EPFL

NX-414: Brain-like computation and intelligence

Alexander Mathis
alexander.mathis@epfl.ch

Lecture 12, May 14

Learning paradigms so far in NX-414

- Supervised learning
- Self-supervised learning
- Unsupervised learning

Later:

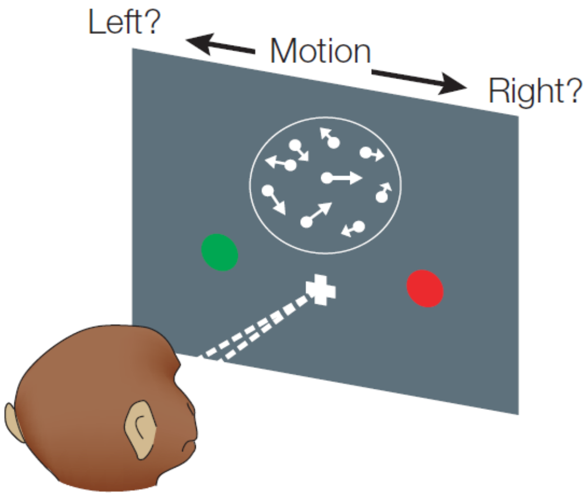
- Transfer and curriculum learning
- *Continual* learning

Can one also learn by interacting with the changing world?

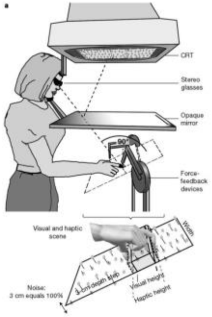
- No labels (not supervised)
- But reward/punishment …

Decision-making and behavior

Perceptual decision-making



Reminder:

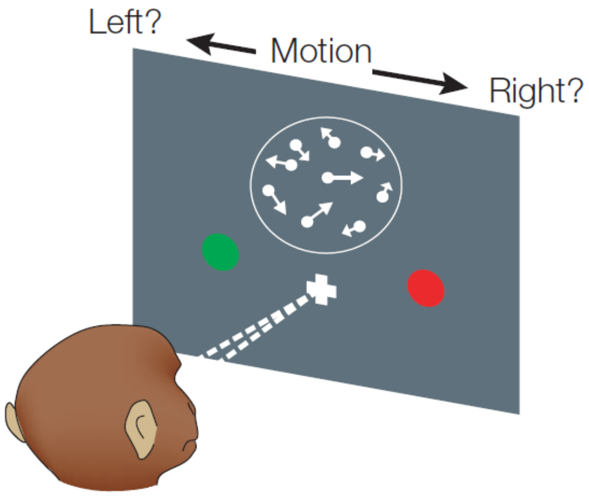


Ernst & Banks, Nature 2002

- Sensory evidence matters

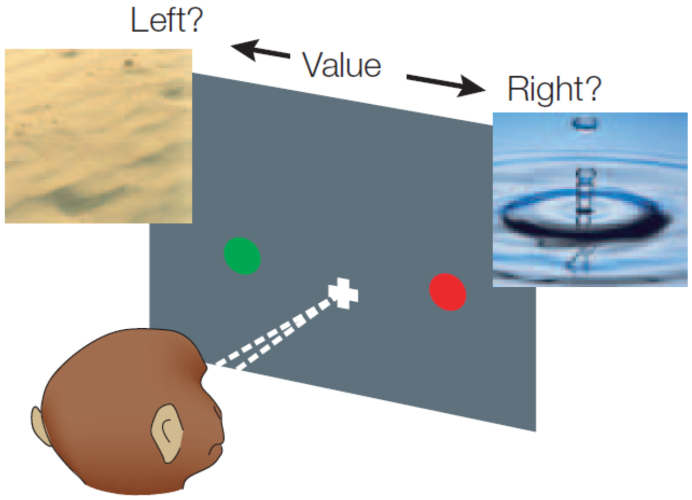
Decision-making and behavior

Perceptual decision-making



- Sensory evidence matters

Value-based decision

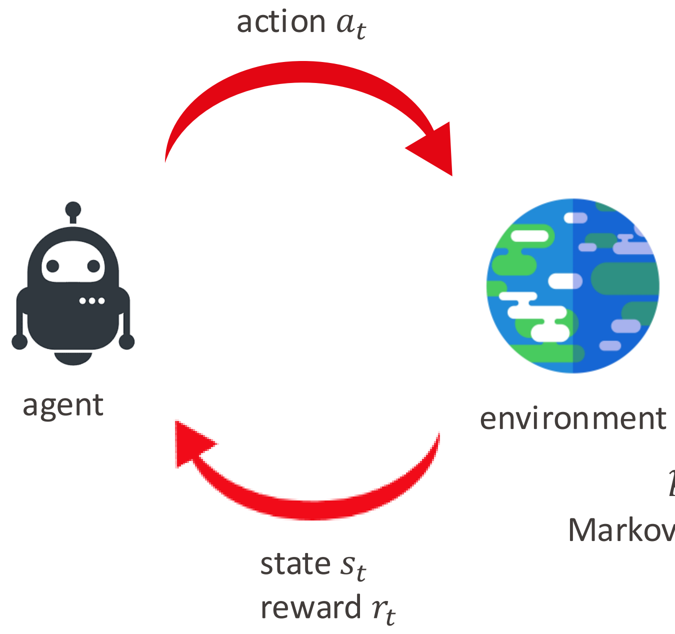


- Costs/benefits
- Value (utility)

Reinforcement learning

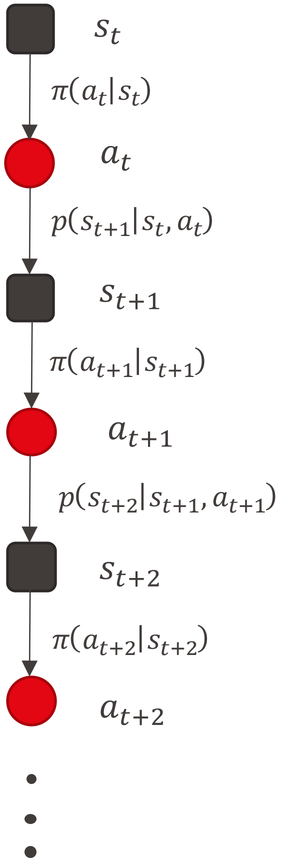
How should one act
(to maximize reward)?

policy $\pi(a_t|s_t)$



$p(s_{t+1}|s_t, a_t)$

Markov Decision Process



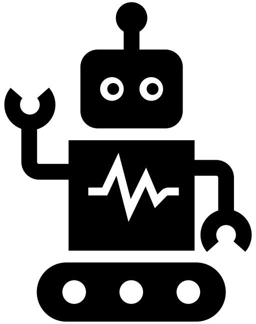
Reinforcement learning: Example

(S, A, R, P)

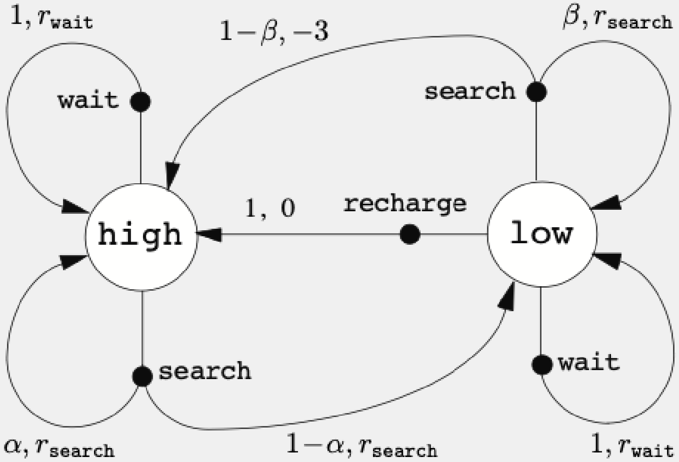
S = {high,low}

A(low) = {search, wait, recharge}

A(high) = {search, wait}



s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1-\alpha$	$r_{\texttt{search}}$
low	search	high	$1-\beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-



Sutton & Barto 2018

Reinforcement Learning

- Given a Markov decision process (S, A, R, P) find the policy $\pi(a|s)$, which **maximizes the cumulative future reward**
- (Typically) the transition probability $p(s', r|s, a)$ is unknown
- RL poses a challenging credit-assignment problem

Cumulative reward $$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

Action-value methods
Select the action with the highest expected cumulative reward

Policy-gradient methods
Update the policy in the direction which maximizes the expected cum. reward

Value and action-value functions

State-value function

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \;\middle|\; S_t = s\right], \text{ for all } s \in \mathcal{S}$$

Quality function / state-action value function

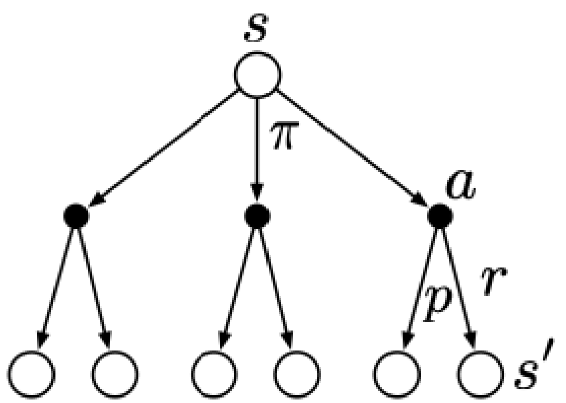
$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \;\middle|\; S_t = s, A_t = a\right]$$

Sutton & Barto 2018

Bellman Equation

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

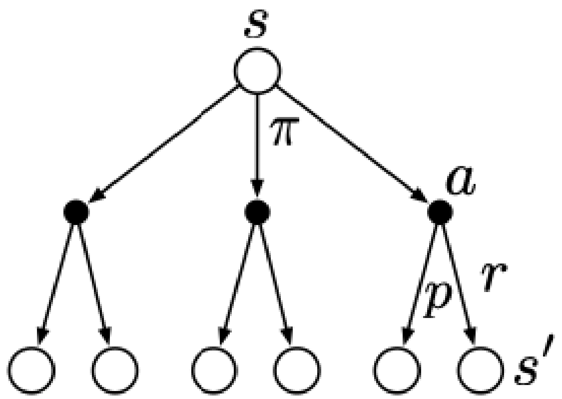
$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \end{aligned}$$



Backup diagram for v_π

Bellman Equation

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$



Backup diagram for v_π

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S}, \end{aligned}$$

Sutton & Barto 2018

Optimal value functions and policies

$$v_*(s) \doteq \max_{\pi} v_\pi(s)$$

$$q_*(s, a) \doteq \max_{\pi} q_\pi(s, a)$$

Give a state-action function, the optimal (greedy) policy is given by:

$$\pi(s) = argmax_a \, q_*(s, a)$$

Sutton & Barto 2018

Bellman optimality equations

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_*(s')\Big],$$

$$q_*(s,a) = \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big]$$
$$= \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big],$$

Sutton & Barto 2018

Temporal difference learning

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_*(s')\Big],$$

Core idea for a learning algorithm:
use difference in expected and received reward to **update the value function**:

$$V(s_t) \leftarrow V(s_t) + \alpha_t(R_t + \gamma V(s_{t+1}) - V(s_t))$$

Values are updated based on *previous* values plus reward prediction error weighed by learning rate

EPFL

A Neural Substrate of Prediction and Reward

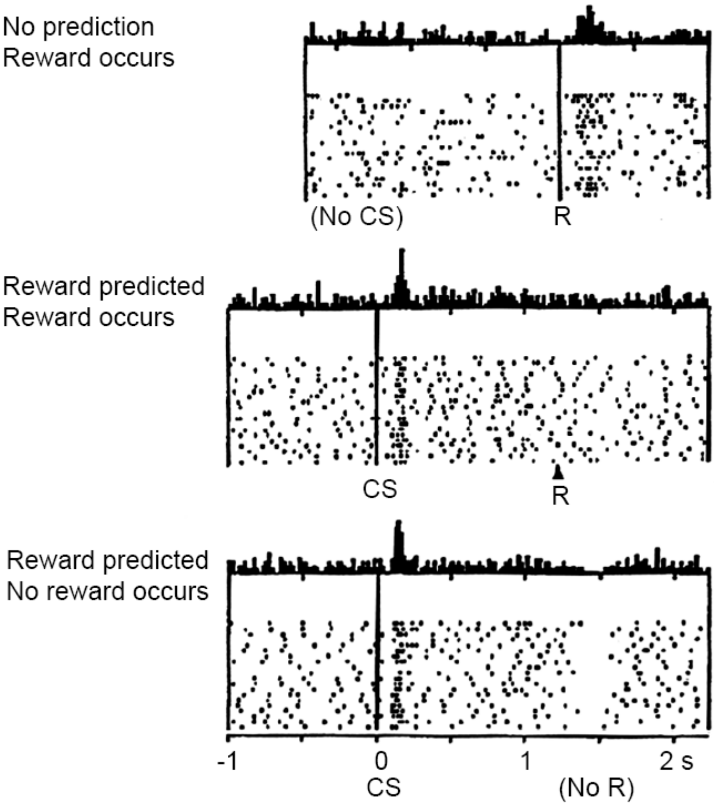
Wolfram Schultz, Peter Dayan, P. Read Montague*

The capacity to predict future events permits a creature to detect, model, and manipulate the causal structure of its interactions with its environment. Behavioral experiments suggest that learning is driven by changes in the expectations about future salient events such as rewards and punishments. Physiological work has recently complemented these studies by identifying dopaminergic neurons in the primate whose fluctuating output apparently signals changes or errors in the predictions of future salient and rewarding events. Taken together, these findings can be understood through quantitative theories of adaptive optimizing control.

$$V(s_t) \leftarrow V(s_t) + \alpha_t \underbrace{(R_t + \gamma V(s_{t+1}) - V(s_t))}_{\text{Dopamine?}}$$

Schultz, Dayan, Montague, Science 1997

Dopamine as temporal difference (TD) error: reward prediction errors



Schultz, Dayan, Montague, Science 1997

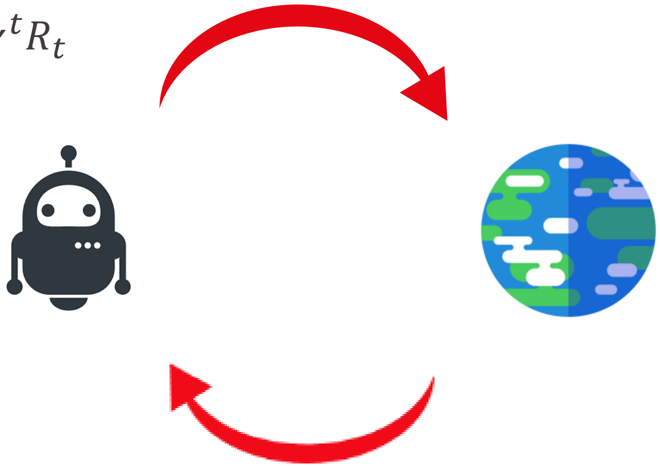
EPFL

Reinforcement Learning

Given a Markov decision process (S, A, R, P) find the policy $\pi(a|s)$ which maximizes the *cumulative future discounted reward*

$$\sum_{t=0}^{\infty} \gamma^t R_t$$

$$\pi(a|s)$$



TD learning (and related algorithms) have helped us model many basic learning paradigms in a satisfactory way.

- But does it scale?

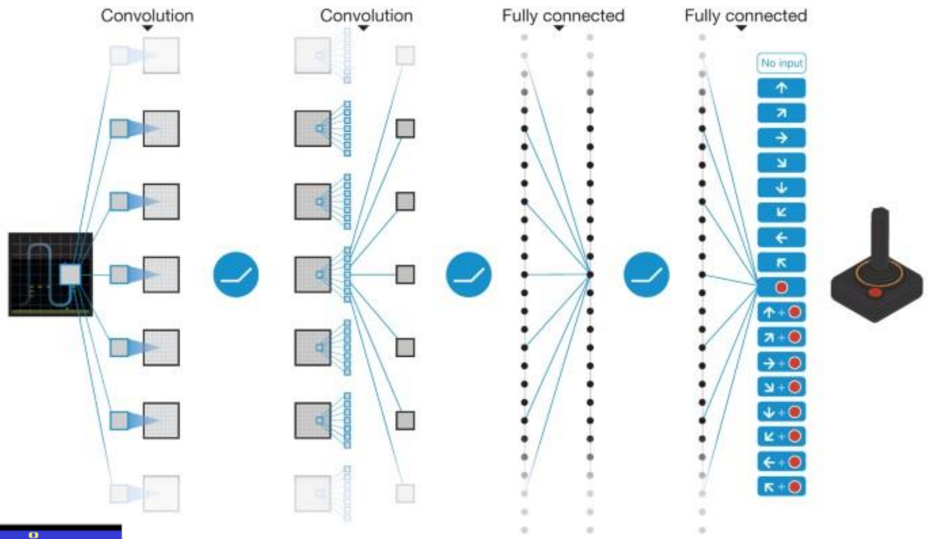
Fundamental Challenge and representation learning...

- The computational and memory requirements (even) for games is enormous!

Chess: 10^{120}

Go: 3^{361}

Parameters θ_i



$$L_i\left(\theta_i\right) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q\left(s,a;\theta_i\right)\right)^2\right]$$

$$\nabla_{\theta_i} L_i\left(\theta_i\right) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

-

Mnih et al., Nature 2013 (Deep Mind)

EPFL

"DQN" algorithm

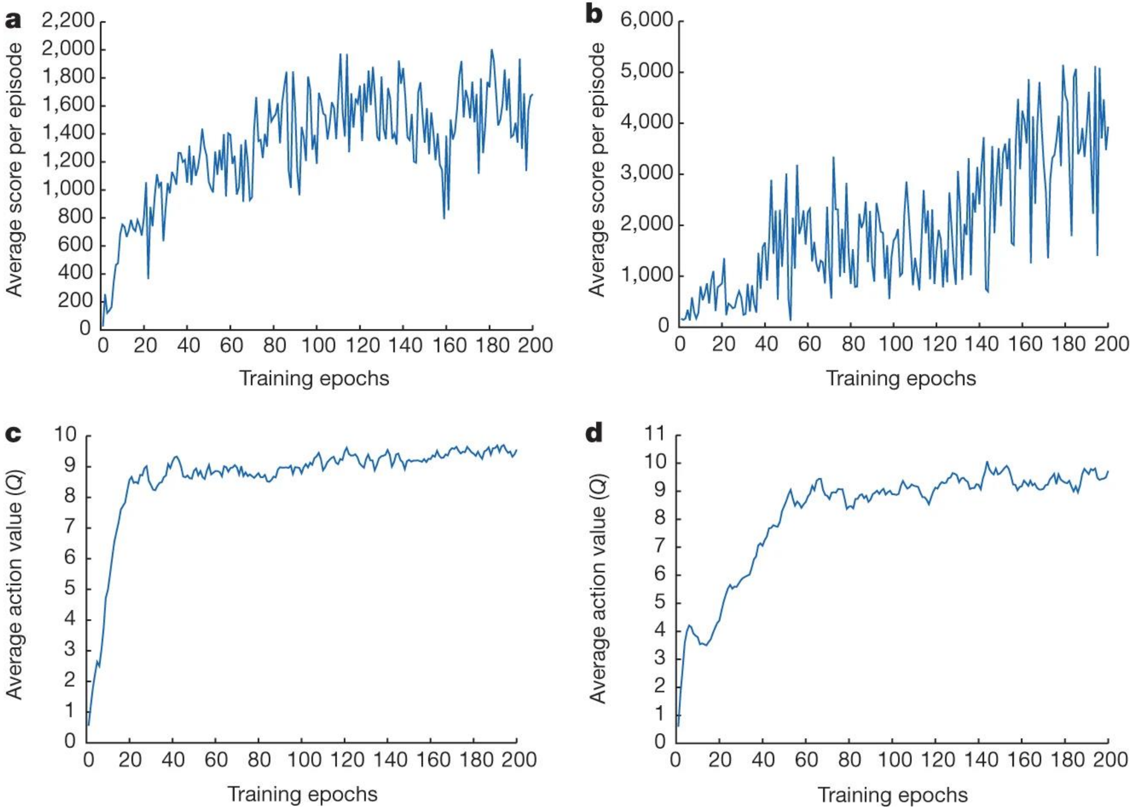
Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Mnih et al., Nature 2013 (Deep Mind)

DQN learns stable Q-functions

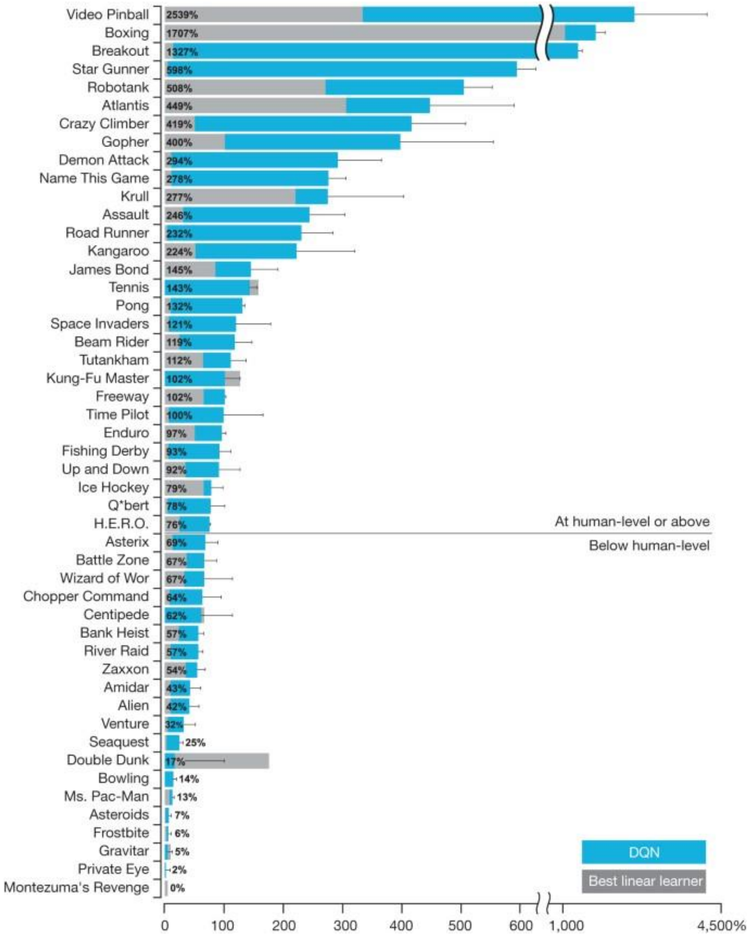
The player-controlled laser cannon shoots the aliens as they descend (from Wiki).



The player uses a submarine to shoot at enemies and rescue divers (from Wiki).

Mnih et al., Nature 2013 (Deep Mind)

DQN performs at superhuman level for some games



DQN vs. humans: The performance of DQN is normalized with respect to a professional human games tester (that is, 100% level) and random play (that is, 0% level). Note that the normalized performance of DQN, expressed as a percentage, is calculated as: 100 × (DQN score − random play score)/(human score − random play score). Audio output was disabled for both human players and agents. Error bars indicate s.d. across the 30 evaluation episodes, starting with different initial conditions.

- DQN outperformed the best existing reinforcement learning methods on 43 of the games without incorporating any of the additional prior knowledge about Atari 2600 games used by other approaches
- DQN outperformed human gamers for many games

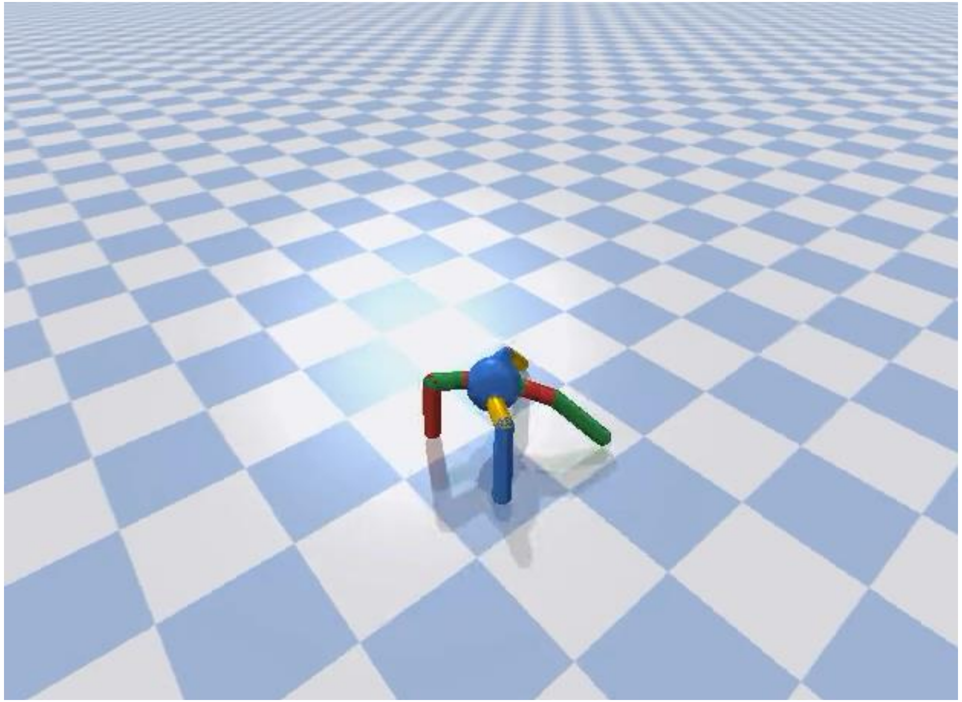
Mnih et al., Nature 2013 (Deep Mind)

DQN was just the start …

- **AlphaGo**: combined deep neural networks with advanced search algorithms. It won against the best human player (Lee Sedol). Check out: https://deepmind.google/technologies/alphago/
- Check out Chapters 16 & 17 in Sutton & Barto's textbook: http://incompleteideas.net/book/the-book.html

RL for continuous control

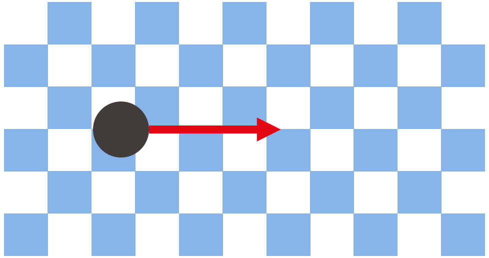
A typical reinforcement problem (continuous control)



Ant (OpenAI Gym)

- Quadruped robot
- Action size: 8
- State size: 28

- Objective: run as fast as possible



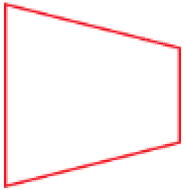
Continuous control…

Parametrize the policy…

$$\pi(A|S,\theta)$$

$\boldsymbol{s}_t$

Proprioceptive state



$\boldsymbol{a}_t$

Action

Policy Gradient Methods

Given a parametrization θ of the policy, find an iterative method of the form

$$\theta_{t+1} = \theta_t + \alpha \, \nabla \widehat{\mathrm{J}(\theta_t)}$$

For cumulative future rewards:

$$J(\theta) = v_{\pi_\theta}(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| \, S_0 = s_0\right]$$

Policy Gradient Theorem

The following proportionality relation for the gradient of J holds:

$$\nabla J(\theta) \propto \sum_{s} \mu(s) \sum_{a} q_{\pi}(s,a) \nabla \pi(a|s,\theta) = \mathbb{E}_{\pi}[a_{\pi}\,(\mathrm{S},\mathrm{A}) \nabla \ln \pi(A|S,\theta)]$$

μ is the distribution of the state under policy π

q is the state-action value

$a_{\pi} = q_{\pi} - v_{\pi}$ is the advantage function

EPFL

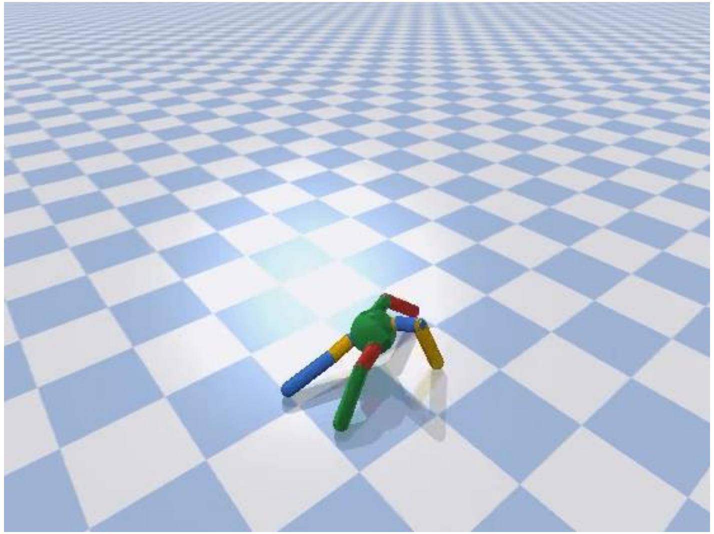
Soft actor-critic (SAC) algorithm

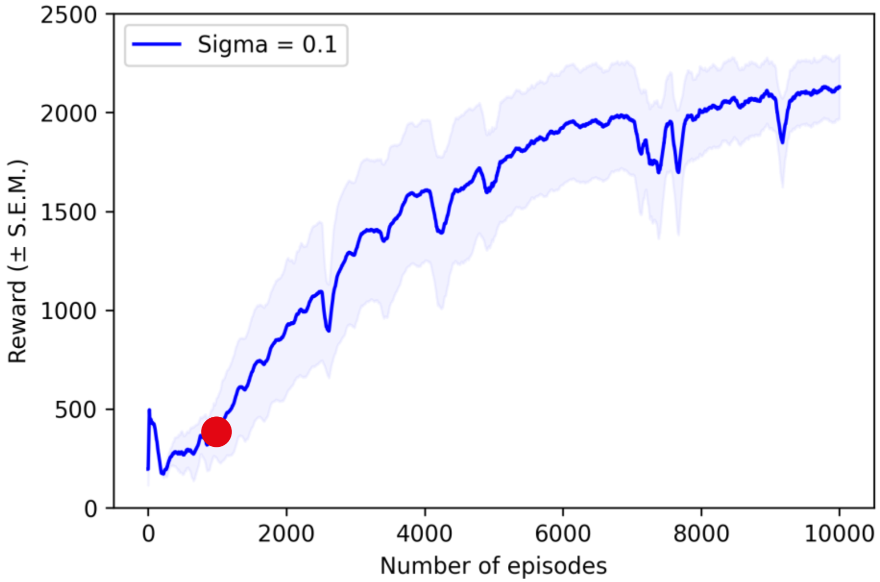
Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

- A strong & popular ”policy gradient method”
- Proposed by Haarnoja et al. 2018
- Off-policy actor-critic algorithm
- Actor is the policy, critic learns Q.
- Agents maximize expected reward and also entropy (i.e. succeeding at the task while acting as randomly as possible)
- D is the distribution of sampled states and actions (replay buffer)

Continuous control with policy-gradient methods

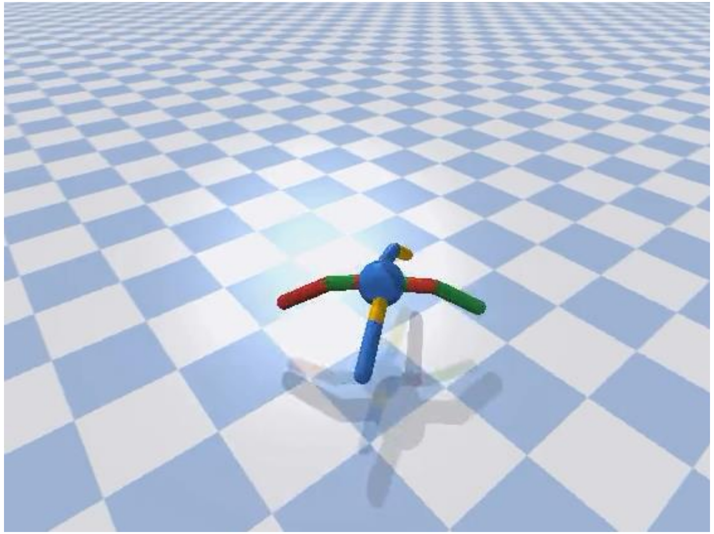


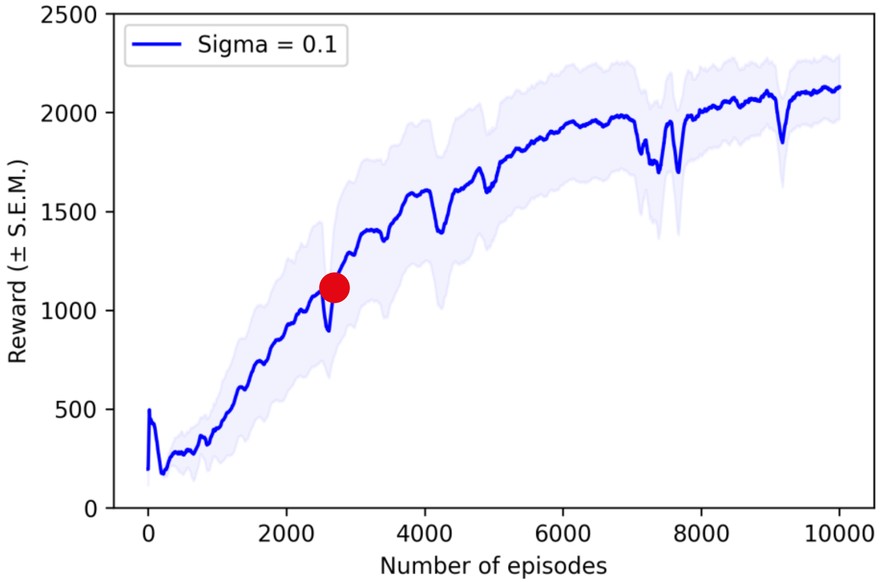


SAC baseline from:

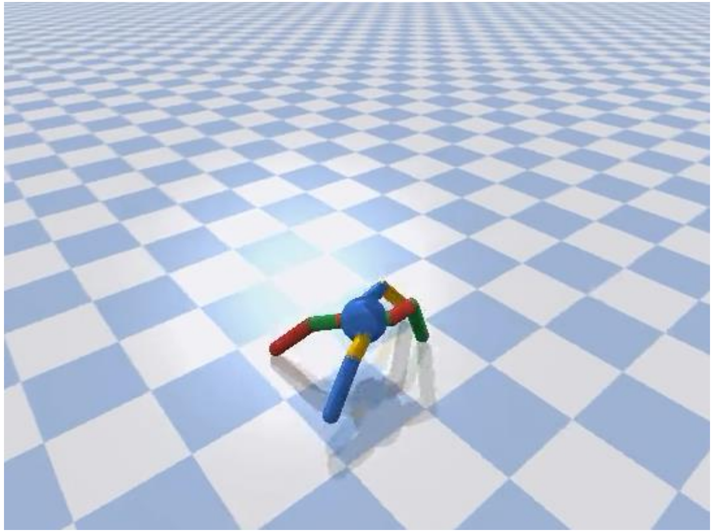
Chiappa, Marin Vargas, Mathis, Neurips 2022 /arxiv

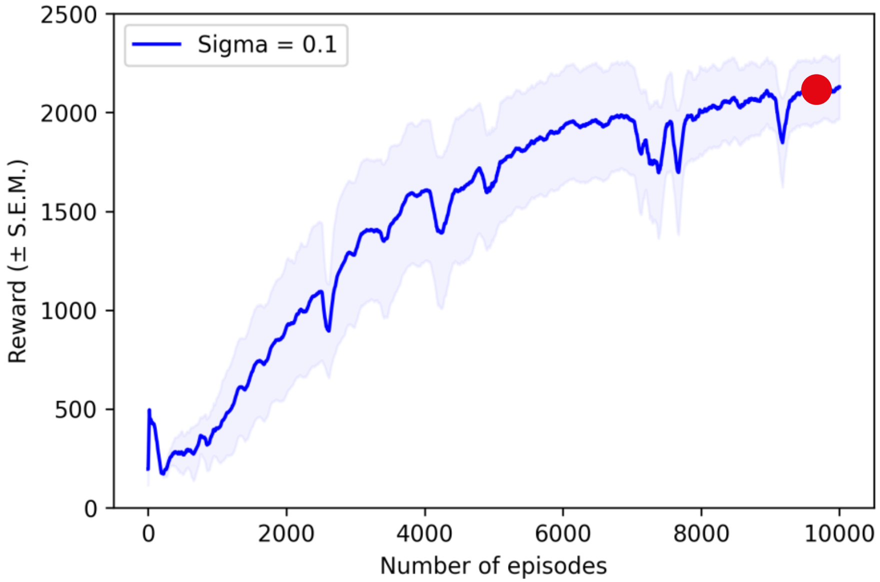
Continuous control with policy-gradient methods





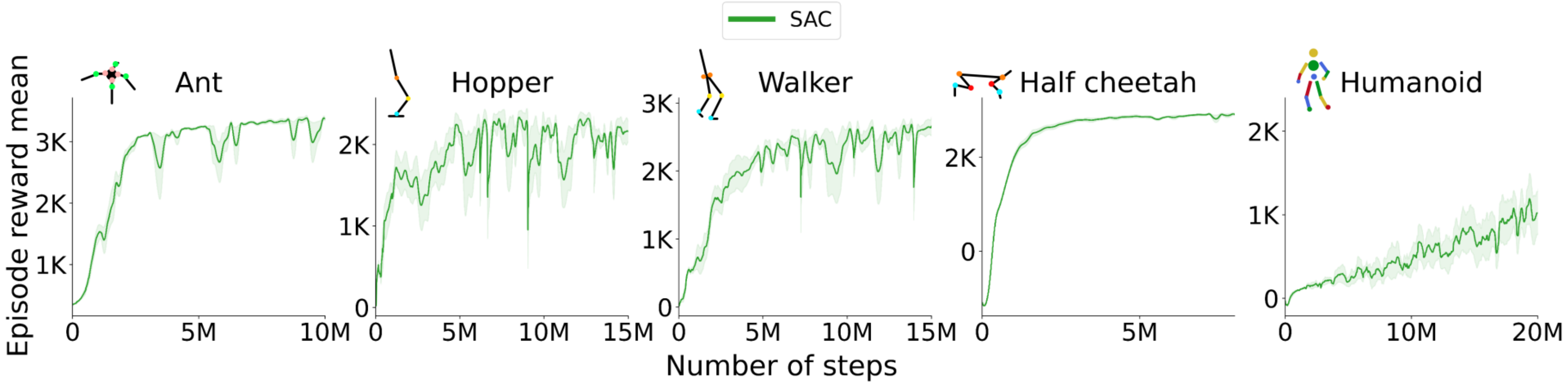
Continuous control with policy-gradient methods

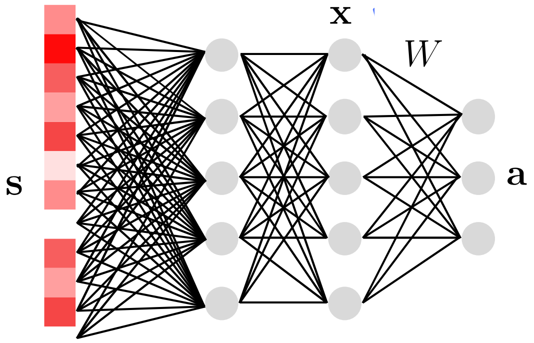




Better exploration?

Learning sensorimotor skills



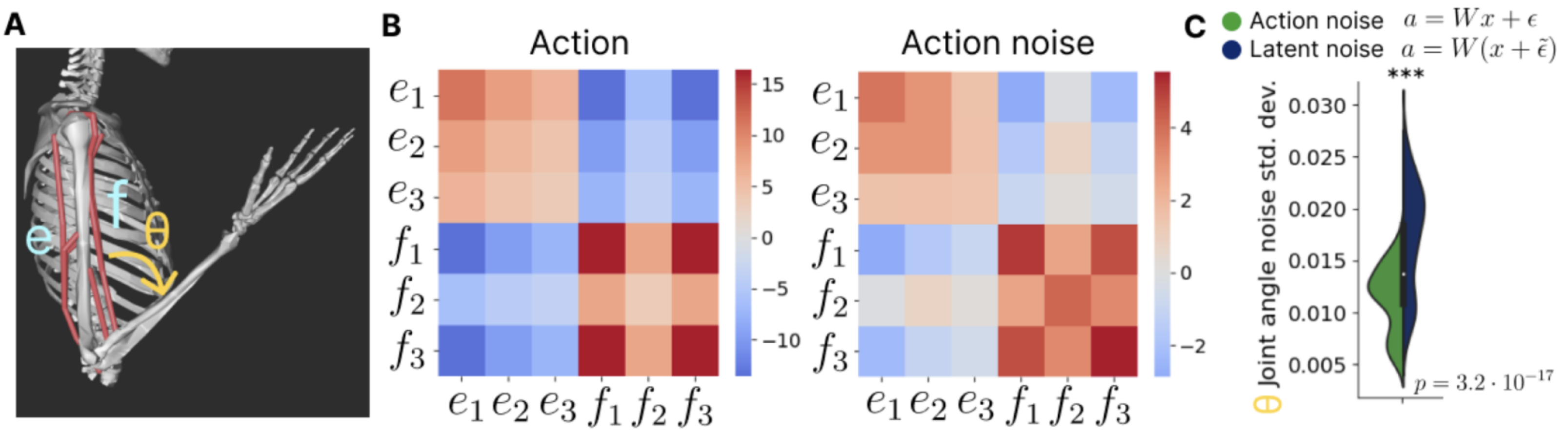


Default $\mathbf{a} = W\mathbf{x} + \epsilon$

Isotropic exploratory noise

Chiappa, Marin Vargas, Huang, Mathis, NeurIPS 2023

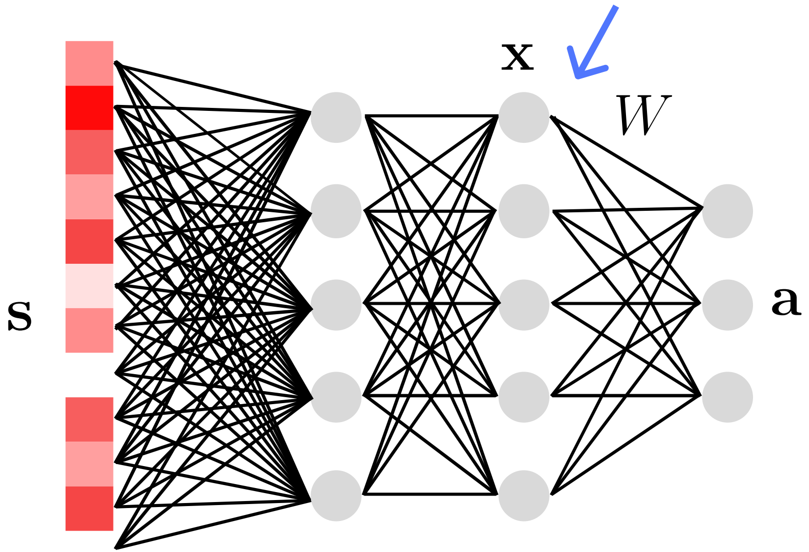
Basic intuition for better exploration



Chiappa, Marin Vargas, Huang, Mathis, NeurIPS 2023

Latent time-correlated exploration

LATTICE - **LAT**ent **TI**me-**C**orrelated **E**xploration



Perturbation matrices

$N_{\mathbf{a}}$ × $N_{\mathbf{x}}$: $P_{\mathbf{a}}$

$$(P_{\mathbf{a}})_{i,j} \sim \mathcal{N}\left(0, (S_{\mathbf{a}})_{i,j}\right)$$

$N_{\mathbf{x}}$ × $N_{\mathbf{x}}$: $P_{\mathbf{x}}$

$$(P_{\mathbf{x}})_{i,j} \sim \mathcal{N}\left(0, (S_{\mathbf{x}})_{i,j}\right)$$

LATTICE $\mathbf{a} = (W + P_{\mathbf{a}} + W P_{\mathbf{x}})\mathbf{x}$

State-Dependent Exploration

gSDE $\mathbf{a} = (W + P_{\mathbf{a}})\mathbf{x}$

Default $\mathbf{a} = W\mathbf{x} + \epsilon$

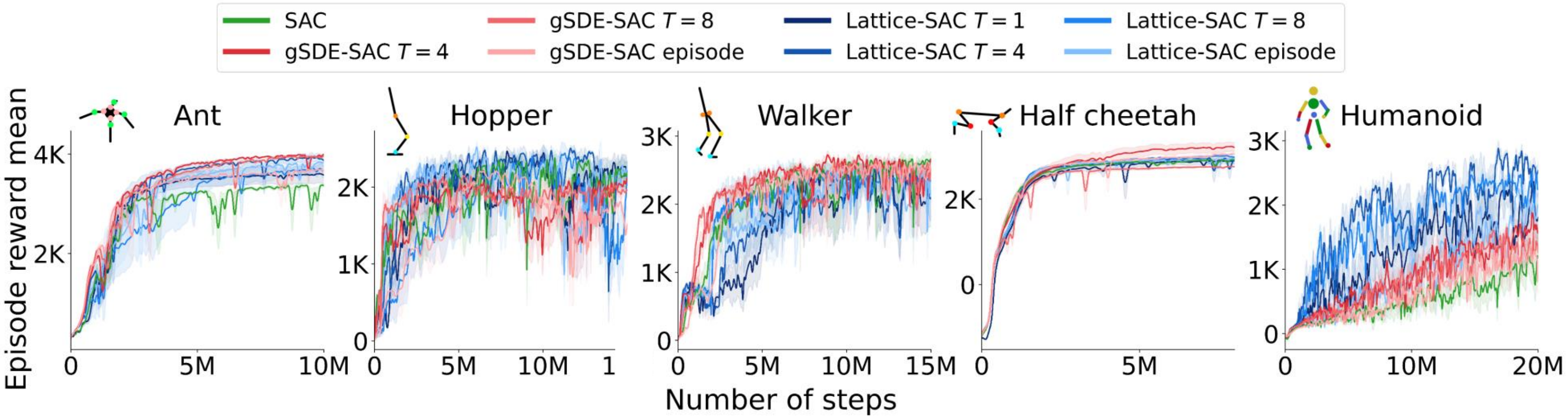
Time

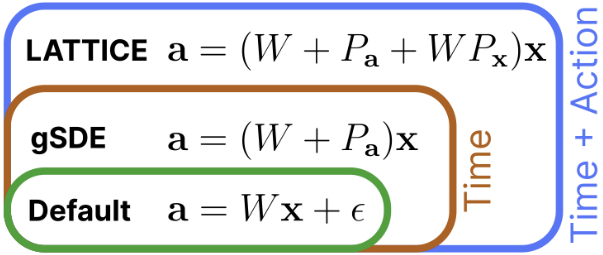
Time + Action

Chiappa, Marin Vargas, Huang, Mathis, NeurIPS 2023

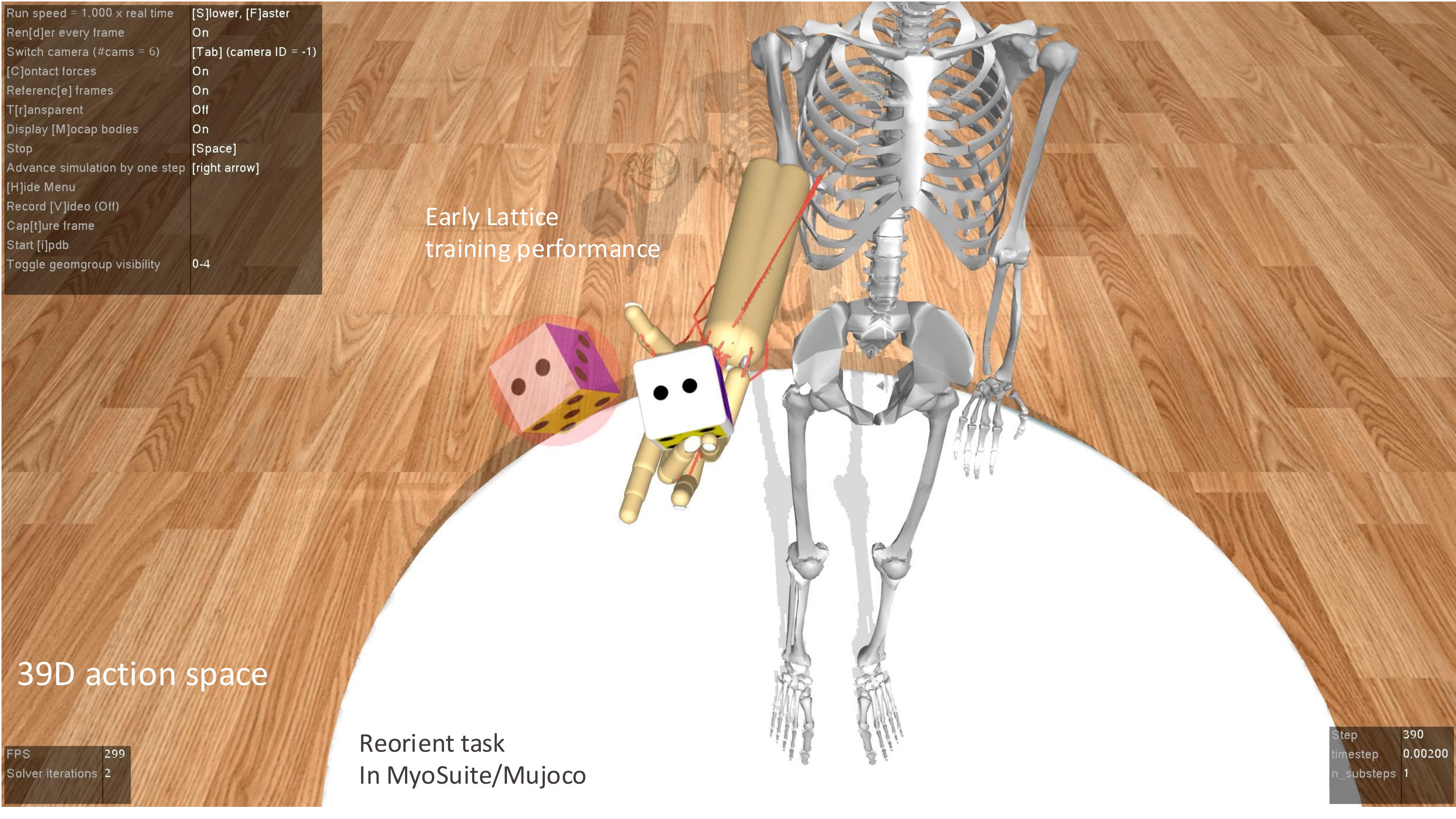
EPFL

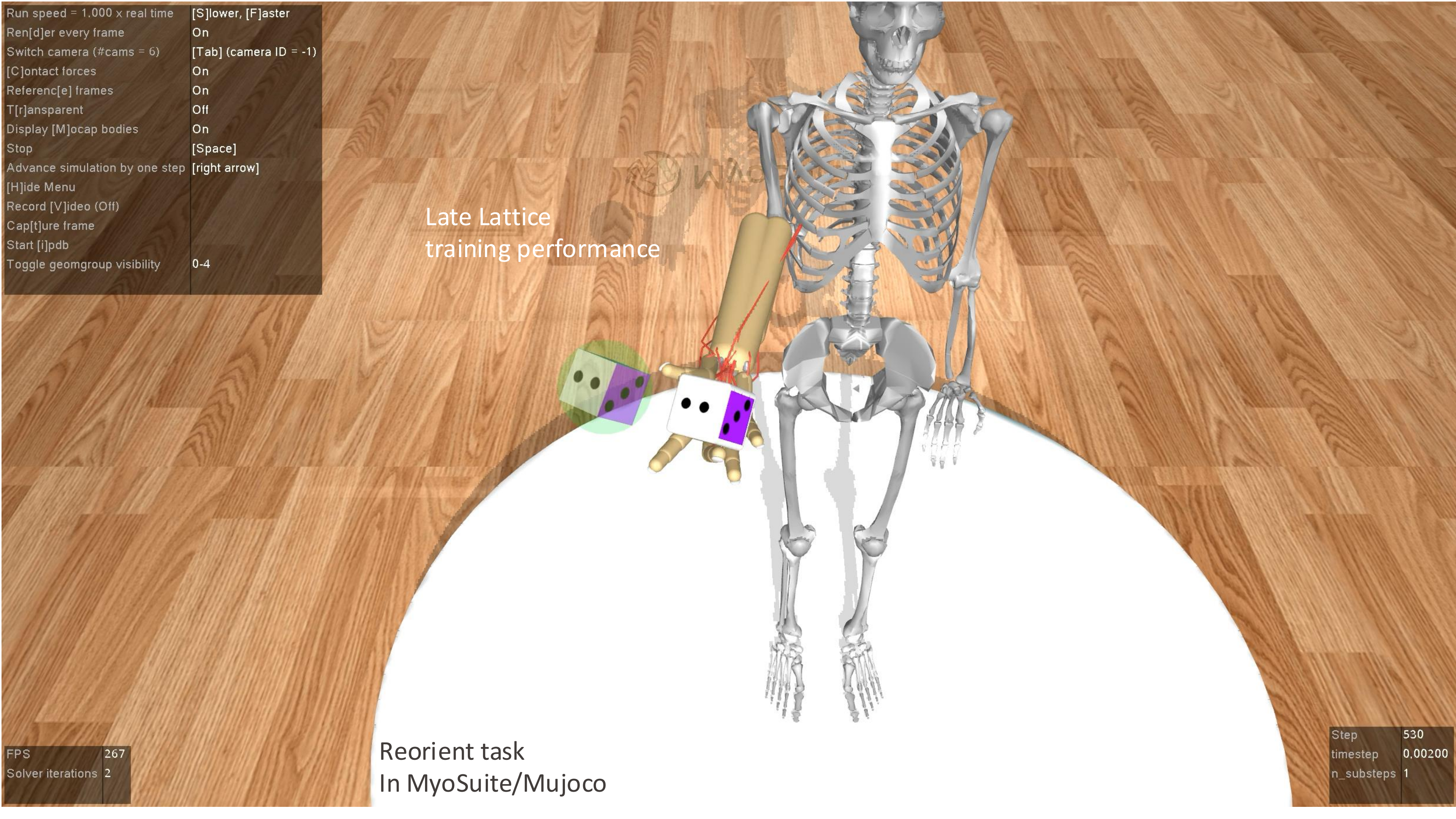
Benchmarking learning to locomote



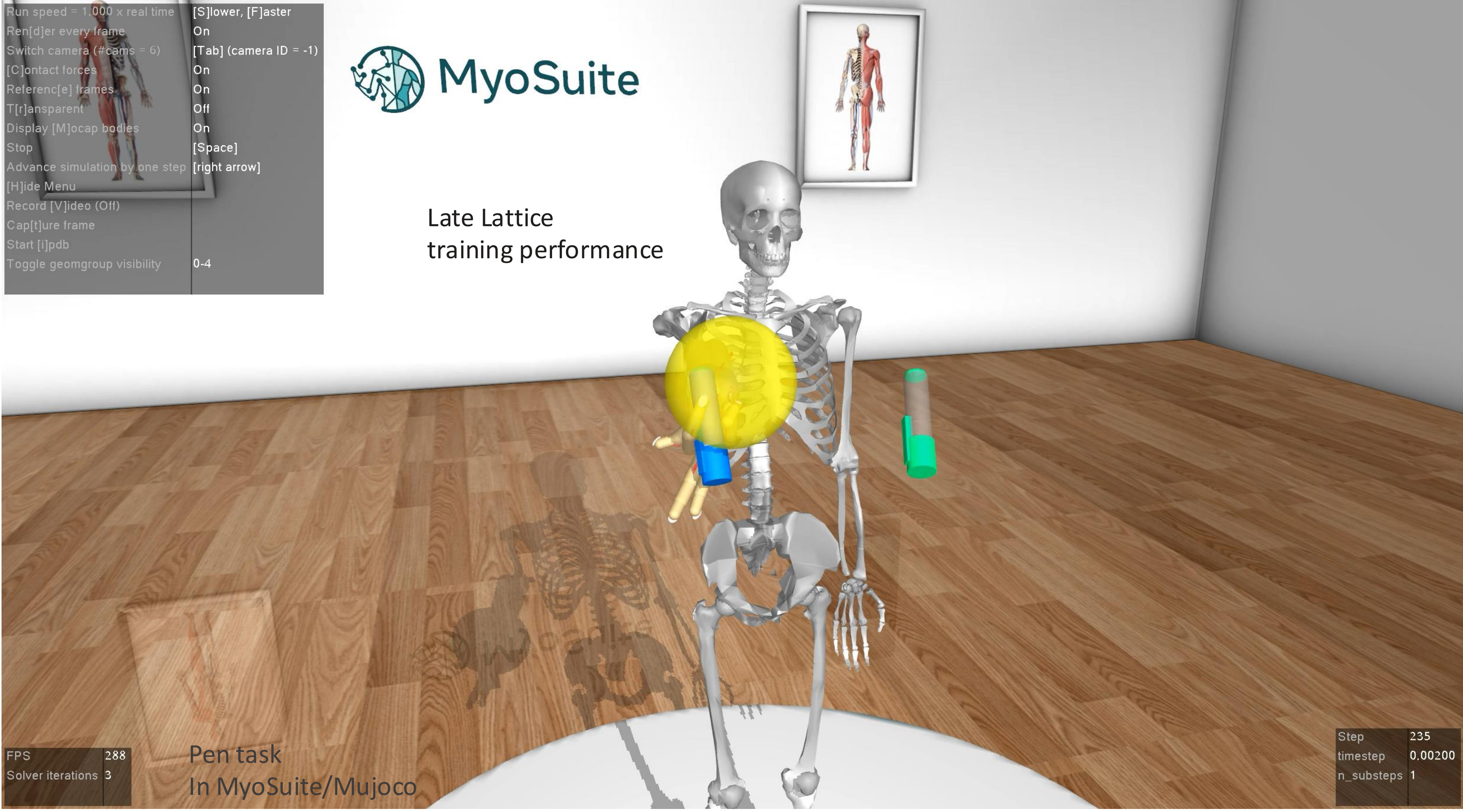


Chiappa, Marin Vargas, Huang, Mathis, NeurIPS 2023


Run speed = 1.000 x real time
[S]lower, [F]aster
Ren[d]er every frame
On
Switch camera (#cams = 6)
[Tab] (camera ID = -1)
[C]ontact forces
On
Referenc[e] frames
On
T[r]ansparent
Off
Display [M]ocap bodies
On
Stop
[Space]
Advance simulation by one step
[right arrow]
[H]ide Menu
Record [V]ideo (Off)
Cap[t]ure frame
Start [i]pdb
Toggle geomgroup visibility
0-4
Early Lattice
training performance
39D action space
Reorient task
In MyoSuite/Mujoco
FPS
299
Solver iterations
2
Step
390
timestep
0.00200
n_substeps
1

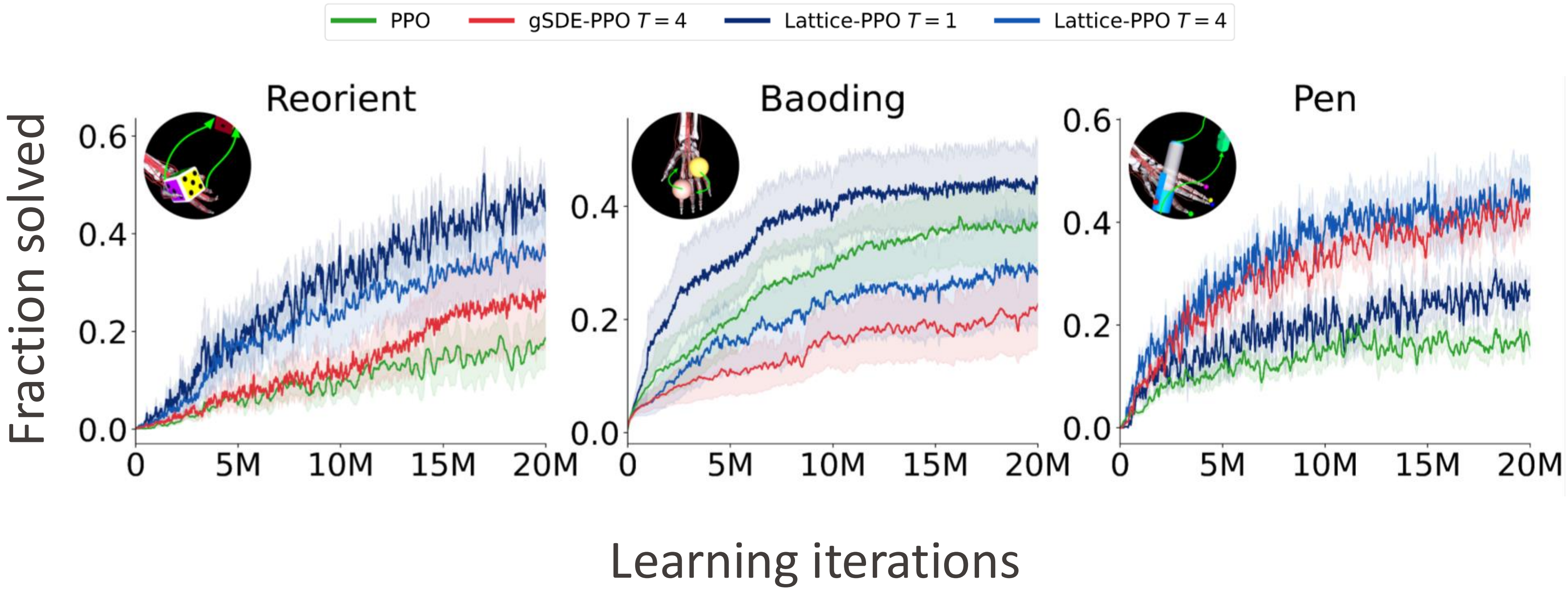

Run speed = 1.000 x real time
[S]lower, [F]aster
Ren[d]er every frame
On
Switch camera (#cams = 6)
[Tab] (camera ID = -1)
[C]ontact forces
On
Referenc[e] frames
On
T[r]ansparent
Off
Display [M]ocap bodies
On
Stop
[Space]
Advance simulation by one step
[right arrow]
[H]ide Menu
Record [V]ideo (Off)
Cap[t]ure frame
Start [i]pdb
Toggle geomgroup visibility
0-4
Late Lattice
training performance
FPS
267
Solver iterations
2
Reorient task
In MyoSuite/Mujoco
Step
530
timestep
0.00200
n_substeps
1

Late Lattice
training performance



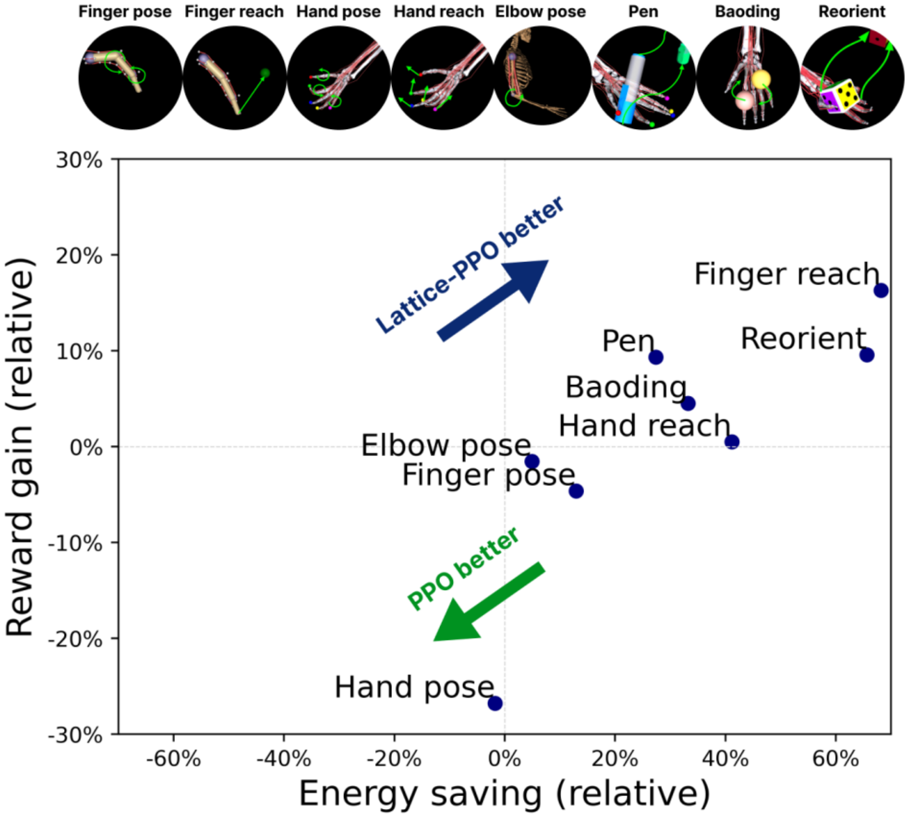
Pen task
In MyoSuite/Mujoco

Object manipulation learning curves



Chiappa, Marin Vargas, Huang, Mathis, NeurIPS 2023

Lattice learns more energy efficient solutions



Chiappa, Marin Vargas, Huang, Mathis, NeurIPS 2023

Different approaches to motor control

Optimal control

+ can handle constraints & finds optimal, adaptive strategies

- computationally intensive

- requires a good model of the system

Reinforcement learning

+ flexible to design

+ finds novel solutions

+ adaptive to changes in the environment

- large-scale simulation

- high sample complexity

Take-home messages

- Bellman optimality equations describe a consistency requirement that value and state-value functions need to satisfy
- They motivate many algorithms (Q-learning, TD-learning, Fix-point perspective, …)
- Parametrizing policy/q-functions with neural networks
- For continuous control, policy gradient methods are much more powerful
- Efficient exploration for large action spaces is an active area of research.
- Reinforcement learning provides a theory for adaptive behavior and adaptive, optimizing control