchical 2-level policy (Hier)** denotes a hierarchical policy where the high level policy is trained using only reinforcement learning, and the lower level policy is trained using RL and IRL using primitive expert demonstrations. We use it to show the importance of curriculum based subgoal generation and consequent IRL based regularization on the higher level policy. The hierarchical 2-level policy where both the upper and lower levels are trained using only RL failed to show good performance compared to other methods. Hence, we do not include it in the baseline comparisons.
- **Discriminator Actor Critic (DAC)** Kostrikov et al. (2018) uses IRL to learn a single-level policy using low level expert demonstrations $\mathcal{D}$. Using this baseline, we demonstrate the advantage of using hierarchy, curriculum based subgoal generation and consequent IRL based regularization in our approach.

5.2 ROBOTIC MAZE NAVIGATION ENVIRONMENT

Here we provide details about the maze navigation environment, its implementation and results.

5.2.1 ENVIRONMENT SETUP

In this environment, a 7-DOF robotic arm gripper navigates across random four room mazes. The gripper arm is kept closed and the positions of walls and gates are randomly generated. The height of gripper is kept fixed at table height, and it has to navigate across the maze to the goal position(shown as red sphere). The maximum task horizon T is kept at 225 timesteps, and the lower primitive is allowed to execute for $c = 15$ timesteps.

5.2.2 IMPLEMENTATION DETAILS

The following implementation details refer to both the higher and lower level polices, unless otherwise explicitly stated. The state and action spaces in the environment are continuous. The actor, critic and discriminator networks are formulated as 3 layer fully connected neural networks with 512 neurons in each layer. The state is represented as the vector $[p, \mathcal{M}]$, where p is current gripper position and $\mathcal{M}$ is the sparse maze array. The higher level policy input is thus a concatenated vector $[p, \mathcal{M}, g]$, where g is the target goal position, whereas the lower level policy input is concatenated vector $[p, \mathcal{M}, s_g]$, where s_g is the sub-goal provided by the higher level policy. The current position of the gripper is the current achieved goal. The sparse maze array $\mathcal{M}$ is a discrete $2D$ one-hot vector array, where 1 represents presence of a wall block, and 0 absence. In our experiments, the size of p and $\mathcal{M}$ are kept to be 3 and 110 respectively. The upper level predicts subgoal s_g, hence the higher level policy action space dimension is the same as the dimension of goal space. The lower primitive action a which is directly executed on the environment, is a 4 dimensional vector with every dimension $a_i \in [0, 1]$. The first 3 dimensions provide offsets to be scaled and added to gripper position for moving it to the intended position. The last dimension provides gripper control(0 implies a fully closed gripper, 0.5 implies a half closed gripper and 1 implies a fully open gripper). While training our hierarchical approach, we select 100 randomly generated mazes each for training, testing and validation. Each experiment is run on 4 parallel workers. We use off-policy Soft Actor Critic Haarnoja et al. (2018b) algorithm for optimizing RL objective in our experiments. We keep the regularization weight hyperparameter as $\lambda = 0.0078$ in our experiments. The hyperparameter u which is the number of training iterations after which the replay buffer is flushed and re-populated is set as 200. The experiments are run for $2.93e6$ timesteps.

Table 1: Success rates: Maze navigation and Pick and place environment

Method	Maze Navigation	Pick and Place
CRISP-IRL	**0.65** $\pm$ **0.07**	**0.66** $\pm$ **0.02**
CRISP-BC	0.53 $\pm$ 0.02	**0.73** $\pm$ **0.01**
RPL	0.57 $\pm$ 0.035	0.02 $\pm$ 0.01
Hier	0.52 $\pm$ 0.08	0.02 $\pm$ 0.02
DAC	0.27 $\pm$ 0.06	0.38 $\pm$ 0.07

5.2.3 RESULTS

In Table 1, we report the success rate performance of the proposed methods and other baselines in the maze navigation and pick and place environments averaged over 3 seeds. CRISP-IRL is CRISP with Inverse RL(IRL) regularization objective, and CRISP-BC is CRISP with behavior cloning(BC) regularization. While training and testing, we evaluate success rates over $N = 100$ random episodic rollouts. Since the test mazes are randomly generated, the performance also proves a measure of generalization capability of our proposed approach. From the results, it is clear that the proposed approach CRISP-IRL outperforms the baselines, and demonstrates impressive generalization, compared to other methods.

5.3 ROBOTIC PICK AND PLACE ENVIRONMENT

Here we provide details about the pick and place environment, its implementation and results.

5.3.1 ENVIRONMENT SETUP

In this environment, a 7-DOF robotic arm gripper has to pick a square block and bring/place it to a goal position. We set the goal position(shown as red sphere) slightly higher than table height. The maximum task horizon T is kept at 225 timesteps, and the lower primitive is allowed to execute for $c = 15$ timesteps. The higher level thus predicts a subgoal s_g to the lower primitive after every $c = 15$ timesteps. In this complex task, the gripper has to navigate to the block, close the gripper to hold the object, and then bring the object to the desired goal position to succeed. Since this is a long horizon complex task, this provides a good test bed for testing the intuitive benefits of our hierarchical framework(temporal abstraction and increased exploration). We provide the success rate results and compare with baseline algorithms in section 5.3.3

5.3.2 IMPLEMENTATION DETAILS

In this continuous state and action spaces environment, the actor, critic, and discriminator networks are formulated as 3 layer fully connected networks with 512 neurons in each layer. The state is represented as the vector $[p, o, q, e]$, where p is current gripper position, o is the position of the block object placed on the table, q is the relative position of the block with respect to the gripper, and e consists of linear and angular velocities of the gripper and the block object. The higher level policy input is thus a concatenated vector $[p, o, q, e, g]$, where g is the target goal position. The lower level policy input is concatenated vector $[p, o, q, e, s_g]$, where s_g is the sub-goal provided by the higher level policy. The current position of the block object is the current achieved goal. In our experiments, the sizes of p, o, q, e are kept to be 3, 3, 3 and 11 respectively. The upper level predicts subgoal s_g, hence the higher level policy action space dimension is the same as the dimension of the goal space. The lower primitive action a which is directly executed on the environment, is a 4 dimensional vector with every dimension $a_i \in [0, 1]$. The first 3 dimensions provide gripper position offsets, and the last dimension provides gripper control(0 implies a fully closed gripper and 1 implies a fully open gripper). While training our hierarchical approach, the position of block object and goal are randomly generated(block is always initialized on the table, and goal is always above the table at a fixed height). Each experiment is run on 4 parallel workers. We use off-policy Soft Actor Critic Haarnoja et al. (2018b) algorithm for the RL objective in our experiments. We keep the regularization weight hyperparameter as $\lambda = 0.005$ in our experiments. The hyperparameter u which is the number of training iterations after which the replay buffer is flushed and re-populated is set as 100. The experiments are run for $6.75e6$ timesteps.

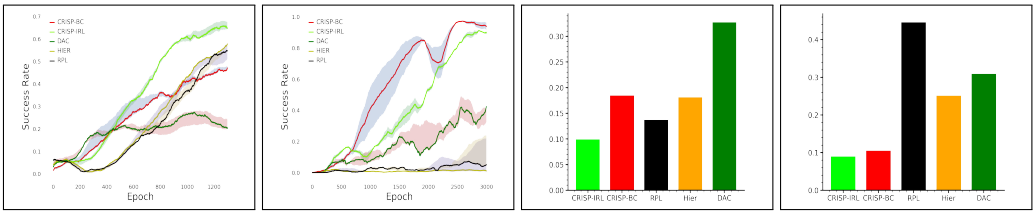

Figure 3: In columns 1 and 2, the success rate plots show performance comparison between our method and baselines in maze navigation(Column 1) and pick and place(Column 2). Columns 3 and 4 plots compare the methods via distance metric(average distance between achieved goal and final goal in 100 episodic rollouts) in room maze navigation (Column 3) and pick and place (Column 4). As can be seen, our methods show impressive performance compared to the baseline approaches.

5.3.3 RESULTS

In Table 1, we report the success rate performance of the proposed methods and other baselines, averaged over 3 seeds. While training and testing, we evaluate success rates over $N = 100$ random episodic rollouts. In complex long horizon tasks, hierarchical methods benefit from increased exploration Nachum et al. (2019). Indeed, from Table 1 it is apparent that CRISP-BC and CRISP-IRL clearly outperform the baselines by a large margin. This provides convincing evidence that stable hierarchical learning indeed demonstrate better performance on complex long horizon tasks.

5.4 ABLATIVE STUDIES

To elucidate the importance of various constituent design choices in our proposed approach, we show the training curves success rate comparison of our methods with baselines in Figure 3 columns 1 and 2. We compare our proposed approach with *RPL* method to demonstrate the advantage of adaptive parsing over fixed window parsing, segmenting the task into meaningful subtasks based on lower primitive, and using curriculum of subgoals for evolving lower primitive. The comparison with *Hier* method shows advantage of curriculum based subgoal regularization using imitation learning. Finally, *DAC* highlights the importance of using hierarchy and curriculum based subgoal regularization using imitation learning. CRISP shows faster convergence and stable learning, when compared to other approaches. In Figure 3 columns 3 and 4, we compare the methods in terms of distance between final achieved goal and desired goal averaged over 100 episodic rollouts. This metric gives an idea of how accurately an approach solves the task. CRISP clearly demonstrates better accuracy in reaching the final goal.

For RPL experiments, we choose the window size hyper-parameter c by running multiple RPL experiments for different values of c. The experiments are shown in Figure 4 in Appendix A. After experiments, tn maze navigation and pick and place environments, c is set to 4 and 8 respectively.

For choosing the imitation learning weight hyperparameter λ, we performed ablation analysis for various values of λ. The experiments are shown in Figure 5 in Appendix A.

6 DISCUSSION AND FUTURE WORK

We introduce our generally applicable approach CRISP, which is a lower level primitive informed method for efficient hierarchical reinforcement learning. CRISP leverages primitive parsed expert demonstrations and populates higher level subgoal transition dataset to regularize higher level policy using IRL. Our approach employs hierarchical curriculum learning to deal with the curse of non-stationarity. We evaluate our method on complex robotic maze navigation and robotic pick and place manipulation tasks, and demonstrate that it makes substantial gains over its baselines.

However, CRISP assumes the ability to reset the environment to any state from the dataset of expert demonstrations while collecting subgoal dataset using PIP. This assumption can be relaxed by combining CRISP with Eysenbach et al. (2017) that learns a backward controller which tries to reset the environment. While this wasn't explored in the work, it remains an interesting avenue for future work.

REFERENCES

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning, 2020.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel H Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NIPS*, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. *CoRR*, abs/1609.05140, 2016.

Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. 13:2003, 2003.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pp. 271–278, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-274-7.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *CoRR*, cs.LG/9905014, 1999. URL https://arxiv.org/abs/cs/9905014.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/c8d3a760ebab631565f8509d84b3b3f1-Paper.pdf.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning, 2017.

David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49 (2-3):325–346, 2002.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options, 2017.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pp. 1259–1277. PMLR, 2020.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *CoRR*, abs/1910.11956, 2019.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 1851–1860. PMLR, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018b. URL http://arxiv.org/abs/1801.01290.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option : Learning options with a deliberation cost. *CoRR*, abs/1709.04571, 2017. URL http://arxiv.org/abs/1709.04571.

Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowé. Learning with options that terminate off-policy. *CoRR*, abs/1711.03817, 2017. URL http://arxiv.org/abs/1711.03817.

Anna Harutyunyan, Will Dabney, Diana Borsa, Nicolas Heess, Rémi Munos, and Doina Precup. The termination critic. *CoRR*, abs/1902.09996, 2019. URL http://arxiv.org/abs/1902.09996.

Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476.

Leslie Pack Kaelbling. Learning to achieve goals. In *IN PROC. OF IJCAI-93*, pp. 1094–1098. Morgan Kaufmann, 1993.

Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pp. 3418–3428. PMLR, 2019.

Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learnings options end-to-end for continuous action tasks. *CoRR*, abs/1712.00004, 2017. URL http://arxiv.org/abs/1712.00004.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. DDCO: discovery of deep continuous options forrobot learning from demonstrations. *CoRR*, abs/1710.05421, 2017a. URL http://arxiv.org/abs/1710.05421.

Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations, 2017b.

Sanjay Krishnan, Animesh Garg, Richard Liaw, Brijen Thananjeyan, Lauren Miller, Florian T Pokorny, and Ken Goldberg. Swirl: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *The International Journal of Robotics Research*, 38(2-3):126–145, 2019. doi: 10.1177/0278364918784350. URL https://doi.org/10.1177/0278364918784350.

Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *CoRR*, abs/1604.06057, 2016.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015.

Andrew Levy, Robert Platt Jr., and Kate Saenko. Hierarchical actor-critic. *CoRR*, abs/1712.00948, 2017.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016. URL http://arxiv.org/abs/1611.04076.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *CoRR*, abs/1805.08296, 2018.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *CoRR*, abs/1709.10089, 2017.

Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In M. Jordan, M. Kearns, and S. Solla (eds.), *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *CoRR*, abs/1709.10087, 2017.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/schaul15.html`.

T. Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *ICML*, 2020.

Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. TACO: Learning task decomposition via temporal alignment for control. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4654–4663. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/shiarlis18a.html`.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(99)00052-1. URL `https://www.sciencedirect.com/science/article/pii/S0004370299000521`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Matej Vecerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin A. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017.

Markus Wulfmeier, Abbas Abdolmaleki, Roland Hafner, Jost Tobias Springenberg, Michael Neunert, Tim Hertweck, Thomas Lampe, Noah Y. Siegel, Nicolas Heess, and Martin A. Riedmiller. Regularized hierarchical policies for compositional transfer in robotics. *CoRR*, abs/1906.11228, 2019. URL `http://arxiv.org/abs/1906.11228`.

Markus Wulfmeier, Dushyant Rao, Roland Hafner, Thomas Lampe, Abbas Abdolmaleki, Tim Hertweck, Michael Neunert, Dhruva Tirumala, Noah Y. Siegel, Nicolas Heess, and Martin A. Riedmiller. Data-efficient hindsight off-policy option learning. *CoRR*, abs/2007.15588, 2020. URL `https://arxiv.org/abs/2007.15588`.

A APPENDIX

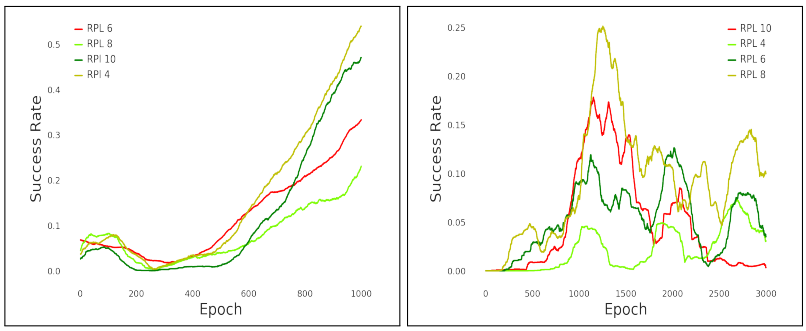

Figure 4: The success rate plots show the performance of RPL for various values of k window size parameter. Column 1 shows experiments on maze navigation environment and column 2 should experiments on pick and place environment

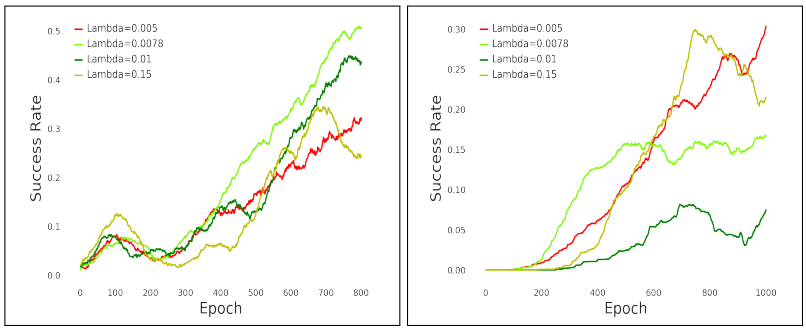

Figure 5: The success rate plots show the performance of CRISP for various values of imitation learning weight parameter λ. Column 1 shows experiments on maze navigation environment and column 2 should experiments on pick and place environment