

RL: Introduction to Hierarchical Reinforcement Learning


Andrea Mattei

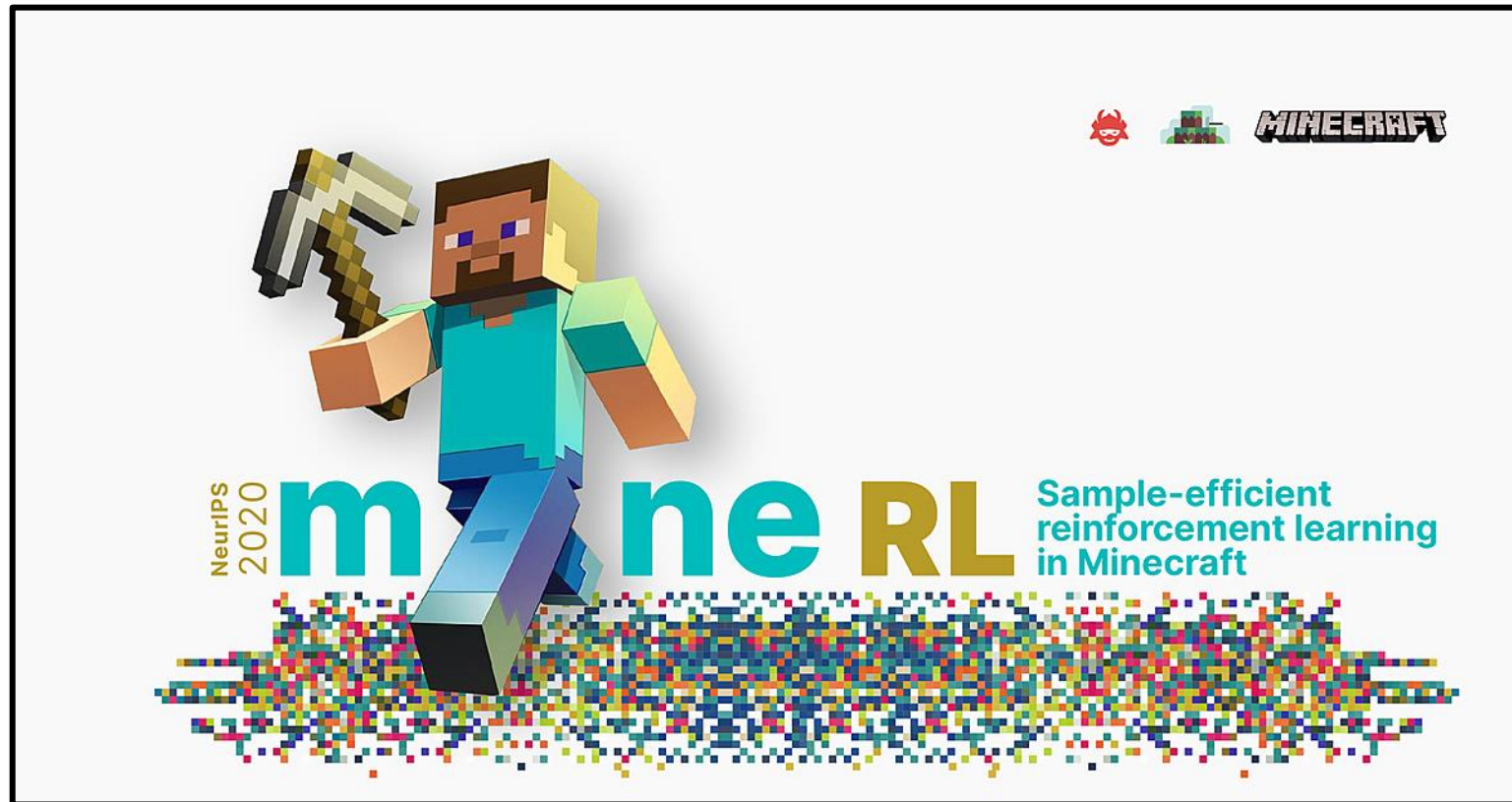


You are surfing the web, when you stumble on this website

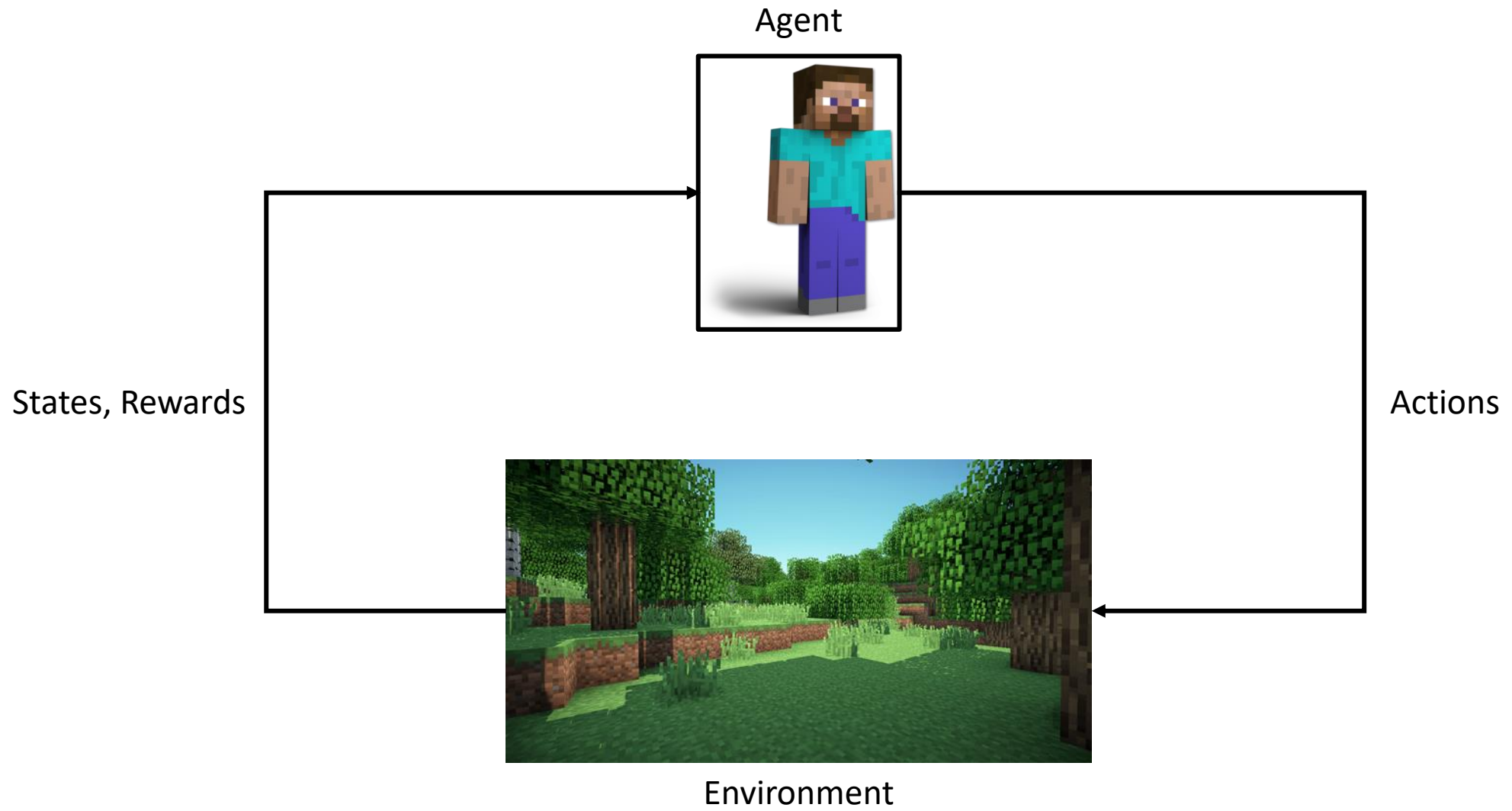


MineRL

Finds diamonds  as fast as possible



First Try

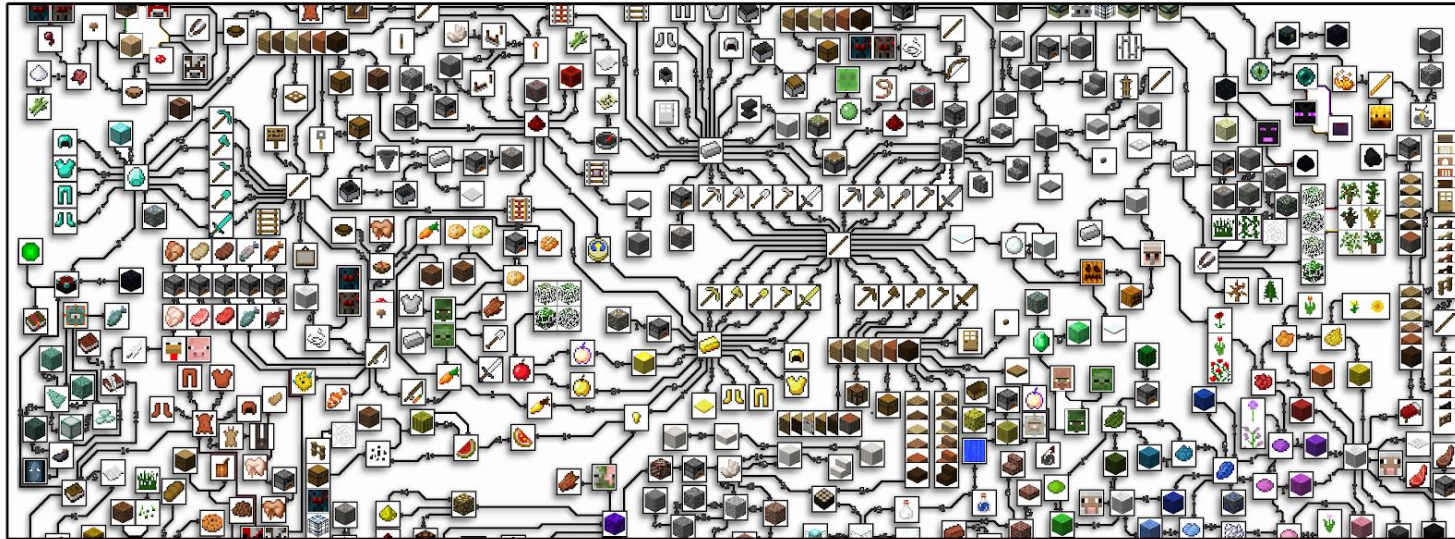


First Failure

In Minecraft tools have **many iterations**



We need generalization and transfer learning



First Failure

States are **big** and **complex**!

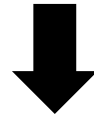


We need to deal with big states efficiently



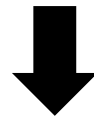
First Failure

Limited training **samples!**



We need sample efficiency

The environment gives **sparse rewards!**



We need to deal with sparse rewards



What we need

- We need generalization and transfer learning
- We need to deal with big states efficiently
 - We need sample efficiency
- We need to deal with sparse rewards



What we need

- We need generalization and transfer learning
- We need to deal with big states efficiently
 - We need sample efficiency
- We need to deal with sparse rewards

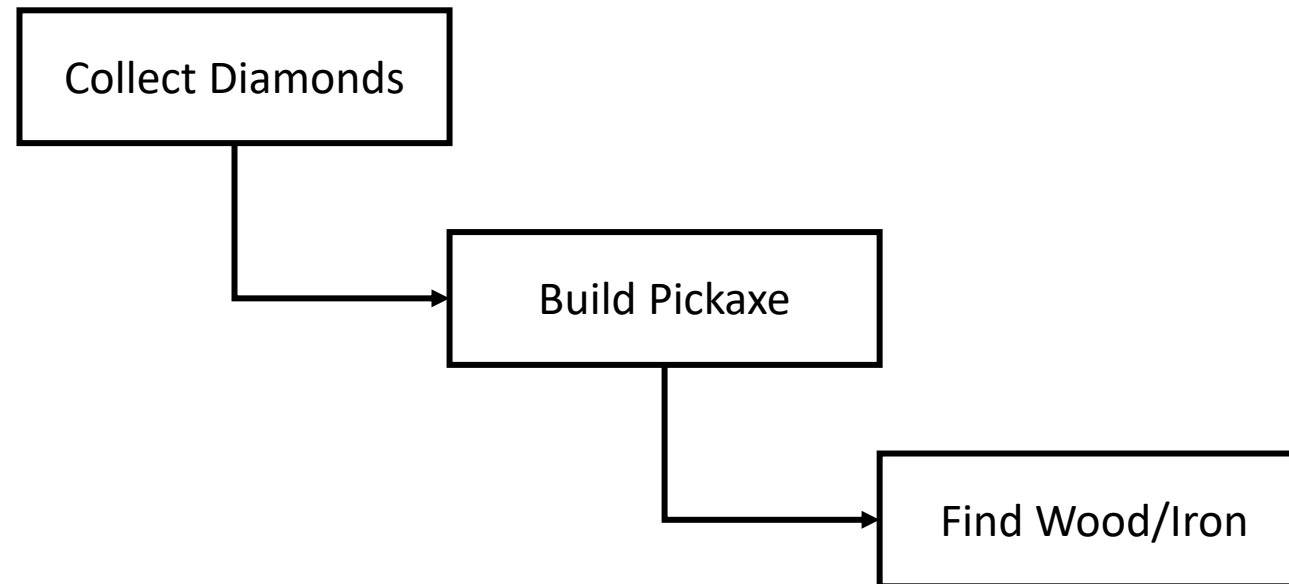


Hierarchical Reinforcement Learning (HRL) can help us!



Hierarchical Reinforcement Learning (HRL)

In Hierarchical Reinforcement Learning the agent divides the task into sub-tasks

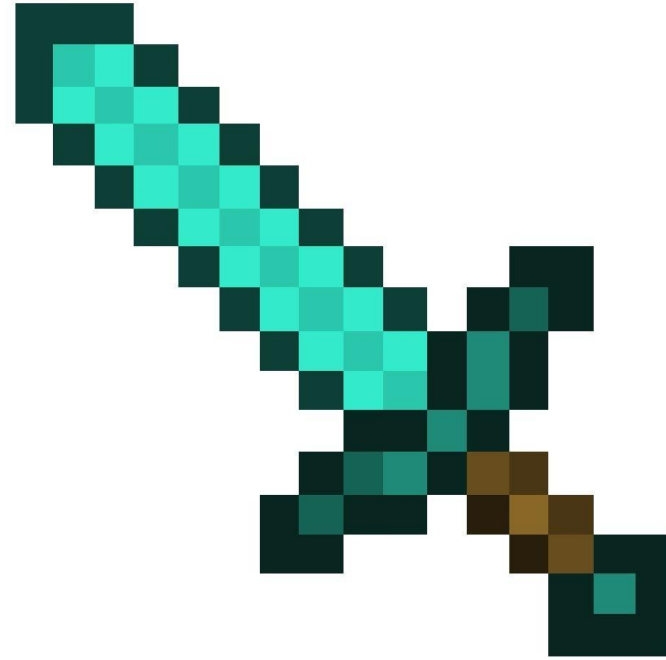
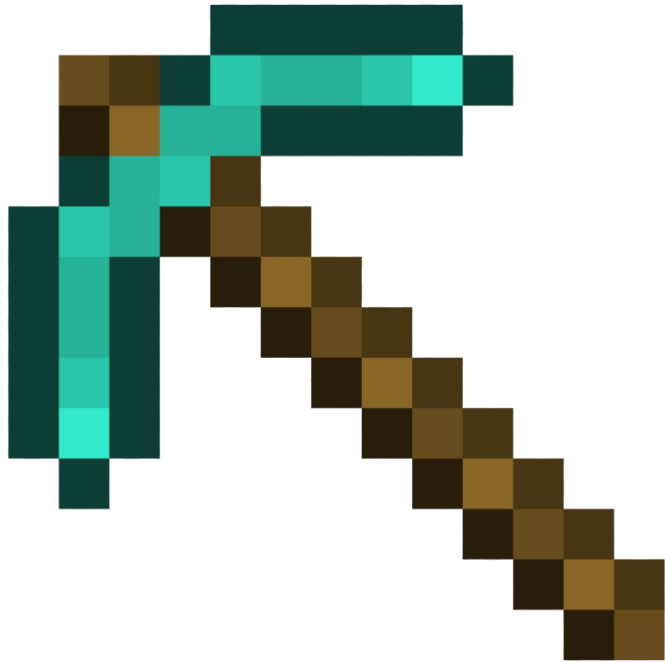


How HRL Help Us

- We need generalization and transfer learning
HRL: sub-tasks can be reused
- We need to deal with big states efficiently
HRL: sub-tasks can operate on a subset of the state space
- We need sample efficiency
HRL: sub-tasks are uniquely trained
- We need to deal with sparse rewards
HRL: sub-tasks can model sparse reward goals



Let's Pick Our Tools



Markov Decision Process (MDP)

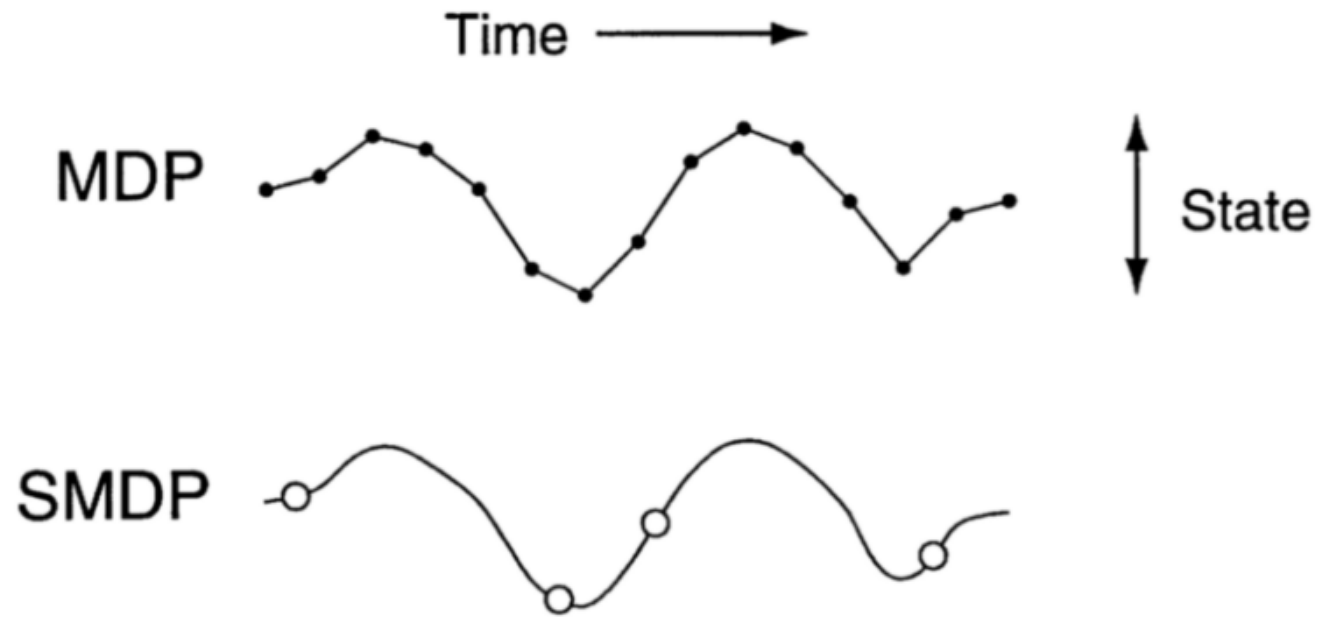
A Markov Decision Process is defined as a tuple $\langle S, A, P, R, \gamma \rangle$ where:

- S is a set of states
- A is a set of actions
- $P_{ss'}^a$ is a state transition function defined as $P_{ss'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$
- R_s^a is a reward function defined as $R_s^a = E[R_{t+1} | S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0,1]$



Semi-Markov Decision Process (SMDP)

SMDPs allow modelling of **continuous-time discrete-event** systems.



Options

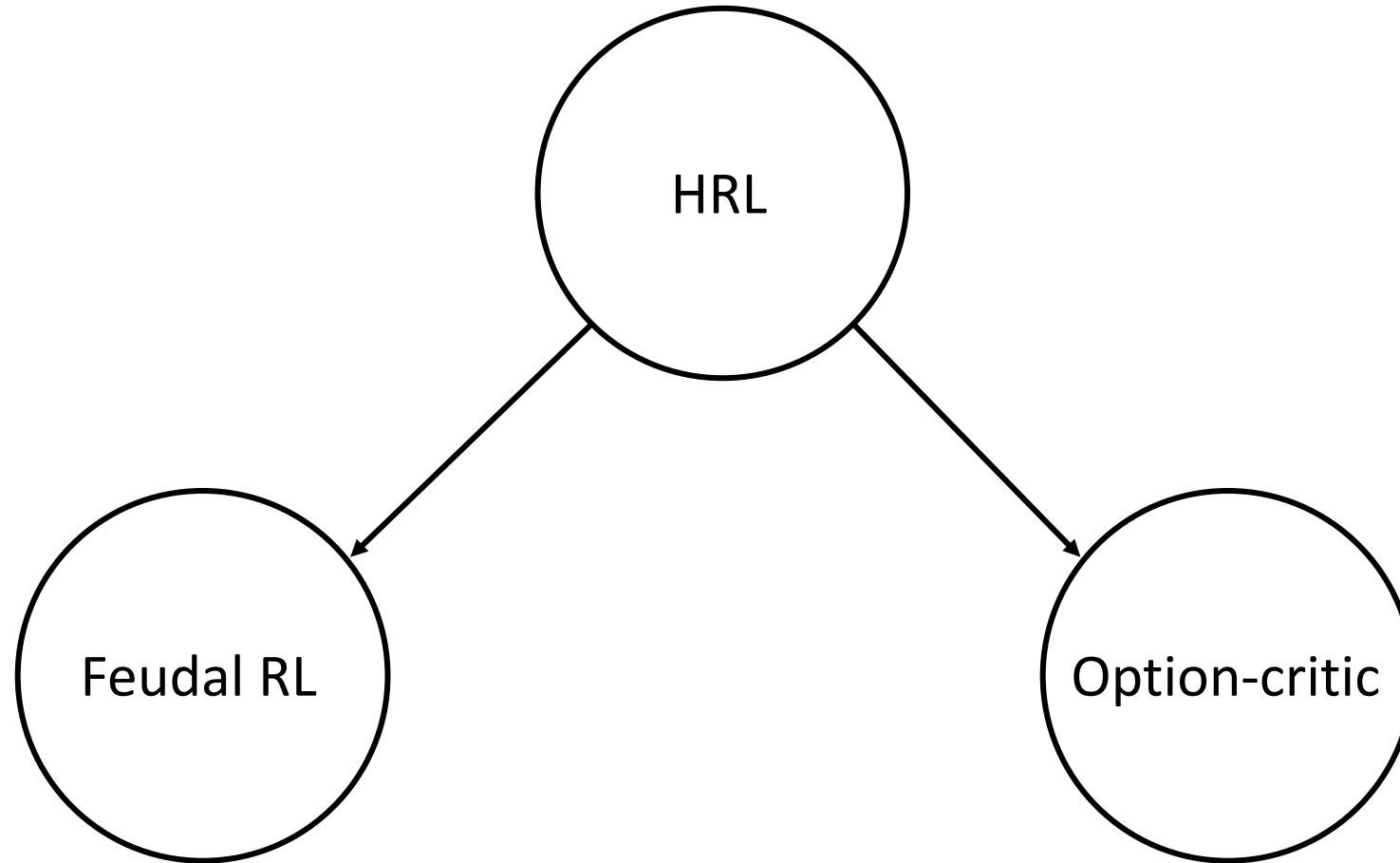
An option ω is defined by 3 parameters $\langle I_\omega, \pi_\omega, \beta_\omega \rangle$:

- I_ω is the initiation set with $I_\omega \subseteq S$
- π_ω is a policy
- β_ω is the termination conditions $\beta_\omega: S \rightarrow [0,1]$

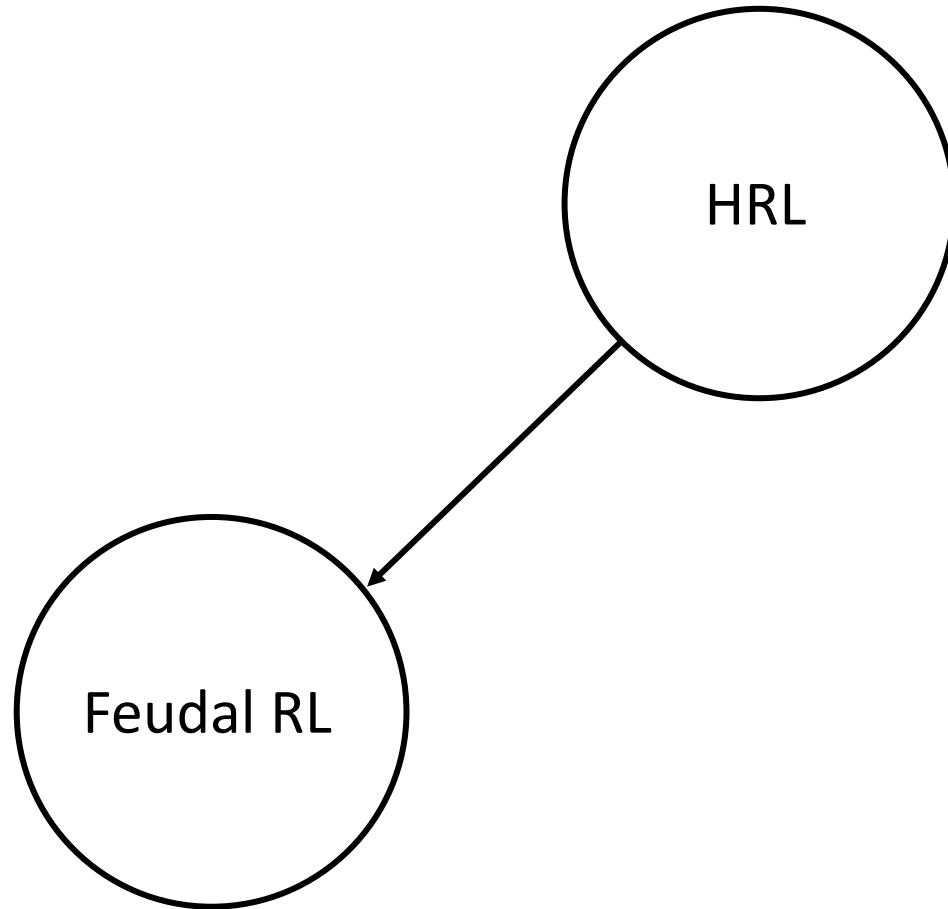
Options
over MDP



The 2 Main Approaches of HRL



Feudal Reinforcement Learning



Feudal Reinforcement Learning

The main idea is to have an hierarchy of managers. Like in Feudalism, managers have **absolute** power over their sub-managers

Feudal Reinforcement Learning

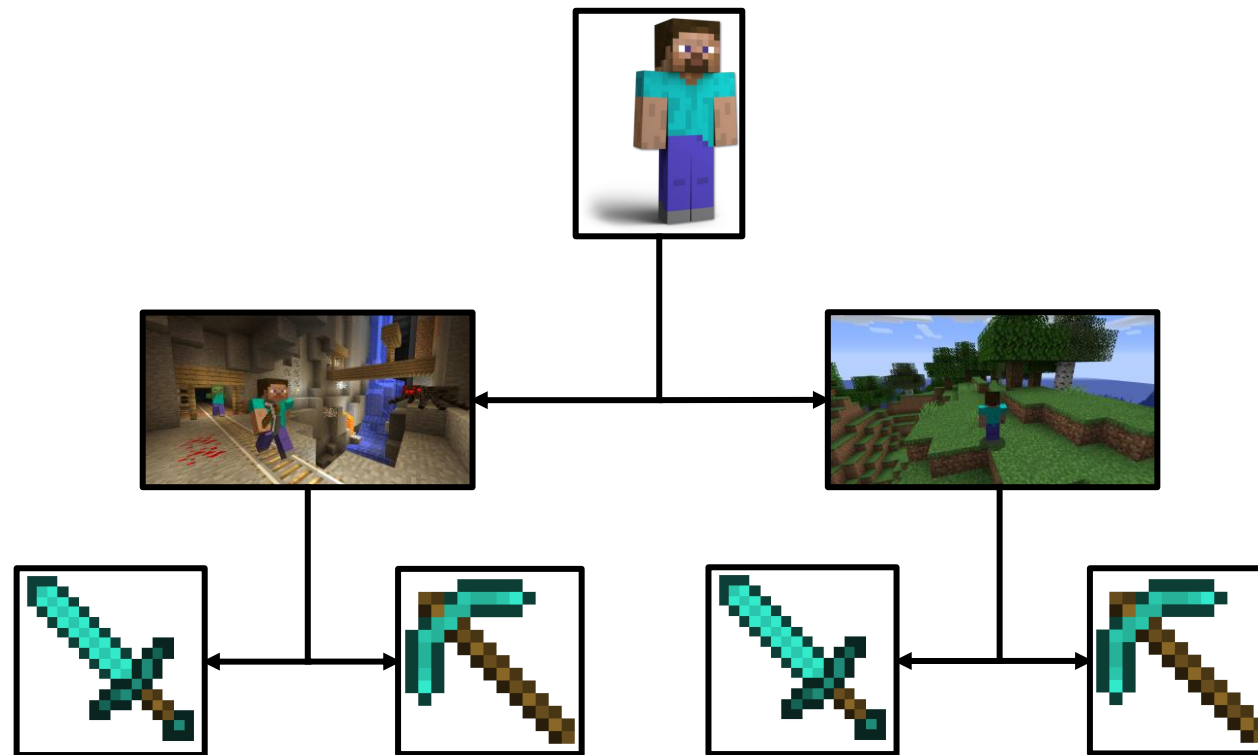
Peter Dayan
CNL
The Salk Institute
PO Box 85800
San Diego CA 92186-5800, USA
dayan@helmholtz.sdsc.edu

Geoffrey E Hinton
Department of Computer Science
University of Toronto
6 Kings College Road, Toronto,
Canada M5S 1A4
hinton@ai.toronto.edu



Feudal Reinforcement Learning

The main idea is to have an hierarchy of managers. Like in Feudalism, managers have **absolute** power over their sub-managers



Reward Hiding

«Managers must reward sub-managers for doing their bidding **whether or not** this satisfies the commands of the super-managers»



Information Hiding

«Managers only need to know the state of the system at the granularity of their own choices of tasks»



From the idea to the implementation

In 2017 Vezhnevets et al. proposed a Deep Reinforcement Learning implementation (**FuN**) of Feudal Reinforcement Learning

FeUdal Networks for Hierarchical Reinforcement Learning

Alexander Sasha Vezhnevets
Simon Osindero
Tom Schaul
Nicolas Heess
Max Jaderberg
David Silver
Koray Kavukcuoglu
DeepMind

VEZHNICK@GOOGLE.COM
OSINDERO@GOOGLE.COM
SCHAUL@GOOGLE.COM
HEESS@GOOGLE.COM
JADERBERG@GOOGLE.COM
DAVIDSILVER@GOOGLE.COM
KORAYK@GOOGLE.COM

Abstract

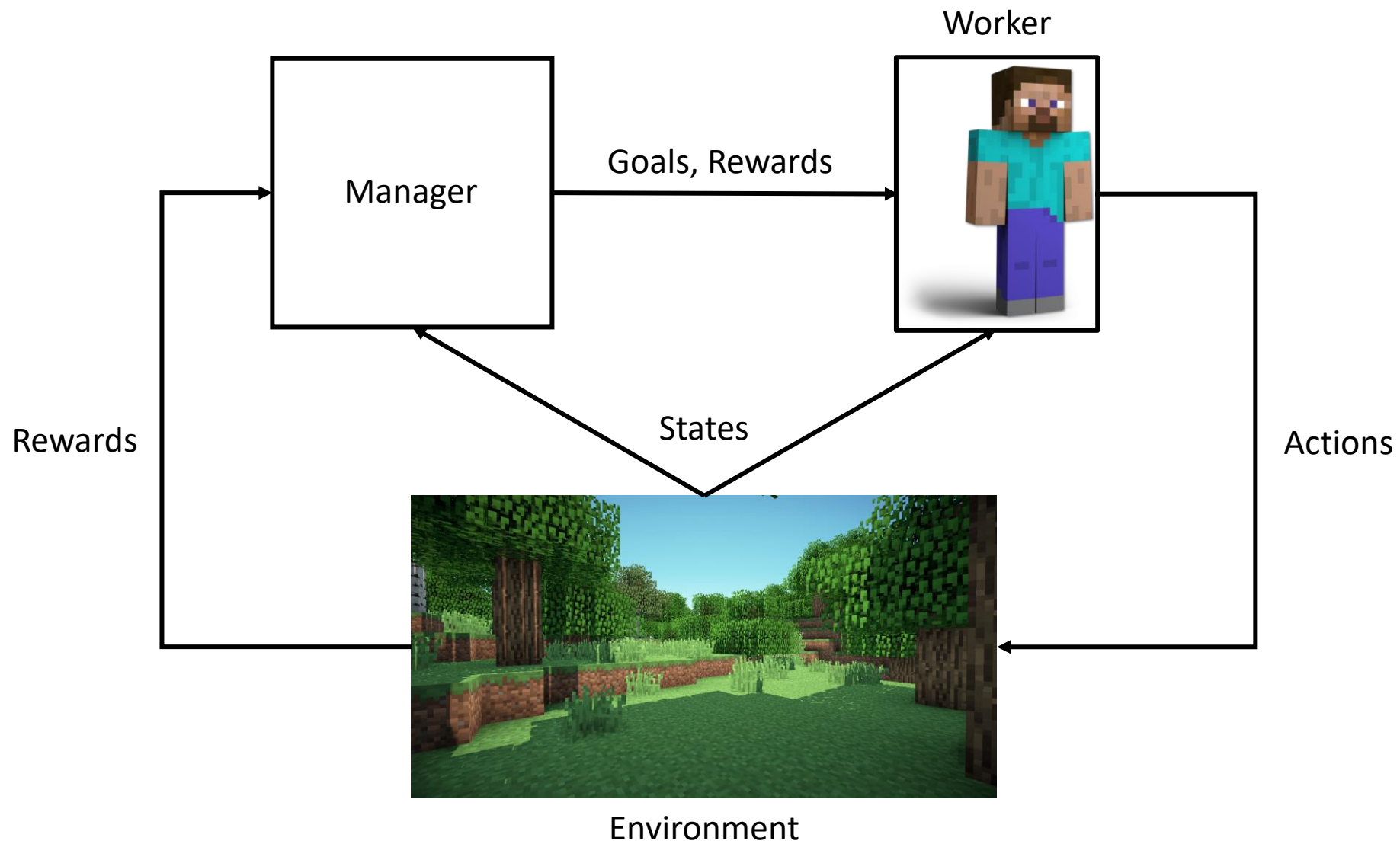
We introduce FeUdal Networks (FuNs): a novel architecture for hierarchical reinforcement learning. Our approach is inspired by the feudal reinforcement learning proposal of Dayan and Hinton, and gains power and efficacy by decoupling end-to-end learning across multiple levels – allowing it to utilise different resolutions of time. Our framework employs a Manager mod-

ains a major challenge for these methods, especially in environments with sparse reward signals, such as the infamous Montezuma's Revenge ATARI game. It is symptomatic that the standard approach on the ATARI benchmark suite (Bellemare et al., 2012) is to use an action-repeat heuristic, where each action translates into several (usually 4) consecutive actions in the environment. Yet another dimension of complexity is seen in non-Markovian environments that require memory – these are particularly challenging, since the agent has to learn which parts of ex-

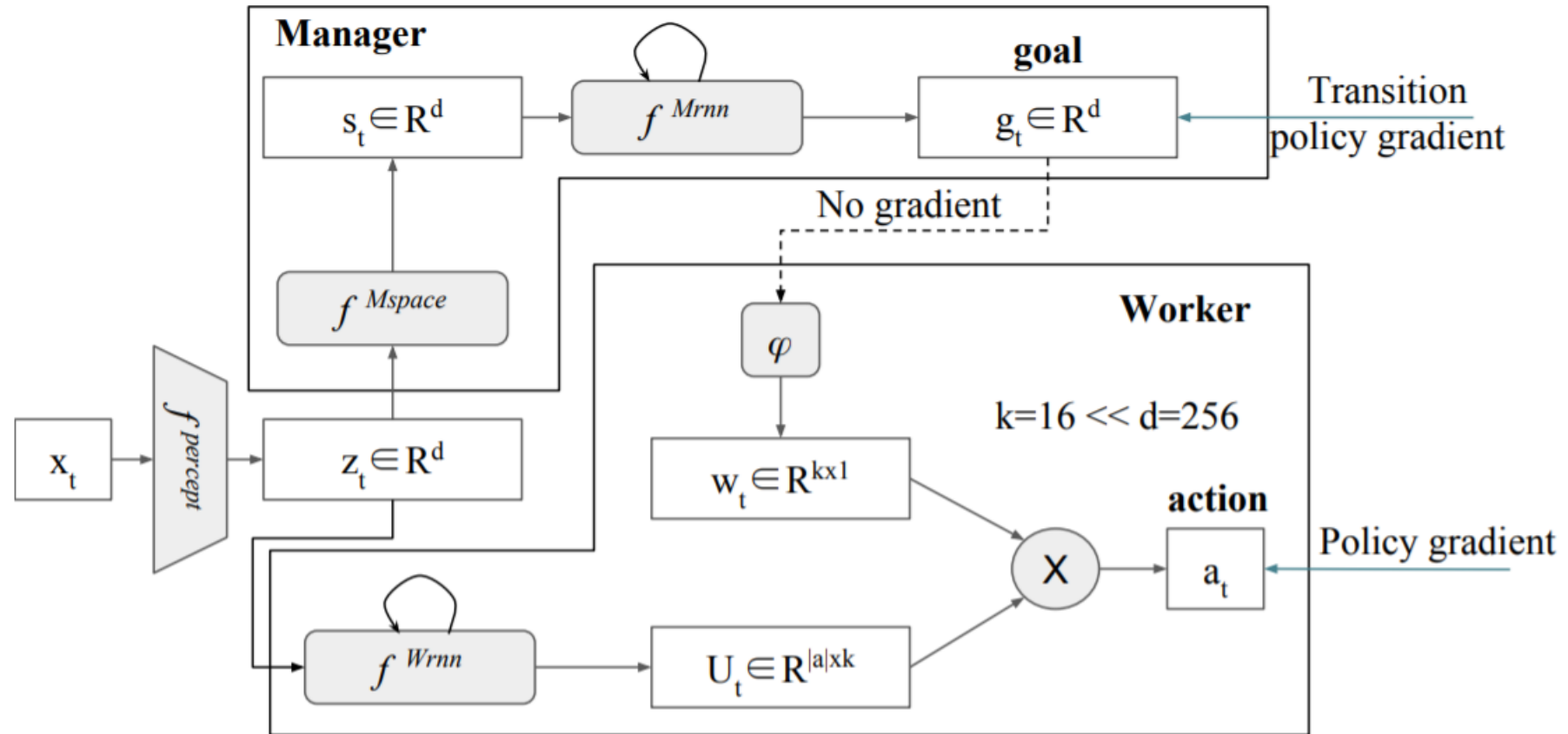
[cs.AI] 6 Mar 2017



FuN architecture

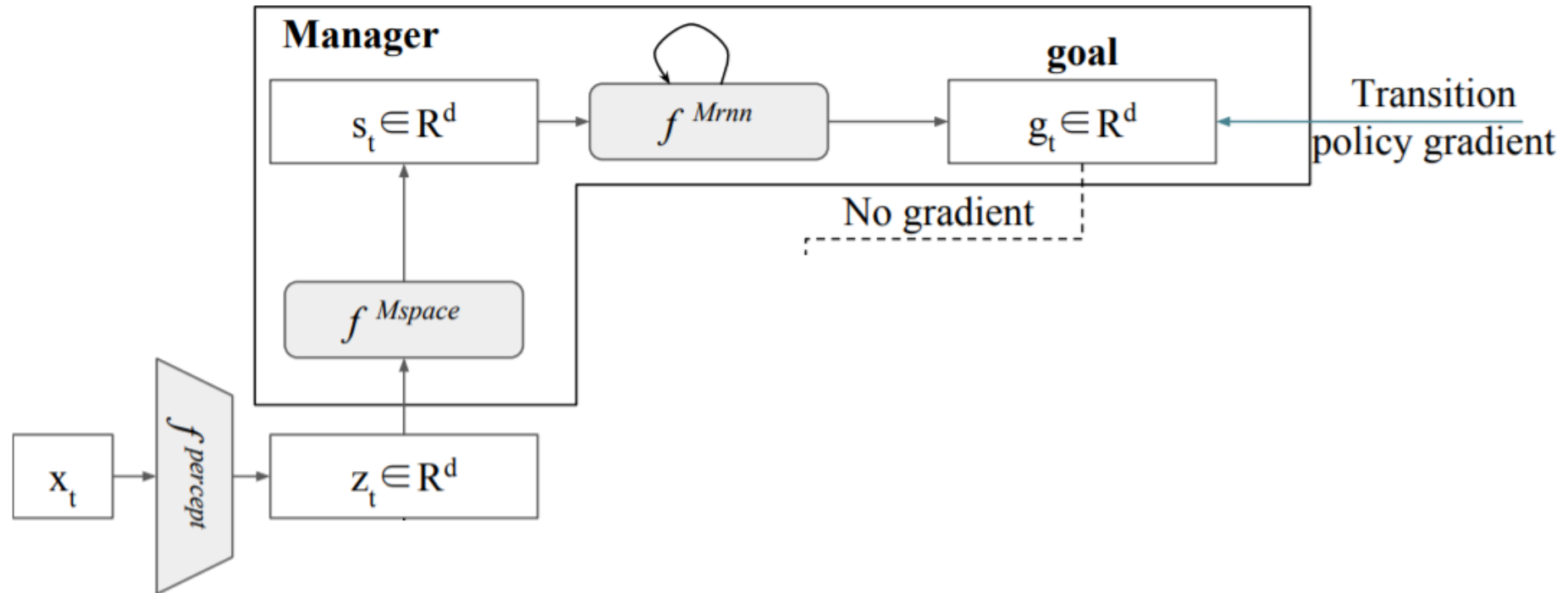


An in-depth look at the agent



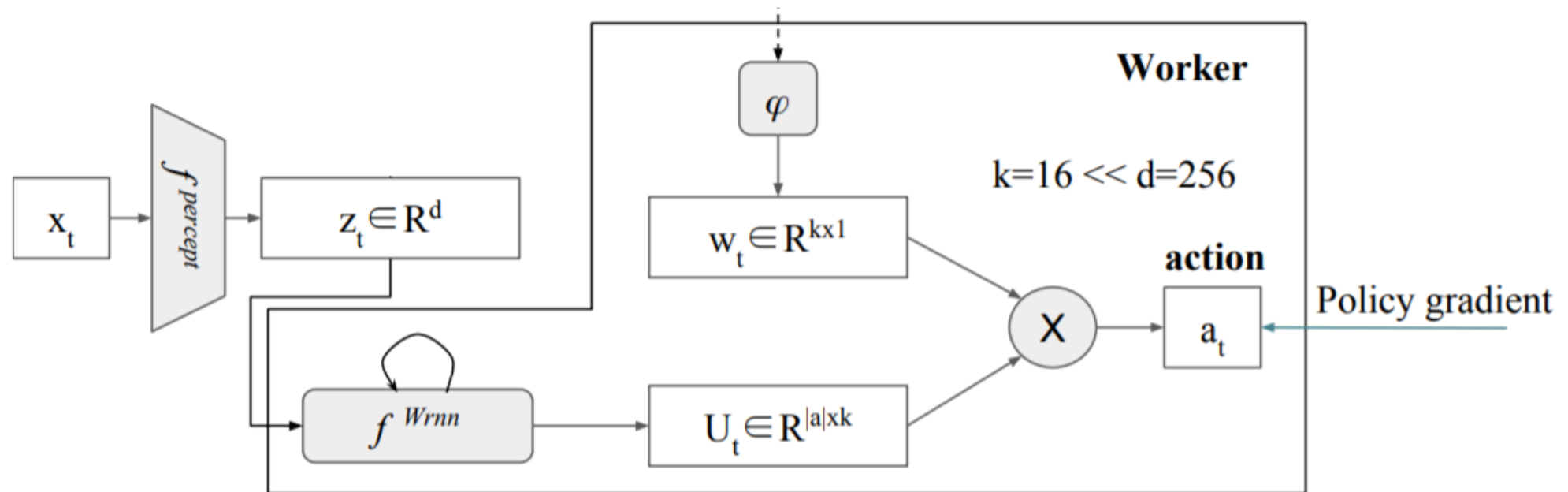
Manager

The manager produces goals for the worker



Worker

The worker acts on the environment according to its goal



Worker's Reward

- Unlike the original FRL formulation, reward hiding is not strictly enforced
- The overall reward is given as:

$$R = R_t + \alpha R_t^I$$

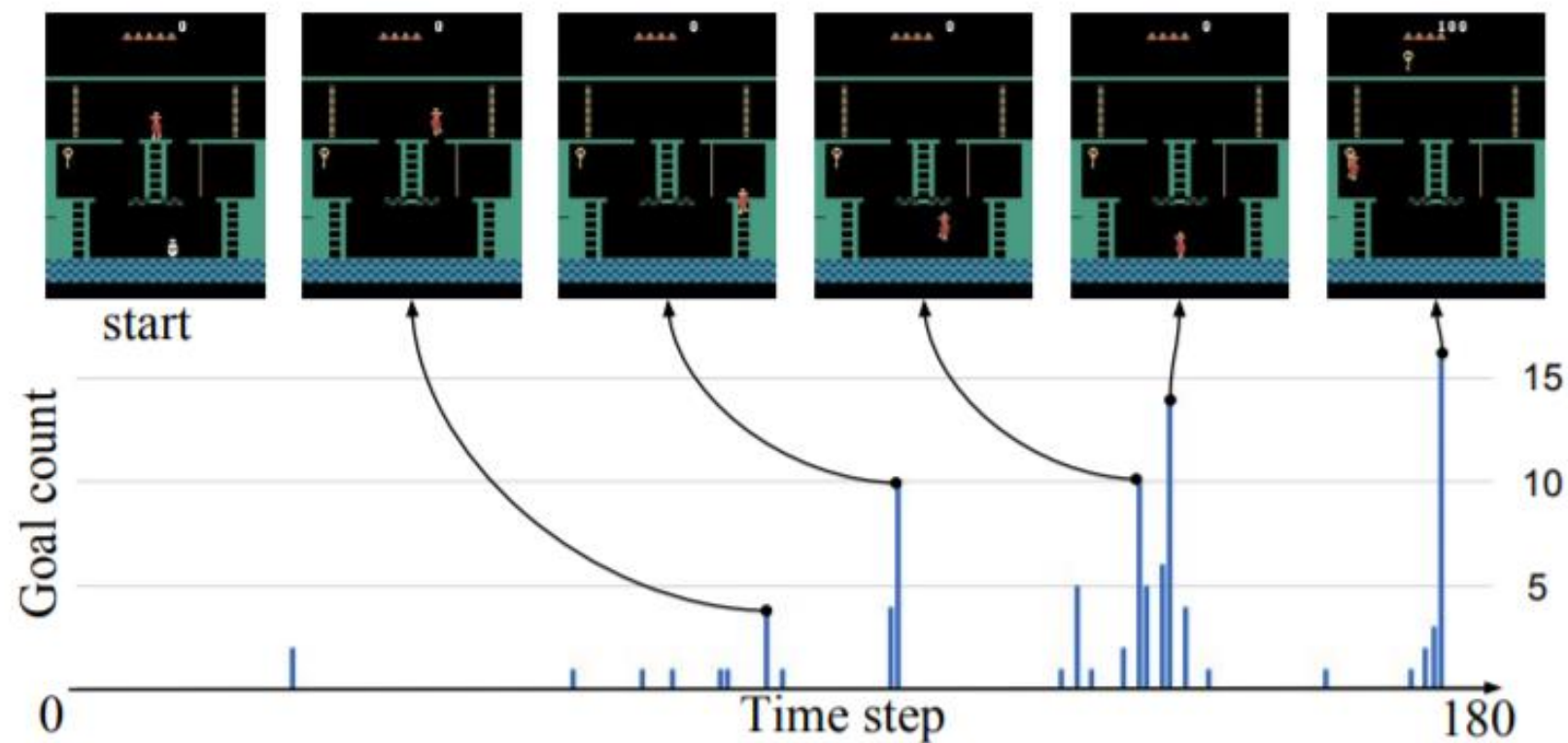
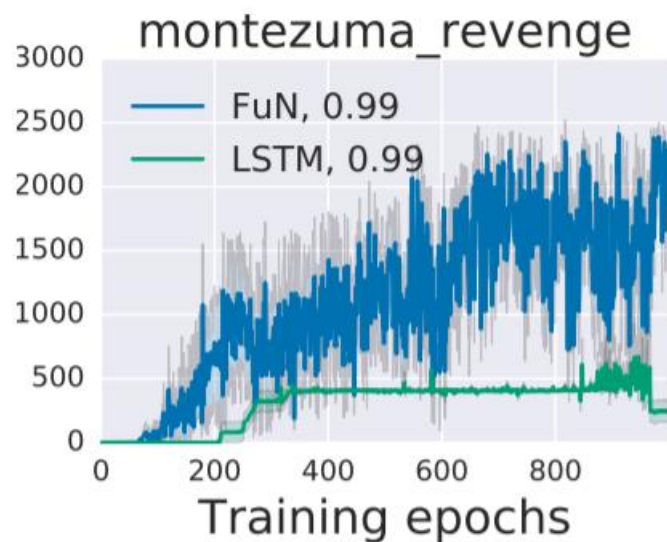
- The intrinsic reward is calculated starting from the goal as:

$$R_t^I = 1/c \sum_1^c d_{\cos}(s_t - s_{t-1}, g_{t-1})$$

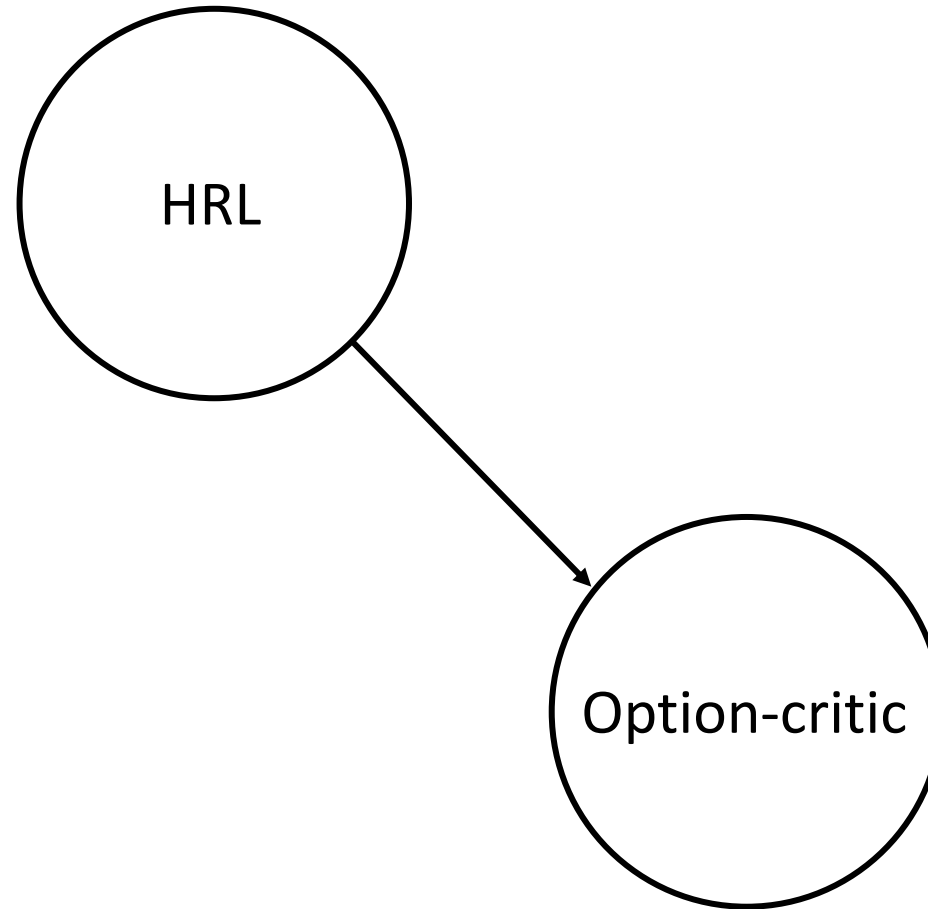


Result on Montezuma's Revenge

FuN obtained great result on Montezuma's Revenge, a game infamous for its **sparse rewards**



Option-Critic



Option-Critic

The Option-Critic architecture extends the Actor-Critic one by introducing options

The Option-Critic Architecture

Pierre-Luc Bacon and Jean Harb and Doina Precup

Reasoning and Learning Lab, School of Computer Science

McGill University

{pbacon, jharb, dprecup}@cs.mcgill.ca

Abstract

Temporal abstraction is key to scaling up learning and planning in reinforcement learning. While planning with temporally extended actions is well understood, creating such abstractions autonomously from data has remained challenging. We tackle this problem in the framework of options [Sutton, Precup & Singh, 1999; Precup, 2000]. We derive policy gradient theorems for options and propose a new *option-critic* architecture capable of learning both the internal policies and the termination conditions of options, in tandem with the policy over options, and without the need to provide any additional rewards or subgoals. Experimental results in both discrete and continuous environments showcase the flexibility and efficiency of the framework.

Introduction

Temporal abstraction allows representing knowledge about courses of action that take place at different time scales. In reinforcement learning, *options* (Sutton, Precup, and Singh 1999; Precup 2000) provide a framework for defining such courses of action and for seamlessly learning and planning with them. *Discovering* temporal abstractions au-

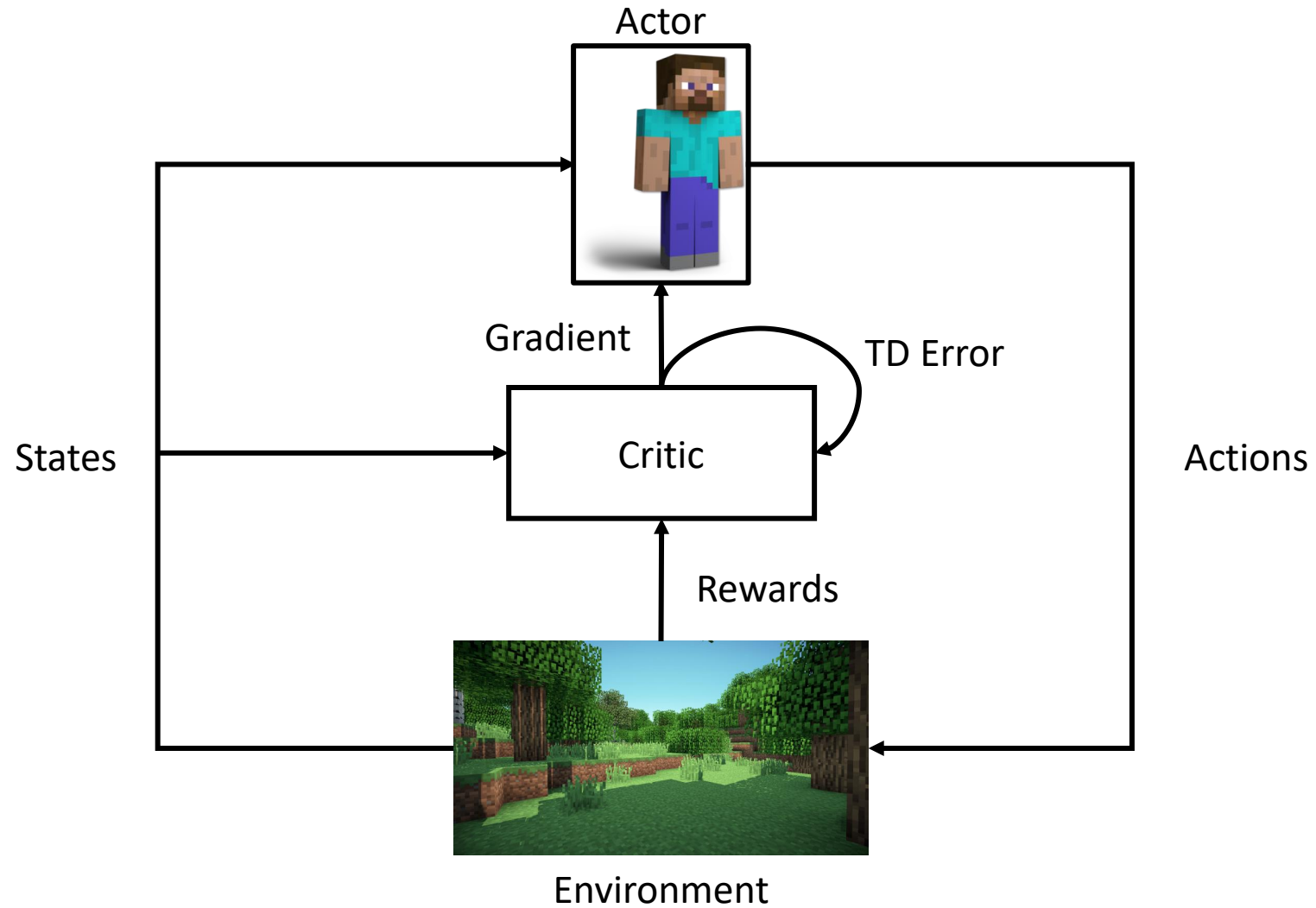
learning process of the intra-option policies and termination functions, simultaneously with the policy over them. This approach works naturally with both linear and non-linear function approximators, under discrete or continuous state and action spaces. Existing methods for learning options are considerably slower when learning from a single task: much of the benefit comes from re-using the learned options in similar tasks. In contrast, we show that our approach is capable of successfully learning options within a single task without incurring any slowdown and while still providing benefits for transfer learning.

We start by reviewing background related to the two main ingredients of our work: policy gradient methods and options. We then describe the core ideas of our approach: the intra-option policy and termination gradient theorems. Additional technical details are included in the appendix. We present experimental results showing that our approach learns meaningful temporally extended behaviors in an effective manner. As opposed to other methods, we only need to specify the number of desired options; it is not necessary to have subgoals, extra rewards, demonstrations, multiple problems or any other special accommodations (however, the approach can take advantage of pseudo-reward functions

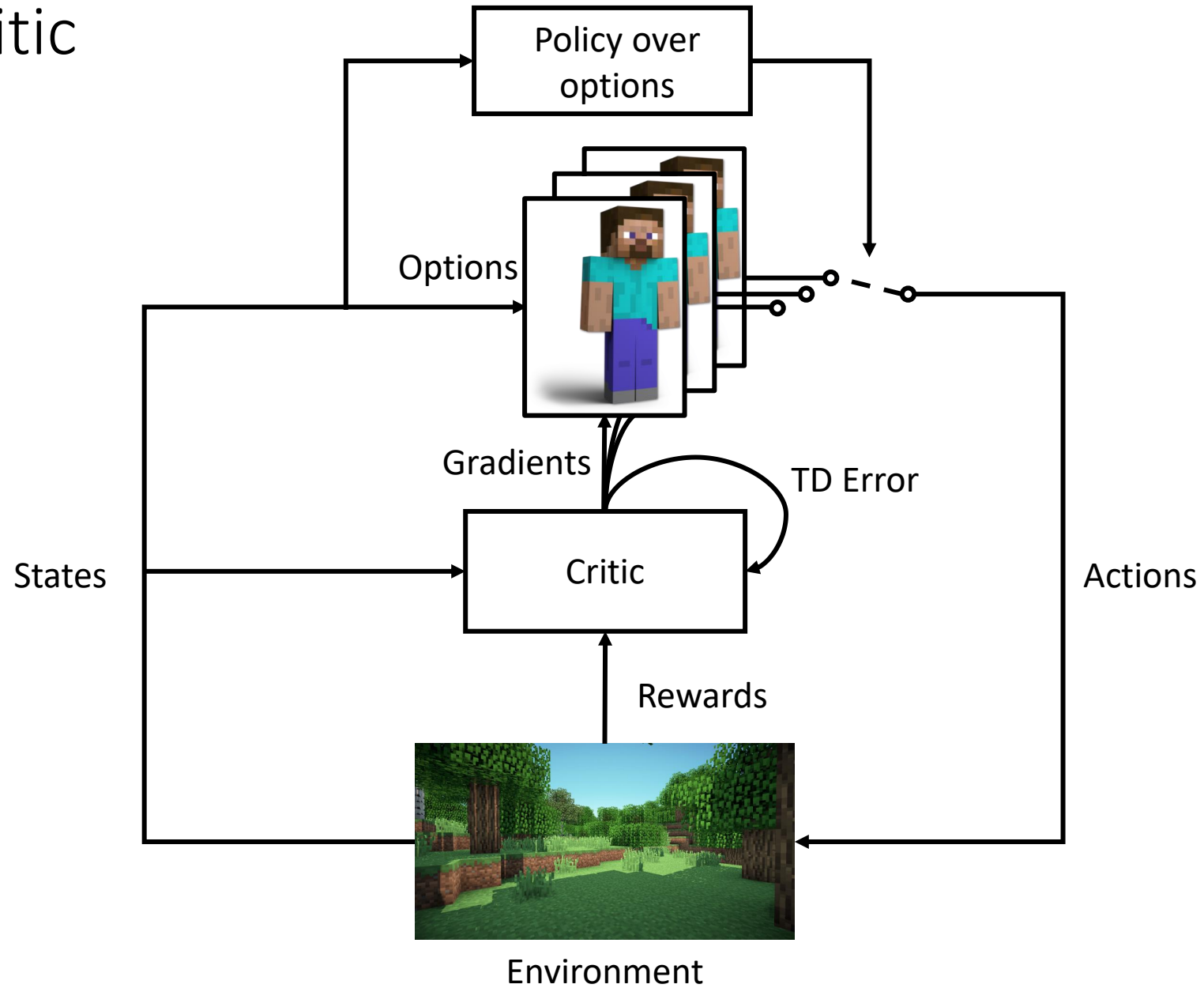
.05140v2 [cs.AI] 3 Dec 2016



A Recap on Actor-Critic



Option-Critic



Intra-Option Learning

For learning an option ω , it is necessary to learn both the option policy $\pi_{\omega,\theta}$ and the termination function $\beta_{\omega,\theta}$.

Intra-Option Policy Gradient Theorem:

$$\frac{\partial Q_{\Omega}(\omega, s)}{\partial \theta} = \mathbb{E} \left[\frac{\partial \pi_{\omega,\theta}(a|s)}{\partial \theta} Q_U(w, s, a) \right]$$

Termination Gradient Theorem:

$$\frac{\partial U(\omega, s)}{\partial v} = \mathbb{E} \left[-\frac{\partial \beta_{\omega,v}(s)}{\partial v} A_{\Omega}(w, s) \right]$$



Critic

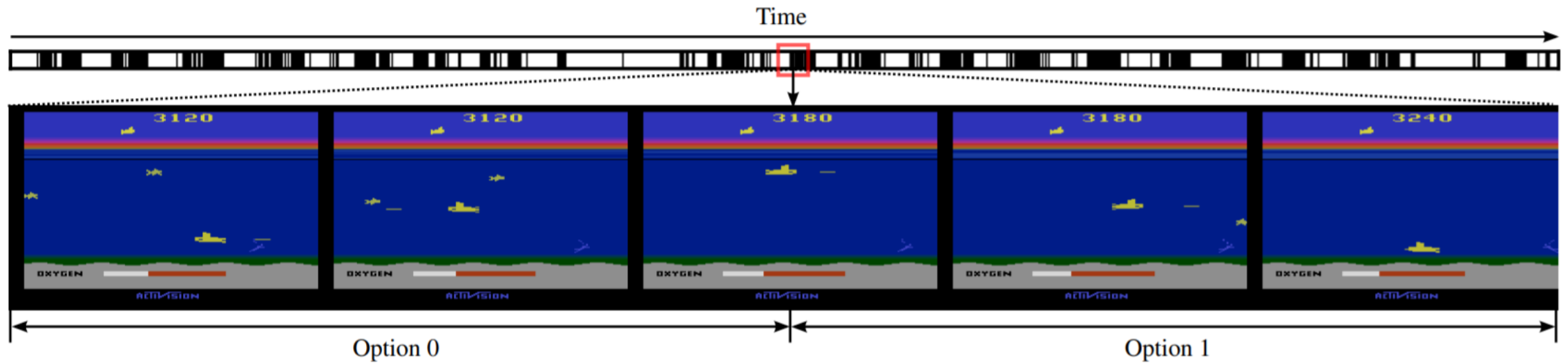
The Critic model learns to approximate the state-option value function $Q_{\Omega}(s, \omega)$.

With it the critic estimates:

- The action-option-state value function $Q_U(w, s, a)$
- The advantage function over options $A_{\Omega}(w, s) = Q_{\Omega}(s, \omega) - V_{\Omega}(s)$ with $V_{\Omega}(s)$ as the value function



Visualizing the Options

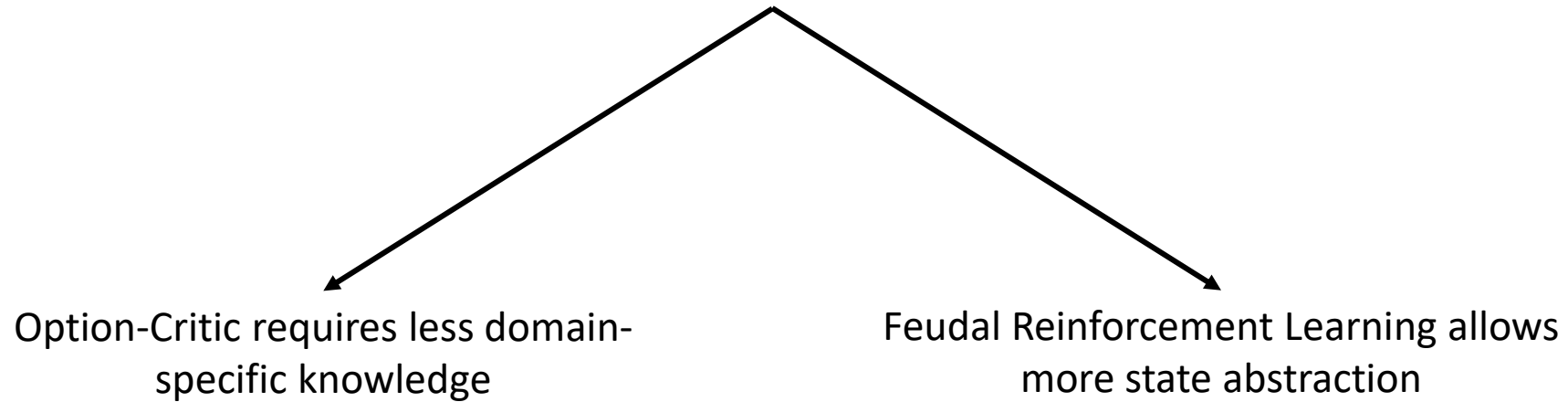


The image is the trajectory of the original implementation of the Option-Critic playing Seaquest with 2 options



FRL vs Option-Critic

In Feudal Reinforcement Learning, sub-tasks must be fixed by the programmer. On the contrary, in Option-Critic sub-tasks are learned automatically



Challenges of Option-Critic

During training, options could collapse into:

- A single active option that completes the entire task
- A set options that changes at every step

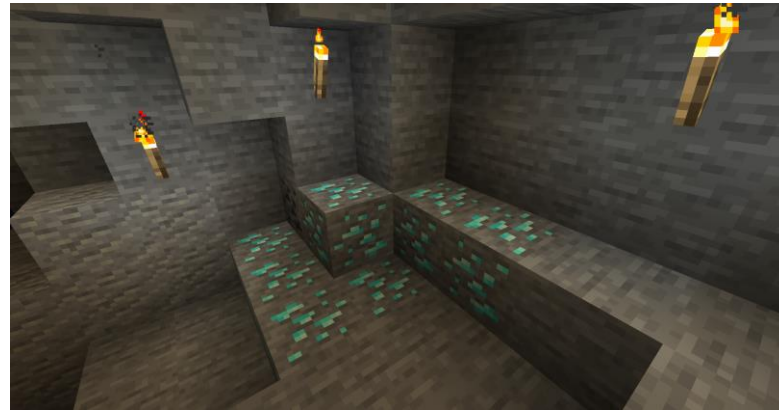
Another limitation of the architecture is **the assumption that options can apply everywhere**



Challenges of FuN

The main challenge of FuN is the fact that the goal depends on the state. In particular, it depends on the distance function used to generate the rewards.

Superficially similar states can lead to wrong/ineffective goal generations!



Thank for your attention
I hope we will have a nice discussion!

To be continued in Edoardo's presentation...



References

Presented Papers

- [Feudal Reinforcement Learning \(Dayan et al., 1993\)](#)
- [FeUdal Networks for Hierarchical Reinforcement Learning \(Vezhnevets et al., 2017\)](#)
- [The Option-Critic Architecture \(Precup et al., 2016\)](#)

Additional Resources

- [Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning \(Sutton et al. 1999\)](#)
- <https://thegradients.pub/the-promise-of-hierarchical-reinforcement-learning/>
- <https://towardsdatascience.com/hierarchical-reinforcement-learning-a2cca9b76097>

MineRL

- Official website: <https://minerl.io/>
- Presentation paper: <https://arxiv.org/pdf/2101.11071.pdf>