Asynchronous & Parallel Algorithms

Sergey Levine

UC Berkeley

Overview

1. We learned about a number of policy search methods
2. These algorithms have all been *sequential*
3. Is there a natural way to parallelize RL algorithms?
    - Experience sampling vs learning
    - Multiple learning threads
    - Multiple experience collection threads

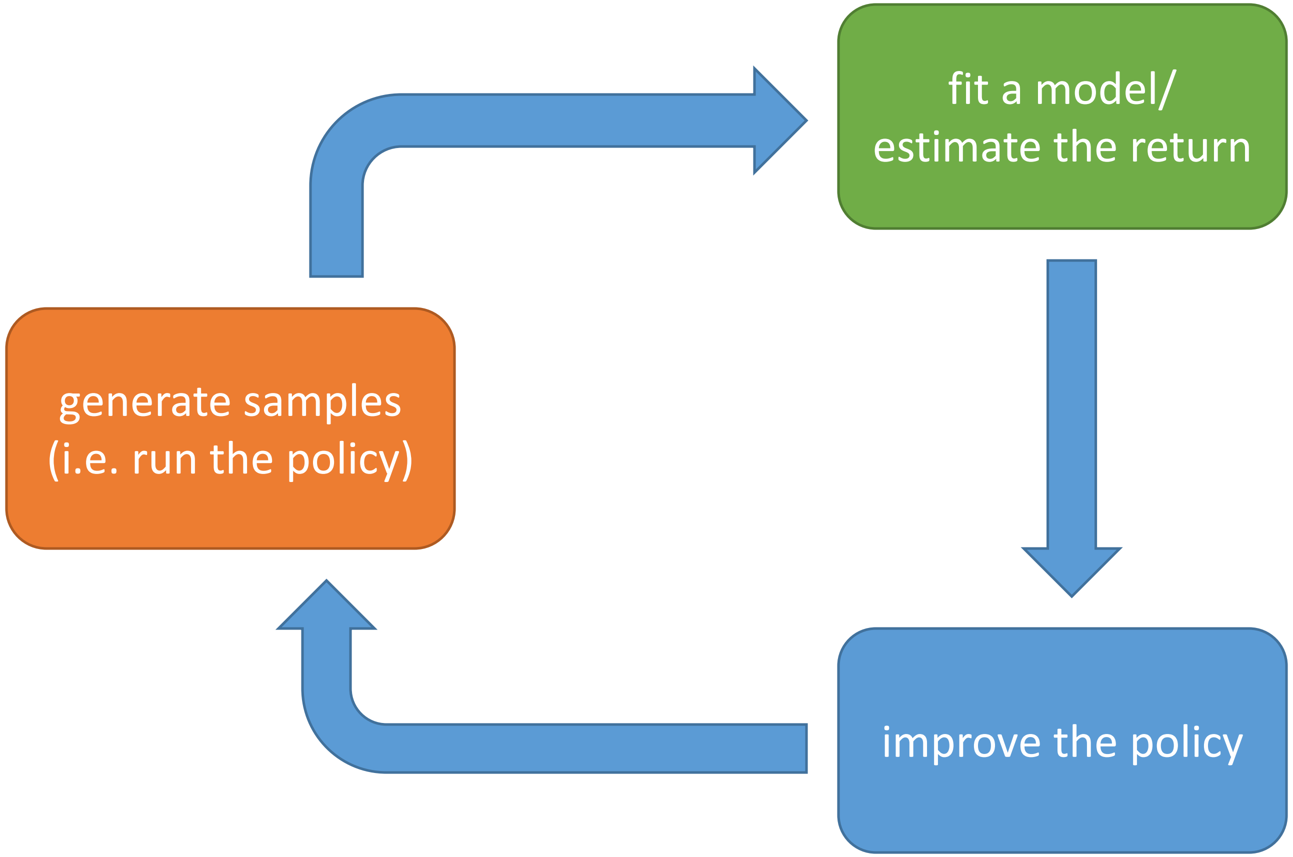
Today's Lecture

1. High-level schematic of a generic RL algorithm
2. What can we parallelize?
3. Case studies: specific parallel RL methods
4. Tradeoffs & considerations

- Goals
  - Understand the high-level anatomy of reinforcement learning algorithms
  - Understand standard strategies for parallelization
  - Tradeoffs of different parallel methods

REMINDER: PROJECT GROUPS DUE TODAY! SEND TITLE & GROUP MEMBERS TO berkeleydeeprlcourse@gmail.com

High-level RL schematic



estimate $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ (model-based)

compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)

fit $Q_\phi(\mathbf{s},\mathbf{a})$ (actor-critic, Q-learning)

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

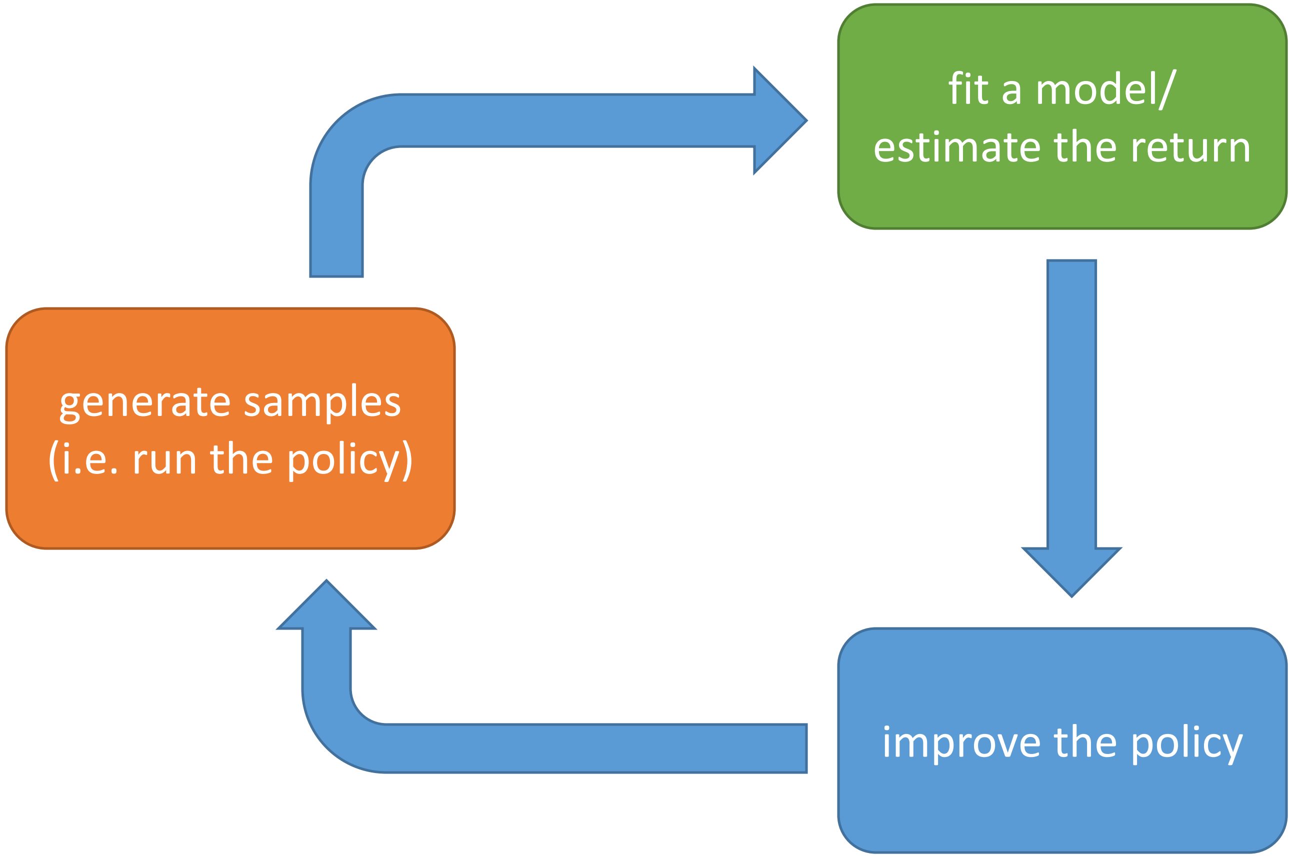
$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)

$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s},\mathbf{a})$ (Q-learning)

Which parts are slow?

real robot/car/power grid/whatever:
1x real time, until we invent time travel

MuJoCo simulator:
up to 10000x real time



$\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$

trivial, fast

fit $Q_\phi(\mathbf{s}, \mathbf{a})$

expensive, but non-trivial to parallelize

$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$

trivial, nothing to do

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

expensive, but non-trivial to parallelize

Which parts can we parallelize?

generate samples
(i.e. run the policy)

fit a model/
estimate the return

fit $Q_\phi(\mathbf{s}, \mathbf{a})$

parallel SGD

improve the policy

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

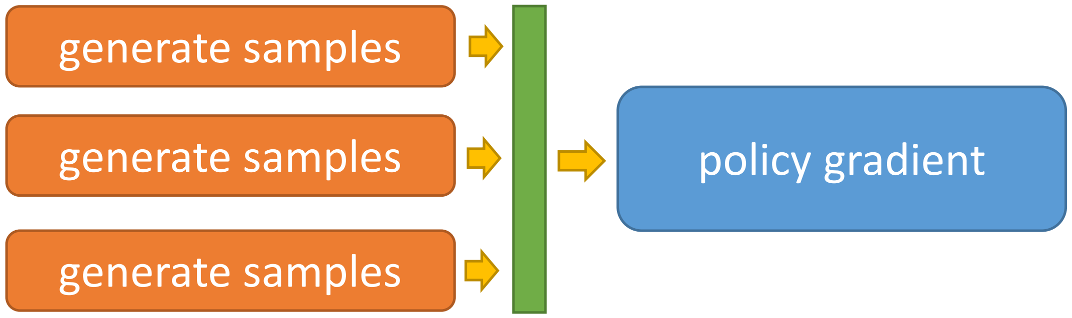
parallel SGD

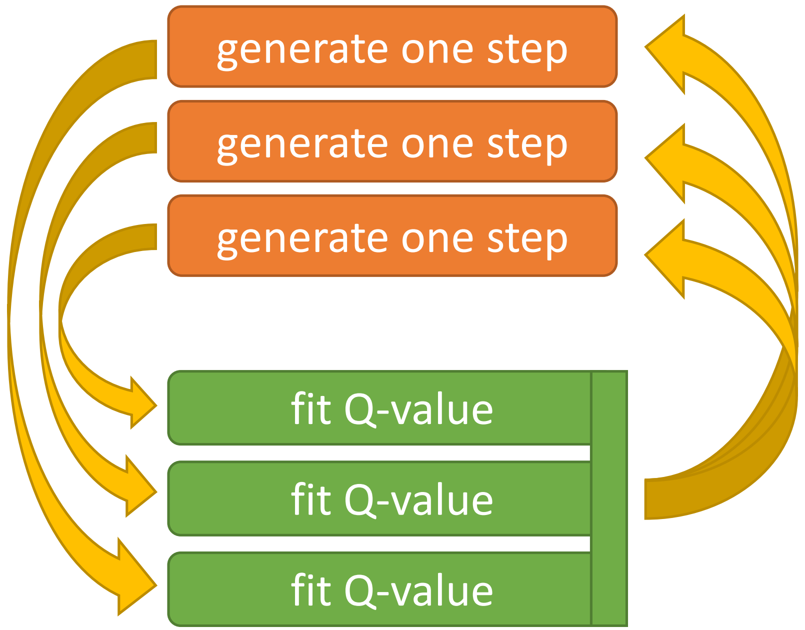
Helps to group data generation and training
(worker generates data, computes gradients, and gradients are pooled)

High-level decisions

1. Online or batch-mode?

2. Synchronous or asynchronous?

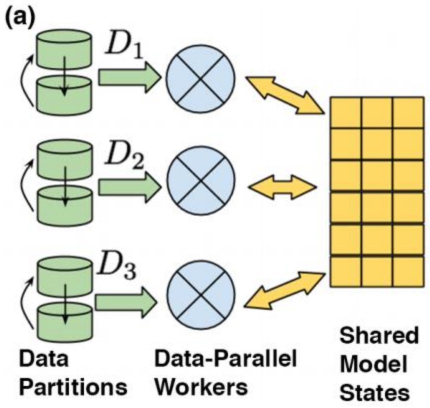




Relationship to parallelized SGD

fit a model/
estimate the return

improve the policy

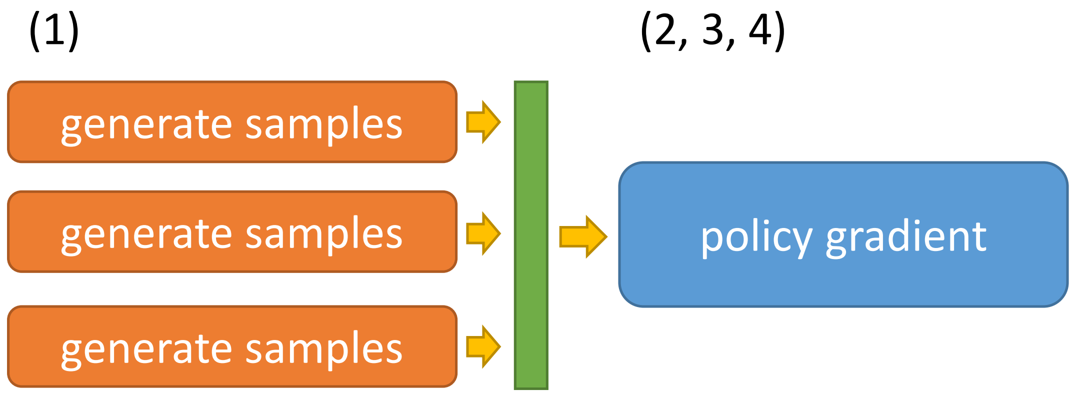


Dai et al. '15

1. Parallelizing model/critic/actor training typically involves parallelizing SGD
2. Simple parallel SGD:
   1. Each worker has a different slice of data
   2. Each worker computes gradients, sums them, sends to parameter server
   3. Parameter server sums gradients from all workers and sends back new parameters
3. Mathematically equivalent to SGD, but not asynchronous (communication delays)
4. Async SGD typically does not achieve perfect parallelism, but lack of locks can make it much faster
5. Somewhat problem dependent

Simple example: sample parallelism with PG

1. collect samples $\tau_i = \{\mathbf{s}_1^i, \mathbf{a}_1^i, \dots, \mathbf{s}_T^i, \mathbf{a}_T^i\}$ by running $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ N times
2. compute $r_i = r(\tau_i)$
3. compute $\nabla_i = \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)(r_i - b)$
4. update: $\theta \leftarrow \theta + \alpha \sum_i \nabla_i$

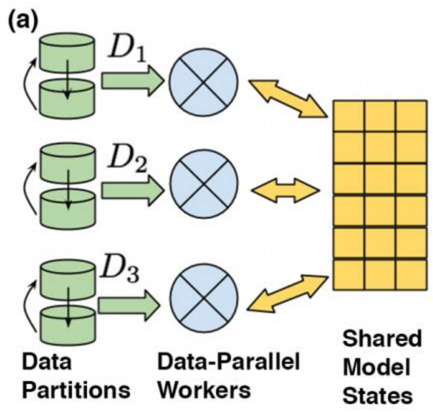


Simple example: sample parallelism with PG

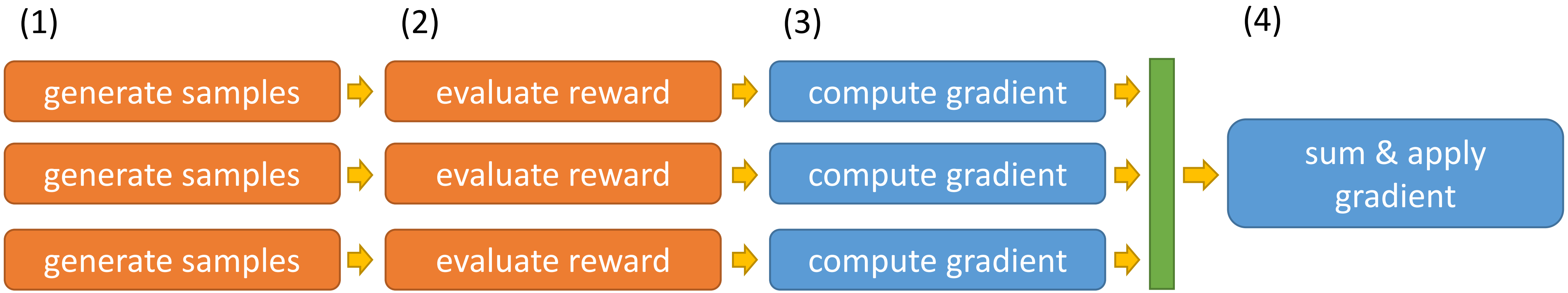
1. collect samples $\tau_i = \{\mathbf{s}_1^i, \mathbf{a}_1^i, \ldots, \mathbf{s}_T^i, \mathbf{a}_T^i\}$ by running $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ N times
2. compute $r_i = r(\tau_i)$
3. compute $\nabla_i = \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)(r_i - b)$
4. update: $\theta \leftarrow \theta + \alpha \sum_i \nabla_i$

Simple example: sample parallelism with PG

1. collect samples $\tau_i = \{\mathbf{s}_1^i, \mathbf{a}_1^i, \ldots, \mathbf{s}_T^i, \mathbf{a}_T^i\}$ by running $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ N times
2. compute $r_i = r(\tau_i)$
3. compute $\nabla_i = \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)(r_i - b)$
4. update: $\theta \leftarrow \theta + \alpha \sum_i \nabla_i$



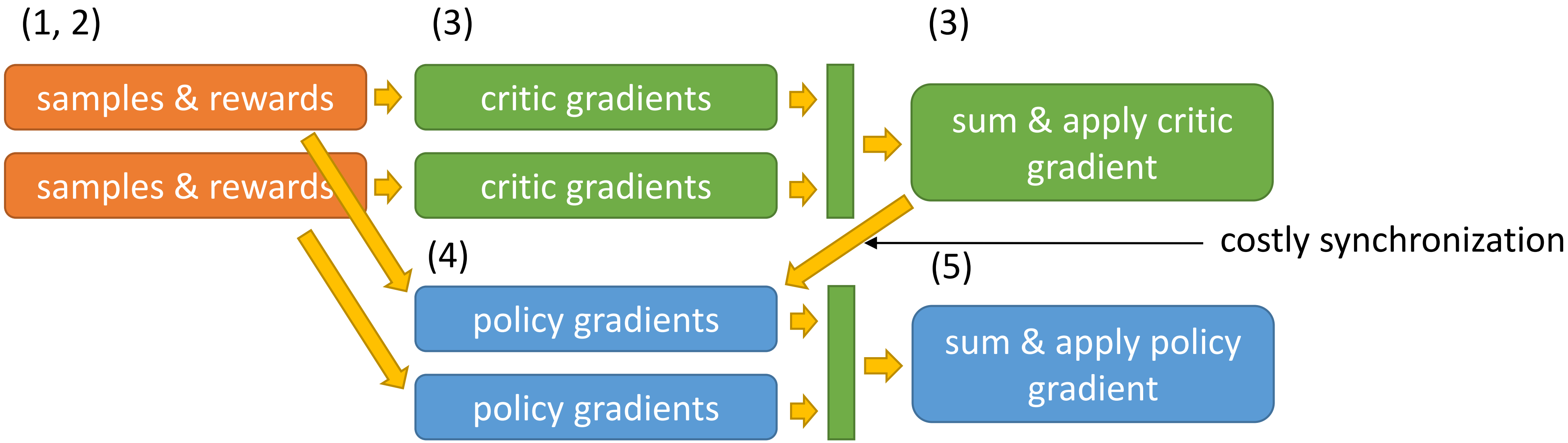
Dai et al. '15



What if we add a critic?

1. collect samples $\tau_i = \{\mathbf{s}_1^i, \mathbf{a}_1^i, \dots, \mathbf{s}_T^i, \mathbf{a}_T^i\}$ by running $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ N times
2. compute $r_i = r(\tau_i)$
3. update $\hat{A}_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i)$ with regression to target values
4. compute $\nabla_i = \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \hat{A}_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i)$
5. update: $\theta \leftarrow \theta + \alpha \sum_i \nabla_i$

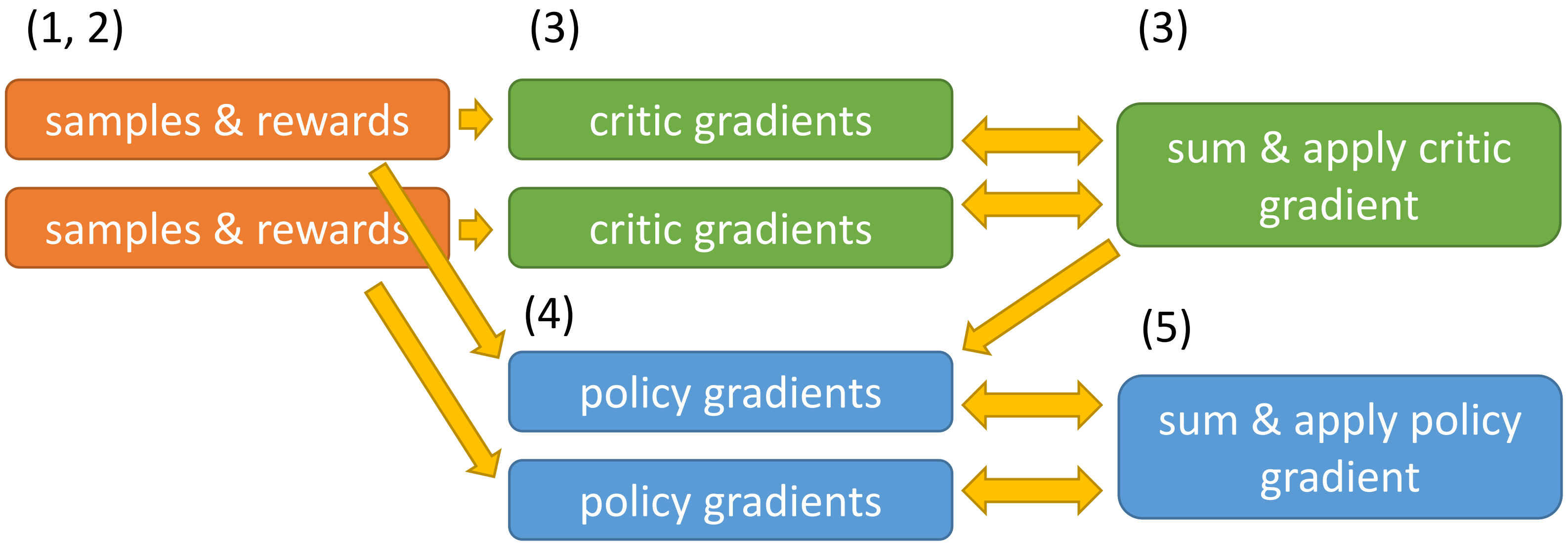
see John's actor-critic lecture for what the options here are



What if we add a critic?

1. collect samples $\tau_i = \{\mathbf{s}_1^i, \mathbf{a}_1^i, \dots, \mathbf{s}_T^i, \mathbf{a}_T^i\}$ by running $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ N times
2. compute $r_i = r(\tau_i)$
3. update $\hat{A}_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i)$ with regression to target values
4. compute $\nabla_i = \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \hat{A}_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i)$
5. update: $\theta \leftarrow \theta + \alpha \sum_i \nabla_i$

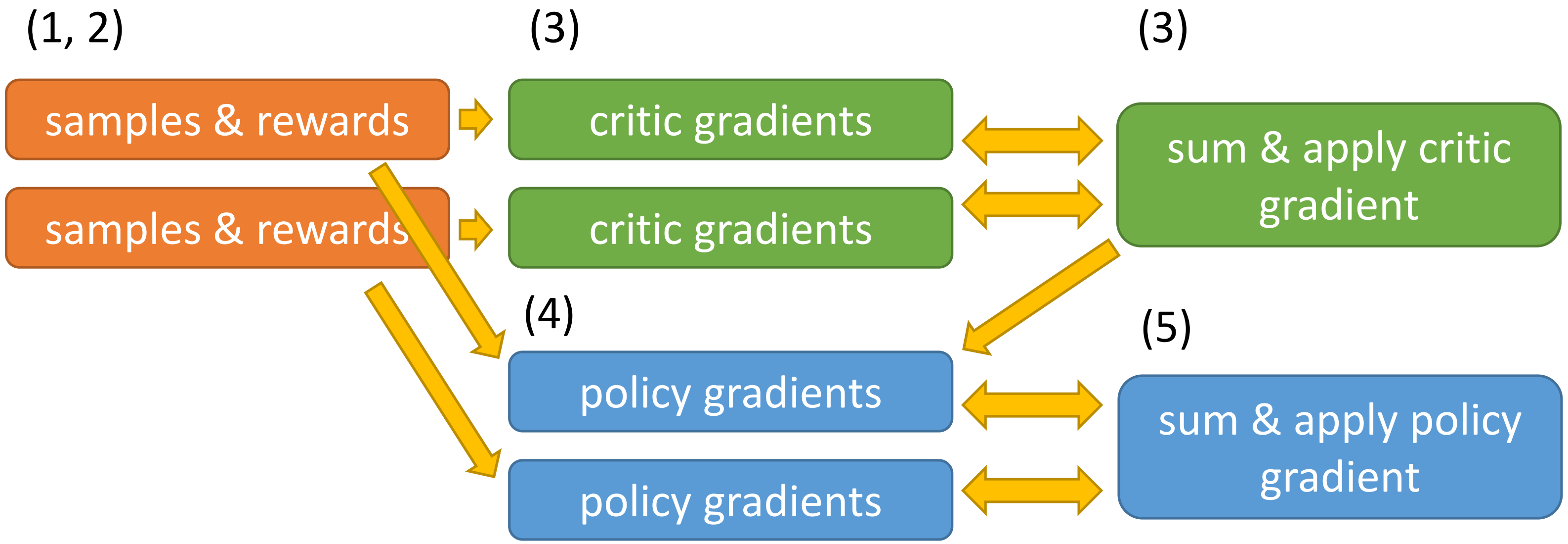
see John's actor-critic lecture for what the options here are



What if we run online?

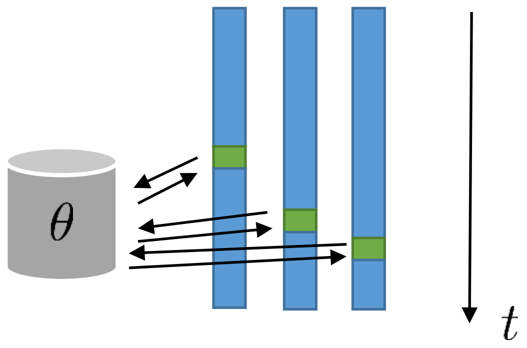
1. collect sample $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)$ by running $\pi_\theta(\mathbf{a}|\mathbf{s})$ for 1 step
2. compute $r_i = r(\mathbf{s}_i, \mathbf{a}_i)$
3. update $\hat{A}_\phi(\mathbf{s}_t^i, \mathbf{a}_t^i)$ with regression to target values
4. compute $\nabla_i = \nabla_\theta \log \pi_\theta(\mathbf{a}^i|\mathbf{s}^i)\hat{A}_\phi(\mathbf{s}^i, \mathbf{a}^i)$
5. update: $\theta \leftarrow \theta + \alpha \sum_i \nabla_i$

only the parameter update requires synchronization (actor + critic params)



Actor-critic algorithm: A3C

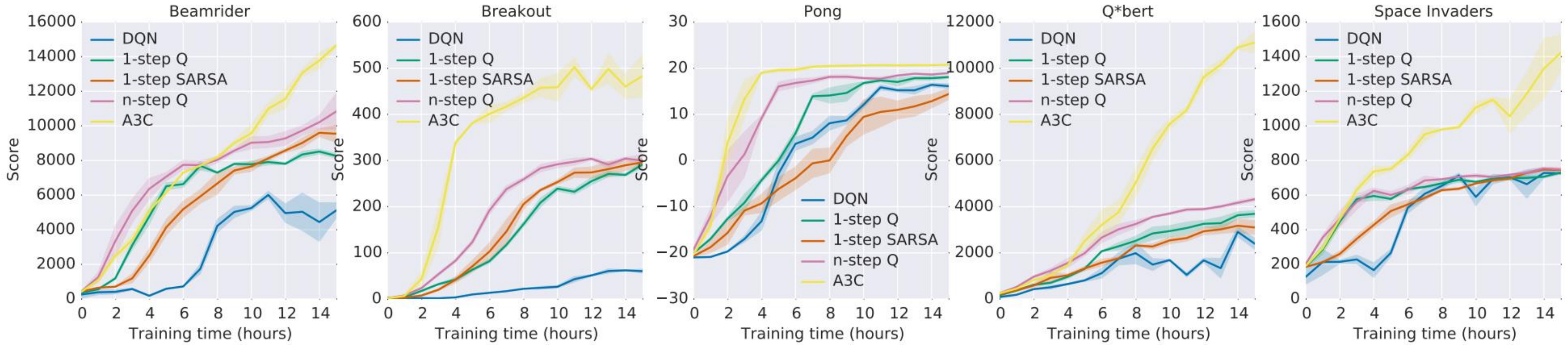
1. collect sample $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)$ by running $\pi_\theta(\mathbf{a}|\mathbf{s})$ for 1 step
2. compute $r_i = r(\mathbf{s}_i, \mathbf{a}_i)$
3. update $\hat{A}_\phi(\mathbf{s}^i_t, \mathbf{a}^i_t)$ with regression to target values
4. compute $\nabla_i = \nabla_\theta \log \pi_\theta(\mathbf{a}^i|\mathbf{s}^i)\hat{A}_\phi(\mathbf{s}^i, \mathbf{a}^i)$
5. update: $\theta \leftarrow \theta + \alpha \sum_i \nabla_i$ (only do this every n steps)

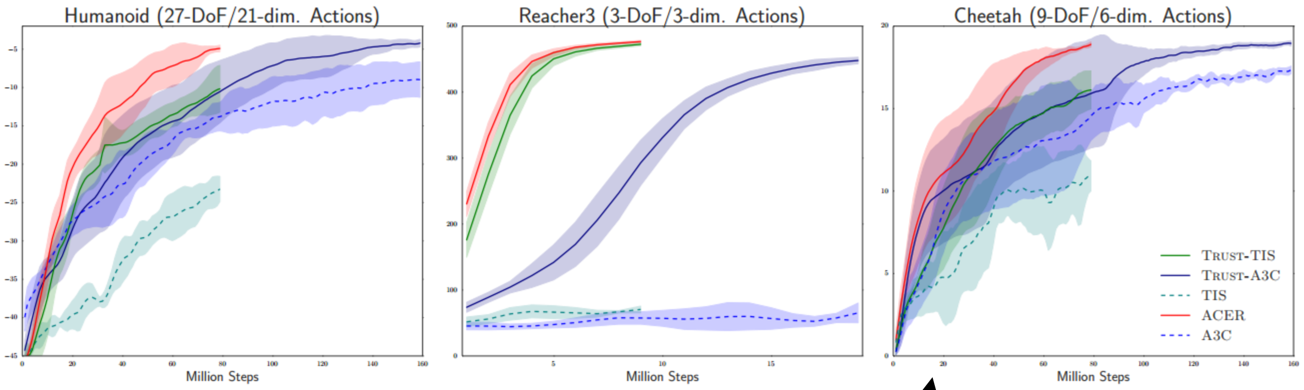


- Some differences vs DQN, DDPG, etc:
  - No replay buffer, instead rely on diversity of samples from different workers to decorrelate
  - Some variability in exploration between workers
- Pro: generally much faster in terms of wall clock
- Con: generally must slower in terms of # of samples (more on this later…)

Mnih et al. ‘16

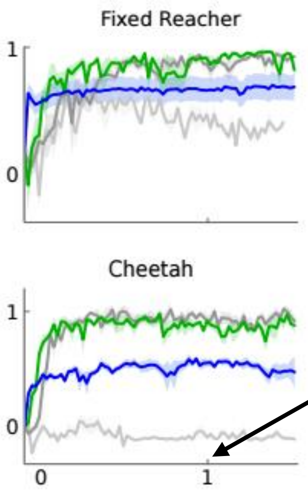
Actor-critic algorithm: A3C





20,000,000 steps

DDPG:

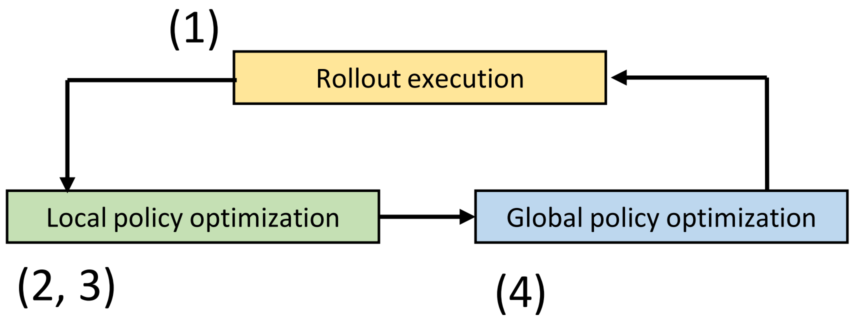


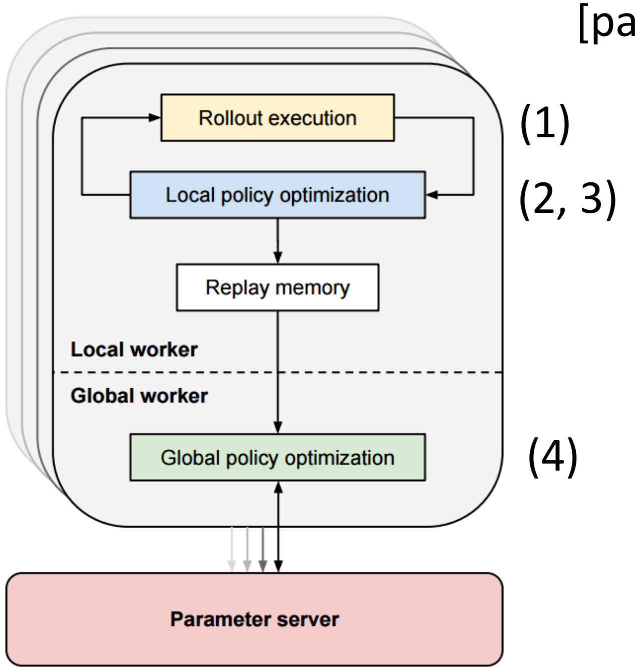
more on this later...

1,000,000 steps

Model-based algorithms: parallel GPS

1. get N samples τ_i by running $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ N times for each initial state $\mathbf{s}_0^j$ [parallelize sampling]
2. fit local models for each initial state [parallelize dynamics]
3. use LQR to get updated local policies $p_j(\mathbf{a}_t|\mathbf{s}_t)$ for each initial state $\mathbf{s}_0^j$ [parallelize LQR]
4. update policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ by imitating all $p_j(\mathbf{a}_t|\mathbf{s}_t)$ [parallelize SGD]

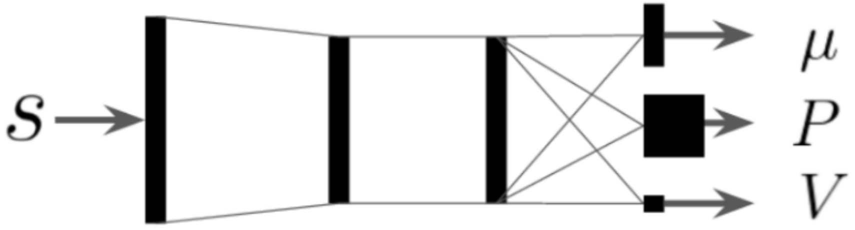




Yahya, Li, Kalakrishnan, Chebotar, L., '16

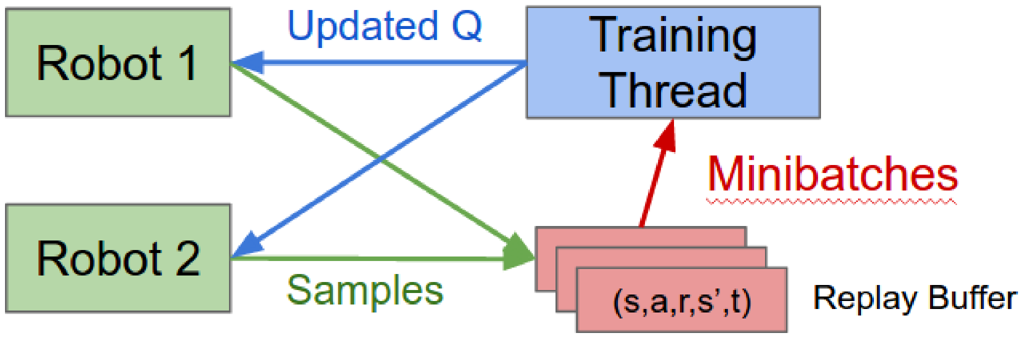
Model-based algorithms: parallel GPS

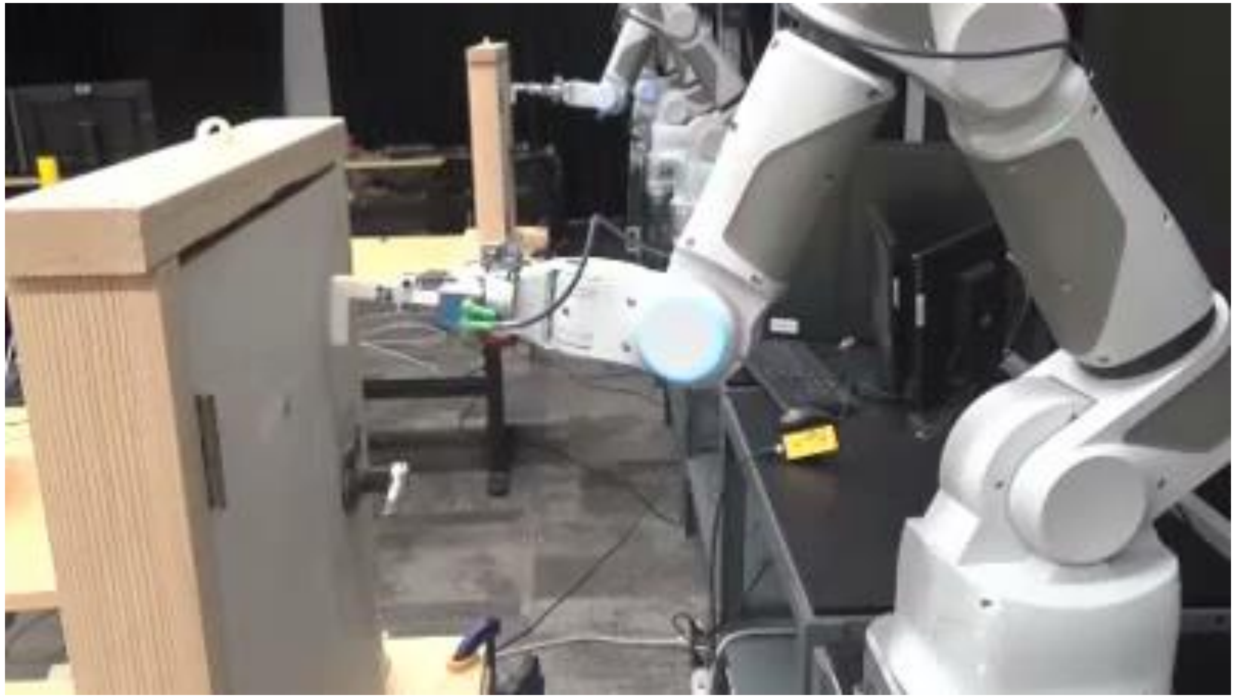
Real-world model-free deep RL: parallel NAF



NAF Architecture.

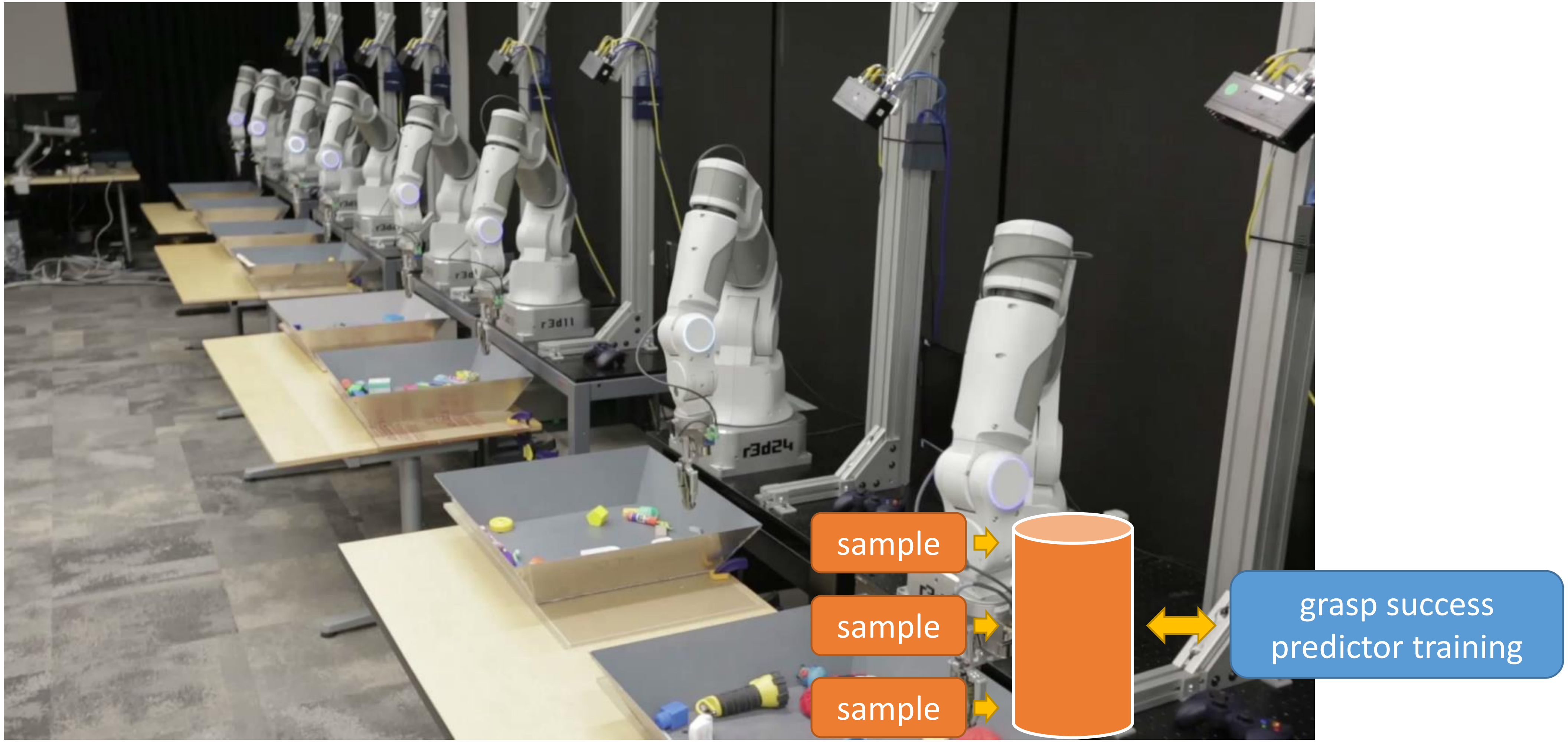
$$Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q) = A(\boldsymbol{x},\boldsymbol{u}|\theta^A) + V(\boldsymbol{x}|\theta^V)$$
$$A(\boldsymbol{x},\boldsymbol{u}|\theta^A) = -\frac{1}{2}(\boldsymbol{u}-\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))^T \boldsymbol{P}(\boldsymbol{x}|\theta^P)(\boldsymbol{u}-\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))$$





Gu*, Holly*, Lillicrap, L., '16

Simplest example: sample parallelism with off-policy algorithms



Break

Challenges in Deep Reinforcement Learning

Sergey Levine

UC Berkeley

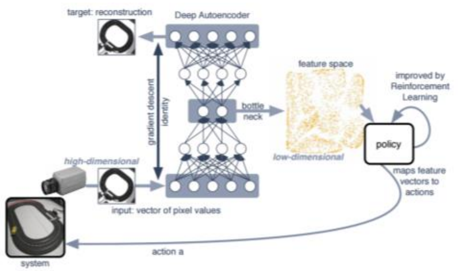
Today's Lecture

1. High-level summary of deep RL challenges
2. Stability
3. Sample complexity
4. Scaling up & generalization
5. Reward specification

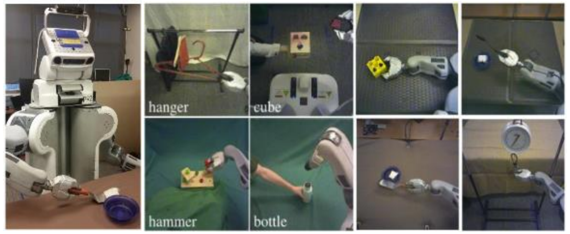
- Goals
  - Understand the open problems in deep RL
  - Understand tradeoffs between different algorithms

Some recent work on deep RL

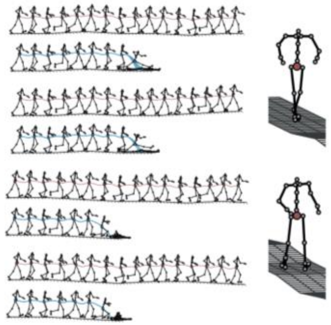
stability | efficiency | scale



RL on raw visual input
Lange et al.
2009

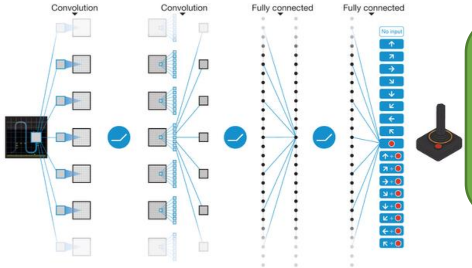


End-to-end visuomotor policies
Levine*, Finn* et al.
2015

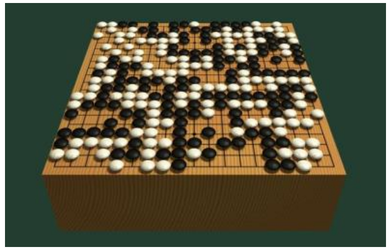


Guided policy search
Levine et al.
2013

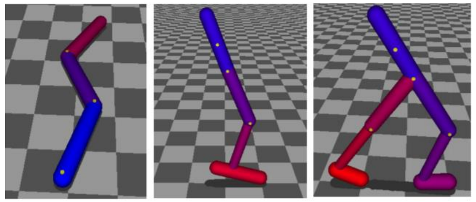
Deep deterministic policy gradients
Lillicrap et al.
2015



Deep Q-Networks
Mnih et al.
2013



AlphaGo
Silver et al.
2016

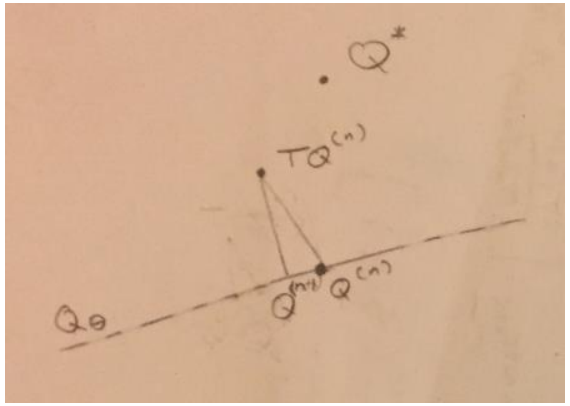


Trust region policy optimization
Schulman et al.
2015

Supersizing self-supervision
Pinto & Gupta
2016

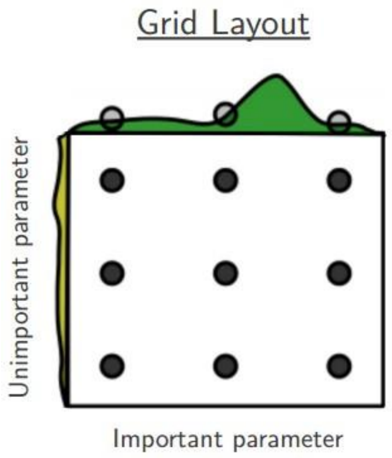
Stability and hyperparameter tuning

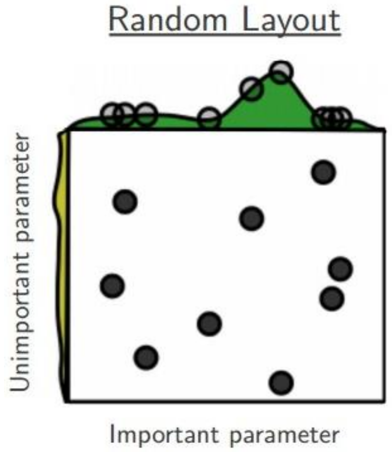
- Devising stable RL algorithms is very hard
- Q-learning/value function estimation
  - Fitted Q/fitted value methods with deep network function estimators are typically not contractions, hence no guarantee of convergence
  - Lots of parameters for stability: target network delay, replay buffer size, clipping, sensitivity to learning rates, etc.
- Policy gradient/likelihood ratio/REINFORCE
  - Very high variance gradient estimator
  - Lots of samples, complex baselines, etc.
  - Parameters: batch size, learning rate, design of baseline
- Model-based RL algorithms
  - Model class and fitting method
  - Optimizing policy w.r.t. model non-trivial due to backpropagation through time



Tuning hyperparameters

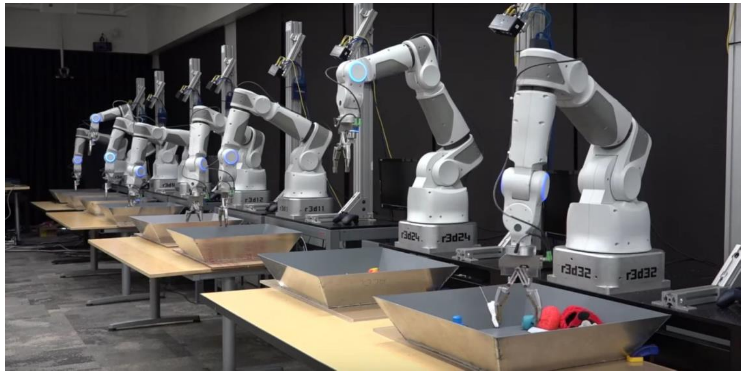
- Get used to running multiple hyperparameters
  - `learning_rate = [0.1, 0.5, 1.0, 5.0, 20.0]`
- Grid layout for hyperparameter sweeps OK when sweeping 1 or 2 parameters
- Random layout generally more optimal, the only viable option in higher dimensions
- Don’t forget the random seed!
  - RL is self-reinforcing, very likely to get local optima
  - Don’t assume it works well until you test a few random seeds
  - Remember that random seed is not a hyperparameter!





The challenge with hyperparameters

- Can't run hyperparameter sweeps in the real world
  - How representative is your simulator? Usually the answer is "not very"
- Actual sample complexity = time to run algorithm x number of runs to sweep
  - In effect stochastic search + gradient-based optimization
- Can we develop more stable algorithms that are less sensitive to hyperparameters?



What can we do?

- Algorithms with favorable improvement and convergence properties
  - Trust region policy optimization [Schulman et al. ‘16]
  - Safe reinforcement learning, High-confidence policy improvement [Thomas ‘15]
- Algorithms that adaptively adjust parameters
  - Q-Prop [Gu et al. ‘17]: adaptively adjust strength of control variate/baseline

- More research needed here!
- Not great for beating benchmarks, but absolutely essential to make RL a viable tool for real-world problems

Sample Complexity

gradient-free methods
(e.g. NES, CMA, etc.)

10x

fully online methods
(e.g. A3C)

10x

policy gradient methods
(e.g. TRPO)

10x

replay buffer value estimation methods
(Q-learning, DDPG, NAF, etc.)

10x

model-based deep RL
(e.g. guided policy search)

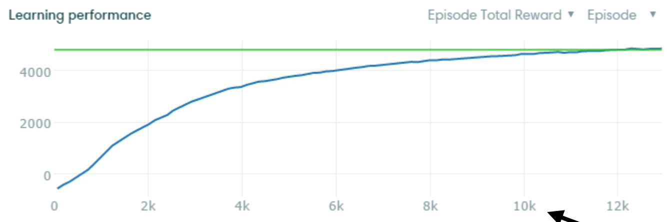
10x

model-based “shallow” RL
(e.g. PILCO)

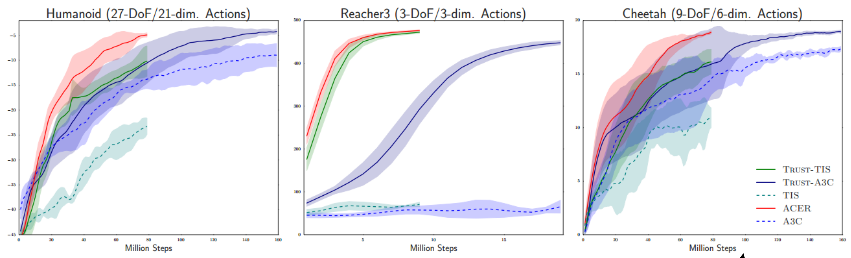
Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Tim Salimans[1] Jonathan Ho[1] Xi Chen[1] Ilya Sutskever[1]

half-cheetah (slightly different version)



TRPO+GAE (Schulman et al. ‘16)

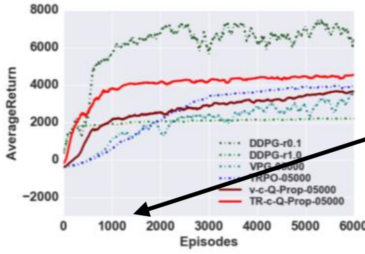


Wang et al. ‘17

100,000,000 steps
(100,000 episodes)
(~ 15 days real time)

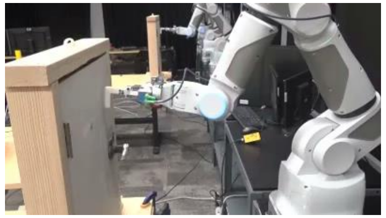
10,000,000 steps
(10,000 episodes)
(~ 1.5 days real time)

half-cheetah

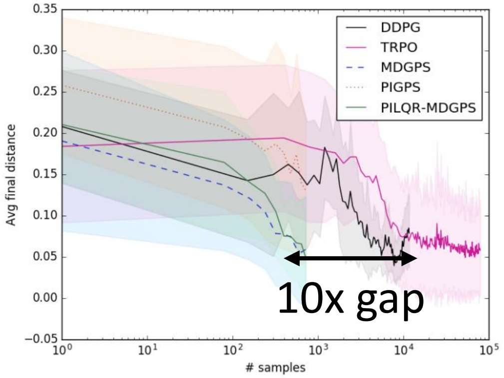


Gu et al. ‘16

1,000,000 steps
(1,000 episodes)
(~ 3 hours real time)



	cart-pole	cart-double-pole	unicycle
state space	$\mathbb{R}^4$	$\mathbb{R}^6$	$\mathbb{R}^{12}$
# trials	≤ 10	20–30	≈ 20
experience	$\approx 20\,\text{s}$	$\approx 60\,\text{s}$–$90\,\text{s}$	$\approx 20\,\text{s}$–$30\,\text{s}$
parameter space	$\mathbb{R}^{305}$	$\mathbb{R}^{1816}$	$\mathbb{R}^{28}$

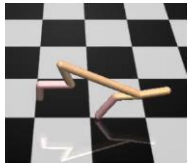


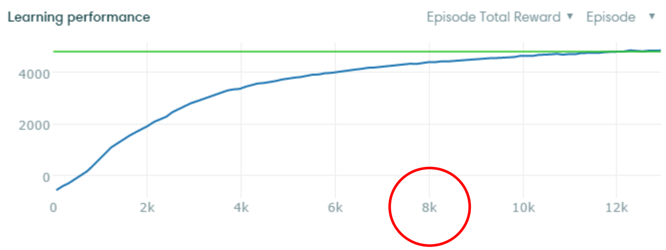
Chebotar et al. ’17 (note log scale)

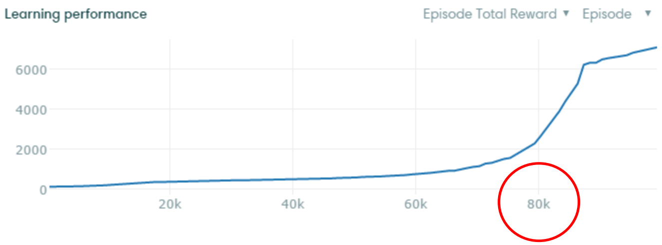
about 20 minutes of experience on a real robot

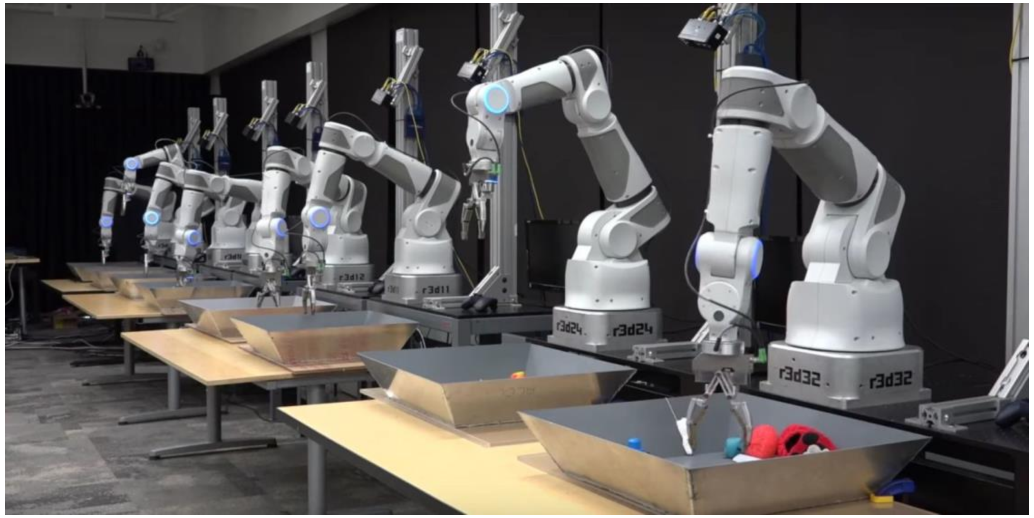
What about more realistic tasks?

- Big cost paid for dimensionality
- Big cost paid for using raw images
- Big cost in the presence of real-world diversity (many tasks, many situations, etc.)



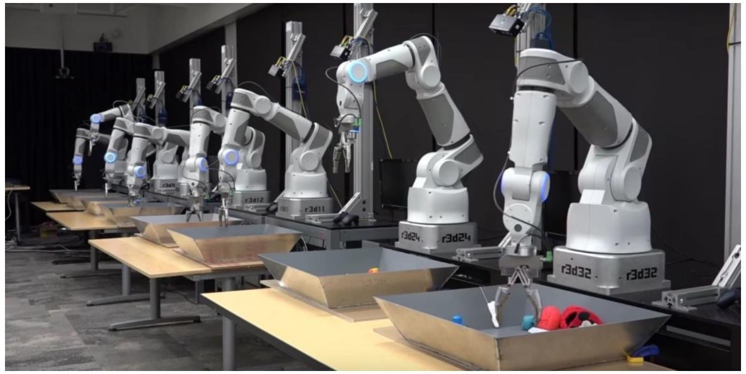






The challenge with sample complexity

- Need to wait for a long time for your homework to finish running
- Real-world learning becomes difficult or impractical
- Precludes the use of expensive, high-fidelity simulators
- Limits applicability to real-world problems

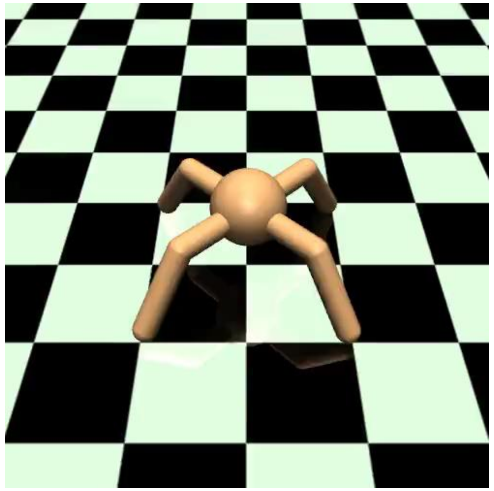


What can we do?

- Better model-based RL algorithms
- Design faster algorithms
  - Q-Prop (Gu et al. ‘17): policy gradient algorithm that is as fast as value estimation
  - Learning to play in a day (He et al. ‘17): Q-learning algorithm that is much faster on Atari than DQN
- Reuse prior knowledge to accelerate reinforcement learning
  - RL2: Fast reinforcement learning via slow reinforcement learning (Duan et al. ‘17)
  - Learning to reinforcement learning (Wang et al. ‘17)
  - Model-agnostic meta-learning (Finn et al. ‘17)

Scaling up deep RL & generalization

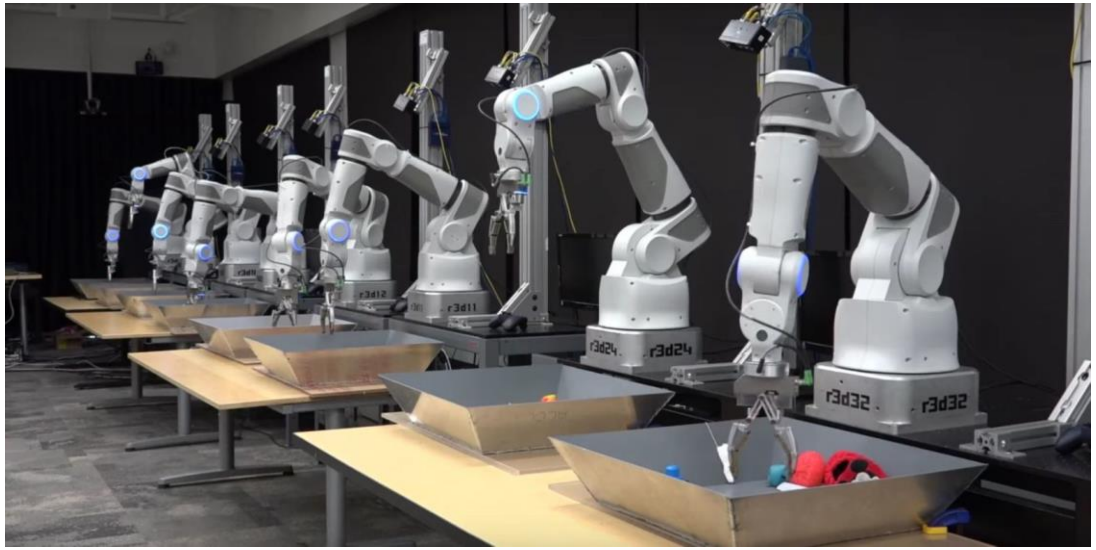
- Large-scale
- Emphasizes diversity
- Evaluated on generalization



- Small-scale
- Emphasizes mastery
- Evaluated on performance
- Where is the generalization?

Generalizing from massive experience

Pinto & Gupta, 2015



Levine et al. 2016

Generalizing from multi-task learning

- Train on multiple tasks, then try to generalize or finetune
  - Policy distillation (Rusu et al. ‘15)
  - Actor-mimic (Parisotto et al. ‘15)
  - Model-agnostic meta-learning (Finn et al. ‘17)
  - many others…
- Unsupervised or weakly supervised learning of diverse behaviors
  - Stochastic neural networks (Florensa et al. ‘17)
  - Reinforcement learning with deep energy-based policies (Haarnoja et al. ‘17)
  - many others…

Generalizing from prior knowledge & experience

- Can we get better generalization by leveraging off-policy data?
- Model-based methods: perhaps a good avenue, since the model (e.g. physics) is more task-agnostic
- What does it mean to have a "feature" of decision making, in the same sense that we have "features" in computer vision?
  - Options framework (mini behaviors)
    - Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning (Sutton et al. '99)
    - The option-critic architecture (Bacon et al. '16)
  - Muscle synergies & low-dimensional spaces
    - Unsupervised learning of sensorimotor primitives (Todorov & Gahramani '03)

Reward specification

- If you want to learn from many different tasks, you need to get those tasks somewhere!
- Learn objectives/rewards from demonstration (inverse reinforcement learning)
- Generative objectives automatically?

reward

Mnih et al. '15

reinforcement learning agent

what is the reward?