How to Train Your Robot with Deep Reinforcement Learning – Lessons We've Learned

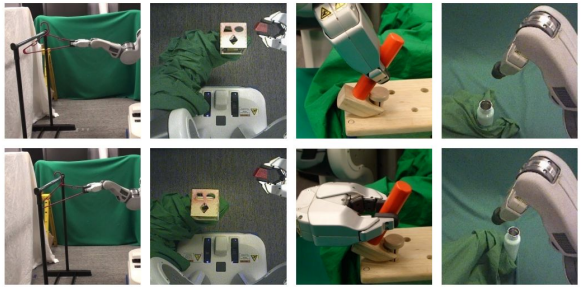
Authors: Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, Sergey Levine
Presenters: Sawan Patel, Oliver Wang

Case Studies

Learning Manipulation Skills

- Guided Policy Search
  - ‘Global’ policy ‘guided’ by some model-based RL algorithm
  - Global policy trained on raw sensory observations to perform some task with moderate variability in initial conditions
  - Supervision of global policy provided by individual ‘model-based’ learners that learn local policies
    - Learns ‘alongside’ global policy
- Model-free Skill Learning
  - Optimal policy learned through interaction with environment
  - Advantages over GPS
    - More robust to increasing size of initial state space
    - Don’t require lower-dimensional state for optimization
  - On-policy vs Off-policy
- Visual Foresight Approach
  - Learning from data at scale to perform multiple related tasks (i.e. vision-based manipulation skills)
    - Removes assumptions of environmental reset ability or engineering of environment to measure reward

Figure 1. A PR2 learns to place a red trapezoid block into a shape-sorting cube. With Levine et al. (2016), it learns local policies for each initial position of the cube, which can be reset automatically using the robot’s left arm. The local policies are distilled into a global policy that takes images as input, rather than the cube’s location.



(a) hanger (b) cube (c) hammer (d) bottle

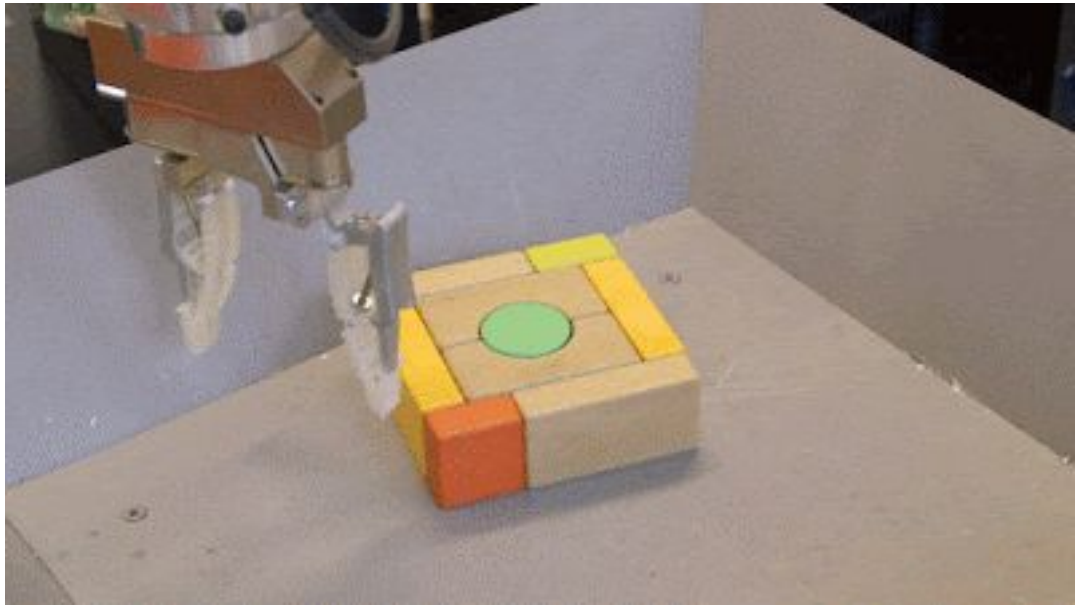
Figure 8: Illustration of the tasks in our visuomotor policy experiments, showing the variation in the position of the target for the hanger, cube, and bottle tasks, as well as two of the three grasps for the hammer, which also included variation in position (not shown).

Grasping — QT-Opt

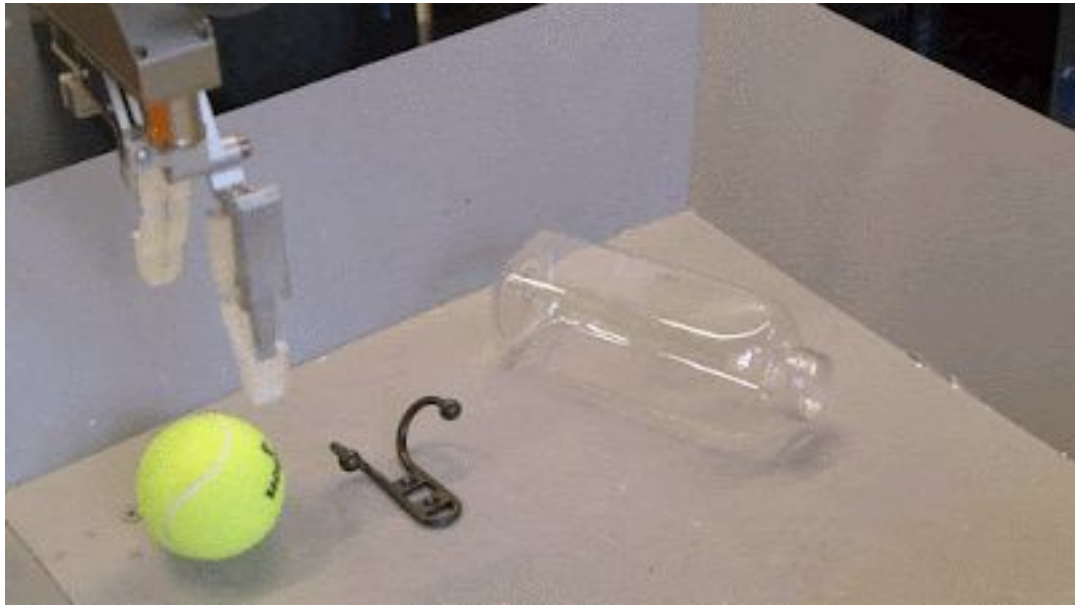
- Combination of offline and online data
- Closed loop control (continuously using feedback to update trajectory)
- QT-Opt
  - 86% grasp accuracy purely offline, 96% with additional 28,000 online grasps
  - Pre-grasp manipulation, grasp readjustment, picking up dynamic objects, recovery from perturbations, grasping in clutter
  - Optimizes reward along entire trajectory, replanning mid-grasp

Figure 4: Eight grasps from the QT-Opt policy, illustrating some of the strategies discovered by our method: pregrasp manipulation (a, b), grasp readjustment (c, d), grasping dynamic objects and recovery from perturbations (e, f), and grasping in clutter (g, h). See discussion in the text and Appendix A.

Grasping — QT-Opt



Grasping — QT-Opt



Legged Locomotion

- Training in simulation
  - Quadruped robot can learn to walk in sim in 2-3hrs
  - Reality gap
- Difficulties in learning locomotion
  - Sample Efficiency: tens of millions of data samples to learn meaningful locomotion gaits, hyperparameter tuning
  - Robot Safety: jerky actuation patterns from robot exploration, lack of balance
  - Asynchronous control: delay between perception and action violates core assumption of Markovian Decision Process, dropping performance

Figure 5. The Minitaur robot learns to walk from scratch using deep RL.

Outstanding Challenges in Deep RL + Strategies to Mitigate Them

Reliable and Stable Learning

- Performance of Deep RL algorithms susceptible to variability due to inherent randomness of methods, requires careful tuning of hyperparameters
- Challenges:
  - Reducing hyperparameter sensitivity
    - Running multiple learning processes with different hyperparameters on same data buffer
      - Limitation: restricted to off-policy methods
    - Develop algorithms that automatically tune hyperparameters
      - Issue: Challenging to understand sensitivity of RL algorithms exactly
  - Reducing issues prompted by local optima and delayed rewards
    - RL algorithm optimization is challenging, even if policy function is over-parameterized
    - Solutions:
      - Stochastic policies that can explore multiple strategies simultaneously

Sample Efficiency

- Popular RL algorithms require millions of SGD steps
  - Often infeasible in a real-world setting
  - On-policy < off-policy < model based
- Off-Policy Algorithms
  - One behavior sample can be used for multiple gradient descent steps, compared to one training step per sample in on-policy learning
- Model Based Algorithms
  - Much less data needed for training if you have/generate a model of the environment
  - Difficult to learn an accurate model
- Input Remapping
  - Ex. privileged teacher → student which learns to transform raw input into teacher output
  - Encoding image observations into lower dimensional feature space
- Offline Training
  - Using past collected data to pre-train RL tasks

Use of Simulation

- Collecting real data is challenging - training algorithms through simulation can be far more effective
  - Issue: **Reality Gap**
    - Simulations don't account for detailed dynamics, partial observability, latency and safety aspects of robotics
  - Addressing partial observability
    - Limitation of onboard sensors results in partial observation of state in real experiments (noisy, delayed readings)
  - Better simulation
    - Real-world physics vs simulation 'physics'
      - Incorrect physical parameters, unmodeled dynamics and general stochasticity
    - Authors suspect that actuator dynamics and lack of latency modeling are primary issues
  - Domain randomization
    - Randomly sample different simulation parameters while training RL policy (i.e. environment texture, lighting)
  - Domain adaptation
    - Creation of 'adaptor networks' that convert simulated images to more resemble their real-world 'counterparts' ***or*** converting real images to 'simulated' images for evaluation via algorithms trained in simulation

Side-Stepping Exploration Challenges

- Sparse reward conditions prove challenging for typically ‘effective’ exploration methods
  - Many tasks begin by specifying a binary reward condition
    - e.g. grasping robot can either succeed or fail
- Problem can be ‘side-stepped’ via manual engineering and/or demonstration data
  - Idea 1: Initialization
    - Pre-train policy network with demonstrations via ‘imitation learning’
      - Issues:
        - Lacks effective guarantees in performance in theory and in practice
        - Learned initialization be easily forgotten (given high exploration factor)
  - Idea 2: Scripted policies
    - Designing ‘scripted’ policies to serve as initialization
      - E.g. in grasping system, pre-populate data buffer with higher proportion of successful grasps than what would be obtained via random exploration
  - Idea 3: Reward shaping
    - Shaping reward function provides RL algorithm additional guidance for exploration
      - E.g. In reaching task, use distance to goal as a negative reward to speed up exploration
  - Others: Joint training and demonstrations in model-based RL

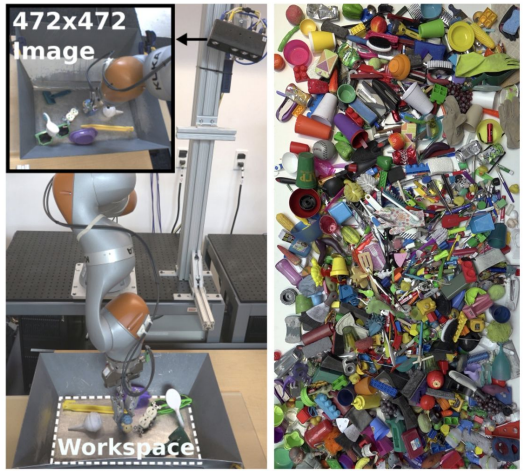


Figure 3. Close-up of our robot grasping setup in our setup (left) and about 1000 visually and physically diverse training objects (right). Each robot consists of a KUKA LBR IIWA arm with a two-finger gripper and an over-the-shoulder RGB camera.

Improving Generalization

- Data diversity
  - Good generalization requires large diversity of samples
  - e.g. 1000 ImageNet classes vs. 100 CIFAR classes
  - May require larger and deeper networks to achieve good performance with varied data
  - Varied data in-simulation
- Correct train/test evaluation protocol
  - Separate MDPs for train and test based on what features you care about
    - If you care about object generalization: different objects between train and test
    - If you care about generalization of dynamics: different positions or situations

Avoiding Model Exploitation

- Model exploitation: learned model is imperfect in some areas of state space
  - Can lead to poor action selection
- Mitigations:
  - Broad data distribution over actions and states
  - Interleave data collection and model learning
    - Whenever the model is inaccurate, collect more real world data to train the model
  - Estimate + account for the uncertainty in your model
    - Difficult to do accurately
  - Multistep losses, shorter horizons, and replanning can help limit error accumulation from an inaccurate model over longer horizon

Robot Operation at Scale

- Reduced barrier of entry in deep RL through data efficiency
- Issues:
  - RL algorithms require fair amount of data - particularly when dealing with images
  - Human operator required to monitor robot
    - E.g. Has to reset the scene, stop the robot in hazardous situations, or reset/restart robot
- Experiment design
  - Setup can be engineered or task can be chosen such that robot can reset scene
- Facilitating continuous operation
  - Repeated contact with objects/environment can wear out experimental setup, should be considered up front
    - Adding ‘sanity checks’
  - Increases mean-time-between-failure and ensure collected data is useful
    - Develop fail-safe redundancies
- Non-stationarity due to environment changes
  - e.g. hardware degradation or changes in lighting conditions can doom learned policies

Robot Operation at Scale

Asynchronous Control

- Delays between execution of action and sensor readings violate MDP formulation
  - i.e. existence of latency means next state of system does not necessarily depend on measured state
  - Planning component for model-based methods is computationally expensive, even if parallelized via GPU
- Solutions:
  - Model-based: Plan optimal action sequence based on a future state predicted via learned dynamics model
  - Model-free: Add recurrence to policy network, including previous actions taken
    - Or augment observation space with window of prior observations/actions

Setting Goals and Specifying Rewards

- Few rewards in the real world
- Setting up the environment with sensors to provide rewards
  - Measuring door and handle angle to provide feedback in door opening task
  - Motion capture device to measure locomotion speed
- Simple heuristics as rewards
  - Gripper encoder values
  - Analyzing successful grasp from raw images
- Learning a visual prediction model
- Learning rewards from demonstrations, annotations, human feedback and preferences

Multi-Task Learning and Meta Learning

- Multi-task learning approaches learn multiple tasks at once
- Meta-learning algorithms train across multiple tasks such that a novel future task can be performed efficiently
- Challenges:
  - Specifying task collection
    - Specifying a reward function for a single task is already challenging
  - Optimization landscape of multiple tasks
    - Different tasks learned at different rates
    - How to resolve conflicting gradient signals between tasks?

Safe Learning

- Safe action spaces
  - Selecting actions through sampling, rejecting “unsafe” ones
  - Velocity constraints and limits on where/how the robot can move
- Smooth actions
  - Penalize jerkiness in reward function, sampling actions from “smooth noise”, smoothing actions with filters, gait controller + feedback policy for adjustments
- Recognizing unsafe actions
  - e.g. learn a policy to stand up when the robot is losing balance
- Constraining learned policies
- Robustness to unseen observations
  - Test safety behaviors in random situations

Robot Persistence

- Training is highly dependent on quality and quantity of samples collected
- Self persistence: the robot needs to keep itself alive
- Task persistence: accomplishing a range of tasks repeatedly
  - Ideally able to reset the environment without much human intervention
    - Repositioning a gripper over a bin
  - Automating reset may be as hard as the task itself
    - Screwing/unscrewing bottle cap
  - Irreversible state = hard to collect lots of data
    - Welding two pieces of metal

Discussion

- Deep RL and real-world learning scenarios
  - e.g. walking, dexterous manipulation
  - Generalization ability
- Challenges
  - Efficiency/stability remain challenges even if all else is optimal
  - Simulation training aids with efficiency but faces issues during deployment
  - Exploration challenges can be ‘side-stepped’ in practical problems
  - Issues with generalization may be resolved with the expansion of data quantity
  - Violation of synchronous MDP model assumption
  - Definition of reward function across multiple tasks
- Future
  - Expansion in capabilities of robots can increase with more robot time ‘available’ to learn skills

Questions

1. Has anyone in the class had their own challenges with training a deep RL model? Was your experience similar to that of the paper authors?

2. Say we are designing a deep RL model for the task of autonomous vehicles.
    a. Which of the aforementioned challenges might be more critical to address than others?
    b. At what point is the model “good enough” to release in the real world?