Learning Control from minimal prior knowledge

IPAM workshop on control, learning and optimization, Feb 2020

Martin Riedmiller, DeepMind

Roland Hafner, Tom Erez, Yuval Tassa, Thomas Lampe, Tobias Springenberg, Jonas Degrave, Jonas Buchli, Abbas Abdolmaleki, Tim Hertweck, Nicolas Heess, Michael Neunert, Markus Wulfmeier, Noah Siegel, Michael Bloesch, and many others at DM + previous research groups

DeepMind's mission

Solve Intelligence. Use it to solve everything else.

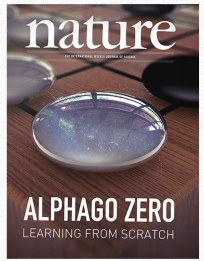
Intelligence is the ability to solve a wide variety of tasks:
Artificial General Intelligence (AGI)

riedmiller@google.com

- Atari (2014): Learning super human behaviour on 50+ Atari games from pixels (Mnih et.al, 2014)

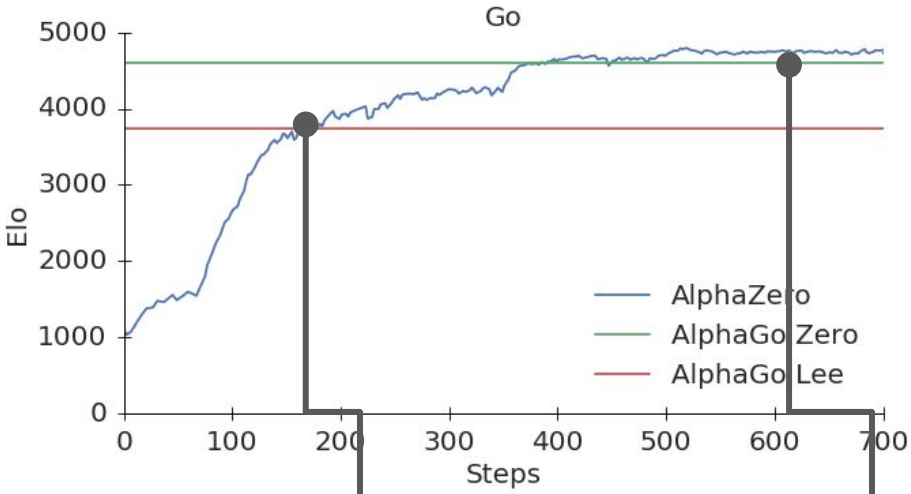
- AlphaGo (2016): Beating the human world champion Lee Sedol (Silver etl.al, 2016)

- AlphaZero (2018): Learning from zero knowledge: Chess, Go, Shogi (Silver et.al, 2018)



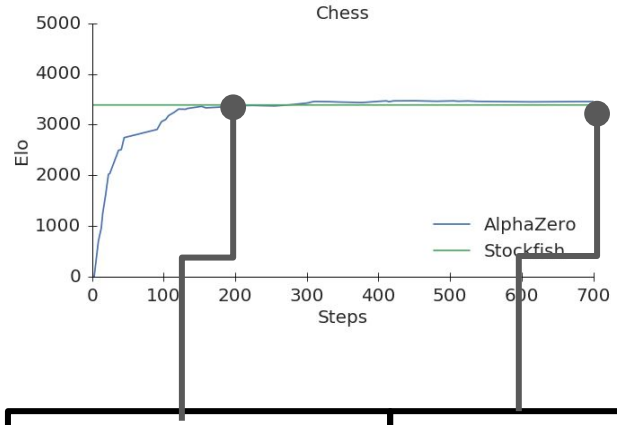
riedmiller@google.com

AlphaZero in Go

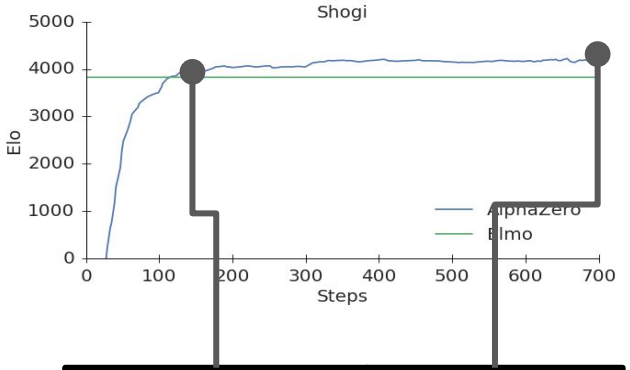


AlphaZero beats *AlphaGo* after 8 hours

AlphaZero wins 100
AlphaGo wins 0



4h beats *Stockfish* | wins 155: 6



2h beats *Elmo* | wins 911: 77

What would AGI mean for control?

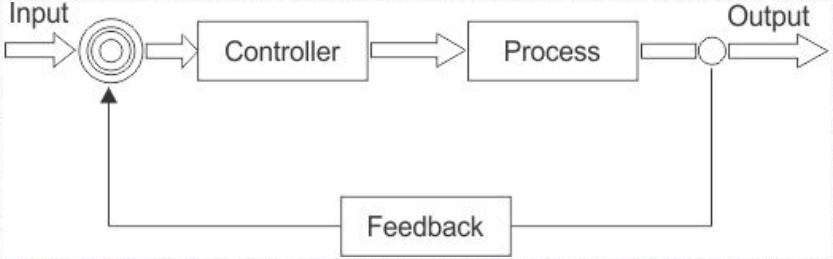
from scratch, from raw signals, yet efficient

riedmiller@google.com

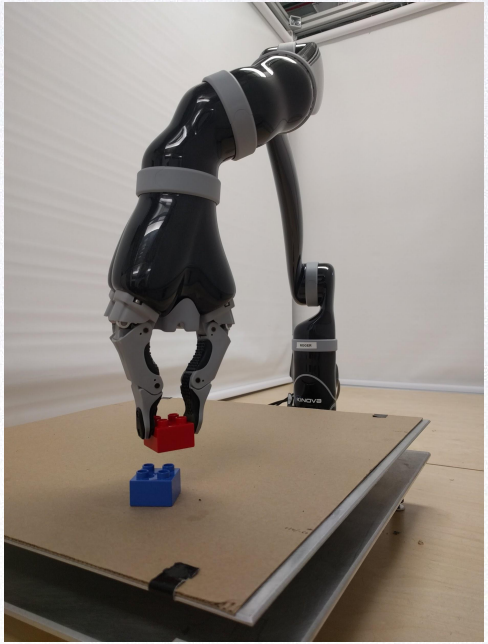
Control team: our mission

learning reactive (closed-loop) control from scratch

Artificial General Control Intelligence (AGCI)



$$\max_{\pi} \left\langle \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right\rangle$$



riedmiller@google.com

Overview

- Basics: the ‘collect and infer’ viewpoint of RL
- Advanced aspects
- Challenges/ Wishlist

riedmiller@google.com

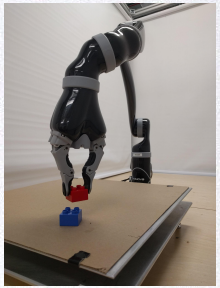
The promise of RL: Learn by success/ failure

(Sutton and Barto, 1983, 1998, Bertsekas, 1996)

given: stochastic dynamical system **p(s' | s,a)**

immediate reward per decision: **r(s,a)**

e.g.: r = 1, if object is lifted, 0, else (-> 'sparse')



wanted: optimal policy $\pi^*(s) = argmax_\pi E\{\sum_{t=0}^{\infty} \gamma^t r(s,a) | s_0 = s, \pi\}$

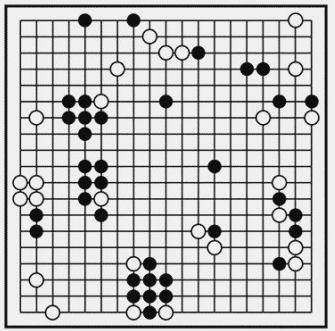
Methods: dynamic programming $Q_{k+1}(s,a) := \sum_{s'} p(s'|s,a)(r(s,a) + \gamma \max_b Q(s',b))$

(Bellman, 1959, Watkins 1989)

- can be made model-free and adapted to continuous states: Neural networks to approximate Q (e.g., Riedmiller, 1996)

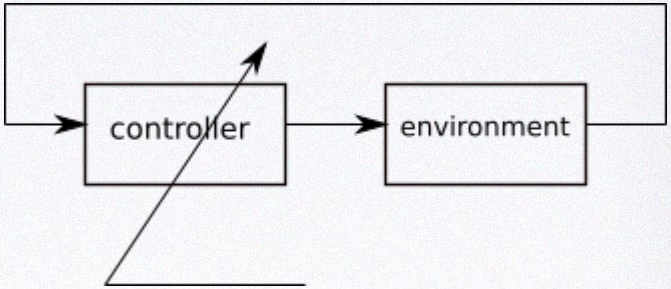
Challenges for control

- continuous states, continuous actions, high number of DoFs
- huge search tree
  -> sparse rewards are difficult to find
- e.g. 9 DoFs, 3 actions/ DoF, 50 steps:
  (3^9)^50 ≈ 10^200 paths
- approaches:
  - sim2real (e.g. OpenAI et.al 2018, Chebotar et.al, 2018, Rusu et.al, 2016))
  - learning from demonstrations (e.g. Peters et.al, 2010)
  - from scratch with minimal prior knowledge (e.g. Kalashinikov, et.al, 2018):
    data-efficeny is key for real robots/ systems



Classical RL

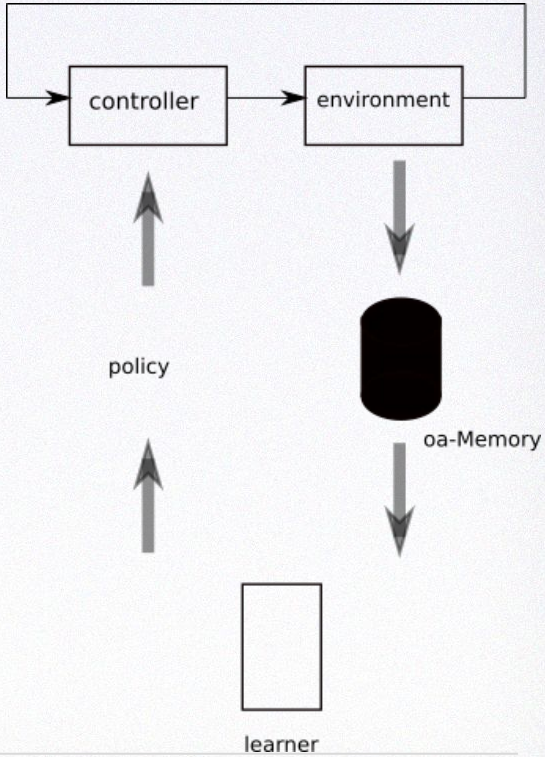
- ‘classical’ RL (Sutton, Barto, 1998, 2018): act - observe - update - act
- highly data-consuming



riedmiller@google.com

Data-efficient RL

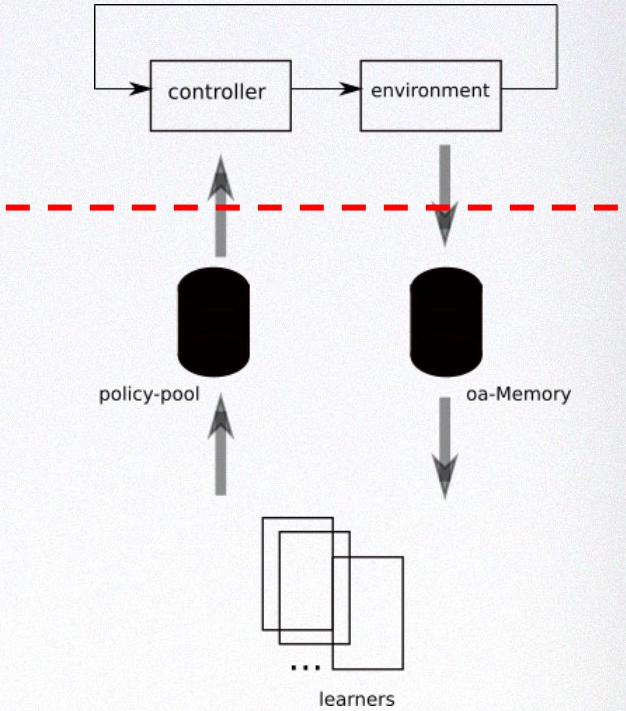
- key insight:
  Data is precious and data is true
  -> store all transitions (s,a,s') in a transition memory and learn on entire memory:
  'sample based model of the world'
- 'infer' knowledge: off-policy RL learning
  e.g. Neural Fitted Q Iteration (NFQ, NFQCA)
  (Riedmiller, 2005, Ernst, 2005, Lin 1992, Hafner & Riedmiller, 2011)



Data-efficient RL (2)

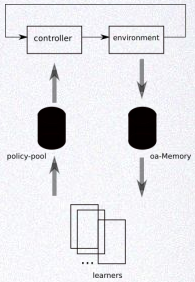
- key insight 2 (‘collect’): collecting the ‘right’ transitions. objective: learn fast

- interaction with the environment and learning are detached:
  real time capability
  + exploit huge compute (‘Dreaming’)



riedmiller@google.com

‘Collect and Infer’ (Riedmiller, EWRL 2018)



- infer: how to squeeze out most information of the experience in the transition memory -> efficient off-policy learning methods
- collect: how to efficiently sample the relevant information about the environment by interacting with it -> efficient exploration

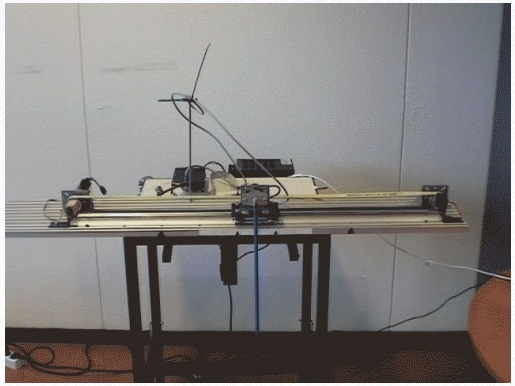
riedmiller@google.com

*Collect and **Infer**:*
Learning from transition memories

Neural Fitted Q: RL from transition memories

(Riedmiller, 2005, related: Ernst 2005)

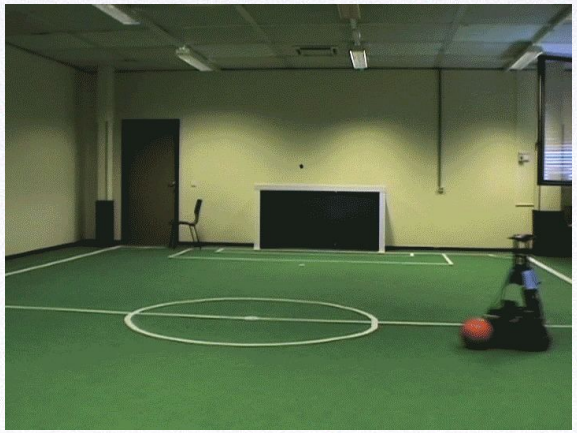
- given a (large) set of transitions ('batch') *D={(s,a,s')_i, i=1...N}*
- Approximate Q by NN
- iteratively minimize $(Q(s,a) - (r(s,a) + \gamma max_b Q(s',b)))^2$
  - turns RL into a series of supervised learning problems
  - personal experience: online TD update > 1 Mio episodes for cart-pole, NFQ: < 300



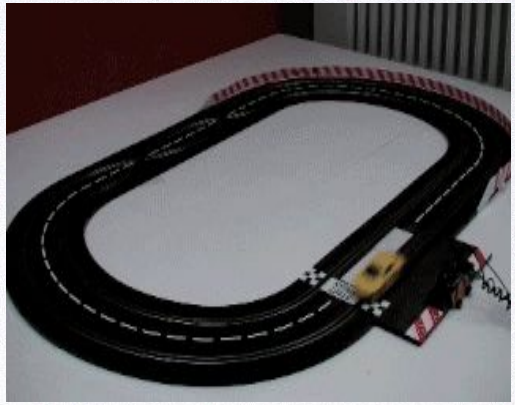
riedmiller@google.com

NFQ examples

RL learns to drive
Riedmiller et.al 2008



RL in RoboCup, 1998-2008
Riedmiller et.al, 2007



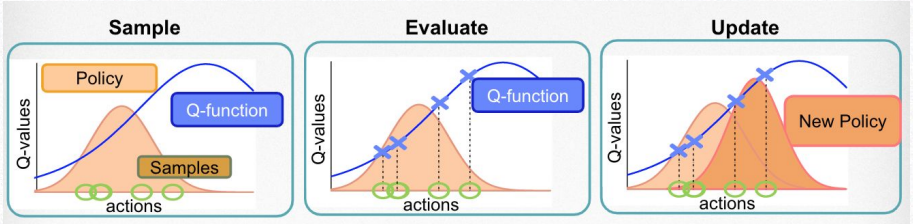
Deep RL from pixels,
Lange and Riedmiller, 2009

model-free, from scratch

riedmiller@google.com

Memory-based, model-free RL beyond NFQ

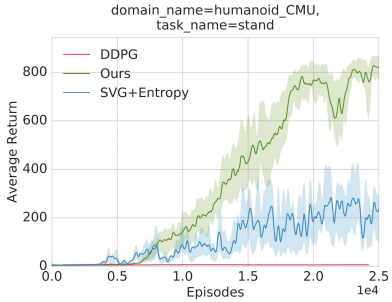
- NFQCA (Hafner and Riedmiller, 2011): continuous actions
  actor-critic uses dQ/da to update actor
- DQN (Mnih et.al, 2014): huge transition sets
  mini-batch updates, target network, convolutional layers
- DDPG (Lillicrap et.al, 2015): huge transition sets, continuous actions
- SVG (Heess et.al, 2015): stochastic policies
- Soft-Actor-Critic (Haarnoja et.al 2018): stochastic policies
- MPO (Maximum a posteriori policy optimisation; Abdolmaleki et.al, 2018, 2019):
  - policy improvement by sampling
  - KL constraints on mean and variance
  - Q-learning as before



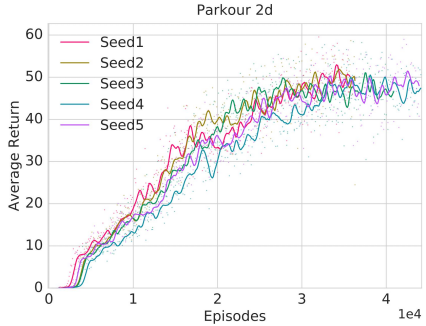
riedmiller@google.com

Example results MPO (Abdolmaleki et.al, 2018, 2019)

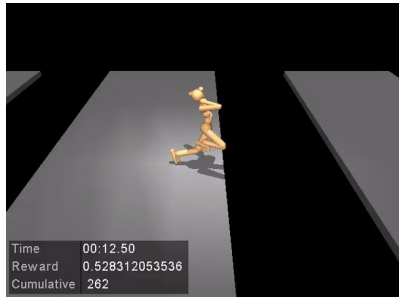
- Humanoid, action dim: 22, state dim: 534
- 2017: 128 actors, several days
- MPO: 1 actor, 1 learner, < 2 days



data efficiency



reliability



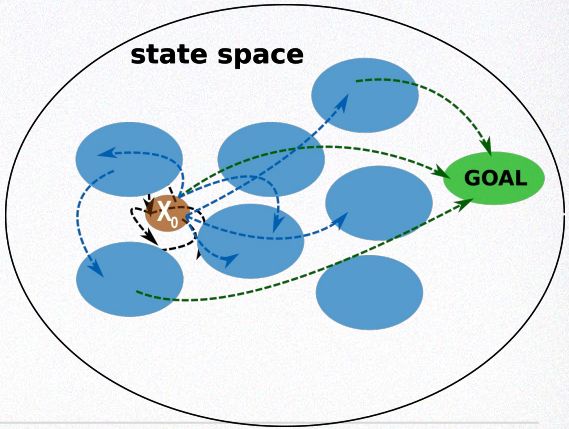
*Collect and Infer - **Collect**:*
Collecting the ‘right’ data

Learning by Playing: Scheduled Auxiliary Control

(Riedmiller, Hafner, et.al, 2018)

What to do when goal is far and rewards are sparse?

- babys play: learn to activate their senses deliberately
- robots play: deliberately activate sensors: touch, move objects, arrange objects, …



riedmiller@google.com

Scheduled Auxiliary Control (SAC-X) main principles

- Extrinsic intention + intrinsic/ auxiliaries intentions : reward vector

$$T = \{\mathcal{M}\} \cup \mathcal{A} \; with \; \mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$$
$$r_{\mathcal{A}_1}(s,a), \ldots, r_{\mathcal{A}_K}(s,a), r_{\mathcal{M}}(s,a)$$

- Execute all intentions during learning

$$\pi_\theta(a|s, \mathcal{T}) \; with \; \mathcal{T} \in T$$

- Put all data in one transition memory and share transitions
(related: HER, Andrychowicz et.al, 2017)

$$(s_t, a_t, \mathcal{T}_t, s_{t+1}, r_{\mathcal{T}_t})$$
$$(s_t, a_t, \hat{\mathcal{T}}, s_{t+1}, r_{\hat{\mathcal{T}}}) \; for \; \hat{\mathcal{T}} \in T$$

- Learn all intentions in parallel by off-policy RL method (e.g. MPO)

$$\arg\max_\theta \mathcal{L}(\theta)$$
$$\mathcal{L}(\theta) = \mathcal{L}(\theta; \mathcal{M}) + \sum_{k=1}^{|A|} \mathcal{L}(\theta; \mathcal{A}_k)$$

Meta-learning in Scheduled Auxiliary Control: SAC-Q

- in general, a random selection of intentions works (SAC-U)
- a learning-to-learn ('meta-learning') approach helps to improve data-efficiency
- SAC-Q scheduler:
  - learn Q function
  - tabular MC approximation conditioned on last intention

$$\mathcal{T}_{0:H-1} = \{\mathcal{T}_0, \ldots, \mathcal{T}_{H-1}\}$$

$$R_{\mathcal{M}}(\mathcal{T}_{0:H-1}) = \sum_{h=0}^{H} \sum_{t=h\xi}^{(h+1)\xi-1} \gamma^t r_{\mathcal{M}}(\mathbf{s}_t, \mathbf{a}_t),$$

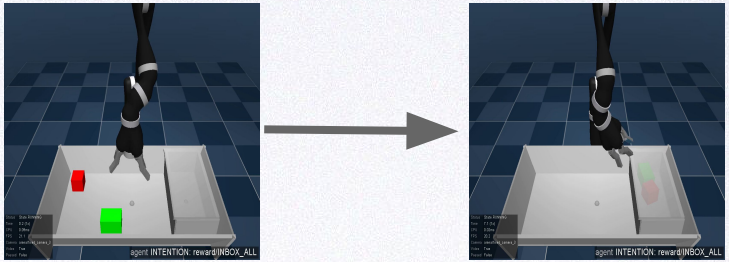
where $\mathbf{a}_t \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_t, \mathcal{T}_h)$.

$$Q(\mathcal{T}_{0:h-1}, \mathcal{T}_h) = \frac{1}{M} \sum_{i=1}^{M} R^{\tau}_{\mathcal{M}}(\mathcal{T}_{h:H}),$$

riedmiller@google.com

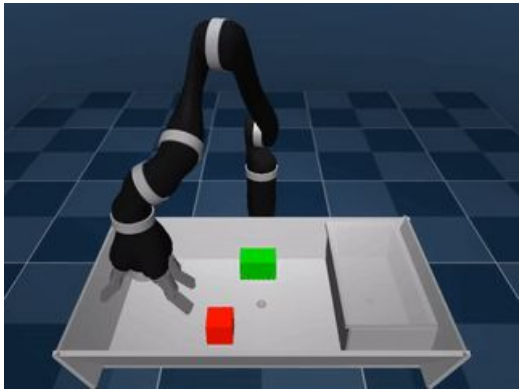
The ‘Cleanup’ task

- 9 DOF continuous control
- 15 sparse auxiliary rewards
  - MOVED(obj)
  - TOUCH, NOTOUCH
  - LEFTOF(obj1, obj2), RIGHTOF(obj1,obj2), ABOVE(obj1,obj2)
  - CLOSE(obj1, obj2)
- 6 extrinsic task rewards

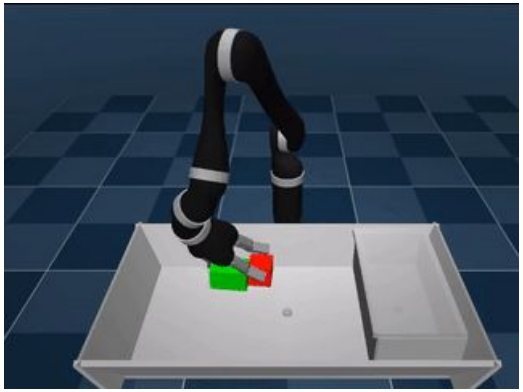


riedmiller@google.com

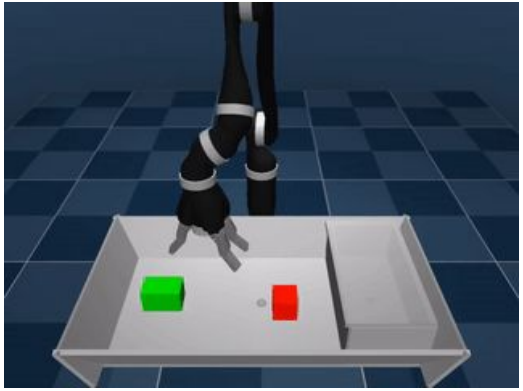
Learning by playing



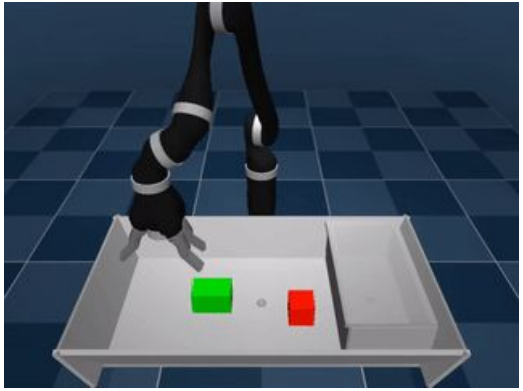
Untrained



Learned to touch and move



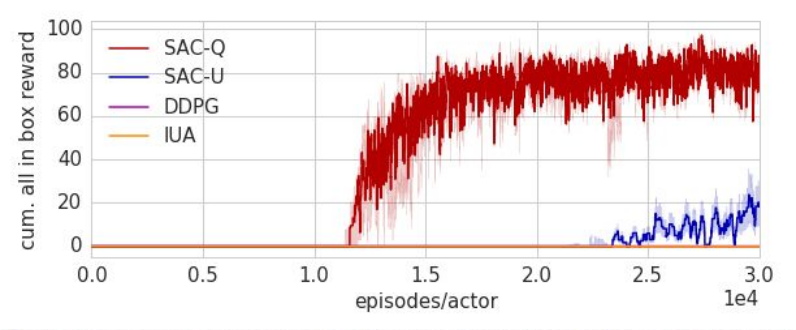
Learned to put objects left and right

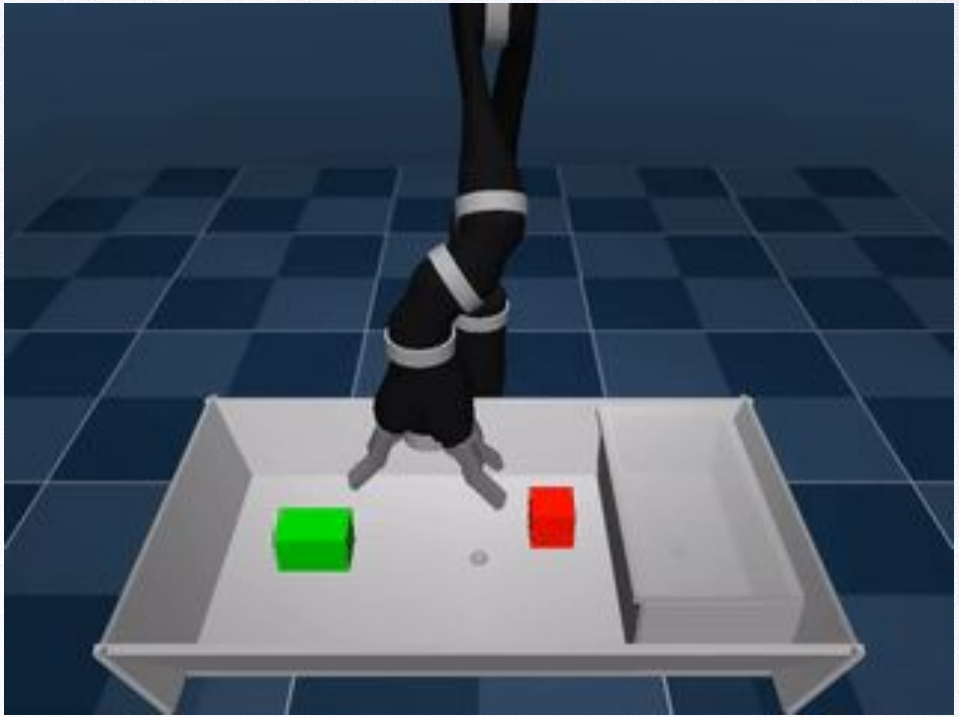


Learned to stack objects

The ‘Cleanup’ task - final policy

- all ‘flat’ RL approaches fail
- SAC-Q 20,000 episodes with 36 actors

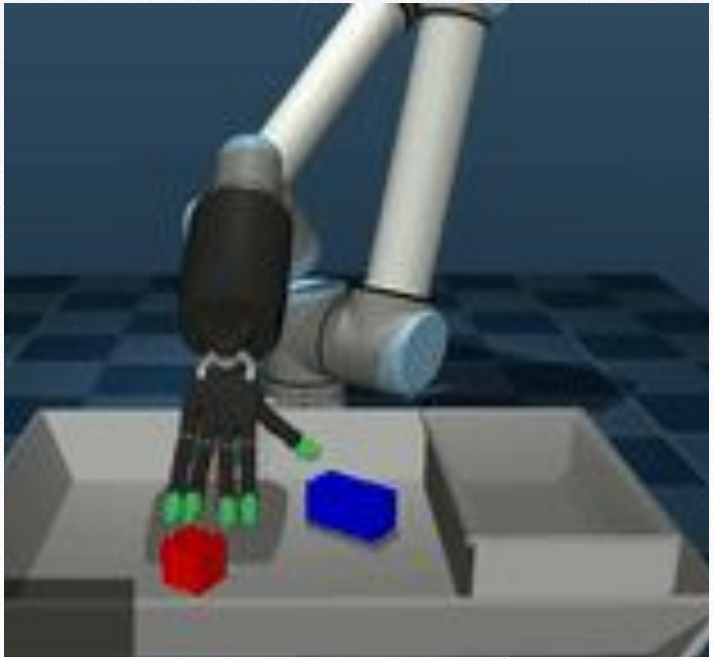




riedmiller@google.com

Generality: different robot

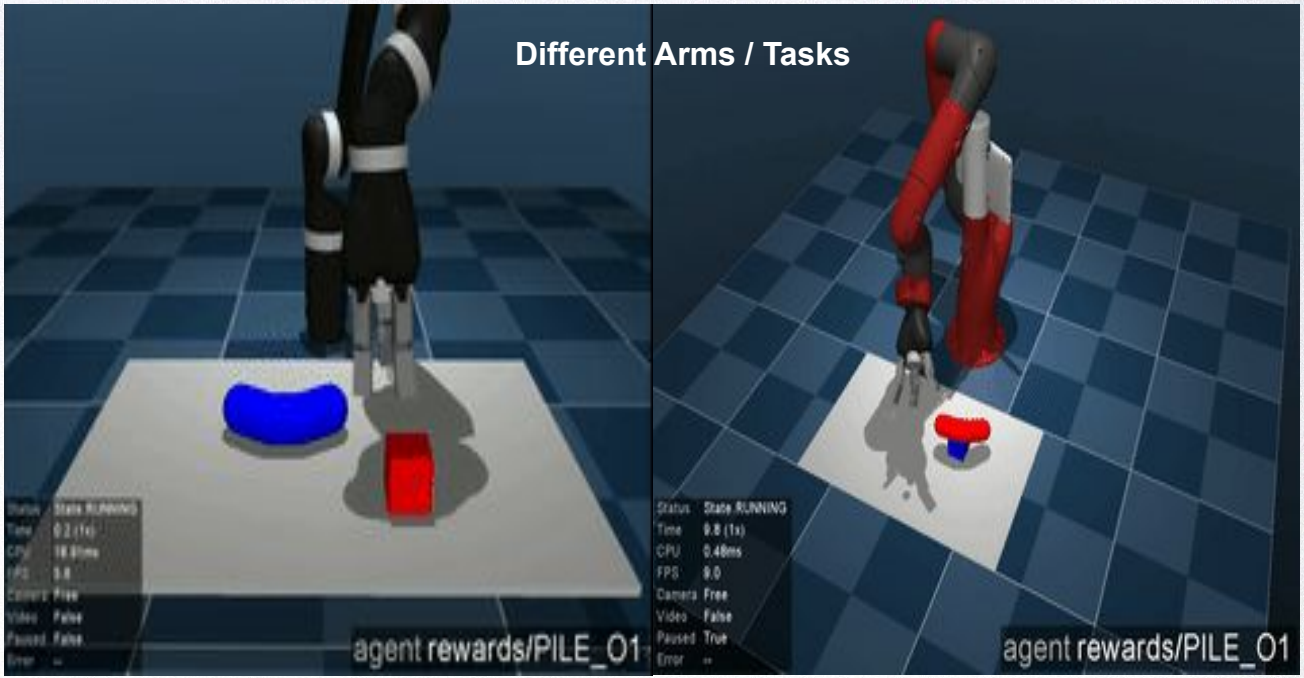
- same auxiliary rewards are used as before



riedmiller@google.com

Generality: varying environments

- same auxiliaries as before



riedmiller@google.com

SAC-X applied to general control domains

- additional auxiliary rewards; stability points: up-down, down-down, down-up, up-up
- accelerates learning for up-up + can solve different tasks

riedmiller@google.com

Intermediate summary

- collect and infer is a powerful principle: offline RL algorithms based on transition memories scale surprisingly well, if right data is collected
- research directions:
  - efficiency, e.g. through better exploration, richer policy classes (hybrid MPO, Neunert et.al, 2019), hierarchies (RHPO, Wulfmeier et.al, 2019), model based support (embed2control, IVG, Byvaran et.al 2019), computationally intense updates (treePI, Springenberg et. al, 2020)
  - less prior knowledge, e.g. raw signals: learning from pixels, simple reward functions, ...
  - extending application cases, e.g. learning from pre-recorded batches of data (batch RL), constraints, ...

riedmiller@google.com

Research directions (1): use of learned models

The use of learned models

Philosophy: learned models to increase data-efficiency by generalisation

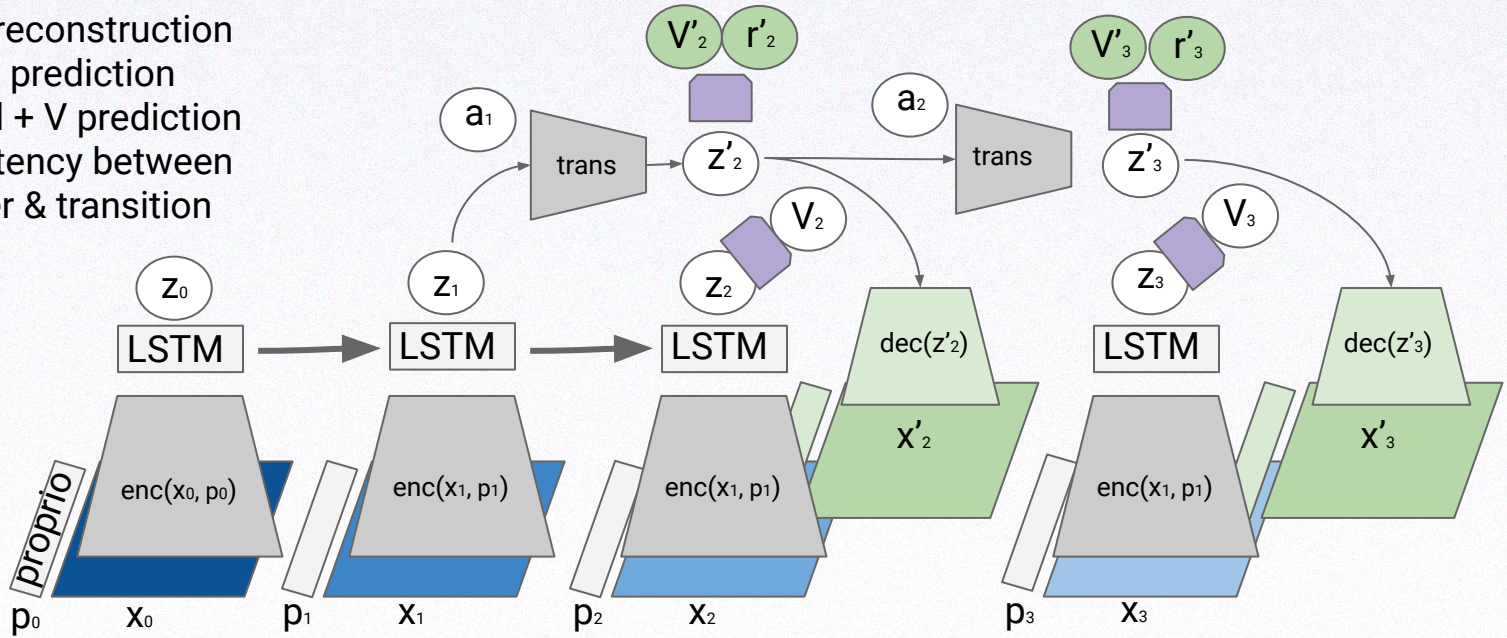
- actor: better action selection through planning (e.g. Embed 2 Control, Watter et. al, 2016)
- learner: improve learning, e.g. IVG: better gradients (Byravan et.al, 2019)
    1. train model from data (various losses)
    2. train policy by backpropagating value estimates through model

Imagined Value Gradients (IVG) (Byravan et.al, 2019)

- **model to predict observations, r, and V**

Training losses:

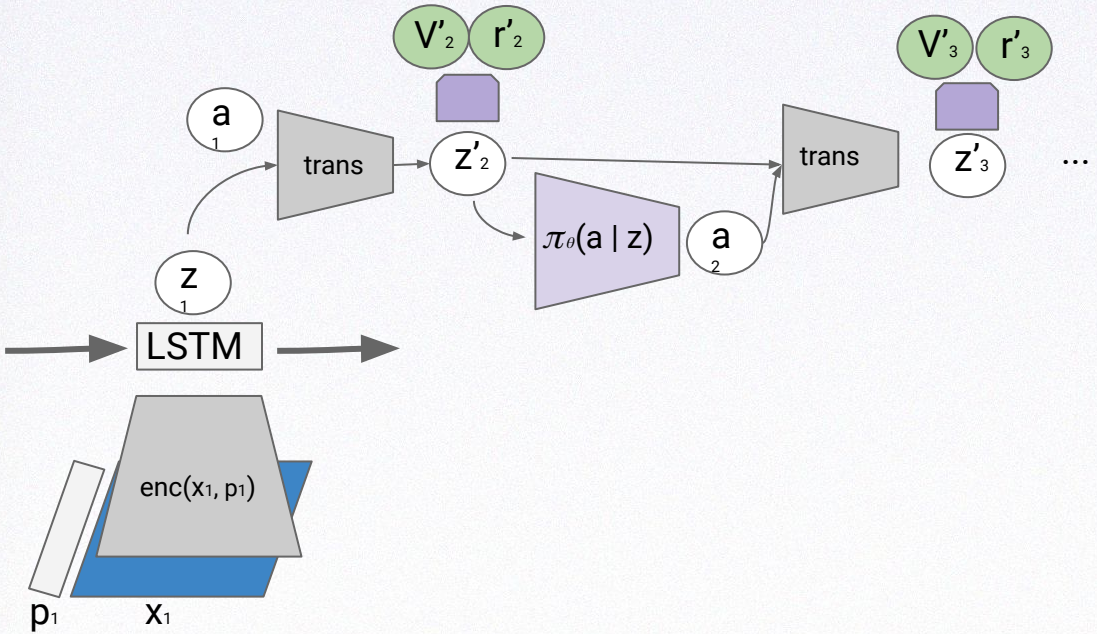
- Image reconstruction
- Proprio prediction
- Reward + V prediction
- Consistency between encoder & transition model



DeepMind

Integrating model with SVG(n) -- Policy loss

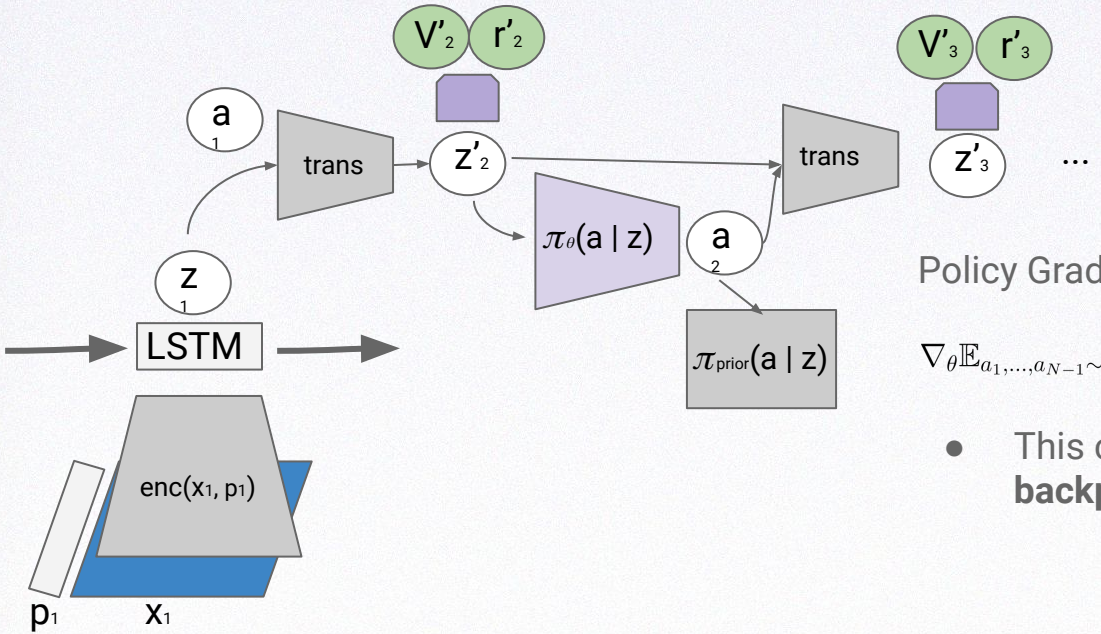
- **Then Train policy with regularized imagined rollouts alongside!**



DeepMind

Integrating model with SVG(n) -- Policy loss

- **Then Train policy with regularized imagined rollouts!**



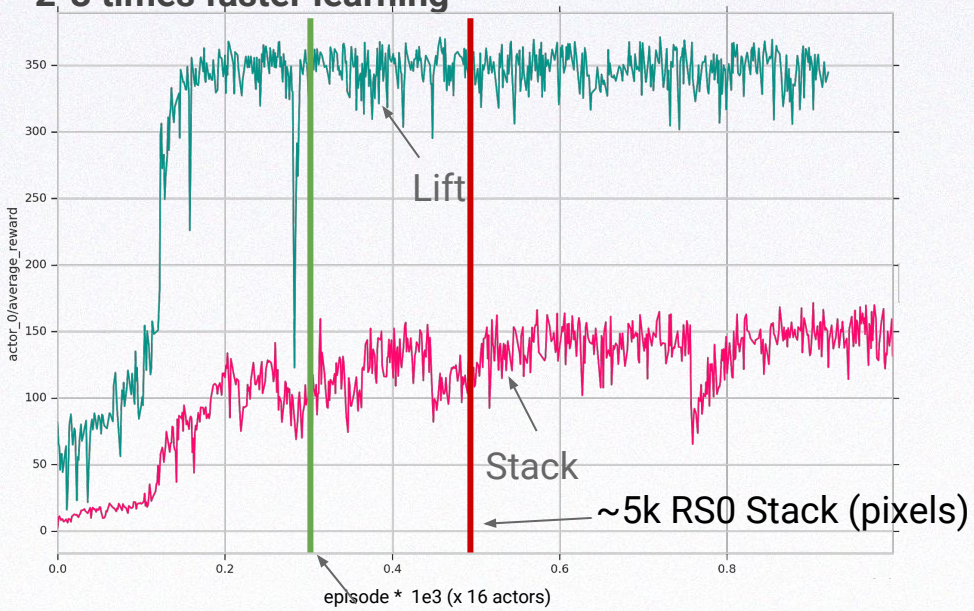
Policy Gradient of KL Regularized Objective:

$$\nabla_\theta \mathbb{E}_{a_1,\ldots,a_{N-1}\sim\pi_\theta}\left[\gamma^T V'_\theta(z'_N) + \sum_{t=0}^{N-1}\gamma^t\left(r'_t - \alpha log\frac{\pi_\theta(a_t|z'_t)}{\pi_{\text{prior}}(a_t|z'_t)}\right)\right]$$

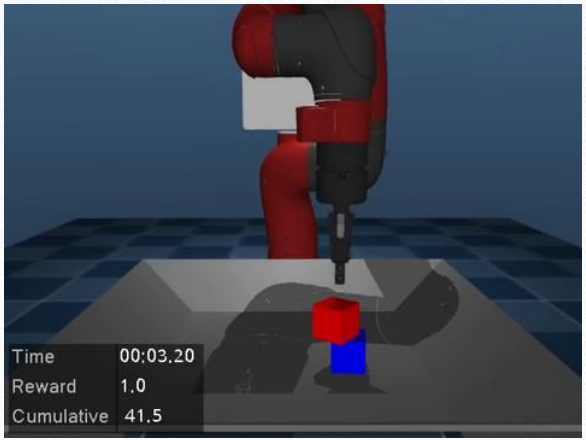
- This can be calculated analytically via **backprop through the model**!

IVG results

- **lift and stack from scratch**
- **2-3 times faster learning**



3.5k RS0 Lift (pixels)



DeepMind

Research directions (2): richer policy representations

Hybrid MPO: mixed continuos/ discrete policies

Neunert et.al, 2019

- hybrid policies using Gaussian and categorical distributions
- fits nicely into MPO policy improvement scheme: ‘hybrid MPO’

  (Q-learning remains unchanged)

$$\pi_\theta(a|s) = \pi^c_\theta(a^{\mathcal{C}}|s)\pi^d_\theta(a^{\mathcal{D}}|s) = \prod_{a^i \in a^{\mathcal{C}}} \pi^c_\theta(a^i|s) \prod_{a^i \in a^{\mathcal{D}}} \pi^d_\theta(a^i|s)$$

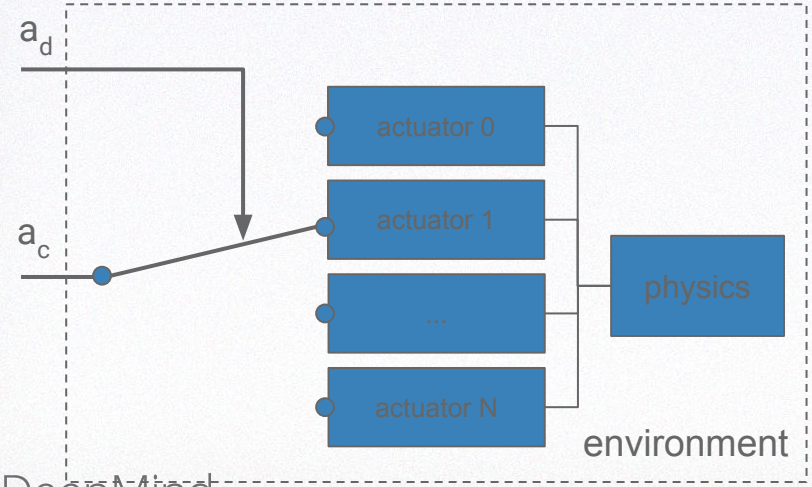
$$\pi^c_\theta(a^i|s) = \mathcal{N}(\mu_{i,\theta}(s), \sigma^2_{i,\theta}(s))$$

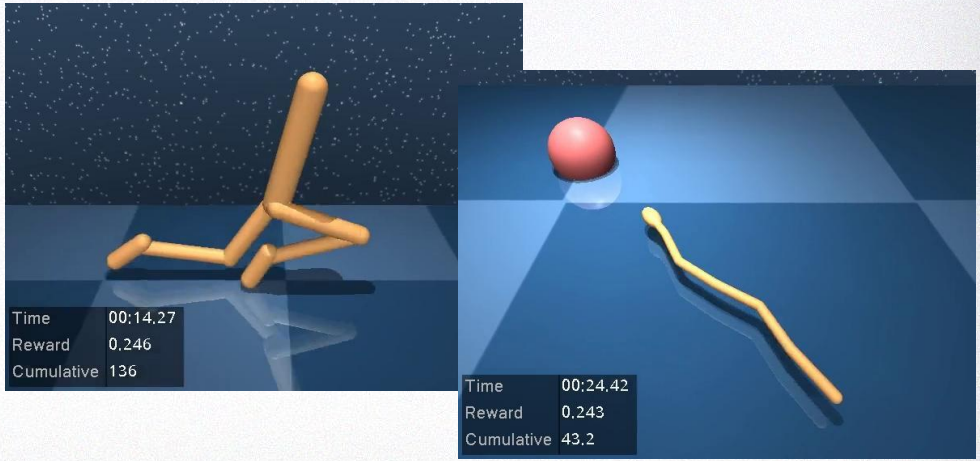
$$\pi^d_\theta(a^i|s) = \mathrm{Cat}^i(\alpha^i_\theta(s))$$

riedmiller@google.com

Hybrid MPO - results

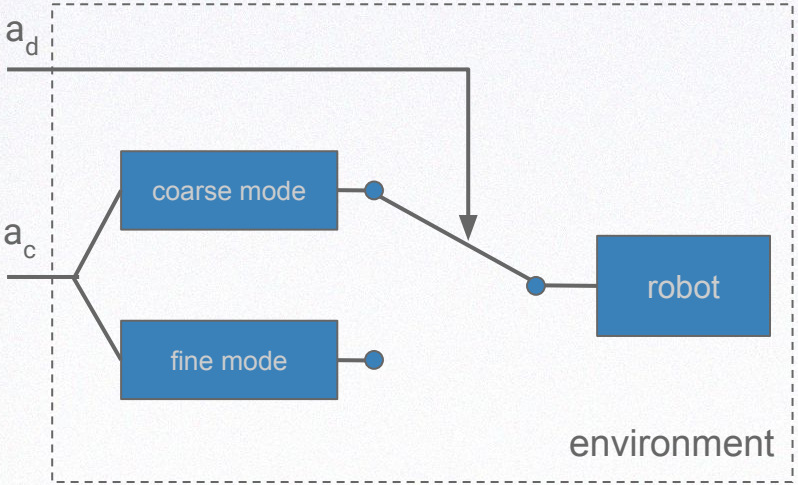
- "Action attention" applied to control suite
- 2-dimensional agent action space:
  - 1-dimensional **continuous actuator activation**
  - 1-dimensional **discrete actuator selection**
- Due to loss of control authority, the agent is finding alternative solutions.

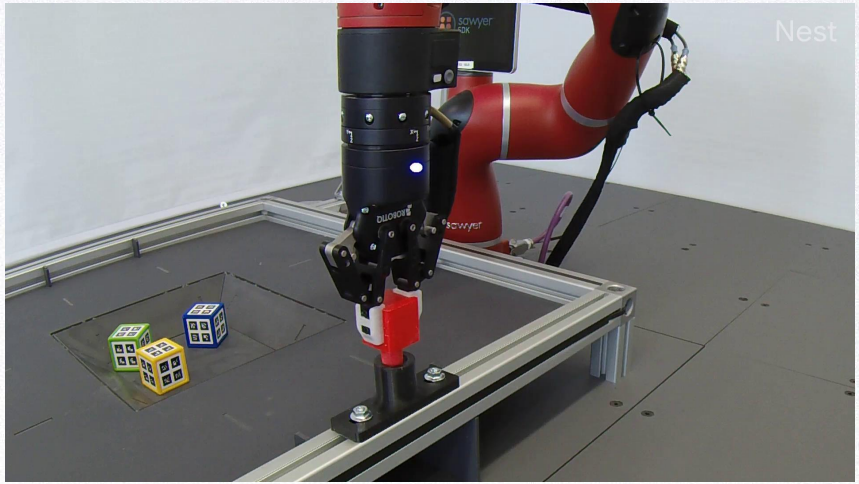




Hybrid MPO - results

- Real robot insertion task
- Two control modes: Coarse and fine
- Agent learns state-dependent switching and control law



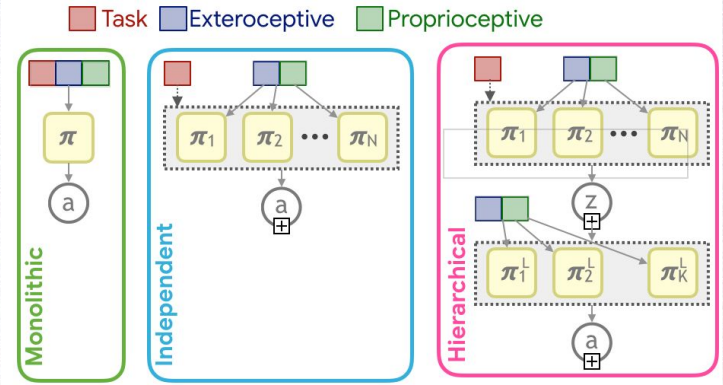


RHPO: Multi Task and Hierarchical Learning

Wulfmeier et.al, 2019

- learn multiple tasks by a single agent
- idea: categorical selection of gaussian policy + task id is hidden from low-level policies enforces learning of re-usable sub-skills
- significant increase in data-efficiency in multi-task settings

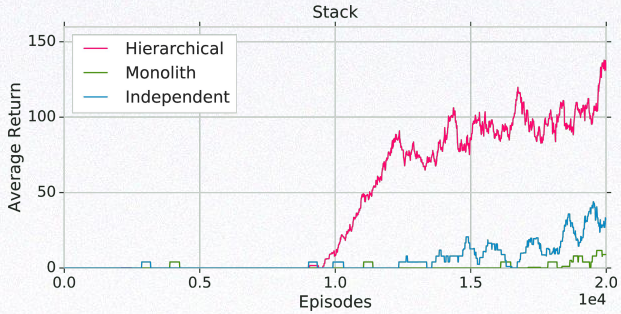
$$\pi_\theta(a|s,i) = \sum_{z=1}^{M} \pi_\theta^H(z|s,i)\,\pi_\theta^L(a|s,z)$$



riedmiller@google.com

Real Robot - Pile Cube with SAC-U and RHPO

- Learning to pile a cube on real robot
- 5 DOF, from scratch
- auxiliaries: reach, grasp, lift, place, stack



DeepMind

Final performance

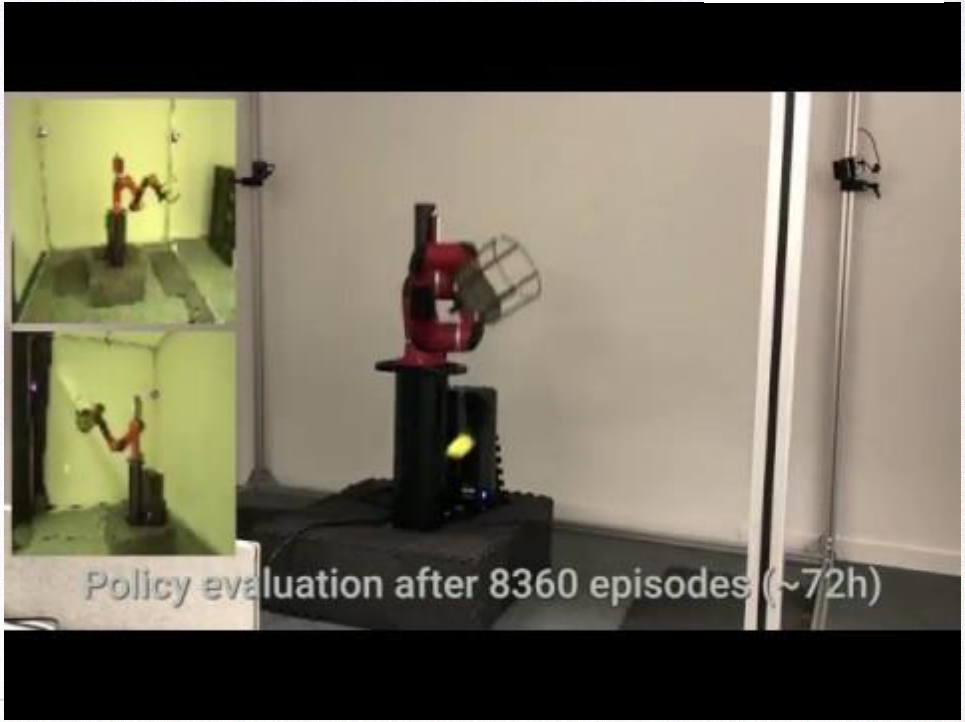
- RHPO + SAC-X: trained from raw pixels in <10 days directly on real robot.

DeepMind

Research directions (3): training from raw inputs

Ball-in-Cup from pixels (Schwab et.al, 2019)

- 4 DOFs, raw joint control
- raw pixels as input
- prior work (Peters et.al 2011): ‘kinesthetic teaching’
- prior work (Schwab et.al 2019): asymetric actor critic: privileged information for learning.
- now: learn directly from raw pixels only! (Improved network structures, improved SAC-Q)



riedmiller@google.com

Conclusion: AGI for Control (AGCI)

- we're aiming at the far end of learning control: AGCI
  ' a minimum set of maximal general priors'
  raw inputs, raw outputs, pure task specification, yet data-efficient
- guided by our principle of 'collect & infer', we can go surprisingly far
- the less we put in a priori, the more creative the solutions get
  interesting read: Sutton's 'bitter lessons'
- wishlist: reliability/ stability proofs, interpretability, sound benchmarks, comparison to classical control

riedmiller@google.com

Supplementary slides

Candidates for Internal Reward Predicates

- abstract principle: deliberately change sensor values
- there is a broad range of concrete implementations:
  from manipulating raw sensor values (very general) to features
- practical realisation:
  - finding a compromise between generality/ simplicty and realistic learning times
- example predicates:
  - activate touch sensor: Touch/ NoTouch
  - camera: relation between objects: e.g. Moved, LeftOf, Above, Near, ...

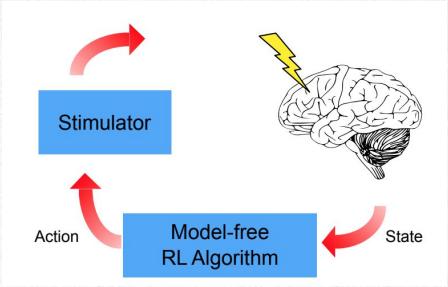
riedmiller@google.com

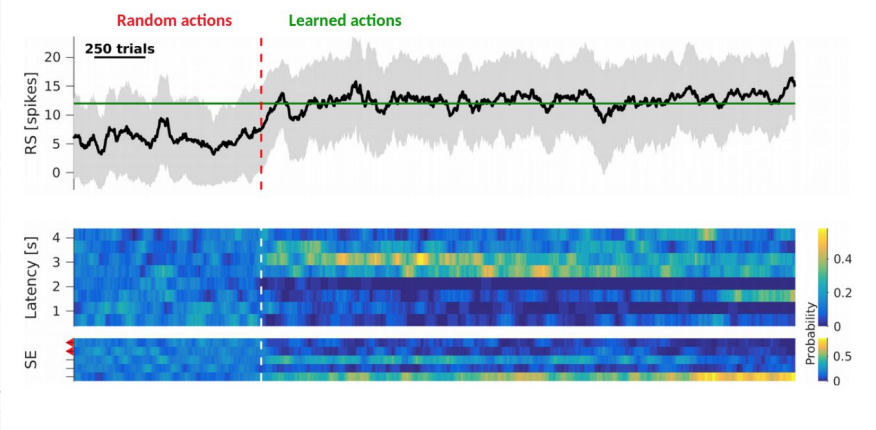
A recent result with NFQ (Wülfing et.al, 2018)

Controlling a set of **real** biological neurons

(BrainLinks BrainTools)

- control signal: place, time and amplitude of stimuli
- target: get desired neural activity
- no model available
- Epilepsy, Alzheimer, ...

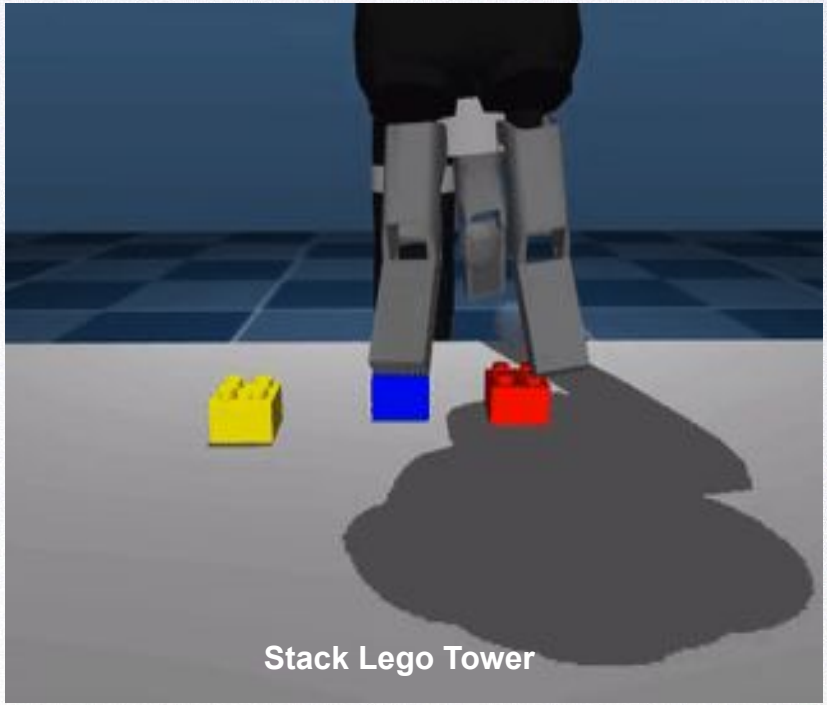




riedmiller@google.com

Generality: varying enviroments

- same auxiliary tasks are used as before
- special adaptations: ‘above close precise’ to enforce incentive for seeing ‘clicks’



riedmiller@google.com

Research directions (3): training from fixed data (Batch RL)

Learning Behavioral Cloning Policies

Optimizer wants to maximize functions:

Behavioral Cloning

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\log \pi_\phi^{\text{bc}}(a|s)\right]$$

Advantage-Weighted Behavioral Cloning

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\log \pi_\phi^{\text{bc}}(a|s)\cdot f(A(s,a))\right]$$

Keep Doing What Worked

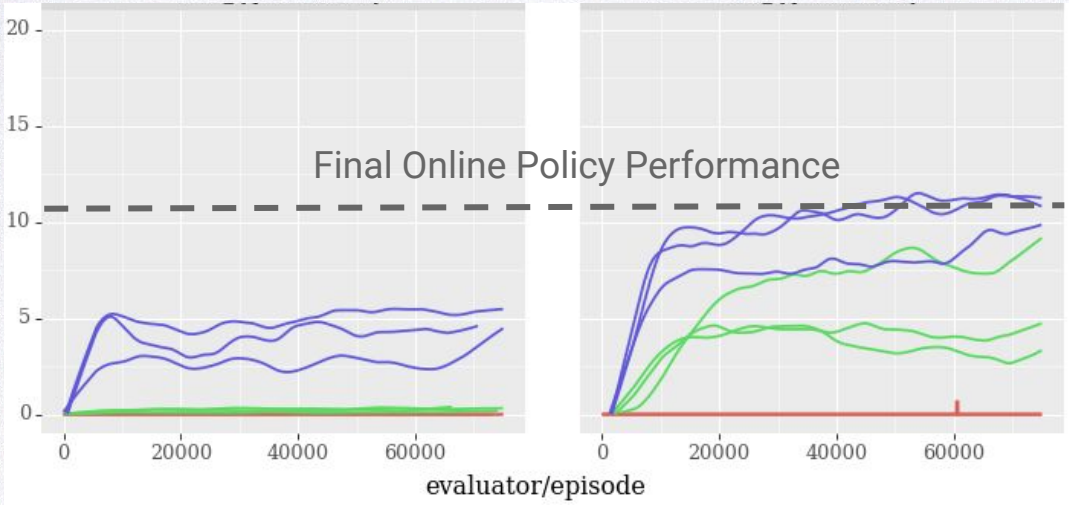
We use $f(x) = 1_+(A)$

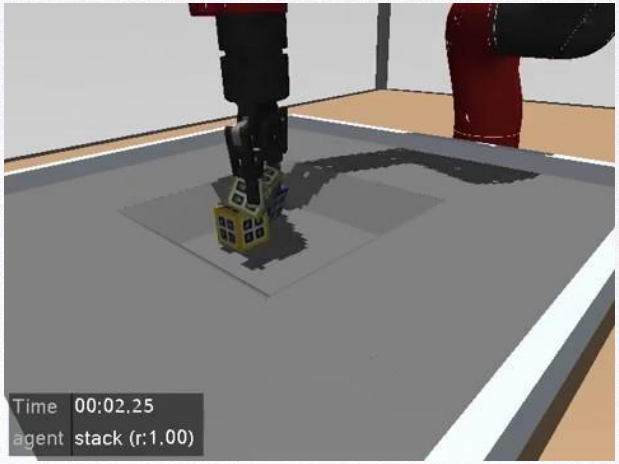
riedmiller@google.com

Block Stacking in Sim, Main Task Performance

Prior (BC) Policy

RL Policy





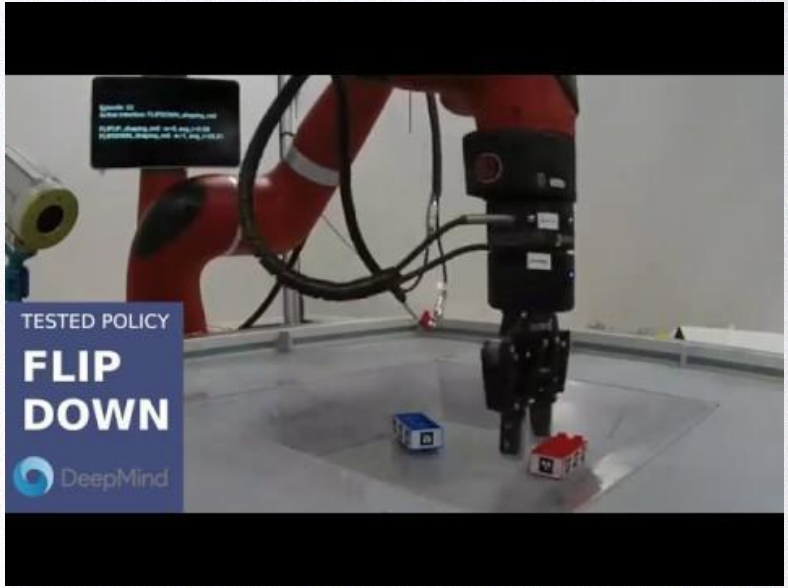
No BC Prior, BC Prior, Advantage-Weighted BC Prior

riedmiller@google.com

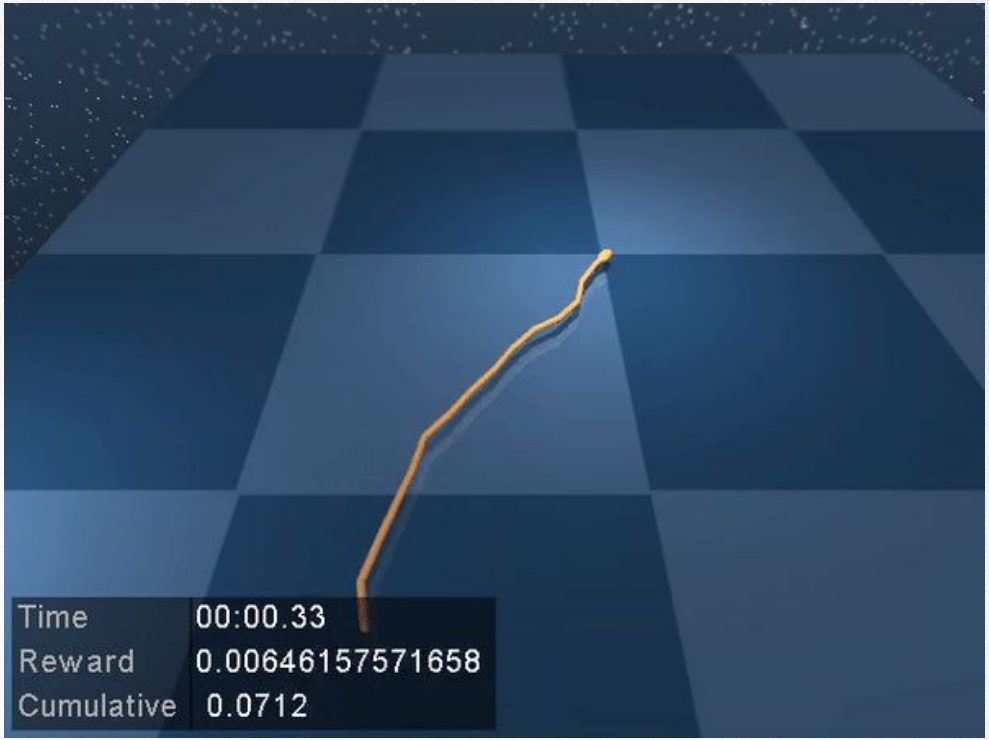
Reward augmentation in off-policy - learning

Real Robot

- Several days of data from learning to grasp
- Trained FlipUp, FlipDown skills **purely offline by reward augmentation**
- ‘creative’ **solutions**: "pushing", "poking", "dropping" using wall and other brick



- 15 dim action space
- dynamics
- model-free, from scratch



riedmiller@google.com

Problem Statement

Iterate :

Policy Evaluation (learning a Q-function or model ...)

Policy Improvement (MPO)

Only accessible data : Samples of Q values given a state,action

Goal: Next policy gives better probability to better actions

MPO and V-MPO

- MPO and V-MPO are simple algorithms for policy improvement given a policy , a state distribution and an evaluation function

- MPO : when we can evaluate multiple actions given a state, e.g:
    1. Access to Q-function
    2. Access to model with state reset

- V-MPO: when we can evaluate only one action given a state e.g:
    1. On-policy setting or transitions from replay buffer
    2. Expensive compute (still use model or Q-function)

Maximum A Posteriori Policy Optimisation

Given Q(s,a) and state distribution

1-Step :

Find non-parametric distribution:

$$q(a|s) \propto \pi_{\text{old}}(a|s) \exp\left(\frac{Q(s,a)}{\eta}\right)$$

Step 2 :

1. Supervised learning:

$$\theta_{\text{new}} = \arg\min_{\theta} \int \mu(s) \text{KL}\big(q(a|s) \| \pi_\theta(a|s)\big) \text{d}s$$

riedmiller@google.com

Step (1) MPO

$$\max_q \int \mu(s) \int q(a|s) Q(s,a) ds da$$

$$s.t. \int \mu(s) \mathrm{KL}(q(a|s), \pi(a|s,\theta_t)) ds < \epsilon$$

Closed Form Solution:

1. Sample actions from current policy
2. Weight action samples via Q values
3. Note that this standard RL objective
4. Assuming parametric q would lead to policy gradient methods

sampling weights

$$q(a|s) \propto \pi(a|s,\theta_t) \exp(\frac{Q(a,s)}{\eta})$$

Step (2) MPO

Objective: Fit a policy $\pi(a|s;\theta)$ to non-parametric distributions, i.e,

sampling weights

$$q(a|s) \propto \pi(a|s,\theta_t)\exp(\frac{Q(a,s)}{\eta})$$

Conservative supervised learning:

1. Weighted maximum likelihood

$$\max_\theta \int \mu(s) \int q(a|s) \log \pi(a|s;\theta) dsda$$

2. Bound information loss
   Controls learning rate directly on policy space

$$s.t. \quad \int_s \mu(s) \mathrm{KL}(\pi(a|s;\theta_t), \pi(a|s;\theta)) ds < \epsilon$$