Microsoft

Reinforcement Learning

Lecture at MLSS 2019
Part 2

Katja Hofmann

Game Intelligence
Microsoft Research, Cambridge, UK
aka.ms/gameintelligence

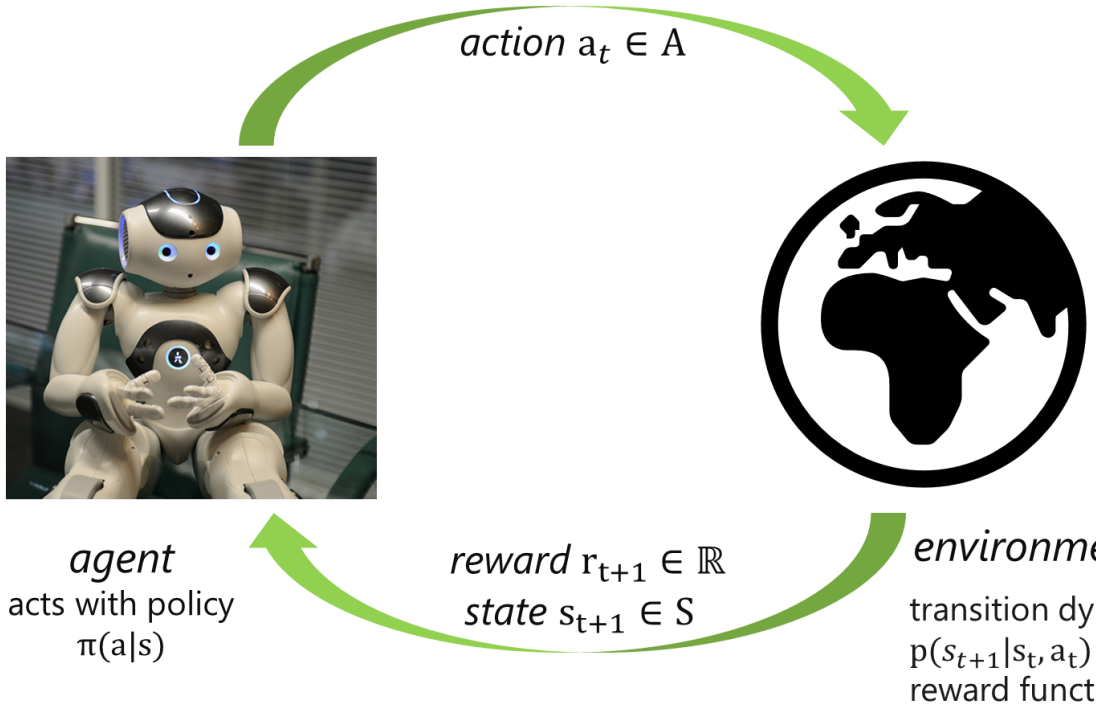
@katjahofmann

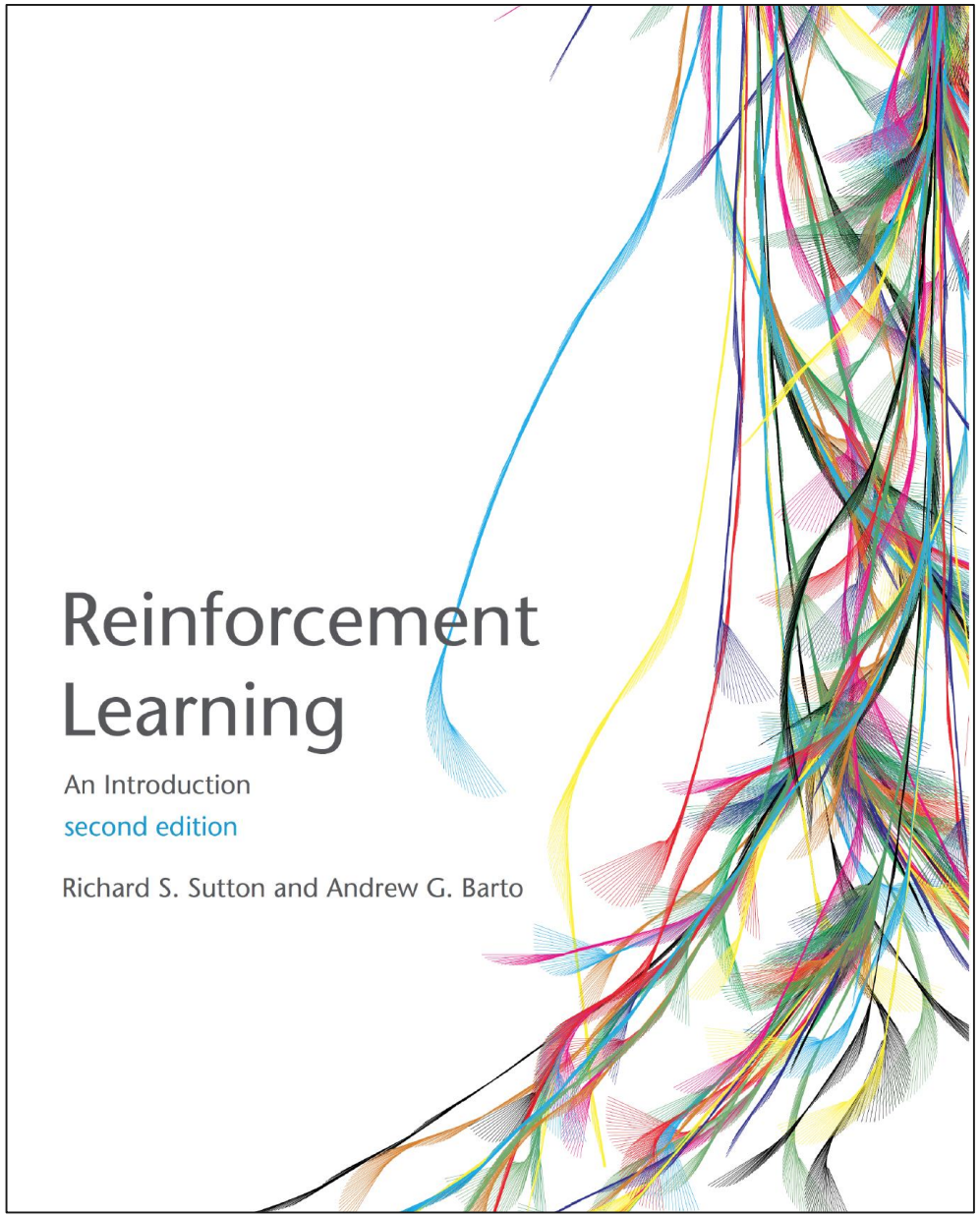
Yesterday

History

Formalizing RL

Exploration and Exploitation





http://incompleteideas.net/book/the-book-2nd.html

Agenda

Lecture 1

Case Studies (and a bit of history)

Formalizing RL

Exploration and Exploitation

Lecture 2

RL Approaches 1:
Policy Gradient Methods

RL Approaches 2:
Temporal Differences, Q-Learning

Optional: Multi-task RL

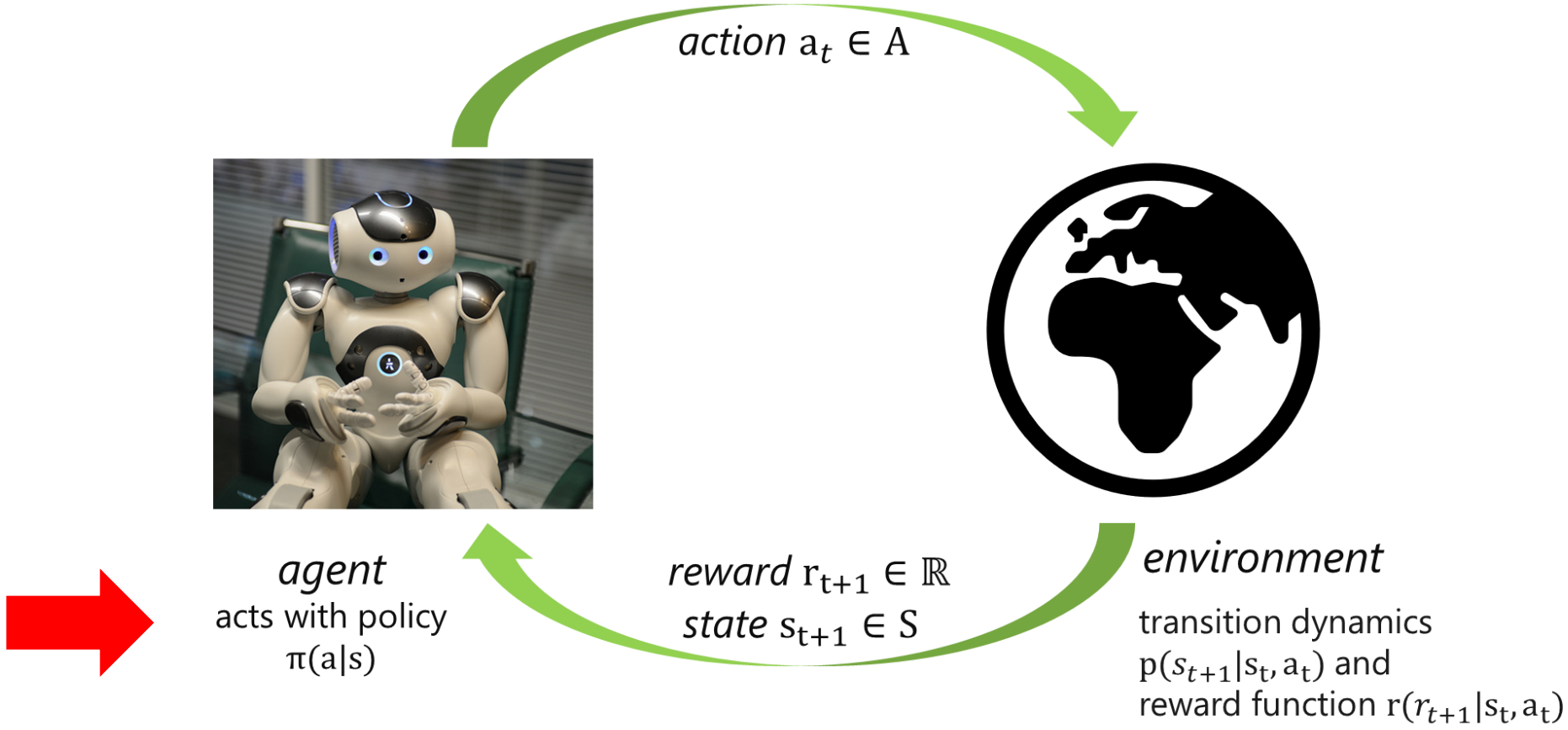
Tutorial

Implementing a Deep Q Agent in Minecraft

RL Approaches 1:
Policy Gradient Methods

Policy Gradient: Intuition

Focus on learning a good behaviour policy



Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems

π	.5	.5

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems

π	.5	.5

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems

π	.5	.5

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems

π	.45	.55

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

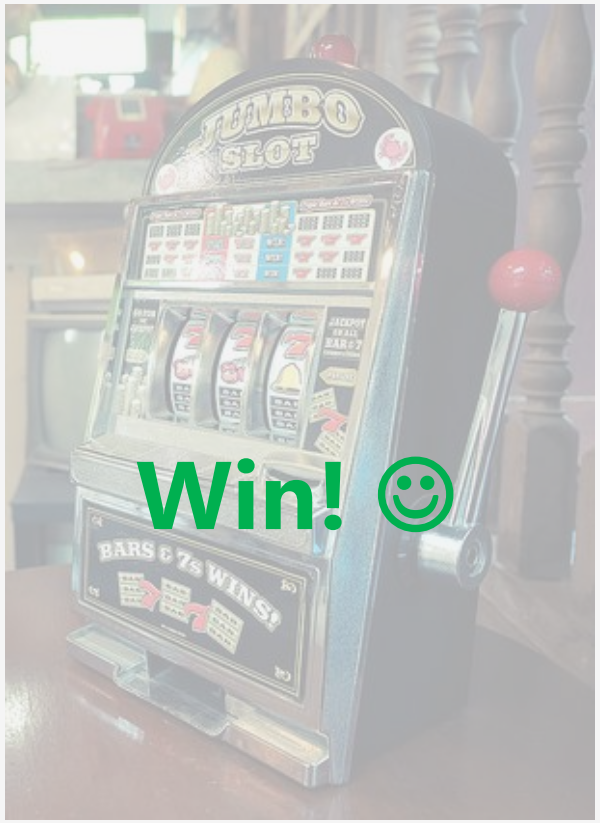
Example: Learning in Multi-armed bandit problems

π	.45	.55

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems



π	.45	.55

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems

π	.4	.6

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

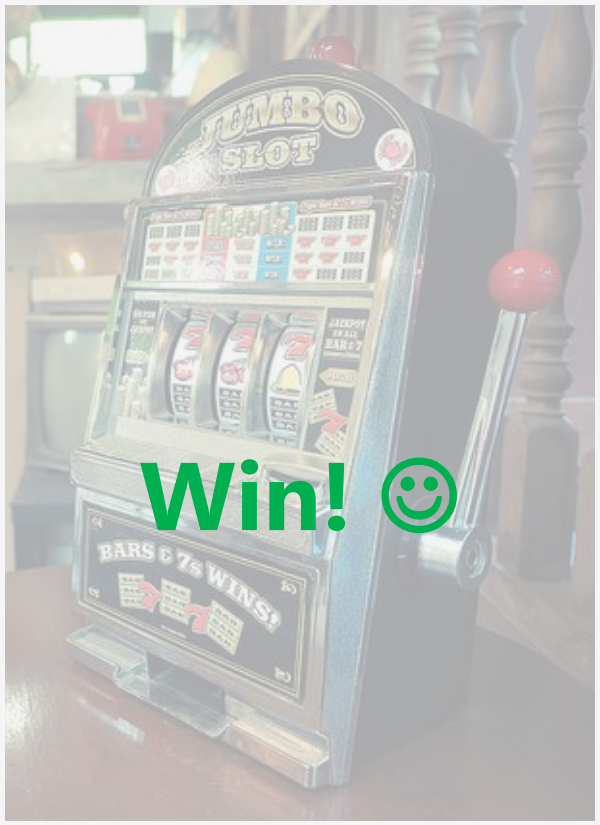
Example: Learning in Multi-armed bandit problems

π	.4	.6

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems



π	.4	.6

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Example: Learning in Multi-armed bandit problems

π	.45	.55

Photo credit: https://www.flickr.com/photos/knothing/11264853546/

Policy Gradient: Intuition

Focus on learning a good behaviour policy

Repeat:

1. Collect experience using the current policy
2. Update the policy towards better outcomes

Focus on the Policy: Parametric Form

Most common parameterization:

$$\pi(a|s;\theta) = \frac{e^{h(s,a;\theta)}}{\sum_{a' \in A} e^{h(s,a';\theta)}}$$

Focus on the Policy: Parametric Form

Most common parameterization:

$$\pi(a|s;\theta) = \frac{e^{h(s,a;\theta)}}{\sum_{a' \in A} e^{h(s,a';\theta)}}$$

Policy: **probability distribution** over actions, given the current state

Focus on the Policy: Parametric Form

Most common parameterization:

$$\pi(a|s;\theta) = \frac{e^{h(s,a;\theta)}}{\sum_{a' \in A} e^{h(s,a';\theta)}}$$

The policy has **learnable parameters**

Focus on the Policy: Parametric Form

Most common parameterization:

$$\pi(a|s;\theta) = \frac{e^{h(s,a;\theta)}}{\sum_{a' \in A} e^{h(s,a';\theta)}}$$

The policy parameters encode **action preferences** – more preferred actions are more likely

Focus on the Policy: Parametric Form

Most common parameterization:

$$\pi(a|s;\theta) = \frac{e^{h(s,a;\theta)}}{\sum_{a' \in A} e^{h(s,a';\theta)}}$$

The denominator ensures **normalized probabilities**

Policy Gradient Objective

Goal: find parameters θ that maximize expected reward

$$J(\theta) = \sum_{s \in S} p^{\pi}(s) \sum_{a \in A} \pi(a|s;\theta) R_s^a$$

Policy Gradient Objective

Goal: find parameters θ that maximize expected reward

$$J(\theta) = \sum_{s \in S} p^{\pi}(s) \sum_{a \in A} \pi(a|s;\theta) R_s^a$$

Stationary state distribution under π_θ

Policy Gradient Objective

Goal: find parameters θ that maximize expected reward

$$J(\theta) = \sum_{s \in S} p^{\pi}(s) \sum_{a \in A} \pi(a|s;\theta) R_s^a$$

Stationary state distribution under π_θ

The parameterized policy

Policy Gradient Objective

Goal: find parameters θ that maximize expected reward

$$J(\theta) = \sum_{s \in S} p^{\pi}(s) \sum_{a \in A} \pi(a|s;\theta) R_s^a$$

Stationary state distribution under π_θ

The parameterized policy

Expected return when starting from state s and taking action a

Policy Gradient Objective

Goal: find parameters θ that maximize expected reward

$$J(\theta) = \sum_{s \in S} p^{\pi}(s) \sum_{a \in A} \pi(a|s;\theta) R_s^a$$

$$\nabla J(\theta) \propto \sum_{s \in S} p^{\pi}(s) \sum_{a \in A} \pi(a|s;\theta) \nabla \log \pi(a|s;\theta)\, R_s^a$$

Challenge: compute update to parameterized policy – which depends on the unknown environment dynamics

The Policy Gradient Theorem

Key insight: gradient of J does not require derivatives of $p^{\pi}(s)$

$$\nabla J(\theta) \propto \sum_{s \in S} p^{\pi}(s) \sum_{a \in A} \pi(a|s;\theta) \nabla \log \pi(a|s;\theta)\, R_s^a$$

$$= \mathbb{E}_{\pi}[\nabla \log \pi(a|s;\theta)\, R_s^a]$$

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *NIPS 2000*.
See Sutton & Barto 2018, chapter 13

The Policy Gradient Theorem

Terms can be estimated from data!

$$\mathbb{E}_{\pi}[\nabla \log \pi(a|s;\theta)\, R_s^a]$$

Rollout the current policy π

Implement forward pass in your favourite deep learning framework + auto-diff to compute gradients

Episode returns under π

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *NIPS 2000*.
See Sutton & Barto 2018, chapter 13

Policy Gradient Algorithm: REINFORCE

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229-256.
Algorithm from: Sutton & Barto 2018, chapter 13, page 328

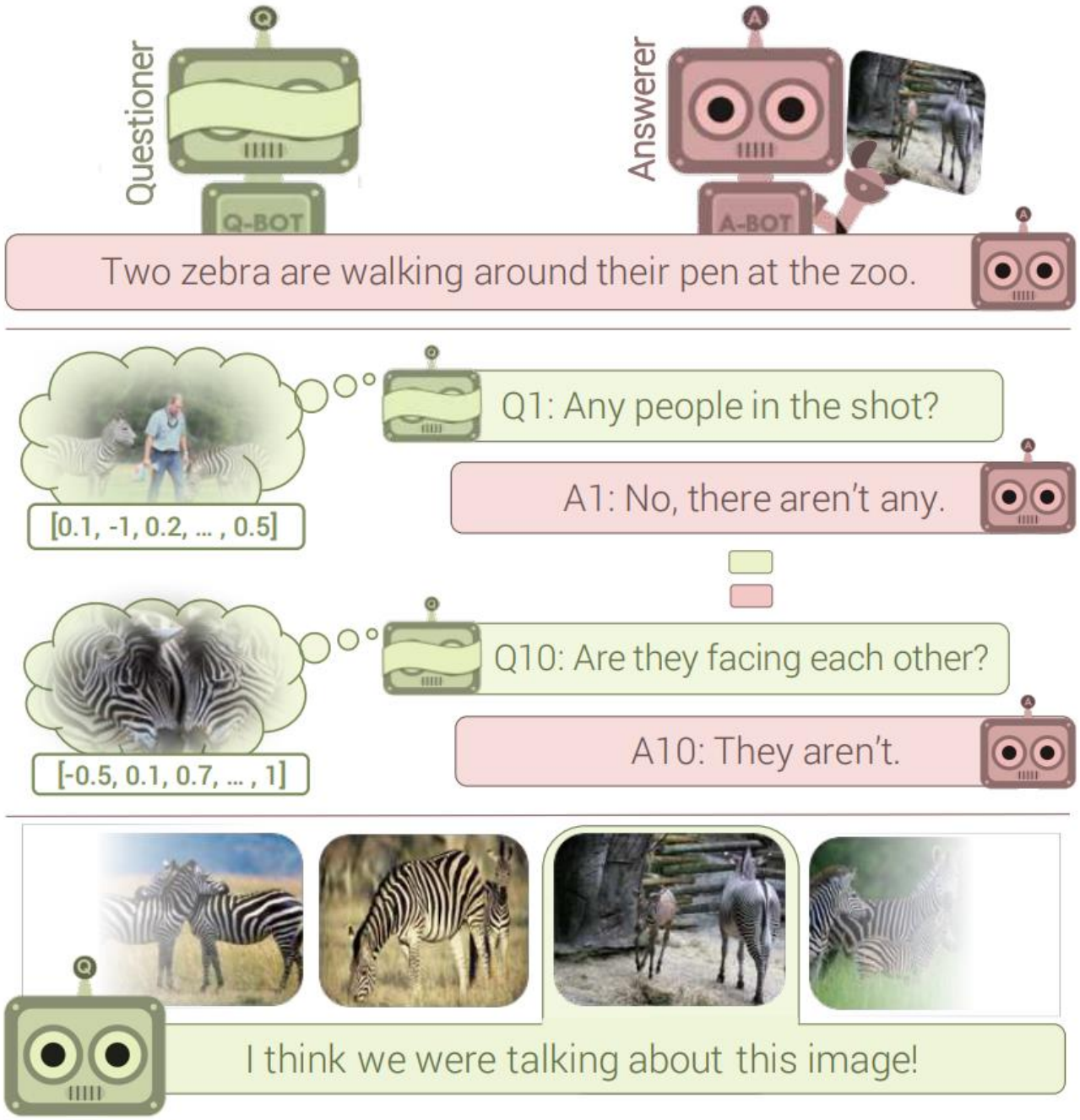
Case Study: Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning

Das, A., Kottur, S., Moura, J. M., Lee, S., & Batra, D. (2017). *ICCV, 2017*.

https://visualdialog.org/

Key insight: REINFORCE can improve dialog policy on top of supervised (imitation) learning

$$r_t\Big(\underbrace{s_t^Q}_{\text{state}}, \underbrace{(q_t, a_t, y_t)}_{\text{action}}\Big) = \underbrace{\ell\left(\hat{y}_{t-1}, y^{gt}\right)}_{\text{distance at } t\text{-}1} - \underbrace{\ell\left(\hat{y}_t, y^{gt}\right)}_{\text{distance at } t}$$



Case Study: Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning

Das, A., Kottur, S., Moura, J. M., Lee, S., & Batra, D. (2017). *ICCV, 2017.*

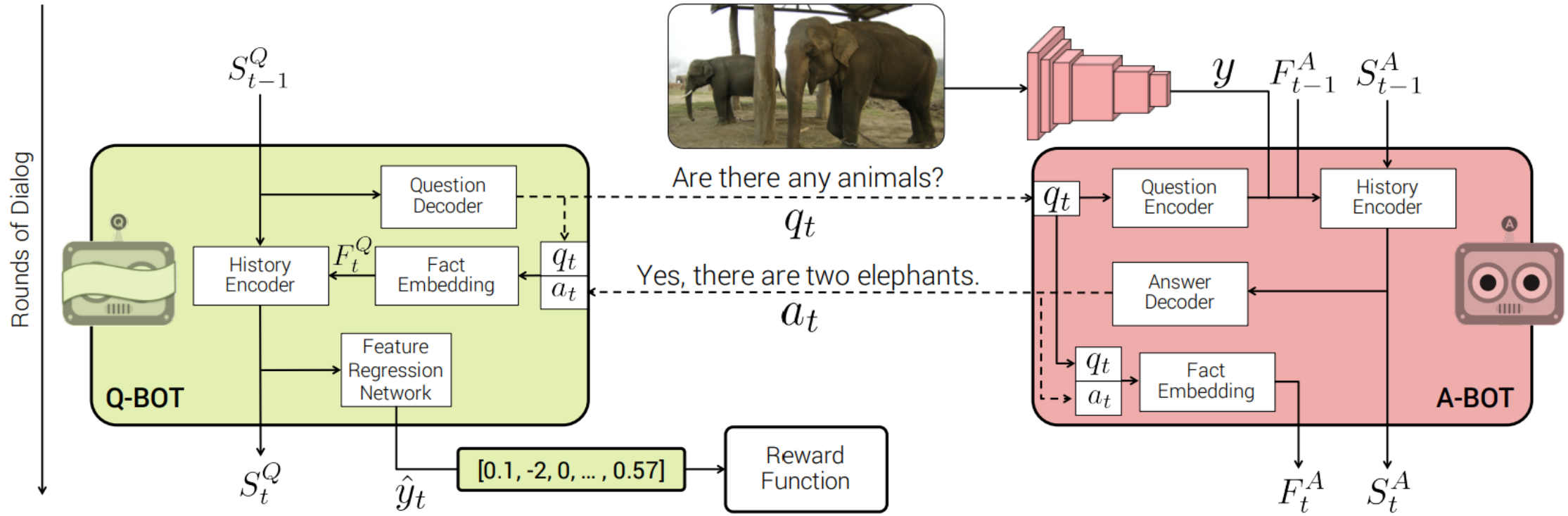


Figure 2: Policy networks for Q-BOT and A-BOT. At each round t of dialog, (1) Q-BOT generates a question q_t from its question decoder conditioned on its state encoding S^Q_{t-1}, (2) A-BOT encodes q_t, updates its state encoding S^A_t, and generates an answer a_t, (3) both encode the completed exchange as F^Q_t and F^A_t, and (4) Q-BOT updates its state to S^Q_t, predicts an image representation $\hat{y}_t$, and receives a reward.

Further Reading

Research focus: reduce variance, local vs global optima, exploration.

Examples:

Lilian Weng's Blog: https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

[Haarnoja et al. '18] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine: *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. ICML 2018.

[Schulman et al. '17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. *Proximal Policy Optimization Algorithms*. Arxiv: 1707.06347

[Sutton & Barto '98] R.S. Sutton & A.G. Barto: *Reinforcement Learning: An Introduction*. MIT Press, 1998.

RL Approaches 2:
Temporal Differences, Q-Learning

Temporal Difference - Overview

Policy Gradient: focused policy improvement

+ Can learn stochastic policies, effective in continuous or high-dimensional action spaces
- Typically converge to local optimum, often high variance

Next – TD methods:

- Focus on estimating policy **value**
- Intuitively – estimate how good an action is in a given situation
- Key insight: values can be estimated efficiently by bootstrapping from previous estimates

Running Example: TD-Learning in Malmo

Task: cliff walking – the agent has to learn to navigate to the blue goal block

Adapted from Sutton & Barto 2018, chapter 6

Try this at home, see https://github.com/Microsoft/malmo - tutorial 6

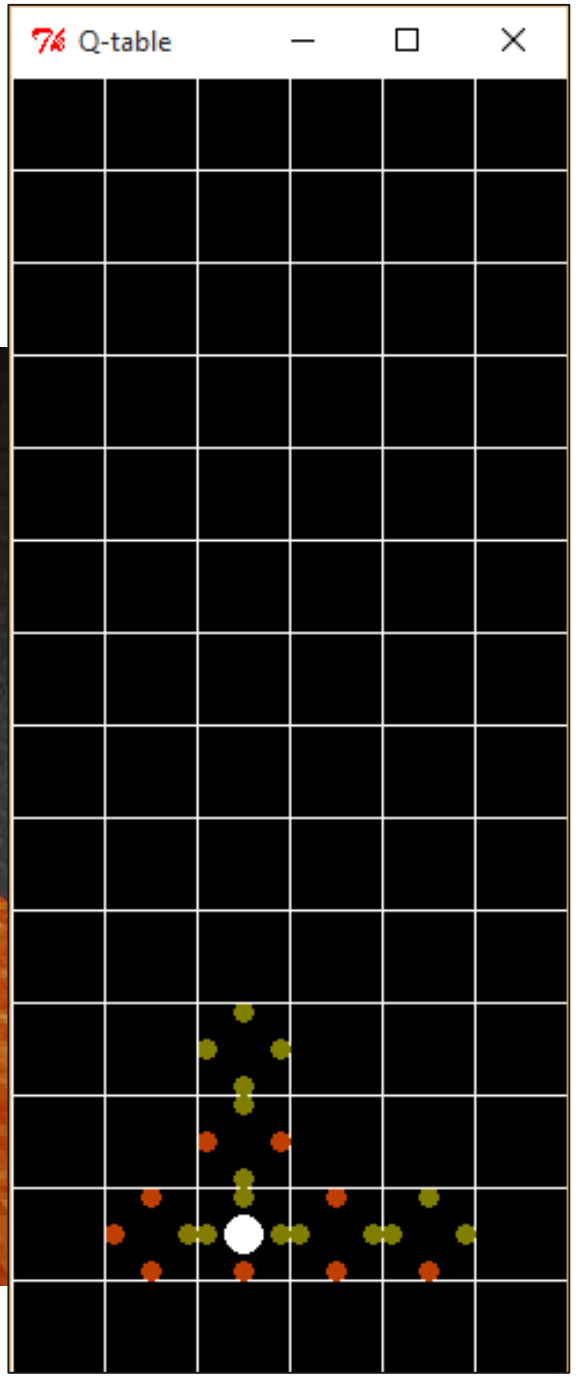
Running Example: TD-Learning in Malmo

Task: cliff walking – the agent has to learn to navigate to the blue goal block

Adapted from Sutton & Barto 2018, chapter 6

Try this at home, see https://github.com/Microsoft/malmo - tutorial 6

States, actions, rewards …



Performance of a Random Policy

Challenge: Data Efficiency

With basic policy gradient / REINFORCE, require many policy rollouts to estimate returns.

Can we do better?

Challenge: Data Efficiency

With basic policy gradient / REINFORCE, require many policy rollouts to estimate returns.

Can we do better?

Yes – using ideas from *dynamic programming*

Action-Value (Q) Function

Define the action-value function

$$\mathrm{Q}_{\pi}(s_t, a_t) \equiv \mathbb{E}_{\pi}[G_t | s_t, a_t]$$

Action-Value (Q) Function

Define the action-value function

$$\mathrm{Q}_\pi(s_t, a_t) \equiv \mathbb{E}_\pi[G_t | s_t, a_t] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t, a_t\right]$$

Bellman Equation

The Bellman equation for Q defines recursively:

$$\mathrm{Q}_\pi(s_t, a_t) \equiv \mathbb{E}_\pi[G_t|s_t, a_t] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t, a_t\right]$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma \mathbb{E}_\pi[G_{t+1}|s_{t+1}, a_{t+1}]|s_t, a_t]$$

Bellman Equation

The Bellman equation for Q defines recursively:

$$\mathrm{Q}_\pi(s_t, a_t) \equiv \mathbb{E}_\pi[G_t|s_t, a_t] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t, a_t\right]$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma \mathbb{E}_\pi[G_{t+1}|s_{t+1}, a_{t+1}]|s_t, a_t]$$

$$= \mathbb{E}_\pi[r_{t+1} + \gamma \mathrm{Q}_\pi(s_{t+1}, a_{t+1})|s_t, a_t]$$

Temporal Difference (TD) Error

If Q-value estimates are accurate, the following must hold:

$$\mathrm{Q}_{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[r_{t+1} + \gamma \mathrm{Q}_{\pi}(s_{t+1}, a_{t+1}) | s_t, a_t]$$

Temporal Difference (TD) Error

If Q-value estimates are accurate, the following must hold:

$$\mathrm{Q}_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma \mathrm{Q}_\pi(s_{t+1}, a_{t+1}) | s_t, a_t]$$

If not, there is an error:

$$\delta = \mathrm{Q}_\pi(s_t, a_t) - \mathbb{E}_\pi[r_{t+1} + \gamma \mathrm{Q}_\pi(s_{t+1}, a_{t+1}) | s_t, a_t]$$

To learn better Q-value estimates – minimize δ

Q-Learning Algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

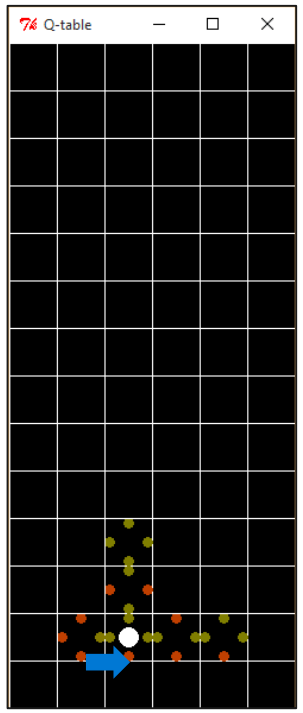
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* (Doctoral dissertation, King's College, Cambridge).
Dayan, P., & Watkins, C. J. C. H. (1992). Q-learning. *Machine learning*, *8*(3).
Algorithm from: Sutton & Barto 2018, chapter 6, page 131

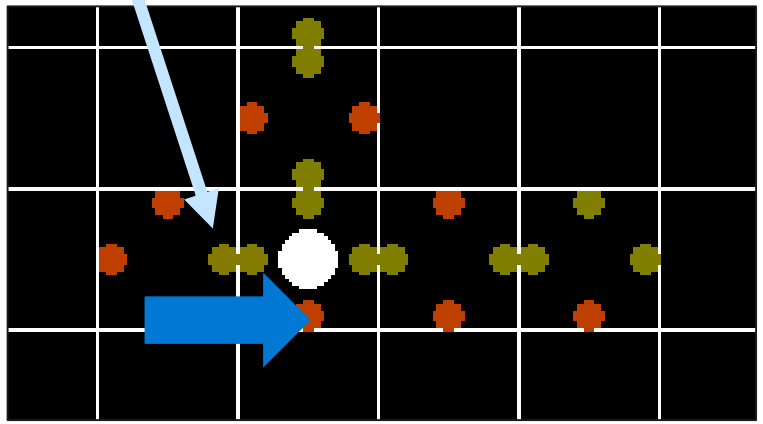
Back to the Cliff Walking Example …

Rewards are propagated backwards in time



Example: agent just took a step to the right

Update action value with observed reward (e.g., r = -0.1) and the current Q value estimate of the state we ended up in



After 10 minutes of training using Q-Learning:

Q-Learning with Function Approximation

To generalize over states and actions, parameterize Q with a function approximator, e.g., a deep neural net

The TD error serves as loss:

$$\mathrm{J}(\theta) = \left\| r_t + \gamma \max_{a \in A} Q\left(s_{t+1}, a; \theta'\right) - Q\left(s_t, a_t; \theta\right) \right\|^2$$

Q-Learning with Function Approximation

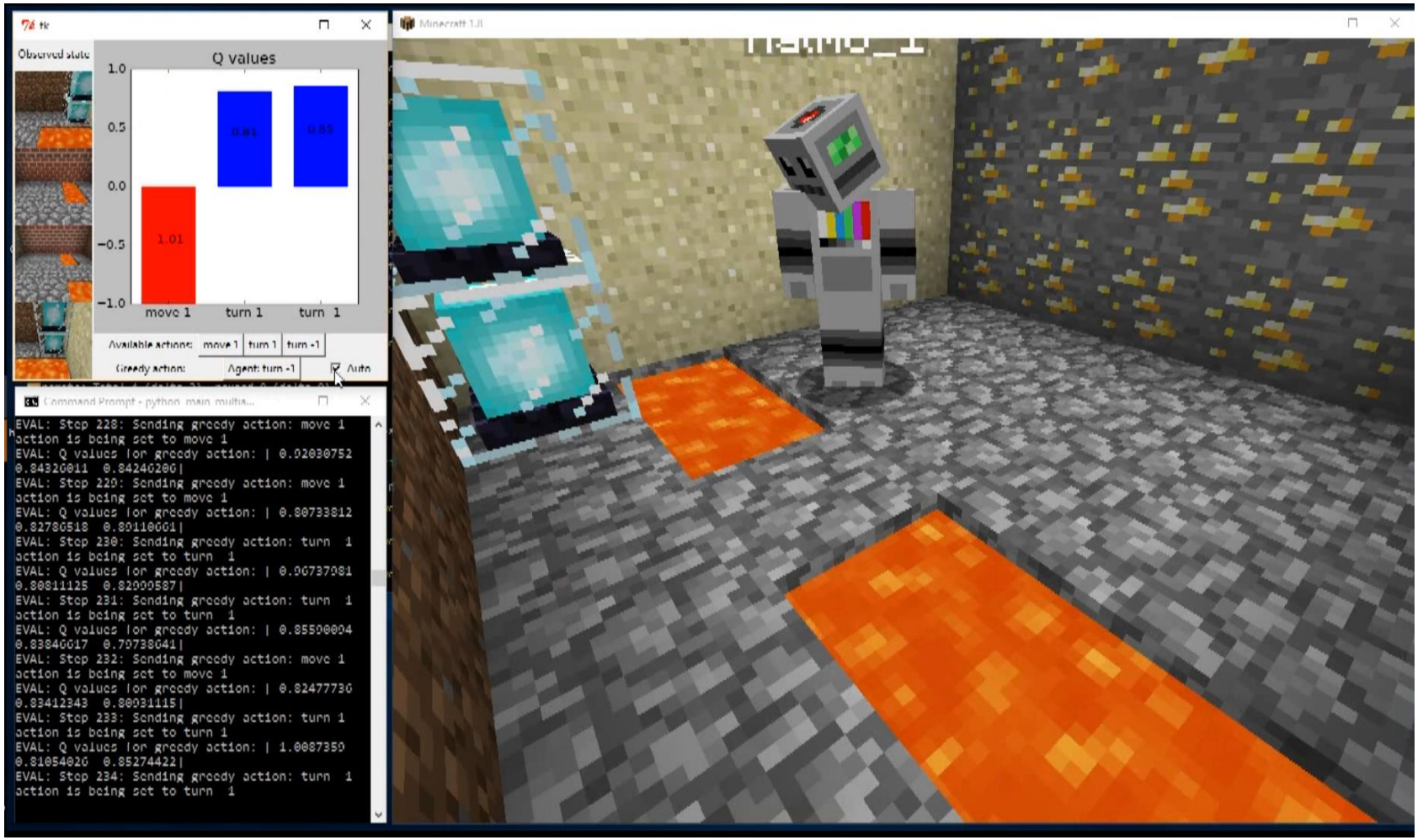
To generalize over states and actions, parameterize Q with a function approximator, e.g., a deep neural net

The TD error serves as loss:

$$\mathrm{J}(\theta) = \left\| r_t + \gamma \max_{a \in A} Q\left(s_{t+1}, a; \theta'\right) - Q\left(s_t, a_t; \theta\right) \right\|^2$$

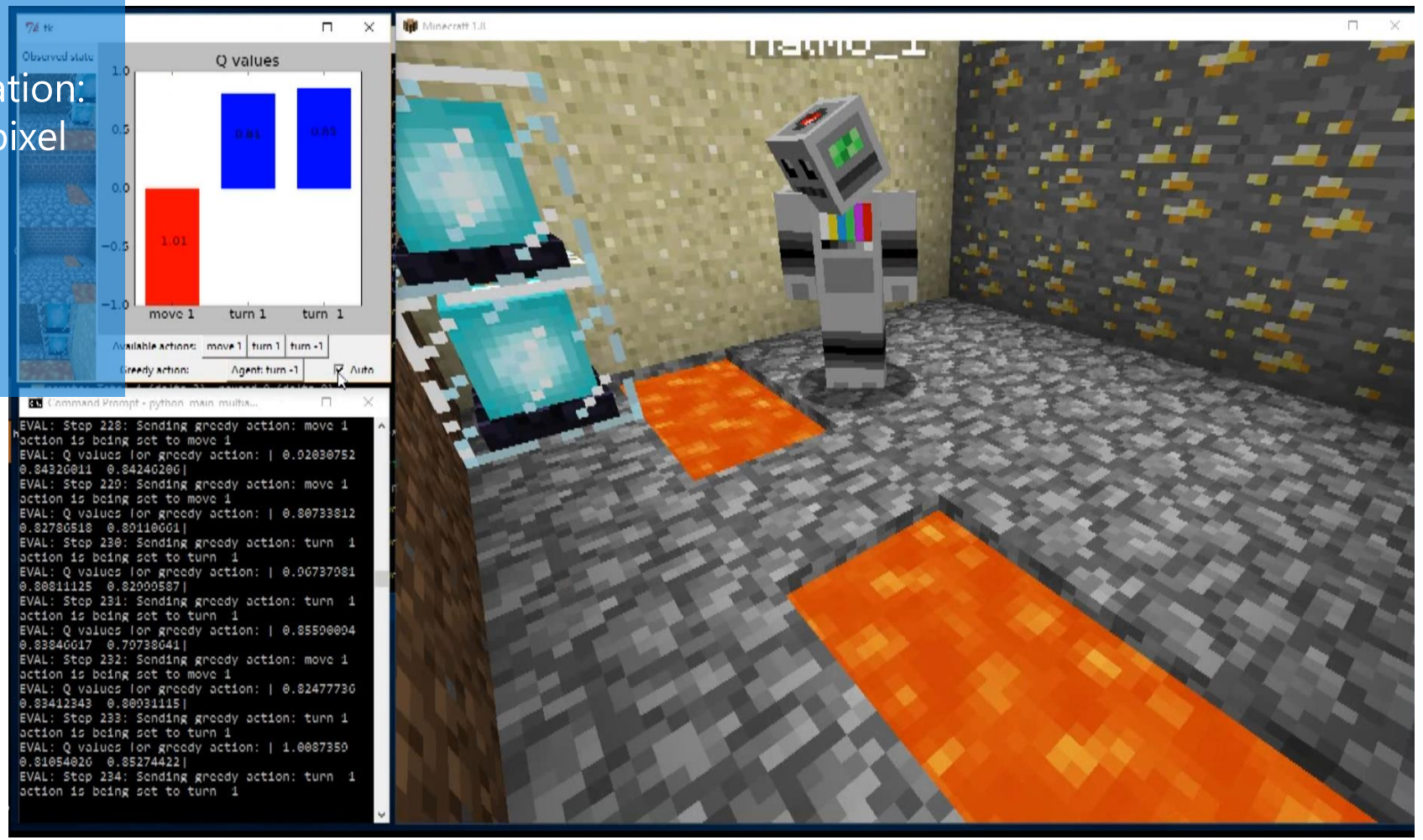
And is optimized using gradient descent

Case Study: Learning to navigate Minecraft from pixels using DQN

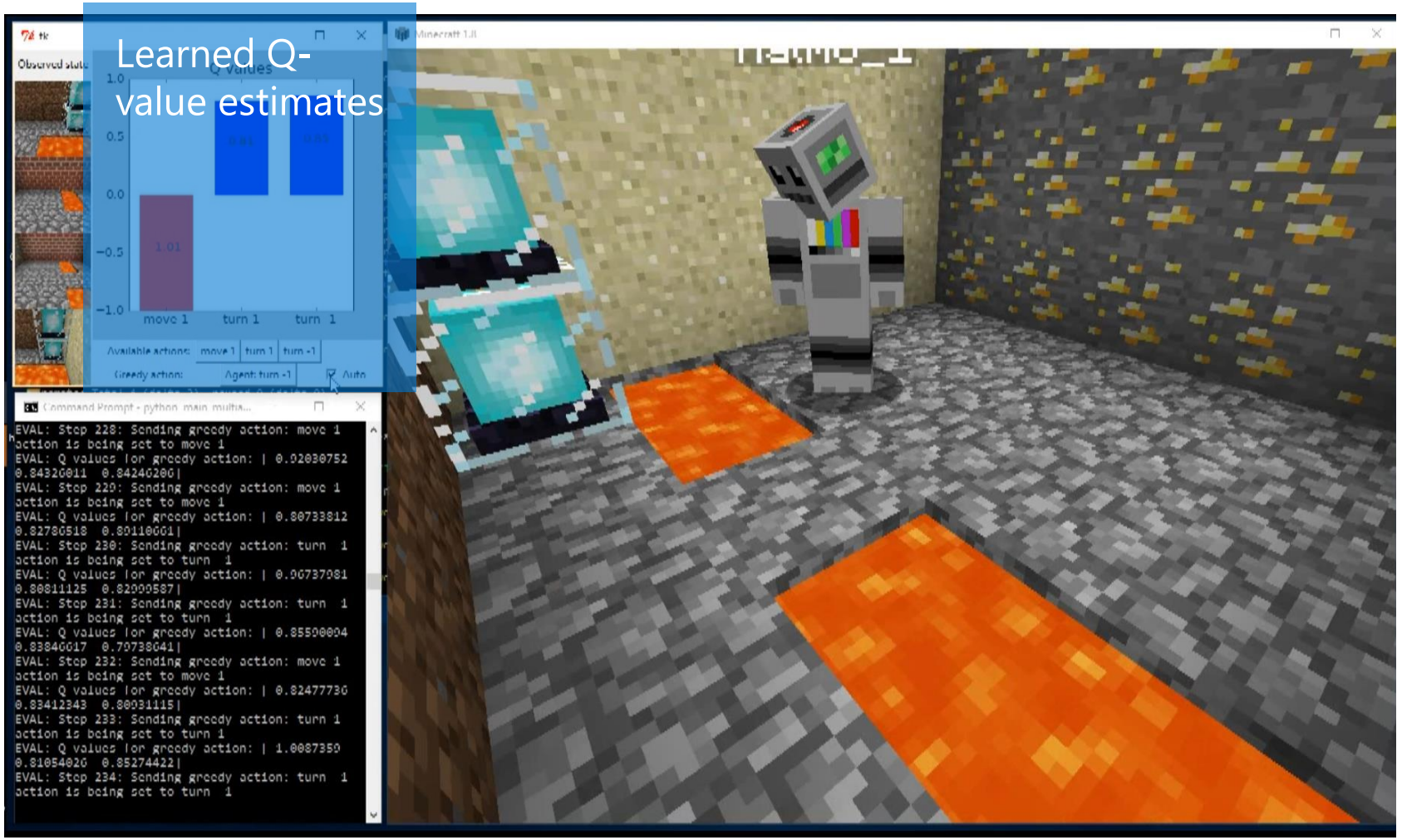


Case Study: Learning to navigate Minecraft from pixels using DQN

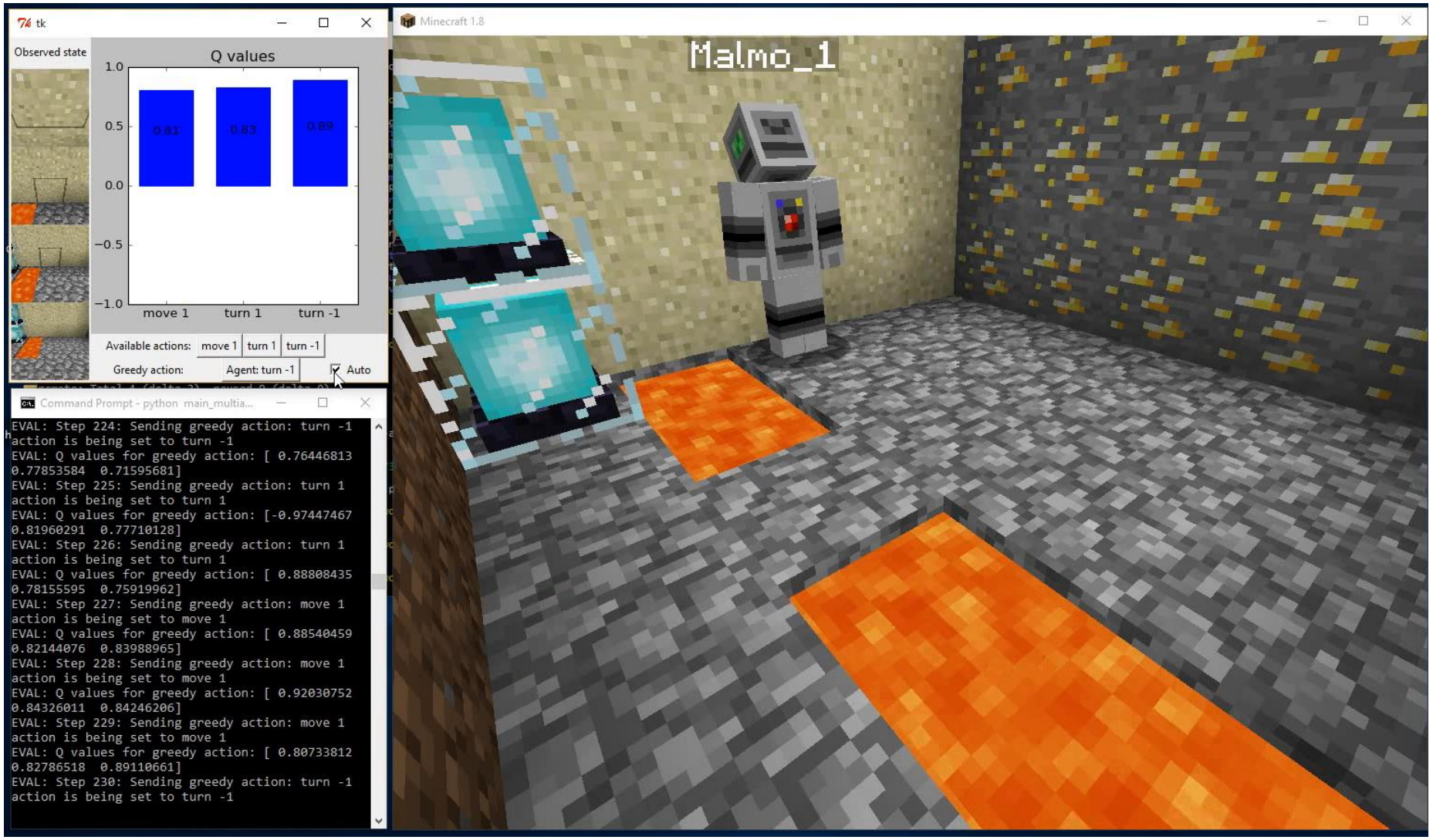
State observation: 4 past pixel images



Case Study: Learning to navigate Minecraft from pixels using DQN

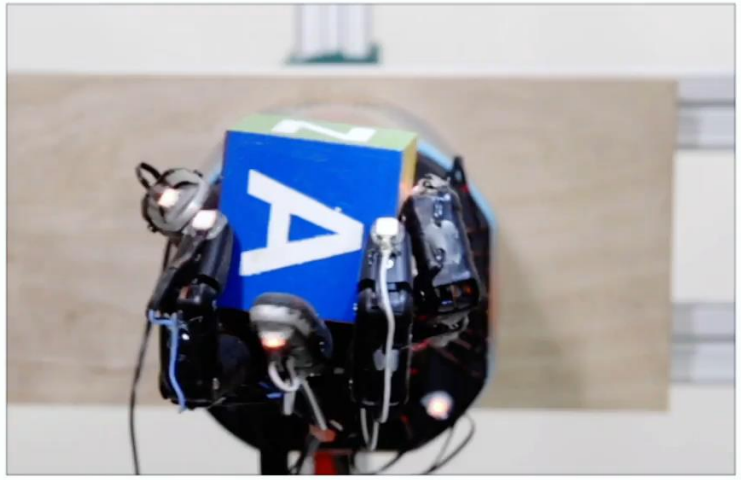


Case Study: Learning to navigate Minecraft from pixels using DQN

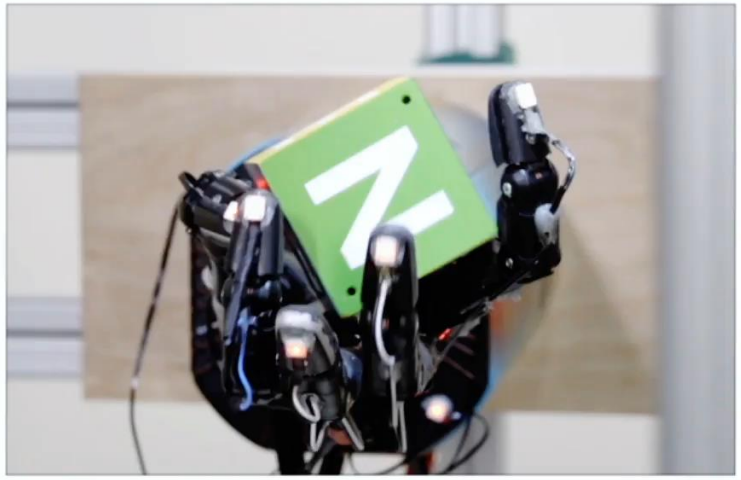


Case Study: **Learning Dexterity**

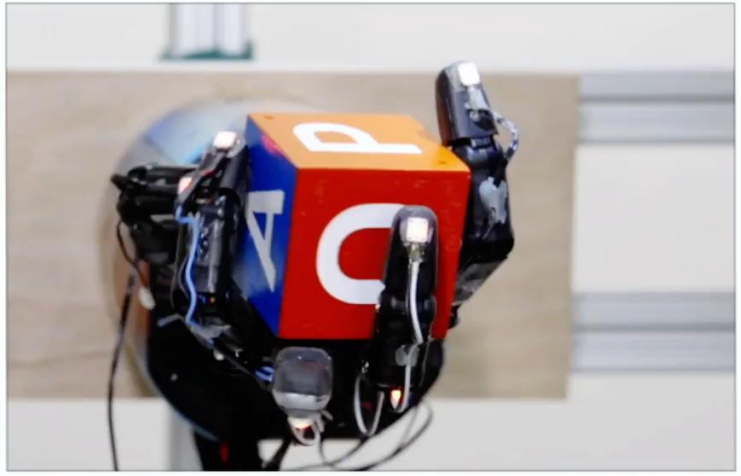
S. Sidor, J. Pachocki, M. Plappert, J. Tobin, P. Welinder, J. Schneider, A. Petron, M. Andrychowicz, A. Ray, L. Weng, R. Józefowicz, B. Baker – OpenAI – https://openai.com/blog/learning-dexterity/



FINGER PIVOTING



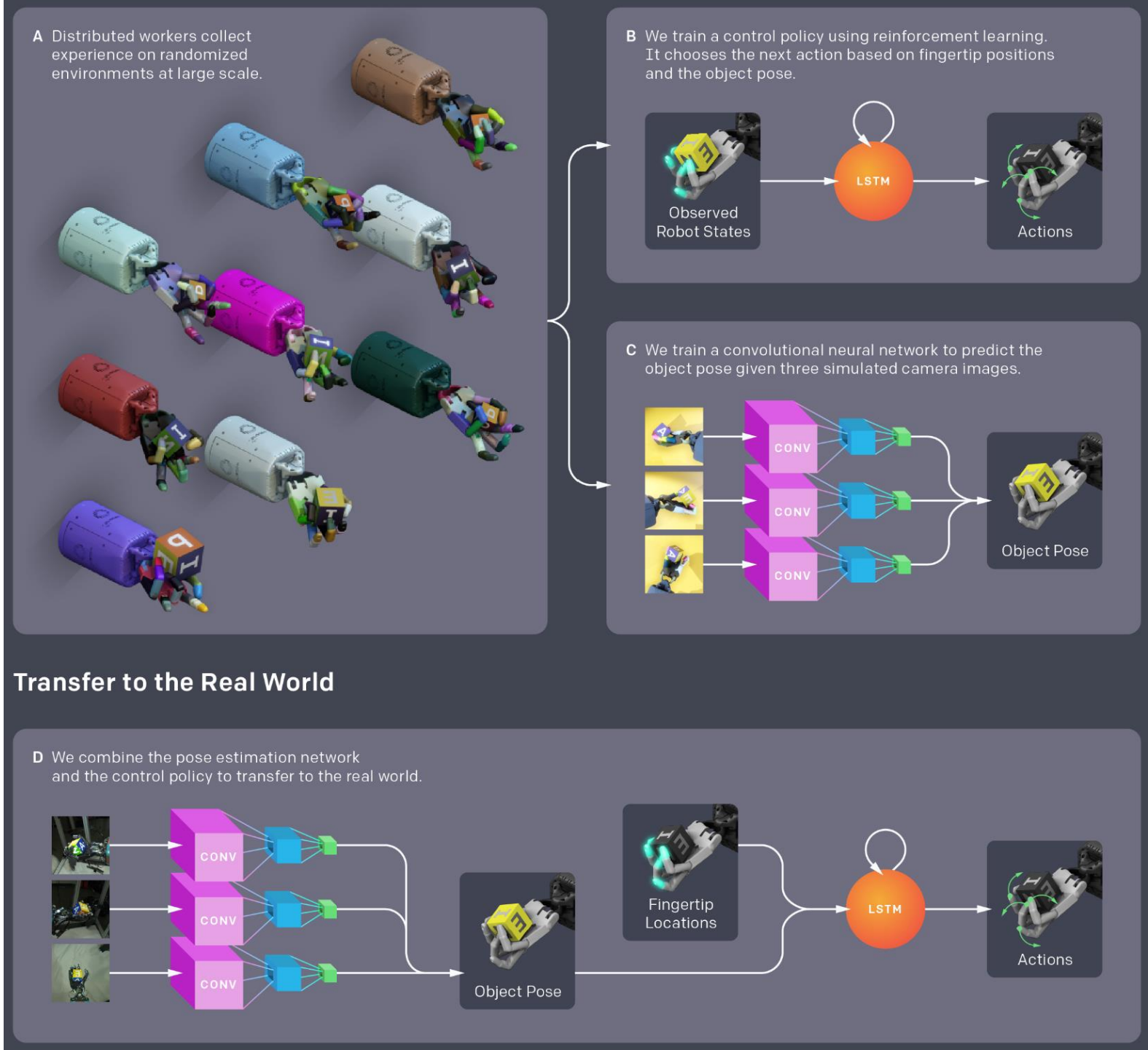
SLIDING



FINGER GAITING

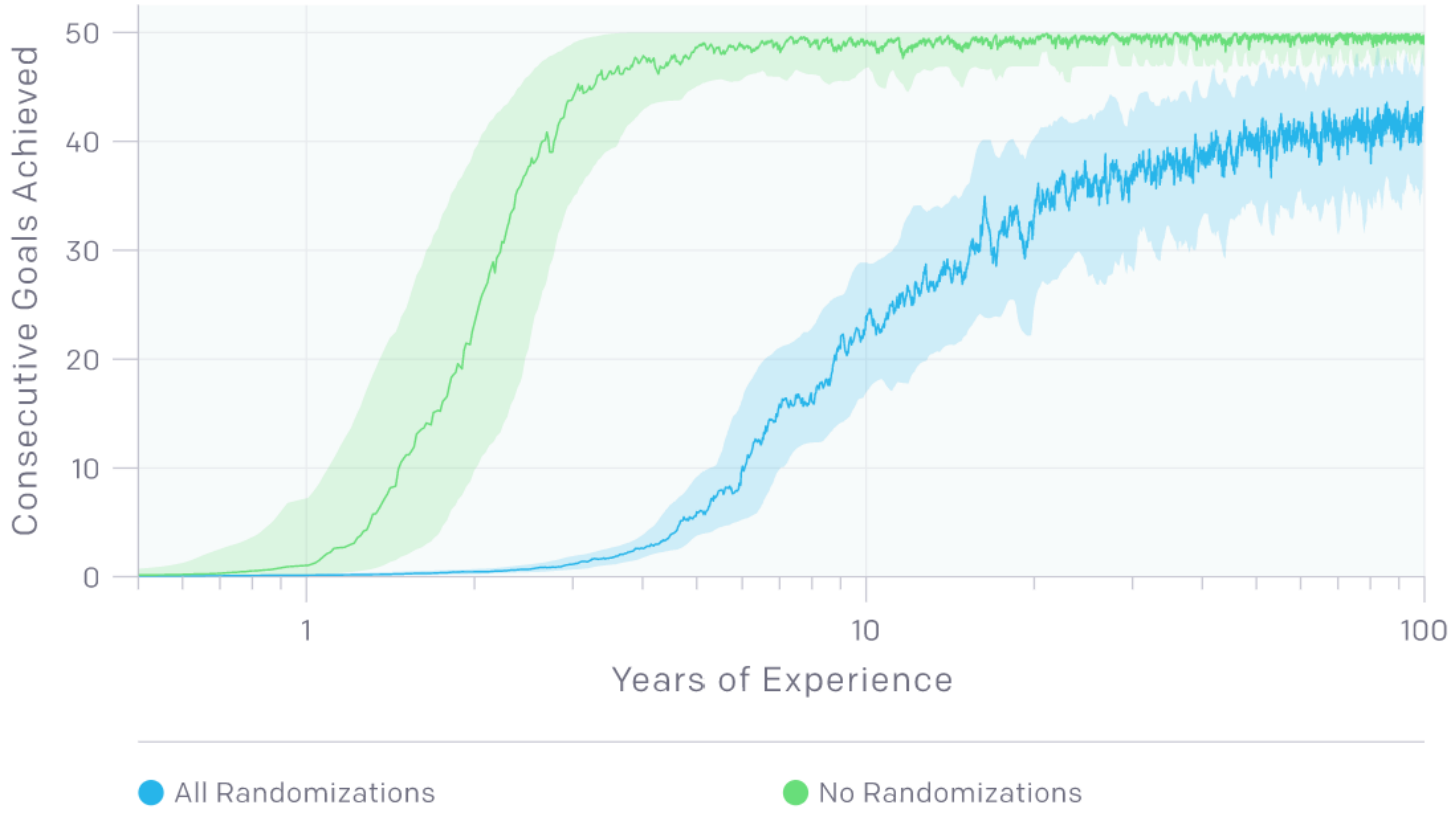
Case Study: **Learning Dexterity**

S. Sidor, J. Pachocki, M. Plappert, J. Tobin, P. Welinder, J. Schneider, A. Petron, M. Andrychowicz, A. Ray, L. Weng, R. Józefowicz, B. Baker – OpenAI – https://openai.com/blog/learning-dexterity/



Case Study: **Learning Dexterity**

S. Sidor, J. Pachocki, M. Plappert, J. Tobin, P. Welinder, J. Schneider, A. Petron, M. Andrychowicz, A. Ray, L. Weng, R. Józefowicz, B. Baker – OpenAI – https://openai.com/blog/learning-dexterity/



Learning progress with and without randomizations over years of simulated experience.

Further Reading

Sutton, R. S., & Barto, A. G. (2017). *Reinforcement learning: An introduction*. MIT press, 2nd Edition. http://incompleteideas.net/book/the-book-2nd.html Chapter 6, 9-11

Abbeel, P., Coates, A., Quigley, M., & Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. *NIPS 2007*. Project homepage: http://heli.stanford.edu/

Edwards, A. L., Dawson, M. R., Hebert, J. S., Sherstan, C., Sutton, R. S., Chan, K. M., & Pilarski, P. M. (2016). Application of real-time machine learning to myoelectric prosthesis control: A case series in adaptive switching. *Prosthetics and orthotics international*, *40*(5).

Advanced Topics: Multi-Task RL

Multi-Task Learning is easy for People

Example:

"According to the Driver and Vehicle Standards Agency (DVSA), it takes most people **45 hours of lessons** to learn how to drive, **plus 22 hours of practising**."

directline.com/car-cover/how-long-does-it-take-to-learn-to-drive

Question: "how long does it take you to get used to a new car?"

"When I first learned (6 years ago) and was just used to driving the one car, took me about **48 hrs** to be comfortable"

"**Minutes**. But i drive a lot of cars and vans."

forums.moneysavingexpert.com/showthread.php?t=5079676

How to achieve efficient multi-task RL in artificial agents?

Problem formulation: Meta-Learning

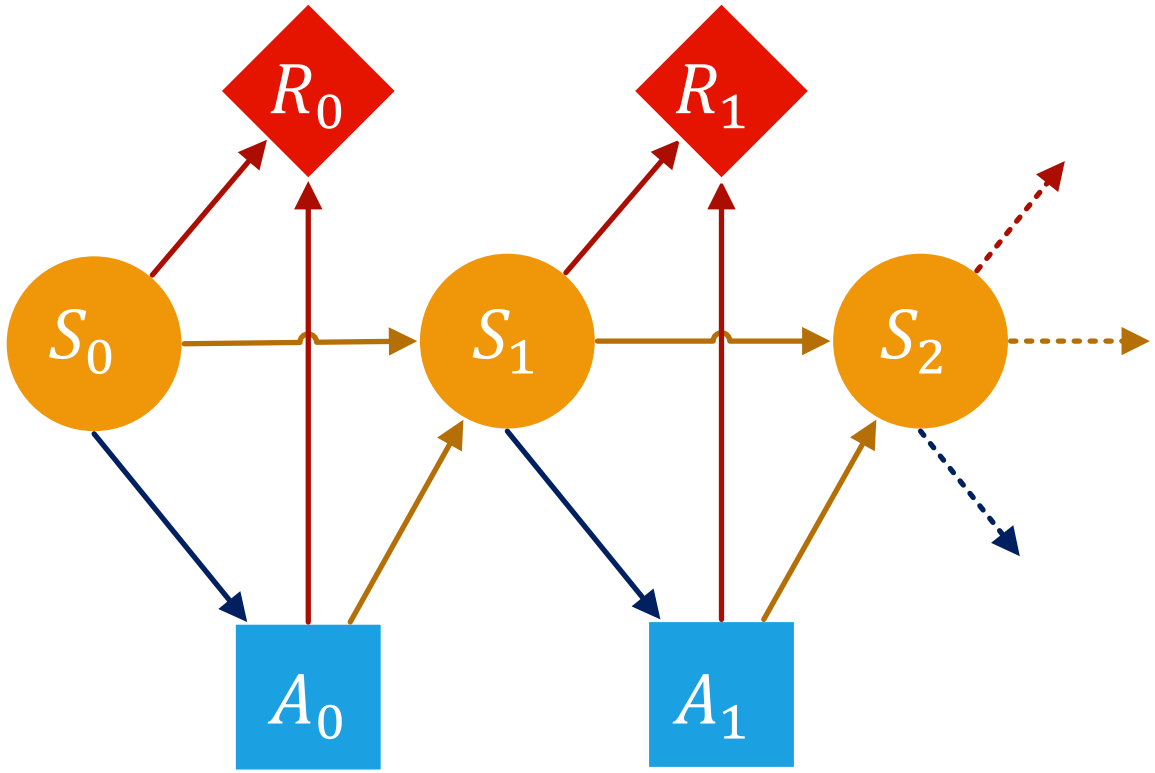
Given: distribution over training and test tasks
$p_{train}(\mathrm{T})$, $p_{test}(\mathrm{T})$

During meta-training:

- Sample tasks $\mathrm{T}_i \sim p_{train}$
- Observe training data for training tasks: $\mathrm{D}_i^{train}, \mathrm{D}_i^{test}$
- Learn to adapt to new tasks

At test time: observe test data $\mathrm{D}_i^{test} \sim p_{test}$
Goal: high performance on test task

Here: MDPs share low dimensional task embedding



States S

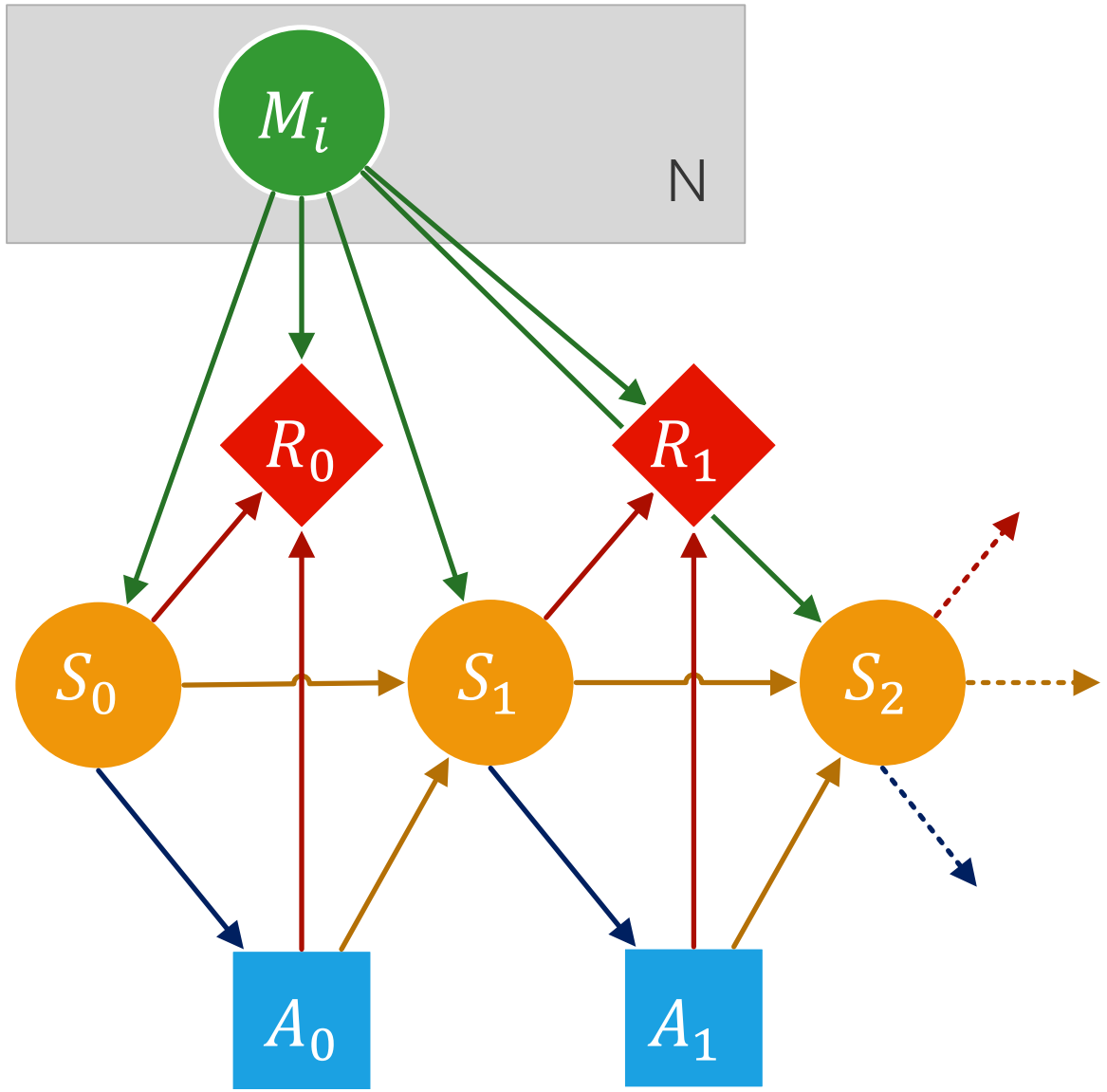
Actions A

Transition function $T(s_{t+1}|s_t, a_t)$

Reward function $R(s, a)$

Policy $\pi(s|a)$

Here: MDPs share low dimensional task embedding



Task embedding: m_i

States S

Actions A

Transition function $T(s_{t+1}|s_t, a_t; m_i)$

Reward function $R(s, a; m_i)$

Policy $\pi(s|a)$

Meta Reinforcement Learning with Latent Variable Gaussian Processes

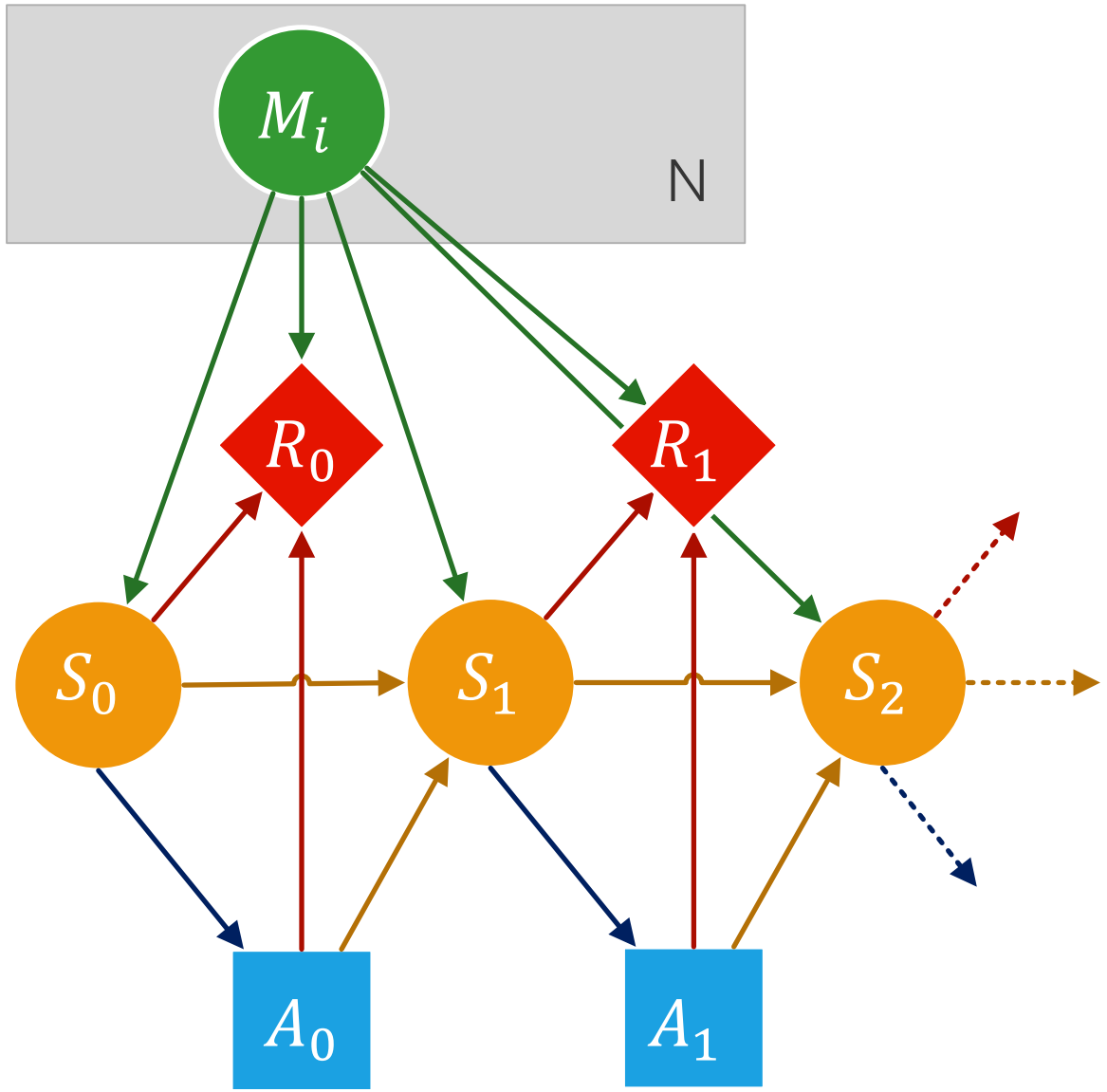
UAI 2018 – Arxiv: 1803.07551

Steindór Sæmundsson
Imperial College London

Katja Hofmann
Microsoft Research

Marc Deisenroth
Imperial College London

Model-based Control with Latent Task Embedding



Task embedding:

$$m_i \sim \mathcal{N}(\mu^i, \Sigma^i)$$

Learn dynamics model:

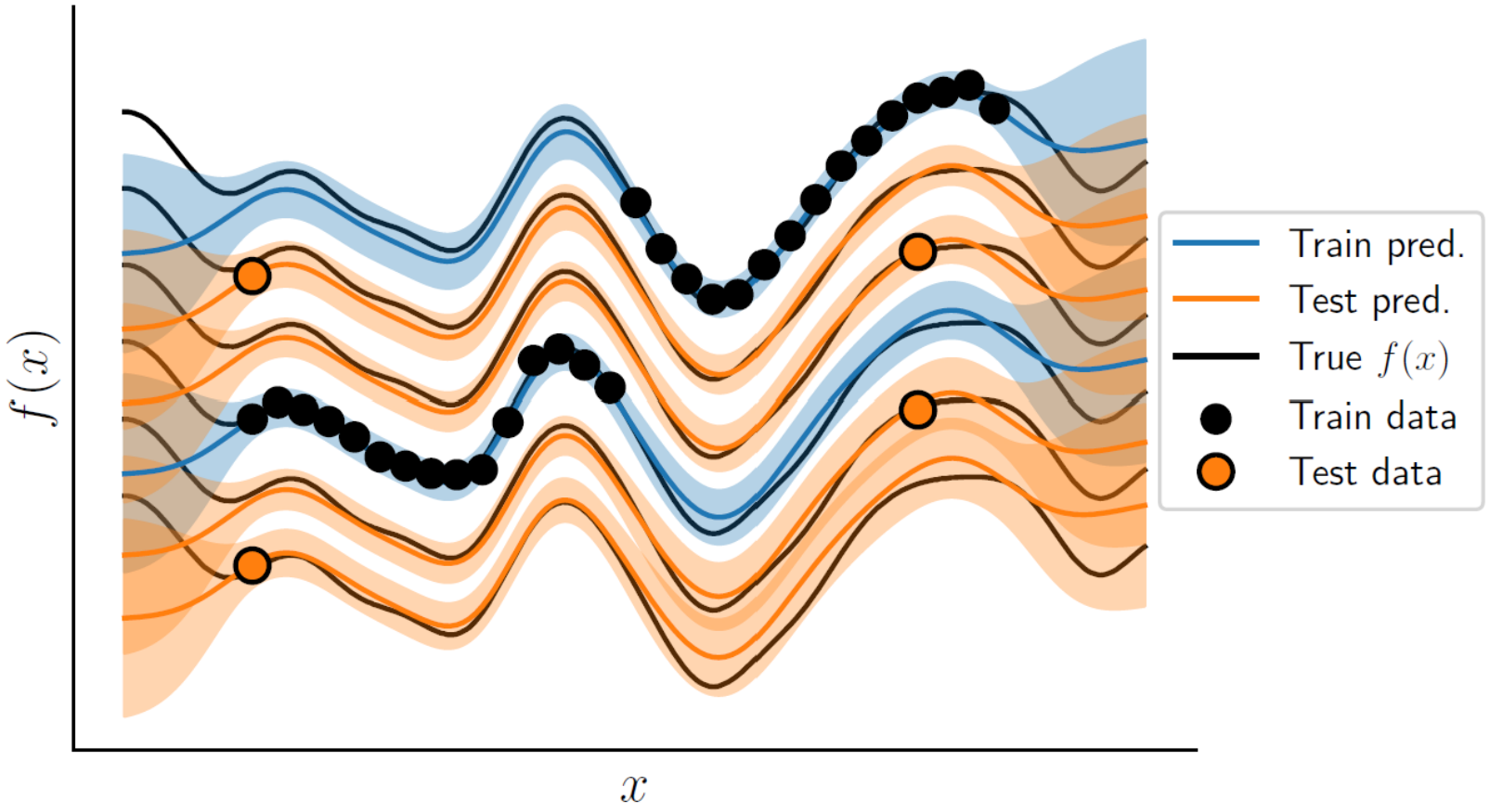
$$s^i_{t+1} = f(s^i_t, a^i_t; m^i) + \epsilon$$

with Gaussian Process prior on f

Training: jointly optimize parameters of f and m_i using stochastic variational inference

Inference: update posterior over m_i (infer task)

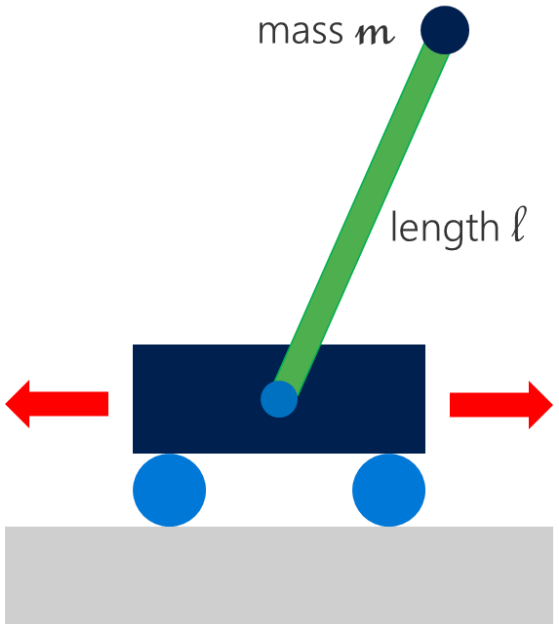
Toy experiment: multi-task prediction



Six toy tasks with a shared structure (the same function) and task specific variation (fixed offset).

- ✓ Automatically disentangles shared and task specific structure, given training data (black discs)
- ✓ Maintains sensible uncertainty estimates
- ✓ Generalizes to test task tasks given limited test data (orange discs)

Experiments: Multi-task Cart-Pole



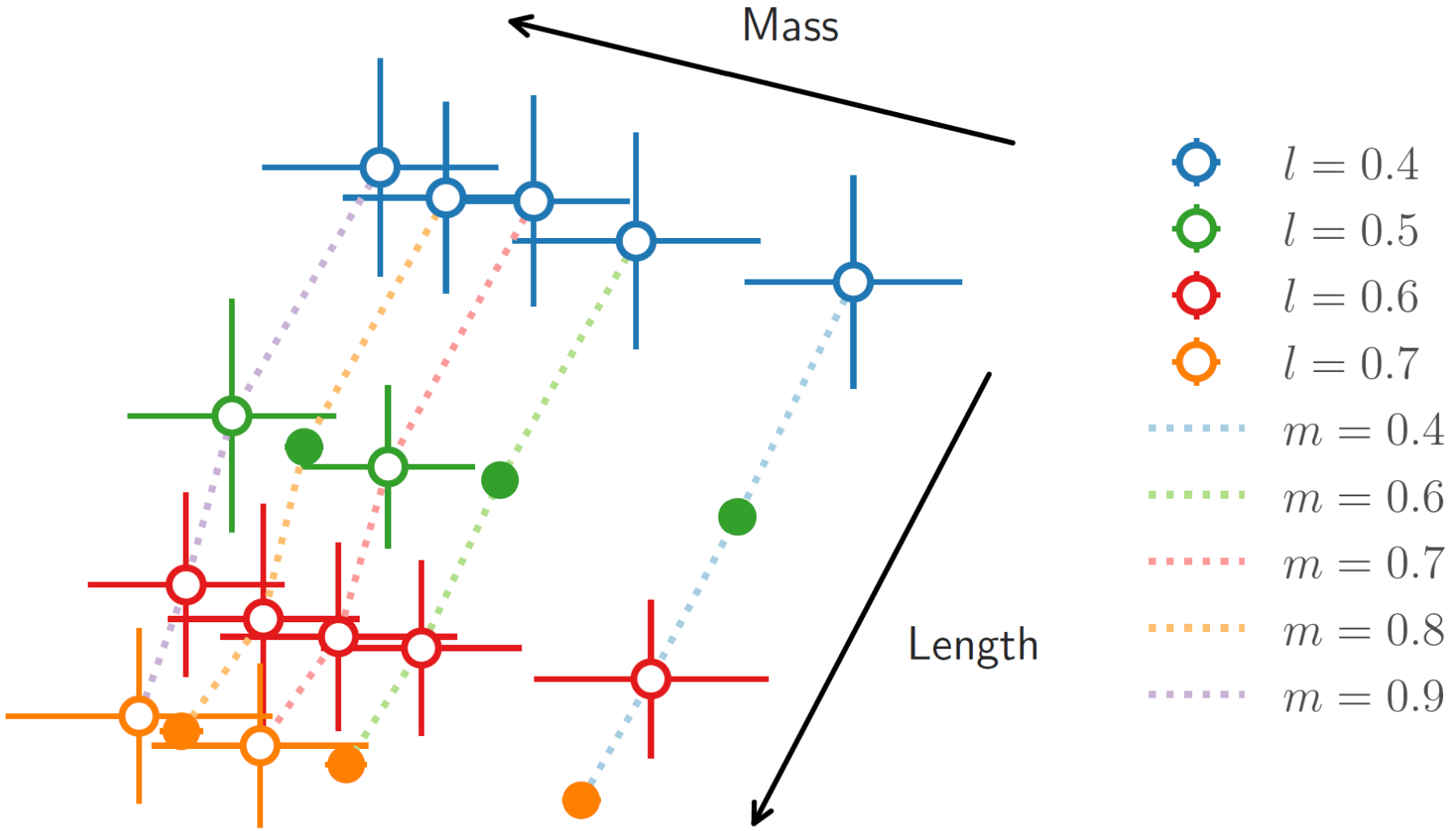
Systems vary in mass m and pendulum length l

6 training tasks: $l \in [.5, .7]$ x $m \in [.4, .6, .8]$

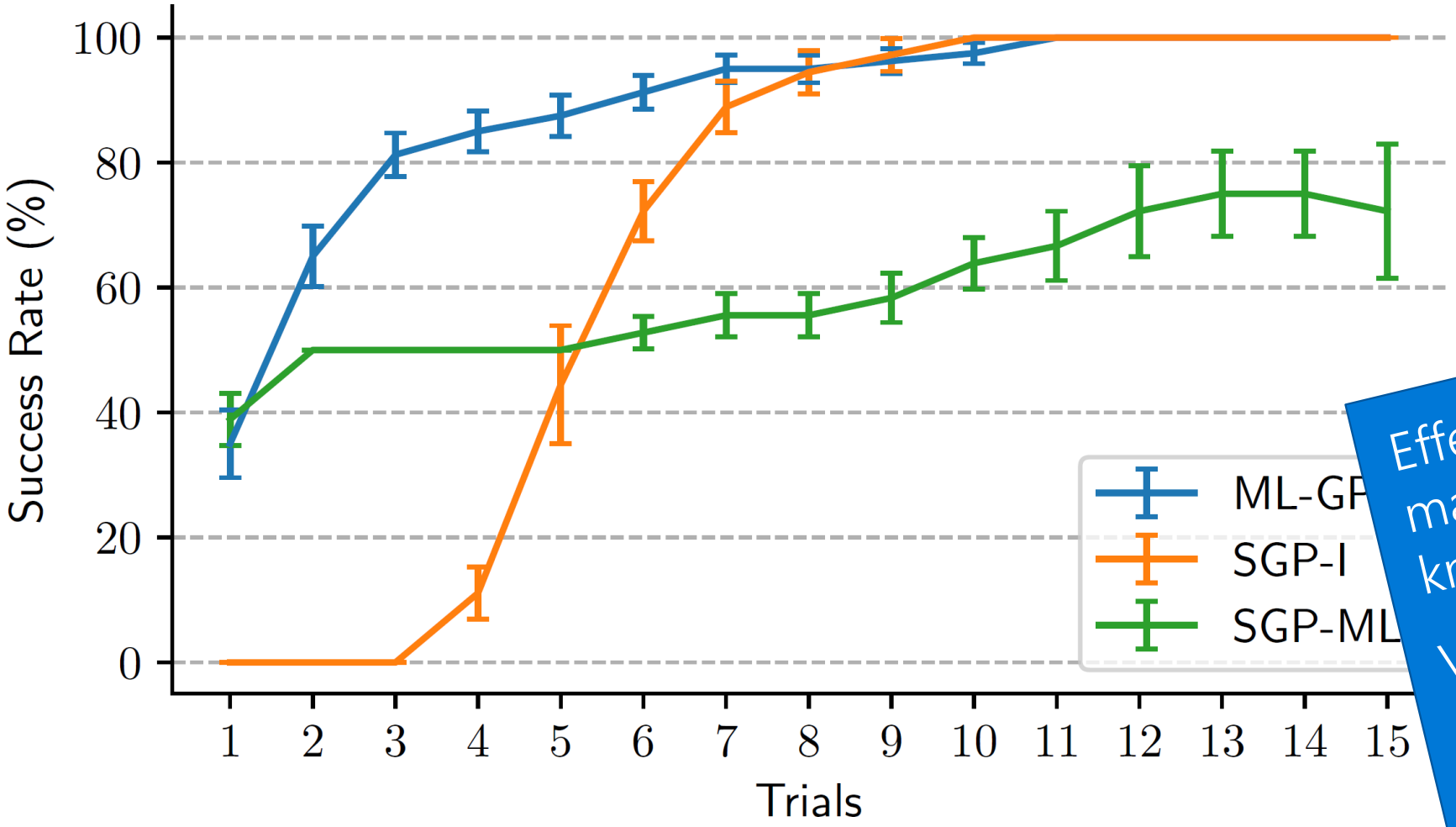
14 held out test tasks require interpolation + extrapolation

$l = 0.4$
$l = 0.5$
$l = 0.6$
$l = 0.7$
$m = 0.4$
$m = 0.6$
$m = 0.7$
$m = 0.8$
$m = 0.9$

Multi-task Cart-Pole: Learned Embedding



Multi-Task CartPole: Control Success Rate



Mean success rate of ML-GP (ours) compared to baselines:

SGP-I (

ual

lti-

Effectively encode, make use of prior knowledge

Very sample efficient

Good uncertainty quantification

Fast Context Adaptation via Meta-Learning (CAVIA)

ICML 2019 + Arxiv: 1810.03642

Luisa Zintgraf
University of Oxford

Kyriacos Shiarlis
Latent Logic

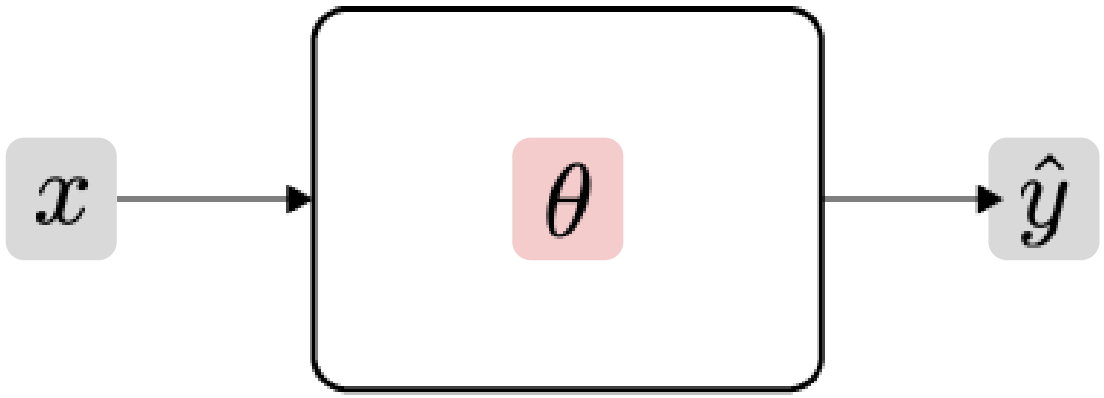
Vitaly Kurin
U of Oxford
Latent Logic

Katja Hofmann
Microsoft Research

Shimon Whiteson
U of Oxford
Latent Logic

Aim: Fast Adaptation + Flexible Approach

Starting point: MAML [Finn et al. 2017] – extremely flexible meta-learning approach using gradient descent



2-stage gradient-based approach on batches of tasks $\mathcal{T}$

1) Inner loop:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

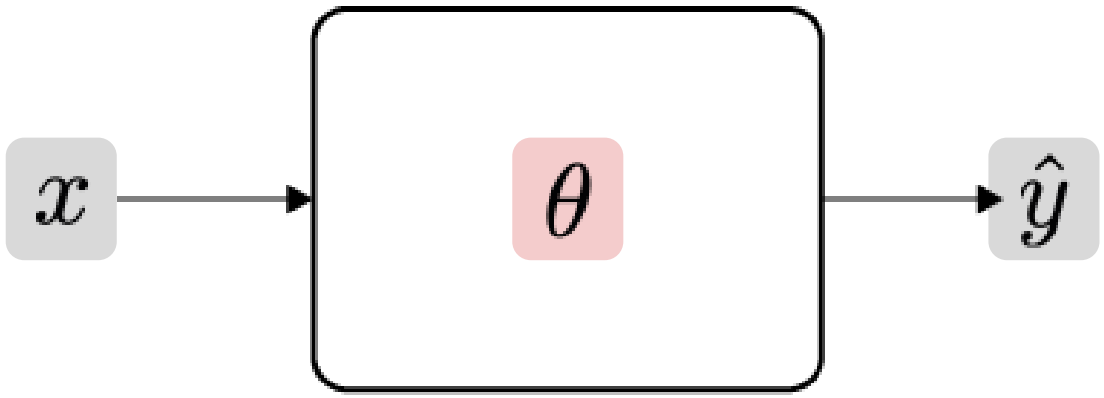
2) Outer loop:

$$\theta = \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

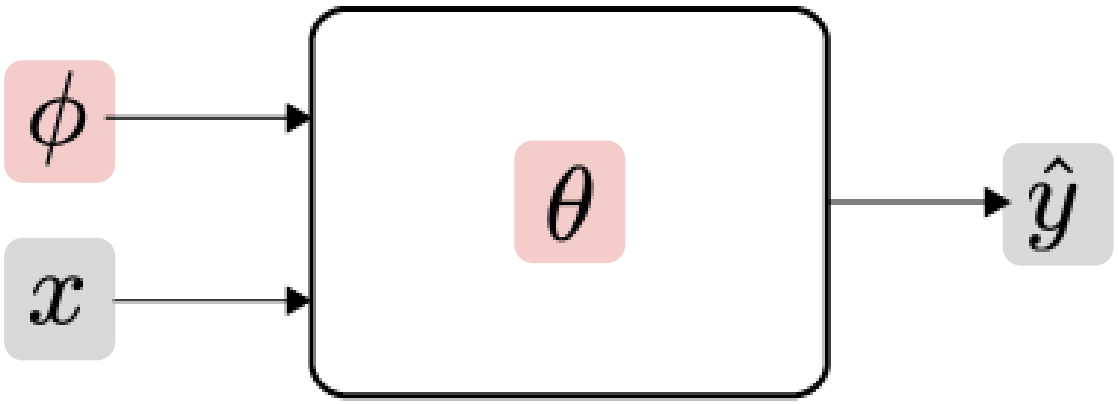
Insight: No need to update all model parameters at test time

Many tasks and current benchmarks only require task identification
Many parameters + few data points can lead to overfitting

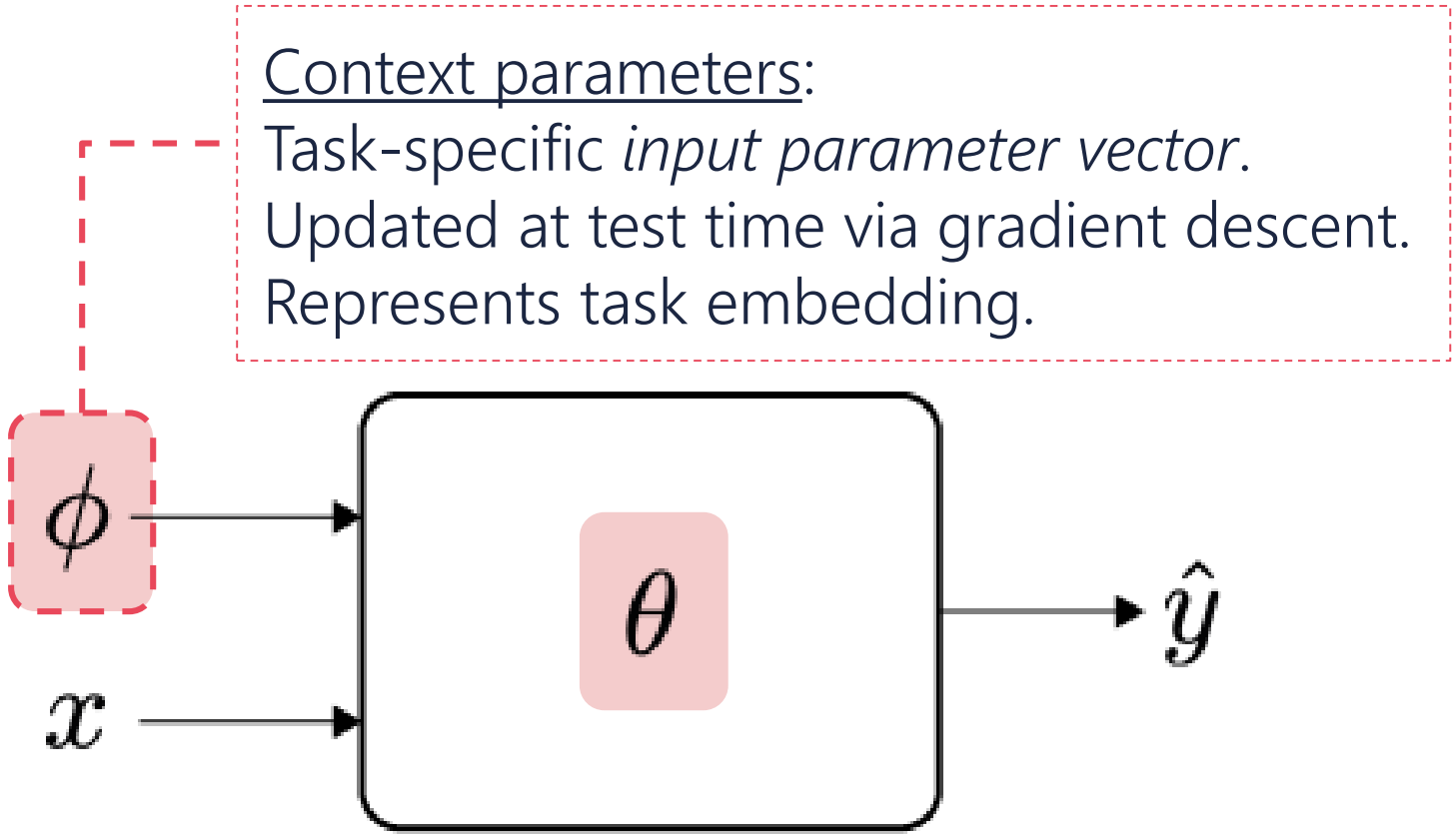
MAML (Finn et al. 2017)



CAVIA



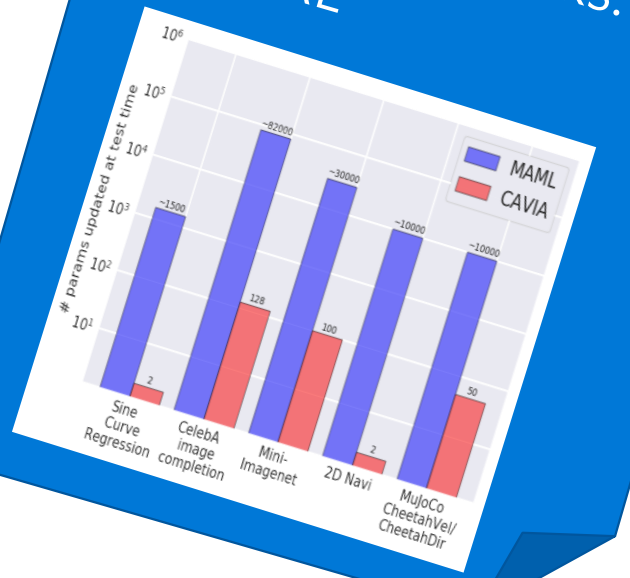
CAVIA: Fast Context Adaptation via Meta-Learning

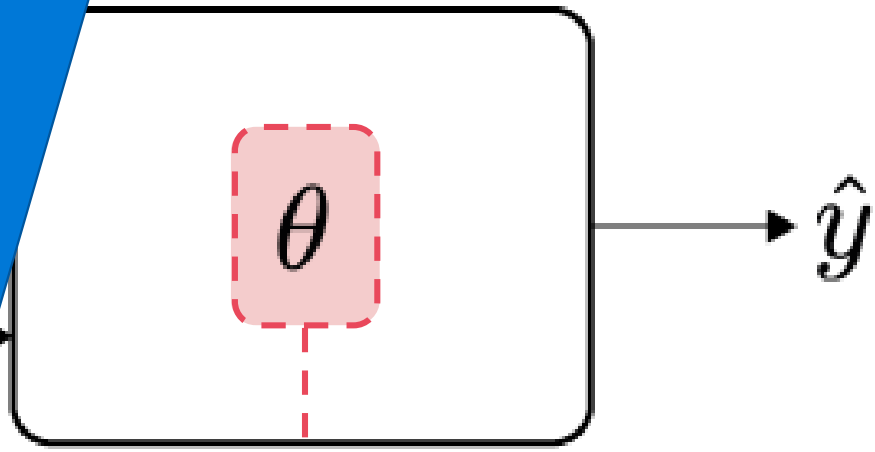


CAVIA: Fast Context Adaptation via Meta-Learning

Context parameters:
Task-specific *input parameter vector*.
Updated at test time via gradient descent.
Represents task embedding.

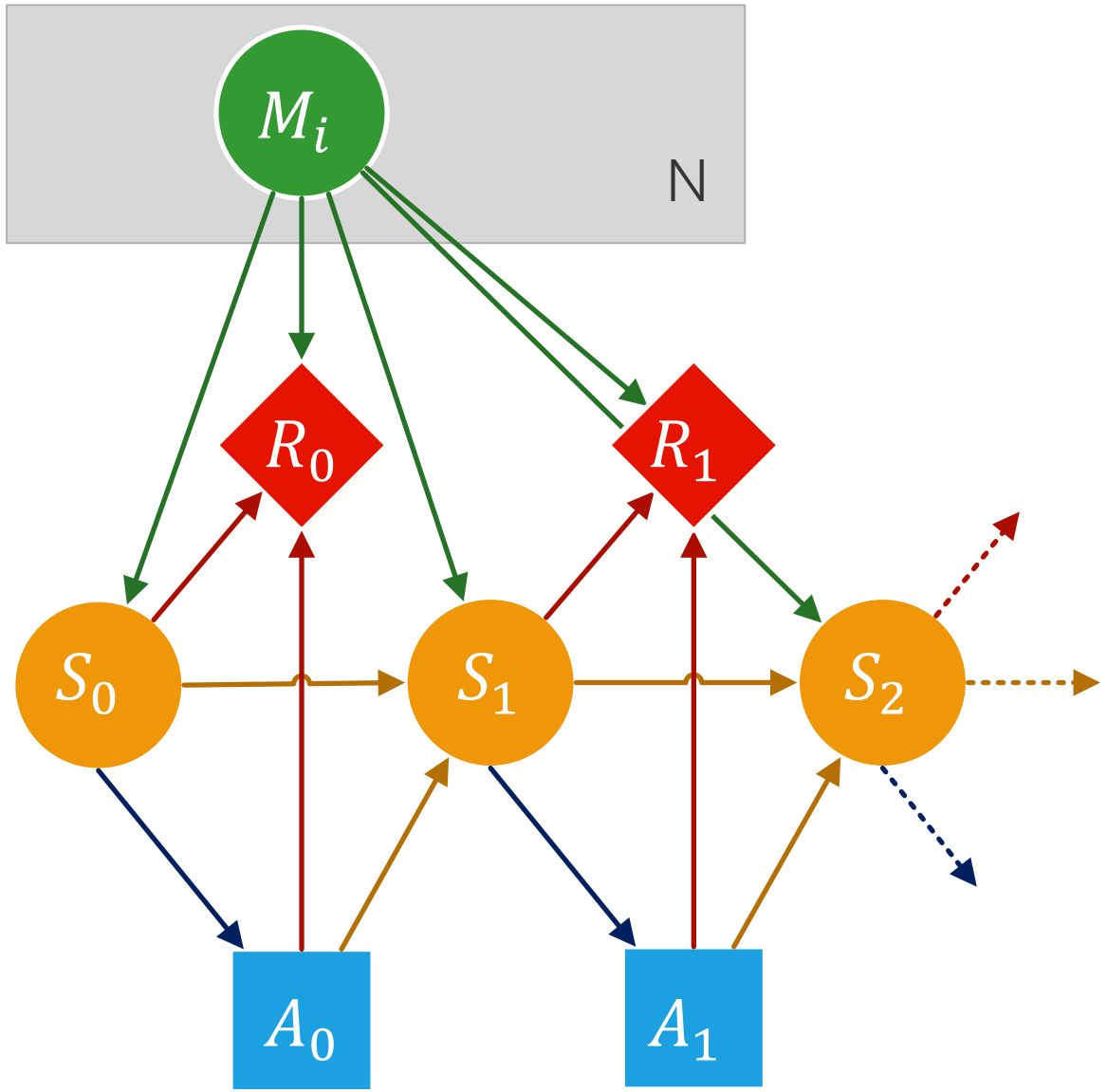
See paper for results on regression and classification tasks.
Here: RL





Network parameters:
Shared across tasks.
Meta-trained, fixed at test time.

Policy Gradient with CAVIA

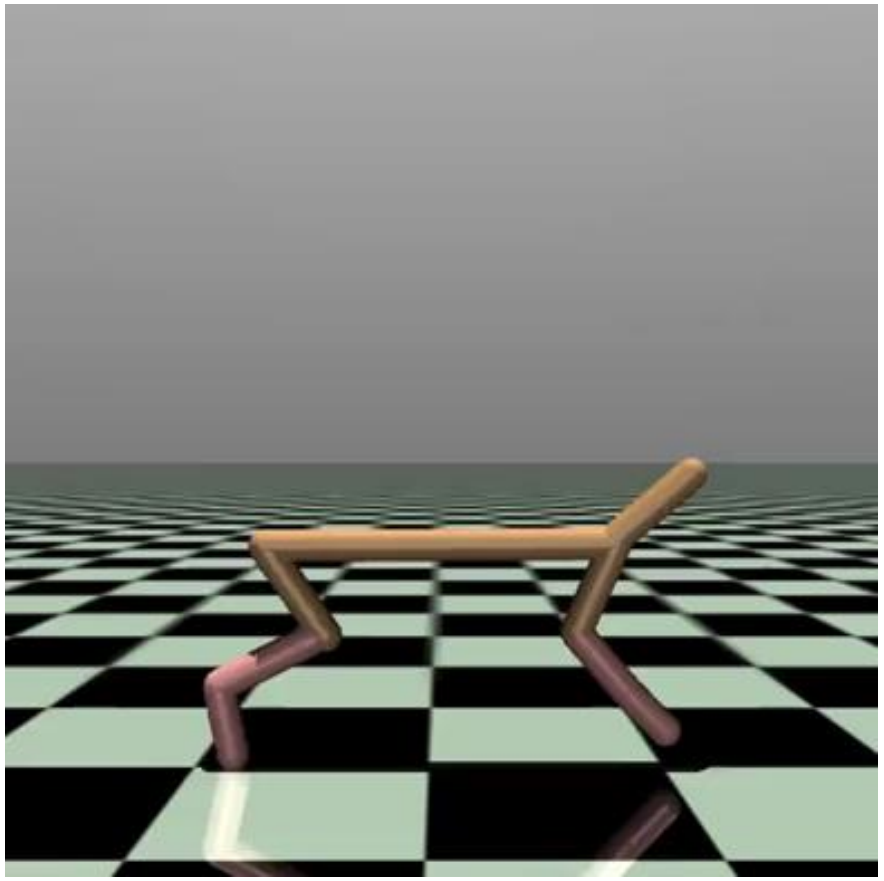


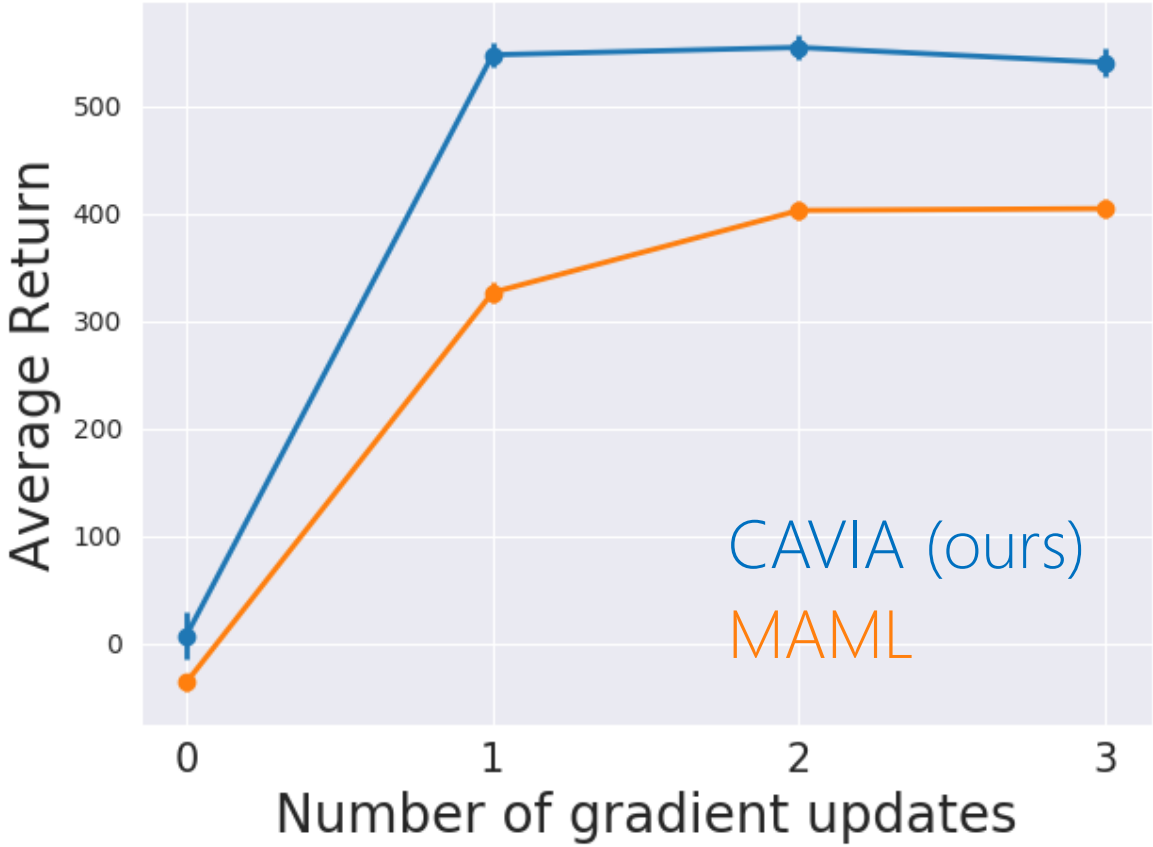
Task embedding: learned implicitly through context parameters ϕ

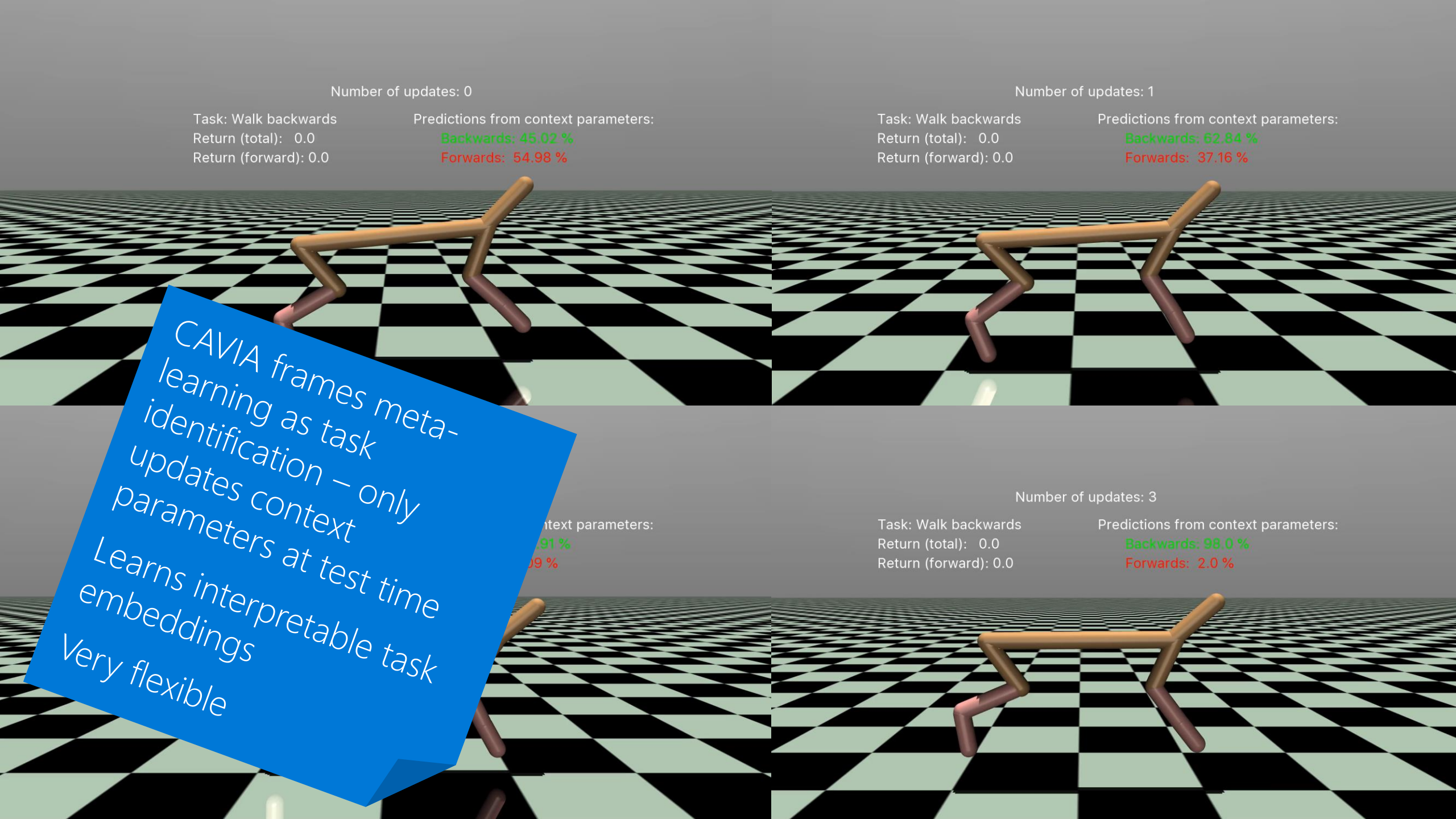
Training follows [Finn et al. 2017]: policy gradient (inner loop) and TRPO (outer loop)

RL results: Half-Cheetah – Direction

Tasks: run in a target direction






Number of updates: 0
Task: Walk backwards
Return (total): 0.0
Return (forward): 0.0
Predictions from context parameters:
Backwards: 45.02 %
Forwards: 54.98 %
Number of updates: 1
Task: Walk backwards
Return (total): 0.0
Return (forward): 0.0
Predictions from context parameters:
Backwards: 62.84 %
Forwards: 37.16 %
CAVIA frames meta-learning as task identification – only updates context parameters at test time
Learns interpretable task embeddings
Very flexible
Number of updates: 3
Task: Walk backwards
Return (total): 0.0
Return (forward): 0.0
Predictions from context parameters:
Backwards: 98.0 %
Forwards: 2.0 %

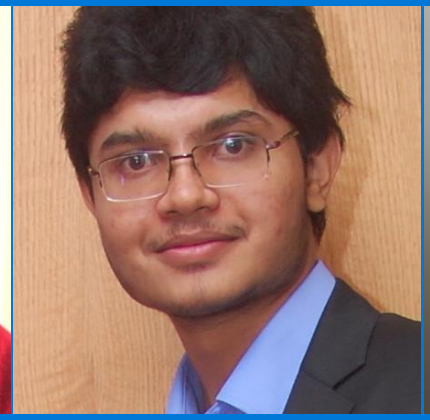
Variational Task Embeddings for Fast Adaptation in Deep RL (VATE)

SPiRL workshop at ICLR 2019 (spirl.info/2019)

Luisa Zintgraf
University of Oxford

Maximilian Igl
University of Oxford

Kyriacos Shiarlis
Latent Logic



Anuj Mahajan
University of Oxford

Katja Hofmann
Microsoft Research

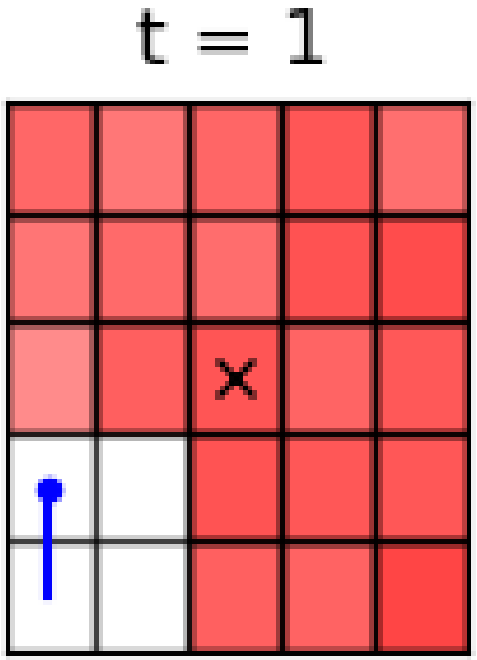
Shimon Whiteson
U of Oxford
Latent Logic

Goal: Full Online Adaptation

CAVIA is very flexible, but requires entire trajectory before adapting to a new task

Challenge: retain flexibility and full online adaptation

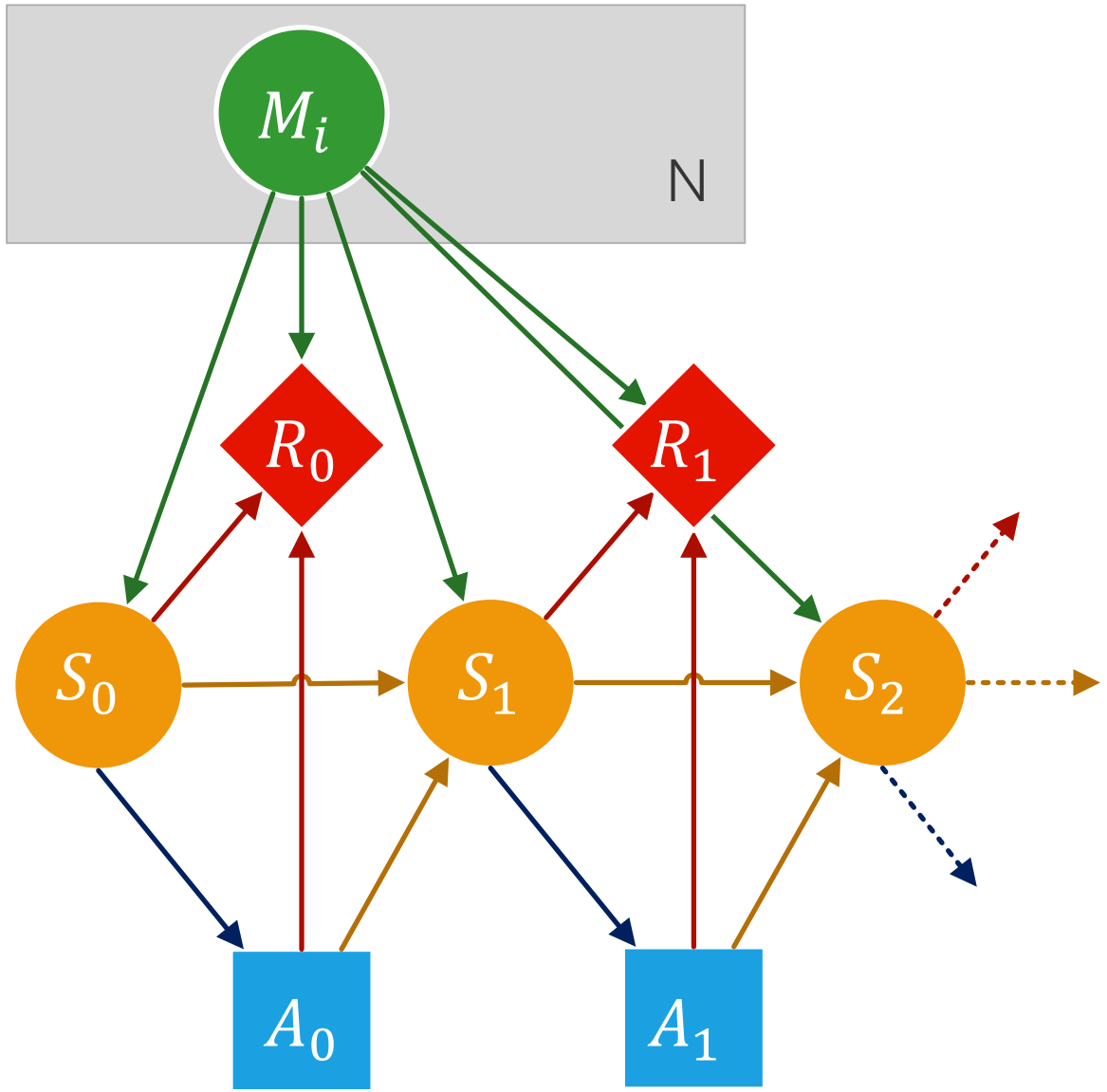
Example: Multi-Task Grid



Task: find and navigate to moving target – location changes every 3 trials

Requires structured exploration!

Instantiation: VATE



Task embedding: m_i as stochastic latent variable, inferred from trajectory $\tau = (s_0, a_0, r_0, \ldots, r_t, s_t)$

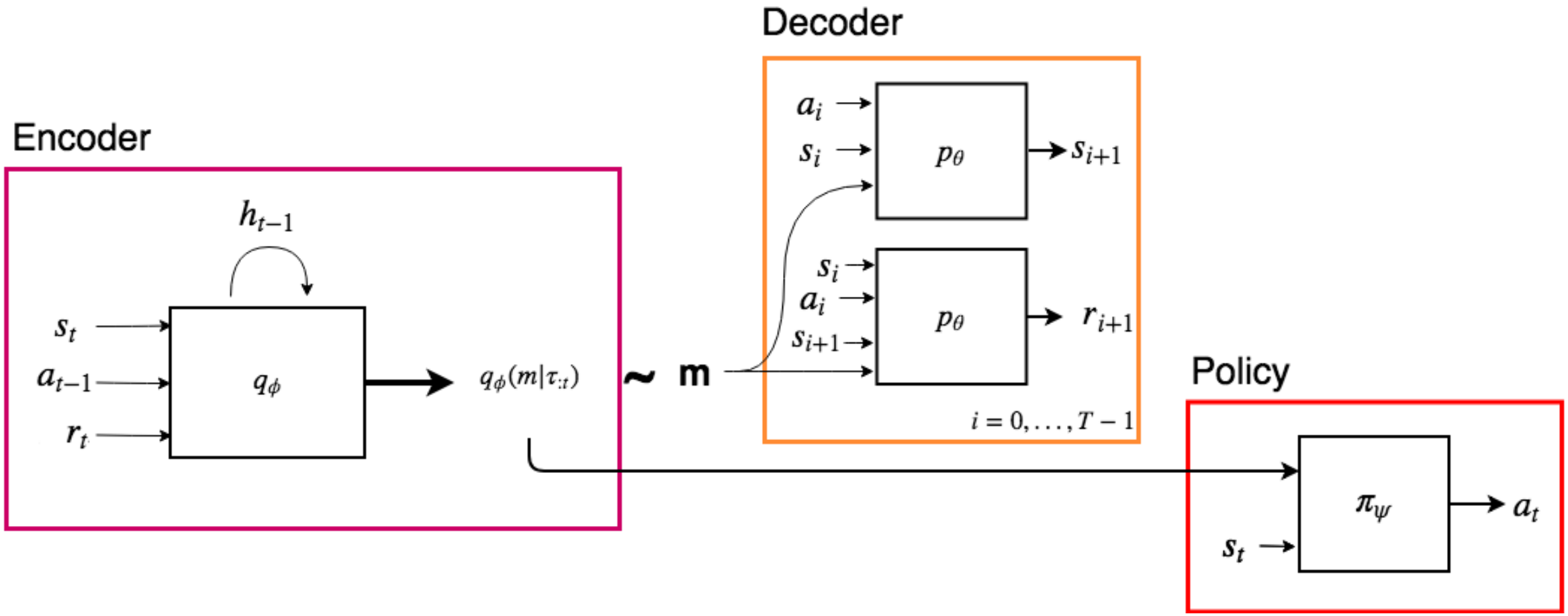
Explicitly condition on m_i:

Transition function $T(s_{t+1}|s_t, a_t; m_i)$

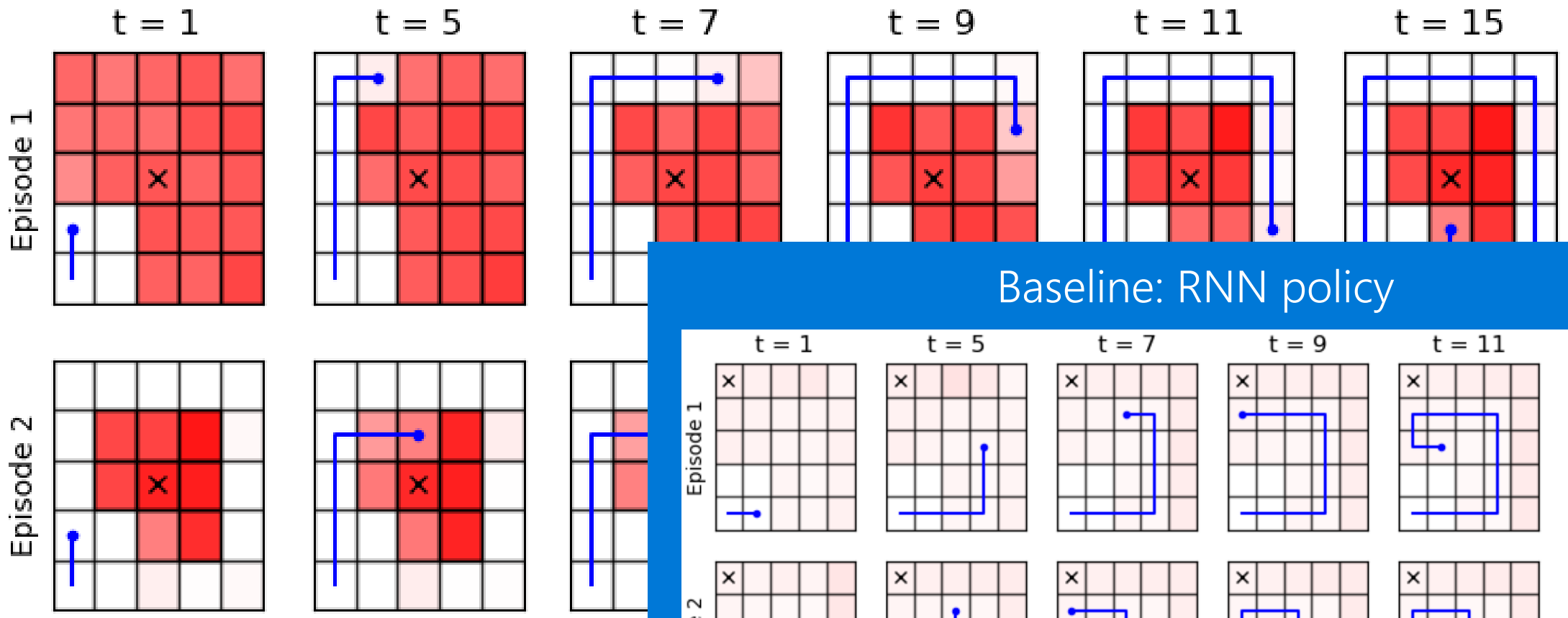
Reward function $R(r_{t+1}|s_t, a_t, s_{t+1}; m_i)$

Policy $\pi(s|a; m_i)$

Approach: VATE combines model-based with model-free elements

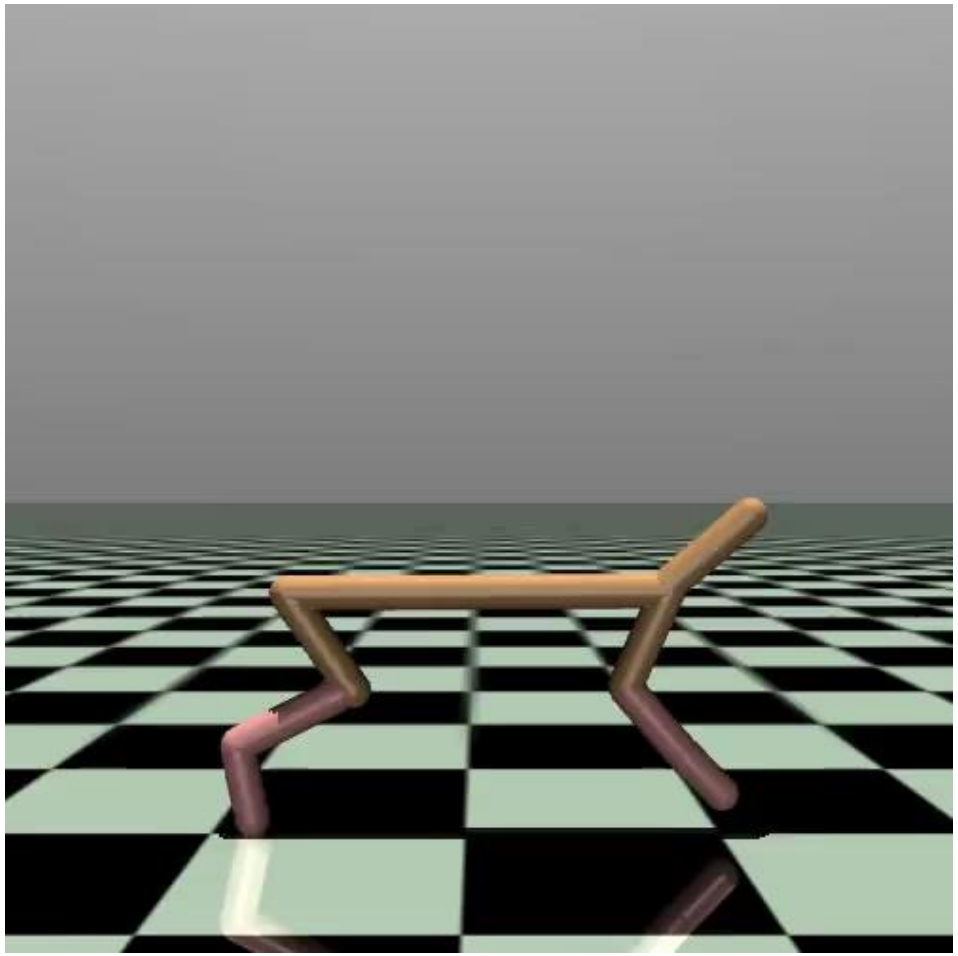


Results: Strategic Exploration in Multi-Task Grid



Baseline: RNN policy

Results: Strategic Exploration in Half-Cheetah



Episode 1: determine target direction

Episode 2: move toward

VATE learns to trade off exploration and exploitation online, while interacting with the environment.

VATE can deduce information about the task even before seeing any reward

Tutorial Introduction:
The MineRL competition

The MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors

NeurIPS 2019 Competition
Arxiv: 1904.10079

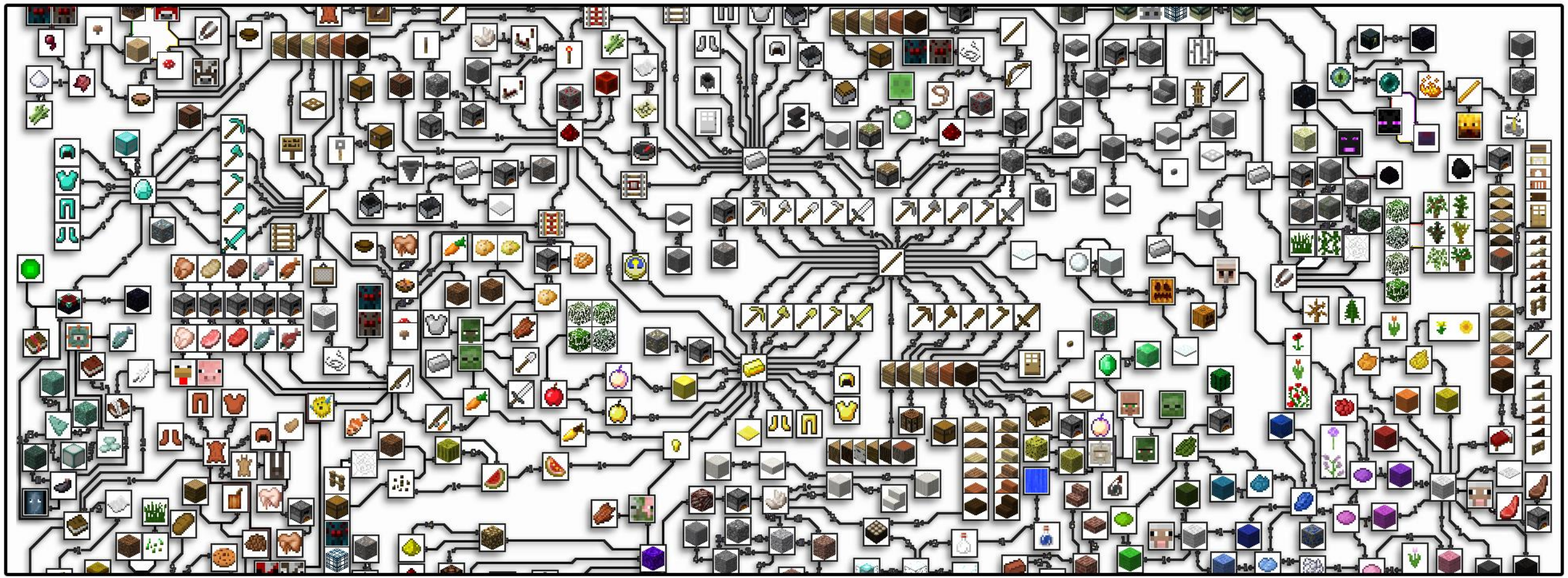
Organizing Team

William H. Guss (Carnegie Mellon University)
Mario Ynocente Castro (Preferred Networks)
Cayden Codel (Carnegie Mellon University)
Katja Hofmann (Microsoft Research)
Brandon Houghton (Carnegie Mellon University)
Noboru Kuno (Microsoft Research)
Crissman Loomis (Preferred Networks)
Keisuke Nakata (Preferred Networks)
Stephanie Milani (University of Maryland and CMU)
Sharada Mohanty (Alcrowd)
Diego Perez Liebana (Queen Mary University of London)
Ruslan Salakhutdinov (Carnegie Mellon University)
Shinya Shiroshita (Preferred Networks)
Nicholay Topin (Carnegie Mellon University)
Avinash Ummadisingu (Preferred Networks)
Manuela Veloso (Carnegie Mellon University)
Phillip Wang (Carnegie Mellon University)

Advisory committee

Chelsea Finn (Google Brain and UC Berkeley)
Sergey Levine (UC Berkeley)
Harm van Seijen (Microsoft Research)
Oriol Vinyals (Google DeepMind)

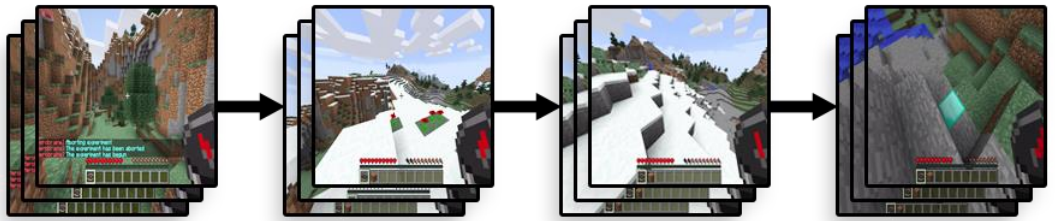
(Part of) the Minecraft Tech Tree



Can multi-task structure help?

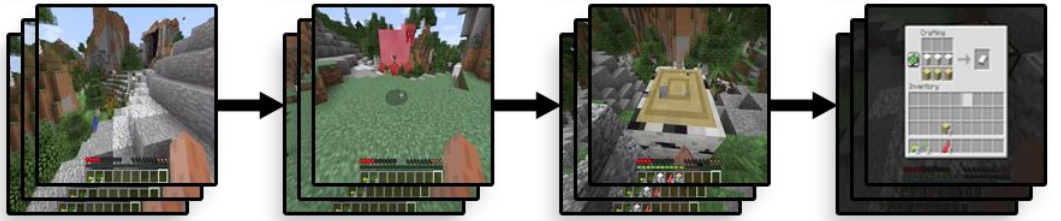
ObtainDiamond and related tasks

Navigate:

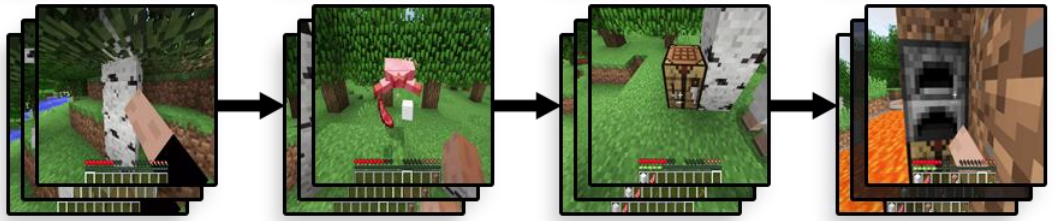


Treechop:

Obtain Bed:

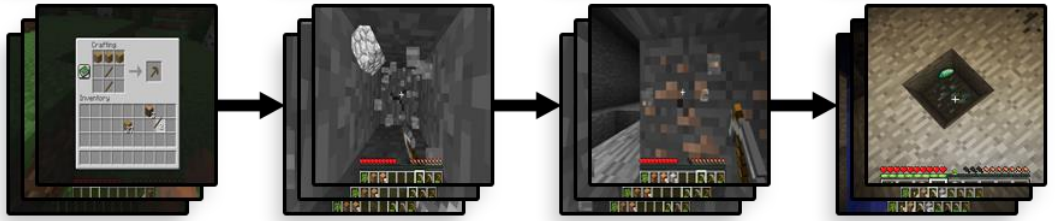


Obtain Meat:



ObtainIron Pickaxe:

Obtain Diamond:



MineRL Competition – Timeline and Resources

- ✓ May 10 – Competition announced
- ✓ June 27 – 9000+ downloads of the minerl competition python package
- Sep 22, 2019: First Round Ends.
- Sep 27, 2019: First Round Results Posted.
- Sep 30, 2019: Final Round Begins.
- Oct 1, 2019 Inclusion@NeurIPS Travel Grant Application Closes (11:00PM EST).
- Oct 25, 2019: Final Round Ends.
- Nov 12, 2019: Final Round Results Posted.
- Dec 8, 2019: Winning teams present their results at the NeurIPS 2019 competition track!

Competition website:

minerl.io/competition

Open now:
Inclusion@NeurIPS
travel grant application

MineRL builds on Project Malmo

A platform for AI experimentation, built on Minecraft

microsoft.com/en-us/research/project/project-malmo/

Open source on github

github.com/Microsoft/malmo

Microsoft / malmo

Unwatch 233 | Unstar 1,998 | Fork 263

Code | Issues 49 | Pull requests 3 | Wiki | Pulse | Graphs | Settings

Project Malmo is a platform for Artificial Intelligence experimentation and research built on top of Minecraft. We aim to inspire a new generation of research into challenging new problems presented by this unique environment. --- For installation instructions, scroll down to *Getting Started* below, or visit the project page for more information: https://www.microsoft.com/en-us/research/project/project-malmo/ — Edit

695 commits | 4 branches | 10 releases | 11 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

timhutton committed on GitHub Merge pull request #300 from Microsoft/xerces_init ... Latest commit efcd5b4 3 days ago

.travis	Minor: removed comments.	20 days ago
ALE_ROMS	Applied MIT license.	2 months ago
Malmo	Fix: having two agent_host's in the same script causes a crash becaus...	4 days ago
Minecraft	Fix: use and attack in discrete movement were being sent to first pla...	4 days ago
Schemas	Fix: time 0 was invalid yet suggested in the documentation.	4 days ago
cmake	Fix: changes to make Lua work on Fedora 23.	2 months ago
doc	Minor: fixed item numbering.	5 days ago
sample_missions	Making cliff_walking_1.xml use discrete actions.	a month ago

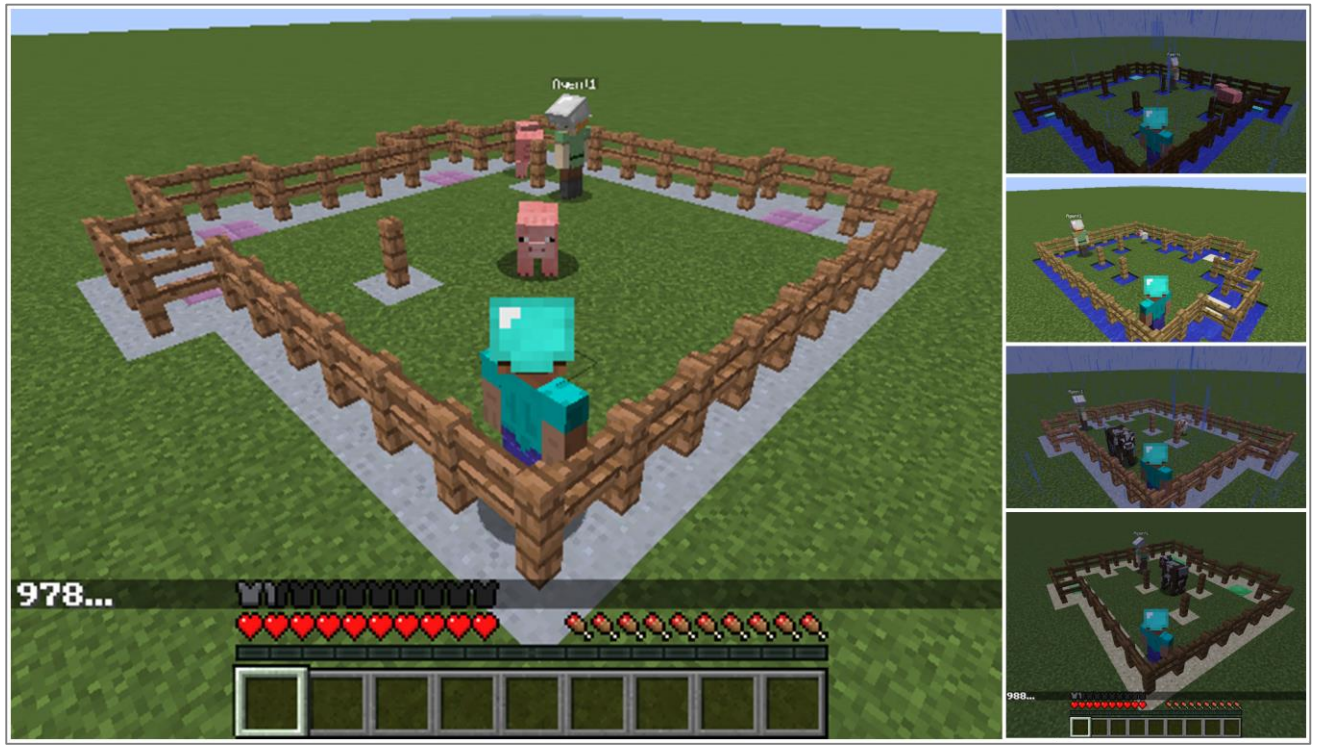
The Malmo Platform for Artificial Intelligence Experimentation
Matthew Johnson, Katja Hofmann, Tim Hutton, & David Bignell 2016

Project Malmo

Open ended experimentation platform

Sample-efficient RL, human priors

Multi-agent Learning, including communication, collaboration



Tomorro

jupyter MLSS-London-2019_Reinforcement-Learning-Tutorial Last Checkpoint: a day ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

Machine Learning Summer School - London 2019

Reinforcement Learning Tutorial

Student version with exercises

Author: Katja Hofmann

This tutorial uses the MineRL package to illustrate how a Reinforcement Learning (RL) agent can learn to interact with the popular video game Minecraft. MineRL was developed by a team led by William H. Guss and Brandon Houghton for the NeurIPS 2019 MineRL competition, hosted by AICrowd and sponsored by Microsoft. MineRL is based on Project Malmo, developed at Microsoft Research.

This tutorial uses the deep learning framework chainer to implement RL algorithms. The tutorial is designed to use as few chainer-specific constructs, but if you'd like to learn more about chainer, take a look at the documentation. Chainer is developed by Preferred Networks (PFN), a co-organizer of the MineRL competition. The code here is designed for educational purposes. If you'd like to explore reinforcement learning and the MineRL competition further, take a look at the chainerrl minerl baselines provided by PFN.

Overview

This tutorial demonstrates how to build an RL agent that learns to navigate: first in SimpleRooms, a task we implement from scratch, then in a MineRL task in Minecraft. Some parts of the tutorial are optional, as shown below - you can learn about key concepts in reinforcement learning and implement your first agent without installing the MineRL package.

1. Setup **Tip: run this section before the start of the tutorial, to make sure you're ready to get started.**
   A. General Prerequisites
   B. Install the MineRL Package (optional): install MineRL. If skipped, you can still follow the agent implementation and run it on the simple maze environment.
   C. Test MineRL (optional): start Minecraft and test out interaction with the game.
2. RL Components: **Learn how to implement the core components of an RL experiment: environment, agent, and the experiment itself.**
   A. Environment
   B. Agent
   C. Experiment