Batch Exploration with Examples for Scalable Robotic Reinforcement Learning

Annie S. Chen*, **HyunJi Nam***, **Suraj Nair***, **Chelsea Finn**
Stanford University
{asc8, hjnam, surajn}@stanford.edu
* equal contribution

1 Introduction

Learning from large and diverse datasets is a paradigm that has seen remarkable success in several domains, from computer vision [1] to natural language [2, 3], with favorable properties like broad generalization. While recent work [4, 5] has aimed to build such datasets for robotic manipulation, acquiring large amounts of meaningful and useful robotic interaction remains a significant challenge. On one hand, using humans to explicitly collect meaningful interaction, for example through teleoperation, is difficult to do at scale. On the other hand, while random exploration techniques can be run at a much larger scale [4], they collect lower quality interaction with the environment due to the lack of human supervision.

To address the above challenge, a number of prior works have explored the problem of task-agnostic exploration [6, 7, 8, 9, 10, 11, 12, 13]. In this setting, agents still explore the world in an unsupervised and scalable manner, but leverage some form of unsupervised intrinsic reward to encourage meaningful interaction. While these approaches have been successful in video games and simulated robotic control domains, they can struggle with the requirement of exploring *everything* in real, high-dimensional scenes, such as those in vision-based robotic manipulation problems.

Our key insight is that by leveraging some *weak human supervision*, we can allow the agent to focus on semantically relevant parts of the state space, greatly accelerating the collection of useful data. Specifically, a human can communicate a prior over relevant states by providing a handful of examples of "interesting" or "meaningful" states ahead of time, which a learning based agent can then use to guide their exploration. Another advantage of such an exploration approach is that it seamlessly integrates with recent progress in the fields of offline or "batch" reinforcement learning [14], where an agent learns from an offline dataset of interaction. Together they provide a scalable approach to robot learning, where first in the batch exploration phase, weak human supervision is used to collect a large and diverse dataset of meaningful interaction, and second in the batch reinforcement learning phase, policies can be learned from this data and used for downstream task execution.

Concretely, our main contribution is a batch exploration framework, **B**atch **E**xploration with **E**xamples (BEE), which leverages weak supervision to efficiently explore and can enable scalable collection of robotic datasets. BEE starts with a handful of examples of relevant states provided by a human, which only takes a few minutes to collect. BEE then learns to estimate whether a state is relevant or not, and explores around states which it estimates are relevant. We observe that BEE is able to explore effectively in challenging high-dimensional robotic environments, and unlike standard task agnostic exploration techniques, is able to guide its exploration towards relevant states. Across a range of simulated and real robot vision based manipulation tasks, BEE interacts more than twice as often with relevant objects than prior state-of-the-art unsupervised and weakly supervised exploration methods, and as a result collects higher quality data, enabling better downstream task performance.

2 The Batch Exploration + Batch RL Framework

We begin by describing the batch exploration + batch reinforcement learning framework, with the goal of a *scalable*, data-driven approach to robotic learning, illustrated in Figure 1. **First**, in the

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2020 (NeurIPS 2020), Vancouver, Canada.

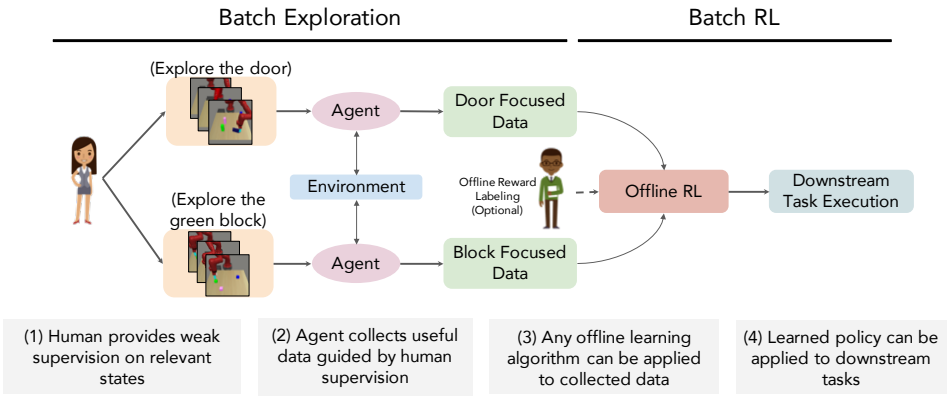

Figure 1: **The Batch Exploration + Batch RL Framework**. First, a human provides weak supervision to the agent to indicate what regions of the state space it should explore. Second, the agent explores around these regions, collecting useful data. Third, offline RL is run on the collected data (either goal-directed or optionally using offline reward labeling). Lastly, the learned policy is applied to downstream tasks. By minimizing the need for a human in the loop, this framework enables scalably collecting and learning from robotic data.

batch exploration phase, a human provides some weak supervision to the agent which indicates what regions of the state space it should explore. **Second**, also in the batch exploration phase, the agent uses the supervision to guide its exploration to collect and store the relevant data *without needing a human in the loop*. **Third**, in the batch reinforcement learning phase, the collected datasets can be used with any model-based or model-free offline RL algorithm to learn a policy or model. This offline RL phase either can use self-supervised RL techniques (e.g. goal-conditioned model-free RL [15] or visual foresight [16, 17]) or can label the offline dataset with rewards and then use standard batch RL algorithms [14, 18, 19, 20, 21, 22]. **Lastly**, the policy can then be applied to any downstream tasks for which the initial guidance was relevant.

Batch Exploration. During the batch exploration phase, let the agent be exploring in a fixed horizon controlled Markov process (CMP) $\mathcal{M}$ defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, \mu, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p(s_{t+1}|s_t, a_t)$ represents the stochastic environment dynamics, $\mu(s_0)$ represents the initial state distribution, and T denotes the episode horizon. Additionally, let $\mathcal{S}^* \subset \mathcal{S}$ represent a subset of the state space which is relevant to explore. In the first stage of batch exploration, a human provides some context information $\mathcal{C}$ to guide the agent towards exploring the relevant states. While this context information can take many forms, in this work, we consider the case where it is a set of K states $[\bar{s}_1, ..., \bar{s}_K]$ sampled uniformly from $\mathcal{S}^*$.

In the second stage of batch exploration, the agent aims to learn an exploration policy $a_t \sim \pi_{exp}(\cdot \mid s_t, \mathcal{C})$ conditioned on the human provided context to maximize an exploratory reward $\mathcal{R}_{exp}(s_t, \mathcal{C})$, which gives high reward for visiting states in $\mathcal{S}^*$, that is find π_{exp} which maximizes the expected exploratory reward. The exact reward function implementation will depend on the form of the context $\mathcal{C}$, which we discuss further in Section 3.

Batch Reinforcement Learning. Given the dataset $\mathcal{D}$, collected by the agent, a number of downstream offline RL approaches can be applied, ranging from learning a goal-conditioned policy on the collected data as is, or labeling the collected dataset $\mathcal{D}$ with a reward function offline and running standard offline RL. As described in Section 4 we use a model based planning approach with either goal images or an offline labeled reward function.

3 Batch Exploration with Examples (BEE)

While a number of works have studied the problem of batch RL [14, 18, 19, 20, 21, 22], fewer works have studied the batch exploration phase of our framework, which we focus on here. Our proposed approach, batch exploration with examples (BEE) receives weak supervision via examples of relevant states, and subsequently aims to collect data around such relevant states. This latter step is difficult, as it requires the algorithm to determine (a) whether a state (in our case an image) is relevant to explore, and (b) how to select actions to reach these states. In this section we describe how the supervision is provided, how BEE determines whether states are relevant, and how we can efficiently learn to reach these relevant states via planning with a learned dynamics model.

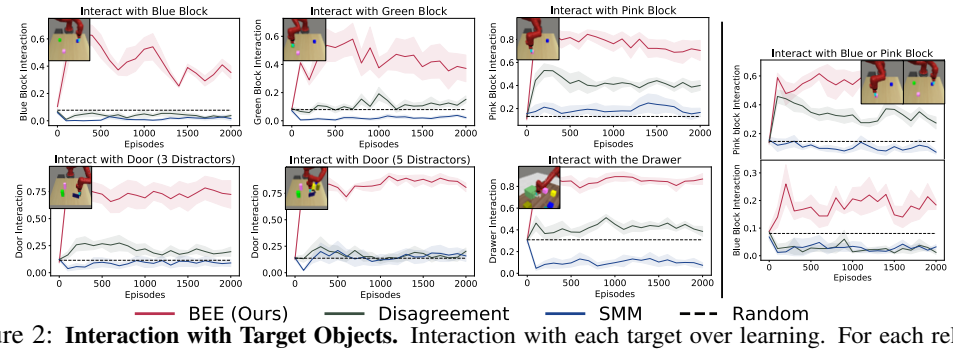

Figure 2: **Interaction with Target Objects.** Interaction with each target over learning. For each relevant object, a sample human provided goal images is shown in the top left. We observe that given a single target object, *BEE interacts with the target object more than twice as often as baselines all domains* **(left)** We also observe that given two targets BEE is able to effectively explore them both **(right)**.

Acquiring Human Supervision. The first step of BEE is collecting weak supervision to guide exploration. In this work this supervision comes in the form of a handful of relevant states $[\bar{s}_1, ..., \bar{s}_K]$ sampled uniformly from $\mathcal{S}^*$. For example, if the relevant region of the state space involves interacting with dishes, these relevant states would consist of images of the robot around and interacting with dishes. On a real robot, these images can be collected in a matter of minutes by manually placing the robot in a relevant configuration.

Exploring with BEE. Online exploration with BEE has two central components, which involve (a) determining whether a state is relevant or not, and (b) selecting actions which will enable the agent to reach and explore relevant states. To tackle (a) we leverage an ensemble of relevance discriminators, and address (b) using a model-based planning approach, both of which we detail next.

Relevance Discriminator. To determine which states are relevant, BEE learns an ensemble of L discriminators online which differentiate between the agent's growing dataset $\mathcal{D}$ of collected experience and the relevant states provided by the human. Given states $s \sim \mathcal{D}$ as negatives and human provided relevant states $[\bar{s}_1, ..., \bar{s}_K] \sim \mathcal{S}^*$ as positives, BEE encodes each using a neural network state encoder f_{enc}, then trains each fully connected network ϕ_l as a binary classifier for each element of the ensemble. We additionally leverage both the mix-up regularization and cropping, details of which can be found in the supplement. Now that we've described how the discriminators are trained, how do they translate to our exploratory reward $\mathcal{R}_{exp}$? We would like to select action sequences which explore around states which either the ensemble $[\phi_1, .., \phi_L]$ estimates are relevant, or states for which the ensemble has high uncertainty. Therefore, rather than exploring under the mean ensemble score, we use an optimistic estimate given by the maximum discriminator score over the ensemble models: $\mathcal{R}_{exp}(s, \mathcal{C}) = \max[\phi_1(f_{enc}(s)), ..., \phi_L(f_{enc}(s))]$, which captures both the predicted relevance and the uncertainty.

Learned Dynamics and Planning. BEE learns a latent dynamics model p_θ consisting of three components, (1) an encoder $f_{enc}(z_t|s_t; \theta_{enc})$ that encodes the state s_t into a latent distribution from which z_t is sampled, (2) a decoder $f_{dec}(s_t|z_t; \theta_{dec})$ that reconstructs the observation, providing a reconstruction $\hat{s}_t$, and (3) a deterministic forward dynamics model in the latent space $f_{dyn}(\hat{z}_{t+1}|z_t, a_t; \theta_{dyn})$ which learns to predict the future latent state z_{t+1} from z_t and action a_t. BEE then uses sampling based planning, specifically the cross-entropy method (CEM)[23], in conjunction with the learned latent dynamics model to plan sequences of actions to maximize the exploratory reward $\mathcal{R}_{exp}$. Concretely, it first encodes its current observation into a latent space z_t using the learned encoder f_{enc}. On each iteration of CEM it then samples M action sequences of length H, which it feeds through the latent dynamics model f_{dyn}, resulting in predicted future states $\hat{z}_{t+1:t+H+1}$, which it selects to maximize $\mathcal{R}_{exp}$. Details of the latent MPC procedure can be found in the supplement.

4 Experiments

In our experiments we aim to assess how effectively BEE can explore relevant regions of the state space compared to state-of-the-art approaches in task-agnostic and weakly supervised exploration, testing on simulated and real robotic domains (See Figure 3). Concretely, we ask the following experimental questions: **1)** Does BEE yield improved interaction with relevant objects, while being robust to irrelevant distractor objects? **2)** Does using data collected from BEE lead to better downstream task performance than using data collected via state-of-the-art task-agnostic and weakly supervised exploration techniques? **3)** How does BEE perform on hard exploration tasks on a real robotic

system from images? We compare **BEE** to model disagreement based exploration (**Disagreement**) [8, 13] , state-marginal matching (**SMM**) [12] which also uses the human provided weak supervision, and random exploration (**Random**). Implementation details can be found in the supplement.

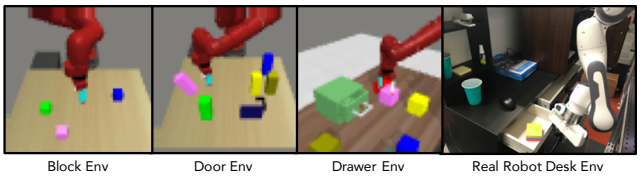

Figure 3: **Experimental Domains.** We consider 3 simulated domains, interacting with blocks, a door, and a drawer. We also test on a real Franka robot interacting with a desk.

Experiment 1: Does BEE Interact More With Relevant Objects?: We begin by measuring how much BEE and the baselines interact with relevant objects specified by the human, shown in Figure 2. We include guided exploration (100 human provided images) toward each of the 3 small blocks, as well as to the door under varying numbers of distractors, and to the drawer. We report the % of the last 100 episodes in which the agent moved the target object more than a threshold every 100 episodes, with the first 100 episodes corresponding to random interaction. We observe that over all domains and targets BEE interacts with the relevant object more than twice as often as the comparisons (Figure 2 (left)). Furthermore, we observe that BEE can be used to guide exploration towards not just a single object, but multiple objects, and find that BEE interacts with both more than the baselines (Figure 2 (right)).

Experiment 2: Does Data from BEE Enable Better Downstream Performance?: For downstream batch RL using the collected data, we consider the model-based self-supervised RL setting. Specifically, we consider the visual foresight algorithm [16, 17], which learns a model of the dynamics from an unlabeled batch of interaction data, then uses this model with planning to reach goals. The model is trained on the full dataset $\mathcal{D}$ collected in the batch exploration phase to predict future states, i.e. $(s_{t+1:t+H}|s_t, a_{t:t+H-1})$. For downstream task planning, we use the SV2P [24] model in conjunction with sampling based planning (CEM), to plan sequences of actions to reach a goal image, under the planning cost of ℓ_2 pixel distance. Further task and planning details can be found in the supplement. We consider five downstream planning tasks, (a) pulling the drawer open, (b) pushing the door with 3 distractors, (c) pushing the door with 5 distractors, (d) pushing the green block left, and (e) pushing the blue block right. We observe in Table 1 that across 4 out of 5 tasks, using the data collected by BEE improves performance over both prior methods and random exploration

	Open Drawer	Push Door (3)	Push Door (5)	Push Green	Push Blue
BEE (Ours)	**0.42**	**0.63**	0.65	**0.47**	**0.50**
Disagreement	0.36	0.59	0.69	0.45	0.44
SMM	0.29	0.58	**0.70**	0.43	0.46
Random	0.31	0.60	0.65	0.45	0.45

Table 1: **Downstream success rates using planning with collected data.** We compare the downstream task performance of using the data generated by BEE for batch RL using the visual foresight method. We observe that across 4 out of 5 tasks BEE is the top performing method. All results are averaged over 1000 trials.

Experiment 3: Is BEE Effective on Hard Exploration Problems on a Real Robot?:

Lastly, we test if BEE can tackle challenging exploration problems on a real robot from pixel inputs. To do so we consider the domain of a Franka robot positioned over a desk, shown in Figure 3. We provide weak supervision (50 human provided images) encouraging interaction with the smaller of the two drawers, and measure the extent to which BEE interacts with the target object compared to Disagreement. We see in Table 1, that not only does BEE interact with the drawer substantially more than Disagreement, but that a dynamics model learned on the collected data is far more effective for the downstream task of closing the drawer. Additional experimental details can be found in the supplement.

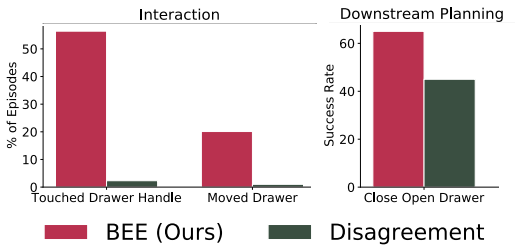

Figure 4: **Performance on a Real Robot.** On the desk interaction task with a real Franka robot, we report the percentage of episodes in which the agent interacts with the target drawer (left, middle), as well as the success rate over 20 trials in the downstream task of closing the drawer (right).

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

[4] 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. In *Conference on Robot Learning*, 2019.

[5] 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.

[6] 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*, 2016.

[7] 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*, 2017.

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

[9] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

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

[11] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *ICML*, 2020.

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

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

[14] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

[15] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel H Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *ArXiv*, abs/1707.01495, 2017.

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

[17] 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:1812.00568*, 2018.

[18] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2019.

[19] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv:1911.11361*, 2019.

[20] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv:2005.05951*, 2020.

[21] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

[22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[23] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

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

[25] 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*, 2020.

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

[27] 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. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[28] 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*, 2018.

[29] 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:1909.12200*, 2019.

[30] 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. 2020.

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

[32] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. In *Robotics Science and Systems*, 2019.

[33] 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. 2016.

[34] 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*, 2016.

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

[36] 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), 2018.

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

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

[39] 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*. Curran Associates, Inc., 2012.

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

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

[42] 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*, 2016.

[43] 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:1802.10567*, 2018.

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

[45] 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:2004.12974*, 2020.

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

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

[48] 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.

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

[50] Lisa Lee, Benjamin Eysenbach, Ruslan Salakhutdinov, Chelsea Finn, et al. Weakly-supervised reinforcement learning for controllable behavior. In *Advances in Neural Information Processing Systems*, 2020.

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

[52] 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*, 2018.

[53] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. 2019.

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

A 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): For all simulation domains, we report results averaged over 5 different random seeds for each method. For each run, we collect a dataset of 2,000 episodes, each of 50 time steps. During online planning, all methods use a single iteration of the cross entropy method to plan a sequence of actions. For each 50-step episode, we replan every 10 steps, i.e. we plan five 10-step trajectories. In simulation, each episode is reset to a fixed initial state. For the real robot domain, we run one random seed for 1000 episodes, each of 100 time steps, also replanning every 10 steps. 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.

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.

B Experimental Details

For the block, door, and drawer domains, we use a Mujoco simulation built off the Meta-World environments [25]. 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. The state space is the space of RGB image observations with size [64, 64, 3]. For the simulation env, we use a continuous action space over the linear and angular velocity of the robot's gripper and a discrete action space over the gripper open/close action, for a total of five dimensions.

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 datasets collected from each of five seeds in the batch exploration phase (a total of 10k episodes) along with 5k random episodes for 100k iterations. For evaluating control on the real robot, for each method we train the SV2P model on the 1000 episodes collected in the batch exploration phase and no random data. We plan 10 actions and execute them in the environment five times for a 100 step trial. Each stage of planning uses the cross entropy method with two iterations, 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 [24]. This architecture is shown in Figure 5, 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.

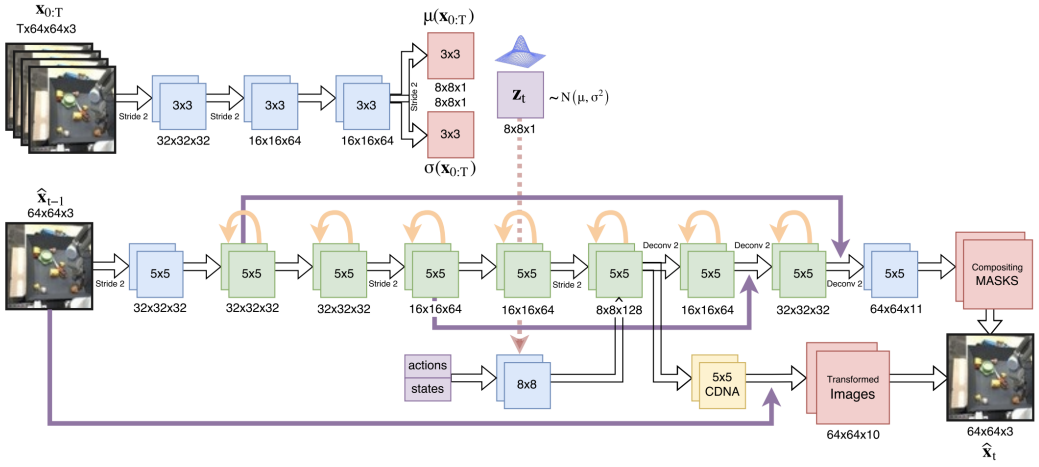

Figure 5: **SV2P architecture.** SV2P estimates the posterior latent distribution $p(z \mid x_{0:T})$ by learning an inference network (top) $q_\phi(z \mid x_{0:T}) = \mathcal{N}(\mu(x_{0:T}), \sigma(x_{0:T}))$. Latent values are sampled from $q_\phi(z \mid x_{0:T})$, and the generative network (bottom) takes in the previous frames, latent values, and actions to predict the next frames. Figure taken from the original paper [24].

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. 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. For the robot downstream task, we labeled the data collected by BEE and Disagreement and trained a reward classifier on 100 examples (labeled as 1) of the drawer open and 200 examples of the drawer closed (labeled as 0), with the gripper sometimes but not always near the drawer. We then conducted planning using this same classifier as the cost for both methods. Success is defined as pushing the drawer closed.

C 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 model 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]. We apply spectral normalization after each layer followed by a ReLU activation (except the final layer, which is followed by a Sigmoid instead).

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.

D 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 [16, 26, 27, 17, 28, 4, 29, 30]. 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 [5, 29] or kinesthetic teaching [31, 32]. 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 [16, 17, 4], pushing [33, 34] and grasping [26, 35, 36]. 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. While heuristics and scripted policies like those employed in grasping can enable more meaningful interactions, designing them for a broad range of tasks can require significant engineering effort.

One way to maintain 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 that optimize for visiting novel states [6, 37, 7, 9, 38], the learning progress of the agent [39, 40], model uncertainty [41, 42, 8, 10, 13], information gain [42], auxiliary tasks [43], generating and reaching goals [11, 44], and state distribution matching [12]. Additionally, a number of these approaches [43, 11, 44, 45] 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 aim to mitigate this challenge by introducing mild supervision into the exploration problem, which we observe empirically yields much more useful exploration than task-agnostic strategies.

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 [46, 47, 48]. Unlike these works, we aim to not rely on any 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, ranging from demonstrations [49], binary labels about state factors of variation [50], and semantic object labels [51] 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 [52, 53] and adversarial imitation learning [54]. 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.