Coping with sample inefficiency of deep-reinforcement learning (DRL) for embodied AI

WiML Un-Workshop @ ICML 2020

July 13, 2020

Session Leaders: Vidhi Jain, Simin Liu
Session Facilitators: Ganesh Iyer

Learning has demonstrated great potential for applications like robotics, self-driving cars and IoT [1]. Many of its notable successes have been in virtual gameplay, where thousands of hours of training is feasible. However, collecting real-world data is laborious. To this end, we discuss ways to make DRL methods more practical for embodied AI. Deep reinforcement learning has had great success in simulation, for example, AlphaGo [2] beat human experts, Deepmind's AlphaStar [3] beat top professional players at StarCraft, a challenging real-time strategy game, in 2019.. Similarly, OpenAI Five's DOTA bot [4] won the championship. But it has been much harder applying it to real world platforms (like robots, autonomous vehicles, process control).

Why learning algorithms haven't been deployed on consumer robotics/AV platforms? For example, Roombas use very simple algorithms like spiral cleaning in conjunction with VSLAM (simultaneous localization and mapping), a very classic robotics method. Roombas rely on a few simple algorithms, such as spiral cleaning (spiraling), room crossing, wall-following and random walk, angle-changing after bumping into an object or wall.

There are many reasons why robotic learning is challenging in real world - lack of safety assurances, reward specification, lack of progress on the continual learning front. One of the major focus is mainly on lack of sample efficiency. One may consider it is not an issue if we have good simulators as we can get loads of training data. But it seems a big issue is for real platforms where acquiring data is laborious and time consuming.

We should look into real embodied AI learning in order to fundamentally enhance the notion of intelligence by incorporating multi-modal interaction. More importantly, we want to be efficient and reduce monotonous burden with AI tools like autonomous vehicle and home assistants.

In our breakout session, we discussed about some ways (either proven or promising) to make DRL feasible for embodied AI. We talked about two divergent approaches: algorithmic approaches to improve sample efficiency and alternatively, circumventing the sample efficiency problem by scaling up data collection for current state-of-the-art algorithms.

1 Choice of paradigm

1.1 Model-based (MB) or model-free (MF)?

Model-based methods learn an explicit model of transition function (i.e. learn the prob of moving to a state if you take an action at your current state). Model-free methods don't learn an explicit model of transition function. Instead, you might learn a value function (i.e. V(s), Q(s, a), A(s, a)) or directly learn a policy. Here the value function refers to the value of a state is the expected cumulative reward one earns by starting at that state.

Model-based is more sample efficient. However, the big caveat is that its asymptotic performance levels out early (unlike model-free).

One reason for this comparative efficiency is that it seems that learning local dynamics models (i.e. models that hold in a subset of the input space) is inherently easy. Transition dynamics (especially for robots) are usually locally linear, which makes them easy to learn.

While model-based learning dominates in the low-data regime, its performance plateaus far below model-free methods in the high-data regime. In other words, there's a worrying phenomenon where adding more data does not help [5]. This can happen because of (1) Dynamics bottleneck, and (2) Planning horizon dilemma. Dynamics bottleneck refers to the infeasibility of learning a globally accurate dynamics model. MBRL algorithms seem to only be learning locally-accurate dynamics models (i.e. accurate over a small subset of the input space). One piece of evidence for the idea that learning accurate dynamics is the bottleneck is that in some cases, if you use ground-truth dynamics with the MB controller, you can the learned-dynamics counterpart.

Planning horizon dilemma refers to the planning done by MB controllers by applying/chaining the model for multiple timesteps to see what happens when you take a certain action sequence and finding a good action sequence. However, task performance tends to degrade after 20-40 timesteps due to accumulated model error. So we can only plan short-term, which is obviously a draw-back for tasks that require longer foresight.

Some real robots that use MB approach include Millirobot path following, which combines MB with random shooting MPC controller [6] and Wayve's AV, which uses a dynamics model where the state is the image input compressed to a latent vector by VAE, as "world" models in [7]. It backpropagation through time (BPTT) to apply the dynamics model for control.

1.2 Off-policy or on-policy?

Though MB excels in the low-data regime, MF methods are more popular and have been more extensively developed. So, it is also important to make MF work for real platforms. There are two choices among MF algorithms - off-policy and on-policy. Off-policy algorithm can use training samples generated by any policy. This includes Q-learning algorithms. On-policy algorithms can only use training samples generated by the current iteration of their learned policy. Includes policy gradient algorithms (everything outlined in red) Off-policy is much more data efficient because you can reuse data from different sources, like previous iterations of the policy that we are currently learning. Off-policy updates have been shown to learn policies for robotics manipulation [8] in a sample efficient way.

2 Using knowledge from other domains

Transfer learning is a kind of "Forward transfer". Transfer learning refers to when a model trained on one task data can be applied or transferred to a new task. from mid level visual feature representations has shown effective success in embodied AI navigation tasks. Similarly, transfer approaches have been applied in language based navigation and modular approach for vision-language grounded navigation tasks.

Multi-task learning implies that the policy is trained to solve many tasks, and show good generalization to an unseen task. This approach is inspired from how human babies learn not a single but many different tasks, and has success in reinforcement learning, particularly evolutionary policy search. DeepMind's UNREAL agent learns to navigate by training on auxillary tasks in an unsupervised manner.

Meta learning refers to learning to learn by solving a variety of tasks. The meta policy learns to minimize the distribution of loss functions that allows it to generalize well to few-shot learning settings. Meta learning has been shown to enable adaptability in legged robots in case of leg joint failures.

Combining modular components of learning systems for embodied AI has been successful approach in Active Neural SLAM and object navigation challenges recently where the policy is decomposed into a mapper, global and local policy. These components have been shown to work on real LocoBot using MaskRCNN for object detection and segmentation.

3 Human Demonstrations and feedback

Copying experts reduce the number of samples needed to explore the state-action space for the optimal policy. Vanilla imitation learning involves extraction of state-action pairs from collected expert demos (trajectories), and minimization of error between policies and next actions. [9] Incorporating demonstrations can also be investigated with learning the Q-value and through inverse reinforcement learning.

4 Scaling up data collection

4.1 Sim-2-real

Can we train primarily in simulation and then transfer the policy to the real world? Sim-2-real approach allows for cheap data collection, effortless scaling and safe setting to learn policies. But we need to deal with the 'sim-2-real gap'. There are often significant visual and physical differences between the simulation and reality.

The simplest transfer case where we can naively go from simulations to real-world, requires that we have a simulator with the perfect model of the world. Domain adaptation and domain randomization have shown empirical success with neural networks. Domain adaptation refers to approach of finding a robust mapping from simulation to reality. Domain randomization refers to perturbing the simulation to different extents in the hope that generalization to reality becomes an interpolation task (rather than a difficult extrapolation one).

4.2 Parallelized methods

4.2.1 Parallelized asynchronous data collection

Edge workers merely send raw data to the server for training. Current attempts to parallelize data collection (Arm Farm at Google), algorithmic changes need to be made to suit in asynchronous, parallel data collection (i.e. straggler mitigation) For embodied AI navigation, [10] scaled the data collection and execution in decentralized way, while maintaining the centralized synchronized training updates.

4.2.2 Federated learning

Edge workers update personal models and asynchronously send model parameters to update the global features on the server. Federated learning allows the robot to learn local features for adaptation, leverage the personal data in privacy preserving way, and communicate the model parameters efficiently for global feature representation. Federated learning approach has been successful in improving on-device voice assistants for keyword spotting [11], and is an open research area for self-driving cars and IoT [12]. Federated deep reinforcement learning is an active area of research [13].

5 Miscellaneous

Embedding strong inductive priors. For example, reward shaping [14], crafting exploratory behavior [15], and incorporating human feedback or demonstrations [11]. Additional topics include algorithmic changes to make more efficient use of data, like experience replay [16] or curriculum learning.

6 Discussion

- When is DRL useful/necessary for embodied AI applications? (i.e. when do data-driven methods have an advantage over traditional planning control methods?)
- Is sample inefficiency a bottleneck in the progress of DRL for robotics? Pro: Sample inefficiency deters real-world applications, adaptation of the algorithms Cons: Sample inefficiency in current methods can

be overcomed with improved faster hardware and realistic simulation. Though sample inefficient, the current algorithms are somewhat self-reliant - perhaps we have a tradeoff here.

- Why is the sample inefficiency of current DRL algorithms not so serious? - improved hardware, simulation power, less reliance on known priors and more on rules extracted from patterns in the data i.e. data-driven?
- Does Sim2Real work? Just by publishing papers on the success of sim2real, we can not believe that sim2real works. There are no negative examples that we can discuss! Since sim-2-real somewhat works, can't we just use any of our DRL algorithms, even if data inefficient?
- Should our focus as a community be on circumventing sample efficiency (gathering data at scale, sim-2-real) or address it at algorithmic level? Should we modify our learning algorithm or not? (focus: embodied AI learning models)
- Do you see ways in which these methods can be combined? What's wrong with the way we currently measure/quantify sample efficiency?

7 Open Questions

- Should we expend more effort in improving the realism of our simulations or on algorithms to compensate for their lack of realism? If the first: how do we measure simulation realism? If the second: how do we make algorithms less ad hoc (arbitrary) than they are currently (esp. Domain Randomization - isn't the way you permute is arbitrary)?
- Federated learning has shown early promise in areas like query suggestions on mobile phones, smart speakers, etc. What other applications can you think of?
- What kind of realism is desirable - like visual realism through meshes or physical realism through CAD models and physics based game engines? For example, game engines like Unity (AI2THOR) can provide similar physical realism. Real world 3D mesh renderers like AI Habitat provide with visual realism in simulators.

References

[1] Lei Lei, Yue Tan, Shiwen Liu, Kan Zheng, and Xuemin Shen. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *CoRR*, abs/1907.09059, 2019.

[2] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[3] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[4] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019.

[5] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019.

[6] Anusha Nagabandi, Guangzhao Yang, Thomas Asmar, Gregory Kahn, Sergey Levine, and Ronald S. Fearing. Neural network dynamics models for control of under-actuated legged millirobots. *CoRR*, abs/1711.05253, 2017.

[7] David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

[8] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[9] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016.

[10] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames, 2019.

[11] Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjan Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-iid data with federated learning, 2020.

[12] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M. Hadi Amini. Federated learning for resource-constrained iot devices: Panoramas and state-of-the-art, 2020.

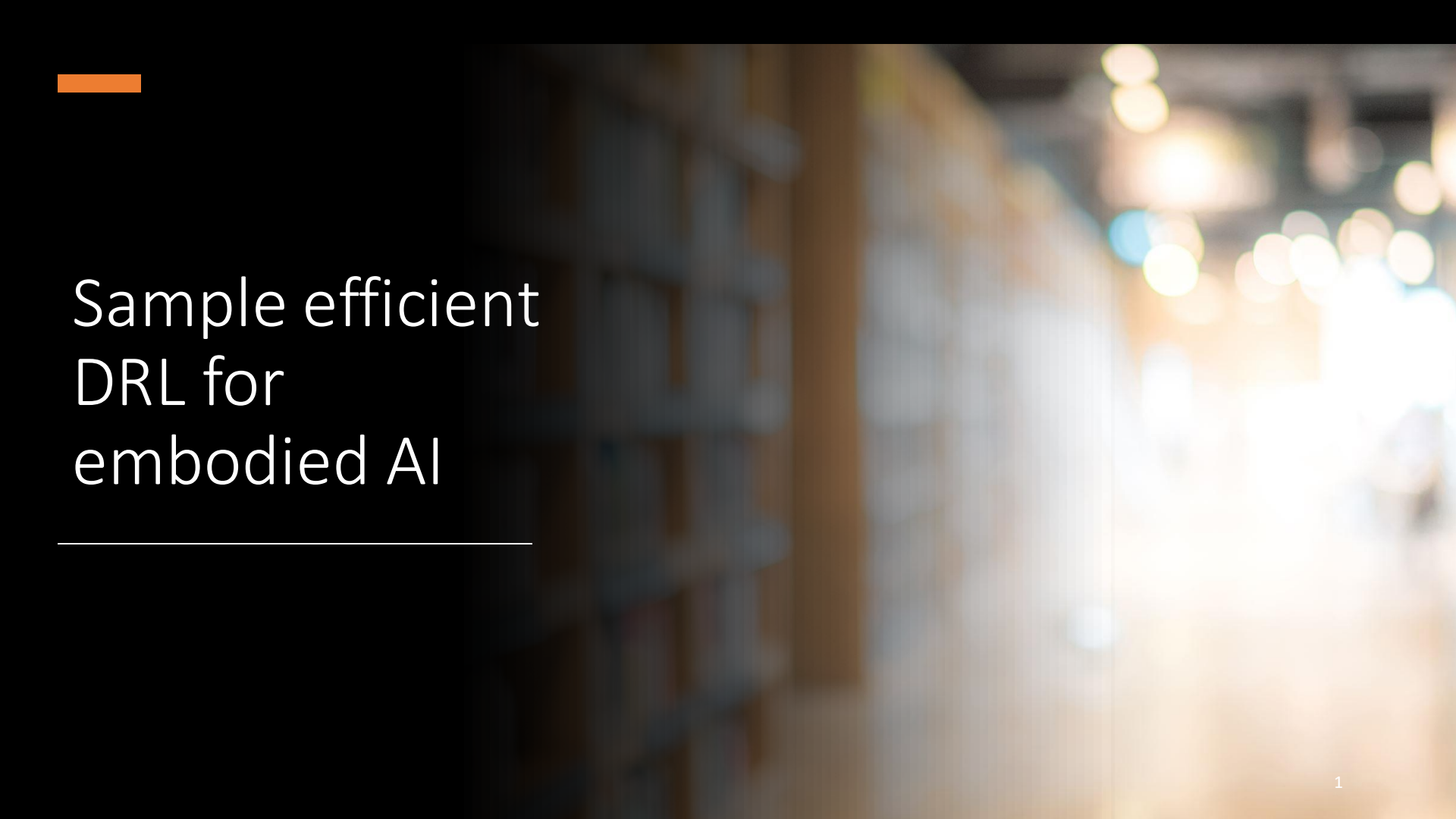
[13] Hankz Hankui Zhuo, Wenfeng Feng, Yufeng Lin, Qian Xu, and Qiang Yang. Federated deep reinforcement learning, 2019.

[14] Glen Berseth, Daniel Geng, Coline Devin, Nicholas Rhinehart, Chelsea Finn, Dinesh Jayaraman, and Sergey Levine. Smirl: Surprise minimizing reinforcement learning in dynamic environments, 2019.

[15] Deepak Pathak, Pulkit Agrawal, Alyosha Alexei Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 488–489, 2017.

[16] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5048–5058. Curran Associates, Inc., 2017.

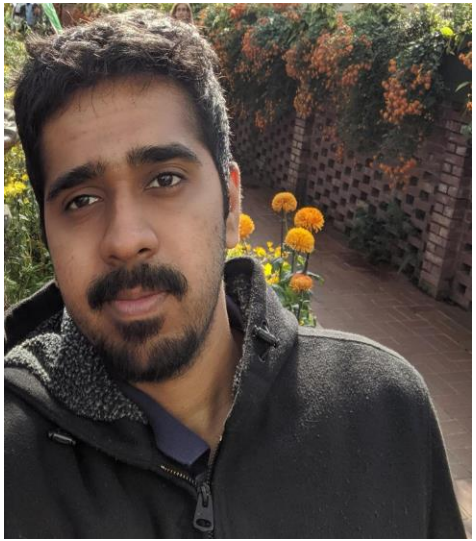
Sample efficient DRL for embodied AI



Leaders and Facilitator

Vidhi Jain
MS student at CMU
Co-leader

Simin Liu
PhD student at CMU
Co-leader



Ganesh Iyer
Applied Scientist at Amazon Lab126
Facilitator

2-minute breakout room introductions

- Name
- Position
- What do you want from this session?

Session format

Presentation: ~30 min

Intro + 4 topics

- Post questions to chat!
- 1-2 clarifying questions after each topic

Discussion: ~30 min

2 sets of questions

- Discuss in breakout rooms, reconvene to share

Why Embodied AI?

Enhancing Intelligence

Learning in real environments through explorative physical interaction

Why Embodied AI?

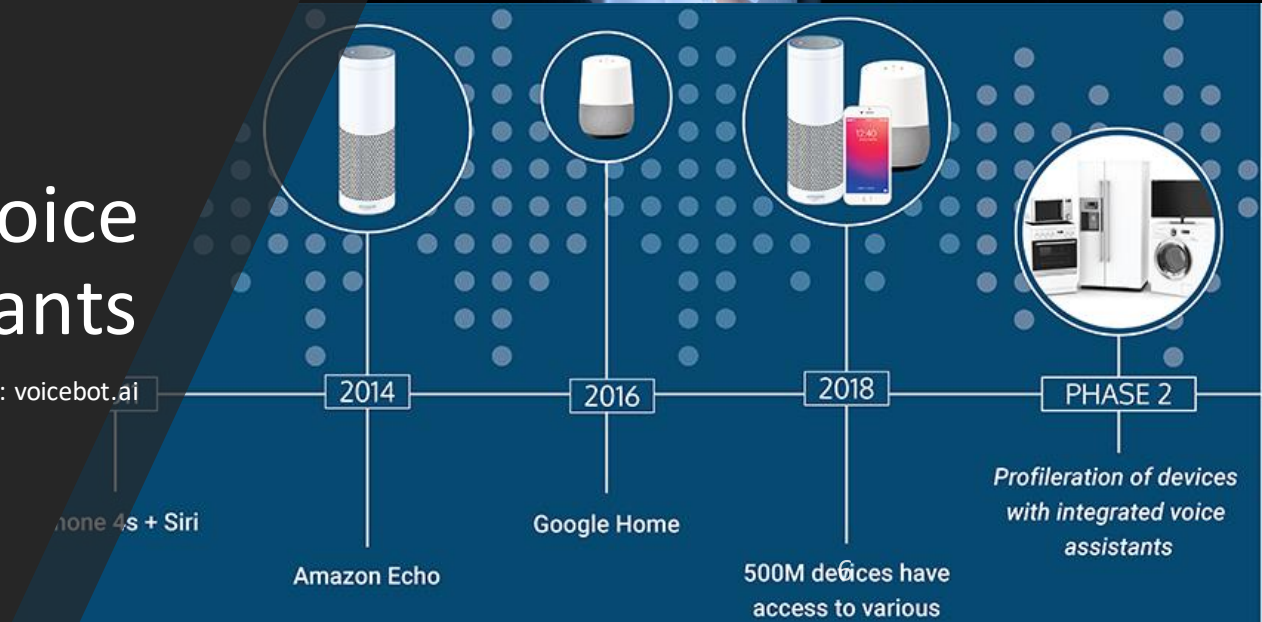
AI tools

Self-driving cars

Source: roboticsbusinessreview

Voice assistants

Source: voicebot.ai



DRL has had great success in simulation!

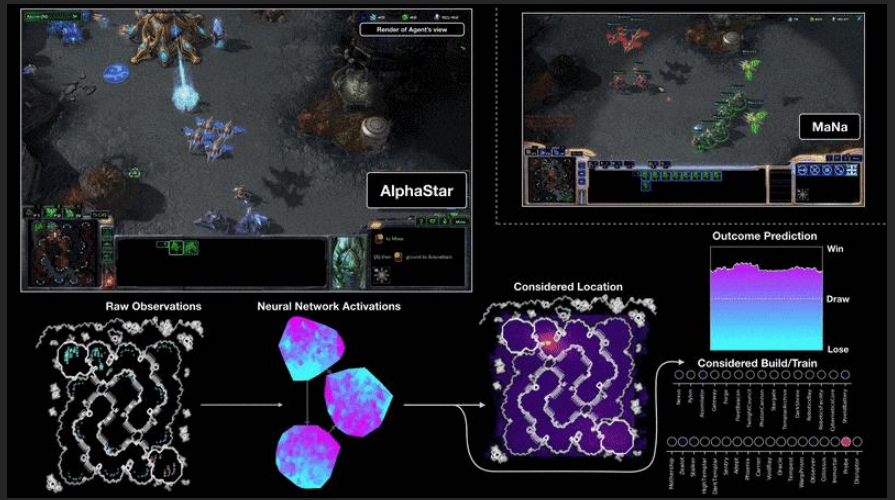


Image credits: Business Insider, DeepMind

But it’s been much harder applying it to real world platforms…

iRobot Roomba

Waymo AV

Boston Dynamics Atlas

What's hindering us?

One main reason is **data efficiency**:

Not a big issue in simulation

Big issue for real platforms!

Two perspectives for solutions:

1. Improvise algorithmically
2. Scale up data collection

Session goals

Share opinions on the comparative merit of each method/perspective

Identify the “gaps” in the current research

Outline

- Careful choice of paradigm
- Using knowledge from other domains
- Human demonstrations and feedback
- Scaling data collection

Model based or model free?

MB: learn an explicit model of the transition function $p(s_{t+1}|s_t, a_t)$

MF: learn value function (i.e. $V(s),\ Q(s,a),\ A(s,a)$) or directly learn a policy

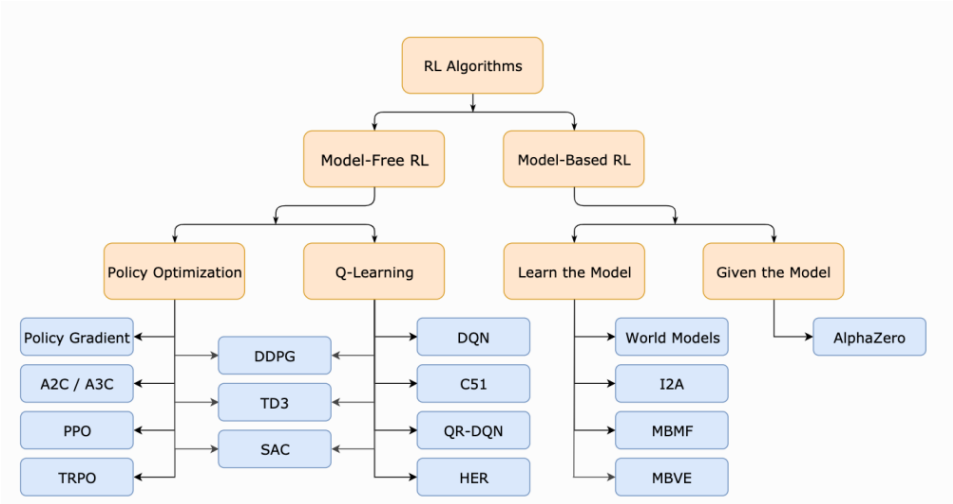


Image from https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Model based or model free?

MB is more sample efficient…but there's a caveat: poor asymptotic performance.

Examples of MBRL in the real world

Self-driving

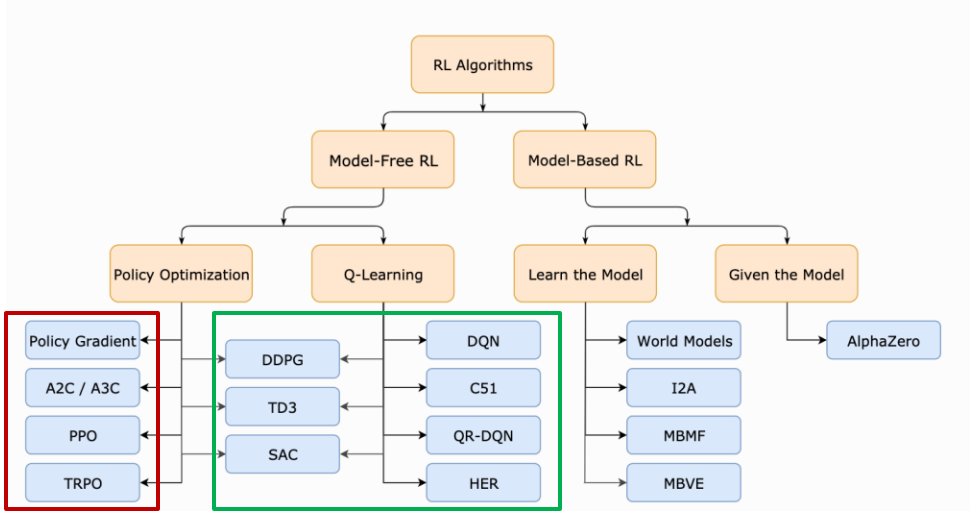
Millirobot path following

Off-policy or on-policy? (MF)

Off-policy: can use samples generated by any policy.
I.e. Q-learning

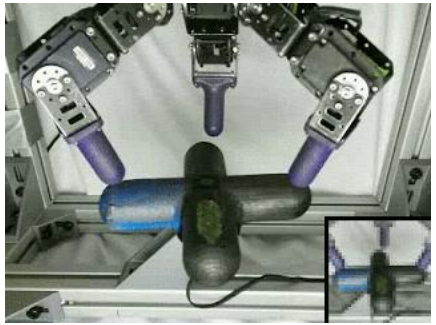
On-policy: can only use samples generated by current policy.
I.e. policy gradient

> Off-policy is more sample efficient

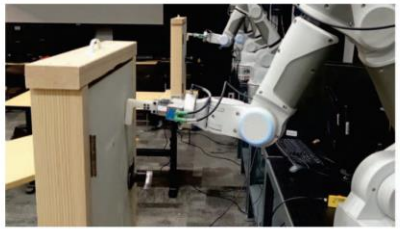


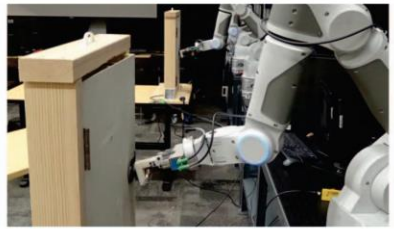
Examples of off-policy algs in the real world

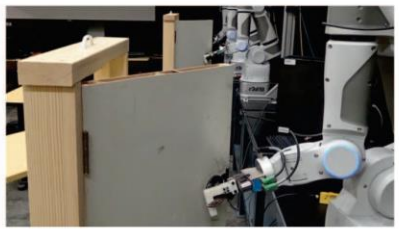
Soft Actor-Critic



Asynchronous Q-learning





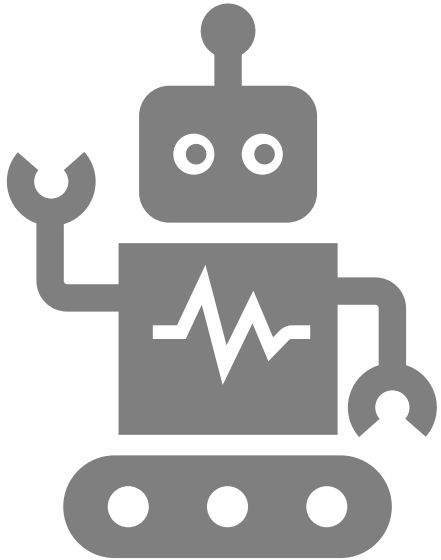


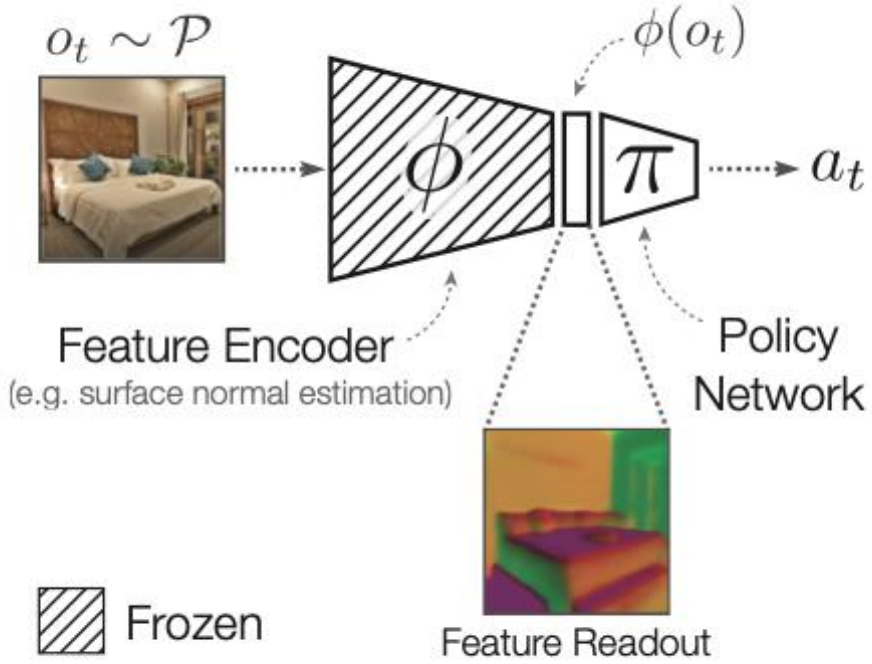
Outline

- Careful choice of paradigm
- Using knowledge from other domains
- Human demonstrations and feedback
- Scaling data collection

Using knowledge from other domains

- Transfer learning
- Multi-task learning
- Meta-learning
- Modular components



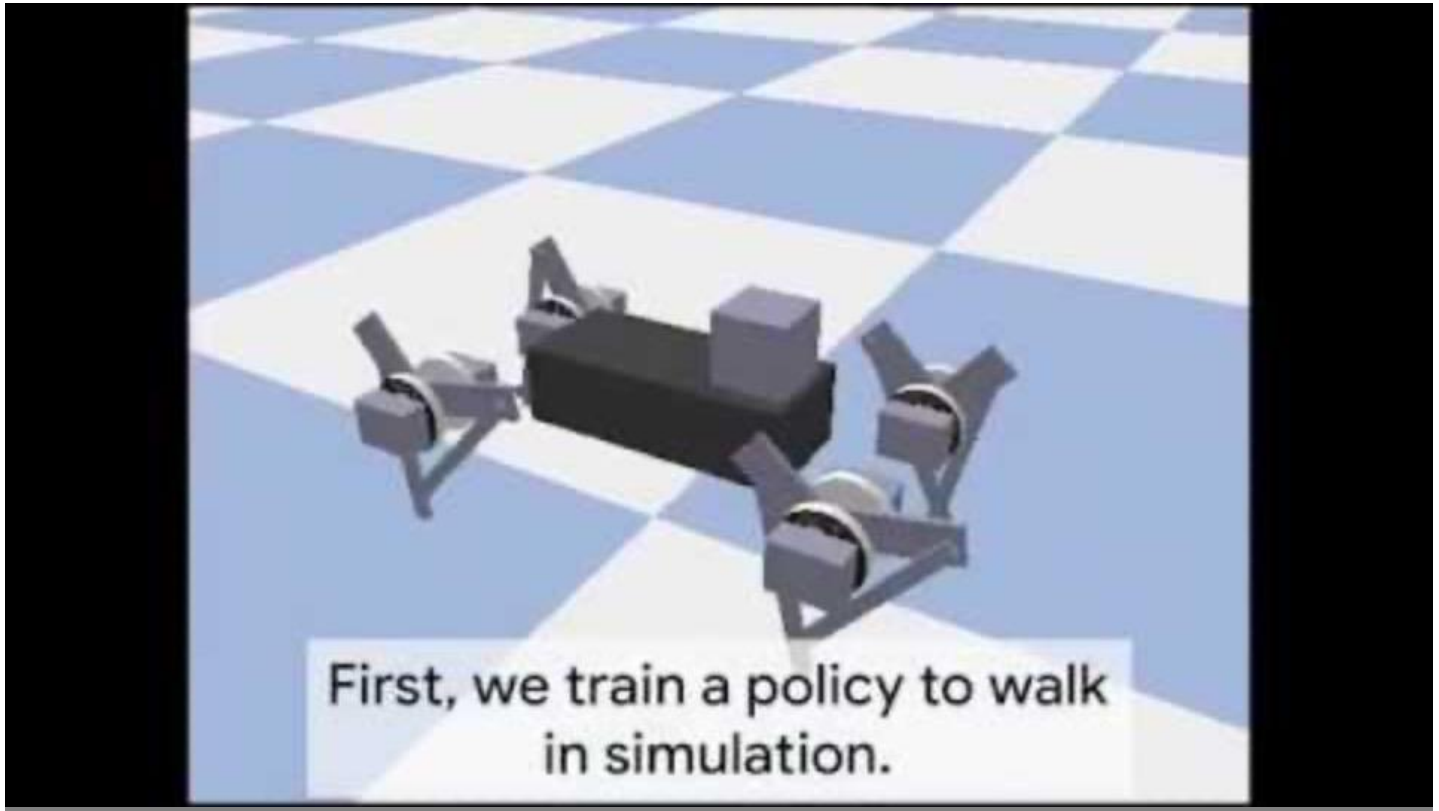


Mid level feature representations

Transfer learning

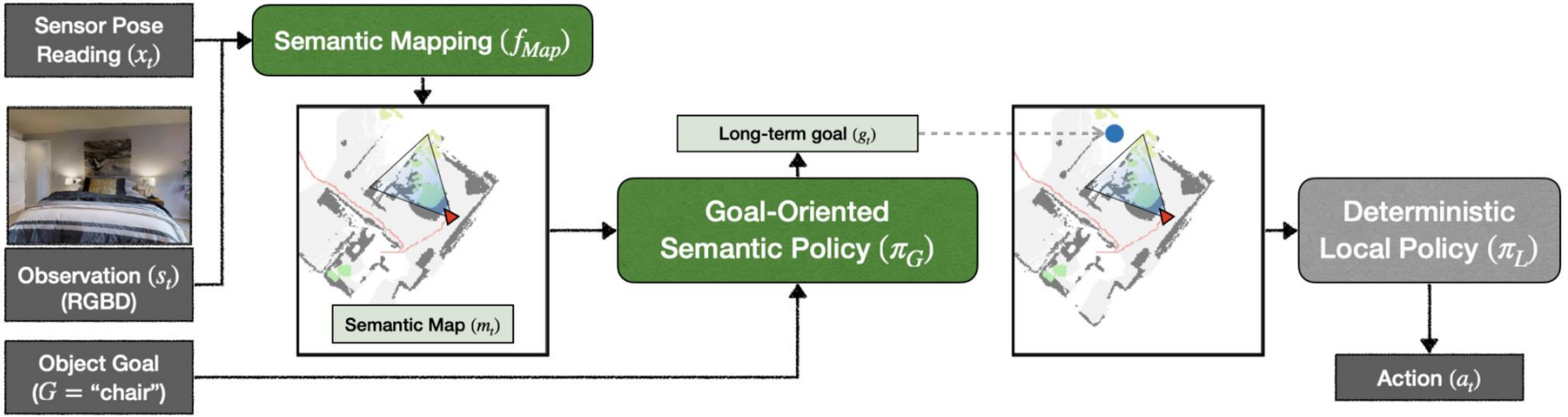
UNREAL: Unsupervised Auxiliary Task for RL Agent

Multi-task learning



Evolutionary meta learning for adaptability in Legged Robots

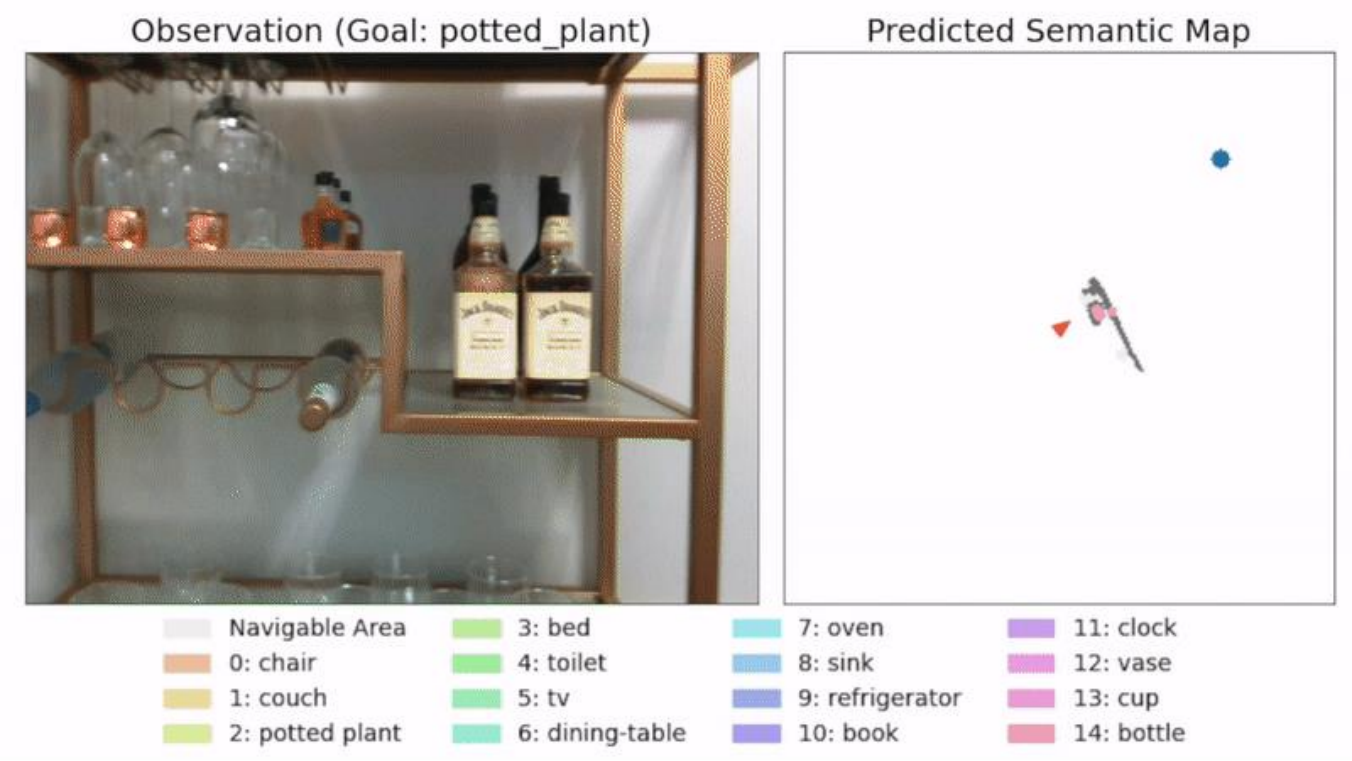
Meta Learning



Object Goal Navigation using Goal-Oriented Semantic Exploration

Modular components for object navigation

Real Transfer: Goal-Oriented Semantic Exploration



Outline

- Careful choice of paradigm
- Using knowledge from other domains
- Human demonstrations and feedback
- Scaling data collection

Expert demonstrations and human feedback

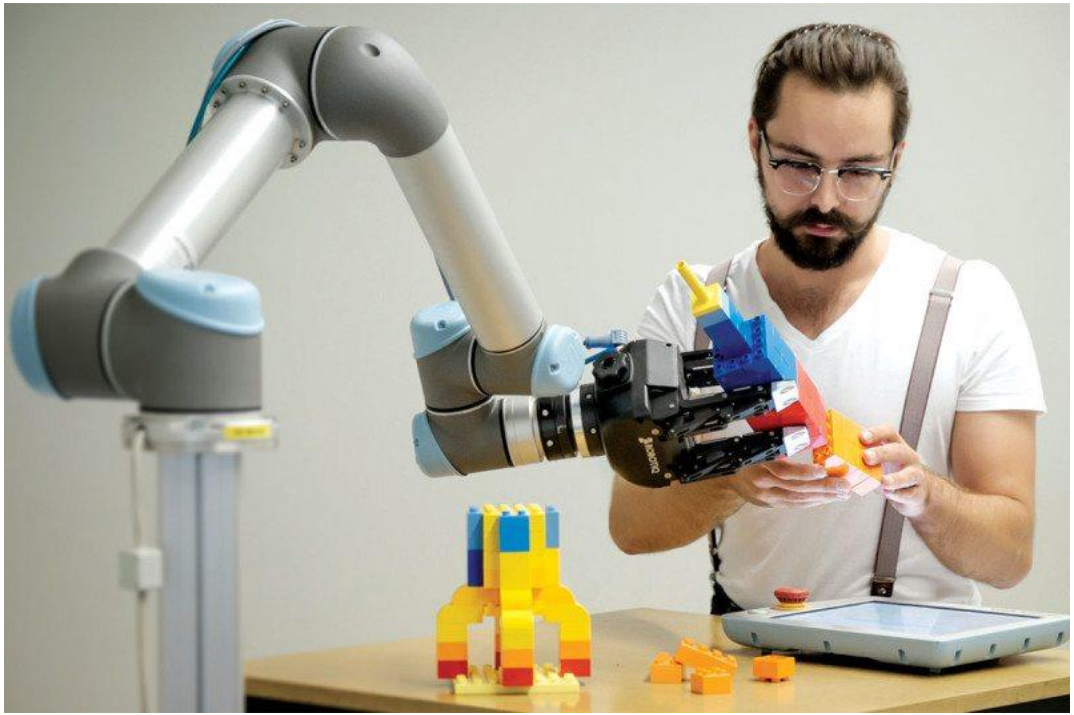


Image credit: http://sainslaboratmarit.blogspot.com/2016/10/robot-learns-to-play-with-lego-by.html

Imitation learning: copying experts

Algorithm:

1. Collect expert demonstrations (trajectories τ^*)
2. Treat demos as i.i.d. state-action pairs and split into dataset: $(s_0^*, a_0^*), (s_1^*, a_1^*), \ldots$
3. Learn policy via supervised learning: minimize $L\big(a^*, \pi_\theta(s)\big)$

Vanilla imitation learning

NVIDIA AV

Combining IL + meta-learning

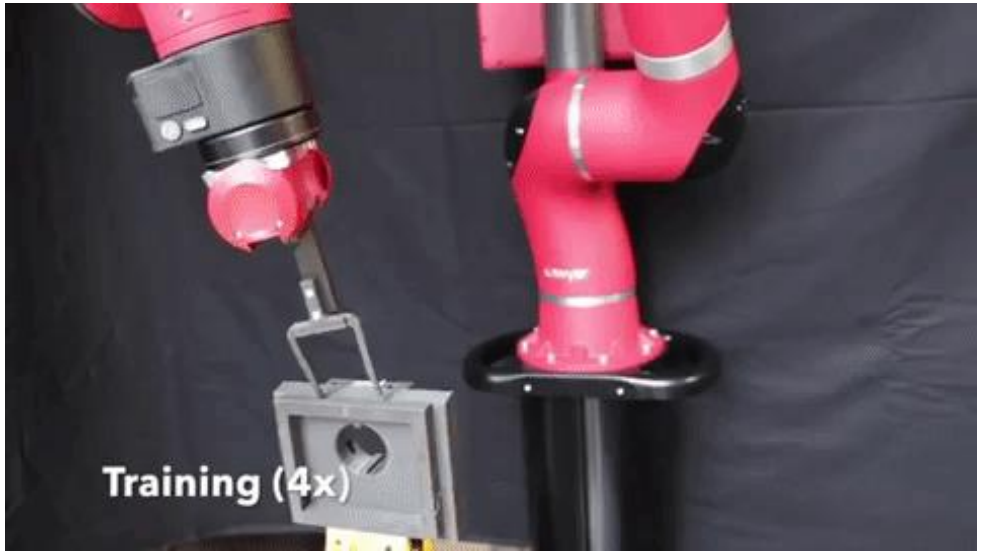
One-shot visual imitation learning

Using demos in an RL fashion

1. Split expert trajectories into (s_t, a_t, r_t, s_{t+1}) tuples
2. Insert into off-policy algorithm's data buffer

Demos for deep Q-learning

Clip insertion task

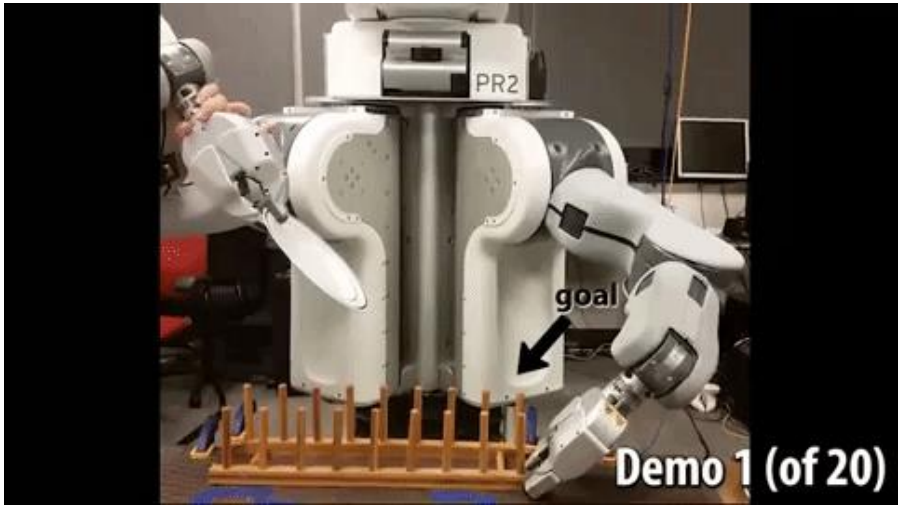


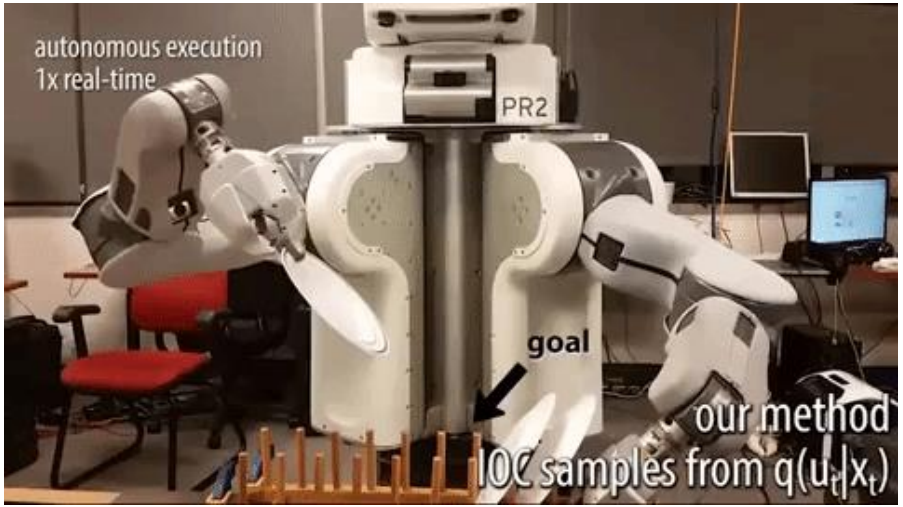
Inverse reinforcement learning

Given (s_t, a_t, s_{t+1}) from expert, assume expert optimality and find $r(s_t, a_t)$

Sample-based maxent IRL

Guided cost learning

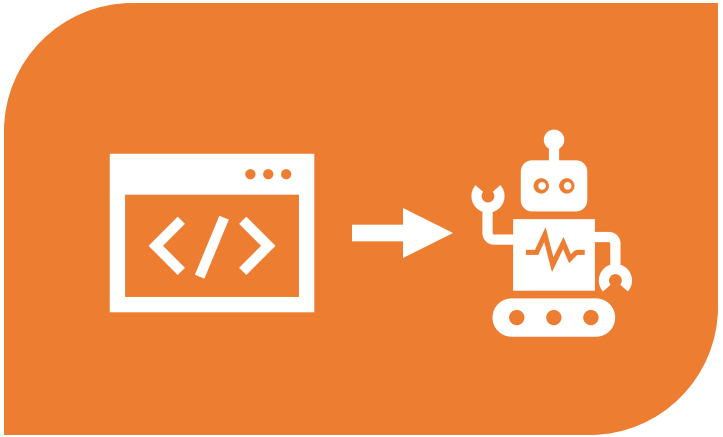




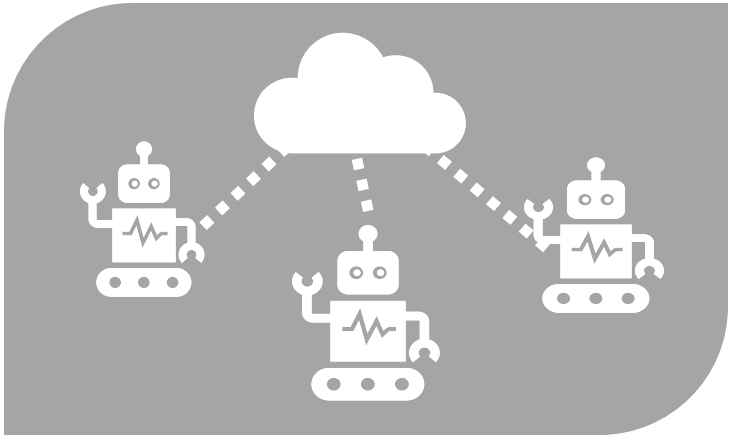
Outline

- Careful choice of paradigm
- Using knowledge from other domains
- Human demonstrations and feedback
- Scaling data collection

Gather data at scale



SIM-2-REAL TRANSFER



PARALLELIZED METHODS

Outline

- Careful choice of paradigm
- Using knowledge from other domains
- Human demonstrations and feedback
- Scaling data collection
  - Sim2real
  - Parallelized methods

Sim2Real

What is sim2real?

Train in simulation, transfer policy to real world

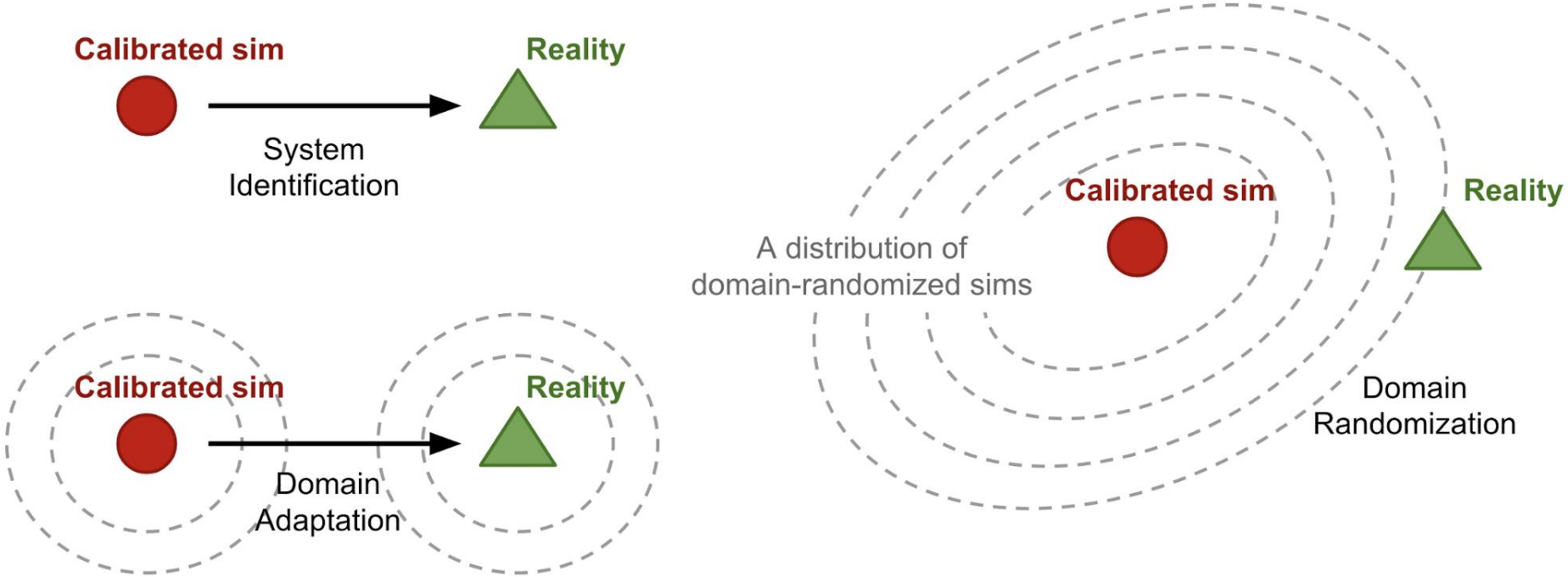
Benefits for training sim2real

- cheap data
- safe to learn and explore
- effortless to scale

Sim2Real gap

visual and physical differences between simulation and reality

Ways of sim-2-real transfer



Source: https://lilianweng.github.io/lil-log/2019/05/05/domain-randomization.html

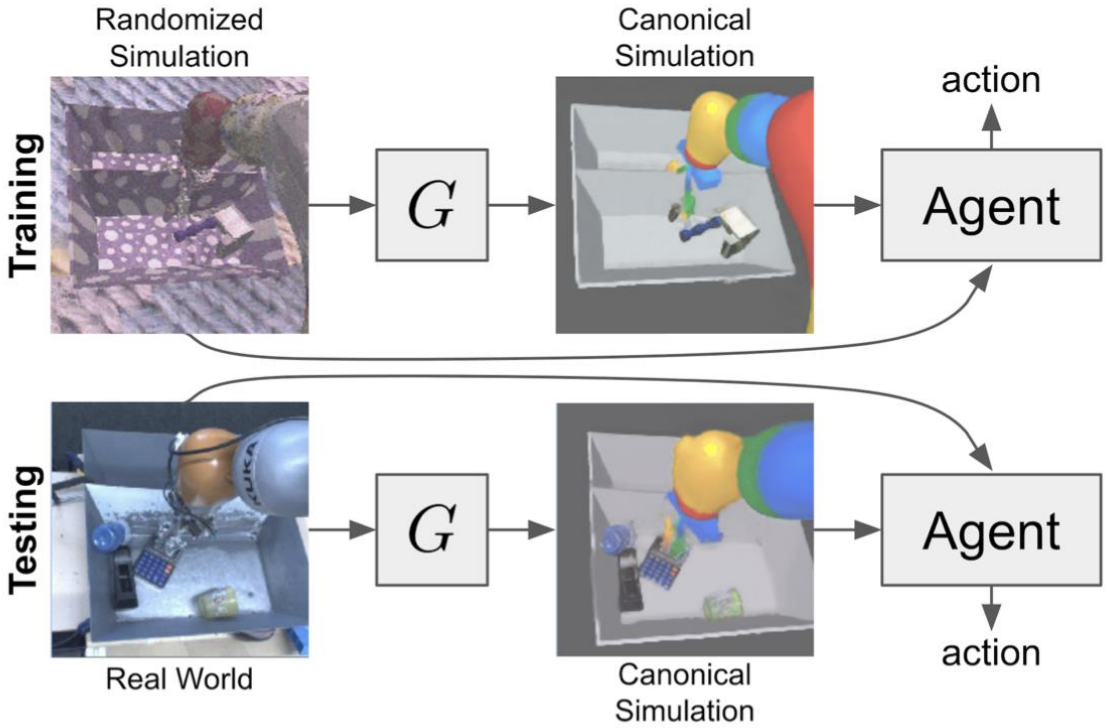
Simulator realism: What kind of realism is desirable?

VISUAL REALISM: MESHES

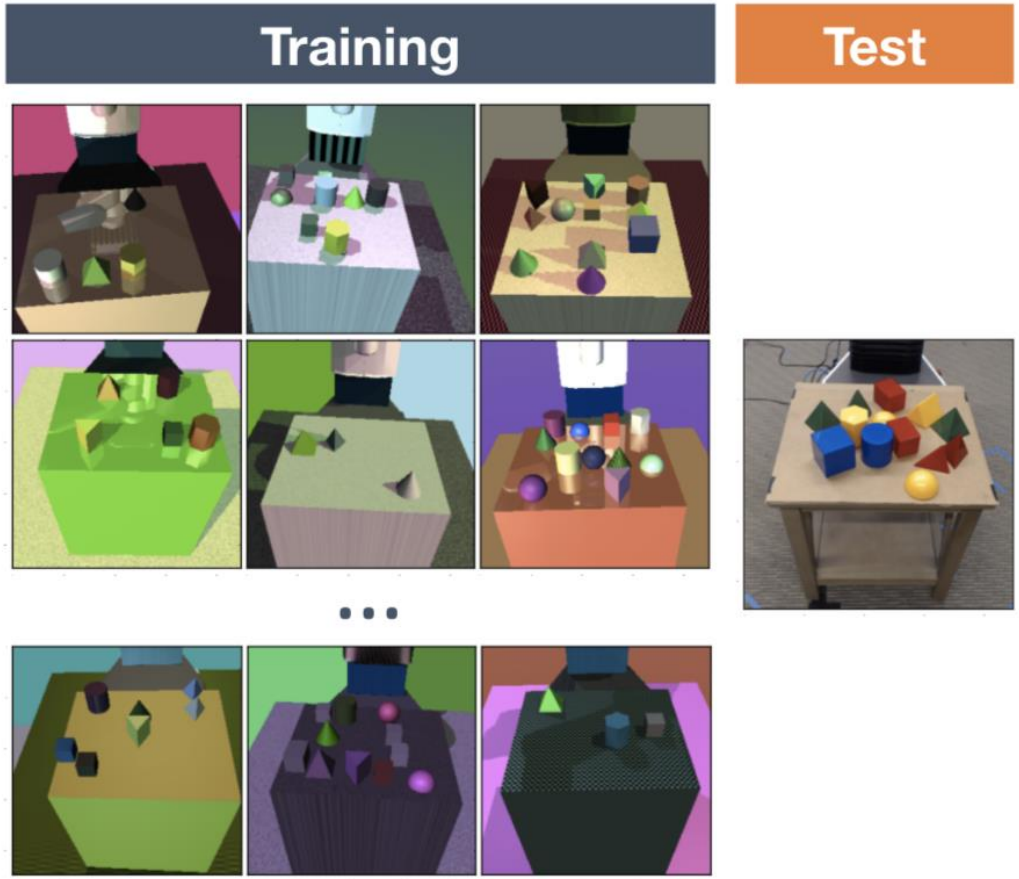
PHYSICAL REALISM: GAME ENGINES, CAD + PHYSICS MODELS

Combining both aspects of realism?

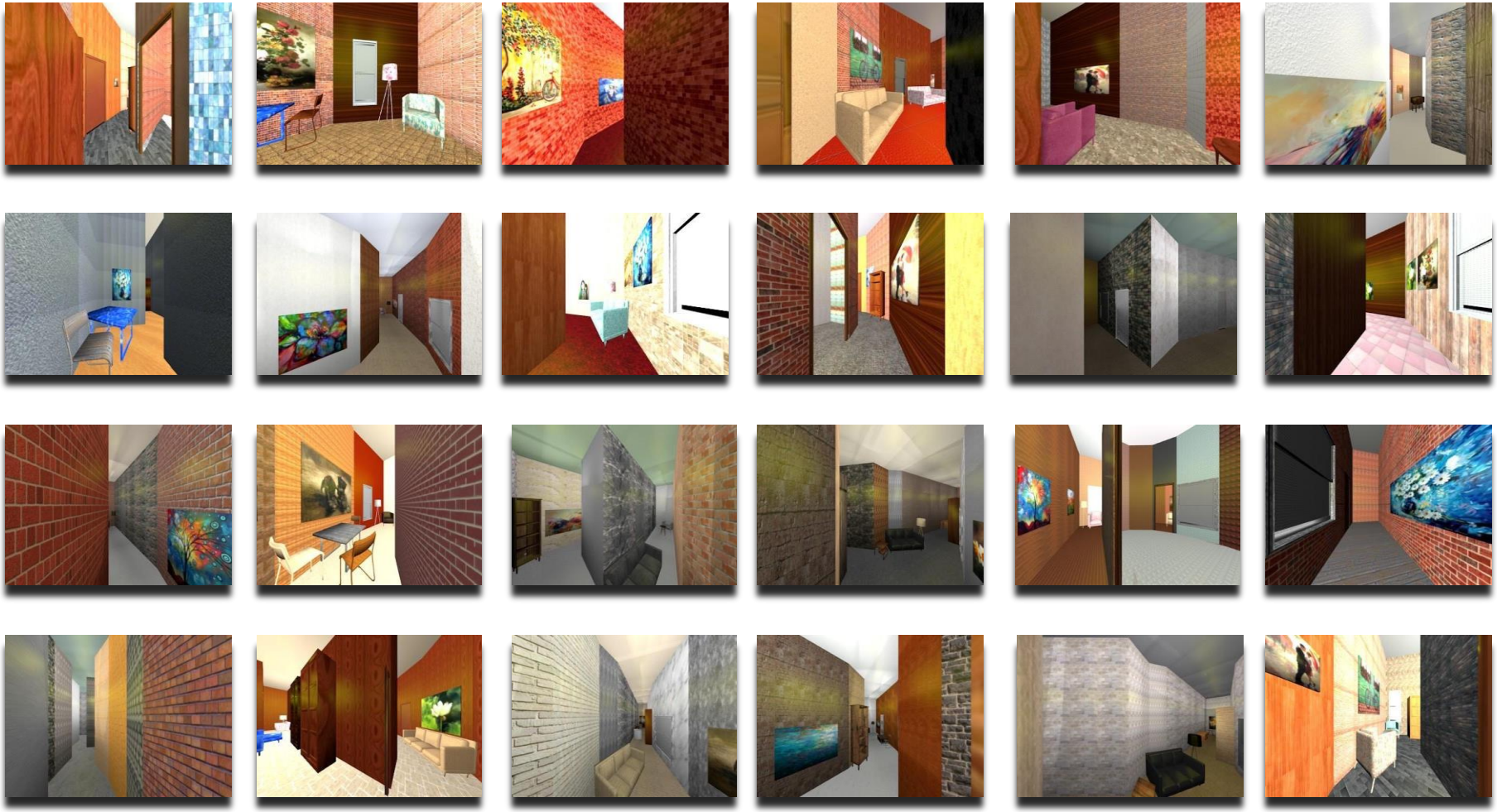
Source: iGibson, http://svl.stanford.edu/igibson/



Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks



Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World



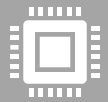
CAD2RL: Real Single-Image Flight Without a Single Real Image

Outline

- Careful choice of paradigm
- Using knowledge from other domains
- Human demonstrations and feedback
- Scaling data collection
  - Sim2real
  - Parallelized methods

Parallelized methods with multiple devices

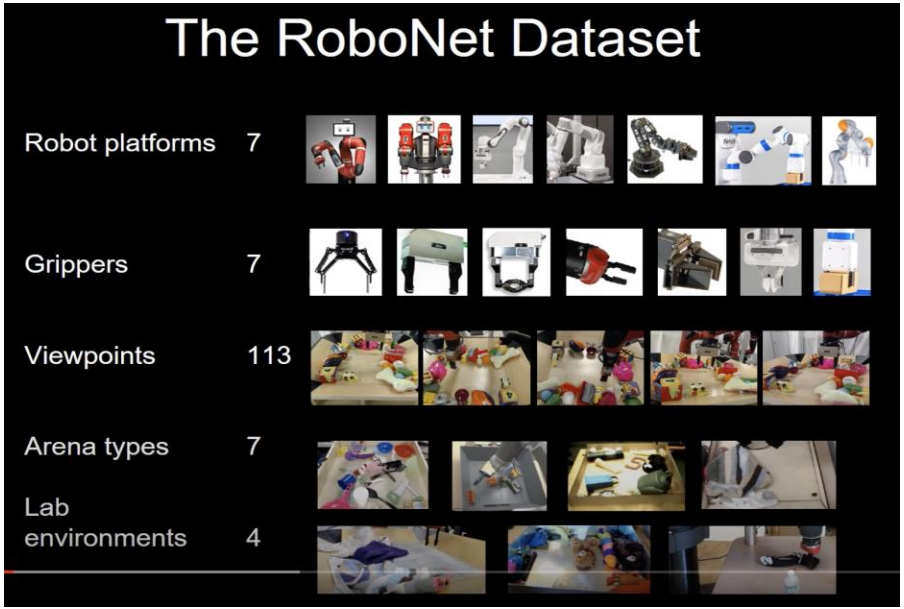
Parallelized, asynchronous data collection: edge workers merely send data to server



Federated learning: edge workers update personal models; asynchronously send model parameters to update the global features on server

Parallelized, asynchronous data collection

Distributed Q-learning algorithm with Google Arm Farm for grasping from vision



ROBONET: Scaling up data collection with multiple robots

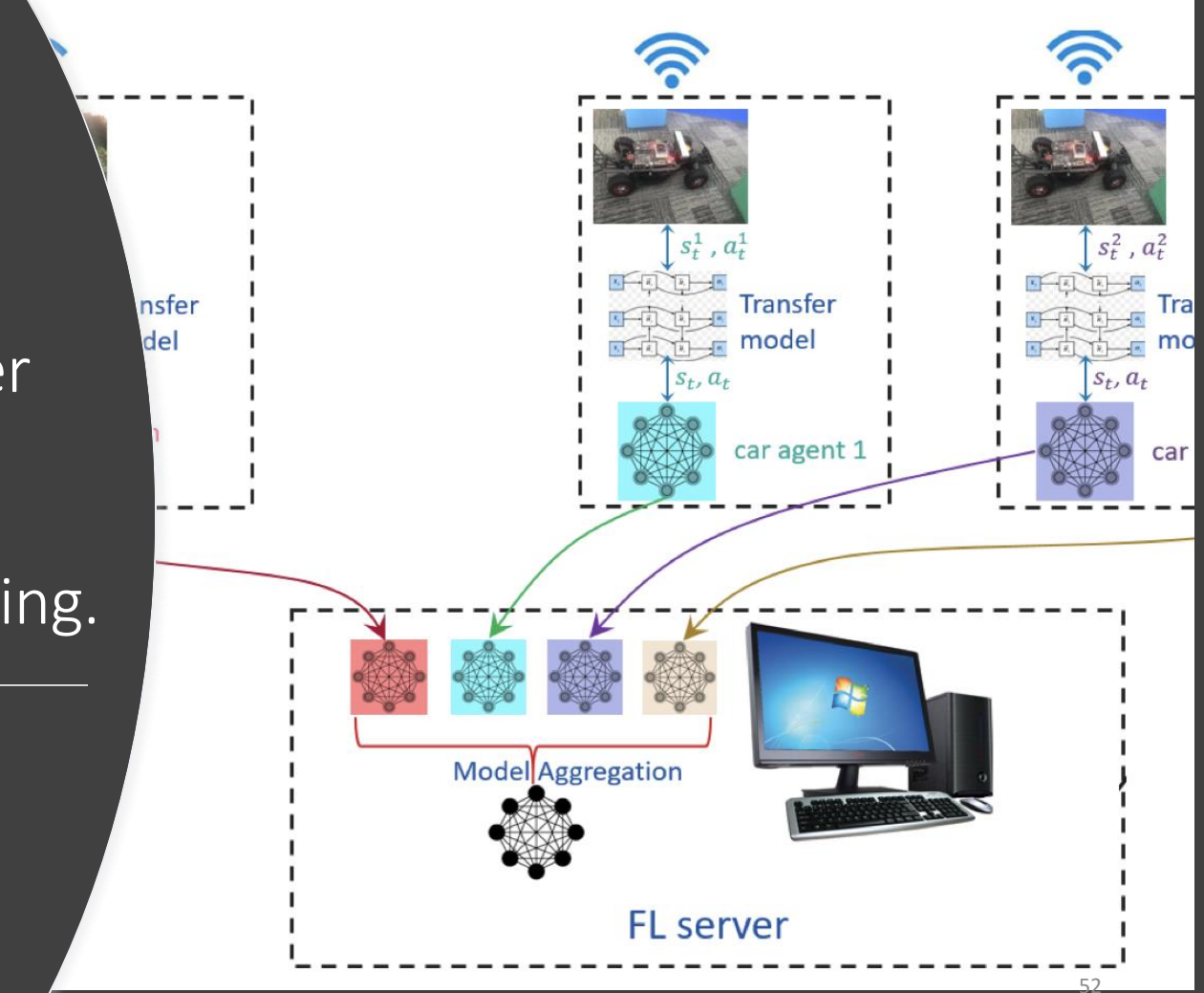
Federated learning

Local adaptation of robot

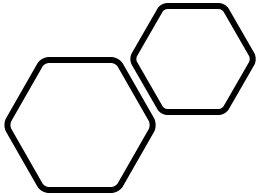
Global features in communication-efficient way

Privacy preserving way to leverage personal data

Federated Transfer Reinforcement Learning for Autonomous Driving.



Discussion section format



2 sets of questions. For each:

- Break-out room – 8 minutes
- Reconvene + share – 5 minutes

1. When is DRL useful/necessary for embodied AI applications? (i.e. when do data-driven methods have an advantage over traditional planning & control methods?)

2. Is sample inefficiency a bottleneck in the progress of DRL for robotics?

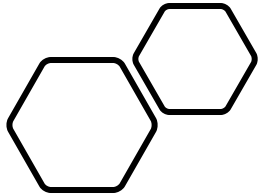
3. Should our focus as a community be on circumventing sample efficiency issues (i.e. thru gathering data at scale) or addressing it head-on?

4. Does sim2real work? If sim2real works, can't we just use any of our DRL algorithms, even if data inefficient?

Questions part 2

5. Are there any approaches that we missed?
6. Do you see ways in which these methods can be combined?
7. What's wrong with the way we currently measure/quantify sample efficiency?
8. Federated learning has shown early promise in areas like query suggestions on mobile phones, smart speakers, etc. What other applications can you think of?

Wrapping up



- Slides will be posted to our WiML Slack
    - Channel name: #breakout_session_4-3
- You can contact us by private message on WiML Slack