Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation

PRESENTER: NIMA AKBARZADEH – BILKENT UNIVERSITY – SPRING 2017 – DEEP LEARNING COURSE – INSTRUCTOR: GOKBERK CINBIS

Outline

- Introduction
- Temporal Abstraction
- Intrinsic Motivation
- Deep Reinforcement Learning
- Model Structure
- Experiments
- Results

Introduction

- Nonlinear function approximation + reinforcement learning → Abstraction over state space ☺
- Exploration with sparse feedback → A challenge ☹
- Proposed method includes hierarchical approach to the problem (h-DQN) → Efficient exploration with sparse and delayed feed-back.

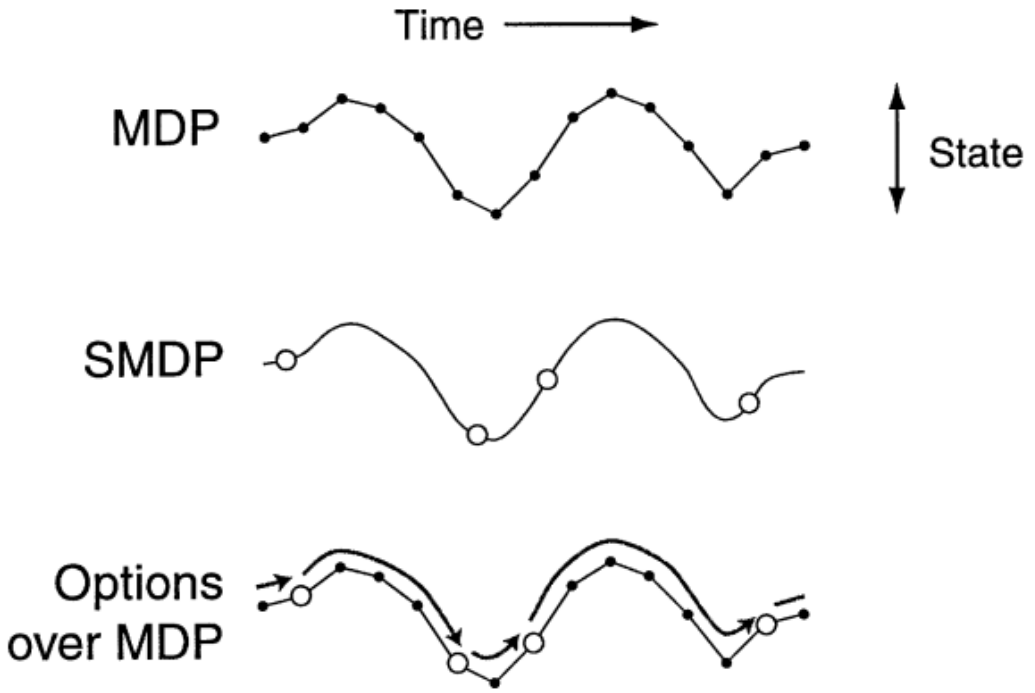
Introduction

- Value functions may take goals as inputs as well: $V(s, g)$.
- ANN with parameter set θ can be used for approximation: $V(s, g; \theta)$.
- Two control level is proposed with two sets of rewards.
- The top level controller (meta-controller) defines a goal.
- The low level one (controller) select actions in order to achieve the goal.

Temproal Abstaction (TA)

- Dealing with stochastic environments.
- Useful for learning and planning.
- In complex environments, it makes the solution simple and efficient.
- Semi-Markov Decision Process (SMDP) is one of the keys to treat TA as an extension over reinforcement learning framework.

R.S. Sutton et al. / Artificial Intelligence 112 (1999) 181–211



Intrinsic Motivation

- Some animals, and this is most prominent in humans, also have more general motivations that push them to **explore, manipulate or probe their environment**, fostering curiosity and engagement in playful and new activities. This kind of motivation, which is called intrinsic motivation by psychologists (Ryan and Deci, 2000), is paramount for sensorimotor and cognitive development throughout lifespan.
- Measure of dissonances between the situations experienced and the knowledge and expectations that an agent has about these situations: representations that estimate the **distributions of probabilities of observing certain events**.
- A second major computational approach to intrinsic motivation is based on measures of competence that an agent has for achieving self-determined results or goals: Thus, a challenge is here a **self-determined goal**.

Deep Reinforcement Learning

- Used for function Approximation.
- Good at handling high-dimensional input data.
- DQN has been used in Atari and Go.
- Poor performance on environments with sparse and delayed reward.

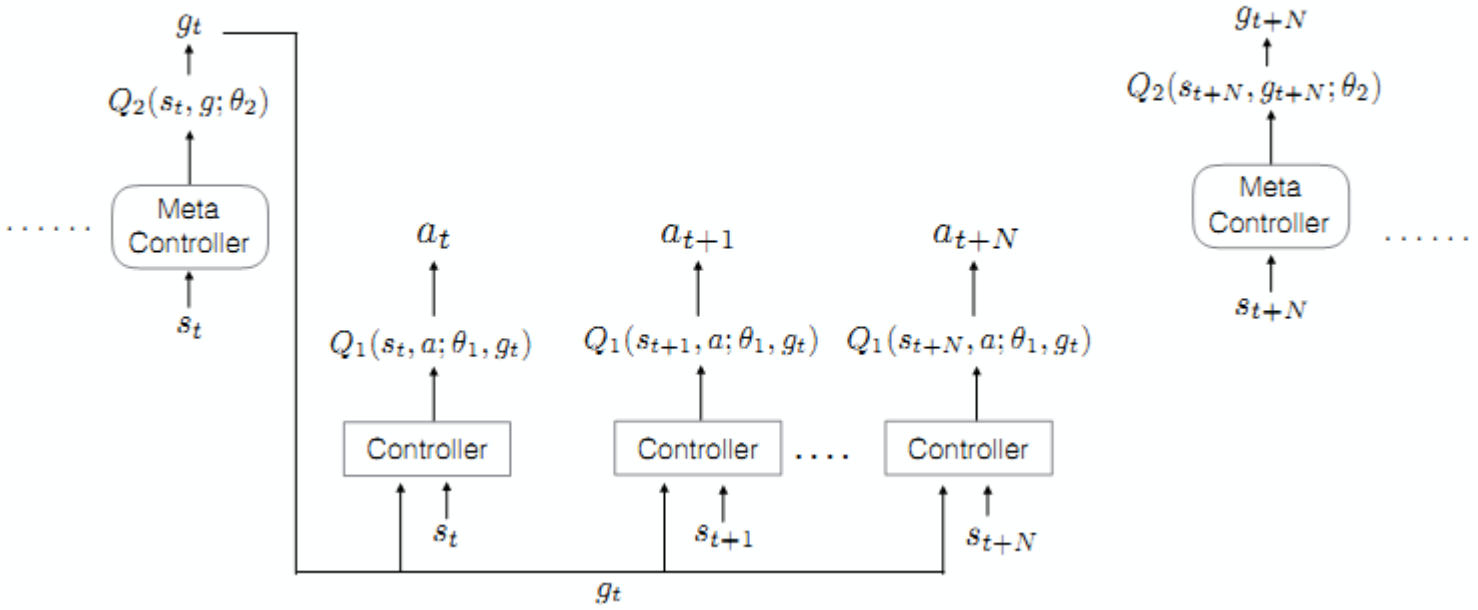
Model – General Description

- State s in state space S.
- Action a in action set A.
- Transition function T which maps (s, a) to s’.
- Extrinsic Reward function F which maps s to real numbers.

Model - Agent

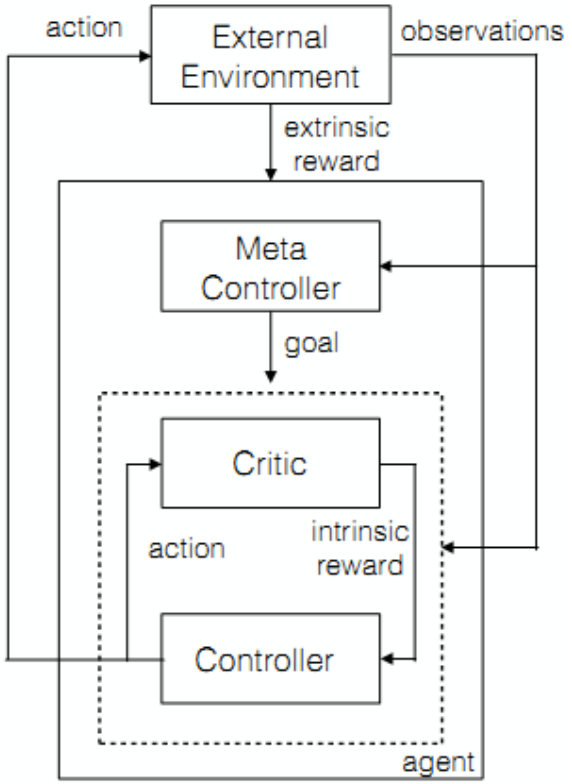
- Exploration vs. Exploitation.
- Exploration helps to learn good policies.
- Epsilon-greedy: useful for local exploration.
- Intrinsic goals (g in G) is utilized to explore.
- Policy \pi_g is applied over sequence of goals to maximize extrinsic reward.
- To learn \pi_g, an intrinsic reward is used.

Model – Temporal Abstraction



Model – Temporal Abstraction

- External environment produces the observation and the extrinsic reward.
- Meta-controller defines the goal based on the given state.
- Controller defines the action based on the current state and the goal.
- Critic provides the intrinsic reward based on the action and observation of the external environment. (Reward is positive only if goal is reached!)



Model – Objectives

- Controller tries to maximize:

$$R_t(g) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}(g).$$

- Meta-controller tries to maximize:

$$F_{t'} = \sum_{t'=t}^{\infty} \gamma^{t'-t} f_{t'}.$$

- Note that the time scale of the cumulative reward functions are different.

Model - Deep RL + TA

Q – value functions for each controller can be expressed as

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

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

Model - DQN

- Each Q-value function can be approximated by a non-linear function with parameters θ.
- Each Q-function can be trained by stochastic gradient descent using loss functions.

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

$$y_{1,i} = r + \gamma \max_{a'} Q_1(s',a';\theta_{1,i-1},g).$$

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

Model - Algorithm

- Experiences for the controllers are stored in memory D1 and D2.
- For Q1, (s_t, a_t, g_t, s_{t+1}) is stored and for Q2 (s_t, g_t, f_t, s_{t+N}) is stored.

Algorithm 2 : 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 : UPDATEPARAMS($\mathcal{L}, \mathcal{D}$)

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

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
    $g \leftarrow$ 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**
            $a \leftarrow$ 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 $\mathcal{D}_1$
            UPDATEPARAMS$(\mathcal{L}_1(\theta_{1,i}), \mathcal{D}_1)$
            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**
            $g \leftarrow$ EPSGREEDY$(s, \mathcal{G}, \epsilon_2, Q_2)$
        end if
    end while
    Anneal ϵ_2 and ϵ_1.
end for

Experiments

- Discrete stochastic decision process with delayed rewards:
  - The agent starts at state S2.
  - Each state is considered as a candidate for goal state.
  - \eps anneal from 1 to 0.1 after 50k.
  - learning rate is set to 2.5×10^{-4}.

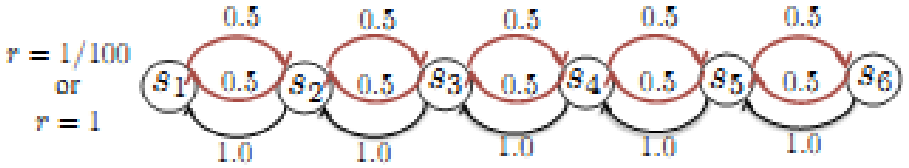


Figure 2: A stochastic decision process where the reward at the terminal state s_1 depends on whether s_6 is visited ($r = 1$) or not ($r = 1/100$). Edges are annotated with transition probabilities (Red arrow: move right, Black arrow: move left).

Result

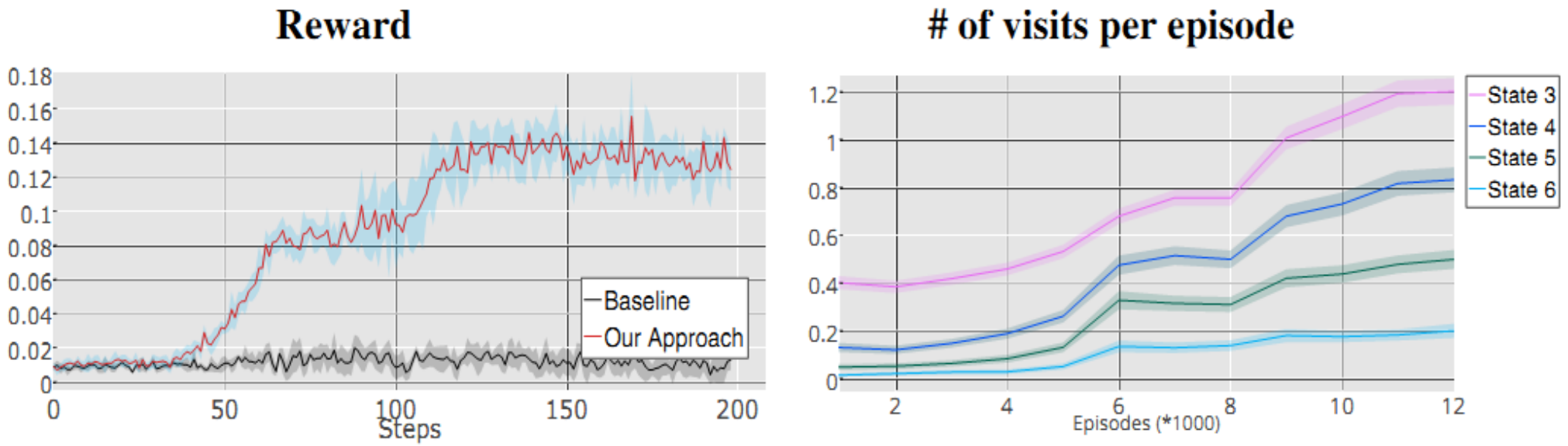


Figure 3: **(left)** Average reward (over 10 runs) of our approach compared to Q-learning. **(right)** #visits of our approach to states s_3-s_6 (over 1000 episodes). Initial state: s_2, Terminal state: s_1.

Experiments

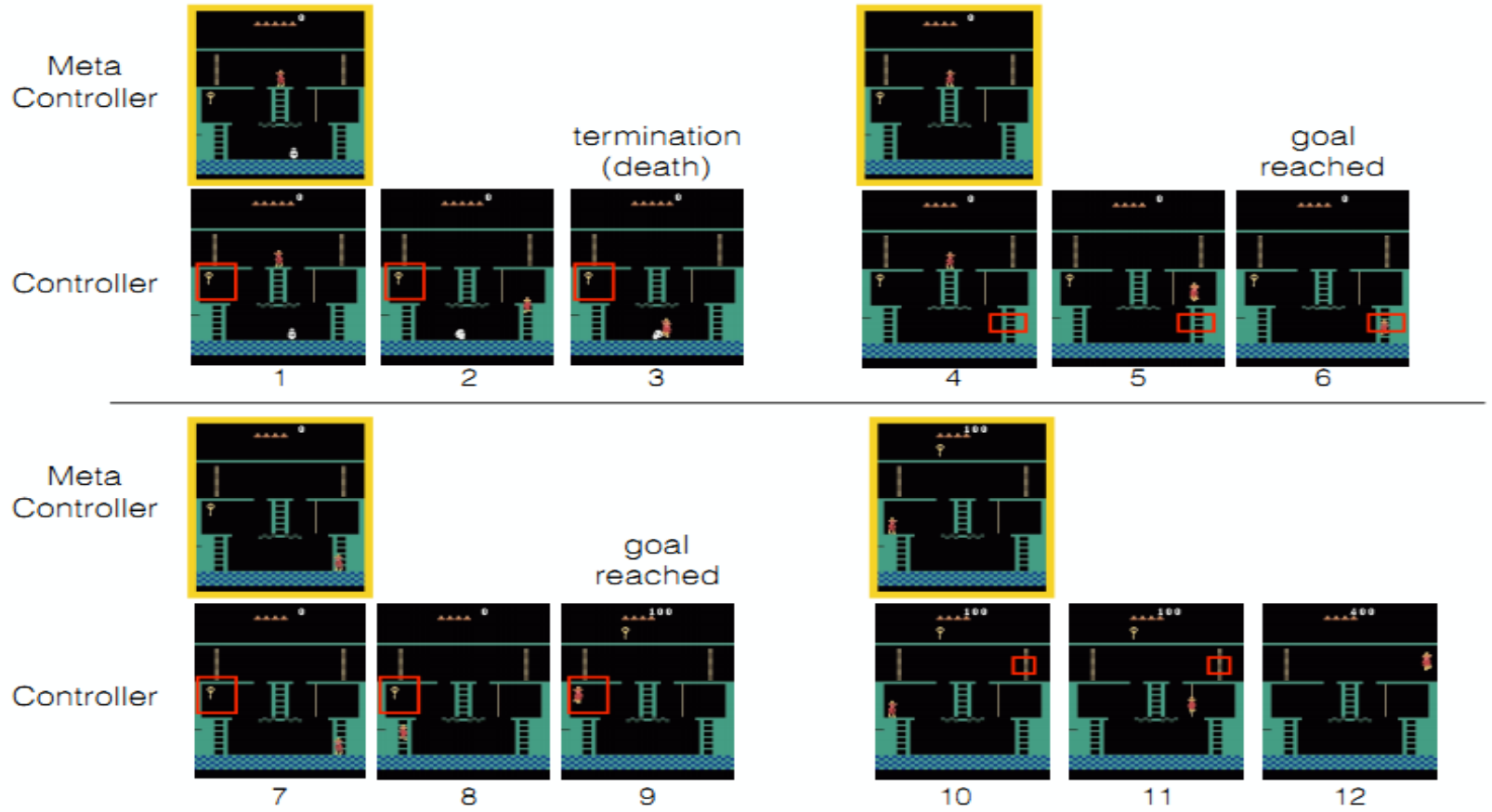
- Atari Games with delayed rewards (Montezuma's Revenge Game):

- The player receives reward by picking up a key (+100) and opening the door (+300).
- Long sequence of actions has to be taken.

Setup

- Exploration should be done.
- Objects are set as possible goals.
- Unsupervised detection of the objects.
- Agent should learn which of the objects worth to be set as a goal.
- CNN are used for the controllers.
- Internal critic is defined in the space of (entity1 – relation – entity2).
- The critic computes binary reward.

Montezuma's Revenge Game

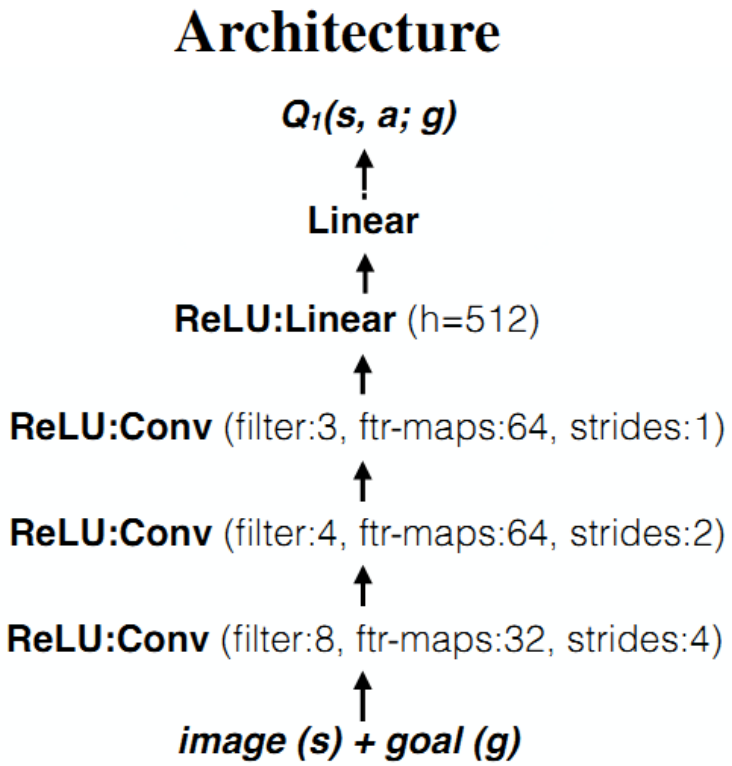


Model Architecture

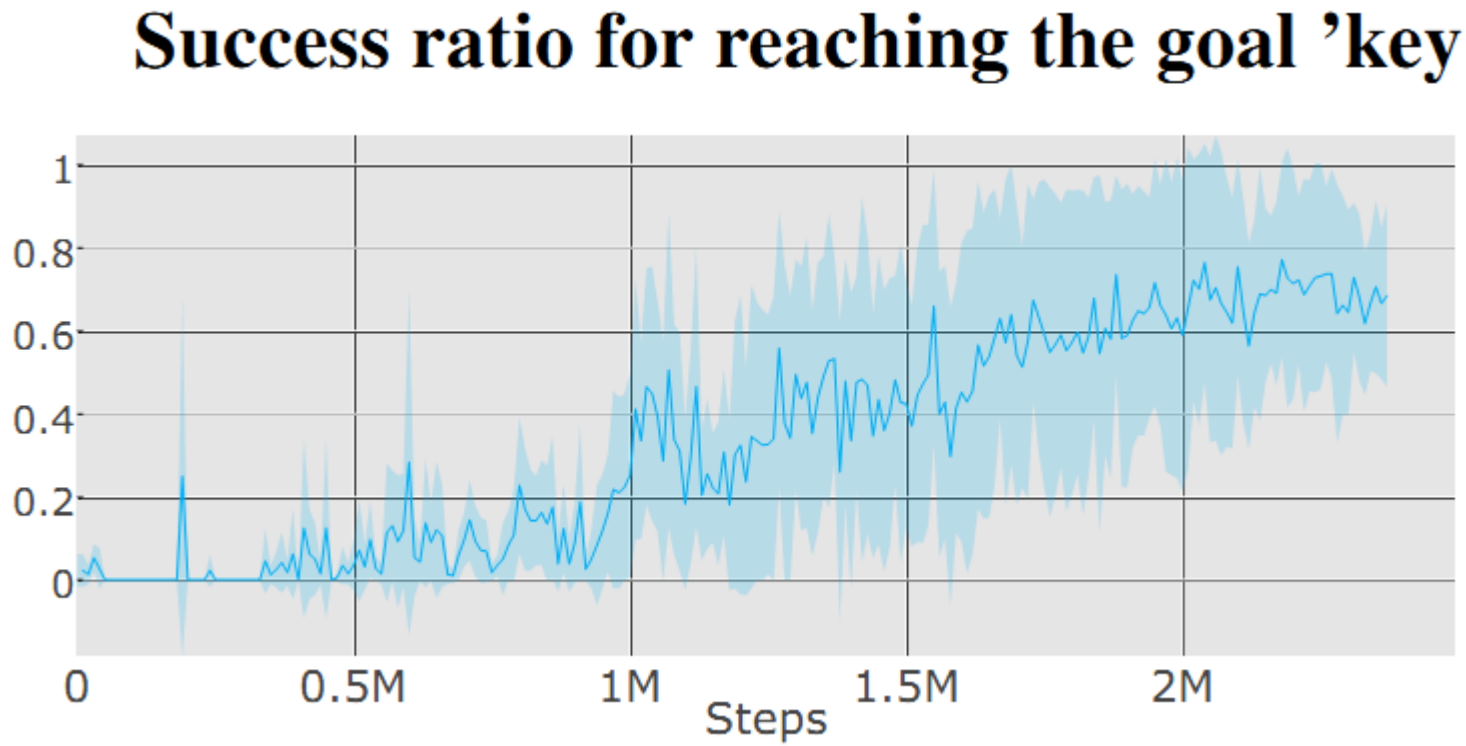
- Input images: 84×84.
- learning rate: 2.5×10^{-4}.
- Discount factor: 0.99.
- Two phase training procedure:

(1) \eps 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) Jointly train the controller and meta-controller.



Results


Success ratio for reaching the goal 'key'
1
0.8
0.6
0.4
0.2
0
0
0.5M
1M
1.5M
2M
Steps

Results

Total Extrinsic reward: