Microsoft

Reinforcement Learning: Past, Present, and Future Perspectives

Katja Hofmann

Principal Researcher, Game Intelligence
Microsoft Research, Cambridge, UK

aka.ms/gameintelligence
Twitter: @katjahofmann

Reinforcement Learning =

Decision Making and

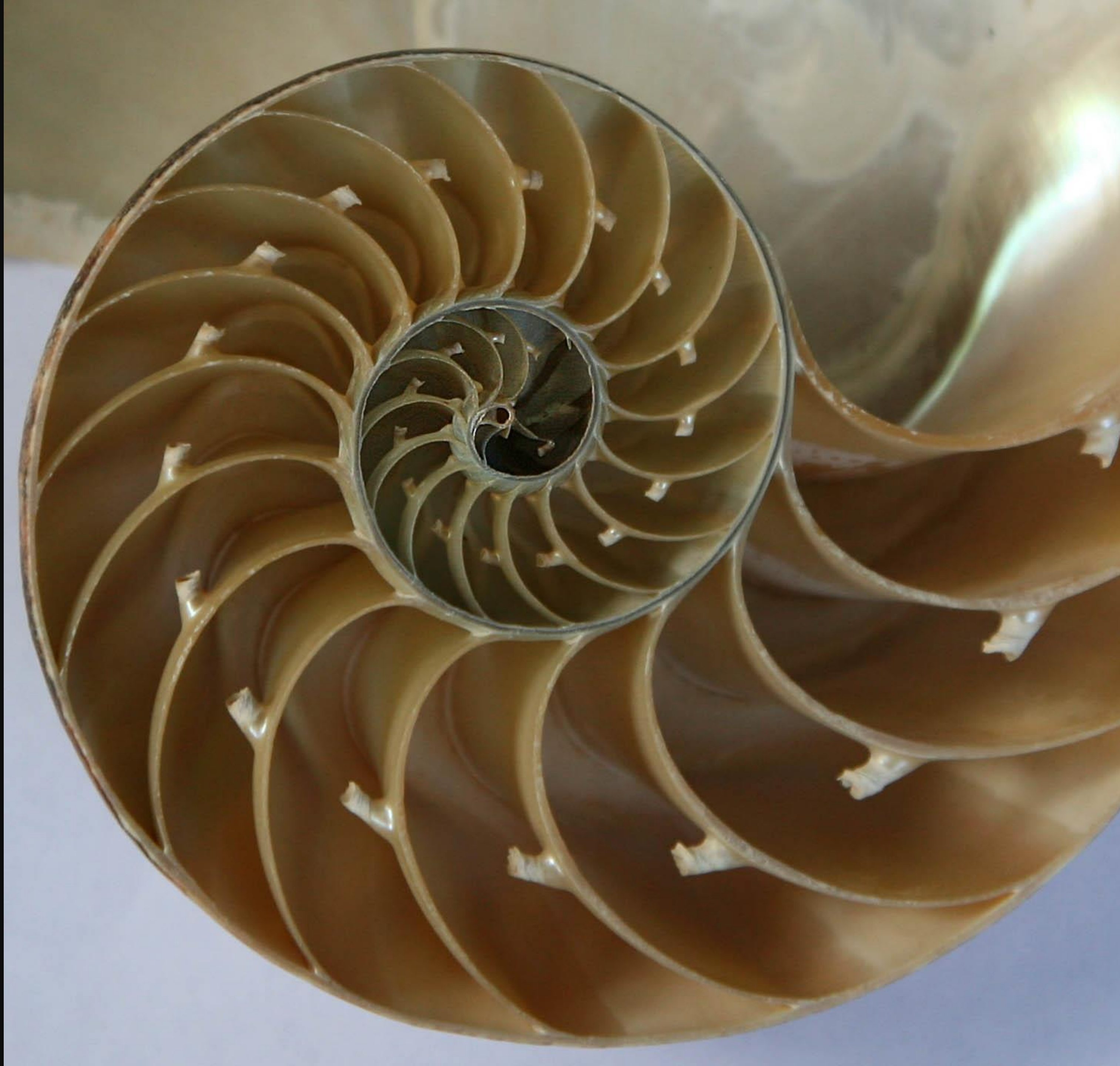
Learning

under Uncertainty


Under
Rookie
Microsoft

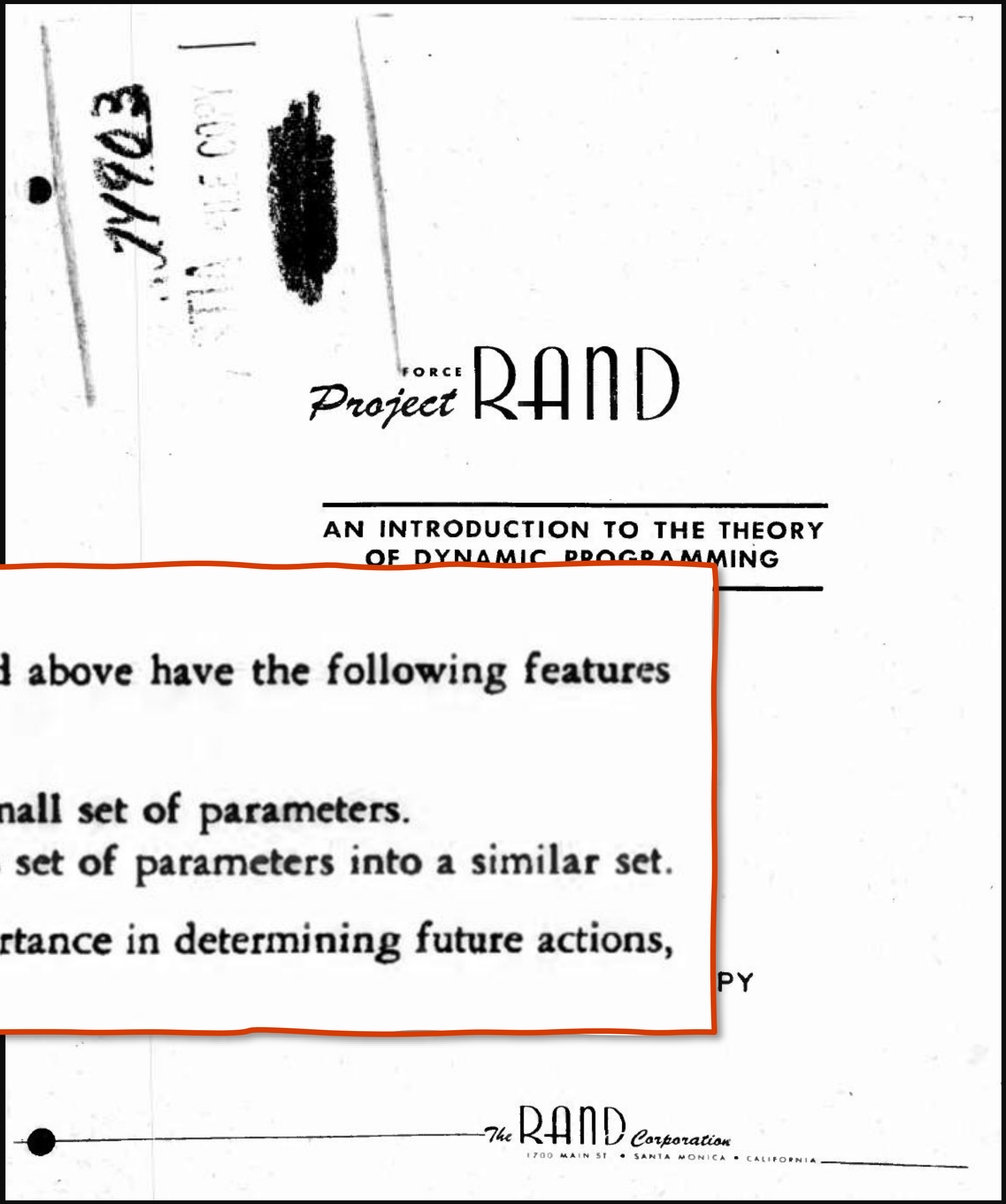
Plan for Today

1. Formalizing RL
2. Value Functions
3. Exploration
4. Policy Gradient and Actor Critic Approaches
5. Generalization
6. Structure
7. Models
8. New Challenges



1. Formalizing RL

Markov Decision Processes (MDPs)


FORCE Project RAND

AN INTRODUCTION TO THE THEORY OF DYNAMIC PROGRAMMING

PY

The RAND Corporation

1700 MAIN ST • SANTA MONICA • CALIFORNIA

1.6. The Functional Equation Approach

Let us begin by observing that the problems posed above have the following features in common:

1. The state of the system is described by a small set of parameters.
2. The effect of a decision is to transform this set of parameters into a similar set.
3. The past history of the system is of no importance in determining future actions, a Markovian property.

[Bellman 1953, 1954, 1957; Puterman 1994]

Markov Decision Processes (MDPs)

agent
with a (learnable)
behaviour policy

environment
with initially unknown
dynamics and reward

Markov Decision Processes (MDPs)

agent
with a (learnable)
behaviour policy

state $s_0 \in S$

environment
with initially unknown
dynamics and reward

Markov Decision Processes (MDPs)

action $a_t \in A$

action

reward $r_t \in \mathbb{R}$
state $s_t \in S$

agent
with a (learnable)
behaviour policy

environment
with initially unknown
dynamics and reward

Markov Decision Processes (MDPs)

action $\mathrm{a}_t \in A$

reward $\mathrm{r}_{t+1} \in \mathbb{R}$

state $\mathrm{s}_{t+1} \in S$

agent
with a (learnable)
behaviour policy

environment
with initially unknown
dynamics and reward

Markov Decision Processes (MDPs)

action $\mathrm{a}_t \in A$

reward $\mathrm{r}_{t+1} \in \mathbb{R}$

state $\mathrm{s}_{t+1} \in S$

Behavior policy:
$\pi(\mathrm{a}|\mathrm{s})$

Dynamics: $\mathrm{T}(s_{t+1}|s_t, a_t)$

Reward: $R(r_{t+1}|s_t, a_t)$

Optimality in Markov Decision Processes

Finite-horizon:

$$\mathbb{E}\left(\sum_{t=0}^{h} r_t\right)$$

Infinite-horizon:

$$\mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

Average-reward:

$$\lim_{h\to\infty} \mathbb{E}\left(\frac{1}{h}\sum_{t=0}^{h} r_t\right)$$

[Kaelbling, Littman & Moore, 1996]

Learning performance

Asymptotic convergence:

$$\pi_n \to \pi^* \; as \;\; n \to \infty$$

PAC:

$$P(N_{errors} > F(\cdot, \epsilon, \delta)) \leq \delta$$

Regret (e.g., bound B on total regret):

$$\max_j \sum_{t=0}^{T} r_{tj} - r_t < B$$

[Dann, Lattimore & Brunskill 2017] unify notion of PAC and regret into Uniform-PAC

[Kaelbling, Littman & Moore, 1996]

Key RL challenges

- Explore – exploit
- Credit assignment
- Function approximation

2. Value Functions

Dynamic Programming and Bellman Equations

Optimal state-value function:

$$V^*(s_t) = \max_{\pi} \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

Bellman equation defines recursively:

$$V^\pi(s_t) = R\big(s_t, \pi(s_t)\big) + \gamma \sum_{s_{t+1}} T(s_{t+1}|s_t, \pi(s_t))V^\pi(s_{t+1})$$

Bellman optimality equation = Bellman eq for π^*

$$V^{\pi^*}(s_t) = \max_{a} R(s_t, a) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a)V^\pi(s_{t+1})$$

[Bellman 1957]

Temporal Difference (TD) Error and TD(0)

Observe samples $< s_t, a_t, r_t, s_{t+1} >$. If value estimates are accurate, the following must hold:

$$V(s_t) = r_t + \gamma V(s_{t+1})$$

If not, there is an error (TD error):

$$\delta = r_t + \gamma V(s_{t+1}) - V(s_t)$$

To learn better estimates – minimize δ (TD(0)):

$$V(s) \leftarrow V(s) + \alpha\left(r_t + \gamma V(s_{t+1}) - V(s_t)\right)$$

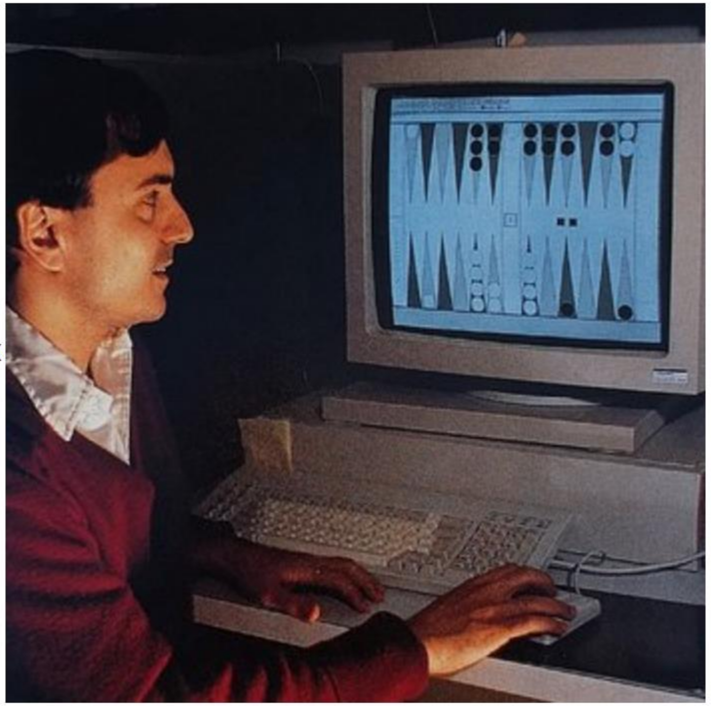
[Samuel 1959; Sutton 1984, 1988]

TD-Gammon

Artificial Intelligence Accomplishment | 1990s

IBM researchers: Gerald Tesauro

Where the work was done: T.J. Watson Research Center



What we accomplished: Gerald Tesauro (pictured) developed an innovative combination of nonlinear function approximation with reinforcement learning (RL) techniques and showed it could achieve success in large-scale complex decision making problems. The approach was tested in a self-teaching backgammon program called TD-Gammon. Starting from a random initial strategy, and learning its strategy almost entirely from self-play, TD-Gammon achieved a remarkable level of performance. When operating without any lookahead search, it demonstrated a highly sophisticated sense of positional judgement rivaling that of human masters. When its positional evaluation was augmented by very shallow (2-ply, 3-ply) search procedures, the program matched and ultimately surpassed the playing ability of world-champion human players. This achievement has been highly influential in the AI and computer gaming communities, and has inspired numerous real-world applications of similar RL techniques.

Related links: Temporal difference learning and TD-Gammon, March 1995 paper in Communications of the ACM.

Image credit: IBM Think Magazine, December 1992

Image credit: https://en.wikipedia.org/wiki/TD-Gammon

Credit: IBM Research https://researcher.watson.ibm.com/researcher/view_page.php?id=6853

Q-Learning

Bellman optimality equation for Q:

$$Q^*(s_t, a_t) = \mathbb{E}_{\pi^*}\left(r_t + \gamma \max_a Q^*(s_{t+1}, a)\right)$$

$$\delta = r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a)$$

[Watkins 1989; Watkins & Dayan 1992]

Q-Learning Algorithm

```
For each episode:
  Observe initial state s_0
  for each step t = 0,1,2 ... in the episode:
    Select action a_t using Q(a,s) (e.g., ϵ-greedy)
    Take action a_t, observe r_t, s_{t+1}
    Q(s_t, a_t) = Q(s_t, a_t) + α[r_t + γ max_a Q(s_{t+1}, a) − Q(s_t, a)]
    s = s_{t+1}
```

For each episode:

Observe initial state s_0

for each step $t = 0,1,2 \ldots$ in the episode:

Select action a_t using $Q(a, s)$ (e.g., ϵ-greedy)

Take action a_t, observe $\mathrm{r}_t, \mathrm{s}_{t+1}$

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a)]$$

$s = \mathrm{s}_{\mathrm{t+1}}$

[Watkins, 1989; Dayan & Watkins, 1992]

Regret bounds for Q-Learning:
Chi Jin, Allen-Zhu, Bubeck & Jordan: "Is q-learning provably efficient?" NeurIPS 2018

Project Malmo

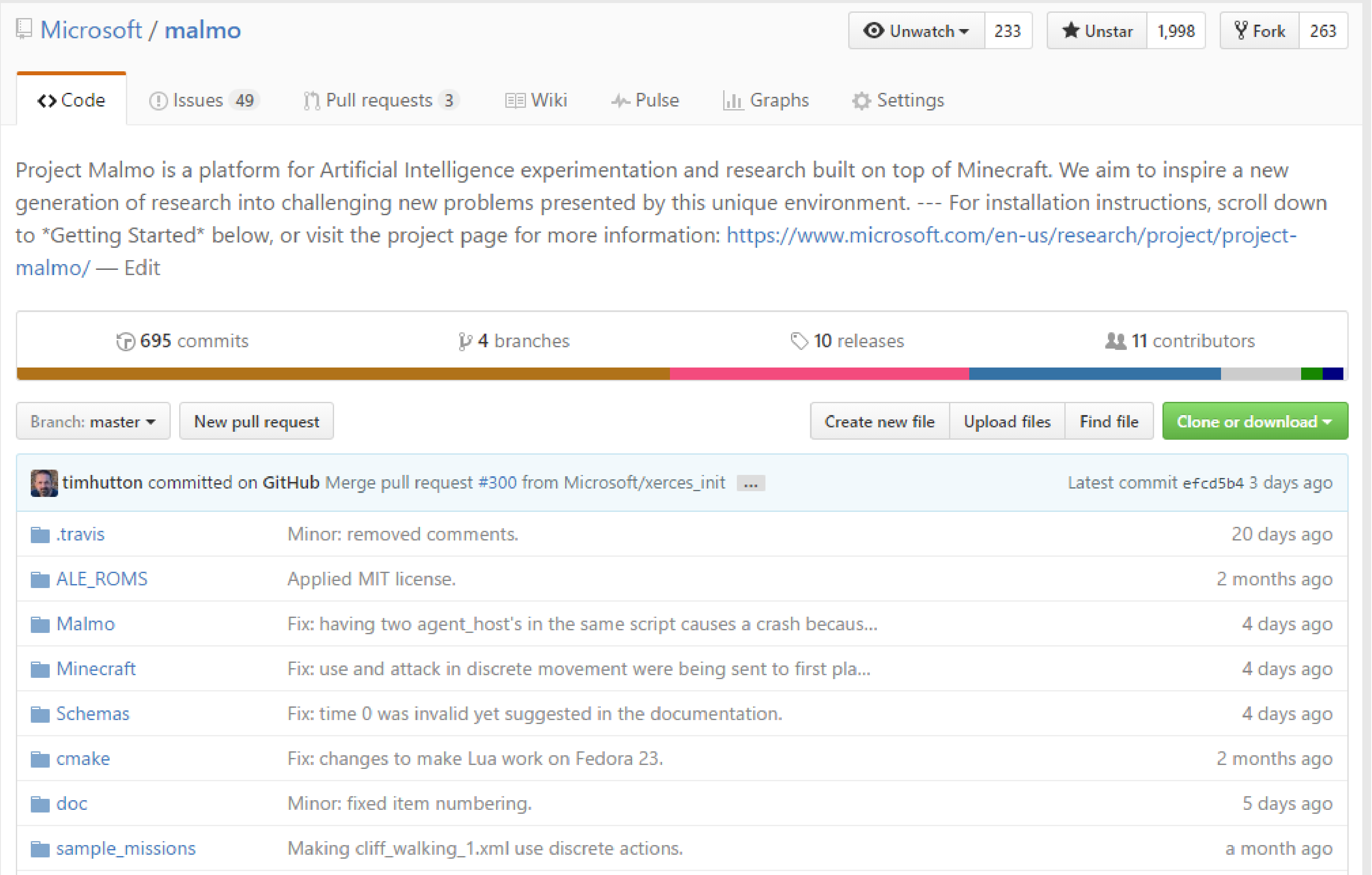
A platform for AI experimentation, built on Minecraft

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

Open source on github

github.com/Microsoft/malmo

[Johnson, **Hofmann**, Hutton & Bignell, 2016]



Q-Learning in Malmo

Task: navigate an initially unknown environment

Adapted from Sutton & Barto (2018) chapter 6

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

Q-Learning in Malmo: Task Definition

Positive reward

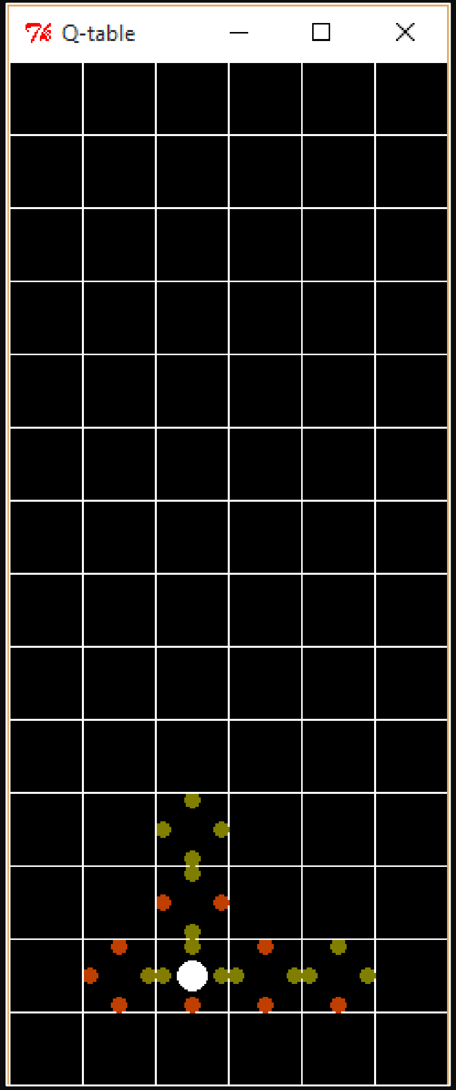
Negative reward

Task: navigate an initially unknown environment

Adapted from Sutton & Barto (2018) chapter 6

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

Q-Learning in Malmo: Q-table



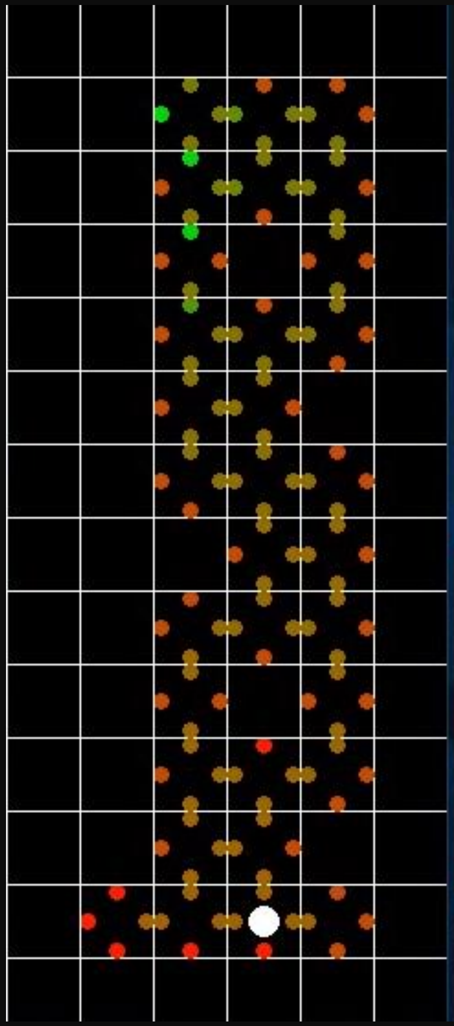
Try this at home, see https://github.com/Microsoft/malmo - tutorial 6

Q-Learning in Malmo: Initial policy

The agent has to explore to learn about consequences of it's actions

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

Q-Learning in Malmo:



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

3. Function Approximation

Q-Learning with Function Approximation

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

$$\delta = r_t + \gamma \max_a Q(s_{t+1}, a; \theta) - Q(s_t, a; \theta)$$

Turn into an optimization problem by minimizing the loss on the TD error:

$$\begin{aligned} \mathrm{J}(\theta) &= \|\delta\|^2 \\ &= \left\| r_t + \gamma \max_{a \in A} Q\left(s_{t+1}, a; \theta\right) - Q\left(s_t, a_t; \theta\right) \right\|^2 \end{aligned}$$

[Watkins 1989, Riedmiller 2000, 2005]

Stability

The “deadly triad” [Sutton & Barto, 2018]

1) Off-policy learning
2) Flexible function approximation
3) Bootstrapping

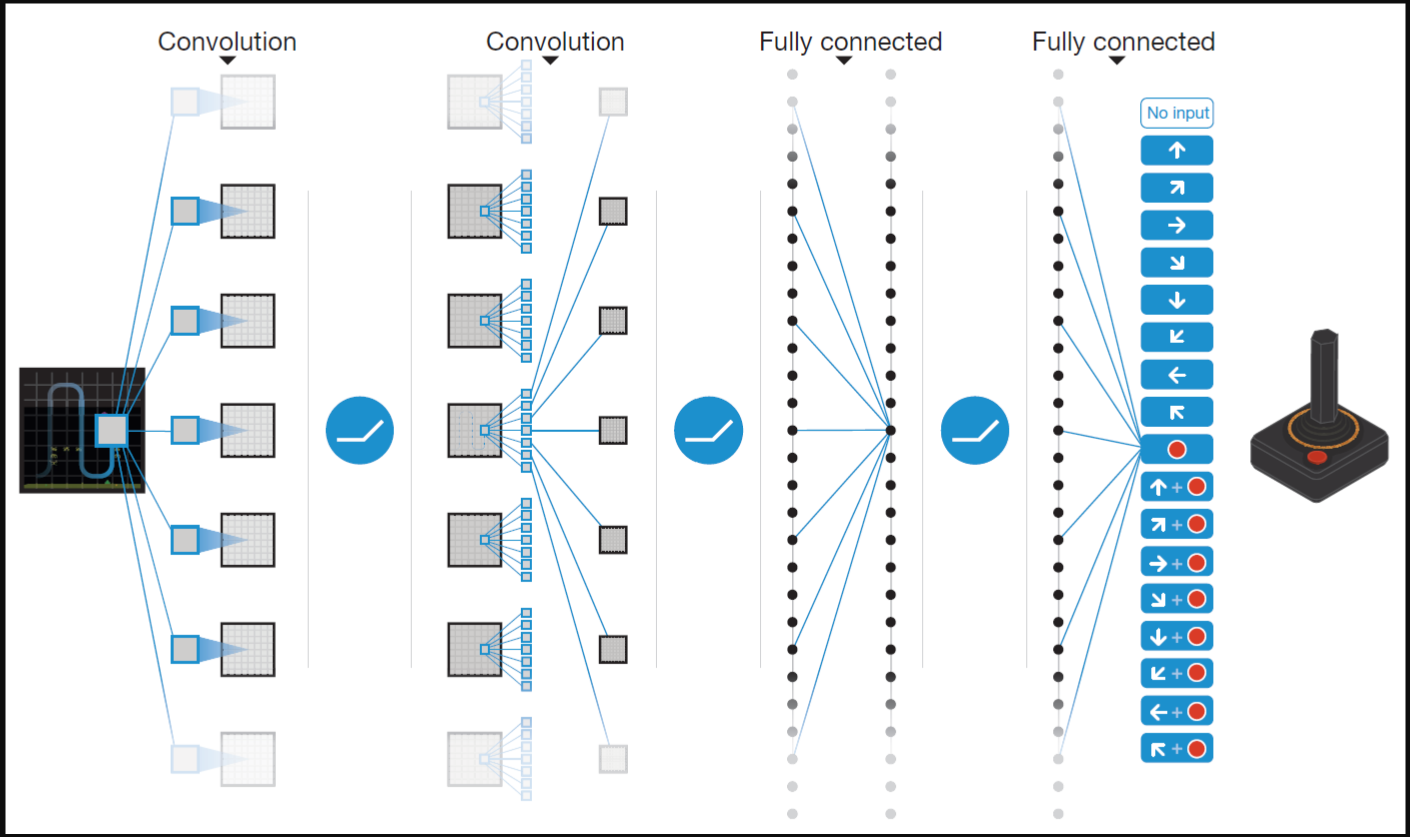
In the face of all three, learning is unstable (can and will diverge) [Baird 1995; Tsitsiklis & Van Roy 1997]

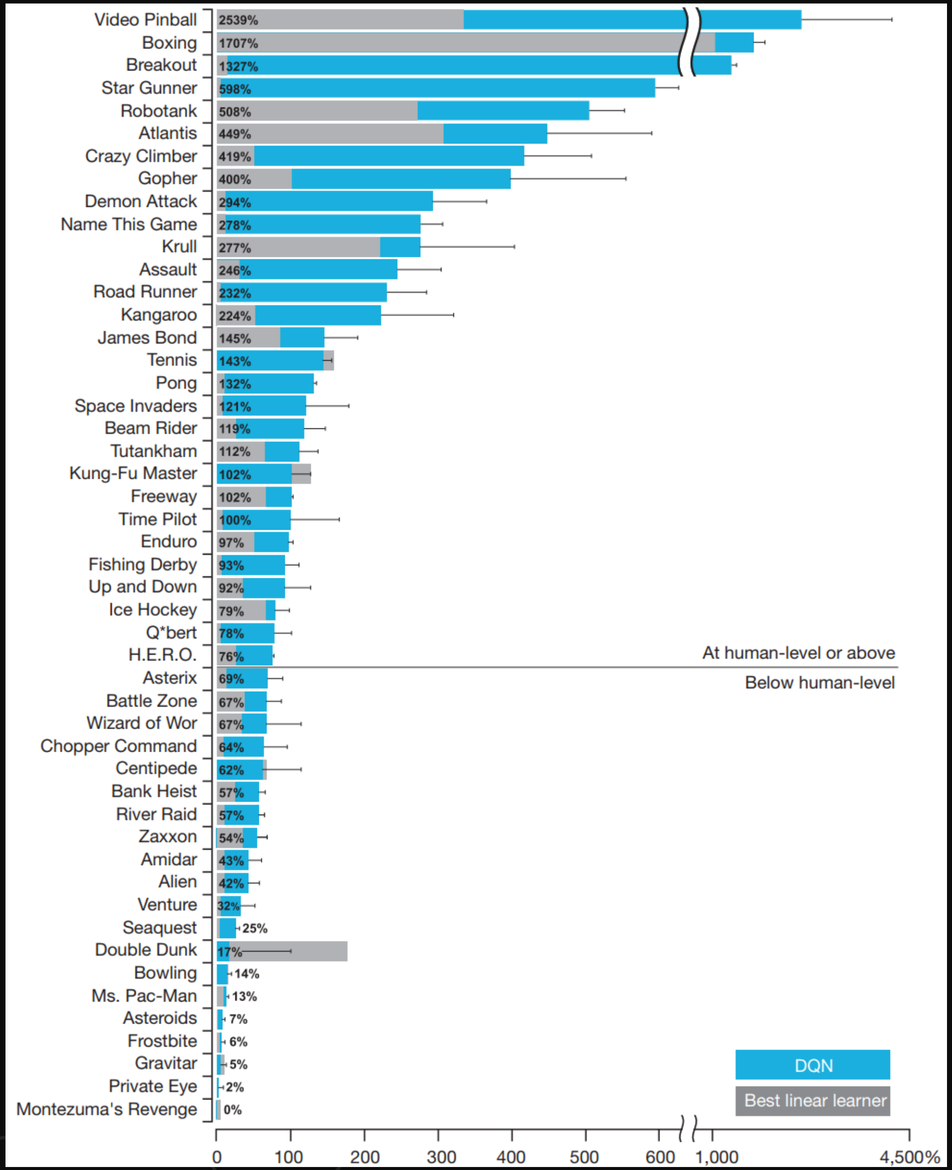
DQN [Mnih et al. 2013, 2015] stabilizes learning:

1) Experience replay buffer [Lin 1993] + mini-batch SGD
2) Separate target network stabilizes optimization targets: $\delta = r_t + \gamma \max_{a \in A} Q(s_{t+1}, a; \theta') - Q(s_t, a_t; \theta)$
3) Clip δ to $[-1, 1]$

Great blog post with code (DQN, Double DQN):
https://davidsanwald.github.io/2016/12/11/Double-DQN-interfacing-OpenAi-Gym.html

Results





Figures from [Mnih et al. 2015]. Training setup across all 49 Atari games (above); Results in terms of human-normalized scores (right)

Figure from: Mr

Improving DQN (Selection)

[Van Hasselt et al. 2016] Double Q-Learning – reduce bias

[Anschel et al. 2017] Average Q-Learning – reduce variance

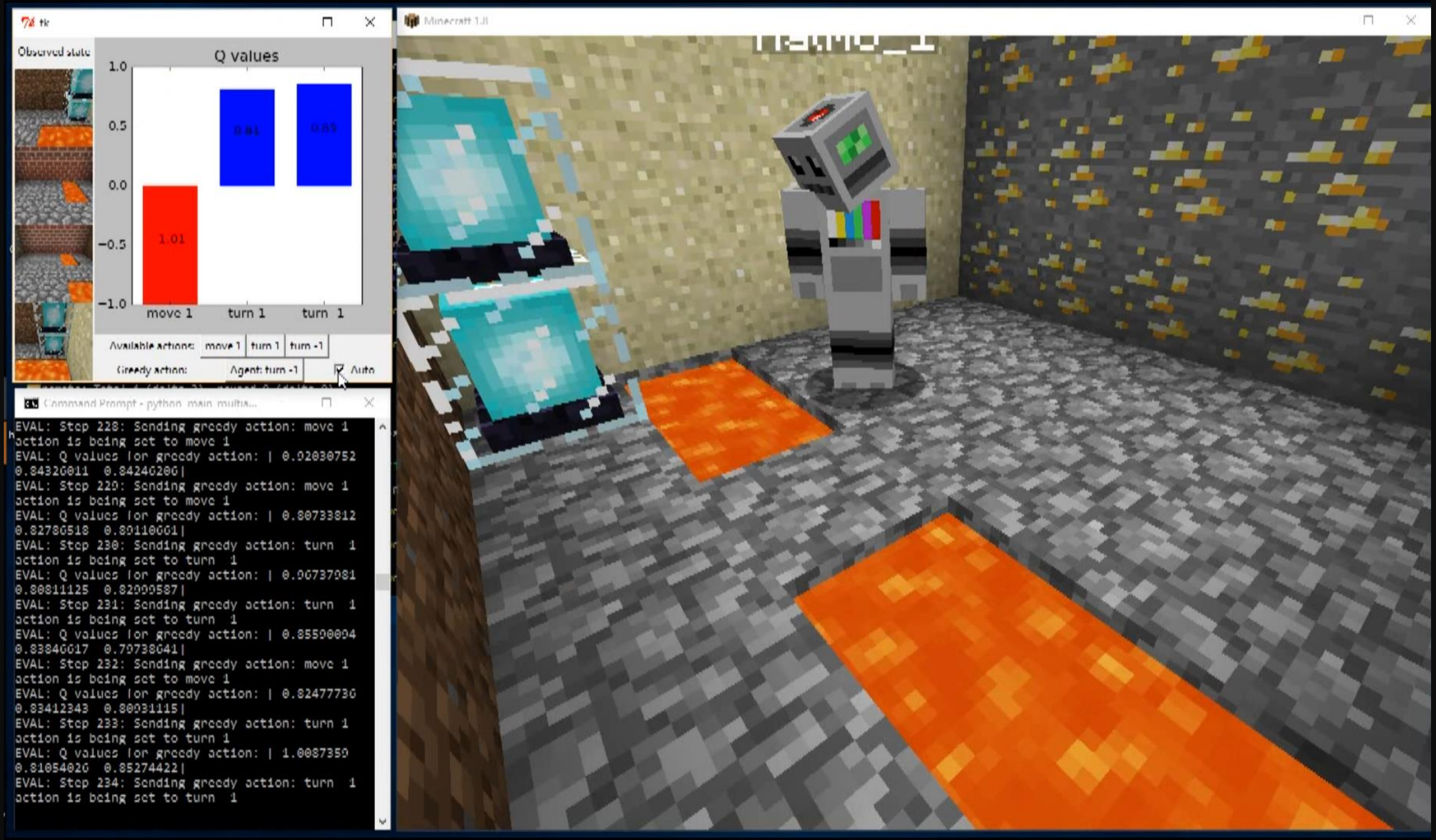
[Andrychowicz et al. 2017] Hindsight Experience replay

[Dabney et al. 2018] Distributional RL (quantile regression)

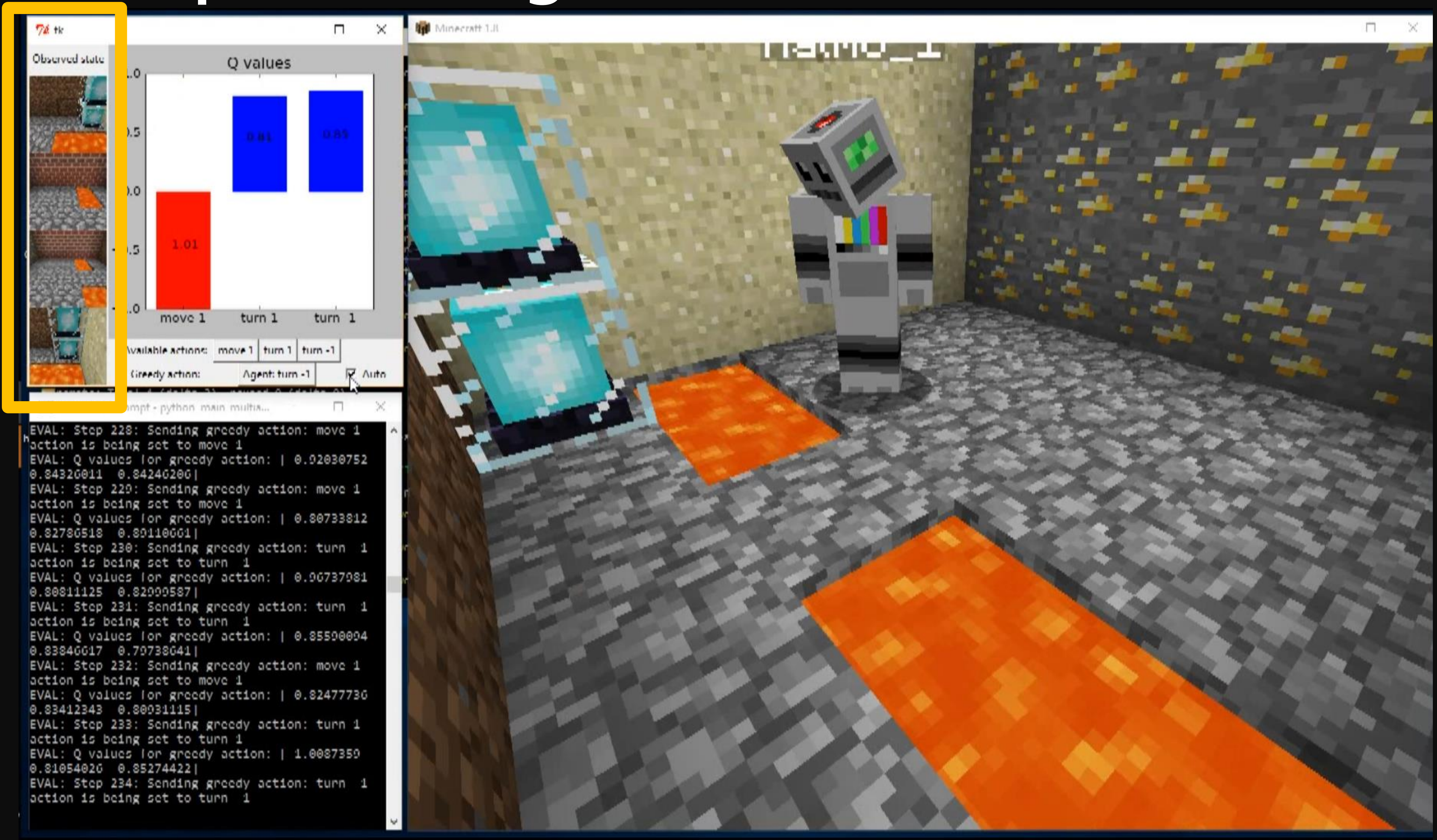
[Horgan et al. 2018] **Ape-X** – distributed replay buffer

For further study check David Silver's ICML 2016:
https://www.icml.cc/2016/tutorials/deep_rl_tutorial.pdf

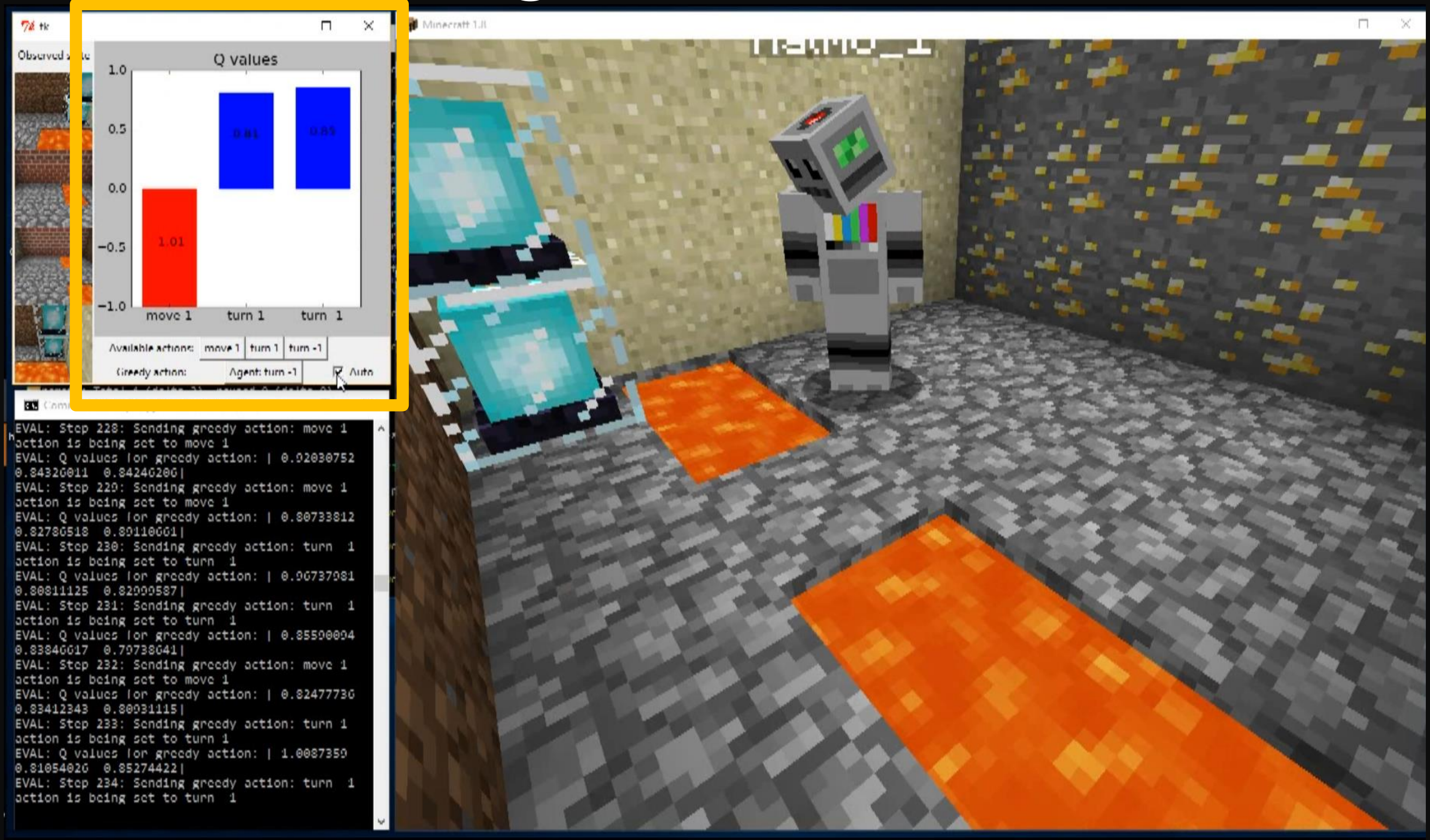
Case Study: Learning to navigate Minecraft from pixels using DQN



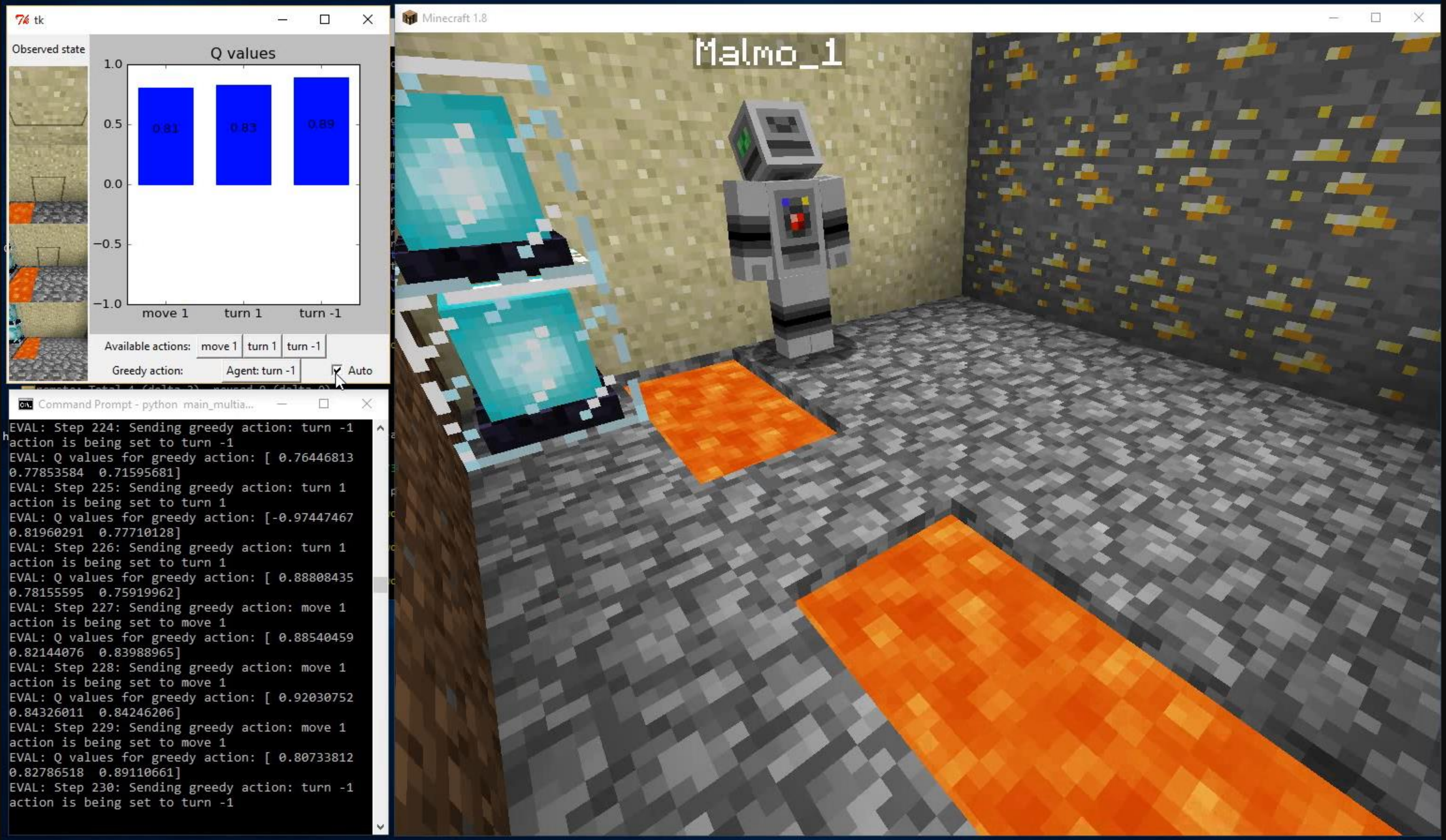
Case Study: Learning to navigate Minecraft from pixels using DQN



Case Study: Learning to navigate Minecraft from pixels using DQN



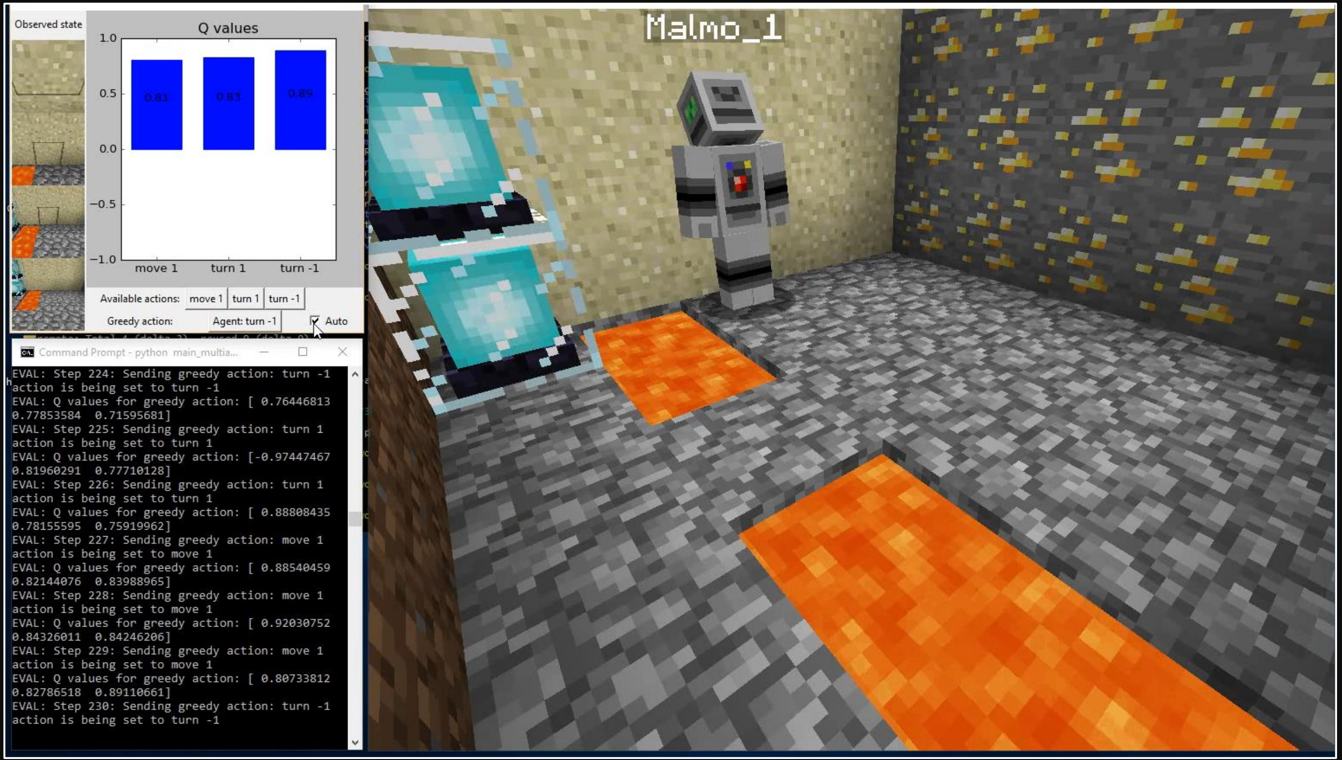
Case Study: Learning to navigate Minecraft from pixels using DQN

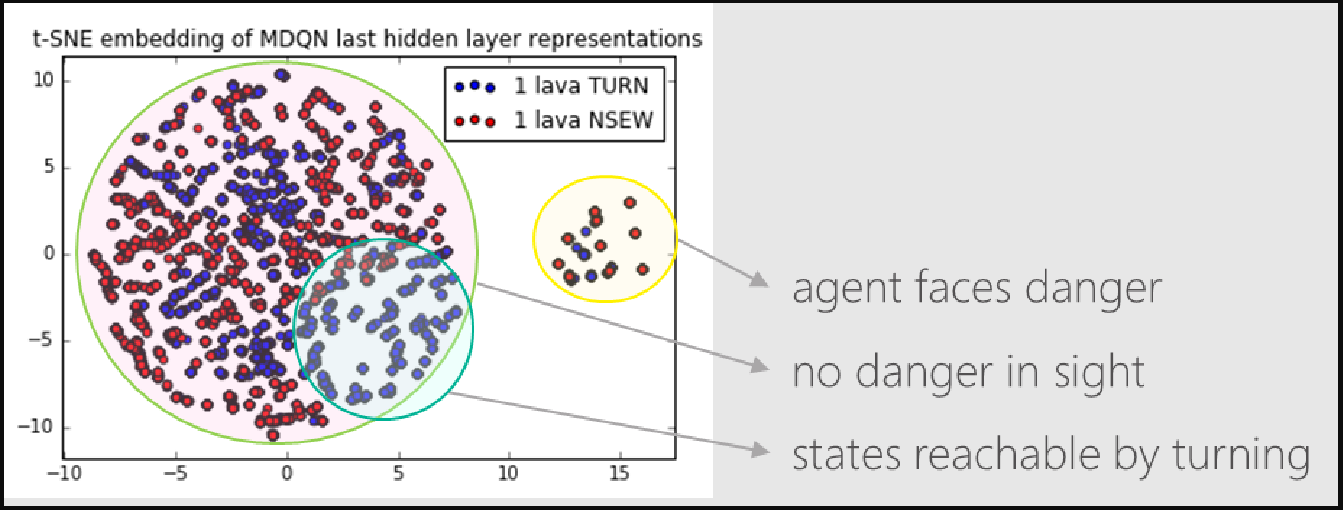


Decoding multitask DQN in the world of Minecraft

Lydia Liu, Urun Dogan, Katja Hofmann

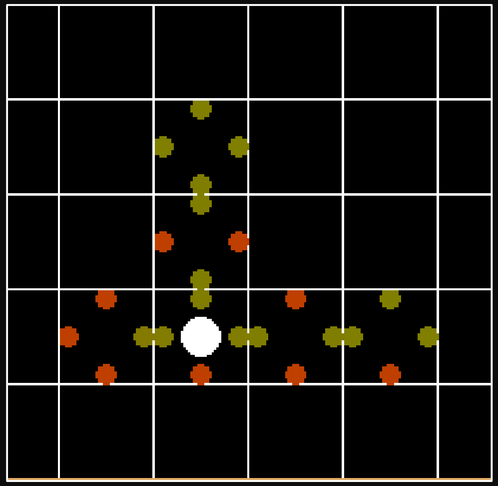
EWRL 2016
Deep Learning Workshop @ NIPS 2016





3. Exploration

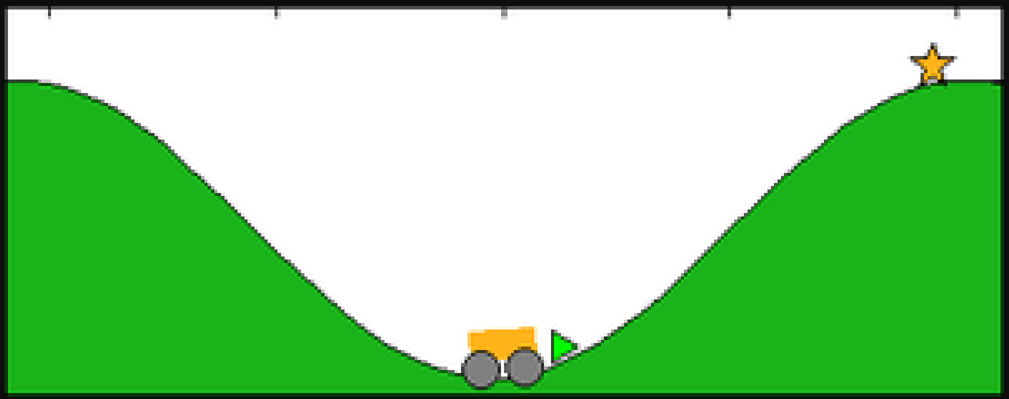
Exploration vs Exploitation – Common Approaches



Optimistic initialization

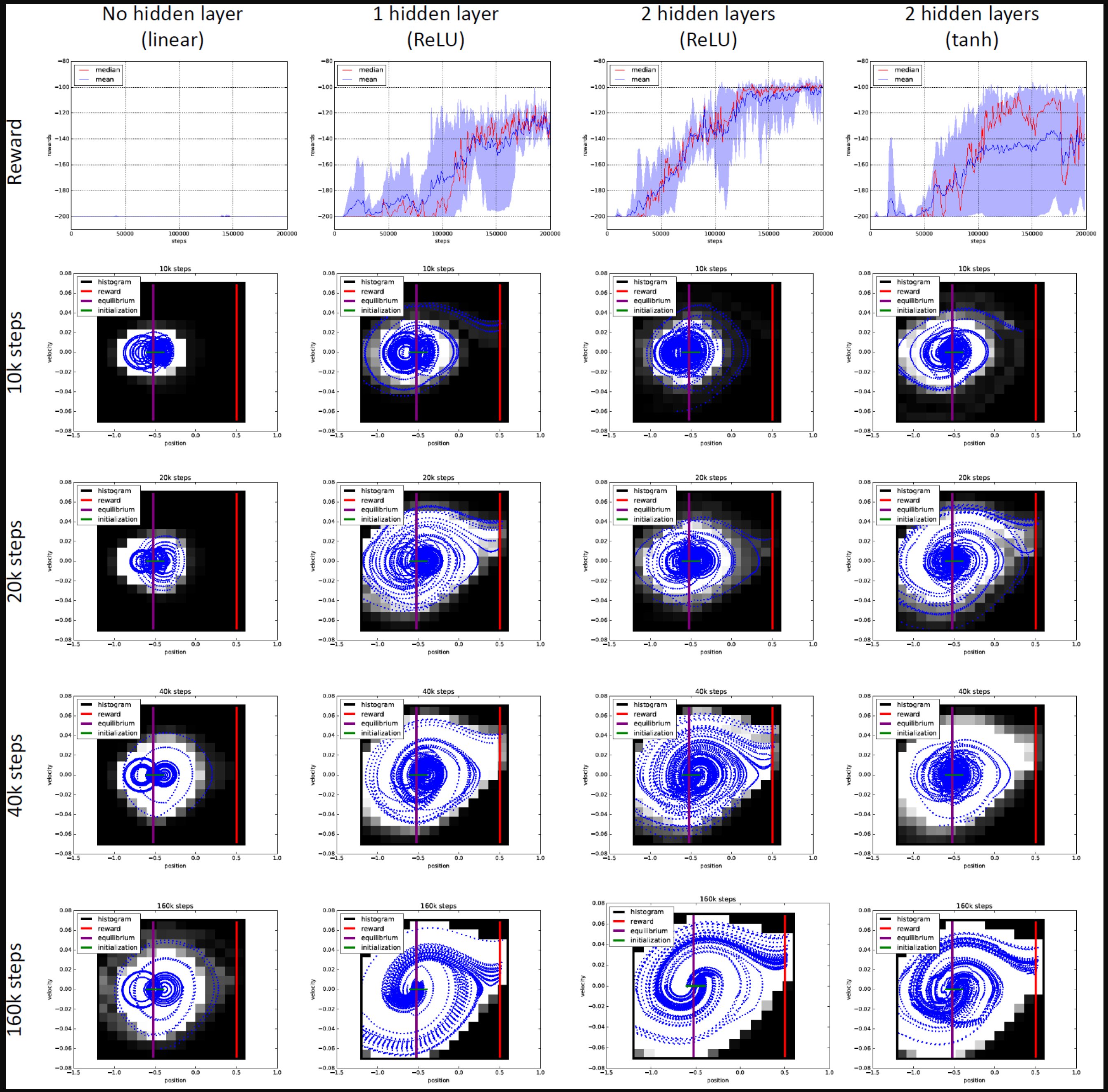
If upper bound is known (e.g., on Q), initialize all estimates to the upper bound.

Example: Interaction between optimistic initialization and function approximation

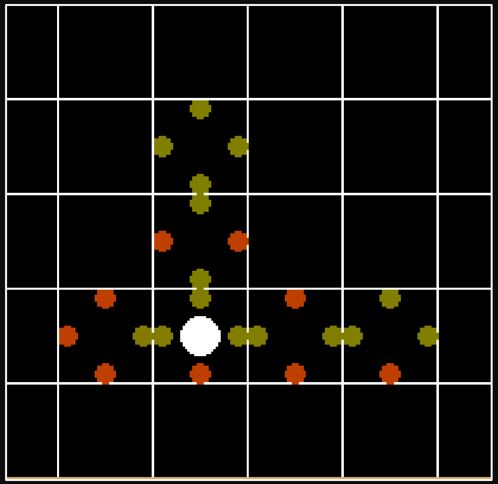


https://en.wikipedia.org/wiki/Mountain_car_problem

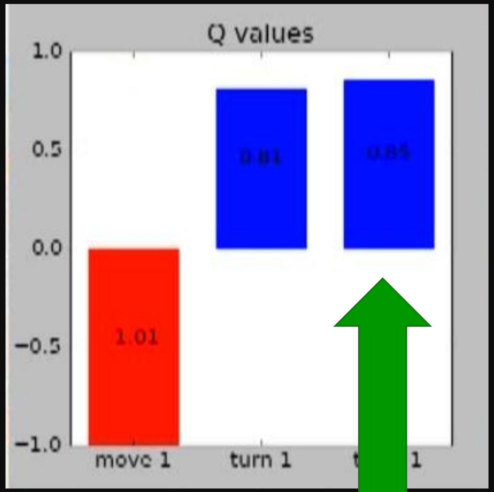
[Dauparas, Tomioka & Hofmann, 2018]



Exploration vs Exploitation – Common Approaches



Optimistic initialization

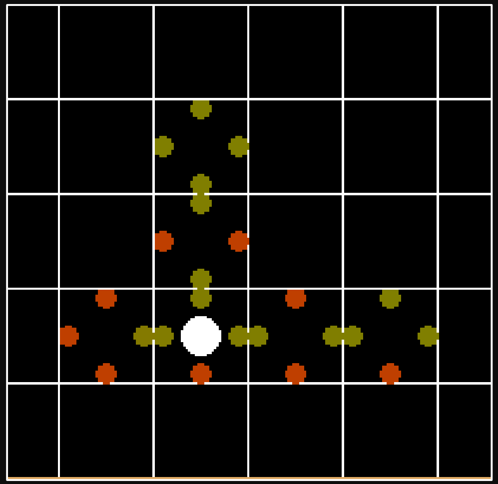


"greedy" action

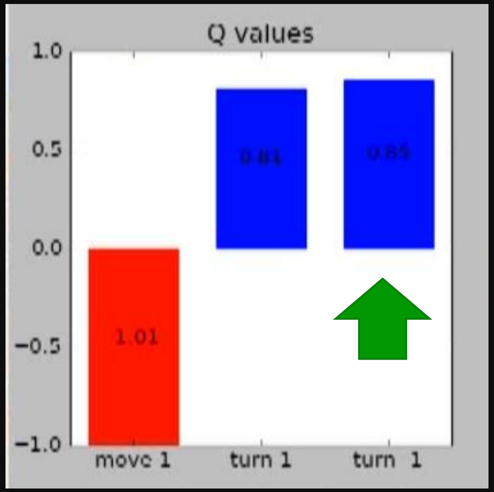
Epsilon-greedy

$$\pi_t = \begin{cases} \underset{a \in A}{\operatorname{argmax}}\, \hat{r}_t(a) & w.prob.\, 1-\varepsilon \\ rand(a) & w.prob.\, \varepsilon \end{cases}$$

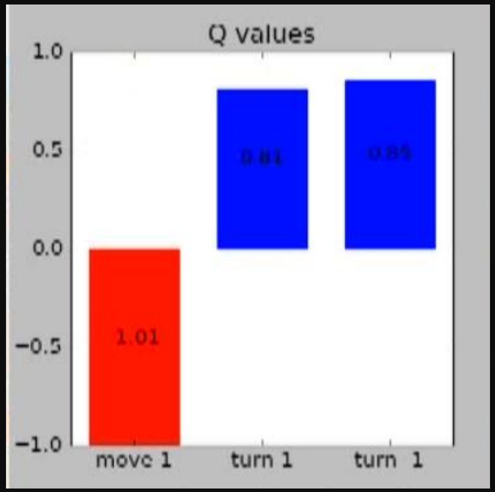
Exploration vs Exploitation – Common Approaches



Optimistic initialization



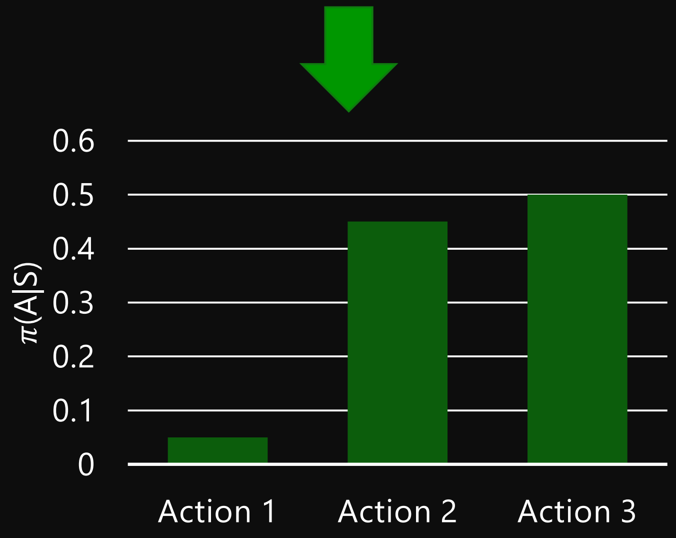
Epsilon-greedy



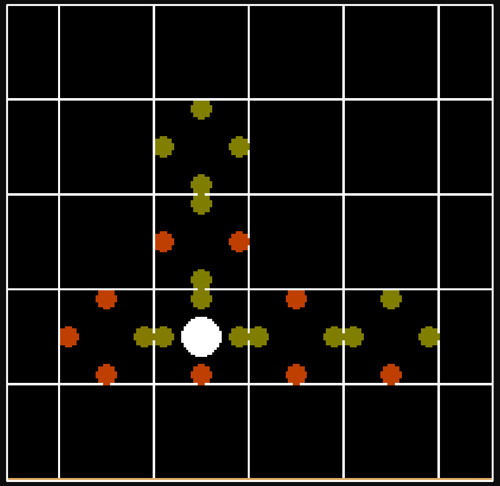
Soft-max

Sample from the Softmax policy:

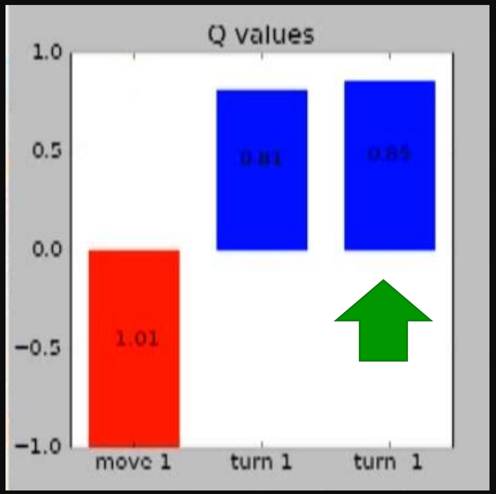
$$\pi(a|s) = \frac{e^{h(s,a)}}{\sum_{a' \in A} e^{h(s,a')}}$$



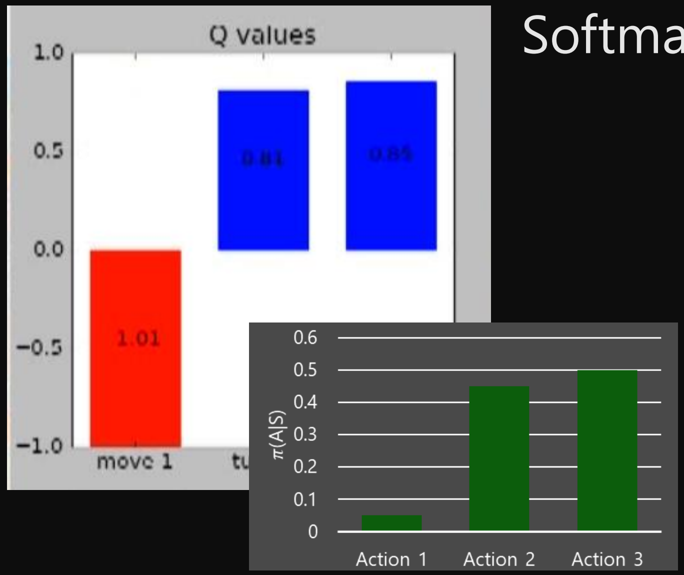
Exploration vs Exploitation – Optimistic initialization



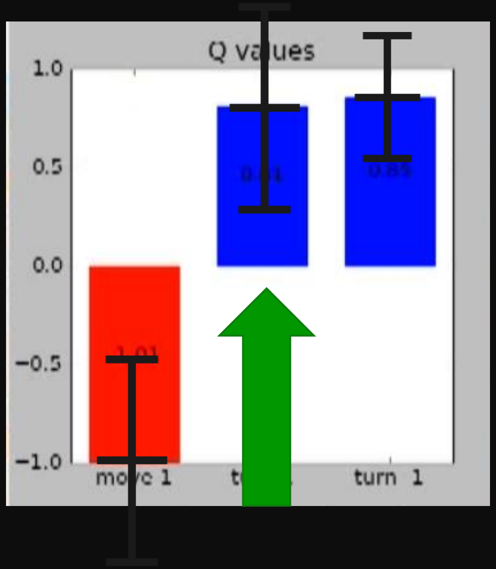
Optimistic initialization



Epsilon-greedy



Softmax



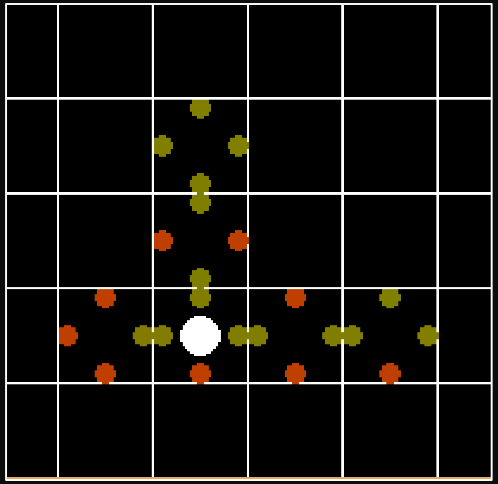
Upper confidence bound

Derive Upper Confidence Bound (UCB), e.g., for bandits:

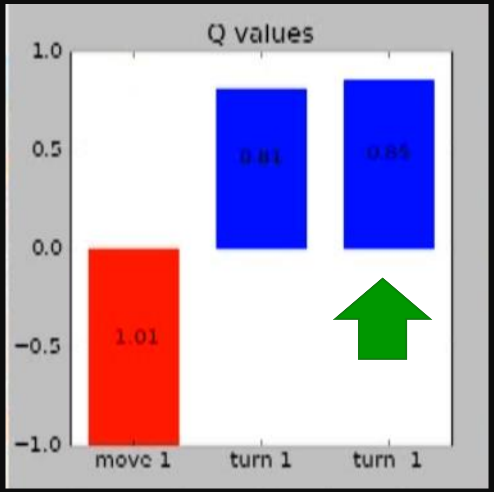
$$\pi_t = \underset{a \in A}{\operatorname{argmax}} \, \hat{r}_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$$

[Auer et al. ’02]

Exploration vs Exploitation – Optimistic initialization

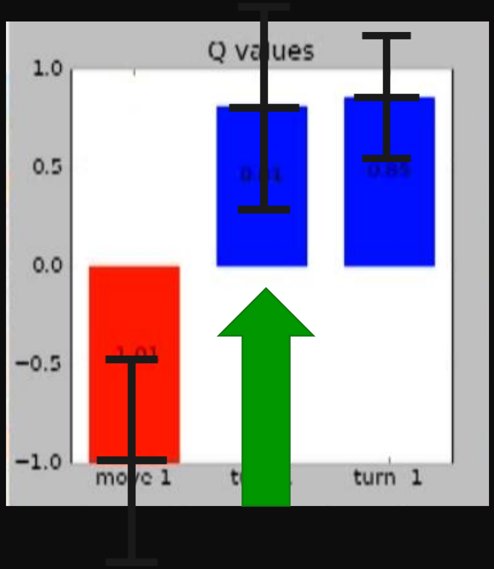


Optimistic initialization

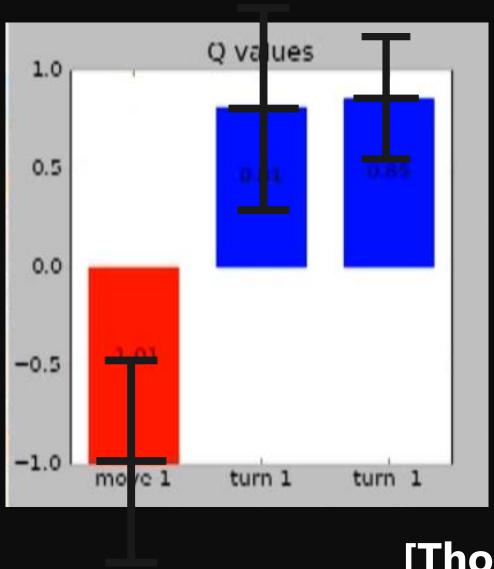


Epsilon-greedy

Softmax



Upper confidence bound



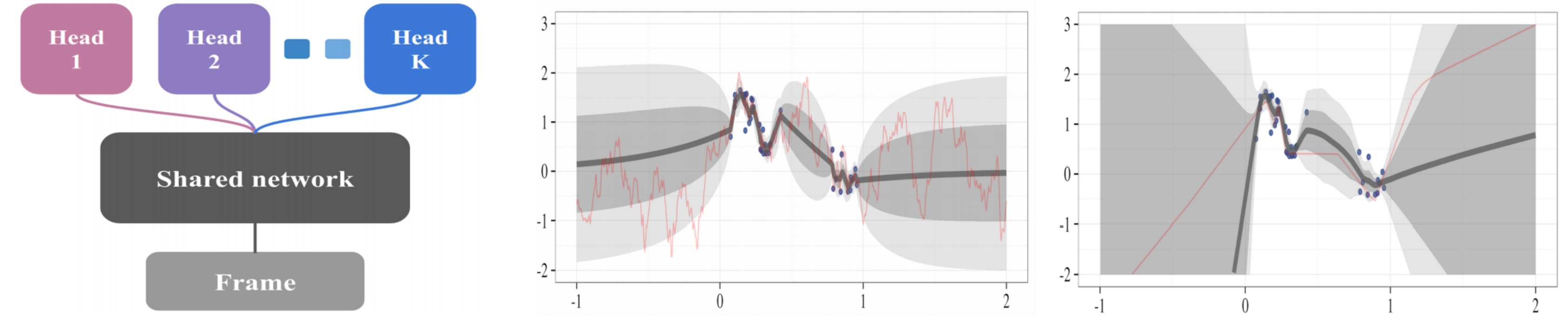
Posterior sampling

Maintain distribution $P(r|a)$. At time t sample from this distribution, and take the optimal action according to the sample; update P.

[Thompson ‘33, Chapelle & Li ‘11, Russo & Van Roy ‘14]

Deep exploration using Bootstrapped DQN

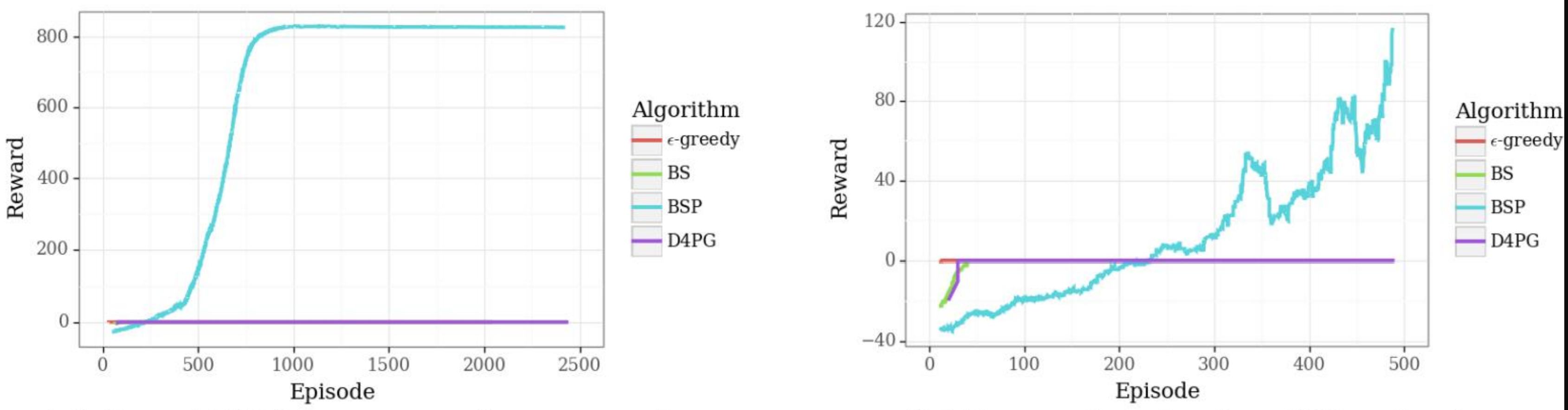
Idea (BDQN): Approximate uncertainty over Q using deep ensembles [Osband et al. 2016]



(a) Shared network architecture (b) Gaussian process posterior (c) Bootstrapped neural nets

Figure 1: Bootstrapped neural nets can produce reasonable posterior estimates for regression.

[Osband et al. 2018] extend BDQN with randomized prior function



(a) Only BSP learns a performant policy. (b) Inspecting the first 500 episodes.

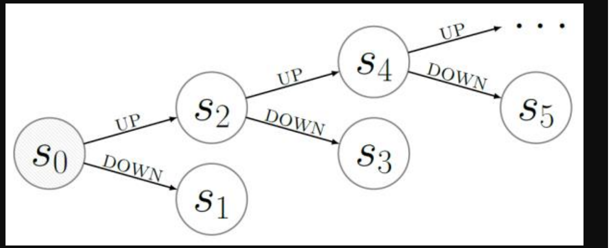
Figure 5: Learning curves for the modified cartpole swing-up task.

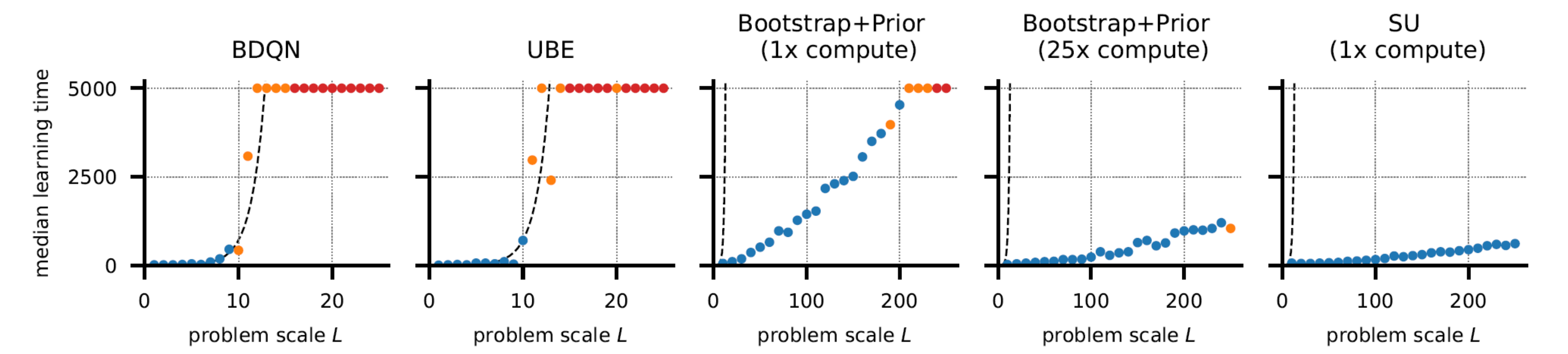
Successor Uncertainties

Idea: approximate uncertainty over Q as a function of successor features [Dayan 1993]

Objective: $\underbrace{|\langle \phi_t, w \rangle - r_{t+1} - \langle \psi_{t+1}, w \rangle^{-}|^2}_{\text{standard Q value loss}} + \underbrace{\|\psi_t - \phi_t - \gamma \psi_{t+1}^{-}\|^2}_{\text{succ. feat. regularisation}} + \underbrace{|\langle \phi_t, w \rangle - r_{t+1}|^2}_{\text{reward prediction loss}}$

Results: chain MDP



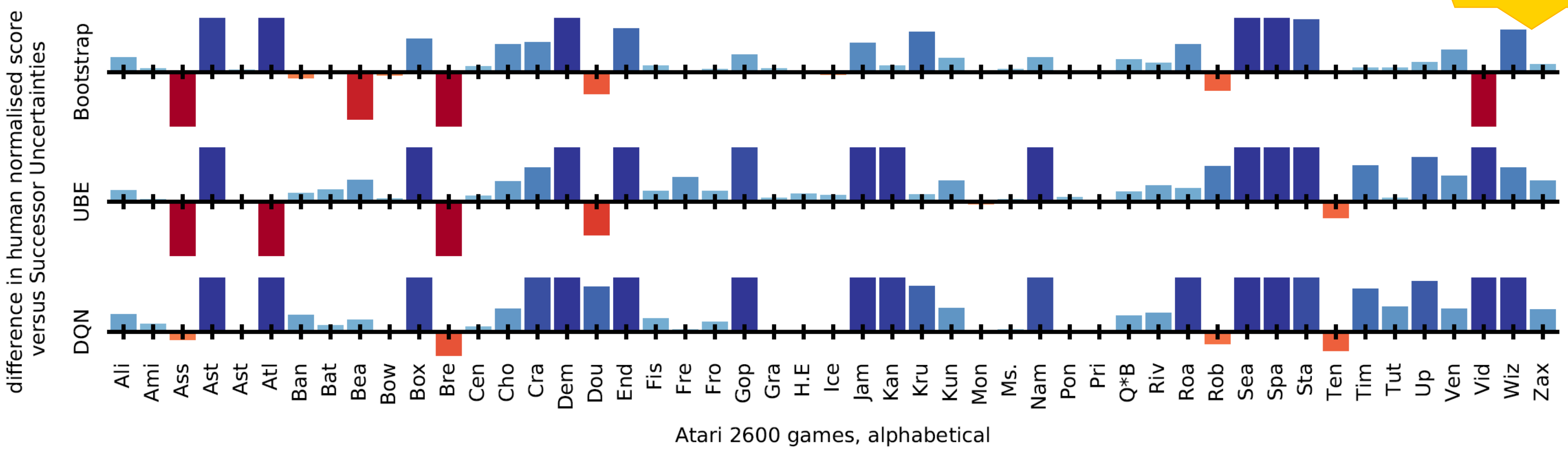


Median # episodes to solve the tree MDP (5 seeds). Blue = all (5), orange = some (1-4), red = none of the 5 runs finished within 5000 episodes. Dashed line for uniform policy. Note the varying x-axis scale!

[Janz*, Hron*, Mazur, **Hofmann**, Hernández-Lobato, Tschiatschek, NeurIPS 2019]

Successor Uncertainties

Tue 10:45a
#198



Bars show the difference in human normalised score between SU and BootDQN (top), UBE (middle) and DQN (bottom) for each of the 49 Atari 2600 games. Blue indicates SU performed better, red worse. SU outperforms the baselines on $36/49$, $43/49$ and $42/49$ games respectively. Y-axis clipped to $[-2.5, 2.5]$.

[Janz*, Hron*, Mazur, Hofmann, Hernández-Lobato, Tschiatschek, NeurIPS 2019]

4. Policy Gradient and Actor Critic Approaches

Policy Gradient Algorithm: REINFORCE

```
For each episode:
  Generate τ = s_0, a_0 r_1, s_1, a_1, r_1 … s_{t-1}, a_{t-1}, r_t by following π_θ(a|s)
  for each step i = 0 … t − 1:
    R_i = ∑_{k=i}^{t} γ^{t−k} r_k
    Â_i = R_i − b
    θ = θ + α ∇_θ log π_θ(a|s_i) Â_i
```

For each episode:

Generate $\tau = s_0, a_0 r_1, s_1, a_1, r_1 \dots s_{t-1}, a_{t-1}, r_t$ by following $\pi_\theta(a|s)$

for each step $\mathrm{i} = 0 \dots t-1$:

$R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k$

$\hat{A}_i = R_i - b$

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a|s_i)\, \hat{A}_i$

[Williams 1992]

Policy Gradient Algorithm: REINFORCE

```
For each episode:
```
  Generate $\tau = s_0, a_0 r_1, s_1, a_1, r_1 \dots s_{t-1}, a_{t-1}, r_t$ by following $\pi_\theta(a|s)$

  for each step $\mathrm{i} = 0 \dots t-1$:

    $R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k$

    $\hat{A}_i = R_i - b$

    $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a|s_i)\, \hat{A}_i$

Policy parameterized by learnable θ

[Williams 1992]

Policy Gradient Algorithm: REINFORCE

```
For each episode:
  Generate τ = s_0, a_0 r_1, s_1, a_1, r_1 … s_{t-1}, a_{t-1}, r_t by following π_θ(a|s)
  for each step i = 0 … t − 1:
    R_i = Σ_{k=i}^{t} γ^{t-k} r_k
    Â_i = R_i − b
    θ = θ + α ∇_θ log π_θ(a|s_i) Â_i
```

$R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k \longrightarrow$ Unbiased estimate of remaining episode return under π_θ starting from i

[Williams 1992]

Policy Gradient Algorithm: REINFORCE

For each episode:

Generate $\tau = s_0, a_0 r_1, s_1, a_1, r_1 \dots s_{t-1}, a_{t-1}, r_t$ by following $\pi_\theta(a|s)$

for each step $\mathrm{i} = 0 \dots t-1$:

$R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k$

$\hat{A}_i = R_i - b$ ⟶ Subtract baseline b to lower variance, e.g., episode return $\mathrm{R} = \sum_1^t r_t$ (intuition: advantage)

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a|s_i) \hat{A}_i$

[Williams 1992]

Policy Gradient Algorithm: REINFORCE

```
For each episode:
  Generate τ = s_0, a_0 r_1, s_1, a_1, r_1 ... s_{t-1}, a_{t-1}, r_t by following π_θ(a|s)
  for each step i = 0 ... t − 1:
```

$R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k$

$\hat{A}_i = R_i - b$

$\theta = \theta + \alpha \, \nabla_\theta \log \pi_\theta(a|s_i) \, \hat{A}_i$

Gradient with respect to policy parameters estimated from samples

[Williams 1992]

Policy Gradient Algorithm: REINFORCE

```
For each episode:
  Generate τ = s_0, a_0 r_1, s_1, a_1, r_1 … s_{t-1}, a_{t-1}, r_t by following π_θ(a|s)
  for each step i = 0 … t-1:
```

$R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k$

$\hat{A}_i = R_i - b$

$\theta = \theta + \alpha \underbrace{\nabla_\theta \log \pi_\theta(a|s_i)}_{\hat{g}} \hat{A}_i$

Objective: $J(\theta) = \sum_\tau P_\theta(\tau) R(\tau)$

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_\tau P_\theta(\tau) R(\tau)$$

$$= \sum_\tau \nabla_\theta P_\theta(\tau) R(\tau)$$

$\hat{g}$ is an unbiased estimate: Policy gradient theorem [Sutton et al. 2000]

[Williams 1992]

Policy Gradient Algorithm: REINFORCE

```
For each episode:
```

Generate $\tau = s_0, a_0 r_1, s_1, a_1, r_1 \ldots s_{t-1}, a_{t-1}, r_t$ by following $\pi_\theta(a|s)$

for each step $\mathrm{i} = 0 \ldots t-1$:

$R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k$

$\hat{A}_i = R_i - b$

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a|s_i)\, \hat{A}_i$ ⟶ Actor-critic approaches use learned estimate (e.g., $\hat{A}(s,a) = \hat{Q}(s,a) - \hat{V}(s)$)

Policy Gradient Algorithm: REINFORCE

For each episode:

Generate $\tau = s_0, a_0 r_1, s_1, a_1, r_1 \ldots s_{t-1}, a_{t-1}, r_t$ by following $\pi_\theta(a|s)$

for each step $\mathrm{i} = 0 \ldots t - 1$:

$R_i = \sum_{k=i}^{t} \gamma^{t-k} r_k$

$\hat{A}_i = R_i - b$

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(a|s_i)\, \hat{A}_i$

NeurIPS 2016 Tutorial by Pieter Abbeel John Schulman: Deep Reinforcement Learning through Policy Optimization (https://media.nips.cc/Conferences/2016/Slides/6198-Slides.pdf)

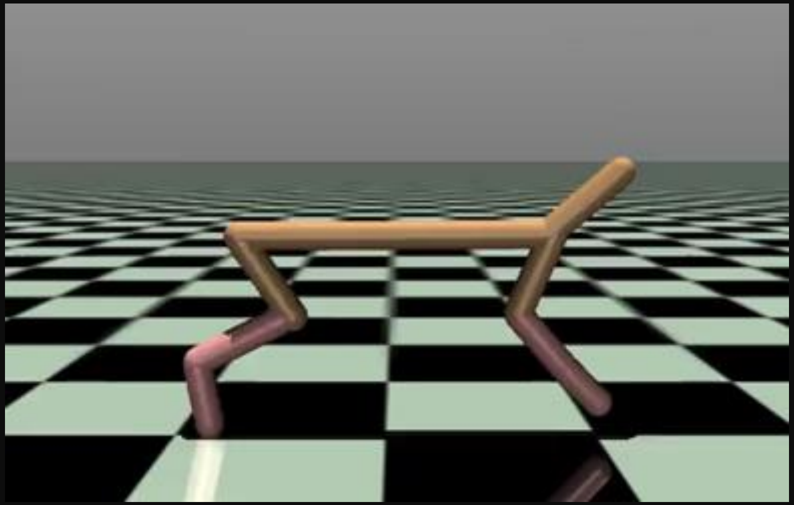
Actor-Critic with Deep Function Approximation

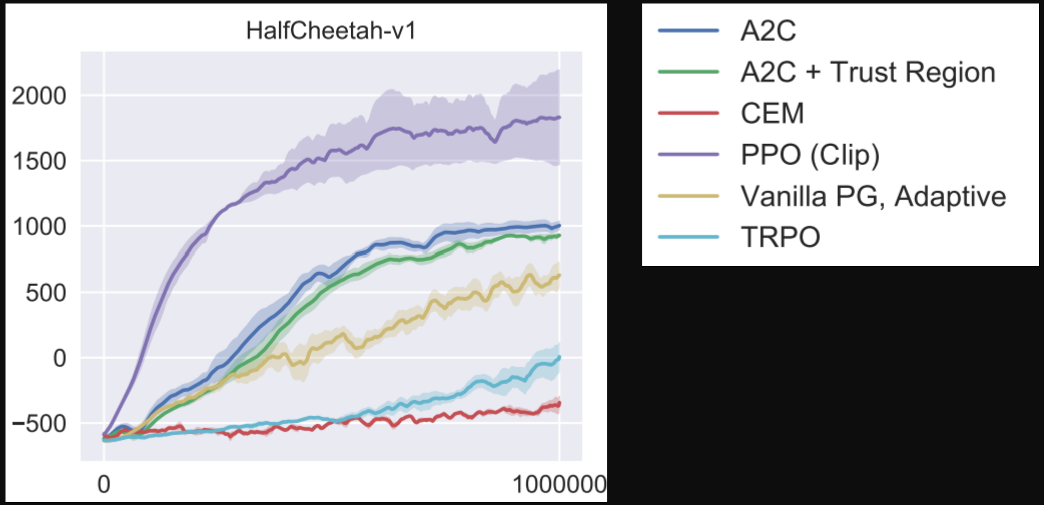
Need to balance between learning speed, stability

[Kakade & Langford 2002] Conservative Policy Iteration (CPI): propose surrogate objective, guarantee monotonic improvement under specific state distribution

[Schulman et al. 2015] Trust Region Policy Optimization (TRPO): approximates CPI with trust region constraint

[Schulman et al. 2017] Proximal Policy Optimization (PPO): replace TRPO constraint with KL penalty + clipping (computationally efficient)





Actor-Critic with Deep Function Approximation

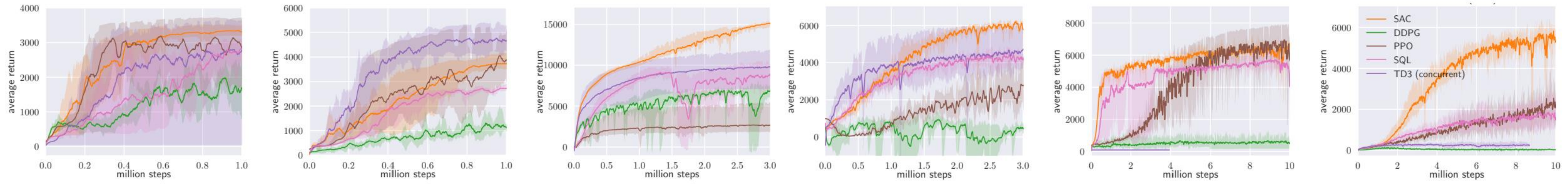
Need to balance between learning speed, stability

[Kakade & Langford 2002] Conservative Policy Iteration (CPI): propose surrogate objective, guarantee monotonic improvement under specific state distribution

[Schulman et al. 2015] Trust Region Policy Optimization (TRPO): approximates CPI with trust region constraint

[Schulman et al. 2017] Proximal Policy Optimization (PPO): replace TRPO constraint with KL penalty + clipping (computationally efficient)

[Haarnoja et al. 2018] Soft Actor-Critic (SAC): stabilize learning by jointly maximizing expected reward and policy entropy (based on maximum entropy RL [Ziebart et al. 2008])



(a) Hopper-v1 (b) Walker2d-v1 (c) HalfCheetah-v1 (d) Ant-v1 (e) Humanoid-v1 (f) Humanoid (rllab)

Optimistic Actor Critic (OAC)

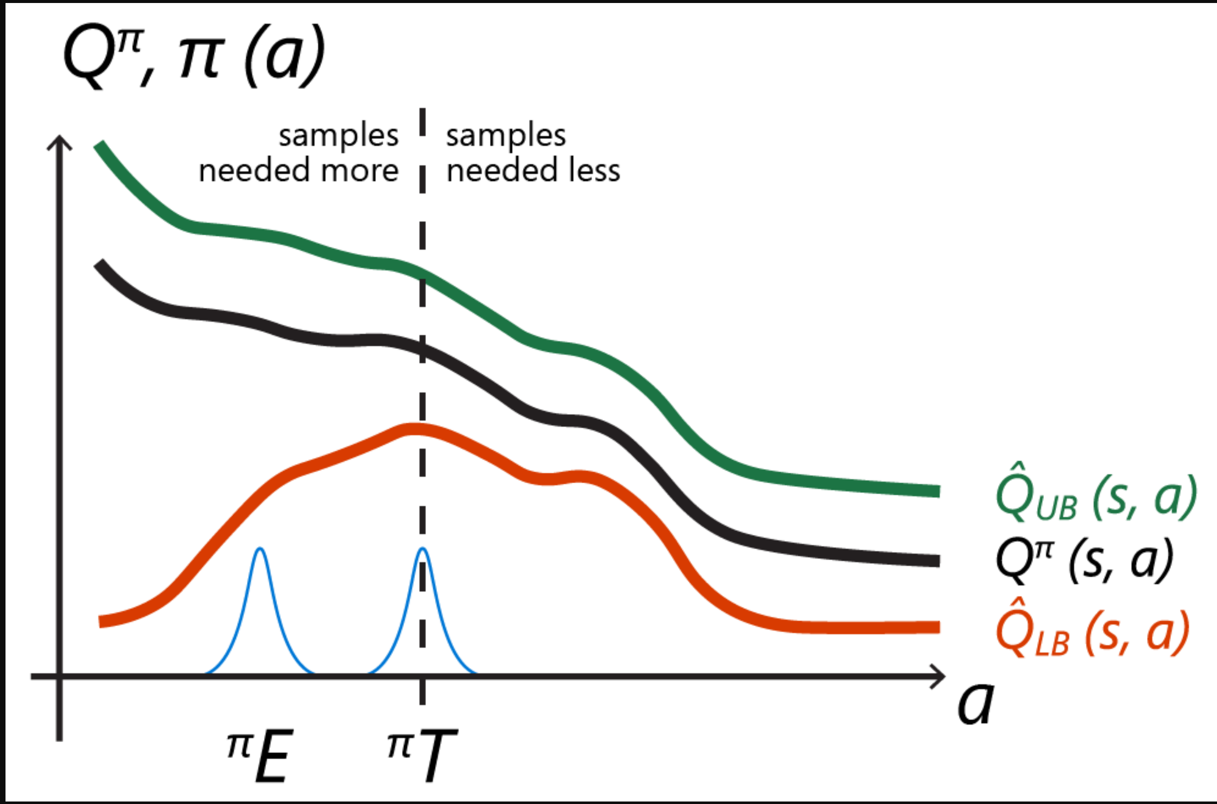
Focus on exploration in deep Actor Critic approaches

Insight: existing approaches tend to explore conservatively

Solution: more principled exploration using optimism

Upper confidence bound (optimistic estimate) on $\hat{Q}$:

$$\hat{Q}_{UB}(x,a) = \underbrace{\mu_Q(x,a)}_{\text{mean belief about } \hat{Q}} + \underbrace{\beta_{UB}\sigma_Q(x,a)}_{\text{uncertainty about } \hat{Q}}$$



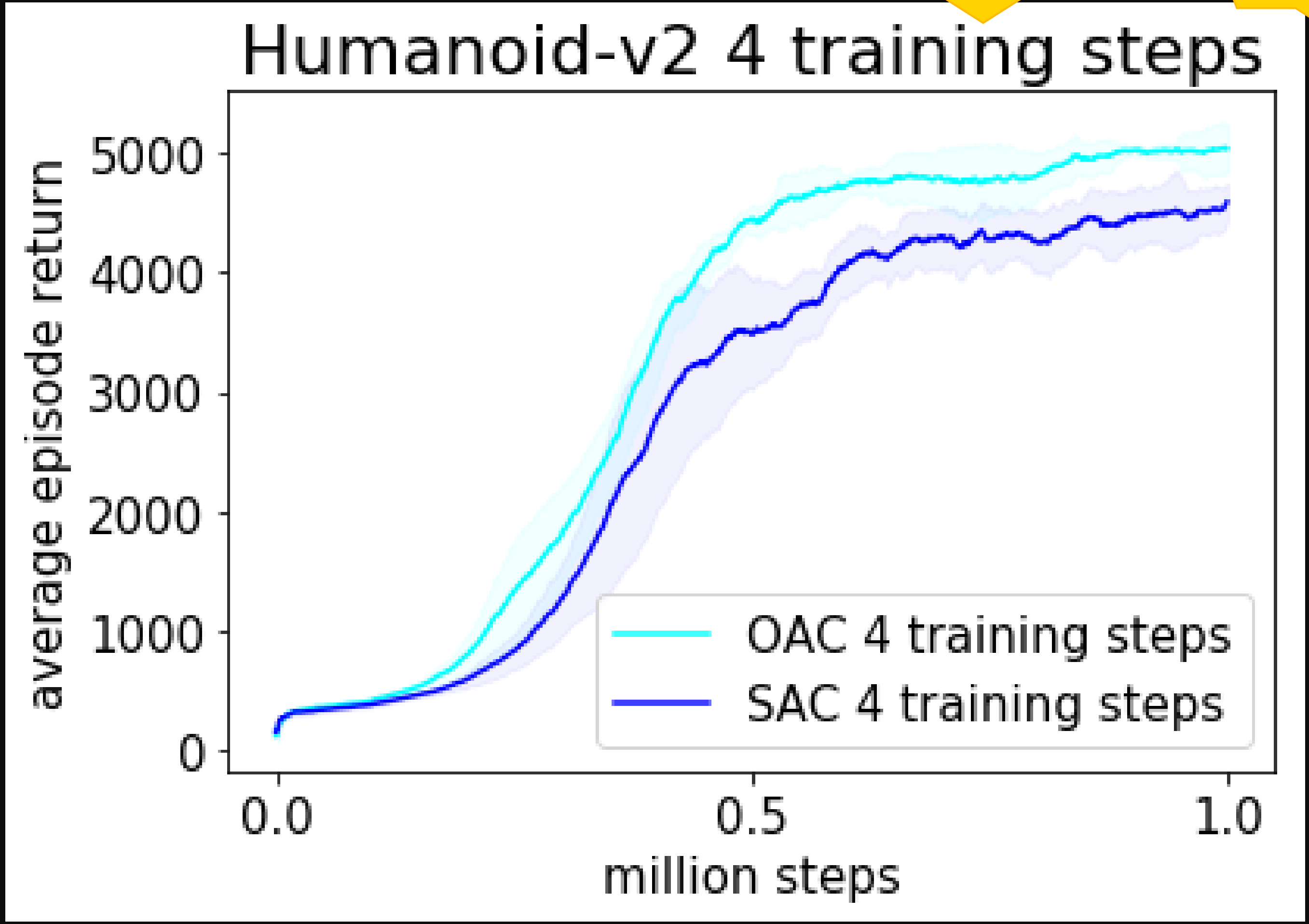
[Kamil Ciosek, Vuong, Loftin, Hofmann 2019]

Optimistic Actor Critic (OAC)

Tue
spotlight
5:05PM
T3-S2

Tue
5:30PM
#179

Key result: Optimistic exploration leads to efficient, stable learning in modern Actor Critic methods



[Kamil Ciosek, Vuong, Loftin, Hofmann 2019]

RL Applications

Example: Personalizer

Further study: ICML 2017 tutorial on Real World Interactive Learning by Alekh Agarwal and John Langford http://hunch.net/~rwil/

Example: Robotics

Further study: ICML 2017 tutorial on Deep Reinforcement Learning, Decision Making, and Control by Chelsea Finn and Sergey Levine https://sites.google.com/view/icml17deeprl

Example: Tutoring systems

Further study: NeurIPS 2017 tutorial on Reinforcement Learning for the People and/or by the People https://cs.stanford.edu/people/ebrun/NIPS_2017_tutorial_brunskill.pdf

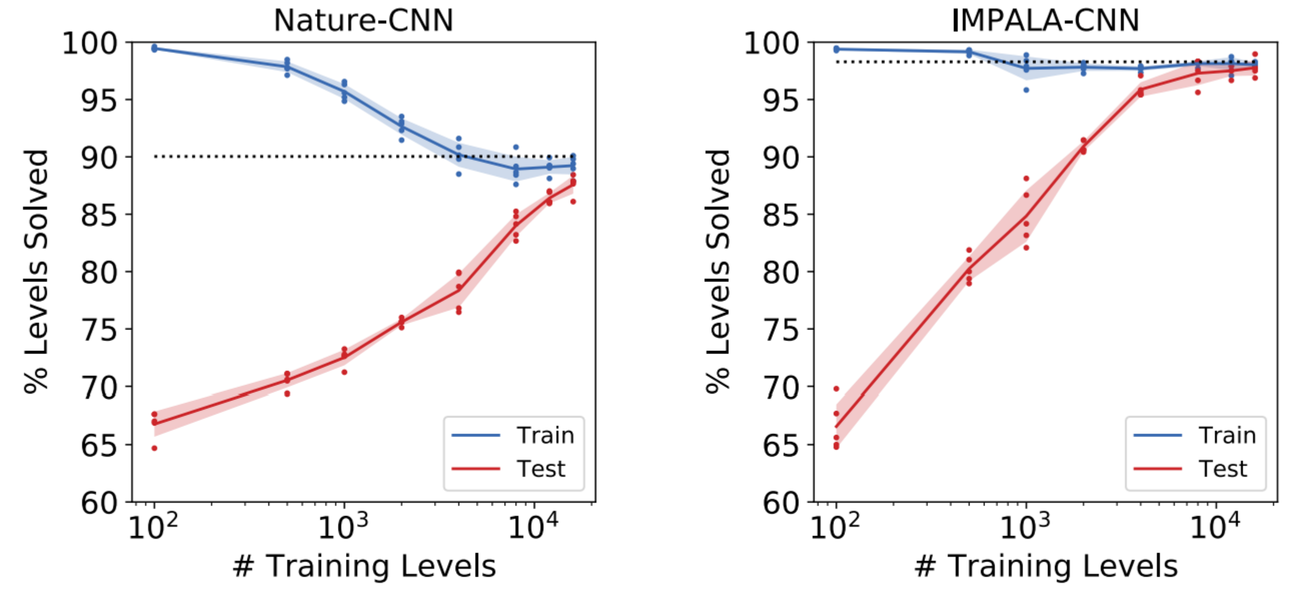
5. Generalization

Generalization in RL

Example: generalization using successor features [Dayan 1993], rapidly adapt to new reward structure [Barreto et al. 2018]

How many tasks are needed before modern approaches generalize?

[Cobbe et al. 2019]



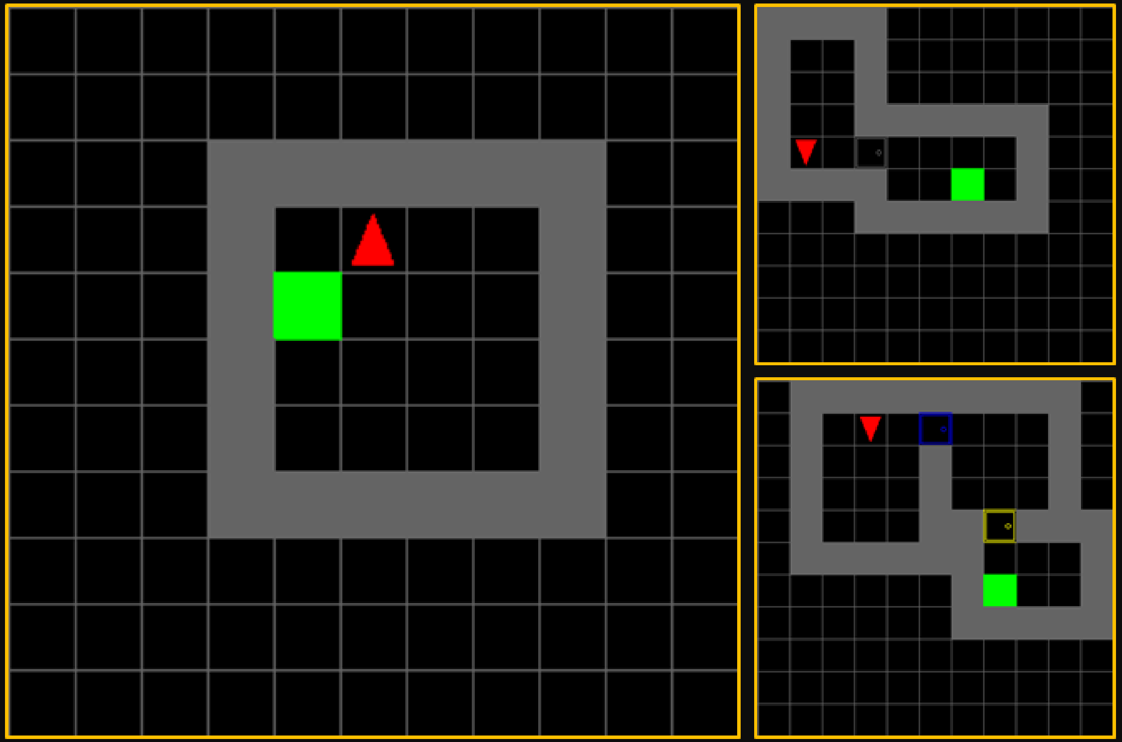
(a) Final train and test performance of Nature-CNN agents after 256M timesteps, as a function of the number of training levels.

(b) Final train and test performance of IMPALA-CNN agents after 256M timesteps, as a function of number of training levels.

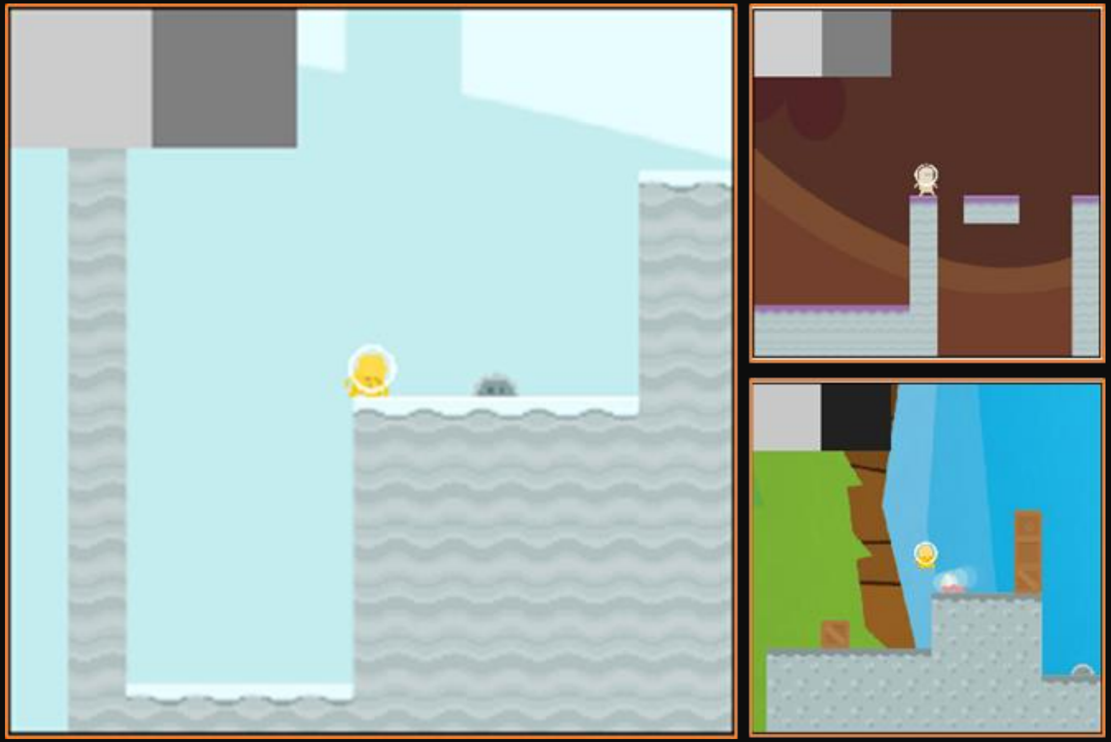
Figure 2. Dotted lines denote final mean test performance of the agents trained with an unbounded set of levels. The solid line and shaded regions represent the mean and standard deviation respectively across 5 seeds. Training sets are generated separately for each seed.

Generalization in RL

Recently proposed benchmarks:



Multi-Room
Chevalier-Boisvert et al. (2018)



CoinRun
Cobbe et al. (2019)

Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottleneck

Previous regularization approaches developed for supervised learning, not RL!

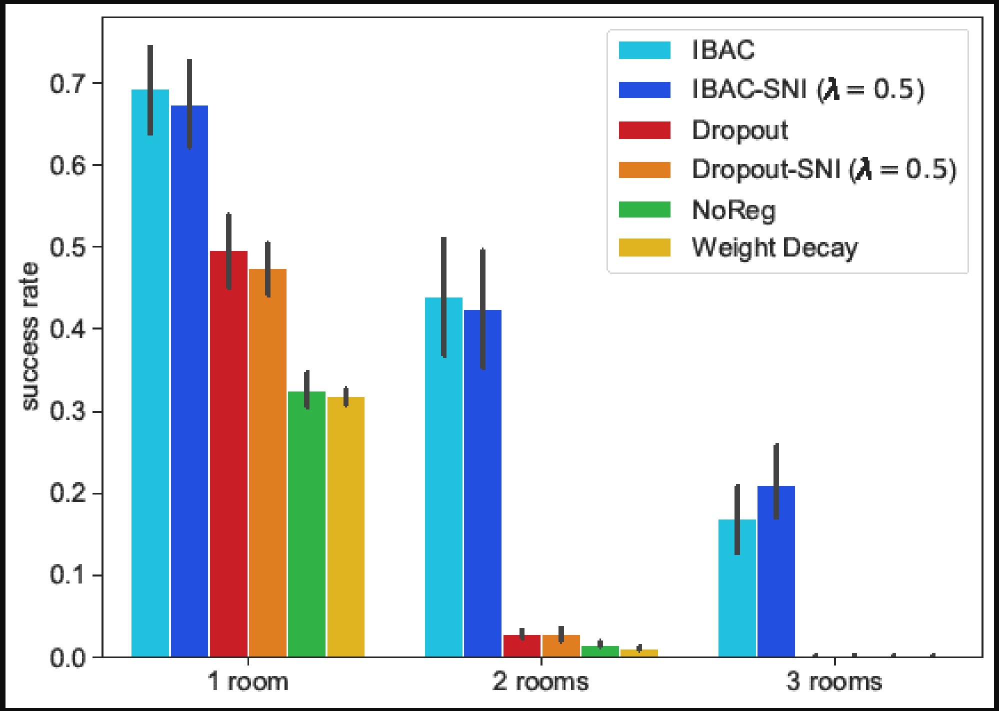
Insight 1: Selective noise injection for gradient update but not behavior (rollout) policy speeds learning

Insight 2: regularization with Information bottleneck is particularly effective

$$\nabla_\theta J(\pi_\theta) = \hat{\mathbb{E}}_{\pi_\theta^r(a_t|x_t)}\left[\sum_t^T \frac{\pi_\theta(a_t|x_t)}{\pi_\theta^r(a_t|x_t)} \nabla_\theta \log \pi_\theta(a_t|x_t)\hat{A}_t\right]$$

[Igl, Ciosek, Li, Tschiatschek, Zhang, Devlin, Hofmann 2019]

Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottleneck

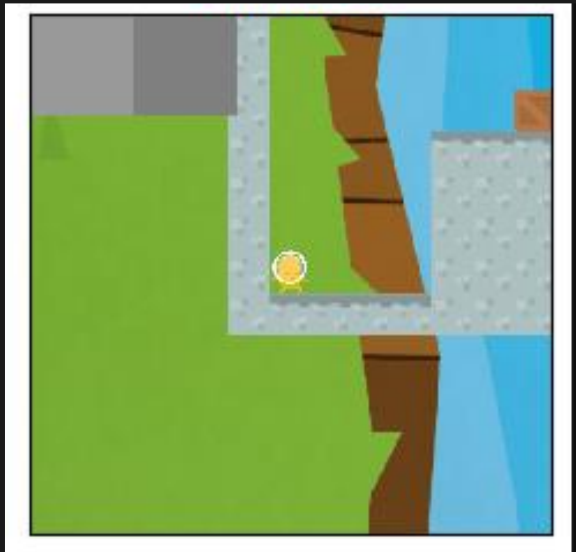


Key result: Dramatically improve performance on generalization benchmarks

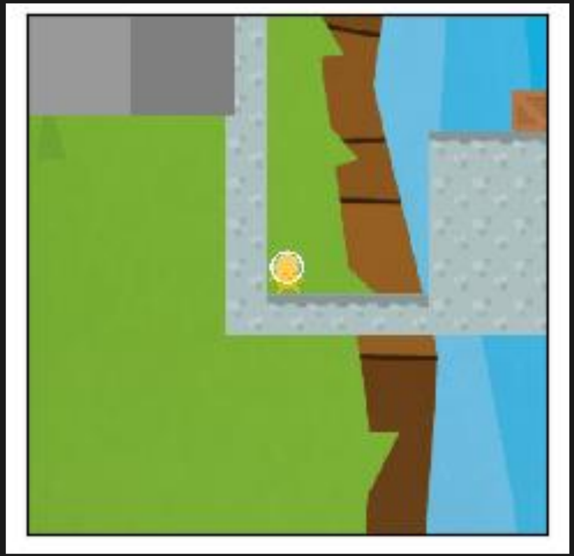
[Igl, Ciosek, Li, Tschiatschek, Zhang, Devlin, Hofmann 2019]

Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottlene

Thu
10:45AM
#228



Baseline BatchNorm regularizer



Our IBAC-SNI approach

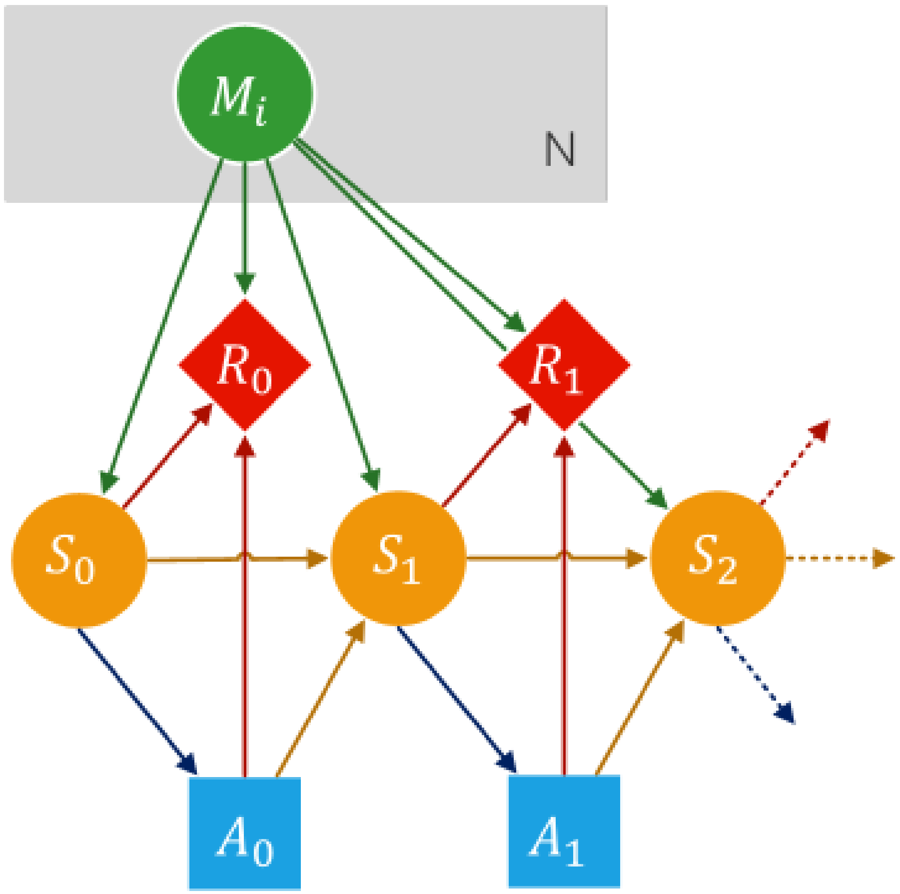
Sat Dec 14th 8:00AM – 6:00PM
@ West 211 - 214
Learning Transferable Skills
Marwan Mattar · Arthur Juliani
Danny Lange · Matthew Crosby
Benjamin Beyret
https://www.skillsworkshop.ai/

[Igl, Ciosek, Li, Tschiatschek, Zhang, Devlin, Hofmann 2019]

6. Structure

Meta Learning

= Learn to Learn, e.g., learn an update rule from related tasks



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 $\pi(s|a)$

Example, tasks are related through low-dimensional embedding

Model-Agnostic Meta Learning (MAML) [Finn et al. 2017]

Flexible meta-learning approach based on 2nd order 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:

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

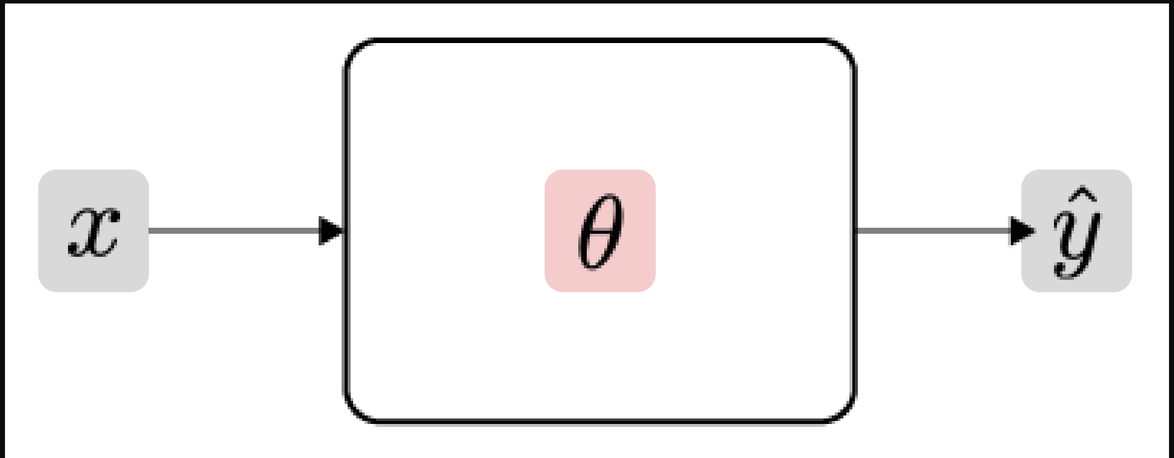
For more on Meta-Learning see ICML 2019 tutorial by Chelsea Finn and Sergey Levine https://sites.google.com/view/icml19metalearning

Fast Context Adaptation via Meta-Learning (CAVIA)

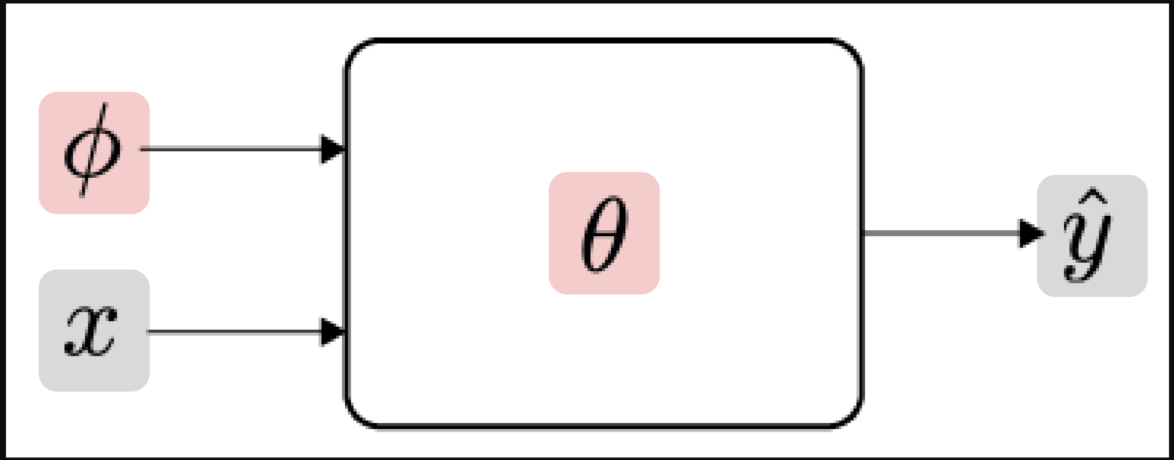
Problem: Many parameters + few data points can lead to overfitting

Key insight: Many tasks only require task identification – no need to update all model parameters at test time

MAML (Finn et al. 2017)

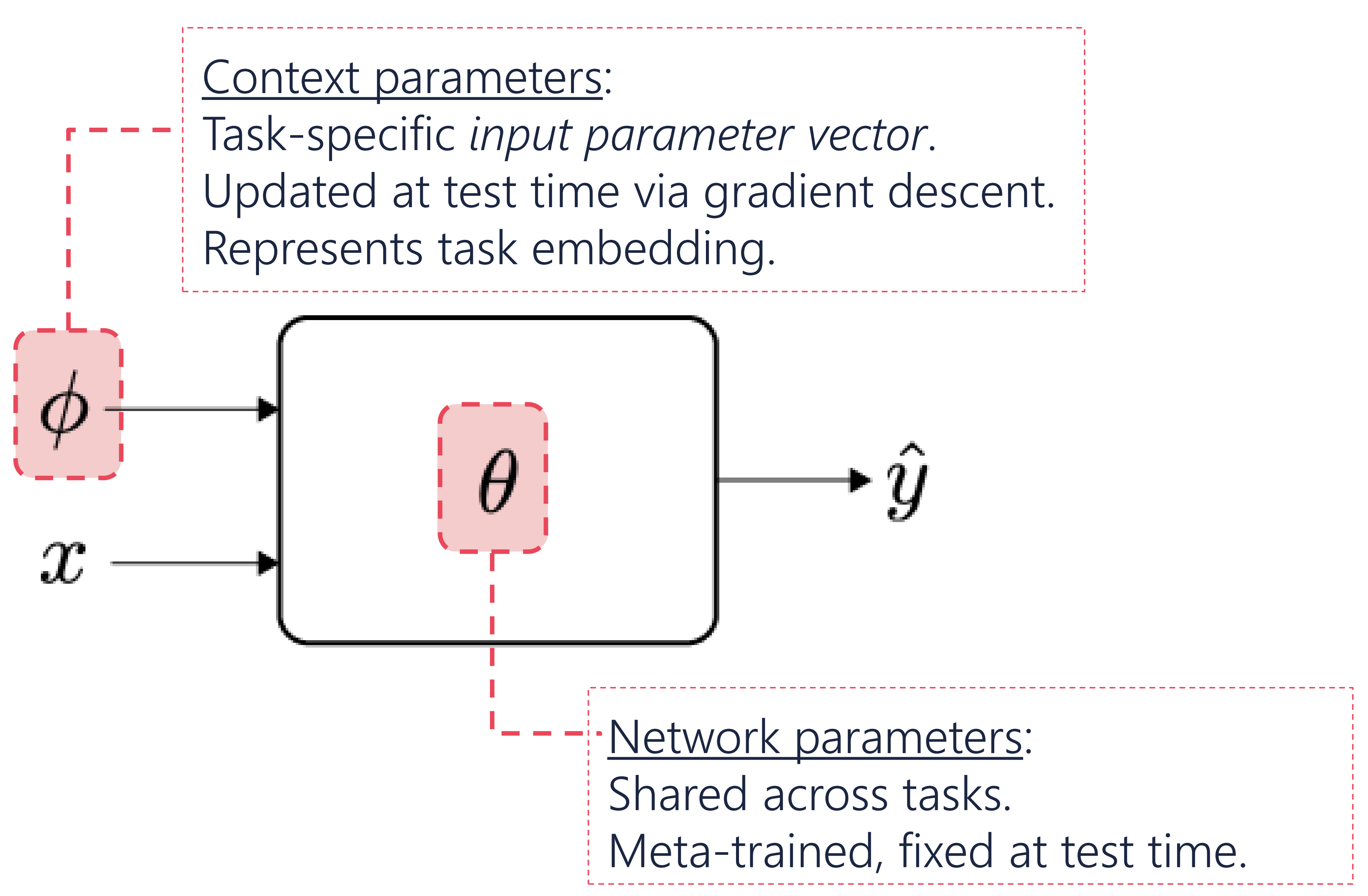


CAVIA



[Zintgraf, Shiarli, Kurin, Hofmann & Shimon Whiteson, 2019]

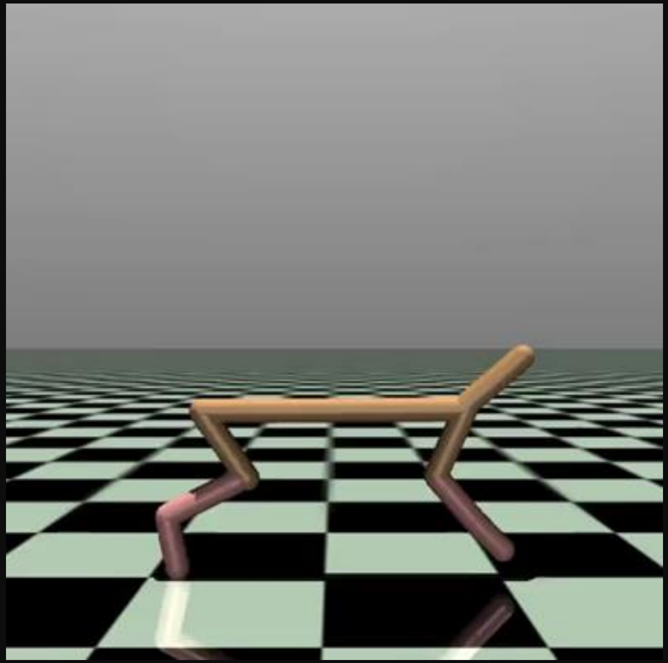
Overview

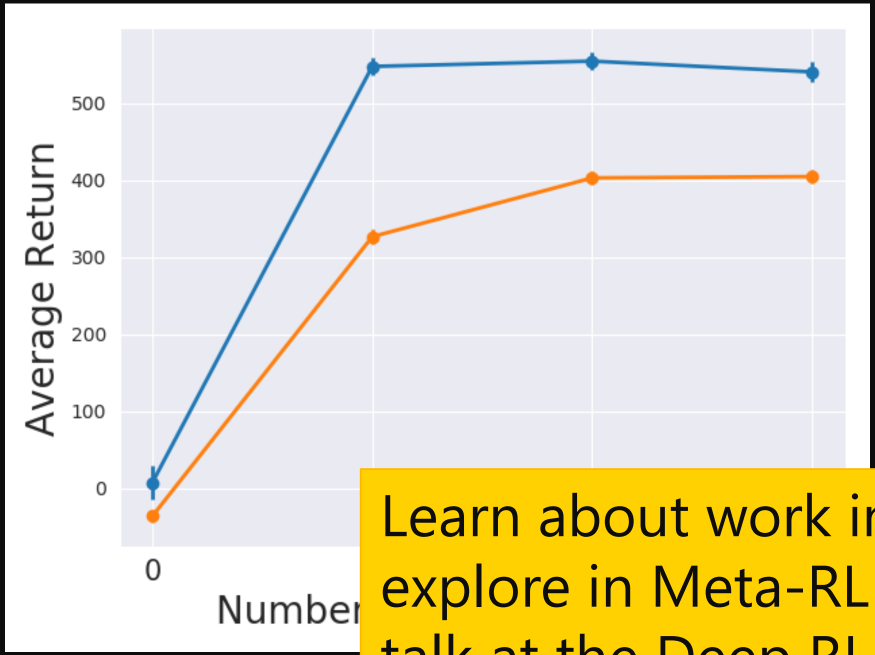


[Zintgraf, Shiarli, Kurin, Hofmann & Shimon Whiteson, 2019]

Fast Context Adaptation via Meta-Learning (CAVIA)

Results: Half-Cheetah directions task





CAVIA is less prone to overfitting

Learn about work in progress: learning to explore in Meta-RL settings – Shimon's invited talk at the Deep RL workshop on Sat, 10AM
https://sites.google.com/view/deep-rl-workshop-neurips-2019/home

[Zintgraf, Shiarli, Kurin, Hofmann & Shimon Whiteson, 2019]

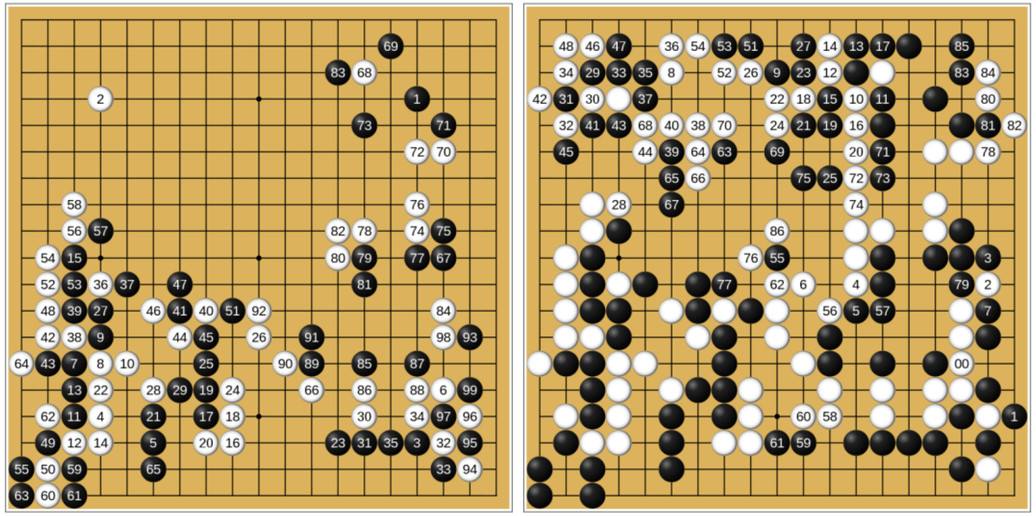
7. Models

Model-based RL

Model: Dynamics: $\mathrm{T}(s_{t+1}|s_t, a_t)$, Reward: $R(r_{t+1}|s_t, a_t)$

[Silver et al. 2016] – AlphaGo: Model is fully known

AlphaGo Master (white) v. Tang Weixing (31 December 2016), AlphaGo won by resignation. White 36 was widely praised.



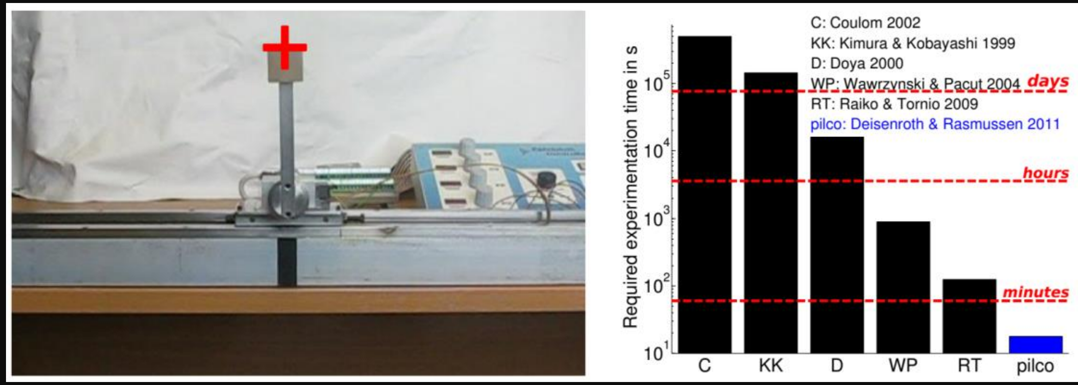
First 99 moves

Moves 100–186 (149 at 131, 150 at 130)

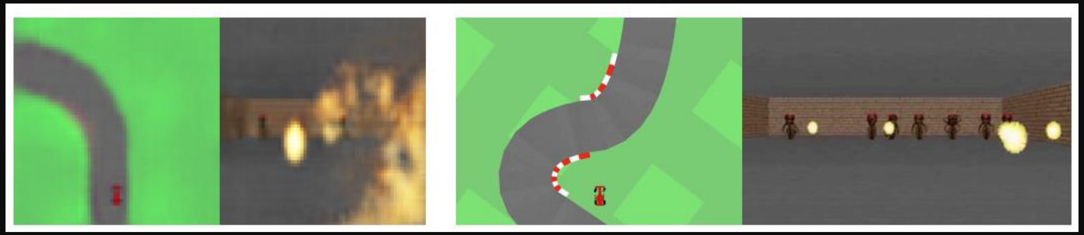
Image credit: https://en.wikipedia.org/wiki/AlphaGo

Model-based RL

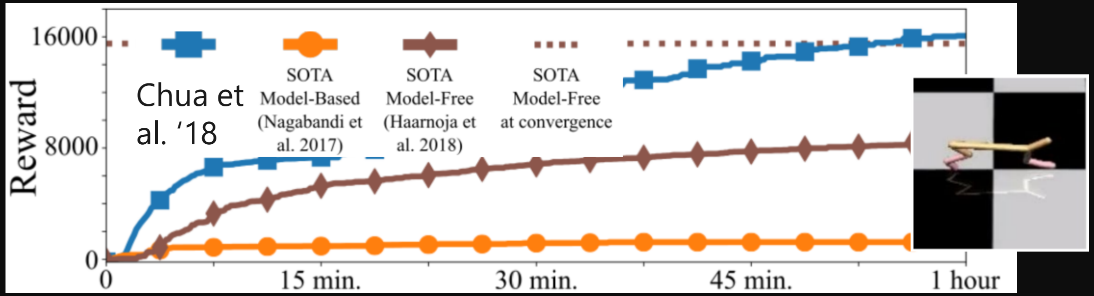
What if we don't know the model – learn from data?



[Deisenroth & Rasmussen 2011] – PILCO – learns model parameterized as Gaussian Process



[Ha & Schmidhuber 2018] – World Models – learn models for policy optimization in visual domains



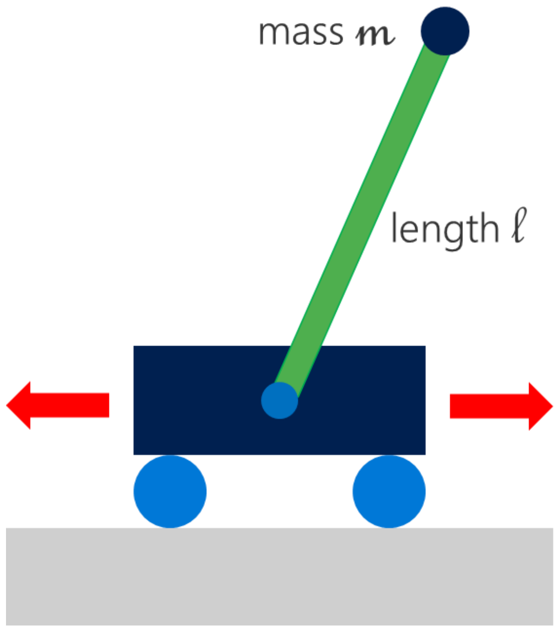
[Chua et al. 2018] – Learn flexible models that quantify uncertainty using ensembles of Bayesian NNs

[Sun et al. 2019] Identify settings where model-based RL provably faster than model-free approaches

Meta-Learning for Model Identification

Goal: use data from related tasks to rapidly adapt model to new task

Approach: Gaussian Process dynamics conditioned on NN latent variable (optimized jointly)



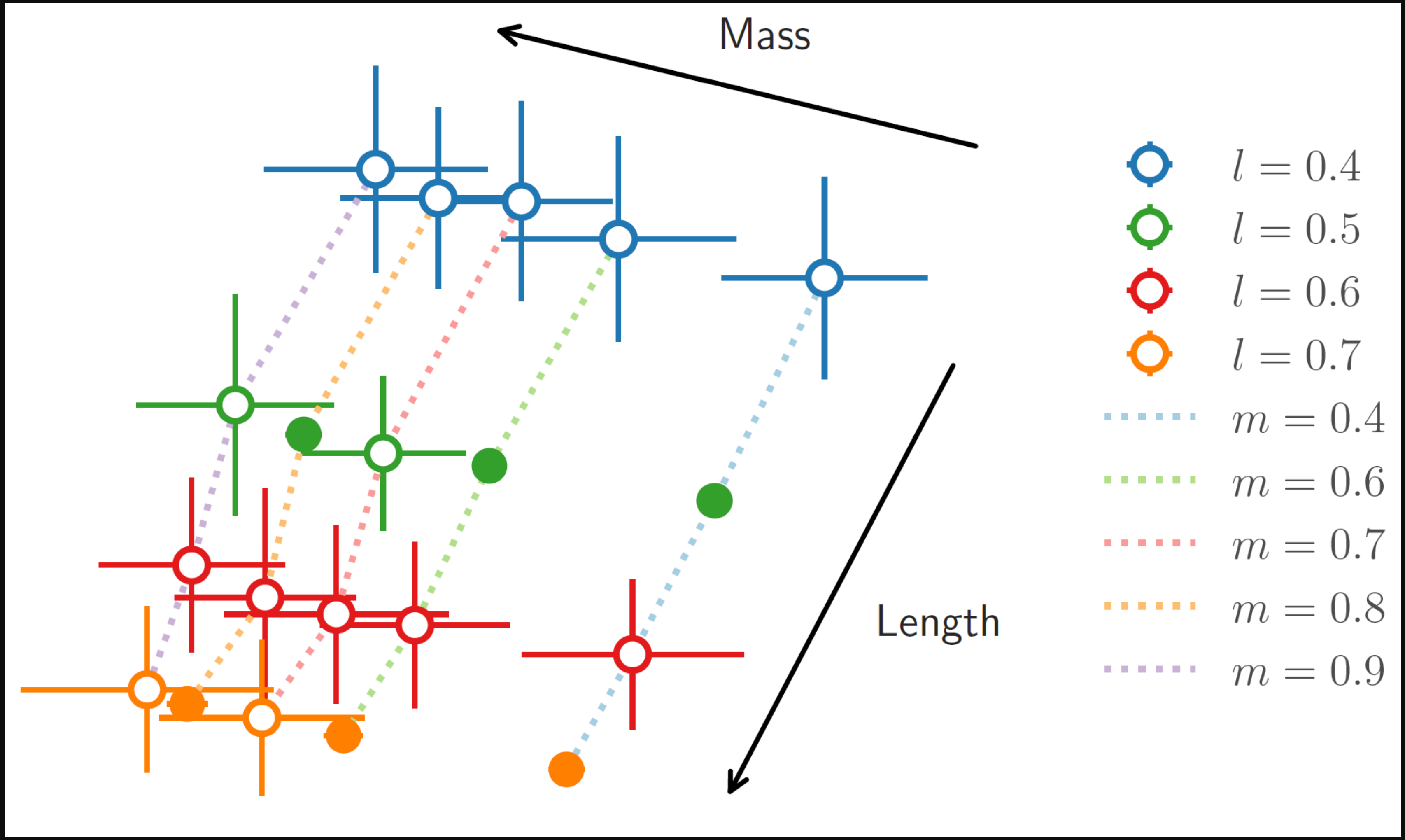
Systems vary in mass m and pendulum length l

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

14 held out test tasks require interpolation + extrapolation

[Sæmundsson, Hofmann & Deisenroth, 2018]

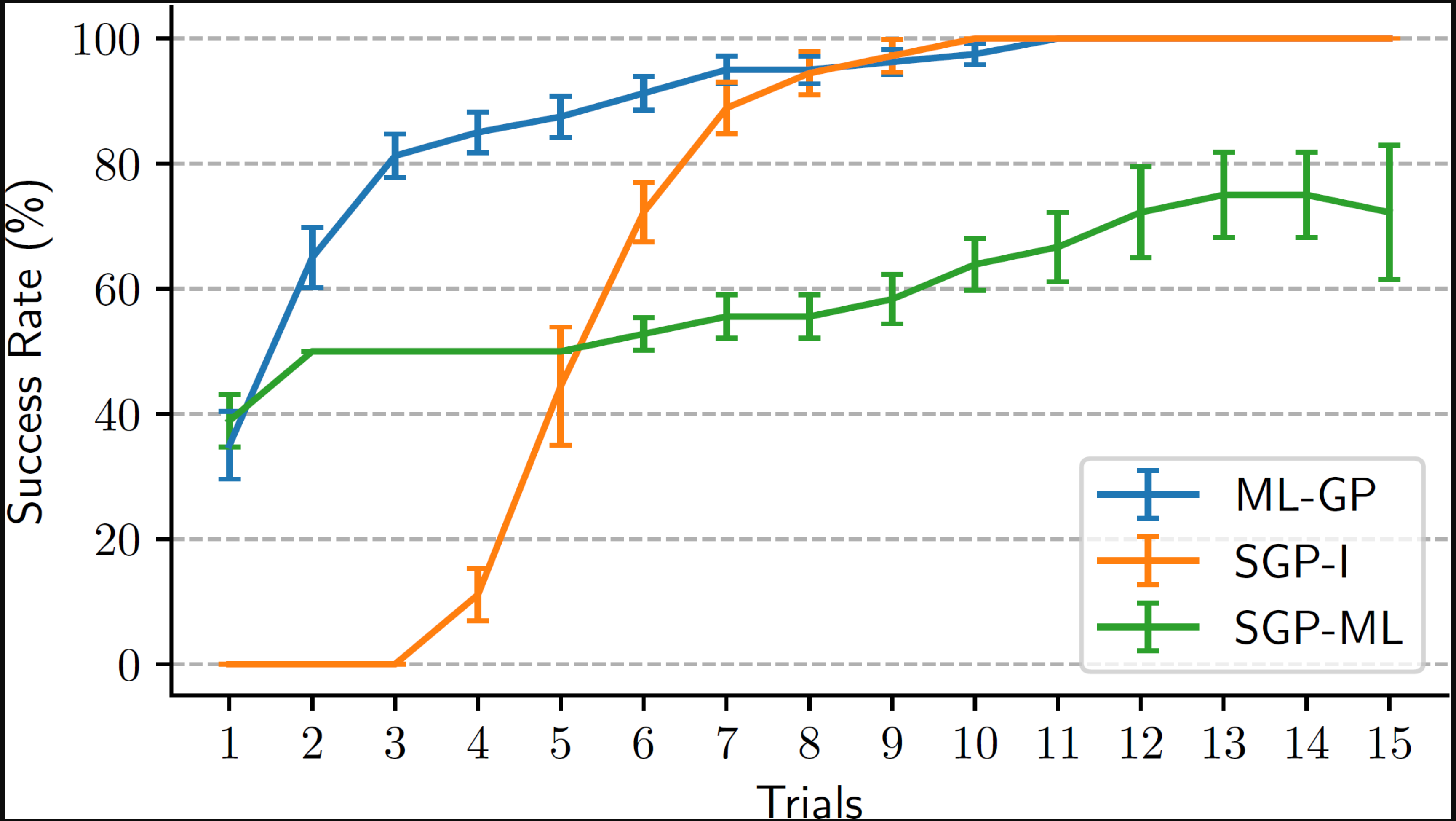
Multi-task Cart-Pole



Result 1: Learned embeddings accurately capture task structure

[Sæmundsson, Hofmann & Deisenroth, 2018]

Multi-task Cart-Pole



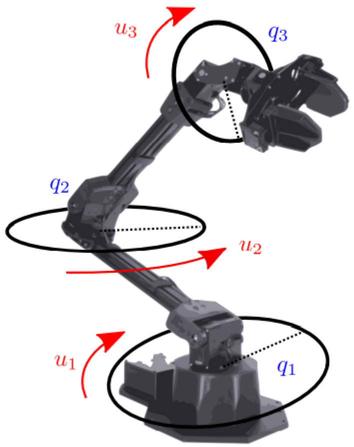
Result 2: dynamics model effectively uses multi-task structure for rapid adaptation

[Sæmundsson, Hofmann & Deisenroth, 2018]

Using more (known) structure

Structural Priors

High-level prior knowledge: e.g., laws of physics or configuration constraints



Equations of motion

$$u = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q}$$

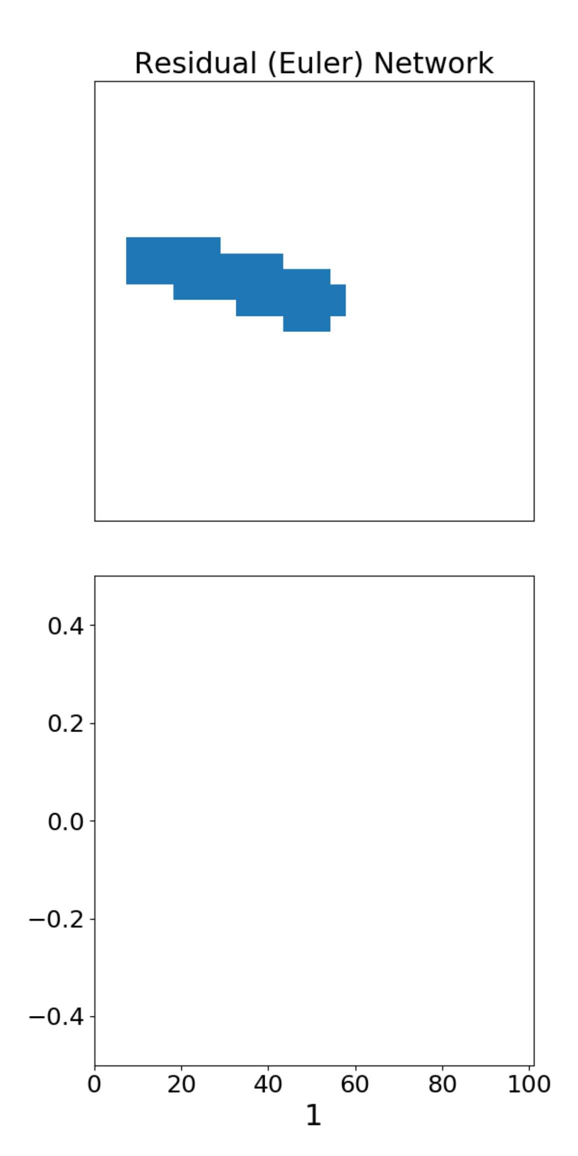
⏩ Improve data efficiency and generalization

Image credit: Marc Deisenroth

Insight: propose Variational Integrator Networks (VINs) with built-in physics and geometric structure

[Sæmundsson, Terenin, Hofmann & Deisenroth, 2019]

Using more (known) structure

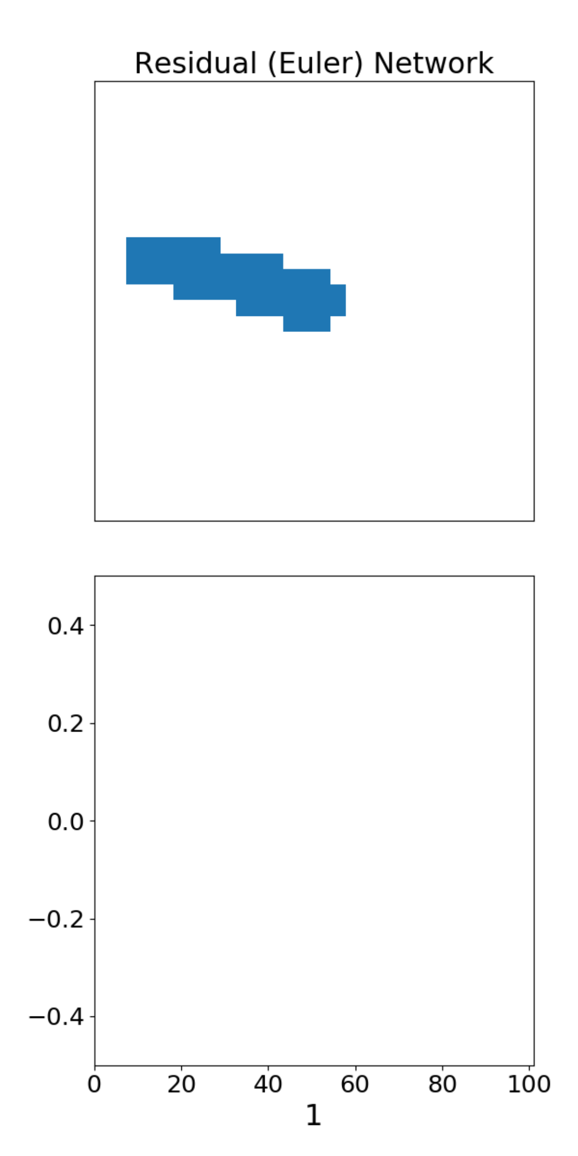


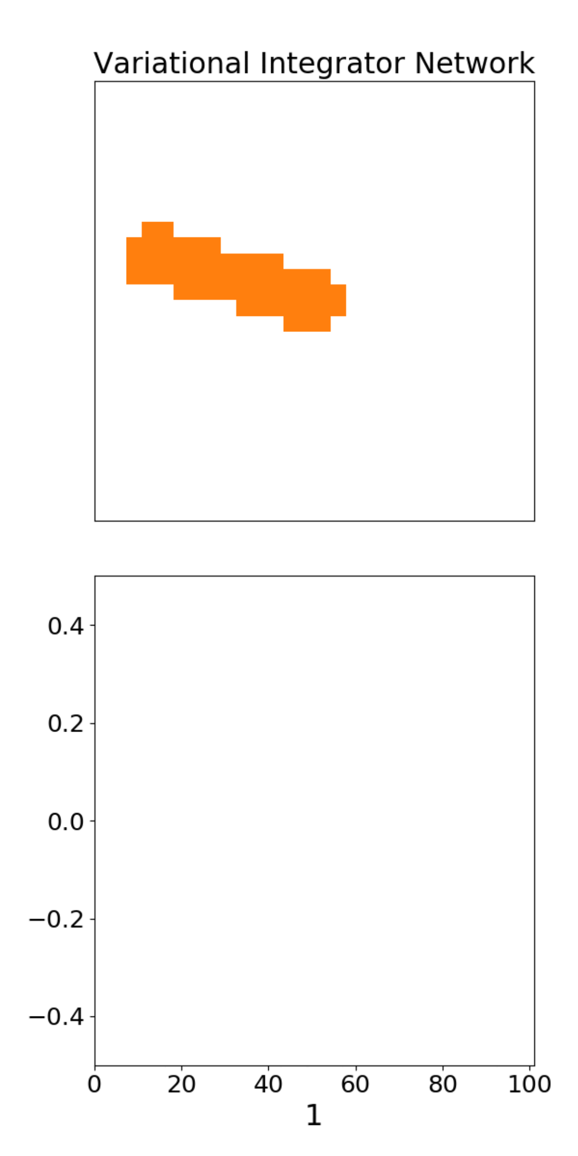
Result: VINs within auto-encoder setup effectively constrains latent space, learns from limited data.

Here: training on 40 images (28x28)

[Sæmundsson, Terenin, Hofmann & Deisenroth, 2019]

Using more (known) structure





Result: VINs within auto-encoder setup effectively constrains latent space, learns from limited data.

Here: training on 40 images (28x28)

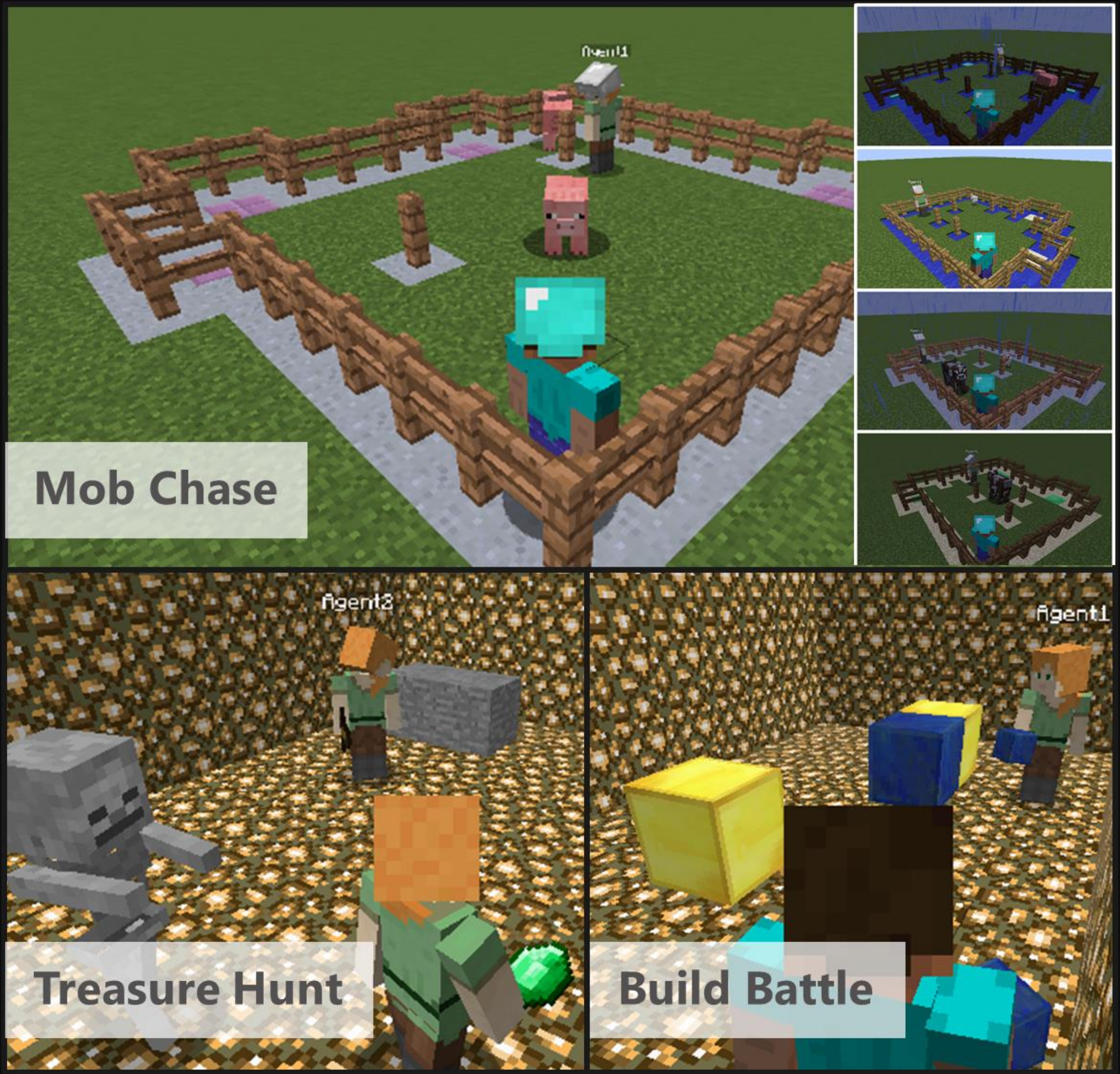
For more details see Steindor's poster at the Bayesian Deep Learning workshop: Fri 9:35AM
http://bayesiandeeplearning.org/

[Sæmundsson, Terenin, Hofmann & Deisenroth, 2019]

8. New Challenges

Multi-Agent Reinforcement Learning in Malmo (MARLO)

Agents collaborate to catch pig, chicken, or other mob in a small enclosure



The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition by Perez-Liebana et al. https://arxiv.org/abs/1901.08129

One agent collects and caries treasure to a goal, the other defends the team from attackers

Agents collaborate to build a structure, but the faster agent earns more rewards

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 (AIcrowd)
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)

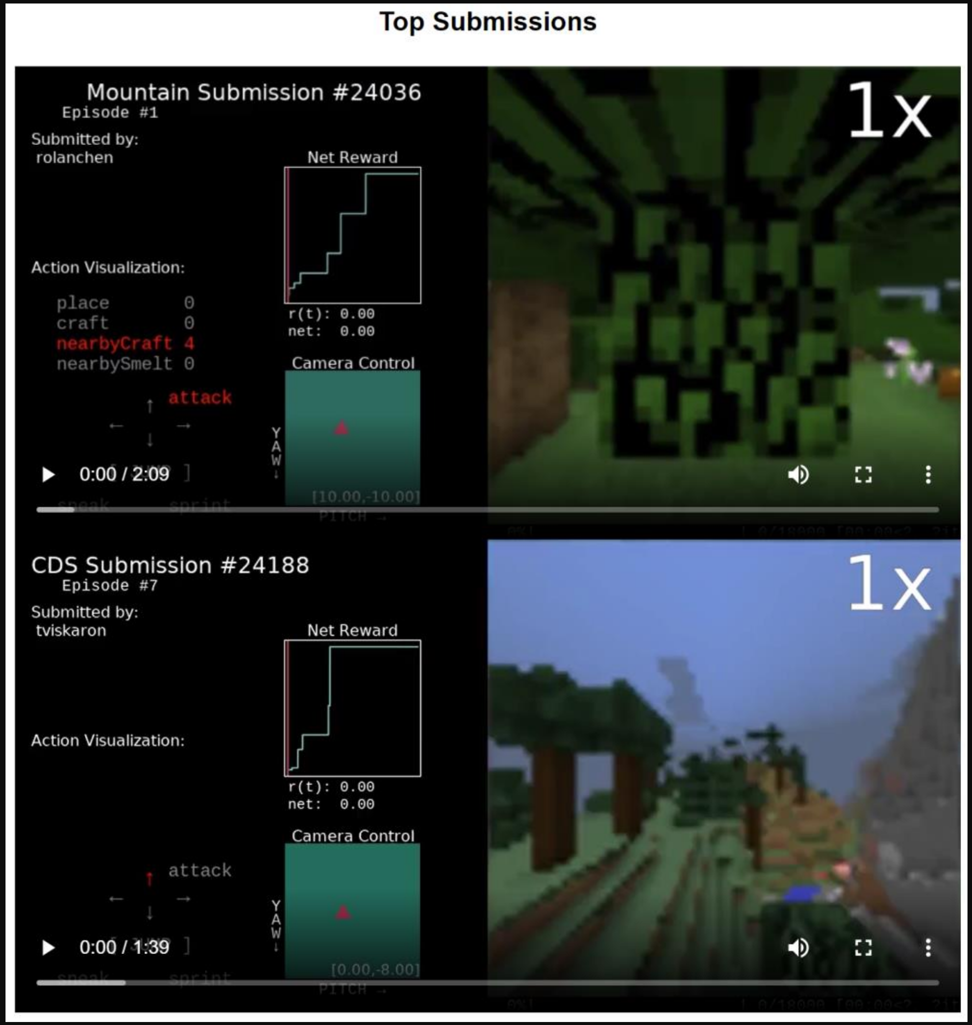
Advisory committee

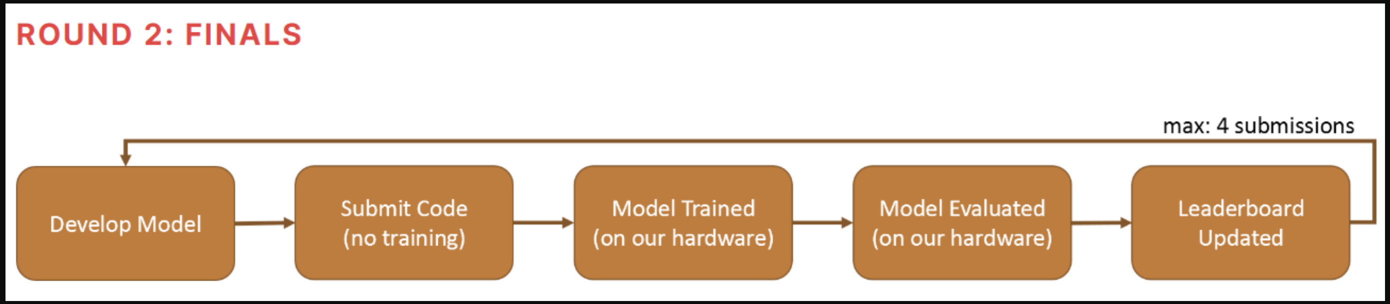
Chelsea Finn (Google Brain and UC Berkeley)
Sergey Levine (UC Berkeley)
Harm van Seijen (Microsoft Research)
Oriol Vinyals (Google DeepMind)

Click to play (in powerpoint)

Video link: https://www.microsoft.com/en-us/research/video/minerl-competition-2019/

MineRL @ NeurIPS 2019 Competition Track





Winners announced this Saturday (Competition Track Day 2): 9AM

http://minerl.io/competition/

RL@NeurIPS

Enjoy the conference!!

Microsoft

References

References and Further Study: Surveys and Textbooks

[Bosunio et al. 2010] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010.

[Kaelbling et al.] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 4: 237–285, 1996.

[Silver 2015] David Silver. UCL Course on RL, 2015.

[Sutton & Barto] Rich S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. 2nd edition. Cambridge: MIT press, 2018.

[Szepesvári 2010] Csaba Szepesvári. *Algorithms for reinforcement learning*. Synthesis lectures on artificial intelligence and machine learning 4.1 (2010): 1-103.

References and Further Study: Benchmarks & Evaluation

[Bellemare et al. 2013] M. G. Bellemare, Y. Naddaf, J. Veness and M. Bowling. *The Arcade Learning Environment: An Evaluation Platform for General Agents*. In Journal of Artificial Intelligence Research 47, pages 253-279, 2013.

[Brockman et al. 2016] Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI gym. arXiv preprint arXiv:1606.01540.

[Duan et al. 2016] Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In Proceedings of the 33rd International Conference on Machine Learning (ICML).

[Johnson et al. 2016] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell. *The Malmo platform for artificial intelligence experimentation*. In International joint conference on artificial intelligence (IJCAI), 2016.

[Machado et al. 2017] Machado, M. C.; Bellemare, M. G.; Talvitie, E.; Veness, J.; Hausknecht, M.; and Bowling, M. 2017. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. arXiv preprint arXiv:1709.06009.

[Todorov et al. 2012] Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, 2012, 5026–5033.

References 1

[Auer et al. 2002a] P. Auer, N. Cesa-Bianchi, P. Fischer: *Finite-time analysis of the Multiarmed Bandit Problem*. Machine Learning 47, 2002a.

[Baird 1995] Leemon Baird. *Residual algorithms: Reinforcement learning with function approximation*. In Machine Learning Proceedings 1995, pp. 30-37. Morgan Kaufmann, 1995.

Barreto et al. 2018] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. *Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement*. In International Conference on Machine Learning, pp. 510-519. 2018.

[Bellman 1953] Richard Bellman. *An introduction to the theory of dynamic programming*. Vol. 245. RAND CORP, 1953.

[Bellman 1954] Richard Bellman. *The theory of dynamic programming*. Bulletin of the American Mathematical Society 60, no. 6 (1954): 503-515.

[Bellman 1957] Richard Bellman. *A Markovian decision process*. Journal of mathematics and mechanics (1957): 679-684.

References 2

[Ciosek et al. 2019] Ciosek, Kamil, Quan Vuong, Robert Loftin, and Katja Hofmann. "Better Exploration with Optimistic Actor Critic." In Advances in Neural Information Processing Systems, pp. 1785-1796. 2019.

[Dann et al. 2017] Christoph Dann, Tor Lattimore, and Emma Brunskill. "Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning." In Advances in Neural Information Processing Systems, pp. 5713-5723. 2017.

[Dauparas et al. 2018] Justas Dauparas, Ryota Tomioka, and Katja Hofmann. *Depth and nonlinearity induce implicit exploration for RL*. ICML workshop on Exploration in RL (2018).

[Deisenroth & Rasmussen 2011] Marc Deisenroth & Carl Rasmussen (ICML, 2011): PILCO: A Model-based and Data-efficient Approach to Policy Search.

[Igl et al. 2019] Max Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, Katja Hofmann: Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottleneck. NeurIPS, 2019

References 3

[Janz et al. 2019] Janz, David, Jiri Hron, Przemysław Mazur, Katja Hofmann, José Miguel Hernández-Lobato, and Sebastian Tschiatschek. "Successor Uncertainties: exploration and uncertainty in temporal difference learning." In Advances in Neural Information Processing Systems, pp. 4509-4518. 2019.

[Lin 1993] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report No. CMU-CS-93-103. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[Liu et al. 2016] L. Liu, U. Dogan and K. Hofmann. *Decoding multitask DQN in the world of Minecraft.* European Workshop on Reinforcement Learning (EWRL), 2016.

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

[Osband et al. 2016] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. *Deep exploration via bootstrapped DQN*. In Advances in neural information processing systems, pp. 4026-4034. 2016.

References 4

[Puterman 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1994.

[Riedmiller 2000] Martin Riedmiller. *Concepts and facilities of a neural reinforcement learning control architecture for technical process control*. Neural computing & applications, 8(4), 323-338.

[Riedmiller 2005] Martin Riedmiller. *Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method*. In Machine Learning: ECML 2005, pages 317–328. Springer, 2005.

[Russo & Van Roy '14] D. Russo & B. Van Roy: *An Information-Theoretic Analysis of Thompson Sampling*. JMLR.

[Sæmundsson et al. 2018] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. "Meta reinforcement learning with latent variable gaussian processes." arXiv:1803.07551 (UAI 2018).

[Sæmundsson et al. 2019] Saemundsson, Steindor, Alexander Terenin, Katja Hofmann, and Marc Peter Deisenroth. "Variational Integrator Networks for Physically Meaningful Embeddings." arXiv preprint arXiv:1910.09349 (2019).

References 5

[Schulman et al. 2015] Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning (ICML).

[Schulman et al. 2017] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[Silver et al. 2016] David Silver; Huang, Aja; Maddison, Chris J.; Guez, Arthur; Sifre, Laurent; Driessche, George van den; Schrittwieser, Julian; Antonoglou, Ioannis; Panneershelvam, Veda; Lanctot, Marc; Dieleman, Sander; Grewe, Dominik; Nham, John; Kalchbrenner, Nal; Sutskever, Ilya; Lillicrap, Timothy; Leach, Madeleine; Kavukcuoglu, Koray; Graepel, Thore; Hassabis, Demis (28 January 2016). "Mastering the game of Go with deep neural networks and tree search". Nature. 529 (7587): 484–489.

[Sun et al. 2019] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. "Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches." In Conference on Learning Theory, pp. 2898-2933. 2019.

[Sutton et al. 2000] Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems.

References 6

[Szepesvári & Littman 1996] Csaba Szepesvári, and Michael L. Littman. "Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms." In Proceedings of International Conference of Machine Learning, vol. 96. 1996.

[Thomas & Brunskill, 2016] Thomas, P., and Brunskill, E. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In International Conference on Machine Learning, 2139–2148.

[Thompson 1933] W. R. Thompson: *On the likelihood that one unknown probability exceeds another in view of the evidence of two samples*. Biometrika, 25(3–4):285–294, 1933.

[Tsitsiklis & Van Roy 1997] John N. Tsitsiklis & Benjamin Van Roy. *Analysis of temporal-diffference learning with function approximation*. In Advances in neural information processing systems, pp. 1075-1081 (1997).

[Watkins 1989] Christopher J. C. H. Watkins. "Learning from delayed rewards." (1989).

[Zintgraf et al. 2019] Zintgraf, Luisa, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. "Fast Context Adaptation via Meta-Learning." In International Conference on Machine Learning, pp. 7693-7702. 2019.