Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills

Yevgen Chebotar [1] **Karol Hausman** [1] **Yao Lu** [1] **Ted Xiao** [1] **Dmitry Kalashnikov** [1] **Jake Varley** [1]
Alex Irpan [1] **Benjamin Eysenbach** [1 2] **Ryan Julian** [1 3] **Chelsea Finn** [1 4] **Sergey Levine** [1 5]

Abstract

We consider the problem of learning useful robotic skills from previously collected offline data without access to manually specified rewards or additional online exploration, a setting that is becoming increasingly important for scaling robot learning by reusing past robotic data. In particular, we propose the objective of learning a functional understanding of the environment by learning to reach any goal state in a given dataset. We employ goal-conditioned Q-learning with hindsight relabeling and develop several techniques that enable training in a particularly challenging offline setting. We find that our method can operate on high-dimensional camera images and learn a variety of skills on real robots that generalize to previously unseen scenes and objects. We also show that our method can learn to reach long-horizon goals across multiple episodes through goal chaining, and learn rich representations that can help with downstream tasks through pre-training or auxiliary objectives. The videos of our experiments can be found at https://actionable-models.github.io

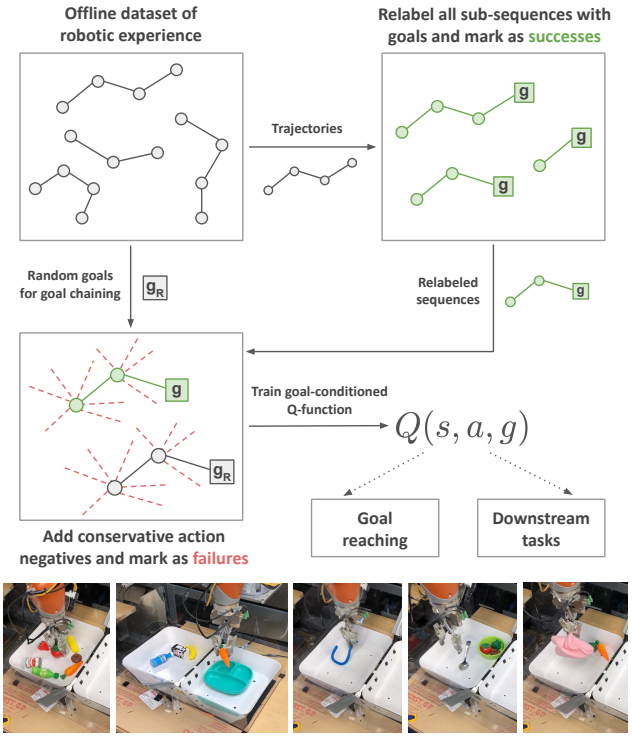


Figure 1. We propose a system for learning goal-reaching behaviors from offline datasets of real-world image-based experience. Our method enables a real-world robotic system to accomplish a wide range of visually indicated manipulation skills and learn rich representations that can help with downstream tasks.

1. Introduction

General-purpose robots will need large repertoires of skills. While reinforcement learning (RL) provides a way to learn such skills automatically, learning each skill individually can be prohibitive, particularly when task rewards must be programmed by hand and data must be collected anew for each task. Other fields of machine learning, such as natural language processing, have seen impressive progress from applying general-purpose training objectives on large and diverse datasets for either pre-training a general model that can be further fine-tuned on task-specific objectives (Devlin et al., 2018) or training a model that can be directly used for zero-shot or few-shot generalization (Brown et al., 2020). Reusing data in robotic control therefore seems like an appealing prospect, but raises a number of questions: 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?

We study these questions in the setting where the agent must learn entirely from *offline* data, without hand-specified rewards or online interaction. This is particularly compelling for a general-purpose robotic system, which might already

[1]Robotics at Google [2]Carnegie Mellon University [3]University of Southern California [4]Stanford University [5]UC Berkeley. Correspondence to: Yevgen Chebotar <chebotar@google.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

have data of past tasks that have been attempted (successfully or otherwise). Since the goal is to utilize past data from prior tasks to acquire transferable knowledge for *new* tasks, we must determine what type of knowledge such data actually provides. One answer offered in prior work is to use past experience to learn a functional understanding of the environment, as represented by a predictive model (Deisenroth & Rasmussen, 2011; Finn & Levine, 2017; Bauza & Rodriguez, 2017). In this paper, we take a different perspective: instead of learning predictive models, which must predict future world observations in all of its complexity, we will learn goal-conditioned policies via offline, model-free RL. These policies must learn to reach any possible goal state from any current state, which as we show in this work, provides an effective general-purpose training objective that synthesizes varied tasks automatically, by considering every possible state reached in the data as a potential task.

While model-free training avoids the problem of predicting images, in the offline setting it carries its own challenges, such as distributional shift (Laroche et al., 2017; Kumar et al., 2019a). Although prior works have sought to mitigate this challenge (Lange et al., 2012; Laroche et al., 2017; Kalashnikov et al., 2018; Kumar et al., 2019a; 2020), the goal-conditioned setting presents additional issues, since the offline data was not necessarily collected by a policy *attempting* to reach any single goal. In this case, we must learn entirely from "positive" examples (Xu & Denil, 2019; Zolna et al., 2020), which we show experimentally is challenging for current RL methods. We propose a strategy to provide synthetic "negative" labels that stabilize training by prescribing low Q-values for unseen actions, which, as indicated in prior work (Kumar et al., 2019a), also mitigates distributional shift in the offline setting. We further propose to extend this approach to learn how to reach goals that are *more* temporally extended than the longest trajectory in the data, "stitching" together parts of the dataset into longer skills through *goal chaining*. We call our framework *Actionable Models* to emphasize the model-level granularity of offline data reuse and the ability to directly obtain an *actionable* robotic policy.

Finally, having trained such *Actionable Models* using a self-supervised goal-conditioned RL objective, we can utilize them to solve downstream tasks in three different ways: we can directly command goals to attain zero-shot generalization to new visually indicated tasks, we can use the goal-conditioned RL objective as an auxiliary loss for joint training, and we can fine-tune an Actionable Model with a task-specific reward.

2. Related Work

Goal-conditioned RL. The aim of this work is to learn a goal-conditioned policy, a problem studied in many prior works (Kaelbling, 1993; Sutton et al., 2011; Schaul et al., 2015; Andrychowicz et al., 2017; Pong et al., 2018; Eysenbach et al., 2020; Kadian et al., 2020; Fujita et al., 2020). Like prior work, we employ hindsight relabeling, reusing each trajectory to learn how to reach many goals. Analogous to prior work (Kaelbling, 1993; Pong et al., 2018), we interpret the predictions from our Q-function as indicating the likelihood of reaching a particular goal in the future. We note that this is different from predicting an expected future state or representation (Dayan, 1993; Barreto et al., 2016; Janz et al., 2018). Importantly, our setting differs from prior work in this area because we assume an *offline* setting where the agent cannot interact with the environment to collect new data. Additionally, our system learns entirely from image observations for both the state and goal, without any manually specified similarity measure or reward function, in contrast to prior works that use distances (Pong et al., 2018) or ϵ-ball threshold functions (Andrychowicz et al., 2017).

Goal-conditioned supervised learning. Prior methods for goal-conditioned supervised learning (Oh et al., 2018; Ding et al., 2019; Ghosh et al., 2019; Sun et al., 2019; Lynch et al., 2019) have a similar motivation: learning "useful" information for control without access to rewards. While these methods employ demonstrations (Lynch et al., 2019) or online data collection (Ghosh et al., 2019), our approach requires neither, and makes use of dynamic programming (Q-learning) to "stitch" together multiple trajectories to reach goals from states without ever having seen a complete trajectory from the state to that goal (Sutton, 1988; Greydanus & Olah, 2019; Singh et al., 2020).

Techniques from RL. Our method builds upon a number of tools from the RL literature. To handle offline Q-learning, we develop a technique similar to Conservative Q-Learning (CQL) (Kumar et al., 2020), which regularizes the Q-function on out-of-distribution actions. To handle large datasets and complex real-world tasks, we build on the QT-Opt algorithm (Kalashnikov et al., 2018), using the same training system and target value computation. Similar to (Fujita et al., 2020), we extend QT-Opt to the goal-conditioned setting. We further combine it with conservative strategies to enable learning in an offline setting.

***Explicit* dynamics models.** Learning general behavior from offline datasets has been studied in the context of model-based RL, which learns explicit forward models (Deisenroth & Rasmussen, 2011; Watter et al., 2015; Williams et al., 2015; Chua et al., 2018; Buckman et al., 2018; Wang & Ba, 2019; Kaiser et al., 2019; Hafner et al., 2019; Nagabandi et al., 2020). The problem of learning dynamics models is a special case of a more general problem of auto-regressive time-series prediction (e.g., video prediction (Mathieu et al., 2015; Denton & Fergus, 2018; Kumar et al., 2019b)). Our work is related in that we can learn our model from an offline

dataset without any online interaction or reward function. These dynamics models offer different capabilities than a goal-conditioned Q-function. Whereas dynamics models answer the question "What does the future look like?", goal-conditioned Q-functions answer the opposite: "What is the probability the future looks like *this*?" For reaching a particular goal, the goal-conditioned Q-function directly provides for the optimal action. To select actions, most model-based methods require an additional cost function; our method has no such requirement.

3. Preliminaries

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, p_0, \gamma, T)$ define a Markov decision process (MDP), where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is a state-transition probability function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $p_0 : \mathcal{S} \to \mathbb{R}_+$ is an initial state distribution, γ is a discount factor, and T is the task horizon. We use $\tau = (s_0, a_0, \ldots, s_T, a_T)$ to denote a trajectory of states and actions and $R(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$ to denote the 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 $a_t \sim \pi(a_t|s_t)$. Instead of manually defining a set of reward functions, we will define one sparse reward function per goal:

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

The set $\mathcal{G}$ is a space of goals, which in our work we consider to be the space of goal images. We assume that the episode terminates upon reaching the specified goal, so the maximum total reward is 1. We use temporal-difference (TD) learning to maximize the expected return (Kaelbling, 1993; Dayan, 1993), yielding a goal-conditioned Q-function, which describes the probability of reaching the goal state g at time $t = T$:

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

We then obtain a goal-reaching policy by acting greedily with respect to the goal-conditioned Q-function: $\pi(a|s, g) = \arg\max_a Q(s, a, g)$.

4. Actionable Models

In this section, we present our method, which we call *Actionable Models*, for learning goal-conditioned skills from offline datasets. Our method is based on goal-conditioned Q-functions with hindsight relabeling, for which offline data presents a particularly challenging setting. 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, which might result in over-estimation of their Q-values (Schroecker & Isbell, 2020). As we show in our experiments, without proper regularization, standard hindsight relabeling generally fails to learn effective Q-functions in an offline setting. We present a scalable way to regularize Q-values in the offline setting by taking a *conservative* approach for unseen actions, which was recently shown to help with distributional shift in offline learning (Kumar et al., 2020; Singh et al., 2020). After introducing a framework for training goal-conditioned Q-functions offline, we further extend our method to enable *goal chaining*, a way to reach goals across multiple episodes in the dataset, which enables long-horizon goal-reaching tasks.

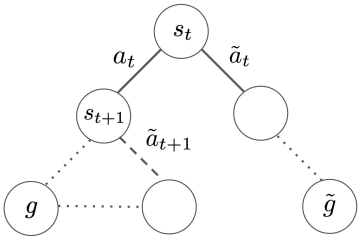


Figure 2. Two scenarios that can occur when deviating from the original action on the way to the goal g: without recovery ($\tilde{a}_t$) and with recovery ($\tilde{a}_{t+1}$).

4.1. Goal-conditioned offline Q-learning

Given a dataset $\mathcal{D}$ of previously collected trajectories $\tau \sim \mathcal{D}$, we aim to learn to reach all goals in the dataset using a goal-conditioned policy $\pi(a \mid s, g) = \arg\max_a Q(s, a, g)$. Each trajectory τ provides evidence of which states are reachable from which other states in the trajectory. More formally, each sub-sequence $\tau_{0:i} = (s_0, a_0, ..., s_i)$ with $i \in (0, ..., T)$ is a *positive* example of reaching the goal $g = s_i$. Thus, we label each such sub-sequence with the success-reward $\mathbb{1}$ given the goal $g = s_i$:

$$R(\tau_{0:i}, g = s_i) = \mathbb{1}[s_i = s_i] = 1.$$

Such relabeling alone is not enough to train a well-conditioned Q-function with respect to the action a_t. In particular, we do not know which actions result in not reaching the desired goal.

Let $\tilde{\mathcal{A}}(s, g)$ denote a set of unseen actions, for which we do not have evidence of reaching the goal g from state s in the dataset $\mathcal{D}$, and $\tilde{a} \sim p_{\tilde{\mathcal{A}}}(\tilde{a}|s, g)$ 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. Fig. 2 demonstrates two possible scenarios when deviating from original actions seen in the dataset: either the policy does not recover and lands in another state $\tilde{g} \sim p_{\tilde{\mathcal{G}}}(g)$ (e.g., $\tilde{a}_t$ in the figure), or it recovers from the deviation and reaches the desired goal g (e.g., $\tilde{a}_{t+1}$ in the figure).

We follow a conservative approach that avoids overestimation of Q-values for unseen actions. We assume there is no recovery unless there are trajectories in the dataset $\mathcal{D}$ 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 $\tilde{a} \sim p_{\tilde{\mathcal{A}}}(\tilde{a}|s, g)$ leads to some other goal $\tilde{g} \sim p_{\tilde{\mathcal{G}}}(g)$:

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

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:

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

This means, we aim at minimizing Q-values for unseen actions $\tilde{a}_t \sim p_{\tilde{\mathcal{A}}}(\tilde{a}_t|s_t, g)$, which we denote as *negative* actions as opposed to *positive* actions a_t obtained through relabeling from the dataset. We find that sampling contrastive $\tilde{a}_t$ close to the decision boundary of the Q-function – i.e., unseen actions that have high Q-values – to be the most effective implementation of this objective. Therefore, we define $p_{\tilde{\mathcal{A}}}(\tilde{a}|s, g) \sim \frac{1}{Z}\exp\left(Q(s_t, \tilde{a}_t, g)\right)$, sampling negative actions according to the soft-max distribution over their Q-values. This has an interesting connection to conservative Q-learning (CQL) (Kumar et al., 2020), which although having a different motivation of improving out-of-distribution behavior of Q-learning, arrives at a similar objective of minimizing Q-values for unseen actions. In particular, CQL shows that by sampling $\tilde{a}_t \sim \exp(Q^\pi(s_t, \tilde{a}_t, g))$, it is possible to learn a Q-function that lower bounds the true Q-function on unseen actions.

4.2. Goal chaining

Using the procedure in the previous subsection, we can train a policy to reach goals within all sub-sequences $\tau_{0:i}$ in the dataset $\tau \sim \mathcal{D}$, but the policy would only learn to reach goals reachable within a single trajectory τ in the dataset. However, parts of tasks can often be spread out across multiple trajectories, such as when accomplishing the goal of one trajectory is a pre-requisite for beginning to attempt the goal of another trajectory.

Ideally, in such situations we would like to be able to specify the final goal without manually specifying intermediate goals. The dynamic programming nature of Q-learning is able to concatenate or *chain* trajectories in state space, but further modifications are needed in order to enable chaining in *goal* space. Although similar techniques have been proposed in previous works (Andrychowicz et al., 2017; Nair et al., 2018; Lin et al., 2019) for online learning, we develop a way to perform goal chaining in an offline setting.

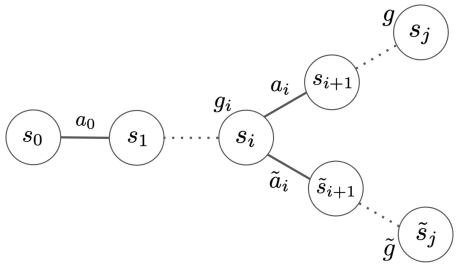


Figure 3. Illustrating goal chaining between two trajectories: $\tau_{0:i}$ and $\tau_{i:j}$ through the chaining point s_i.

To enable long-horizon tasks through goal chaining, we propose a simple modification to our goal-conditioned Q-learning method. Given a sequence $\tau_{0:i}$, instead of limiting the goal to be within the sequence (i.e., $g \sim (s_0, ..., s_i)$), we redefine it to be any state observed in the dataset (i.e., $g \sim \mathcal{D}$). If $g = s_i$ (e.g. when the goal is the final state of the sub-sequence) then similarly as before, we label such trajectories with $R(\tau_{0:i}, s_i) = 1$. Otherwise, since we do not know whether $g \neq s_i$ can be reached from the states within $\tau_{0:i}$, 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 $\tau_{0:i}$. Fig. 3 illustrates this intuition. Without loss of generality, assume that there exist a sequence $\tau_{i:j}$ that starts with (s_i, a_i) and leads to $s_j = g$. Then there will be a constant reward $R(s_j, a_j, g) = 1$, which will propagate to $Q^\pi(s_i, a_i, g)$ and eventually to $Q^\pi(s_{0:i-1}, a_{0:i-1}, g)$. In this case, we call $Q^\pi(s_i, a_i, g)$ a *chaining point* between two trajectories $\tau_{0:i}$ and $\tau_{i:j}$. It should be noted that without introducing such chaining points, e.g. through additional random goal labeling as done in this work, Q-learning alone might not be able to connect these trajectories as they will have distinct goals from within their corresponding sequences. To handle the situation when there is no pathway from s_i to g, we can still apply the conservative regularization technique described in the previous subsection. In particular, as we minimize Q-values for actions $\tilde{a}_i \sim \exp(Q^\pi(s_i, \tilde{a}_i, g))$, without an evidence for reachability of g our method will assume that the policy ends up in a different goal $\tilde{g} \sim \tilde{\mathcal{G}}$ and eventually also minimize $Q^\pi(s_{0:i-1}, a_{0:i-1}, g)$.

Interestingly, this kind of chaining works across more than two episodes. In fact, it is only limited by the long-horizon propagation of values through the Bellman recursion and by the discount factor γ, which attenuates the constant reward from the last trajectory in the chain. More generally, this procedure connects to previous works on learning goal-distance functions, such as Kaelbling (1993), where the objective is

to learn distances between all states. Furthermore, we can generalize the goal sampling procedure by defining a goal sampling function $g \sim G(\mathcal{D}, s_i, a_i)$, which can further be adjusted, e.g. by skewing the sampling distribution towards goals with high Q-values similar to (Eysenbach et al., 2020) to reduce the number of unreachable goals. It should also be noted that 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. As we show in the ablation experiments in Section 6.5, the goal chaining procedure significantly improves the capabilities of our learned policy for accomplishing longer-horizon tasks with high-dimensional image goals.

4.3. Method Summary

We now summarize our complete method, which we call *Actionable Models*. Our method takes as input a dataset of trajectories $\mathcal{D} = \{\tau\}$, 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: $\pi(a|s, g) = \arg\max_a Q(s, a, g)$. 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:

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

The first part of the loss $\mathcal{L}_g(\theta)$ increases Q-values for reachable goals and the second part regularizes the Q-function. It should be noted that when optimizing with gradient descent, the loss does not propagate through the sampling of action negatives $\tilde{a} \sim \exp(Q_\theta)$.

5. Applications of Actionable Models

In this section, we present applications of our framework, including learning general goal-reaching policies and using the goal-reaching objective to learn rich representations that can be used in downstream RL tasks, e.g. through pre-training or an auxiliary optimization objective.

5.1. Goal reaching

Our framework can be applied for learning goal reaching skills from previously collected offline datasets. Algorithm 1 outlines the example extraction and Q-target computation components. This method can be integrated into any Q-learning method with an experience replay buffer (Mnih et al., 2015).

The EXTRACTEXAMPLES module samples a trajectory from the dataset and randomly cuts it to produce a sub-sequence $\tau_{0:i}$. We relabel this sub-sequence with $R(\tau_{0:i}, s_i) = 1$ for reaching its final state as in Section 4.1, and with $R(\tau_{0:i}, g_{rand}) = Q_\theta(s_i, a_i, g_{rand})$ for some goal $g_{rand} \sim \mathcal{D}$ to enable goal chaining as in Section 4.2. All transitions from the relabeled trajectories are added to the replay buffer.

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.

In COMPUTEQTARGETS, we first sample transitions with relabeled rewards $(s_t, a_t, g, s_{t+1}, R(s_t, a_t, g))$ from the replay buffer and apply the Bellman equation to compute Q-targets for these transitions. Then, we sample a negative action $\tilde{a}_t \sim \exp(Q_\theta(s_t, \tilde{a}_t, g))$ by first uniformly sampling actions from our action space and then reweighting them by their $\exp(Q_\theta)$-values. When sampling negative actions, we filter out actions that are too close to the seen action a_t.

5.2. Pre-training

Besides using actionable models for goal reaching, we can also leverage their ability to learn functional representations of the world to accelerate acquisition of downstream tasks through conventional reward-driven online reinforcement learning. Given a large dataset $\mathcal{D}$ of previous experience, we 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. As we will show in our experiments in Section 6.6, pre-training on large and diverse datasets with actionable models leads to representations that significantly accelerate the acquisition of downstream tasks.

5.3. Auxiliary objective

In addition to pre-training, we can also 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 $\mathcal{L}_{task}(\theta)$ with the regularized goal-reaching objective $\mathcal{L}_g(\theta)$ from Eq. 1:

$$\mathcal{L}_{augmented}(\theta) = \mathcal{L}_{task}(\theta) + \xi\mathcal{L}_g(\theta).$$

To optimize this joint objective, we add three kinds of data to our replay buffer. With probability ξ, we add relabeled

trajectories using EXTRACTSAMPLES, which will be used to optimize the $\mathcal{L}_g$ term. With probability ξ, we create a relabeled trajectory but replace the action with an unseen action and replace the reward with 0 (following COMPUTEQ-TARGETS). With probability $(1 - 2\xi)$, we add the original trajectories (no relabeling) with the original rewards. When training on original trajectories (no relabeling), we condition the Q-function on a goal image set to all zeros.

6. Experiments

In our experiments, we aim to answer the following questions: 1) How does our method compare to previous methods for learning goal-conditioned policies from offline data, such as goal-conditioned behavioral cloning and standard Q-learning with hindsight relabeling? 2) Can our method learn diverse skills on real robots with high-dimensional camera images? 3) Does our proposed goal chaining technique facilitate learning to reach long-horizon goals? 4) Is goal reaching an effective pre-training step or a suitable auxiliary objective for accelerating learning of downstream reward-driven skills?

6.1. Experimental Setup

We evaluate our method on a variety of robotic manipulation scenarios, with actions corresponding to Cartesian space control of the robot's end-effector in 4D space (3D position and azimuth angle) combined with discrete actions for opening/closing the gripper and terminating the episode. In all experiments, our policy architecture follows the QT-Opt framework (Kalashnikov et al., 2018), with an additional Q-function input for the 472x472x3 goal image, which is concatenated with the current robot camera image before being fed into a convolutional tower. The full set of inputs and the network architecture can be found in Fig. 4. We also use the cross-entropy loss to fit the Q-target values, as in the original QT-Opt framework. The initial variances of the (x, y, z) end-effector positions and the azimuth angle during the CEM-optimization (Rubinstein & Kroese, 2004) are set to (3cm, 3cm, 6cm, 0.16 rad), respectively. We run CEM for 2 iterations with 64 samples per iteration and 10% elite percentile. More details on the implementation of QT-Opt can be found in prior work (Kalashnikov et al., 2018).

6.2. Training data

Acquiring data with good state coverage and visual variety is important for successful application of offline reinforcement learning. In this work, we employ data coming from RL training traces for diverse tasks and environments in both simulation and the real world, which span the gamut from random exploration to fully-trained policies. In practical applications, we might expect Actionable Models to similarly be trained on diverse and heterogeneous datasets,

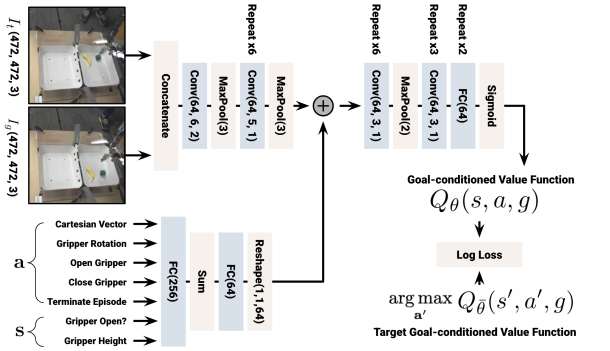


Figure 4. The Q-function neural network architecture follows the QT-Opt framework with an additional goal image input.

which might combine demonstrations or play data (Lynch et al., 2019), random exploration, and experience from other RL policies.

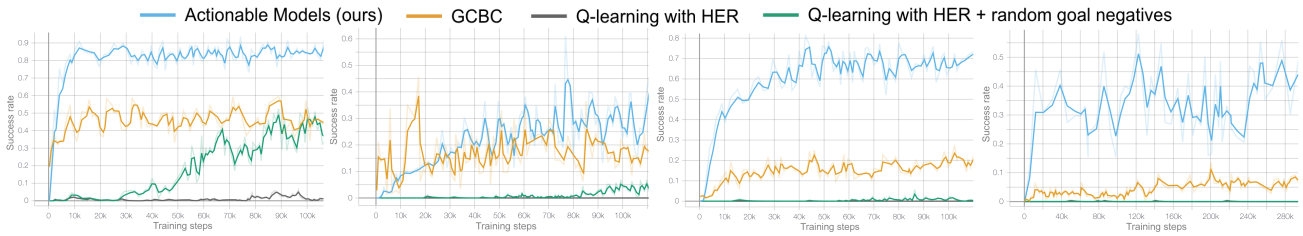
Our real robot training data comes from a wide range of reinforcement learning experiments conducted as part of another research project using the same platform (Kalashnikov et al., 2021). Crucially, none of this data was generated by goal-conditioned policies. The data was produced by policies at various stages of training, from random initializations to near-optimal behaviors, trained for 12 different tasks (with conventional reward functions) using mostly a diverse set of food objects that can be seen in Fig. 7a. The success rates in the data for the different tasks ranged from 5% to 55%, and the tasks included various versions of grasping and pick-and-place skills. The data contains over 800,000 episodes, which altogether amounts to over 6 months of continuous robotic interaction on 7 KUKA robots.

6.3. Simulated visual goal reaching experiments

In our simulated experiments, we generate a dataset of trajectories from RL runs for the four tasks depicted in Fig. 6. We train expert policies for each task with their task-specific rewards using QT-Opt until convergence (all expert policies reach more than 90% success rate at convergence). We then combine transitions from these training runs to train a single unified goal-image conditioned Q-function with our method. To the best of our knowledge, prior methods for model-free goal-conditioned RL require online exploration, whereas our problem setting requires fully offline training. In the absence of clear points of comparison, we devise a number of baselines that we believe most closely reflect widely used prior methods, which we adapt to the offline setting:

Goal-conditioned behavioral cloning (GCBC), used in several previous works to learn goal-conditioned policies (Oh et al., 2018; Ding et al., 2019; Lynch et al., 2019; Ghosh et al., 2019). The policy network follows the same architecture as in Fig. 4, except with actions as the output.

Q-learning with hindsight relabeling, analogously to hindsight experience replay (HER) (Andrychowicz et al.,



(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.

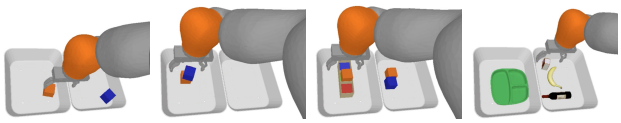


Figure 6. Simulated tasks (robot camera view): pick-and-place, stacking, fixture placing, food object grasping.

2017) and other prior work (Lin et al., 2019; Fujita et al., 2020). Here, we relabel each sub-sequence with the success-reward $R(\tau_{0:i}, s_i) = 1$ without any additional regularization. To ensure a fair comparison, the same QT-Opt RL framework is used for training the Q-function.

Q-learning with hindsight relabeling and negative random goals, where in addition to positive relabeling with $R(\tau_{0:i}, s_i) = 1$ for each sub-sequence, we introduce a reward 0 for relabeling with random goals from the data set, such that $R(\tau_{0:i}, g_{rand}) = 0$ with $g_{rand} \sim \mathcal{D}$, which should approximate minimization on $Q(s, a, \tilde{g})$. This corresponds to the standard hindsight experience replay (Andrychowicz et al., 2017) with additional random goal negatives, which was also used as a baseline by Eysenbach et al. (2020). Here we also use the same QT-Opt RL framework for training the Q-function to ensure fair comparison.

A trained model from each method is evaluated on a variety of goal images from each of the tasks in Fig. 6 over the course of training, with success rates shown in Fig. 5. Note that these online evaluations are only used for reporting the results – the trials are not available for use in training. We also observe that training a unified goal-conditioned policy is harder than training single expert policies with task-specific rewards. Although a goal-conditioned policy covers a larger space of skills, it often leads to a lower performance when evaluated on individual tasks. The results in Fig. 5 show that Actionable Models outperform GCBC on all tasks. Q-learning without any regularization fails to learn any of the tasks, indicating that the Q-function collapses without a presence of negative examples. Q-learning with random goal negatives is only able to learn the simplest of the tasks (block pick-and-place), but still underperforms compared to both Actionable Models and GCBC. With large datasets, random goal negatives can be completely unrelated to the current trajectory, and might even come from a different scene. Such random relabeled goals therefore rarely provide a strong contrastive training signal, making it difficult for the Q-function to learn to distinguish effective actions from ineffective ones. In prior work that employs online exploration (Andrychowicz et al., 2017; Lin et al., 2019), this challenge is mitigated by the fact that the online trials themselves serve as "hard negatives," since they represent failed episodes where the policy was specifically aiming to reach a particular goal. However, this is not available in the offline setting. Since Actionable Models explicitly minimize the value of unseen actions that are very close to the decision boundary, as noted in Section 4.1, they do not require careful strategies for sampling "negative" goals. Without this regularization, we see that other Q-learning methods are not able to learn any of the skills effectively.

6.4. Real-world visual goal reaching experiments

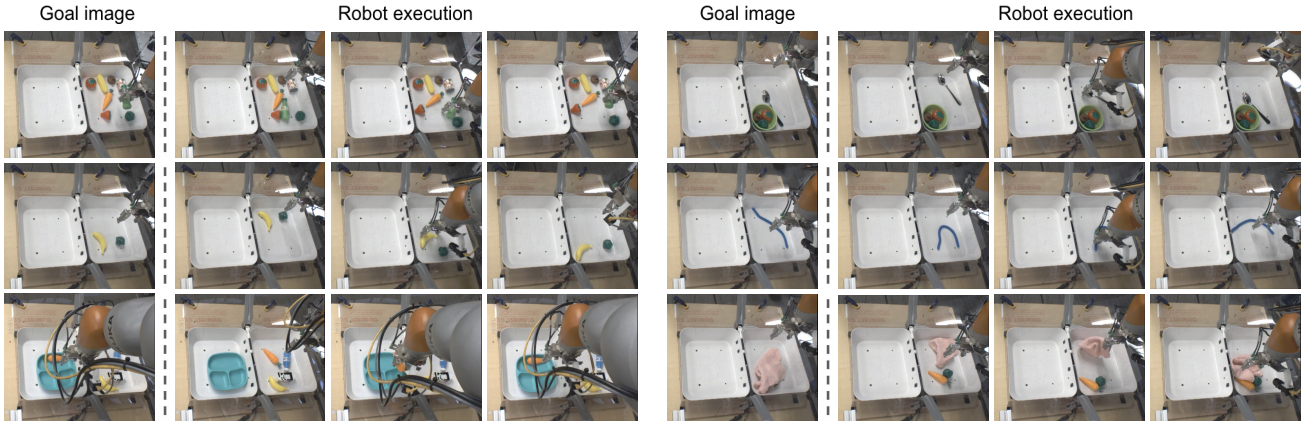
We train a single goal-image conditioned policy on all of the real-world data using the method described in Section 4.1, combined with goal chaining from Section 4.2. Fig. 7 shows examples of skills learned with our method. The qualitative results are best viewed in our supplementary video[1]. In order to perform quantitative evaluation and group various goal reaching tasks, we define the following skills:

Instance grasping. Presented with a goal image of an object in the hand, the robot has to grasp the corresponding object from the table. There are at most 8 objects randomly located on the table including the object that has to be grasped.

Rearrangement. Presented with a goal image of a particular object configuration, the robot has to move displaced objects to their goal positions. We mark episodes as successful if the robot moves the objects within 5cm of their goal positions. There are at most 3 objects located on the table and the robot has to move at most 2 objects.

Container placing. Presented with a goal image of an object placed in a container, such as a plate, the robot has to pick up the object and place it into the container. There are at most 4 objected randomly located on the table and one object has to be moved into the container.

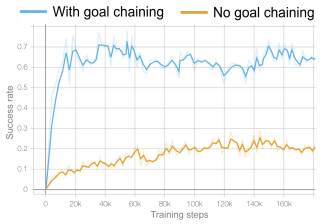
[1] https://actionable-models.github.io

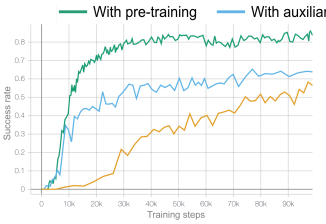


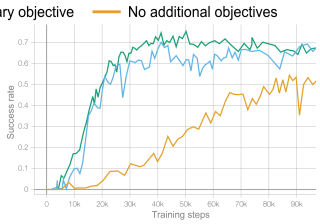
(a) Instance grasping, Rearrangement, Container placing (b) Unseen object manipulation

Figure 7. Real robot goal reaching tasks (robot camera view). Robot has to reach the object configuration specified by the goal image.

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).

To generate goal images, we first place objects in their goal configuration and record the corresponding robot camera image. After that, we reset the objects by moving them to their random initial positions. This setup ensures that the goal image is recorded in the same scene observed during the skill execution. We also include the arm in the goal image as it is usually present in the goal images when performing hindsight relabeling during training. Table 8a shows success rates for each of the skills when evaluated on 50 trials for each skill with the best results on instance grasping and the most challenging task being container placing. In addition, we experiment with our model manipulating unseen objects. Fig. 7b shows examples of our learned policy manipulating objects that were not present in the training data, such as silverware, a towel, and a rubber cord. This demonstrates that our policy is able to learn general object manipulation skills that can be transferred to novel objects.

We compare our method to standard Q-learning with hindsight relabeling and no regularization as described in Section 6.3. This baseline is not able to learn any of the skills and achieves 0% success rate as the Q-function collapses without seeing any negative action examples during training. The breadth of this evaluation provides coverage of a wide range of real world scenes, but unfortunately makes it difficult to compare to the full range of prior methods. We therefore defer a full comparison to prior works to the simulated experiments in Section 6.3.

6.5. Goal chaining

To evaluate the goal chaining method described in Section 4.2, we first ablate the goal chaining component in simulation. We split the fixture placing task trajectories (Fig. 6) in our dataset into two separate trajectories: the grasping part and the placing part. We shuffle the dataset, meaning that these trajectories are not connected in any way. We then train the goal-conditioned Q-function with and without goal chaining, and evaluate by conditioning on the final goal of the fixture placing task, where the object is fully placed in the fixture. The results are shown in Fig. 8b. When we disable goal chaining, performance drops significantly, as the Q-function is only trained on goals within single episodes, which does not include performing the fixture placing task all the way from the beginning of grasping to the end. With goal chaining, the model can learn to reach the placing goal across two episodes and successfully perform the task.

The above setting is in some sense "ideal" for goal chaining. To evaluate goal chaining under more realistic conditions, we also perform an ablation study with real robots

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.

by training our goal-conditioned model on the data from Section 6.4, but with goal chaining disabled. We observe a performance drop from 66% down to 8% success rate on the longer-horizon container placing tasks, which are more likely to be spread out across multiple episodes. This indicates that by enabling goal chaining we can better utilize the dataset, especially for longer-horizon tasks that might require combining several episodes.

6.6. Representation learning

Lastly, we analyze how Actionable Models can help with pre-training rich representations or provide an auxiliary objective for standard RL, as described in Sections 5.2 and 5.3. First, we perform simulated experiments on two instance grasping versions of the food object grasping task (Fig. 6 on the right), which are harder for the standard QT-Opt policy to learn. The goal is to grasp a specific object, such as a bottle or a banana, which requires identifying the object and then singulating it from clutter before grasping and lifting. For pre-training, we use the goal-conditioned model trained on 4 tasks, as in Section 6.3, which we then fine-tune with additional online data collection and a standard task reward. During fine-tuning, the goal-image input is set to all zeros. To add the auxiliary goal reaching objective, we mix in relabeled episodes with a probability of 10% during the online training of the task as described in Section 5.3. Figures 8c and 8d compare learning with pre-training and an auxiliary objective to standard online RL with QT-Opt without any additional objectives. The results show that both pre-training and the auxiliary objective significantly speed up the training of the downstream tasks, suggesting that functional knowledge acquired by Actionable Models is beneficial for learning other robotic skills.

In our real world experiments, we first pre-train a goal-conditioned model on the data from Section 6.4 and then fine-tune it on a small amount of data ($<$10K episodes) for three instance grasping tasks with task-specific rewards, as shown in Table 1. Without any pre-training, the standard offline QT-Opt is not able to learn these tasks and achieves less than 5% success, while fine-tuning a pre-trained model enables reaching at least 20% success rate on all tasks. This confirms that representations learned by pre-training with a goal-reaching objective also improve learning performance in the real world.

7. Discussion and Future Work

In this work, we showed that it is possible to learn diverse skills on real robots with high-dimensional goal images in a completely offline setting. We developed a method that enables doing this with goal-conditioned Q-functions, which are particularly challenging to train in an offline setting. We presented a regularization method that ensures that the Q-values of unseen actions are not overestimated, and a goal chaining technique that enables reaching goals across multiple episodes. We also showed that downstream RL tasks can be learned more efficiently by either fine-tuning pre-trained goal-conditioned models, or through a goal-reaching auxiliary objective during training.

While our approach enables learning diverse goal-reaching tasks and accelerates acquisition of downstream behaviors, it does have a number of limitations. First, specifying a task to the goal-conditioned Q-function requires providing a suitable goal image at test-time, which should be consistent with the current scene. This limits the ability to specify general tasks, e.g. using goal images from a different scene or 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, as discussed in Section 6.6. An exciting direction for future work is to study how representation learning and goal embeddings can further alleviate this limitation for general goal-conditioned policies. Second, reaching some goals requires reasoning over extended horizons (e.g., moving multiple objects to desired locations). Currently, our approach cannot reposition a large number of objects in a single episode. Combining planning algorithms with goal-conditioned RL, as in some recent works (Eysenbach et al., 2019; Ichter et al., 2020), could be an exciting direction for addressing this limitation.

Going forward, we believe that our work suggests a practical and broadly applicable technique for generalizable robotic learning: by training goal-conditioned Q-functions from diverse multi-task offline data, robots can acquire a broad knowledge about the world. This knowledge, in turn, can then be used to directly perform goal-reaching tasks or as a way to accelerate training downstream policies with standard online reinforcement learning methods.

Acknowledgements

We would like to thank Benjamin Swanson, Josh Weaver, Noah Brown, Khem Holden, Linda Luu and Brandon Kinman for their robot operation support, Tom Small for help with videos and project media, Julian Ibarz, Kanishka Rao, and Vincent Vanhoucke for their managerial support, and all of the Robotics at Google team for their support throughout this project.

References

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In *NeurIPS*, pp. 5048–5058, 2017.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Van Hasselt, H., and Silver, D. Successor features for transfer in reinforcement learning. *arXiv*, abs/1606.05312, 2016.

Bauza, M. and Rodriguez, A. A probabilistic data-driven model for planar pushing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3008–3015. IEEE, 2017.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *arXiv*, abs/2005.14165, 2020.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *arXiv*, abs/1807.01675, 2018.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv*, abs/1805.12114, 2018.

Dayan, P. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pp. 465–472, 2011.

Denton, E. and Fergus, R. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pp. 1174–1183. PMLR, 2018.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv*, abs/1810.04805, 2018.

Ding, Y., Florensa, C., Phielipp, M., and Abbeel, P. Goal-conditioned imitation learning. *arXiv*, abs/1906.05838, 2019.

Eysenbach, B., Salakhutdinov, R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv*, abs/1906.05253, 2019.

Eysenbach, B., Geng, X., Levine, S., and Salakhutdinov, R. R. Rewriting history with inverse RL: hindsight inference for policy improvement. In *NeurIPS*, 2020.

Finn, C. and Levine, S. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Fujita, Y., Uenishi, K., Ummadisingu, A., Nagarajan, P., Masuda, S., and Castro, M. Y. Distributed reinforcement learning of targeted grasping with active vision for mobile manipulators. *arXiv*, abs/2007.08082, 2020.

Ghosh, D., Gupta, A., Fu, J., Reddy, A., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals without reinforcement learning. *arXiv*, abs/1912.06088, 2019.

Greydanus, S. and Olah, C. The paths perspective on value learning. *Distill*, 2019. doi: 10.23915/distill.00020. https://distill.pub/2019/paths-perspective-on-value-learning.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019.

Ichter, B., Sermanet, P., and Lynch, C. Broadly-exploring, local-policy trees for long-horizon task planning. *arXiv*, abs/2010.06491, 2020.

Janz, D., Hron, J., Mazur, P., Hofmann, K., Hernández-Lobato, J. M., and Tschiatschek, S. Successor uncertainties: exploration and uncertainty in temporal difference learning. *arXiv*, abs/1810.06530, 2018.

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., and Batra, D. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020.

Kaelbling, L. P. Learning to achieve goals. In Bajcsy, R. (ed.), *IJCAI*, pp. 1094–1099. Morgan Kaufmann, 1993. ISBN 1-55860-300-X.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. *arXiv*, abs/1903.00374, 2019.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv*, abs/1806.10293, 2018.

Kalashnikov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., and Hausman, K. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv*, abs/2104.08212, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*, pp. 11761–11771, 2019a.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv*, abs/2006.04779, 2020.

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., and Kingma, D. Videoflow: A flow-based generative model for video. *arXiv*, abs/1903.01434, 2019b.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Laroche, R., Trichelair, P., and Combes, R. T. d. Safe policy improvement with baseline bootstrapping. *arXiv*, abs/1712.06924, 2017.

Lin, X., Baweja, H. S., and Held, D. Reinforcement learning without ground-truth state. *arXiv*, abs/1905.07866, 2019.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *CoRL*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 2019.

Mathieu, M., Couprie, C., and Lecun, Y. Deep multi-scale video prediction beyond mean square error. *arXiv*, abs/1511.05440, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015. ISSN 14764687.

Nagabandi, A., Konolige, K., Levine, S., and Kumar, V. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pp. 1101–1112. PMLR, 2020.

Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *NeurIPS*, pp. 9209–9220, 2018.

Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

Pong, V., Gu, S., Dalal, M., and Levine, S. Temporal difference models: Model-free deep rl for model-based control. *arXiv*, abs/1802.09081, 2018.

Rubinstein, R. Y. and Kroese, D. P. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2004. ISBN 038721240X.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In Bach, F. and Blei, D. (eds.), *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR.

Schroecker, Y. and Isbell, C. Universal value density estimation for imitation learning and goal-conditioned reinforcement learning. *arXiv*, abs/2002.06473, 2020.

Singh, A., Yu, A., Yang, J., Zhang, J., Kumar, A., and Levine, S. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv*, abs/2010.14500, 2020.

Sun, H., Li, Z., Liu, X., Lin, D., and Zhou, B. Policy continuation with hindsight inverse dynamics. *arXiv*, abs/1910.14055, 2019.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, 2011.

Wang, T. and Ba, J. Exploring model-based planning with policy networks. *arXiv*, abs/1906.08649, 2019.

Watter, M., Springenberg, J. T., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv*, abs/1506.07365, 2015.

Williams, G., Aldrich, A., and Theodorou, E. Model predictive path integral control using covariance variable importance sampling. *arXiv*, abs/1509.01149, 2015.

Xu, D. and Denil, M. Positive-unlabeled reward learning. *arXiv*, abs/1911.00459, 2019.

Zolna, K., Novikov, A., Konyushkova, K., Gulcehre, C., Wang, Z., Aytar, Y., Denil, M., de Freitas, N., and Reed, S. Offline learning from demonstrations and unlabeled experience. *arXiv*, abs/2011.13885, 2020.