# Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation

**Authored by:**

Ardavan Saeedi
Tejas D. Kulkarni
Josh Tenenbaum
Karthik Narasimhan

**Abstract**

Learning goal-directed behavior in environments with sparse feedback is a major challenge for reinforcement learning algorithms. One of the key difficulties is insufficient exploration, resulting in an agent being unable to learn robust policies. Intrinsically motivated agents can explore new behavior for their own sake rather than to directly solve external goals. 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 action-value functions, operating at different temporal scales, with goal-driven intrinsically motivated deep reinforcement learning. A top-level q-value function learns a policy over intrinsic goals, while 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 and delayed feedback: (1) a complex discrete stochastic decision process with stochastic transitions, and (2) the classic ATARI game – 'Montezuma's Revenge'.

## 1 Paper Body

ion and Intrinsic Motivation Tejas D. Kulkarni? DeepMind, London tejasdkulkarni@gmail.com

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. One of the

key difficulties is insufficient exploration, resulting in an agent being unable to learn robust policies. Intrinsically motivated agents can explore new behavior for their own sake rather than to directly solve external goals. Such intrinsic behaviors could eventually help the agent solve tasks posed by the environment. We present hierarchicalDQN (h-DQN), a framework to integrate hierarchical action-value functions, operating at different temporal scales, with goal-driven intrinsically motivated deep reinforcement learning. A top-level q-value function learns a policy over intrinsic goals, while 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 and delayed feedback: (1) a complex discrete stochastic decision process with stochastic transitions, 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 [14, 16, 23] 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 [31, 19] offer significant improvements over -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 action-value functions (h-DQN), where the top-level module learns a policy over options (subgoals) and the bottom-level module learns policies to accomplish the objective of each option. Exploration in the space of goals enables efficient exploration in problems with sparse and delayed rewards. Additionally, our experiments indicate that goals expressed in the space of entities and relations can help constraint the exploration space for data efficient deep reinforcement learning in complex environments. ?

Equal Contribution. Work done while Tejas Kulkarni was affiliated with MIT.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Reinforcement learning (RL) formalizes control problems as finding a policy ? that maximizes expected future rewards [32]. 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 ? G [33, 21]. 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 [34, 35]. In high-dimensional problems, these value functions can be approximated by neural networks as V (s, g; ?). 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) a top level module (meta-controller) takes in the state and picks a new goal, and (b) a lower-level module (controller) uses both the state and the chosen goal to select actions either until the goal is reached or the episode terminates. 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 delayed rewards: (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

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 hierarchical reinforcement learning [2], Sutton et al.[34] proposed the options framework, which involves abstractions over the space of actions. At each step, the agent chooses either a onestep ?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 ?. 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 [35] or by composing existing options [28]. Value functions have also been generalized to consider goals along with states [21]. Our work is inspired by these papers and builds upon them. Other related work for hierarchical formulations include the MAXQ framework [6], which decomposed the value function of an MDP into combinations of value functions of smaller constituent MDPs, as did Guestrin et al.[12] in their factored MDP formulation. Hernandez and Mahadevan [13] combine hierarchies with short-term memory to handle partial observations. In the skill learning literature, Baranes et al.[1] have proposed a goal-driven active learning approach for learning skills in continuous sensorimotor spaces. In this 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 [21]. This has two potential advantages: (1) there is shared learning between different options, and (2) the model is scalable to a

large number of options. Intrinsic Motivation The nature and origin of ?good? intrinsic reward functions is an open question in reinforcement learning. Singh et al. [27] explored agents with intrinsic reward structures in order to learn generic options that can apply to a wide variety of tasks. In another paper, Singh et al. [26] 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 [10] discuss a framework for sub-goal discovery using the structural aspects of a learned policy model. S?ims?ek et al. [24] provide a graph partitioning approach to subgoal identification. 2

Schmidhuber [22] 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 [17] have recently proposed a notion of intrinsically motivated learning within the framework of mutual information maximization. Frank et al. [9] demonstrate the effectiveness of artificial curiosity using information gain maximization in a humanoid robot. Oudeyer et al. [20] categorize intrinsic motivation approaches into knowledge based methods, competence or goal based methods and morphological methods. Our work relates to competence based intrinsic motivation but other complementary methods can be combined in future work. Object-based Reinforcement Learning Object-based representations [7, 4] that can exploit the underlying structure of a problem have been proposed to alleviate the curse of dimensionality in RL. Diuk et al. [7] 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. [11], 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. 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 [16, 15] and Go [23], but still perform poorly on environments with sparse, delayed reward signals. 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 [29]. During curiositydriven 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 subgoals in the space of their core knowledge. Knowledge of space can also be utilized to learn a hierarchical decomposition of spatial environments. This has been explored in Neuroscience with the successor representation, which represents value functions in terms of the expected future

state occupancy. Decomposition of the successor representation have shown to yield reasonable subgoals for spatial navigation problems [5, 30].

3

Model

Consider a Markov decision process (MDP) represented by states s ? S, actions a ? A, and transition function T : (s, a) ? s0 . 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 s0 . We define the extrinsic reward function as F : (s) ? 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 -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 intrinsic goals g ? G. The agent focuses on setting and achieving sequences of goals via learning policies ?g in order to maximize cumulative extrinsic reward. In order to learn each ?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 st and chooses a goal gt ? G, where G denotes the set of all possible current goals. The controller then selects an action at using st and gt . The goal gt 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 provides an appropriate reward rt (g) to the controller. In this work, we make a minimal 3

action

External Environment

observations

extrinsic reward

gt+N

gt

Meta Controller

Q2 (st+N , gt+N ; ?2 )

Q2 (st , g; ?2 )

goal .....

Critic action

Meta Controller

st

intrinsic reward

at

at+1

Meta Controller

.....

st+N Q1 (st , a; ?1 , gt )

Q1 (st+1 , a; ?1 , gt ) Q1 (st+N , a; ?1 , gt )

Controller
Controller
at+N
Controller
st
....
st+1
Controller
st+N
gt
agent

Figure 1: (Overview) The agent receives sensory observations and produces actions. Separate DQNs are used inside the meta-controller and controller. The meta-controller looks at the raw states and produces a policy over goals by estimating the action-value function Q2 (st , gt ; ?2 ) (to maximize expected future extrinsic reward). The controller takes in states and the current goal, and produces a policy over actions by estimating the action-value function Q1 (st , at ; ?1 , gt ) to solve the predicted goal (by maximizing expected future intrinsic reward). The internal critic provides a positive reward to the controller if and only if the goal is reached. 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. assumption of a binary internal reward i.e. 1 if the goal is reached and 0 otherwise. The objective P? 0 function for the controller is to maximize cumulative intrinsic reward: Rt (g) = t0 =t ? t ?t rt0 (g). Similarly, the objective of the meta-controller is to optimize the cumulative extrinsic reward Ft = P? t0 ?t ft0 , where ft are reward signals received from the environment. Note that the time t0 =t ? scales for Ft and Rt are different ? each ft is the accumulated external reward over the time period between successive goal selections. The discounting in Ft , therefore, is over sequences of goals and not lower level actions. This setup is similar to optimizing over the space of optimal reward functions to maximize fitness [25]. In our case, the reward functions are dynamic and temporally dependent on the sequential history of goals. Figure 1 illustrates the agent?s use of the hierarchy over subsequent time steps. Deep Reinforcement Learning with Temporal Abstractions We use the Deep Q-Learning framework [16] to learn policies for both the controller and the meta-controller. Specifically, the controller estimates the following Q-value function: ? X

0 Q?1 (s, a; g) = max E ? t ?t rt0 — st = s, at = a, gt = g, ?ag ?ag

t0 =t

(1)

= max E rt + ? maxat+1 Q?1 (st+1 , at+1 ; g) — st = s, at = a, gt = g, ?ag ?ag

where g is the agent?s goal in state s and ?ag is the action policy. Similarly, for the meta-controller, we have: Q?2 (s, g)

X t+N

= max?g E ft0 + ? maxg0 Q?2 (st+N , g 0 ) — st = s, gt = g, ?g

(2)

t0 =t

where N denotes the number of time steps until the controller halts given the current goal, g 0 is the agent?s goal in state st+N , and ?g is the policy over goals. It is important to note that the transitions (st , gt , ft , st+N ) generated by Q2 run at a slower time-scale than the transitions (st , at , gt , rt , st+1 ) generated by Q1 . We can represent Q? (s, g) ? Q(s, g; ?) using a non-linear function approximator with parameters ?. Each Q ? {Q1 , Q2 } can be trained by minimizing corresponding loss functions ? L1 (?1 ) and L2 (?2 ). We store experiences (st , gt , ft , st+N ) for Q2 and (st , at , gt , rt , st+1 ) for Q1 in disjoint 4

memory spaces D1 and D2 respectively. The loss function for Q1 can then be stated as:

L1 (?1,i ) = E(s,a,g,r,s0 )?D1 (y1,i ? Q1 (s, a; ?1,i , g))2 , where i denotes the training iteration number and y1,i = r + ? maxa0 Q1 (s0 , a0 ; ?1,i?1 , g).

(3)

Following [16], the parameters ?1,i?1 from the previous iteration are held fixed when optimizing the loss function. The parameters ?1 can be optimized using the gradient: ??1,i L1 (?1,i ) = E(s,a,r,s0 ?D1 )

”

#

r + ? max Q1 (s , a ; ?1,i?1 , g) ? Q1 (s, a; ?1,i , g) ??1,i Q1 (s, a; ?1,i , g)

a0

0

0

The loss function L2 and its gradients can be derived using a similar procedure. Algorithm 1 Learning algorithm for h-DQN 1: Initialize experience replay memories {D1 , D2 } and parameters {?1 , ?2 } for the controller and

meta-controller respectively.

2: Initialize exploration probability 1,g = 1 for the controller for all goals g and 2 = 1 for the

meta-controller.

3: for i = 1, num episodes do 4: Initialize game and get start state description s 5: g ? EPS G REEDY(s, G, 2 , Q2 ) 6: while s is not terminal do 7: F ?0 8: s0 ? s 9: while not (s is terminal or goal g reached) do 10: a ? EPS G REEDY({s, g}, A, 1,g , Q1 ) 11: Execute a and obtain next state s0 and extrinsic reward f from environment 12: Obtain intrinsic reward r(s, a, s0 ) from internal critic 13: Store transition ({s, g}, a, r, {s0 , g}) in D1 14: UPDATE PARAMS (L1 (?1,i ), D1 ) 15: UPDATE PARAMS (L2 (?2,i ), D2 ) 16: F ?F +f 17: s ? s0 18: end while 19: Store transition (s0 , g, F, s0 ) in D2 20: if s is not terminal then 21: g ? EPS G REEDY(s, G, 2 , Q2 ) 22: end if 23: end while 24: Anneal 2 and 1 . 25: end for

Algorithm 2 : EPS G REEDY(x, B, , Q) 1: if random() ¡ then 2: return random element from set B 3: else 4: return argmaxm?B Q(x, m) 5: end if

Algorithm 3 : UPDATE PARAMS(L, D) 1: Randomly sample mini-batches from D 2: Perform gradient descent on loss L(?)

(cf. (3))

Learning Algorithm We learn the parameters of h-DQN using stochastic gradient descent at different time scales ? transitions from the controller are collected at every time step but a transition from the meta-controller is only collected when the controller terminates (i.e. when a goal is repicked or the episode ends). Each new goal g is drawn in an -greedy fashion (Algorithms 1 & 2) with the exploration probability 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 1,g , which depends on the current empirical success rate of reaching g. Specifically, if the success rate for goal g is ¿ 90%, we set 1,g = 0.1, else 1. All 1,g values are annealed to 0.1. The model parameters (?1 , ?2 ) are periodically updated by drawing experiences from replay memories D1 and D2 , respectively (see Algorithm 3). 5

4

Experiments

(1) Discrete stochastic decision process with delayed rewards For our first experiment, we consider a stochastic decision process where the extrinsic reward depends on the history of visited states in addition to the current state. This task demonstrates the importance of goal-driven exploration in such environments. There are 6 possible states and the agent always starts at s2 . 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 s1 and the agent receives the reward of 1 when it first visits s6 and then s1 . The reward for going to s1 without visiting s6 is 0.01. This is a modified version of the MDP in [19], with the reward structure adding complexity to the task (see Figure 2). We consider each state as a candidate goal for 0.5 0.5 0.5 0.5 0.5 exploration. This enables and encourages the agent to visit state s6 (if chosen as a goal) and r = or1/100 s1 0.5 s2 0.5 s3 0.5 s4 0.5 s5 0.5 s6 r=1 hence, learn the optimal policy. For each goal, 1.0 1.0 1.0 1.0 1.0 the agent receives a positive intrinsic reward if Figure 2: A stochastic decision process where the and only if it reaches the corresponding state. reward at the terminal state s1 depends on whether Results We compare the performance of s6 is visited (r = 1) or not (r = 1/100). Edges our approach (without deep neural networks) are annotated with transition probabilities (Red aragainst Q-Learning as a baseline (without in- row: move right, Black arrow: move left).

trinsic rewards) in terms of the average extrinsic reward gained in an episode. In our experiments, all parameters are annealed from 1 to 0.1 over 50k steps. The learning rate is set to 2.5 ? 10?4 . 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 s1 directly to obtain a reward of 0.01. In contrast, our approach with hierarchical Q-estimators learns to choose goals s4 , s5 or s6 , which statistically lead the agent to visit s6 before going back to s1 . Our agent obtains a significantly higher average reward of 0.13. 5/18/2016 Reward.html 5/18/2016

Reward 0.18 0.16 0.14 0.12 0.1 0.08 0.06 0.04 0.02 0
0
State 3, State 4, State 5, State 6 — ?lled scatter chart made by Ardavans — plotly
# of visits per episode
1.2
State 3 State 4 State 5 State 6
1 0.8 0.6 0.4
Baseline Our Approach 50
100 Steps
150
0.2 0
200
2
Export?to?plot.ly??
4
6
Episodes (*1000)
8
10
12 Edit chart ?

Figure 3: (left) Average reward (over 10 runs) of our approach compared to Q-learning. (right) #visits of our approach to states s3 -s6 (over 1000 episodes).
Initial state: s2 , Terminal state: s1 . ?le:///Users/tejas/Documents/deepRelationalRL/dqn/Reward.html
1/1

Figure 3 illustrates that the number of visits to states s3 , s4 , s5 , s6 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 s6 more often. https://plot.ly/ardavans/4.embed
1/1

(2) ATARI game with delayed rewards We now consider ?Montezuma?s Revenge?, an ATARI game with sparse, delayed rewards. The game 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). Basic DQN [16] achieves a score of 0 while even the best performing system, Gorila DQN [18], manages only 4.16 on average. Asynchronous actor critic methods achieve a non-zero score but require 100s of millions of training frames [15]. 6
Architecture
Total extrinsic reward
5/18/2016

Reward.html
Q1(s, a; g) Linear
400 350 300 250 200 150 100 50 0
ReLU:Linear (h=512) ReLU:Conv (filter:3, ftr-maps:64, strides:1) ReLU:Conv (filter:4, ftr-maps:64, strides:2) ReLU:Conv (filter:8, ftr-maps:32, strides:4)
Our Approach DQN
0
image (s) + goal (g)
0.5M
1M
5/18/2016 5/18/2016
Steps
1.5M
2M Export?to?plot.ly??
Bar graph.html
subgoal_6.html
Success ratio for reaching the goal ?key?
Success % of different goals over time
1
0.25
0.8
0.2
0.6
0.15
0.4
0.1
0.2
1/1
0.05
0
0
top-left door top-right door middle-ladder bottom-left-ladder bottom-right-ladder key
?le:///Users/tejas/Documents/deepRelationalRL/dqn/Reward.html
0.5M
1M
Steps
1.5M
0
2M
0.5M
Export?to?plot.ly??
1M
Steps
1.5M
2M Export?to?plot.ly??

Figure 4: (top-left) Architecture: DQN architecture for the controller (Q1 ). A similar architecture produces Q2 for the meta-controller (without goal as input). (top-right) The joint training learns to consistently get high rewards. (bottom-left) Goal success ratio: The agent learns to choose the key more often as training proceeds and is successful at achieving it. (bottom-right) 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.

?le:///Users/tejas/Documents/deepRelationalRL/dqn/subgoal_6.html 1/1 ?le:///Users/tejas/Documents/deepRelationalRL/dqn/Bar%20graph.html 1/1

Setup The agent needs intrinsic motivation to explore meaningful parts of the scene before learning about the advantage of obtaining the key. Inspired by developmental psychology literature [29] and object-oriented MDPs [7], 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 [8]. In this work, we built a custom pipeline to provide plausible object candidates. Note that the agent is still required to learn which of these candidates are worth pursuing as goals. The controller and meta-controller are convolutional neural networks (Figure 4) that learn representations from raw pixel data. We use the Arcade Learning Environment [3] to perform experiments. The internal critic is defined in the space of hentity1 , relation, entity2 i, where relation is a function over configurations of the entities. In our experiments, the agent learns to choose entity2 . For instance, the agent is deemed to have completed a goal (and only then receives a reward) if the agent entity reaches another entity such as the door. The critic computes binary rewards using the relative positions of the agent and the goal (1 if the goal was reached). 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 ?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 4, 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 D1 and D2 were set to be equal to 106 and 5 ? 104 respectively. We set the learning rate to be 2.5 ? 10?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 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. 7

Meta Controller termination (death)
goal reached
Controller
1
2
3
4
5
6
Meta Controller goal reached
Controller
7
8
9
10
11
12

Figure 5: Sample game play on Montezuma?s Revenge: The four quadrants are arranged in a temporal order (top-left, top-right, bottom-left and bottom-right). First, 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. Results Figure 4 shows reward progress from the joint training phase ? 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. The agent learns to choose the key more often as training proceeds and is also successful at reaching it. 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 4 also shows the evolution of the success rate of goals that are picked. At the end of training, we can see that the ?key?, ?bottomleft-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 the goal parameterization we considered, a flexible shortterm memory, or the ability to intermittently terminate ongoing options. We also show some screenshots from a test run with our agent (with epsilon set to 0.1) in Figure 5, as well as a sample animation of the run.2

# 2 References

[1] A. Baranes and P.-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. Robotics and Autonomous Systems, 61(1):49?73, 2013. [2] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. Discrete Event Dynamic Systems, 13(4):341?379, 2003. [3] 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. 2

Sample trajectory of a run on ?Montezuma?s Revenge? ? https://goo.gl/3Z64Ji 8

[4] L. C. Cobo, C. L. Isbell, and A. L. Thomaz. Object focused q-learning for autonomous agents. In Proceedings of AAMAS, pages 1061?1068, 2013. [5] P. Dayan. Improving generalization for temporal difference learning: The successor representation. Neural Computation, 5(4):613?624, 1993. [6] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. J. Artif. Intell. Res.(JAIR), 13:227?303, 2000. [7] C. Diuk, A. Cohen, and M. L. Littman. An object-oriented representation for efficient reinforcement learning. In Proceedings of the International Conference on Machine learning, pages 240?247, 2008. [8] 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. [9] M. Frank, J. Leitner, M. Stollenga, A. F?orster, 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. [10] S. Goel and M. Huber. Subgoal discovery for hierarchical reinforcement learning using learned policies. In FLAIRS conference, pages 346?350, 2003. [11] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational mdps. In Proceedings of International Joint conference on Artificial Intelligence, pages 1003?1010, 2003. [12] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored mdps. Journal of Artificial Intelligence Research, pages 399?468, 2003. [13] N. Hernandez-Gardiol and S. Mahadevan. Hierarchical memory-based reinforcement learning. In Advances in Neural Information Processing Systems, pages 1047?1053, 2001. [14] J. Koutn??k, 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. [15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. arXiv preprint arXiv:1602.01783, 2016. [16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529?533, 2015. [17] 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. [18] A. Nair, P. Srini-

vasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, et al. Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296, 2015. [19] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. arXiv preprint arXiv:1602.04621, 2016. [20] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. Frontiers in neurorobotics, 1:6, 2009. [21] 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. [22] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990?2010). Autonomous Mental Development, IEEE Transactions on, 2(3):230?247, 2010. [23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484?489, 2016. ? S?ims?ek, A. Wolfe, and A. Barto. Identifying useful subgoals in reinforcement learning by local graph [24] O. partitioning. In Proceedings of the International conference on Machine learning, pages 816?823, 2005. [25] 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. [26] 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. [27] S. P. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In Advances in neural information processing systems, pages 1281?1288, 2004. [28] J. Sorg and S. Singh. Linear options. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pages 31?38, Richland, SC, 2010. [29] E. S. Spelke and K. D. Kinzler. Core knowledge. Developmental science, 10(1):89?96, 2007. [30] 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. [31] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. arXiv preprint arXiv:1507.00814, 2015. [32] R. S. Sutton and A. G. Barto. Introduction to reinforcement learning. MIT Press Cambridge, 1998. [33] 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, pages 761?768, 2011. [34] 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. [35] 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.

9