Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation

Tejas D. Kulkarni*
BCS, MIT
tejask@mit.edu

Karthik R. Narasimhan*
CSAIL, MIT
karthikn@mit.edu

Ardavan Saeedi
CSAIL, MIT
ardavans@mit.edu

Joshua B. Tenenbaum
BCS, MIT
jbt@mit.edu

Abstract

Learning goal-directed behavior in environments with sparse feedback is a major challenge for reinforcement learning algorithms. The primary difficulty arises due to insufficient exploration, resulting in an agent being unable to learn robust value functions. Intrinsically motivated agents can explore new behavior for its own sake rather than to directly solve problems. Such intrinsic behaviors could eventually help the agent solve tasks posed by the environment. We present hierarchical-DQN (h-DQN), a framework to integrate hierarchical value functions, operating at different temporal scales, with intrinsically motivated deep reinforcement learning. A top-level value function learns a policy over intrinsic goals, and a lower-level function learns a policy over atomic actions to satisfy the given goals. h-DQN allows for flexible goal specifications, such as functions over entities and relations. This provides an efficient space for exploration in complicated environments. We demonstrate the strength of our approach on two problems with very sparse, delayed feedback: (1) a complex discrete stochastic decision process, and (2) the classic ATARI game 'Montezuma's Revenge'.

1 Introduction

Learning goal-directed behavior with sparse feedback from complex environments is a fundamental challenge for artificial intelligence. Learning in this setting requires the agent to represent knowledge at multiple levels of spatio-temporal abstractions and to explore the environment efficiently. Recently, non-linear function approximators coupled with reinforcement learning [21, 28, 37] have made it possible to learn abstractions over high-dimensional state spaces, but the task of exploration with sparse feedback still remains a major challenge. Existing methods like Boltzmann exploration and Thomson sampling [45, 32] offer significant improvements over ϵ-greedy, but are limited due to the underlying models functioning at the level of basic actions. In this work, we propose a framework that integrates deep reinforcement learning with hierarchical value functions (h-DQN), where the agent is motivated to solve intrinsic goals (via learning options) to aid exploration. These goals provide for efficient exploration and help mitigate the sparse feedback problem. Additionally, we observe that goals defined in the space of entities and relations can help significantly constrain the exploration space for data-efficient learning in complex environments.

*Authors contributed equally and listed alphabetically.

Reinforcement learning (RL) formalizes control problems as finding a policy π that maximizes expected future rewards [46]. Value functions $V(s)$ are central to RL, and they cache the utility of any state s in achieving the agent's overall objective. Recently, value functions have also been generalized as $V(s, g)$ in order to represent the utility of state s for achieving a given goal $g \in G$ [47, 34]. When the environment provides delayed rewards, we adopt a strategy to first learn ways to achieve intrinsically generated goals, and subsequently learn an optimal policy to chain them together. Each of the value functions $V(s, g)$ can be used to generate a policy that terminates when the agent reaches the goal state g. A collection of these policies can be hierarchically arranged with temporal dynamics for learning or planning within the framework of semi-Markov decision processes [48, 49]. In high-dimensional problems, these value functions can be approximated by neural networks as $V(s, g; \theta)$.

We propose a framework with hierarchically organized deep reinforcement learning modules working at different time-scales. The model takes decisions over two levels of hierarchy – (a) the top level module (*meta-controller*) takes in the state and picks a new goal, (b) the lower-level module (*controller*) uses both the state and the chosen goal to select actions either until the goal is reached or the episode is terminated. The *meta-controller* then chooses another goal and steps (a-b) repeat. We train our model using stochastic gradient descent at different temporal scales to optimize expected future intrinsic (*controller*) and extrinsic rewards (*meta-controller*). We demonstrate the strength of our approach on problems with long-range delayed feedback: (1) a discrete stochastic decision process with a long chain of states before receiving optimal extrinsic rewards and (2) a classic ATARI game ('Montezuma's Revenge') with even longer-range delayed rewards where most existing state-of-art deep reinforcement learning approaches fail to learn policies in a data-efficient manner.

2 Literature Review

2.1 Reinforcement Learning with Temporal Abstractions

Learning and operating over different levels of temporal abstraction is a key challenge in tasks involving long-range planning. In the context of reinforcement learning [1], Sutton et al.[48] proposed the *options* framework, which involves abstractions over the space of actions. At each step, the agent chooses either a one-step "primitive" action or a "multi-step" action policy (option). Each option defines a policy over actions (either primitive or other options) and can be terminated according to a stochastic function β. Thus, the traditional MDP setting can be extended to a semi-Markov decision process (SMDP) with the use of options. Recently, several methods have been proposed to learn options in real-time by using varying reward functions [49] or by composing existing options [42]. Value functions have also been generalized to consider goals along with states [34]. This universal value function $V(s, g; \theta)$ provides an universal option that approximately represents optimal behavior towards the goal g. Our work is inspired by these papers and builds upon them.

There has also been a lot of work on option discovery in the tabular value function setting [26, 38, 25, 27]. In more recent work, Machado et al. [24] presented an option discovery algorithm where the agent is encouraged to explore regions that were previously out of reach. However, option discovery where non-linear state approximations are required is still an open problem.

Other related work for hierarchical formulations include the model of Dayan and Hinton [6] which consisted of "managers" taking decisions at various levels of granularity, percolating all the way down to atomic actions made by the agent. The MAXQ framework [7] built up on this work to decompose the value function of an MDP into combinations of value functions of smaller constituent MDPs, as did Guestrin et al.[17] in their factored MDP formulation. Hernandez-Gardiol and Mahadevan [19] combined hierarchical RL with a variable length short-term memory of high-level decisions.

In our work, we propose a scheme for temporal abstraction that involves simultaneously learning options and a control policy to compose options in a deep reinforcement learning setting. Our approach does not use separate Q-functions for each option, but instead treats the option as part of the input, similar to [34]. This has two advantages: (1) there is shared

learning between different options, and (2) the model is potentially scalable to a large number of options.

2.2 Intrinsically motivated RL

The nature and origin of 'good' intrinsic reward functions is an open question in reinforcement learning. Singh et al.[41] explored agents with intrinsic reward structures in order to learn generic options that can apply to a wide variety of tasks. Using a notion of "salient events" as sub-goals, the agent learns options to get to such events. In another paper, Singh et al.[40] take an evolutionary perspective to optimize over the space of reward functions for the agent, leading to a notion of extrinsically and intrinsically motivated behavior. In the context of hierarchical RL, Goel and Huber [13] discuss a framework for subgoal discovery using the structural aspects of a learned policy model. Şimşek et al. [38] provide a graph partioning approach to subgoal identification.

Schmidhuber [36] provides a coherent formulation of intrinsic motivation, which is measured by the improvements to a predictive world model made by the learning algorithm. Mohamed and Rezende [29] have recently proposed a notion of intrinsically motivated learning within the framework of mutual information maximization. Frank et al. [11] demonstrate the effectiveness of artificial curiosity using information gain maximization in a humanoid robot.

2.3 Object-based RL

Object-based representations [8, 4] that can exploit the underlying structure of a problem have been proposed to alleviate the *curse of dimensionality* in RL. Diuk et al.[8] propose an *Object-Oriented MDP*, using a representation based on objects and their interactions. Defining each state as a set of value assignments to all possible relations between objects, they introduce an algorithm for solving deterministic object-oriented MDPs. Their representation is similar to that of Guestrin et al.[16], who describe an object-based representation in the context of planning. In contrast to these approaches, our representation does not require explicit encoding for the relations between objects and can be used in stochastic domains.

2.4 Deep Reinforcement Learning

Recent advances in function approximation with deep neural networks have shown promise in handling high-dimensional sensory input. Deep Q-Networks and its variants have been successfully applied to various domains including Atari games [28] and Go [37], but still perform poorly on environments with sparse, delayed reward signals. Strategies such as prioritized experience replay [35] and bootstrapping [32] have been proposed to alleviate the problem of learning from sparse rewards. These approaches yield significant improvements over prior work but struggle when the reward signal has a long delayed horizon. This is because the exploration strategy is not sufficient for the agent to obtain the required feedback.

2.5 Cognitive Science and Neuroscience

The nature and origin of intrinsic goals in humans is a thorny issue but there are some notable insights from existing literature. There is converging evidence in developmental psychology that human infants, primates, children, and adults in diverse cultures base their core knowledge on certain cognitive systems including – entities, agents and their actions, numerical quantities, space, social-structures and intuitive theories [43, 23]. Even newborns and infants seem to represent the visual world in terms of coherent visual entities, centered around spatio-temporal principles of cohesion, continuity, and contact. They also seem to explicitly represent other agents, with the assumption that an agent's behavior is goal-directed and efficient. Infants can also discriminate relative sizes of objects, relative distances and higher order numerical relations such as the ratio of object sizes. During curiosity-driven activities, toddlers use this knowledge to generate intrinsic goals such as building physically stable block structures. In order to accomplish these goals, toddlers seem to construct sub-goals in the space of their core knowledge, such as – putting a heavier entity *on top of* (relation) a lighter entity in order to build tall blocks.

Knowledge of space can also be utilized to learn a hierarchical decomposition of spatial environments, where the bottlenecks between different spatial groupings correspond to sub-goals. This has been explored in neuroscience with the successor representation, which represents a value function in terms of the expected future state occupancy. Decomposition of the successor representation yields reasonable sub-goals for spatial navigation problems [5, 12, 44]. Botvinick et al.[3] have written a general overview of hierarchical reinforcement learning in the context of cognitive science and neuroscience.

3 Model

Consider a Markov decision process (MDP) represented by states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, and transition function $\mathcal{T} : (s, a) \rightarrow s'$. An agent operating in this framework receives a state s from the external environment and can take an action a, which results in a new state s'. We define the extrinsic reward function as $\mathcal{F} : (s) \rightarrow \mathbb{R}$. The objective of the agent is to maximize this function over long periods of time. For example, this function can take the form of the agent's survival time or score in a game.

Agents Effective exploration in MDPs is a significant challenge in learning good control policies. Methods such as ϵ-greedy are useful for local exploration but fail to provide impetus for the agent to explore different areas of the state space. In order to tackle this, we utilize a notion of *goals* $g \in \mathcal{G}$, which provide intrinsic motivation for the agent. The agent focuses on setting and achieving sequences of goals in order to maximize cumulative extrinsic reward.

We use the temporal abstraction of *options* [48] to define policies π_g for each goal g. The agent learns these option policies simultaneously along with learning the optimal sequence of goals to follow. In order to learn each π_g, the agent also has a critic, which provides *intrinsic rewards*, based on whether the agent is able to achieve its goals (see Figure 1).

Temporal Abstractions As shown in Figure 1, the agent uses a two-stage hierarchy consisting of a *controller* and a *meta-controller*. The meta-controller receives state s_t and chooses a goal $g_t \in \mathcal{G}$, where $\mathcal{G}$ denotes the set of all possible current goals. The controller then selects an action a_t using s_t and g_t. The goal g_t remains in place for the next few time steps either until it is achieved or a terminal state is reached. The internal critic is responsible for evaluating whether a goal has been reached and providing an appropriate reward $r_t(g)$ to the controller. The objective function for the controller is to maximize cumulative intrinsic reward: $R_t(g) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}(g)$. Similarly, the objective of the meta-controller is to optimize the cumulative extrinsic reward $F_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} f_{t'}$, where f_t are reward signals received from the environment.

One can also view this setup as similar to optimizing over the space of optimal reward functions to maximize fitness [39]. In our case, the reward functions are dynamic and temporally dependent on the sequential history of goals. Figure 1 provides an illustration of the agent's use of the hierarchy over subsequent time steps.

Deep Reinforcement Learning with Temporal Abstractions

We use the Deep Q-Learning framework [28] to learn policies for both the controller and the meta-controller. Specifically, the controller estimates the following Q-value function:

$$\begin{aligned} Q_1^*(s, a; g) &= \max_{\pi_{ag}} \mathrm{E}[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} \mid s_t = s, a_t = a, g_t = g, \pi_{ag}] \\ &= \max_{\pi_{ag}} \mathrm{E}[r_t + \gamma \max_{a_{t+1}} Q_1^*(s_{t+1}, a_{t+1}; g) \mid s_t = s, a_t = a, g_t = g, \pi_{ag}] \end{aligned} \tag{1}$$

where g is the agent's goal in state s and $\pi_{ag} = P(a|s, g)$ is the action policy.

Similarly, for the meta-controller, we have:

$$Q_2^*(s, g) = \max_{\pi_g} \mathrm{E}[\sum_{t'=t}^{t+N} f_{t'} + \gamma \max_{g'} Q_2^*(s_{t+N}, g') \mid s_t = s, g_t = g, \pi_g] \tag{2}$$

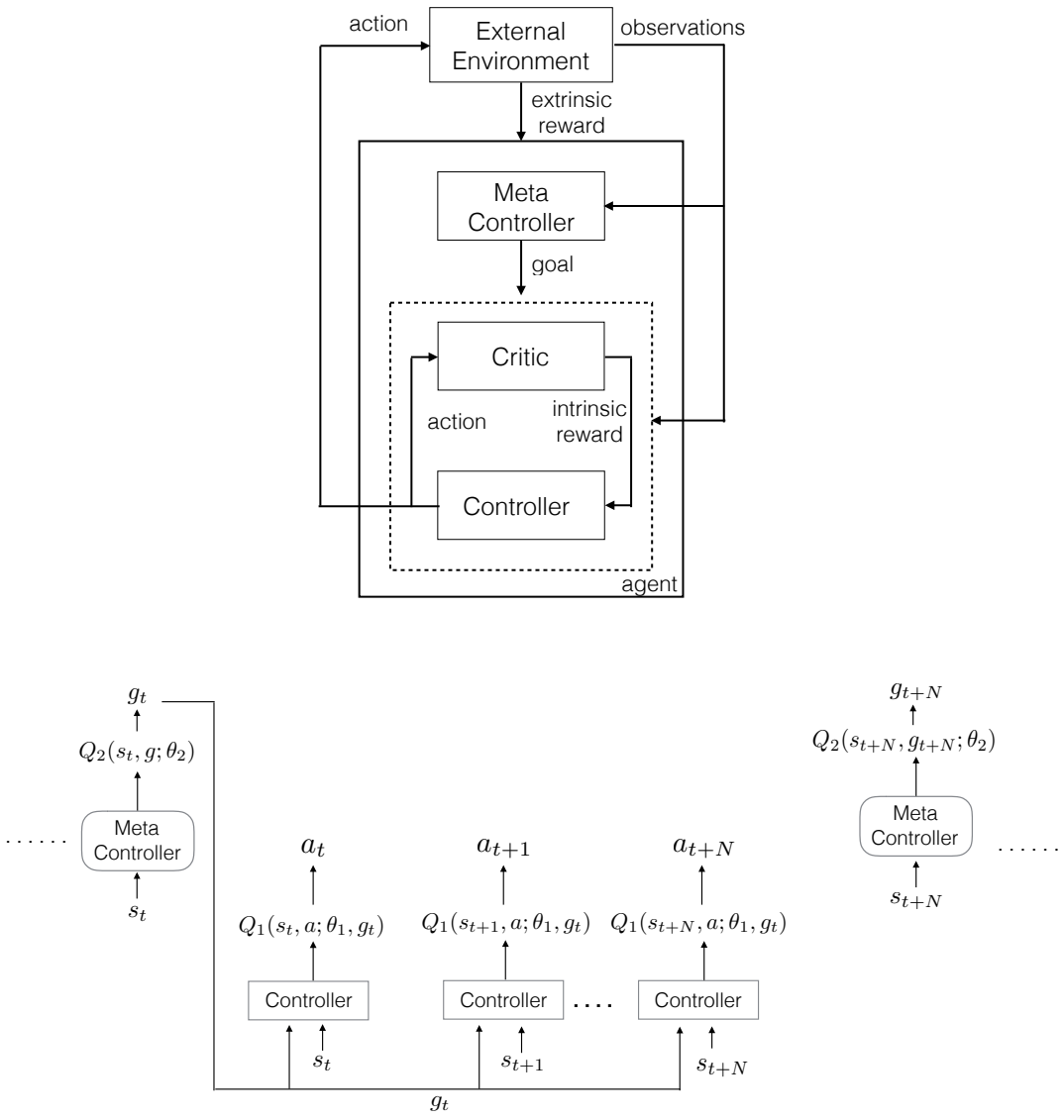


Figure 1: **Overview:** The agent produces actions and receives sensory observations. Separate deep-Q networks are used inside the *meta-controller* and *controller*. The meta-controller that looks at the raw states and produces a policy over goals by estimating the value function $Q_2(s_t, g_t; \theta_2)$ (by maximizing expected future extrinsic reward). The controller takes in states and the current goal, and produces a policy over actions by estimating the value function $Q_2(s_t, a_t; \theta_1, g_t)$ to solve the predicted goal (by maximizing expected future intrinsic reward). The internal critic checks if goal is reached and provides an appropriate intrinsic reward to the controller. The controller terminates either when the episode ends or when g is accomplished. The meta-controller then chooses a new g and the process repeats.

where N denotes the number of time steps until the controller halts given the current goal, g' is the agent's goal in state s_{t+N}, and $\pi_g = P(g|s)$ is the policy over goals. It is important to note that the transitions (s_t, g_t, f_t, s_{t+N}) generated by Q_2 run at a slower time-scale than the transitions $(s_t, a_t, g_t, r_t, s_{t+1})$ generated by Q_1.

We can represent $Q^*(s, g) \approx Q(s, g; \theta)$ using a non-linear function approximator with parameters θ, called a deep Q-network (DQN). Each $Q \in \{Q_1, Q_2\}$ can be trained by minimizing corresponding loss functions – $L_1(\theta_1)$ and $L_2(\theta_2)$. We store experiences (s_t, g_t, f_t, s_{t+N}) for Q_2 and $(s_t, a_t, g_t, r_t, s_{t+1})$ for Q_1 in disjoint memory spaces $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. The loss function for Q_1 can then be stated as:

$$L_1(\theta_{1,i}) = \mathrm{E}_{(s,a,g,r,s')\sim D_1}[(y_{1,i} - Q_1(s, a; \theta_{1,i}, g))^2], \tag{3}$$

where i denotes the training iteration number and $y_{1,i} = r + \gamma \max_{a'} Q_1(s', a'; \theta_{1,i-1}, g)$.

Following [28], the parameters $\theta_{1,i-1}$ from the previous iteration are held fixed when optimising the loss function. The parameters θ_1 can be optimized using the gradient:

$$\nabla_{\theta_{1,i}} L_1(\theta_{1,i})$$
$$= \mathrm{E}_{(s,a,r,s'\sim D_1)}\left[\Big(r + \gamma \max_{a'} Q_1(s', a'; \theta_{1,i-1}, g) - Q_1(s, a; \theta_{1,i}, g)\Big)\nabla_{\theta_{1,i}} Q_1(s, a; \theta_{1,i}, g)\Big)\right]$$

The loss function L_2 and its gradients can be derived using a similar procedure.

Learning Algorithm We learn the parameters of h-DQN using stochastic gradient descent at different time scales – experiences (or transitions) from the controller are collected at every time step but experiences from meta-controller are only collected when the controller terminates (i.e. when a goal is re-picked or the episode ends). Each new goal g is drawn in an ϵ-greedy fashion (Algorithms 1 & 2) with the exploration probability ϵ_2 annealed as learning proceeds (from a starting value of 1).

In the controller, at every time step, an action is drawn with a goal using the exploration probability $\epsilon_{1,g}$ which is dependent on the current empirical success rate of reaching g. The model parameters (θ_1, θ_2) are periodically updated by drawing experiences from replay memories $\mathcal{D}_1$ and $\mathcal{D}_2$), respectively (see Algorithm 3).

4 Experiments

We perform experiments on two different domains involving delayed rewards. The first is a discrete-state MDP with stochastic transitions, and the second is an ATARI 2600 game called 'Montezuma's Revenge'.

4.1 Discrete stochastic decision process

Game Setup We consider a stochastic decision process where the extrinsic reward depends on the history of visited states in addition to the current state. We selected this task in order to demonstrate the importance of intrinsic motivation for exploration in such environments.

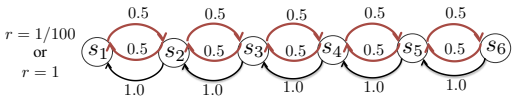


Figure 2: A stochastic decision process where the reward at the terminal state s_1 depends on whether s_6 is visited ($r = 1$) or not ($r = 1/100$).

There are 6 possible states and the agent always starts at s_2. The agent moves left deterministically when it chooses *left* action; but the action *right* only succeeds 50% of the time, resulting in a left move otherwise. The terminal state is s_1 and the agent receives the reward of 1 when it first visits s_6 and then s_1. The reward for going to s_1 without visiting s_6 is 0.01. This is a modified version of the MDP in [32], with the reward structure adding complexity to the task. The process is illustrated in Figure 2.

Algorithm 1 Learning algorithm for h-DQN

Initialize experience replay memories $\{\mathcal{D}_1, \mathcal{D}_2\}$ and parameters $\{\theta_1, \theta_2\}$ for the controller and meta-controller respectively.
Initialize exploration probability $\epsilon_{1,g} = 1$ for the controller for all goals g and $\epsilon_2 = 1$ for the meta-controller.
for $i = 1, num\_episodes$ **do**
    Initialize game and get start state description s
    $g \leftarrow \text{EPSGREEDY}(s, \mathcal{G}, \epsilon_2, Q_2)$
    while s is **not** terminal **do**
        $F \leftarrow 0$
        $s_0 \leftarrow s$
        while not (s is terminal **or** goal g reached) **do**
            $a \leftarrow \text{EPSGREEDY}(\{s, g\}, \mathcal{A}, \epsilon_{1,g}, Q_1)$
            Execute a and obtain next state s' and extrinsic reward f from environment
            Obtain intrinsic reward $r(s, a, s')$ from internal critic
            Store transition $(\{s, g\}, a, r, \{s', g\})$ in $\mathcal{D}_1$
            $\text{UPDATEPARAMS}(\mathcal{L}_1(\theta_{1,i}), \mathcal{D}_1)$
            $\text{UPDATEPARAMS}(\mathcal{L}_2(\theta_{2,i}), \mathcal{D}_2)$
            $F \leftarrow F + f$
            $s \leftarrow s'$
        end while
        Store transition (s_0, g, F, s') in $\mathcal{D}_2$
        if s is **not** terminal **then**
            $g \leftarrow \text{EPSGREEDY}(s, \mathcal{G}, \epsilon_2, Q_2)$
        end if
    end while
    Anneal ϵ_2 and adaptively anneal $\epsilon_{1,g}$ using average success rate of reaching goal g.
end for

Algorithm 2 : $\text{EPSGREEDY}(x, \mathcal{B}, \epsilon, Q)$

if random() $< \epsilon$ **then**
    return random element from set $\mathcal{B}$
else
    return $\text{argmax}_{m \in \mathcal{B}} Q(x, m)$
end if

Algorithm 3 : $\text{UPDATEPARAMS}(\mathcal{L}, \mathcal{D})$

Randomly sample mini-batches from $\mathcal{D}$
Perform gradient descent on loss $\mathcal{L}(\theta)$ (cf. (3))

We consider each state as a possible goal for exploration. This encourages the agent to visit state s_6 (whenever it is chosen as a goal) and hence, learn the optimal policy. For each goal, the agent receives a positive intrinsic reward if and only if it reaches the corresponding state.

Results We compare the performance of our approach (without the deep neural networks) with Q-Learning as a baseline (without intrinsic rewards) in terms of the average extrinsic reward gained in an episode. In our experiments, all ϵ parameters are annealed from 1 to 0.1 over 50,000 steps. The learning rate is set to 0.00025. Figure 3 plots the evolution of reward for both methods averaged over 10 different runs. As expected, we see that Q-Learning is unable to find the optimal policy even after 200 epochs, converging to a sub-optimal policy of reaching state s_1 directly to obtain a reward of 0.01. In contrast, our approach with hierarchical Q-estimators learns to choose goals s_4, s_5 or s_6, which statistically lead the agent to visit s_6 before going back to s_1. Therefore, the agent obtains a significantly higher average reward of around 0.13.

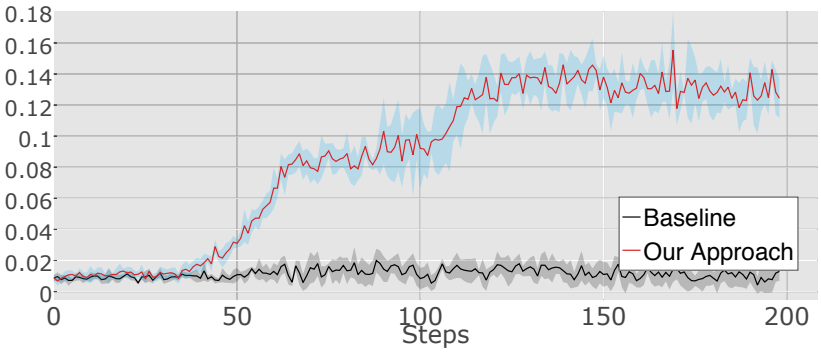


Figure 3: Average reward for 10 runs of our approach compared to Q-learning.

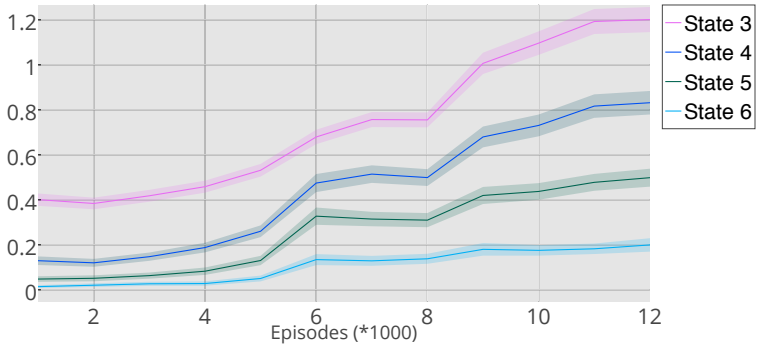


Figure 4: Number of visits (for states s_3 to s_6) averaged over 1000 episodes. The initial state is s_2 and the terminal state is s_1.

Figure 4 illustrates that the number of visits to states s_3, s_4, s_5, s_6 increases with episodes of training. Each data point shows the average number of visits for each state over the last 1000 episodes. This indicates that our model is choosing goals in a way so that it reaches the critical state s_6 more often.

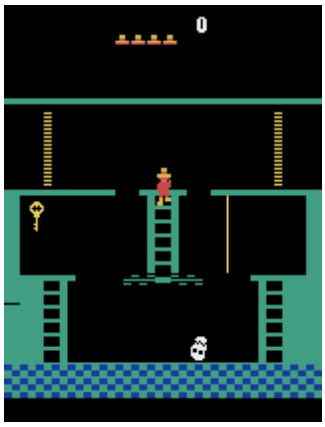
4.2 ATARI game with delayed rewards

Game Description We consider 'Montezuma's Revenge', an ATARI game with sparse, delayed rewards. The game (Figure 5(a)) requires the player to navigate the explorer (in red) through several rooms while collecting treasures. In order to pass through doors (in the top right and top left corners of the figure), the player has to first pick up the key. The player has to then climb down the ladders on the right and move left towards the key, resulting in a long sequence of actions before receiving a reward (+100) for collecting the key. After this, navigating towards the door and opening it results in another reward (+300).

Existing deep RL approaches fail to learn in this environment since the agent rarely reaches a state with non-zero reward. For instance, the basic DQN [28] achieves a score of 0 while even the best performing system, Gorila DQN [30], manages only 4.16 on average.

Setup The agent needs intrinsic motivation to explore meaningful parts of the scene before it can learn about the advantage of getting the key for itself. Inspired by the developmental psychology literature [43] and object-oriented MDPs [8], we use entities or objects in the scene to parameterize goals in this environment. Unsupervised detection of objects in visual scenes is an open problem in computer vision, although there has been recent progress in obtaining objects directly from image or motion data [10, 9, 14]. In this work, we built a custom object detector that provides plausible object candidates. The controller and

(a)



(b)

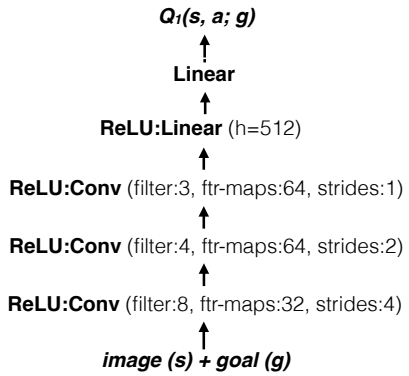


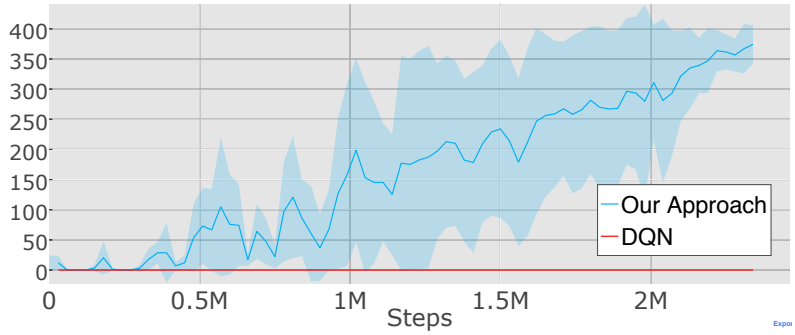
Figure 5: (a) A sample screen from the ATARI 2600 game called 'Montezuma's Revenge'. (b) **Architecture**: DQN architecture for the controller (Q_1). A similar architecture produces Q_2 for the meta-controller (without goal as input). In practice, both these networks could share lower level features but we do not enforce this.

meta-controller are convolutional neural networks (see Figure 5(b)) that learn representations from raw pixel data. We use the Arcade Learning Environment [2] to perform experiments.

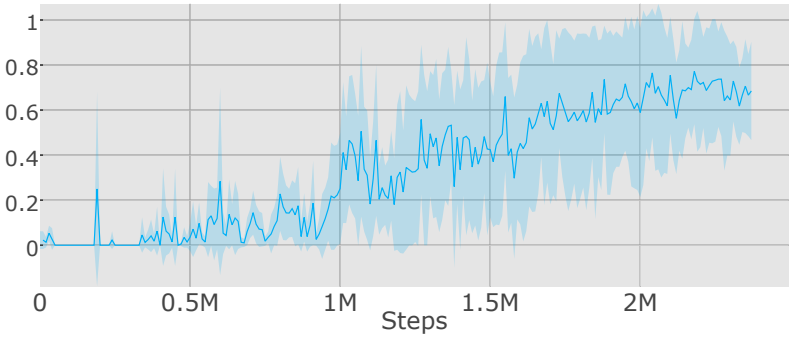
The internal critic is defined in the space of $\langle entity_1, relation, entity_2 \rangle$, where $relation$ is a function over configurations of the entities. In our experiments, the agent is free to choose any $entity_2$. For instance, the agent is deemed to have completed a goal (and receives a reward) if the agent entity *reaches* another entity such as the *door*. Note that this notion of relational intrinsic rewards can be generalized to other settings. For instance, in the ATARI game 'Asteroids', the agent could be rewarded when the bullet **reaches** the asteroid or if simply the ship never **reaches** an asteroid. In the game of 'Pacman', the agent could be rewarded if the pellets on the screen are **reached**. In the most general case, we can potentially let the model evolve a parameterized intrinsic reward function given entities. We leave this for future work.

Model Architecture and Training As shown in Figure 5b, the model consists of stacked convolutional layers with rectified linear units (ReLU). The input to the meta-controller is a set of four consecutive images of size 84×84. To encode the goal output from the meta-controller, we append a binary mask of the goal location in image space along with the original 4 consecutive frames. This augmented input is passed to the controller. The experience replay memories $\mathcal{D}_1$ and $\mathcal{D}_2$ were set to be equal to 1E6 and 5E4 respectively. We set the learning rate to be 2.5E$-$4, with a discount rate of 0.99. We follow a two phase training procedure – (1) In the first phase, we set the exploration parameter ϵ_2 of the meta-controller to 1 and train the controller on actions. This effectively leads to pre-training the controller so that it can learn to solve a subset of the goals. (2) In the second phase, we jointly train the controller and meta-controller.

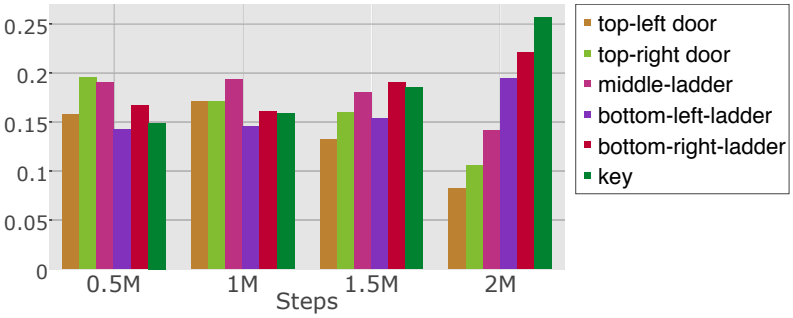
Results Figure 6(a) shows reward progress from the joint training phase from which it is evident that the model starts gradually learning to both reach the key and open the door to get a reward of around +400 per episode. As shown in Figure 6(b), the agent learns to choose the key more often as training proceeds and is also successful at reaching it. As training proceeds, we observe that the agent first learns to perform the simpler goals (such as reaching the right door or the middle ladder) and then slowly starts learning the 'harder' goals such as the key and the bottom ladders, which provide a path to higher rewards. Figure 6(c) shows the evolution of the success rate of goals that are picked. At the end of training, we can see that the 'key', 'bottom-left-ladder' and 'bottom-right-ladders' are chosen increasingly more often. In order to scale-up to solve the entire game, several key ingredients are missing such as – automatic discovery of objects from videos to aid goal parametrization we considered, a flexible short-term memory, ability to intermittently terminate ongoing options.



(a) Total extrinsic reward



(b) Success ratio for reaching the goal 'key'



(c) Success % of different goals over time

Figure 6: **Results on Montezuma's Revenge:** These plots depict the joint training phase of the model. As described in Section 4.2, the first training phase pre-trains the lower level controller for about 2.3 million steps. The joint training learns to consistently get high rewards after additional 2 million steps as shown in **(a)**. **(b) Goal success ratio:** The agent learns to choose the key more often as training proceeds and is successful at achieving it. **(c) Goal statistics:** During early phases of joint training, all goals are equally preferred due to high exploration but as training proceeds, the agent learns to select appropriate goals such as the key and bottom-left door.

We also show some screen-shots from a test run with our agent (with epsilon set to 0.1) in Figure 7, as well as a sample animation of the run.[1]

[1]Sample trajectory of a run on 'Montezuma's Revenge' – `https://goo.gl/3Z64Ji`

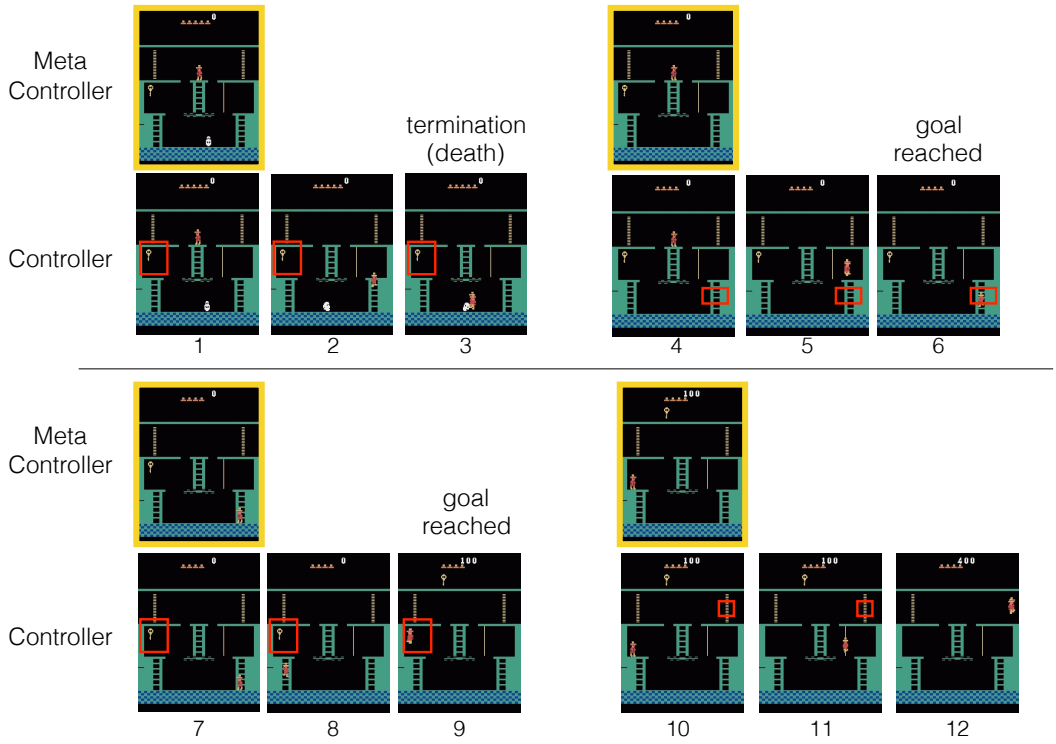


Figure 7: **Sample gameplay by our agent on Montezuma's Revenge:** The four quadrants are arranged in a temporally coherent manner (top-left, top-right, bottom-left and bottom-right). At the very beginning, the meta-controller chooses key as the goal (illustrated in *red*). The controller then tries to satisfy this goal by taking a series of low level actions (only a subset shown) but fails due to colliding with the skull (the episode terminates here). The meta-controller then chooses the bottom-right ladder as the next goal and the controller terminates after reaching it. Subsequently, the meta-controller chooses the key and the top-right door and the controller is able to successfully achieve both these goals.

5 Conclusion

We have presented h-DQN, a framework consisting of hierarchical value functions operating at different time scales. Temporally decomposing the value function allows the agent to perform intrinsically motivated behavior, which in turn yields efficient exploration in environments with delayed rewards. We also observe that parameterizing intrinsic motivation in the space of entities and relations provides a promising avenue for building agents with temporally extended exploration. We also plan to explore alternative parameterizations of goals with h-DQN in the future.

The current framework has several missing components including automatically disentangling objects from raw pixels and a short-term memory. The state abstractions learnt by vanilla deep-Q-networks are not structured or sufficiently compositional. There has been recent work [9, 14, 33, 22, 50, 15, 20] in using deep generative models to disentangle multiple factors of variations (objects, pose, location, etc) from pixel data. We hope that our work motivates the combination of deep generative models of images with h-DQN. Additionally, in order to handle longer range dependencies, the agent needs to store a history of previous goals, actions and representations. There has been some recent work in using recurrent networks in conjunction with reinforcement learning [18, 31]. In order to scale-up our approach to harder non-Markovian settings, it will be necessary to incorporate a flexible episodic memory module.

Acknowledgements

We would like to thank Vaibhav Unhelkar, Ramya Ramakrishnan, Sam Gershman, Michael Littman, Vlad Firoiu, Will Whitney, Max Kleiman-Weiner and Pedro Tsividis for critical feedback and discussions. We are grateful to receive support from the Center for Brain, Machines and Minds (NSF STC award CCF - 1231216) and the MIT OpenMind team.

References

[1] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2012.

[3] M. M. Botvinick, Y. Niv, and A. C. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

[4] L. C. Cobo, C. L. Isbell, and A. L. Thomaz. Object focused q-learning for autonomous agents. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1061–1068. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[5] P. Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[6] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–271. Morgan Kaufmann Publishers, 1993.

[7] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)*, 13:227–303, 2000.

[8] C. Diuk, A. Cohen, and M. L. Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247. ACM, 2008.

[9] S. Eslami, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*, 2016.

[10] K. Fragkiadaki, P. Arbelaez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4083–4090. IEEE, 2015.

[11] M. Frank, J. Leitner, M. Stollenga, A. Förster, and J. Schmidhuber. Curiosity driven reinforcement learning for motion planning on humanoids. *Intrinsic motivations and open-ended development in animals, humans, and robots*, page 245, 2015.

[12] S. J. Gershman, C. D. Moore, M. T. Todd, K. A. Norman, and P. B. Sederberg. The successor representation and temporal context. *Neural Computation*, 24(6):1553–1568, 2012.

[13] S. Goel and M. Huber. Subgoal discovery for hierarchical reinforcement learning using learned policies. In *FLAIRS conference*, pages 346–350, 2003.

[14] K. Greff, R. K. Srivastava, and J. Schmidhuber. Binding via reconstruction clustering. *arXiv preprint arXiv:1511.06418*, 2015.

[15] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[16] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational mdps. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1003–1010. Morgan Kaufmann Publishers Inc., 2003.

[17] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, pages 399–468, 2003.

[18] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

[19] N. Hernandez-Gardiol and S. Mahadevan. Hierarchical memory-based reinforcement learning. *Advances in Neural Information Processing Systems*, pages 1047–1053, 2001.

[20] J. Huang and K. Murphy. Efficient inference in occlusion-aware generative models of images. *arXiv preprint arXiv:1511.06362*, 2015.

[21] J. Koutník, J. Schmidhuber, and F. Gomez. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 541–548. ACM, 2014.

[22] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2530–2538, 2015.

[23] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*, 2016.

[24] M. C. Machado and M. Bowling. Learning purposeful behaviour in the absence of rewards. *arXiv preprint arXiv:1605.07700*, 2016.

[25] S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 71. ACM, 2004.

[26] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. *Computer Science Department Faculty Publication Series*, page 8, 2001.

[27] I. Menache, S. Mannor, and N. Shimkin. Q-cutdynamic discovery of sub-goals in reinforcement learning. In *Machine Learning: ECML 2002*, pages 295–306. Springer, 2002.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[29] S. Mohamed and D. J. Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2116–2124, 2015.

[30] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[31] K. Narasimhan, T. Kulkarni, and R. Barzilay. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*, 2015.

[32] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.

[33] D. J. Rezende, S. Mohamed, I. Danihelka, K. Gregor, and D. Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

[34] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1312–1320, 2015.

[35] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[36] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *Autonomous Mental Development, IEEE Transactions on*, 2(3):230–247, 2010.

[37] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[38] Ö. Şimşek, A. Wolfe, and A. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the International conference on Machine learning*, pages 816–823, 2005.

[39] S. Singh, R. L. Lewis, and A. G. Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pages 2601–2606, 2009.

[40] S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *Autonomous Mental Development, IEEE Transactions on*, 2(2):70–82, 2010.

[41] S. P. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2004.

[42] J. Sorg and S. Singh. Linear options. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 31–38, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[43] E. S. Spelke and K. D. Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

[44] K. L. Stachenfeld, M. Botvinick, and S. J. Gershman. Design principles of the hippocampal cognitive map. In *Advances in neural information processing systems*, pages 2528–2536, 2014.

[45] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[46] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

[47] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[48] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

[49] C. Szepesvari, R. S. Sutton, J. Modayil, S. Bhatnagar, et al. Universal option models. In *Advances in Neural Information Processing Systems*, pages 990–998, 2014.

[50] W. F. Whitney, M. Chang, T. Kulkarni, and J. B. Tenenbaum. Understanding visual concepts with continuation learning. *arXiv preprint arXiv:1602.06822*, 2016.