Efficient Self-Supervised Data Collection for Offline Robot Learning

Shadi Endrawis[12], Gal Leibovich[2], Guy Jacob[2], Gal Novik[2], Aviv Tamar[1]
[1] Technion – Israel Institute of Technology
[2] Intel Labs

***Abstract*— A practical approach to robot reinforcement learning is to first collect a large batch of real or simulated robot interaction data, using some data collection policy, and then learn from this data to perform various tasks, using offline learning algorithms. Previous work focused on manually designing the data collection policy, and on tasks where suitable policies can easily be designed, such as random picking policies for collecting data about object grasping. For more complex tasks, however, it may be difficult to find a data collection policy that explores the environment effectively, and produces data that is diverse enough for the downstream task. In this work, we propose that data collection policies should actively explore the environment to collect diverse data. In particular, we develop a simple-yet-effective goal-conditioned reinforcement-learning method that actively focuses data collection on novel observations, thereby collecting a diverse data-set. We evaluate our method on simulated robot manipulation tasks with visual inputs and show that the improved diversity of active data collection leads to significant improvements in the downstream learning tasks.**

I. INTRODUCTION

Reinforcement learning (RL) is a popular approach for learning robotics skills [1], [2], [3]. In general, RL can be performed in an online manner, where the robot simultaneously interacts with the environment and improves its policy, or offline, where first, some data about the robot-environment interaction is collected, and then, a learning algorithm calculates the desired policy using this data.

The offline learning approach offers many practical benefits: data can be collected in a safe and controlled manner, and collection can be parallelized across different robots/environments [4], [5]. However, the quality of offline learning clearly depends on the quality of the data, which in turn depends on the data collection policy. Indeed, recent studies focused on tasks where a policy that produces high-quality and diverse data can manually be designed, such as randomly picking objects for learning to grasp [3], randomly poking objects for learning to push [6], [7], [8], or randomly throwing an object for learning to grasp and throw [9].

For learning more complex tasks, however, we posit that manually designing data collection policies can become very difficult. As an example, consider a robot that must learn to stack two rigid blocks. If data is collected by a policy that randomly picks and places a block, the chances of observing task-relevant data is small. In this work, we investigate a principled approach for designing data collection policies that actively explore their environment, with the hypothesis that active exploration can lead to more relevant data.

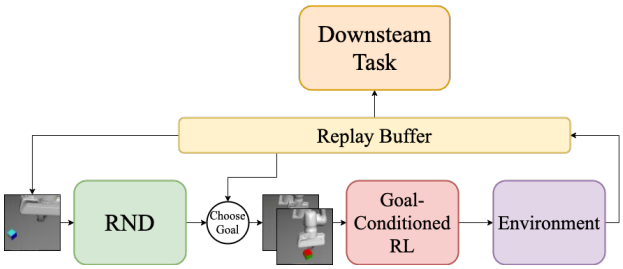


Fig. 1: Goal-Conditioned Exploration for Data Collection: We show that we can improve the learning of downstream tasks by using higher quality data that we produce using an exploration scheme that combines intrinsic motivation and goal-conditioned policies.

We aim for a general method that can produce high-quality data-sets that would broadly improve the performance of downstream robot learning tasks. Our specific desiderata are:

1) Data that exhibits a diversified set of behaviours to allow for building better models and policies.
2) Data that is evenly distributed, and not highly concentrated on a small part of the state space.
3) Efficient data collection, without spending valuable robot time on non-interesting parts of the state space.

We approach the problem by defining the data collection phase as a reinforcement learning (RL) problem, where the goal of the agent is to cover as much of the state space as possible. Such objectives are common in RL exploration strategies based on intrinsic motivation (IM) [10], [11], [12], [13], [14], [15]. We propose an improvement of the random network distillation (RND) method [15] that actively drives the robot to explore novel states, thereby efficiently collecting diverse data. Our key idea is that we can use goal-conditioned policies to quickly reach states that are novel, without waiting for the robot to reach these states through the slow reward maximization of standard RL.

We focus on learning from high dimensional image inputs – a challenging and important robot learning setting. Our empirical results demonstrate significantly better exploration than commonly used RL exploration strategies. More importantly, our data-set collection method significantly improves learning performance in various downstream tasks, such as supervised learning and offline RL.

II. BACKGROUND

We recapitulate several RL ideas we build on in our work:

a) Reinforcement Learning: In Reinforcement Learning, at every time step t an agent is in a state $s_t \in \mathcal{S}$. The agent executes an action $a_t \in \mathcal{A}$, transitioning into a new state following the transition probability $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$, and receiving a reward $r_t = R(s_t, a_t)$. This is often formalized as a Markov Decision Process (MDP). The goal is to find a policy $a_t = \pi(s_t)$ that maximizes expected cumulative reward, $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^t r_t\right]$, where $\gamma \in (0, 1]$ is the discount factor. The value of a state-action pair when following a policy π, $Q^\pi(s, a)$, is defined as the expected sum of discounted rewards from that state-action pair:

$$Q^\pi(s, a) = \mathop{\mathbb{E}}_{a\sim\pi}\left[\sum_{k=t+1}^{\infty} \gamma^{k-t-1} r_k \Big| s_t = s, a_t = a\right].$$

Actor-critic algorithms aim to find the optimal policy $\pi^*(s_t)$ by using two functions approximators, one for the policy $\pi_\theta(s_t)$ (actor) and the other for state-action value function $Q^\pi_\phi(s_t, a_t)$ (critic). Specifically, we use *Twin Delayed Deep Deterministic Policy Gradient* (TD3) [16], which learns $Q^\pi_\phi(s_t, a_t)$ by minimizing the mean squared Bellman error of transition sampled from a replay buffer $\mathcal{D}$:

$$\mathcal{L}(\phi) = \mathop{\mathbb{E}}_{(s_t, a_t, r_t, s_{t+1})\sim\mathcal{D}}\left[(r_t + \gamma Q_\phi(s_{t+1}, \pi_\theta(s_{t+1})) - Q_\phi(s_t, a_t))^2\right],$$

and learns a policy $\pi_\theta(s_t)$ that maximizes $Q_\phi(s_t, a_t)$ using a loss function $\mathcal{L}(\theta) = -\mathop{\mathbb{E}}_{s_t\sim\mathcal{D}}\left[Q_\phi(s_t, \pi_\theta(s_t))\right]$. TD3 additionally applies stability improvements such as a target network, twin critics, delayed updates, and noise regularization. We use TD3 for data collection, by training policies that maximize intrinsic motivation (Sec. II-A) or goal-conditioned policies (Sec. II-B).

A. Intrinsic Motivation

Intrinsic motivation (IM) methods add a reward signal to an RL agent that incentivizes exploratory behavior. Curiosity, one of the most widely used IM methods, adds reward for visiting novel states. For high dimensional problems, the 'novelty' of the state can be hard to define, and several ideas have been explored in the literature [10], [13], [17]; here we focus on random network distillation (RND) [15], [14]. The RND reward bonus is the error in predicting the features of the current state, where the features are defined by a randomly initialized 'target' neural network that is not changed throughout learning. A 'predictor' network with the same architecture as the target network is trained to predict the target output on data collected by the agent. Formally, the target network maps a state to an embedding $f : \mathcal{S} \to \mathbb{R}^k$ and the predictor neural network $\hat{f} : \mathcal{S} \to \mathbb{R}^k$ is trained by gradient descent to minimize the expected MSE loss $\|\hat{f}(s_t|\theta_{\hat{f}}) - f(s_t)\|^2$ with respect to its parameters $\theta_{\hat{f}}$. The prediction error is expected to be higher for novel states than for states similar to the ones the predictor has trained on.

B. Goal-Conditioned Learning

To handle sparse reward RL problems, several recent works proposed learning goal-conditioned polices. The idea is that learning to reach any state in the domain will give a denser reward signal, which can be easier to learn from. Universal Value Function Approximators (UVFA) [18] simply augment the state input to the policy and value function, $s \in \mathcal{S}$, with an additional goal-state input $g \in \mathcal{G}$. Hindsight experience replay (HER) [19] is an off-policy algorithm for training goal-conditioned policies. After collecting a trajectory $s_0, s_1, ..., s_T$, we store in the replay buffer every transition $s_t \to s_{t+1}$ not only with the original goal used for this episode, but also with a goal randomly taken from the trajectory $g' \in s_0, s_1, ..., s_T$. The main idea is that even in trajectories where we failed to achieve the goal we sought to reach, we still have successfully collected data for training the UVFA to reach g'. Notice that the goal being pursued influences the agent's actions but not the environment dynamics and therefore we can replay each trajectory with an arbitrary goal, assuming that we use an off-policy RL algorithm.

III. METHOD

In this section, we describe our RL-based method for automatic data collection. As discussed in Sec. I, we desire data that effectively covers the state space, and IM-based methods are therefore suitable for driving the agent towards parts of the state space that have not yet been visited. However, by simply adding a reward bonus to the RL algorithm, such methods explore the state space inefficiently: typical RL algorithms adapt slowly to changes in the reward function, and the agent will spend a long time visiting already known parts of the state space until its policy changes to visit the novel parts. For online RL, this issue manifests as a slow learning process. In offline RL, however, where the data set size is limited in advance, the problem is more severe: inefficient learning will cover less relevant states, and will therefore hinder performance in downstream tasks.

We propose an algorithm for an agent to autonomously explore its domain and reach novel states quickly. Our main idea is to actively drive the agent towards novel states by training a goal-conditioned policy, and set the goals to be states which are deemed novel according to the current RND criterion. Thus, when a new part of the state space is identified as novel, the agent will not spend time on learning to reach it through maximizing the reward function, but will directly navigate towards it using the goal-conditioned policy.

Our method is specified in Algorithm 1. To act in the environment, at the beginning of every episode the agent picks a goal state with maximal novelty from the replay buffer (line 6), and then uses the goal-conditioned policy to reach that goal. In practice, it is usually infeasible to calculate novelty for the entire replay buffer $\mathcal{R}$ every episode, so in lines 5-6 the maximum is taken only over a subset $\mathcal{R}'$.

To train the goal-conditioned policy using an off-policy algorithm, we follow an approach similar to HER, which samples a goal from the future of the state in the episode and assigns -1 reward to any transition that does not reach the goal. This self-supervised reward definition causes the agent to learn to reach its goal with as few steps as possible. Under such definition, the goal-conditioned value function

Algorithm 1 Data-set Collection with Goal-Conditioned Policy and RND

1: Initialize RND networks $f, \hat{f}$
2: Initialize off-policy algorithm $\mathcal{A}$ with replay buffer $\mathcal{R}$
3: **for** training cycle $= 1, \ldots, N$ **do**
4: **for** episode $= 1, \ldots, M$ **do**
5: Sample $\mathcal{R}'$ from $\mathcal{R}$ ▷ $|\mathcal{R}'| = M \cdot T$
6: $g \leftarrow \text{argmax}_{\mathcal{R}'} \|\hat{f}(s_{k+1}) - f(s_{k+1})\|^2$
7: **for** $t = 0, \ldots, T$ **do**
8: $a_t \leftarrow \pi(s_t, g)$
9: Execute the action a_t and observe new state s_{t+1}
10: Store transition (s_t, a_t, s_{t+1}) in $\mathcal{R}$
11: **end for**
12: **for** $0, \ldots, T$ **do**
13: Sample mini-batch B from $\mathcal{R}$
14: **for** $(s_\tau, a_\tau, s_{\tau+1}) \in B$ **do**
15: Sample goal g_τ from $s_\tau, ..., s_T$
16: $r_\tau \leftarrow -(g_\tau \neq s_\tau)$
17: $d_\tau \leftarrow (g_\tau = s_\tau) \vee (g_\tau = s_{\tau+1})$ ▷ done flag
18: Store $((s_\tau, g_\tau), a_\tau, r_\tau, (s_{\tau+1}, g_\tau), d_\tau)$ in B'
19: **end for**
20: Perform one step of off-policy RL using $\mathcal{A}$ and B'
21: **end for**
22: **end for**
23: Train $\hat{f}$ on the last M episodes in $\mathcal{R}$ for K epochs
24: **end for**
25: **return** $\mathcal{R}$

of the off-policy algorithm is effectively an inverse distance function between two state observations.

However, dissimilar to HER, which samples a goal for each state and stores the state-goal pair statically in the replay buffer (i.e., for each state the goal is chosen once when the state is being inserted into the replay buffer, corresponding to setting the goal before line 10 in Algorithm 1 instead of in line 15), we sample goals dynamically each time a batch of states is sampled to train the network, which means that each state can be paired with many different goals throughout the training. We found that this allows for greater sample efficiency and better use of the data.

Additionally, with a small probability, for state s_τ we sample the goal to be the state itself, and in that case the reward is 0 and the done signal is $True$ (lines 16 and 17 in Algorithm 1), meaning that the value of the state-goal pair should be 0. We found that this helps stabilize training. We hypothesize that since the value function represents an inverse distance function, if the network never encounters identical or very close state-goal pairs during the training, the predictions near the goal would be very noisy.

In order to not be biased towards states that remain in the buffer for a long time, RND is trained only on states from recent experience, similar to the online RND algorithm [15].

IV. RELATED WORK

One approach to curiosity-driven exploration is known as Intrinsically Motivated Goal Exploration Processes (IMGEPs) [20]. IMGEPs equip the agent with a goal space, where each point is a vector of (target) features of behavioural outcomes. During exploration, the agent samples goals in this goal space by maximizing empirical competence progress using multi-armed bandits, enabling a learning curriculum where goals are progressively explored from simple to more complex. One limitation of IMGEPs is the need to manually engineer goal space representations, which is difficult for high-dimensional observations such as images.

Closely related to our approach is Go-Explore [21]. In Go-Explore, the agent is returned to promising states from its previous experience to explore further. The implementation of this idea in [21] has some drawbacks. The simulator's internal state is reset in order to return to promising states, which is not feasible for real robots. Our approach, on the other hand, trains a goal-conditioned policy to reach desired states. Moreover, novelty is measured by counting state visitation, which requires compact state representations that are heavily dependant on prior knowledge of the domain.

In order to alleviate some of the shortcomings of Go-Explore and intrinsic motivation methods, Guo and Brunskill [22] use HER alongside a one-step prediction model to calculate novelty. While their method addresses the non-static nature of the intrinsic reward and does not require a reset like Go-Explore, it does not take full advantage of the data throughout the training due to the fact that both the goal relabeling and novelty assignment are done statically: the goal and novelty value for each state is picked only once when they are inserted into the replay buffer. Moreover, one-step prediction models can be difficult to train on high-dimensional state spaces and are susceptible to random noise from the environment. Indeed, [22] did not show results with image inputs, and to the best of our knowledge, our work demonstrates the first application of goal-conditioned policies and intrinsic motivation with images.

The idea of quickly reaching novel states for efficient exploration was shown to be effective in the tabular case with algorithms such as Explicit Explore or Exploit (E^3) [23]. E^3 divides the state space into two sets, known and unknown, based on how thoroughly a given state has been visited. When exploring, a policy for reaching unknown states quickly is used. In the tabular case, a simple visitation count can be used to measure state novelty; our work extends this idea to high-dimensional state spaces by using a generalized novelty measure and a goal-conditioned policy.

V. RESULTS

In this section we evaluate our algorithm on a robotic manipulation environment using various downstream tasks. We show that using our data collection method we can significantly improve learning performance in downstream tasks, as compared to other collection strategies. We investigate the following questions:

1) Can we improve performance in downstream learning tasks by collecting more diversified data?
2) Can self-supervised data collection be useful for learning multiple downstream tasks using the same data?
3) Does our goal-conditioned algorithm collect better data than vanilla RL exploration?

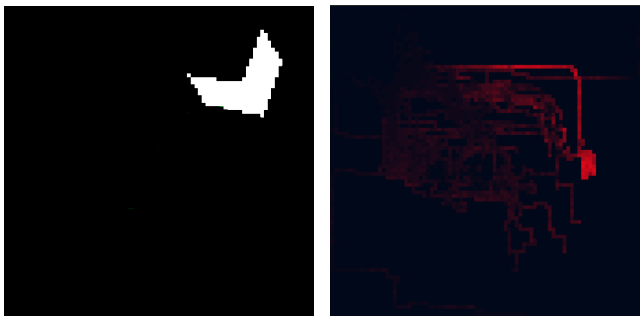


Fig. 2: Illustrative experiment. On the left is the image state that the agent observes of the object moving against a black background. On the right is a visualization of the RND novelty values in states previously visited, the brighter the red the more novel the state is. The goal-conditioned agent will pick goals from the brightest region on the right side of the image.

A. *Illustrative domain*

We begin with a toy domain that will allow us to illustrate the benefits of our goal-conditioned exploration approach. The domain is motivated by a robotic problem of manipulating a rigid object using raw vision input. The observation is a white object against a black background, the object can move up, down, left and right by one pixel or rotate clockwise and counter clock-wise with a resolution of 30 degrees. The size of the image is 84×84. The length of each episode is 250 and the object is returned to the center at the beginning of each episode. An example can be seen in Figure 2.

Our goal in this experiment is to test which approach covers as much of the state space as possible. To do so, we count the number of discrete states the agent visits during the training, where the state is the position and rotation of the object, resulting in a state space of size $84 \times 84 \times 12$.

In addition to the baselines described above, we also consider an ablation where we change the goal picking strategy when acting (line 6 in Algorithm 1) to either picking a random state from the replay buffer or a state with minimum novelty. Figure 3 shows that RND novelty leads to better coverage of the state space compared to random exploration. Moreover, our algorithm exploits novelty information more efficiently than vanilla RL with intrinsic motivation. As expected, the goal-conditioned agent based on minimum novelty goals performs worst. Interestingly, the agent that picks random goals performs worse than the agent that behaves completely randomly, indicating that the goal picking strategy is an important component of the algorithm. We can also see that the random exploration reaches a plateau and doesn't keep exploring further, which means we won't achieve better exploration by simply collecting more samples with random actions.

Additionally, we plot the average number of unique states visited within every episode, and the average number of unique and novel states visited every episode, i.e., the number of states that the agent visited for the first time over the entire training so far. Note that the goal-conditioned agent visits less unique states every episode than the random

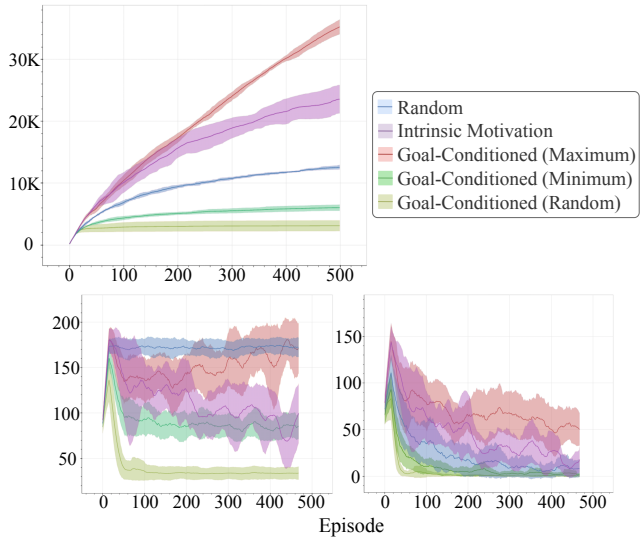


Fig. 3: State visitation counts for the illustrative domain. On top we show the total number of unique states the agent visited. On the bottom left we show the number of unique states the agent visited during each episode. Lastly, on the bottom right we show the number of unique states the agent visited during each episode which have also never been visited in previous episodes.

agent, but significantly more novel states. This is since the goal-conditioned approach leads the agent to discover *novel regions*, all throughout the training process.

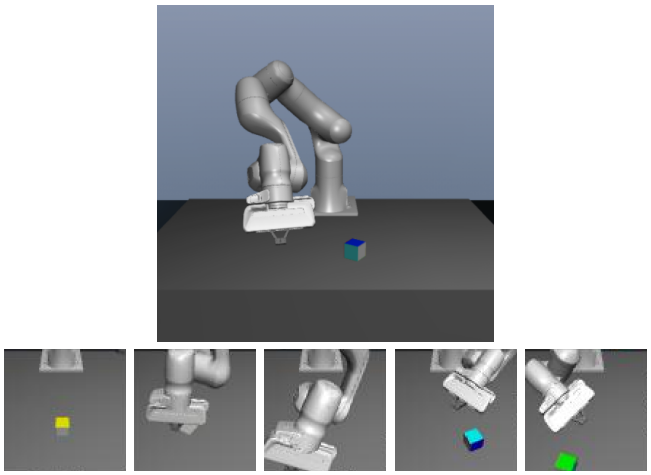
B. *Simulated Robotic Manipulation Domains*

We next describe results on simulated robotic manipulation domains. We begin by describing our simulation setting and baseline comparisons, and then present our results.

Simulation environment: We simulate a 7DoF Franka Panda robotic arm with a closed gripper and cartesian position control of the end-effector. The robot is positioned on a table, and a cube object with colored sides is placed in front of it. The robot can move freely in space, and can move and flip the cube on the table into different configurations. The camera observations are retrieved from a fixed camera positioned in front of the robot looking downwards at the table. An example of the setup can be seen in Figure 4. For studying how the robot's presence in the image affects our exploration algorithms, we also consider an ablation where the robot is not rendered in the camera image. For the simulation we used Robosuite [24].

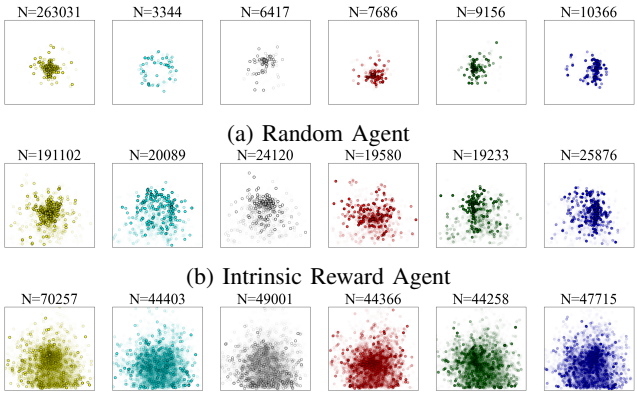
Baselines: We compare our data collection method against two baselines: the first is a fixed random policy which samples actions uniformly from the action space of the robot. The second is a vanilla RND approach: we trained a TD3 agent that maximizes intrinsic reward calculated as novelty by RND. We emphasize that the same RND architecture and training schedule was used for the baseline and our method, with the only difference being how the RND novelty is used by the RL algorithm.

Data collection and organization: For data collection, we used the TD3 implementation in RL Coach [25]. An



Low Novelty — High Novelty

Fig. 4: Simulation environment. Top: a simulated Franka Panda setup on a table and a rigid cube object. Bottom: examples of the observations that the RL agent and RND network get as input. The bottom row is ordered from left to right by the novelty of that state as given by the RND at the end of training.



(a) Random Agent

(b) Intrinsic Reward Agent

(c) Goal-Conditioned Agent

Fig. 5: Visualization of the data-sets collected using each method. The dots represent a position of the cube on the table as seen in the data-set, and the color corresponds to the color of the face at the top. The number at the top signifies that number of dots a plot contains for a certain color. Note that our goal-conditioned approach produces data that is significantly more diverse and evenly distributed than the baselines.

episode in our domain is set to 200 transitions. For each method, we collect 5 separate data-sets, where each data-set contains $M \cdot T = 300K$ transitions (1500 episodes), and the RND network and TD3 agent are reset between collecting the data-sets. After each episode, the cube is reset to a fixed initial position with the yellow face up.

Network architecture: In this work we wish to disentangle the questions of training algorithm and architecture from exploration. Therefore, we chose popular neural network architectures, and used the same architectures for all methods we evaluated. Specifically, we used convolutional network architectures similar to [26] for the agent and [15] for the RND network. For the goal-conditioned agent, we condition the network on a goal image by concatenating it with the state image in the channel dimension. More details on specific parameters can be found in the appendix.

1) Data-Set Collection Analysis: As mentioned above, one of our objectives is to collect data that is diversified and evenly distributed. In Figure 5 we visualize the collected data-sets by plotting the x, y position of the cube on the table for every state in the data. We group the plots by the color of the upper face of the cube (recall that the cube is initialized with the yellow face up). The goal-conditioned exploration covers a significantly larger part of cube positions on the table, and does so more uniformly compared to baselines. Additionally, the distribution of top face color of the cube is more evenly distributed among the different colors, resulting in a less imbalanced data distribution, which is known to greatly effect learning in downstream tasks, as we demonstrate in our subsequent experiments.

It is important to note that even though the robot covers a significantly larger region of the input image than the cube (see Figure 4), our method was able to identify novelty in the cube position and orientation.

2) Supervised Learning Tasks: A common downstream task is supervised learning – learn to predict some property of the scene such as the pose of an object. To evaluate the effect of the collected data on learning quality, we train a NN on two different prediction tasks, for which training labels can easily be extracted from the simulator:

- Classification: predict the upper face color of the cube.
- Regression: predict the position of the cube on the table.

Since it's difficult to produce test data that is independent of all three methods, in order to keep our testing criteria as fair as possible, we produce a train/test split by randomly sampling 5% of the examples from each data-set of each method to create a single test set for all the methods. The remaining 95% samples constitute the training set. This procedure results in 5 training sets for each method and a single test set for all the methods.

The results are presented in Figure 6. For both classification and regression, the goal-conditioned exploration data-sets produced a higher median accuracy and lower MSE with little variance. Also note that the ablation of removing the robot from the image had a positive yet minor effect, showing the robustness of our approach.

3) Offline Reinforcement Learning Tasks: We next evaluate our data collection method for training offline (a.k.a. Batch) RL agents. Specifically, we evaluate the performance of training a policy using Batch Constrained Q-Learning (BCQ[1]) [27], with the data-sets collected using different collection policies.

We consider three different tasks of varying difficulty:

- Flip the cube to have a specific color facing up. This task is specified by a binary reward of 1 when the

[1]The BCQ implementation we used can be found here.

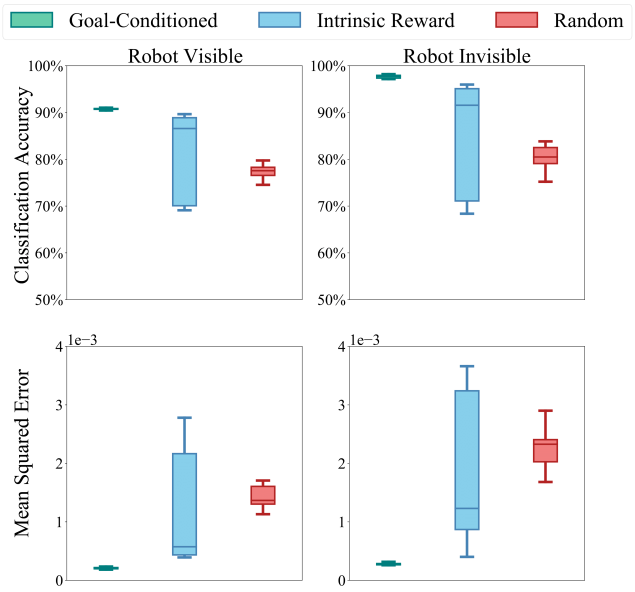


Fig. 6: Supervised learning experiments. The upper row shows the classification accuracy of predicting the color of the upper face of the cube. The lower row shows the mean squared error of predicting the x, y, z position of the cube on the table.

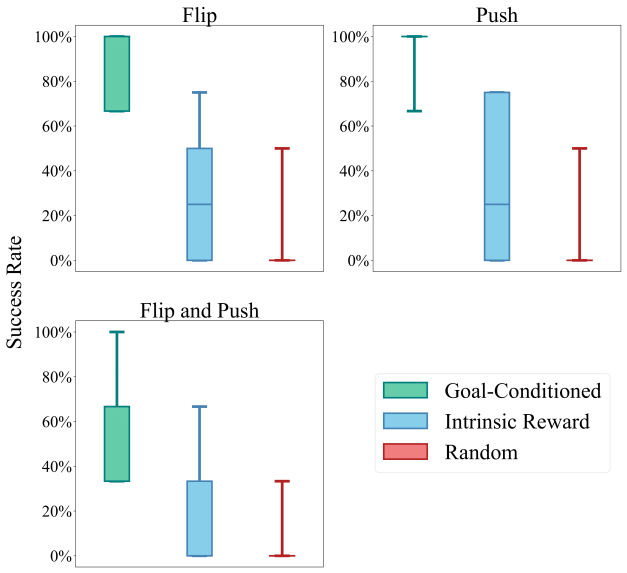


Fig. 7: Offline learning experiments for the different tasks of pushing, flipping and their combination. The numbers indicate the percentage of the times the learned policy was successful at performing the task. For the pushing tasks, success indicates that the cube is in the target region.

correct color is on the upper face and 0 otherwise.

- Push the cube to the rightmost region of the table. The reward is proportional to horizontal distance between the goal region and the cube, specifically, $r = 1 - tanh(\rho(cube_y - \mathcal{T}_y))$, where ρ is a scaling constant, and $\mathcal{T}_y$ is the threshold in the y direction of the region we want to push the cube into.
- Push the cube to the rightmost region of the table and flip it to a specific color. The reward is the sum of the two previous reward functions.

We emphasize that the rewards are not related to the behaviour of the robot during the data collection. In reality, defining rewards can be difficult. To focus our evaluation on data quality, however, we chose tasks for which rewards are easily constructed using data extracted from the simulator.

The results of the offline learning can be seen in Figure 7. Similarly to the supervised learning experiments, the goal-conditioned exploration data-sets produced a higher success rate with lower variance. We also performed an ablation of hiding the robot in the images, which gave similar results; we omit it due to space constraints.

VI. CONCLUSIONS

We proposed a simple-yet-effective method for generating diverse data-sets for offline robot learning. Our approach combines intrinsic motivation with goal-conditioned RL, to produce an agent that can quickly cover novel states.

In simulated robotic manipulation domains, our method produced significantly more diverse data-sets than common baseline methods, and led to improved performance on various downstream tasks. In future work we will evaluate our approach on real robot data collection, and on combinations of simulated and real data using sim-to-real methods.

APPENDIX

For all the algorithms we use the default hyper-parameters, with exception to those detailed in Table I. Additional details and implementation can be found here.

TABLE I: Parameter Changes

Symbol	Usage	Value
N	training cycles	150
M	# episodes each training cycle	10
	probability of sampling self goal	0.04
K	RND training epochs	4
	TD3 actor architecture	DQN + Dense[300, 200]
	TD3 critic architecture	DQN + Dense[400, 300]
	TD3 learning rate	0.0001
	TD3 policy additive noise	linear schedule 1.5→0.5
	supervised learning architecture	DQN
	supervised learning batch size	128
	supervised learning # epochs	25
	supervised learning optimizer	ADAM(0.0001)
	supervised learning # seeds	5
	BCQ # training steps	50K
	BCQ # seeds	3
rho	BCQ # push reward scaling	10.0
$\mathcal{T}_y$	BCQ # push reward threshold	18cm

ACKNOWLEDGMENT

Aviv Tamar is partly funded by the Israel Science Foundation (ISF-759/19) and a grant from Intel Corporation.

REFERENCES

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," The Journal of Machine Learning Research, vol. 17, no. 1, pp. 1334–1373, 2016.

[3] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," CoRR, vol. abs/1806.10293, 2018. [Online]. Available: http://arxiv.org/abs/1806.10293

[4] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in Reinforcement learning. Springer, 2012, pp. 45–73.

[5] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," arXiv preprint arXiv:2005.01643, 2020.

[6] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in Advances in neural information processing systems, 2016, pp. 5074–5082.

[7] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar, "Learning robotic manipulation through visual planning and acting," arXiv preprint arXiv:1905.04411, 2019.

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

[9] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," IEEE Transactions on Robotics, 2020.

[10] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation." in NIPS, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 1471–1479. [Online]. Available: http://dblp.uni-trier.de/db/conf/nips/nips2016.html#BellemareSOSSM16

[11] P.-Y. Oudeyer and F. Kaplan, "What is intrinsic motivation? a typology of computational approaches," Frontiers in Neurorobotics, vol. 1, p. 6, 2009. [Online]. Available: https://www.frontiersin.org/article/10.3389/neuro.12.006.2007

[12] A. G. Barto, "Intrinsic motivation and reinforcement learning," in Intrinsically Motivated Learning in Natural and Artificial Systems, G. Baldassarre and M. Mirolli, Eds. Springer, 2013, pp. 17–47. [Online]. Available: https://doi.org/10.1007/978-3-642-32375-1_2

[13] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, 2017, pp. 2778–2787. [Online]. Available: http://proceedings.mlr.press/v70/pathak17a.html

[14] Y. Burda, H. Edwards, D. Pathak, A. J. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," CoRR, vol. abs/1808.04355, 2018. [Online]. Available: http://arxiv.org/abs/1808.04355

[15] Y. Burda, H. Edwards, A. J. Storkey, and O. Klimov, "Exploration by random network distillation," CoRR, vol. abs/1810.12894, 2018. [Online]. Available: http://arxiv.org/abs/1810.12894

[16] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," CoRR, vol. abs/1802.09477, 2018. [Online]. Available: http://arxiv.org/abs/1802.09477

[17] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-based exploration with neural density models," CoRR, vol. abs/1703.01310, 2017. [Online]. Available: http://arxiv.org/abs/1703.01310

[18] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, 2015, pp. 1312–1320. [Online]. Available: http://jmlr.org/proceedings/papers/v37/schaul15.html

[19] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, 2017, pp. 5055–5065. [Online]. Available: http://papers.nips.cc/paper/7090-hindsight-experience-replay

[20] S. Forestier, Y. Mollard, and P. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," CoRR, vol. abs/1708.02190, 2017. [Online]. Available: http://arxiv.org/abs/1708.02190

[21] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "Go-explore: a new approach for hard-exploration problems," CoRR, vol. abs/1901.10995, 2019. [Online]. Available: http://arxiv.org/abs/1901.10995

[22] Z. D. Guo and E. Brunskill, "Directed exploration for reinforcement learning," CoRR, vol. abs/1906.07805, 2019. [Online]. Available: http://arxiv.org/abs/1906.07805

[23] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," Machine learning, vol. 49, no. 2, pp. 209–232, 2002.

[24] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," in arXiv preprint arXiv:2009.12293, 2020.

[25] I. Caspi, G. Leibovich, G. Novik, and S. Endrawis, "Reinforcement learning coach," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1134899

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[27] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," CoRR, vol. abs/1812.02900, 2018. [Online]. Available: http://arxiv.org/abs/1812.02900