Batch Exploration with Examples for Scalable Robotic Reinforcement Learning Supplementary Material

Anonymous Author(s)
Affiliation
Address
email

1 Related Work

Learning from diverse offline datasets has shown promise as a technique for learning robot policies that can generalize to unseen tasks, objects, and domains [1, 2, 3, 4, 5, 6, 7, 8]. However collecting such large and diverse datasets in robotics remains an open, and challenging problem.

A vast number of prior works have collected datasets for robotic learning under a range of problem settings and supervision schemes. One class of approaches uses humans in the loop and collects datasets of task demonstrations via teleoperation [9, 7], kinesthetic teaching [10, 11], or scripted policies [7]. While these methods can produce useful data, they are difficult to perform at scale, across diverse tasks and environments. Alternatively, many other works have explored collecting large robotic datasets without humans in the loop for tasks like object re-positioning [1, 4, 6], pushing [12, 13] and grasping [2, 14, 15]. While these present a scalable approach to data collection, the unsupervised nature of the exploration policy results in only a small portion of the data containing meaningful interactions.

One way to keep the scalability of random exploration, but acquire more relevant interaction, is to have an agent learn to explore under an intrinsic reward signal, which is task-agnostic but encourages more meaningful interaction. These intrinsic rewards come in many forms, including approaches which optimize for visiting novel states [16, 17, 18, 19, 20], the learning progress of the agent [21, 22], model uncertainty [23, 24, 25, 26, 27], information gain [24], auxiliary tasks [28], generating and reaching goals [29, 30], and state distribution matching [31]. Additionally, a number of these approaches [28, 29, 30, 32] have been demonstrated on real robotics problems. However all of these methods struggle with the issue of having to explore *everything* about a potentially vast state space when only some portion of it is relevant. We attempt to mitigate this challenge by introducing some mild supervision into the exploration problem.

A seemingly obvious approach to incorporating supervision into the exploration problem is to include a task-specific extrinsic reward function which is then combined with the exploration objective. In fact most applications of intrinsic motivation in RL do exactly this, and treat the intrinsic reward as an additional reward bonus. Other works also leverage more complex approaches to combining value functions and exploration [33, 34, 35]. Unlike these works, we focus on the setting which does not rely on supervision in the loop of RL, as is needed when providing a reward function online. Like this work, some prior works have explored how out of the loop weak supervision can be leveraged to acquire better exploratory behavior. These works have explored using supervision ranging from demonstrations [36], binary labels about state factors of variation [37], and semantic object labels [38] to accelerate exploration. Unlike these approaches, our proposed supervision can be collected in a matter of minutes and leads to efficient exploration in real visual scenes of robot manipulation.

Our method draws inspiration from prior work on reward learning [39, 40] and adversarial imitation learning [41]. These approaches aim to tackle the *task-specification problem*, and learn a discriminator over human provided goal state images or demonstrations, which is used to acquire a reward function.

In contrast, our work focuses on how to incorporate scalable sources of supervision into robotic exploration and data collection. We show that an ensemble of such classifiers can be used to guide exploration, and this data can easily be used with any offline reinforcement learning algorithm. By considering the two stage batch exploration + batch reinforcement learning approach, our work depends far less on the accuracy of the specific classifiers used during data collection, and can potentially learn multiple downstream tasks from a single dataset.

2 Architecture Details

In this section, we go over implementation details for our method as well as our comparisons.

During data collection, for each domain (block, door, and drawer domains in simulation as well as the real robot domain), all comparisons are trained on an Nvidia 2080 RTX, and all input observations are [64, 64, 3]. Each domain leverages an identical architecture, which is described as follows.

All comparisons use an encoder f_{enc} with convolutional layers (channels, kernel size, stride): [(32, 4, 2), (32, 3,1), (64, 4, 2), (64,3,1), (128, 4, 2), (128, 3,1), (256, 4, 2), (256, 3,1)] followed by fully connected layers of size [512, 2 $\times L$] where L is the size of the latent space (mean and variance). We use a latent space size of 256. All layers except the final are followed by ReLU activation.

The decoder f_{dec} takes a sample from the latent space of size L and feeds it through fully connected layers [128, 128, 128], followed by de-convolutional layers (channels, kernel size, stride): [(128, 5, 2), (64, 5, 2), (32, 6, 2), (3, 6,2)]. All layers are followed by ReLU activation except the final layer, which is followed by a Sigmoid.

The dynamics model f_{dyn} is an LSTM layer [128] followed by a fully connected network with layers [128, 128, 128, L], which are all followed by ReLU activation except the final layer. For all domains, BEE and SMM learn just one dynamics models while disagreement learns five of these.

For BEE, we learn an ensemble of three relevance discriminators. These take a sample from the latent space of size L and feed it through fully connected layers [128, 64, 64, 1], which are all followed by ReLU activations except the final layer, which is followed by a Sigmoid.

For SMM, we learn two separate VAEs: one to represent the density over the policy's visited states while the other fits a density model to the human provided relevant states. These two VAEs have the same architecture: they both use an encoder g_{enc} that takes in a sample from the latent space of size L and feeds it through fully connected layers [150, 150], which are followed by ReLU activations. This is followed by a fully connected layer [L_2] for the mean and variance each, where L_2 is the size of the latent space. We use $L_2 = 100$. The decoder g_{dec} takes in a sample from the latent space of size L_2 and feeds it through fully connected layers [150, 150, L], where all layers except the last are followed by a ReLU activation.

3 Training Details

Acquiring human supervision: For each comparison in each simulated domain, we supply 100 examples of relevant images. For the block domain, these examples involve the gripper hovering over the target block at a random z position in a region of +/- 0.02 in the x and y directions from the initial block position. For the door domain, the example images involve the gripper next to the door handle with the door set to random angles either between -45 and -5 degrees or between 5 and 45 degrees. For the drawer domain, the example images involve the gripper near the drawer handle, with the drawer open to random amounts between 0 and 0.14. For the robot domain, the example images involve the small corner drawer of the desk opened and the robot arm moved to the handle. For each comparisons in the robot domain, we only supply 50 examples of relevant images.

Planning during online data collection (MPC): We collect a dataset of 2,000 episodes, each of 50 time steps. During online planning, all methods use the cross entropy method (with only one iteration) to plan a sequence of actions. For each 50-step episode, we replan every 10 steps, i.e. we plan five 10-step trajectories. At each stage of planning, we sample 1000 10-step action sequences and sort according to the method used. The agent uniformly randomly chooses one of the top 5 ranked trajectories to execute. With probability 0.1, the agent takes a random action in place of a chosen one from the selected trajectory. On the real robot we use the same process, but collect 1000 episodes of 100 time steps each.

Model training: All models for BEE, disagreement, and SMM are trained with a learning rate of 1e-3. The main VAE (f_{enc}, f_{dec}) for all methods uses a beta of 1e-3, and the separate VAEs for

SMM use beta 0.5, which was the default value used in the codebase of the original paper. After each new episode is collected, it is added as a sample of size [50, 3, 64, 64] into the replay buffer. The encoder/decoder f_{enc} and f_{dec} as well as the dynamics model (all 5 in the case of disagreement) are updated 20 times after each new episode. For SMM, both VAEs are also updated 20 times after each epoch. All models are trained using separate Adam optimizers and using random batches of size 32 length H samples starting from any time step of the most recent 500 episodes, where H is the current training horizon for the dynamics model(s).

For training the dynamics model(s), for the first 50 episodes, we use a training horizon of 2; for the next 100 episodes, we use horizon 4; for the 150 episodes after that, we use horizon 8; and for all remaining episodes we use horizon 10. For all comparisons, the encoder/decoder f_{enc} and f_{dec} are updated for each of the 20 times with 1 batch from observations in the replay buffer that were collected online as well as 1 batch from the provided example images. Cropping regularization is applied to these input batches by expanding the boundaries by 4 pixels each and then choosing a random 64×64 crop of this larger image. For all simulated experiments, balanced batches of both human-provided example images and observations from the replay buffer are used to train the main VAE. Hence, all methods in simulation leverage the same human-provided weak supervision. In the robot domain (for all methods), the VAE is not trained with balanced batches of the human provided images, as there are only a small number (50) such states.

For BEE, the relevance discriminators are each updated once at the end of the episode. To prevent overfitting, the discriminators are trained with mixup and input image cropping. For mixup regularization, hyperparameter $\alpha = 1$ is used to control the extent of mixup.

4 Experimental Details

For the block, door, and drawer domains, we use a Mujoco simulation built off the Meta-World environments [42]. For the robot domain, we consider a real Franka robot operating over a desk, which has two drawers as well as a cabinet and multiple objects on top.

Interaction with Target: For the block and door evaluation of the online data collection, interaction is defined as moving the target block or door at least a distance of 0.05 any time during the episode. For the drawer domain, interaction is defined as pulling the drawer open by at least 0.03 any time during the episode. The drawer begins slightly open (by 0.05 distance). Lastly, for the real robot domain, we define two criteria for interaction: (1) touching the handle of the desk's corner drawer and (2) actually moving the drawer open or closed. We do not reset the drawer position between episodes, so if an episode ends with the drawer open, the next episode will start with it open.

Downstream Planning: For all control experiments, evaluation is done by using model predictive control with SV2P models trained on the full dataset collected in the batch exploration phase for 100k iterations. We plan 10 actions and execute them in the environment five times for a 50 step trial. Each stage of planning uses the cross entropy method, sampling 200 10-step action sequences, sorting them by the mean pixel distance between the goal and the predicted last state of each trajectory, refitting to the top 40, and selecting the lowest cost trajectory.

SV2P Training: SV2P learns an action-conditioned video prediction model by sampling a latent variable and subsequently generating an image prediction with that sample. The architecture and losses used here are identical to the original SV2P paper [43]. This architecture is shown in Figure 1, which is taken from the original paper. The models are trained to predict the next fifteen frames given an input of five frames. All other hyperparameters used for training are default values used in the codebase of the original paper.

Downstream Task Evaluation: In the Open Drawer task, the goal image involves the gripper above the drawer handle, which is open to 0.15 distance. Success is defined by opening the drawer at least 0.03.

In the Blue Block task, the goal image involves the gripper over the initial blue block position and the blue block moved 0.1 to the right. Success is defined as pushing the block more than 0.05 to the right.

In the Green Block task, the goal image involves the gripper over the initial green block position and the green block moved 0.1 to the left and 0.04 downwards. Success is defined as pushing the block more than 0.05 to the left.

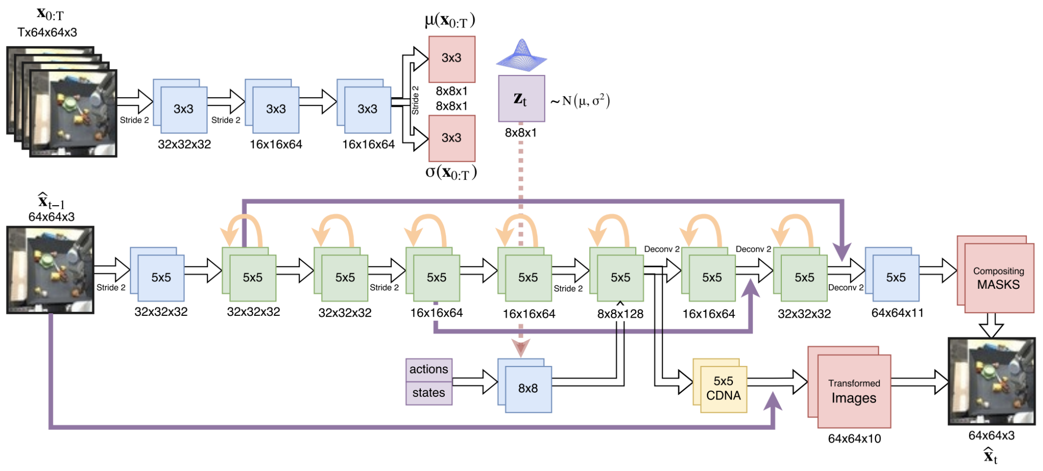

Figure 1: **SV2P architecture.** SV2P estimates the posterior latent distribution $p(z \mid x_{0:T})$ by learning an inference network (top) $q_\phi(z \mid x_{0:T}) = \mathcal{N}(\mu(x_{0:T}), \sigma(x_{0:T}))$. Latent values are sampled from $q_\phi(z \mid x_{0:T})$, and the generative network (bottom) takes in the previous frames, latent values, and actions to predict the next frames. Figure taken from the original paper [43].

In the Door task with five distractors, the goal image involves the gripper above the handle and the door opened to 0.35 radians. Success is defined by opening the door to at least 0.15 radians, measured at the end of each 50-step episode.

In the real robot door closing task, we do not use a goal image, but rather train a single reward classifier for closing the drawer on a few hundred labeled images. This same reward classifier is used with both methods dynamics models. The drawer starts halfway open and is considered success if at the end of the episode the drawer is fully closed at the end of the 100 timestep episode.

References

[1] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[2] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[3] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. 2018.

[4] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[5] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In *Advances in Neural Information Processing Systems*, pages 9094–9104, 2018.

[6] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[7] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv*, pages arXiv–1909, 2019.

[8] Ajay Mandlekar, Fabio Ramos, Byron Boots, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *arXiv preprint arXiv:1911.05321*, 2019.

[9] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[10] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *arXiv preprint arXiv:1810.07121*, 2018.

[11] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019.

[12] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 30–37. IEEE, 2016.

[13] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in neural information processing systems*, pages 5074–5082, 2016.

[14] Yevgen Chebotar, Karol Hausman, Zhe Su, Artem Molchanov, Oliver Kroemer, Gaurav S. Sukhatme, and Stefan Schaal. Bigs: Biotac grasp stability dataset. 2016.

[15] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[16] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016.

[17] Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2577–2587, 2017.

[18] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[19] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[20] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[21] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 206–214. Curran Associates, Inc., 2012.

[22] Pierre-Yves Oudeyer. Computational theories of curiosity-driven learning. *arXiv preprint arXiv:1802.10546*, 2018.

[23] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[24] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[25] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[26] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*, 2019.

[27] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *ICML*, 2020.

[28] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

[29] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[30] Xi Chen, Yuan Gao, Ali Ghadirzadeh, Marten Bjorkman, Ginevra Castellano, and Patric Jensfelt. Skew-explore: Learn faster in continuous spaces with sparse rewards, 2020.

[31] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[32] Archit Sharma, Michael Ahn, Sergey Levine, Vikash Kumar, Karol Hausman, and Shixiang Gu. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. *arXiv preprint arXiv:2004.12974*, 2020.

[33] Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

[34] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *arXiv preprint arXiv:2006.09641*, 2020.

[35] Riley Simmons-Edler, Ben Eisner, Daniel Yang, Anthony Bisulco, Eric Mitchell, Sebastian Seung, and Daniel Lee. {QX}plore: Q-learning exploration by maximizing temporal difference error, 2020.

[36] Léonard Hussenot, Robert Dadashi, Matthieu Geist, and Olivier Pietquin. Show me the way: Intrinsic motivation from demonstrations. *arXiv preprint arXiv:2006.12917*, 2020.

[37] Lisa Lee, Benjamin Eysenbach, Ruslan Salakhutdinov, Chelsea Finn, et al. Weakly-supervised reinforcement learning for controllable behavior. *arXiv preprint arXiv:2004.02860*, 2020.

[38] Devendra Singh Chaplot, Helen Jiang, Saurabh Gupta, and Abhinav Gupta. Semantic curiosity for active visual learning. *arXiv preprint arXiv:2006.09367*, 2020.

[39] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems*, pages 8538–8547, 2018.

[40] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[41] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[42] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100, 2020.

[43] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.