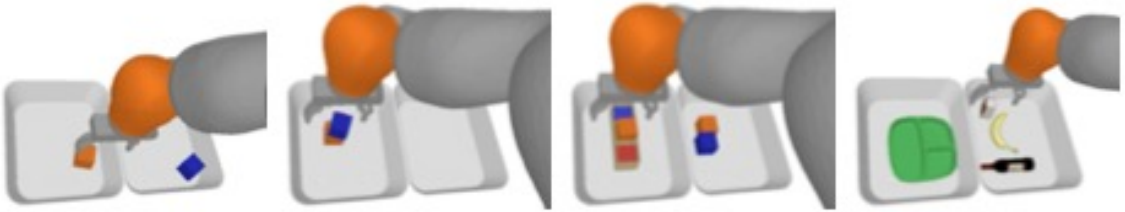
Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills

Presenter: Chang Shi

10/19/2021

Motivation

General-purpose robots need large repertoires of skills.



Simulated tasks: pick-and-place, stacking, fixture placing, food object grasping.

Learning each skill individually can be prohibitive, particularly when

- Task rewards must be programmed by hand
- Data must be collected anew for each task

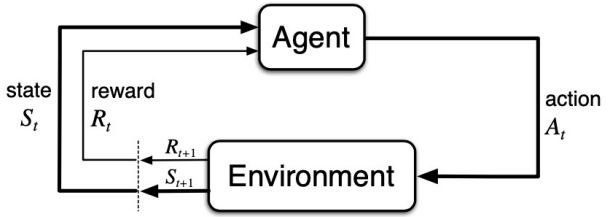
Can we reuse past robotic data efficiently? → **Offline learning**

Main Problem

The problem of learning useful robotic skills from previously collected **offline** data **without access to manually specified rewards** or **additional online exploration**, by reusing past robotic data.

- How can we obtain a general-purpose training objective in robotics?
- How can we train diverse skills so that they are represented by a single model?
- How can we employ this model to perform zero-shot generalization or solve downstream tasks?

Problem Setting



$M = (S, A, P, R, p_0, \gamma, T)$	A Markov decision process (MDP)
S	State space
A	Action space
$P : S \times A \times S \longrightarrow \mathbb{R}_+$	State-transition probability function
$R : S \times A \longrightarrow \mathbb{R}$	Reward function
$p_0 : S \longrightarrow \mathbb{R}_+$	Initial state distribution
γ	Discount factor
T	Task horizon
$\tau = (s_0, a_0, \cdots, s_T, a_T)$	A trajectory
$R(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$	Trajectory reward

Reinforcement learning methods find a policy $\pi(a|s)$ that maximizes the expected discounted reward over trajectories induced by the policy:

$$\mathbb{E}_\pi[R(\tau)]$$

Where $s_0 \sim p_0$, $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ and $a_t \sim \pi(a_t \mid s_t)$.

- Offline
- Model-free

Model-based vs Model-free in the offline setting

Goal: Utilize past data from prior tasks to acquire transferable knowledge for new tasks

Model-based: Train a predictive model for the transition dynamics

- Pros: Allow agent to plan ahead
- Cons: Have to predict future world observations (e.g. images) in all of their complexity

  Most requires an additional cost function for action selection

Model-free: Task-oriented solution, no access to the transition dynamics

- Pros: Avoid complex future prediction
- Cons: Distributional shift -> Overestimation of values

Mutual problem: Long horizons

Goal conditioned strategy

- Often times we care about policies that achieve many related goals
- Training such policies jointly may be beneficial

$(\mathrm{s}, \mathrm{a}, \mathrm{r}, \mathrm{s}') \rightarrow (\mathrm{s}, \mathrm{g}, \mathrm{a}, \mathrm{r}, \mathrm{s}')$

$\pi(a|s) \rightarrow \pi(a|s, \mathrm{g})$

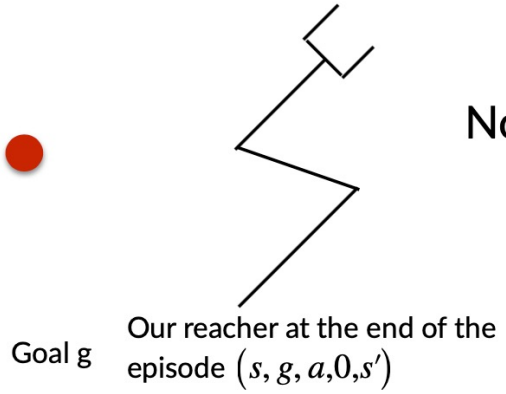
$\mathrm{Q}(\mathrm{s}, \mathrm{a}) \rightarrow \mathrm{Q}(\mathrm{s}, \mathrm{g}, \mathrm{a})$

$\mathrm{V}(s) \rightarrow \mathrm{V}(s, \mathrm{g})$

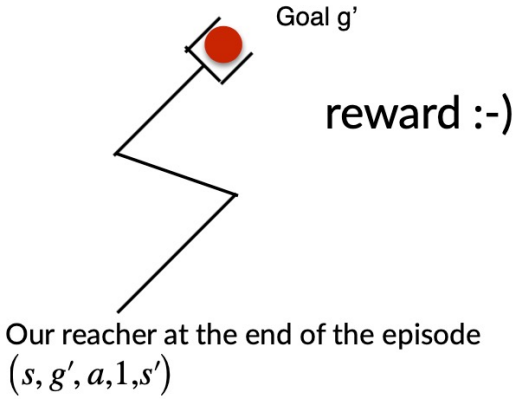
Goal relabeling

- Goal space involves only a small fraction of the state space
- All trials without reaching the goal would return no reward

Low sample efficiency!



No reward :-(



reward :-)

Hindsight Experience Replay (HER)

Algorithm 1 Hindsight Experience Replay (HER)

Given:

- an off-policy RL algorithm $\mathbb{A}$, ▷ e.g. DQN, DDPG, NAF, SDQN
- a strategy $\mathbb{S}$ for sampling goals for replay, ▷ e.g. $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$. ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$

Initialize $\mathbb{A}$ ▷ e.g. initialize neural networks
Initialize replay buffer R
for episode = 1, M **do**
  Sample a goal g and an initial state s_0.
  for $t = 0, T-1$ **do**
    Sample an action a_t using the behavioral policy from $\mathbb{A}$:
      $a_t \leftarrow \pi_b(s_t||g)$ ▷ $||$ denotes concatenation
    Execute the action a_t and observe a new state s_{t+1}
  end for
  for $t = 0, T-1$ **do**
    $r_t := r(s_t, a_t, g)$
    Store the transition $(s_t||g, a_t, r_t, s_{t+1}||g)$ in R ▷ standard experience replay
    Sample a set of additional goals for replay $G := \mathbb{S}$(**current episode**)
    for $g' \in G$ **do**
      $r' := r(s_t, a_t, g')$
      Store the transition $(s_t||g', a_t, r', s_{t+1}||g')$ in R ▷ HER
    end for
  end for
  for $t = 1, N$ **do**
    Sample a minibatch B from the replay buffer R
    Perform one step of optimization using $\mathbb{A}$ and minibatch B
  end for
end for

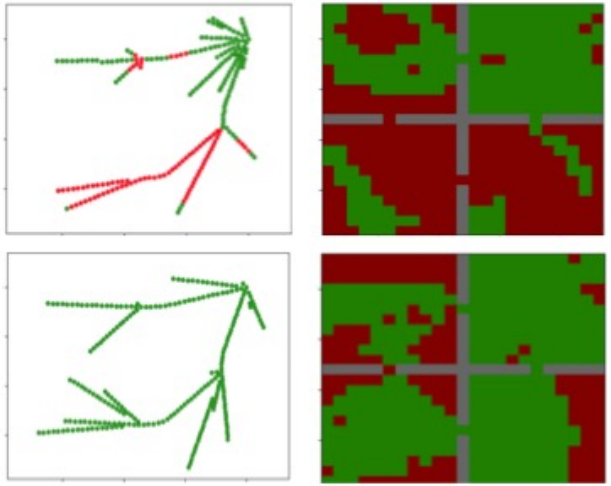
Main idea: use failed executions under one goal g, as successful executions under an alternative goal g′.

$$g' \in \mathbb{S}(\text{current episode})$$

Hindsight Experience Replay (HER)

Behavior Cloning

Behavior Cloning with goal relabeling



(a) Performance on reaching states visited in the 20 given demonstrations. The states are green if reached by the policy when attempting so, and red otherwise.

(b) Performance on reaching any possible state. Each cell is colored green if the policy can reach the center of it when attempting so, and red otherwise.

Hindsight relabeling only generates examples for actions that are needed to reach a goal, and does not provide evidence for which actions are sub-optimal or do not lead to a desired goal (**Only "positive" no "negative" samples**), which might result in **overestimation of their Q-values**.

Conservative Q-Learning (CQL)

Off-policy RL methods can fail due to overestimation of values induced by the distributional shift between the dataset and the learned policy, especially when training on complex and multi-modal data distributions.

Conservative Q-learning learns a conservative Q-function such that the expected value of a policy under this Q-function lower-bounds its true value.

Regularizes the Q-function on out-of-distribution actions

Conservative Q-Learning (CQL)

Additionally minimizing Q-values alongside a standard Bellman error objective.

$$\hat{Q}^{k+1} \leftarrow \arg\min_{Q} \; \alpha \, \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] + \frac{1}{2}\,\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s},\mathbf{a})\right)^{2}\right]$$

We can show that $\hat{Q}^{\pi} \coloneqq \lim_{k\to\infty} \hat{Q}^{k}$, lower-bonds Q^{π} at all (s, a).

Theorem 3.1. *For any $\mu(\mathbf{a}|\mathbf{s})$ with $\operatorname{supp}\mu \subset \operatorname{supp}\hat{\pi}_{\beta}$, with probability $\geq 1-\delta$, $\hat{Q}^{\pi}$ (the Q-function obtained by iterating Equation 1) satisifies:*

$$\forall \mathbf{s}\in\mathcal{D}, \mathbf{a}, \;\; \hat{Q}^{\pi}(s,a) \leq Q^{\pi}(\mathbf{s},\mathbf{a}) - \alpha\left[(I-\gamma P^{\pi})^{-1}\,\frac{\mu}{\hat{\pi}_{\beta}}\right](\mathbf{s},\mathbf{a}) + \left[(I-\gamma P^{\pi})^{-1}\,\frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right](\mathbf{s},\mathbf{a}).$$

Thus, if α is sufficiently large, then $\hat{Q}^{\pi}(\mathbf{s},\mathbf{a}) \leq Q^{\pi}(\mathbf{s},\mathbf{a}), \forall \mathbf{s}\in\mathcal{D}, \mathbf{a}$. When $\hat{\mathcal{B}}^{\pi} = \mathcal{B}^{\pi}$, any $\alpha > 0$ guarantees $\hat{Q}^{\pi}(\mathbf{s},\mathbf{a}) \leq Q^{\pi}(\mathbf{s},\mathbf{a}), \forall \mathbf{s}\in\mathcal{D}, \mathbf{a}\in\mathcal{A}$.

Proposed Approach: Actionable Models

The method takes as input a dataset of trajectories $D = \{\tau\}$, and outputs a goal-conditioned Q-function, $Q_\theta(s, a, g)$. Note that acting greedily w.r.t. this goal-conditioned Q-function provides us with a goal-conditioned policy: $\pi(\mathrm{a}|\mathrm{s}, \mathrm{g}) = argmax_a \mathrm{Q}(s, a, g)$. We train this Q-function by minimizing the following loss:

$$\mathcal{L}_g(\theta) = \min_\theta \mathbb{E}_{(s_t, a_t, s_{t+1}, g) \sim \mathcal{D}} [(Q_\theta(s_t, a_t, g) - y(s_{t+1}, g))^2 + \mathbb{E}_{\tilde{a} \sim \exp(Q_\theta)} [(Q_\theta(s, \tilde{a}, g) - 0)^2]], \quad (1)$$

where the TD-target y is given by:

$$y(s_{t+1}, g) = \begin{cases} 1 & \text{if } s_{t+1} = g \\ \gamma \mathbb{E}_{a \sim \pi} [Q_\theta(s_{t+1}, a, g)] & \text{otherwise.} \end{cases}$$

No manually specified rewards

Only one sparse reward function per goal is defined:

$$R(\tau, g) = R(s_T, a_T, g) = \mathbb{1}[s_T = g], g \in \mathcal{G}.$$

Use temporal-difference (TD) learning to maximize the expected return yielding a goal-conditioned Q-function, which describes the probability of reaching the goal state g at time t = T :

$$\begin{aligned} Q^{\pi}(s_t, a_t, g) &= \mathbb{E}_{\pi}\left[\sum_t \gamma^t R(s_t, a_t, g)\right] \\ &= P^{\pi}(s_T = g | s_t, a_t). \end{aligned}$$

Then obtain a goal-reaching policy by acting greedily with respect to the goal-conditioned Q-function:

$$\pi(a|s, g) = \arg\max_a Q(s, a, g).$$

Conservative approach for unseen actions

Let $\tilde{A}(s, g)$ denote a set of unseen actions, for which we do not have evidence of reaching the goal g from state s in the dataset D, and $\tilde{a} \sim p_{\tilde{A}}(\tilde{a}|s, g)$ some probability distribution with the support on this set, which we will describe below. Furthermore, let $\tilde{\mathcal{G}}(g)$ be a set of goals different from g and $p_{\tilde{\mathcal{G}}}(g)$ a distribution with the support on this set.

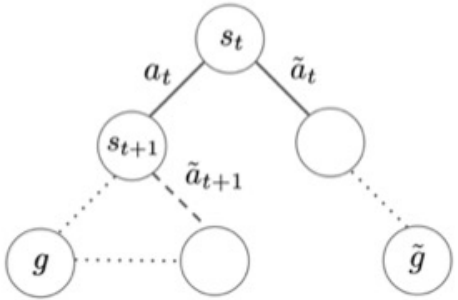


Figure 2. Two scenarios that can occur when deviating from the original action on the way to the goal g: without recovery ($\tilde{a}_t$) and with recovery ($\tilde{a}_{t+1}$).

Conservative approach for unseen actions

We assume there is no recovery unless there are trajectories in the dataset D that demonstrate this recovery, which would be taken into account through labeling them with the success-reward as described above. This amounts to assuming that any deviation $\tilde{a} \sim p_{\tilde{A}}(\tilde{a}|s, g)$ leads to some other goal $\tilde{g} \sim p_{\tilde{\mathcal{G}}}(g)$:

$$\mathbb{E}_{\tilde{a}_t \sim p_{\tilde{A}}}\left[P^{\pi}(s_T \neq g | s_t, \tilde{a}_t)\right] = 1.$$

With high-dimensional goals like images, the set of all other goals $\tilde{\mathcal{G}}(g)$ becomes intractable. We can circumvent dealing with $\tilde{\mathcal{G}}(g)$ by the following transformation:

$$\begin{aligned}
&\mathbb{E}_{\tilde{a}_t \sim p_{\tilde{A}}}\left[P^{\pi}(s_T \neq g | s_t, \tilde{a}_t)\right] \\
&\quad = \mathbb{E}_{\tilde{a}_t \sim p_{\tilde{A}}}\left[1 - P^{\pi}(s_T = g | s_t, \tilde{a}_t)\right] \\
&\quad = \mathbb{E}_{\tilde{a}_t \sim p_{\tilde{A}}}\left[1 - Q^{\pi}(s_t, \tilde{a}_t, g)\right] = 1 \\
&\quad \Rightarrow \mathbb{E}_{\tilde{a}_t \sim p_{\tilde{A}}}\left[Q^{\pi}(s_t, \tilde{a}_t, g)\right] = 0.
\end{aligned}$$

we aim at minimizing Q-values for unseen actions

Goal chaining for long-horizon tasks

Given a sequence $\tau_{0:1}$, instead of limiting the goal to be within the sequence (i.e., $g \sim (s_0, \cdots, s_i)$), we redefine it to be any state observed in the dataset (i.e., $g \sim D$). If g = si (e.g. when the goal is the final state of the sub-sequence) then similarly as before, we label such trajectories with $R(\tau_{0:1}, s_i) = 1$. Otherwise, since we do not know whether $g \neq s_i$ can be reached from the states within $\tau_{0:1}$, instead of assigning a constant reward, we set the reward of the final transition to be its Q-value, such that

$$R(s_i, a_i, g) = Q^{\pi}(s_i, a_i, g):$$

$$R(\tau_{0:i}, g) = \begin{cases} 1, & \text{if } s_i = g \\ Q^{\pi}(s_i, a_i, g), & \text{otherwise.} \end{cases}$$

This procedure follows the intuition that if there is evidence in the dataset that g is reachable from (s_i, a_i), it will eventually propagate into the Q-values of $\tau_{0:1}$.

Goal chaining

- The dynamic programming nature of Q-learning is able to chain trajectories in state space
- In the case when we use function approximators for learning Q-functions, the chaining points does not have to be exactly the same in both trajectories in order to propagate useful reachability information.

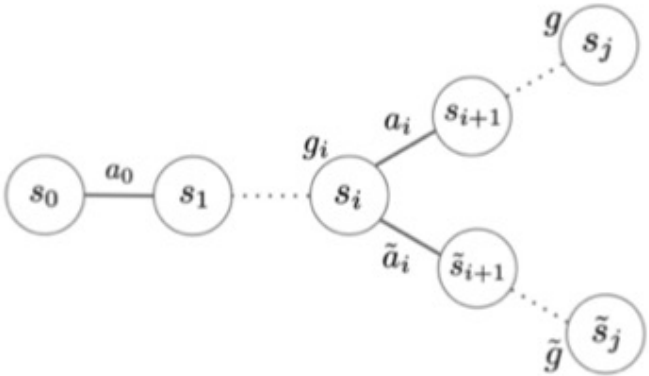


Figure 3. Illustrating goal chaining between two trajectories: $\tau_{0:i}$ and $\tau_{i:j}$ through the chaining point s_i.

Application: Goal reaching

Algorithm 1 Goal reaching with Actionable Models

function EXTRACTEXAMPLES
  $\tau \leftarrow \mathcal{D}$: Sample a trajectory from the dataset.
  $g_{rand} \leftarrow \mathcal{D}$: Sample a random goal from the dataset.
  $\tau_{0:i} \leftarrow$ Randomly cut the trajectory with $i \in \{1, T\}$.
  $R(\tau_{0:i}, s_i) = 1$: Label reaching final state with 1.
  $R(\tau_{0:i}, g_{rand}) = Q_\theta(s_i, a_i, g_{rand})$:
  Label reaching g_{rand} with Q-value at the final state.
  Add transitions from relabeled trajectories to replay buffer.
function COMPUTEQTARGETS
  $(s_t, a_t, g, s_{t+1}, R(s_t, a_t, g)) \leftarrow$
  Sample a transition from replay buffer.
  $Q_{target}(s_t, a_t, g) \leftarrow$
  $R(s_t, a_t, g) + \max_a Q(s_{t+1}, a, g)$
  $\tilde{a}_t \sim \exp(Q_\theta(s_t, \tilde{a}_t, g))$: Sample a negative action.
  $Q_{target}(s_t, \tilde{a}_t, g) \leftarrow 0$:
  Set the target for the negative action to 0.

This method can be integrated into any Q- learning method with an experience replay buffer

Application: Pre-training / Auxiliary objective

- Pre-training: Given a large dataset D of previous experience, pre-train a goal-conditioned Q-function $Q_\theta(S, A, G)$ using our offline method, and then further fine-tune it on a specific task reward.
- Auxiliary objective: Utilize our method to provide an auxiliary objective that can be used in parallel with conventional online RL to encourage learning of functional representations. Given a small mix-in probability ξ we can augment the task-specific objective $L_{task}(\theta)$ with the regularized goal-reaching objective $L_g(\theta)$ from the following joint objective:

$$L_{augmented}(\theta) = L_{task}(\theta) + \xi L_g(\theta)$$

Experimental Setup

- Policy architecture follows the QT-Opt framework, with an additional Q-function input for the 472x472x3 goal image.

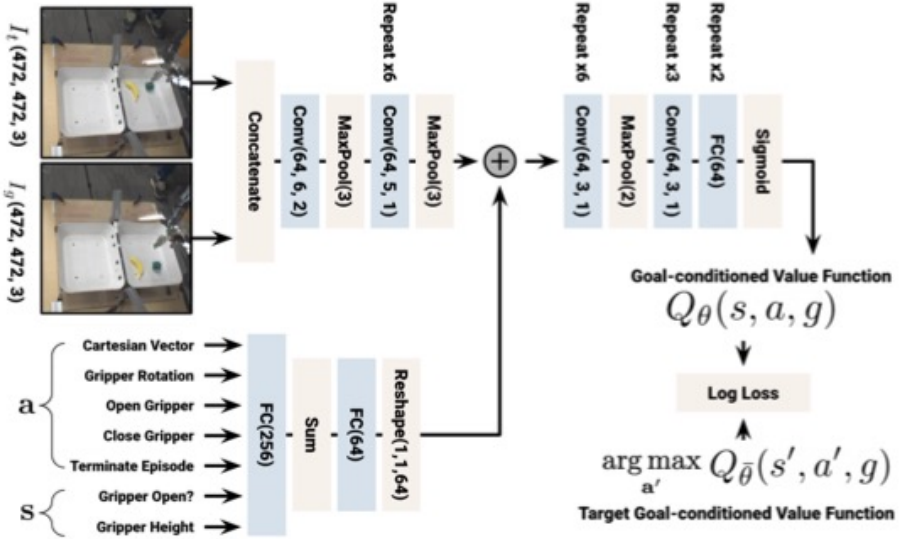
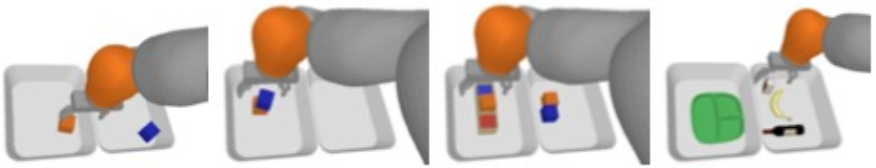
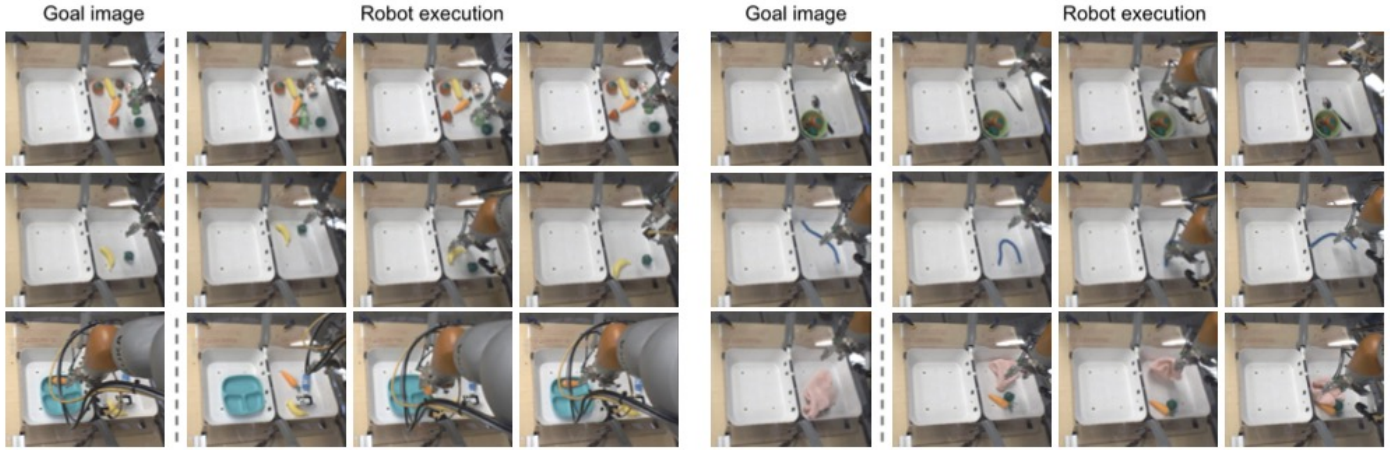


Figure 4. The Q-function neural network architecture follows the QT-Opt framework with an additional goal image input.

Experimental Setup



- Simulated tasks: pick-and-place, stacking, fixture placing, food object grasping
  - baseline: GCBC, Q-learning with HER, Q-learning with HER + random goal negatives
- Real robot goal reaching task



(a) Instance grasping, Rearrangement, Container placing

(b) Unseen object manipulation

Experimental Results

(a) Pick-and-place (b) Stacking (c) Fixture placing (d) Food object grasping

Figure 5. Comparison of goal-conditioned policies trained from offline data in simulation.

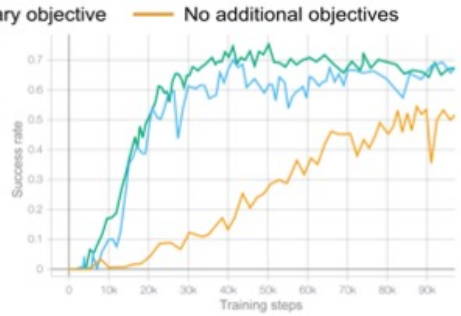
Experimental Results

Task	Success rate
Instance grasping	92%
Rearrangement	74%
Container placing	66%

(a) Real world goal reaching

(b) Goal chaining ablation

(c) Bottle grasping



(d) Banana grasping

Figure 8. a): Success rates of real world goal reaching skills. *b):* Comparison of training the fixture placing task with separate episodes for grasping and placing, with (blue) and without (yellow) goal chaining enabled. *c) and d)*: Comparison of training the instance grasping tasks using standard QT-Opt without any additional objectives (yellow), pre-training with a goal-conditioned model (green) and 10% auxiliary objective mix-in (blue).

Experimental Results

Task	No pre-training	With pre-training
Grasp box	0%	27%
Grasp banana	4%	20%
Grasp milk	1%	20%

Table 1. Success rates of learning real world instance grasping tasks from a small amount of data with task-specific rewards: without any pre-training, pre-training with a goal-conditioned model and fine-tuning with task-specific rewards.

Discussion of Results

- Q-learning without any regularization fails to learn any of the tasks, indicating that the Q-function collapses without a presence of negative examples.

  ➡ **Both positive and negative samples are essential**

- In the context where we split the fixture placing task trajectories into two separate trajectories: the grasping part and the placing part, and shuffle the dataset to make that these trajectories are not connected in any way, disable goal chaining would make performance drops significantly.

  ➡ **With goal chaining, the model can learn to reach the goal across two separate episodes and successfully perform tasks**

- Pre-training on large and diverse datasets with actionable models can lead to representations that significantly accelerate the acquisition of downstream tasks.

Critique / Limitations / Open Issues

Specifying a task to the goal-conditioned Q-function requires suitable goal image at test time

- Goal image requirement limits the ability to specify general tasks

  e.g. Using goal images from a different scene

  e.g. Commanding the robot to grasp a particular type of object instead of reaching a goal image
- Joint training and pre-training with fine-tuning can sidestep this limitation by using additional general task rewards

Reaching some goals requires reasoning over extended horizons

- Currently, the approach cannot reposition a large number of objects in a single episode

  e.g. Moving multiple objects to desired locations

Future Work

- Representation learning and goal embeddings for general goal-conditioned policies
- Combination of planning algorithms with goal-conditioned RL

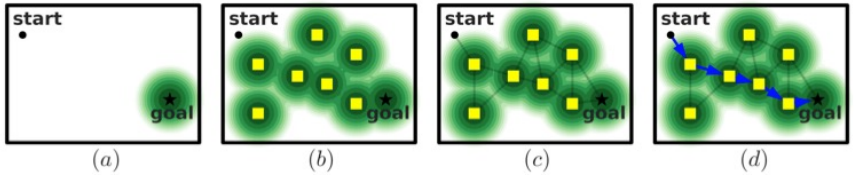


Figure 1: **Search on the Replay Buffer:** (a) Goal-conditioned RL often fails to reach distant goals, but can successfully reach the goal if starting nearby (inside the green region). (b) Our goal is to use observations in our replay buffer (yellow squares) as waypoints leading to the goal. (c) We automatically find these waypoints by using the agent's value function to predict when two states are nearby, and building the corresponding graph. (d) We run graph search to find the sequence of waypoints (blue arrows), and then use our goal-conditioned policy to reach each waypoint.

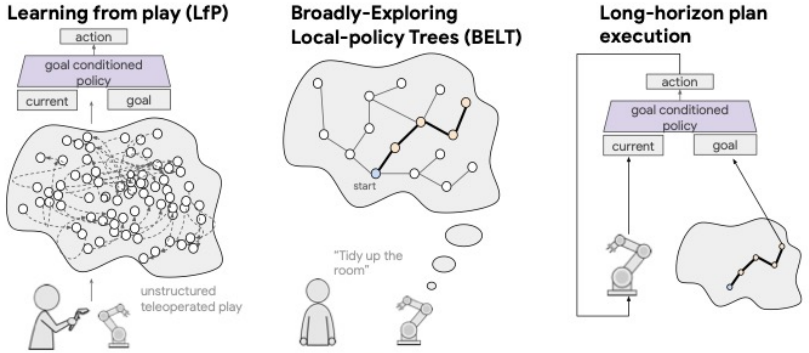


Figure 2: BELT with Play Latent Motor Plans [37] learns a general, goal-conditioned policy as well as a model from teleoperation data. Given a long-horizon task, BELT translates this into an RRT-inspired, model-based tree search through the space, where each edge represents a single task sampled from demonstration data. Trajectories are verified successful via a trajectory-wise success check on the model and executed by the policy.

Replay buffer waypoint-guided planning (Eysenbach et al., 2019)

RRT-inspired tree search planning (Ichter et al., 2020)

Extended Readings

- Kumar, Aviral, et al. "Conservative q-learning for offline reinforcement learning." arXiv preprint arXiv:2006.04779 (2020).
- Andrychowicz, Marcin, et al. "Hindsight experience replay." arXiv preprint arXiv:1707.01495 (2017).
- Ding, Yiming, et al. "Goal-conditioned imitation learning." arXiv preprint arXiv:1906.05838 (2019).
- Kalashnikov, Dmitry, et al. "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation." arXiv preprint arXiv:1806.10293 (2018).
- Eysenbach, Benjamin, Ruslan Salakhutdinov, and Sergey Levine. "Search on the replay buffer: Bridging planning and reinforcement learning." arXiv preprint arXiv:1906.05253 (2019).
- Ichter, Brian, Pierre Sermanet, and Corey Lynch. "Broadly-Exploring, Local-Policy Trees for Long-Horizon Task Planning." arXiv preprint arXiv:2010.06491 (2020).