# Watch and Match: Supercharging Imitation with Regularized Optimal Transport

**Anonymous Author(s)**
Affiliation
Address
email

**Abstract:**
Imitation learning holds tremendous promise in learning policies efficiently for complex decision making problems. Current state-of-the-art algorithms often use inverse reinforcement learning (IRL), where given a set of expert demonstrations, an agent alternatively infers a reward function and the associated optimal policy. However, such IRL approaches often require substantial online interactions for complex control problems. In this work, we present Regularized Optimal Transport (ROT), a new imitation learning algorithm that builds on recent advances in optimal transport based trajectory-matching. Our key technical insight is that adaptively combining trajectory-matching rewards with behavior cloning can significantly accelerate imitation even with only a few demonstrations. Our experiments on 20 visual control tasks across the DeepMind Control Suite, the OpenAI Robotics Suite, and the Meta-World Benchmark demonstrate an average of $7.8\times$ faster imitation to reach $90\%$ of expert performance compared to prior state-of-the-art methods. On real-world robotic manipulation, with just one demonstration and an hour of online training, ROT achieves an average success rate of 90.1% across 14 tasks.

**Keywords:** Imitation Learning, Manipulation, Robotics

## 1 Introduction

Imitation Learning (IL) [1, 2, 3] has a rich history that can be categorized across two broad paradigms, Behavior Cloning (BC) [1] and Inverse Reinforcement Learning (IRL) [4]. BC uses supervised learning to obtain a policy that maximizes the likelihood of taking the demonstrated action given an observation in the demonstration. While this allows for training without online interactions, it suffers from distributional mismatch during online rollouts [5]. IRL, on the other hand, infers the underlying reward function from the demonstrated trajectories before employing RL to optimize a policy through online environment rollouts. This results in a policy that can robustly solve demonstrated tasks even in the absence of task-specific rewards [6, 7].

Although powerful, IRL methods suffer from a significant drawback – they require numerous expensive online interactions with the environment. There are three reasons for this: (a) the inferred reward function is often highly non-stationary, which compromises the learning of the associated behavior policy [7]; (b) even when the rewards are stationary, policy learning still requires effective exploration to maximize rewards [8]; and (c) when strong priors such as pretraining with BC are applied to accelerate policy learning, ensuing updates to the policy cause a distribution shift that destabilizes training [9, 10]. Combined, these issues manifest themselves on empirical benchmarks, where IRL methods have poor efficiency compared to vanilla RL methods on hard control tasks [11].

In this work, we present Regularized Optimal Transport (ROT) for imitation learning, a new method that is conceptually simple, compatible with high-dimensional observations, and requires minimal additional hyperparameters compared to standard IRL approaches. To demonstrate the effectiveness of ROT, we run extensive experiments on 20 simulated tasks across DM Control [12], OpenAI Robotics [13], and Meta-world [14], and 14 robotic manipulation tasks on an xArm (see Fig. 3). Our main findings are summarized below.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

1. ROT outperforms prior state-of-the-art imitation methods, reaching $90\%$ of expert performance $7.8\times$ faster than our strongest baselines on simulated visual control benchmarks.
2. On real-world tasks, with a single human demonstration and an hour of training, ROT achieves an average success rate of 90.1% with randomized robot initialization and image observations. This is significantly higher than behavior cloning (36.1%) and adversarial IRL (14.6%).
3. ROT exceeds the performance of state-of-the-art RL trained with rewards, while coming close to methods that augment RL with demonstrations (Appendix I.3). Unlike standard RL methods, ROT does not require hand-specification of the reward function.
4. Ablation studies demonstrate the importance of every component in ROT, particularly the role that soft Q-filtering plays in stabilizing training and the need for OT-based rewards during online learning (Appendix I.2 & Appendix I.4).

Open-source code and demonstration data will be publicly released on our project website. Videos of our trained policies can be seen here: `submissionrot.github.io`.

## 2 Regularized Optimal Transport

A fundamental challenge in imitation learning is to balance the ability to mimic demonstrated actions along with the ability to recover from states outside the distribution of demonstrated states. Behavior Cloning (BC) specializes in mimicking demonstrated actions through supervised learning, while Inverse Reinforcement Learning (IRL) specializes in obtaining policies that can recover from arbitrary states. Regularized Optimal Transport (ROT) combines the best of both worlds by adaptively combining the two objectives. This is done in two phases. In the first phase, a randomly initialized policy is trained using the BC objective on expert demonstrated data. This 'BC-pretrained' policy then serves as an initialization for the second phase. In the second phase, the policy is allowed access to the environment where it can train using an IRL objective. To accelerate the IRL training, the BC loss is added to the objective with an adaptive weight. Details of each component are described below, with additional algorithmic details in Appendix D.

### 2.1 Phase 1: BC Pretraining

BC corresponds to solving the maximum likelihood problem shown in Eq. 1. Here $\mathcal{T}^e$ refers to expert demonstrations. When parameterized by a normal distribution with fixed variance, the objective can be framed as a regression problem where, given inputs $s^e$, $\pi^{BC}$ needs to output $a^e$.

$$\mathcal{L}^{BC} = \mathbb{E}_{(s^e,a^e)\sim\mathcal{T}^e}\|a^e - \pi^{BC}(s^e)\|^2 \tag{1}$$

After training, it enables $\pi^{BC}$ to mimic the actions corresponding to the observations seen in the demonstrations. However, during rollouts in an environment, small errors in action prediction can lead to the agent visiting states not seen in the demonstrations [5]. This distributional mismatch often causes $\pi^{BC}$ to fail on empirical benchmarks [15, 11] (see Fig. 4 (a) in Appendix C).

### 2.2 Phase 2: Online Finetuning with IRL

Given a pretrained $\pi^{BC}$ model, we now begin online 'finetuning' of the policy $\pi^b \equiv \pi^{ROT}$ in the environment. Since we are operating without explicit task rewards, we use rewards obtained through OT-based trajectory matching, which is described in Appendix A. These OT-based rewards $r^{OT}$ enable the use of standard RL optimizers to maximize cumulative reward from $\pi^b \equiv \pi^{ROT}$. In this work we use n-step DDPG [16], a deterministic actor-critic based method that provides high-performance in continuous control [8].

**Finetuning with Regularization** $\pi^{BC}$ is susceptible to distribution shift due to accumulation of errors during online rollouts [5] and directly finetuning $\pi^{BC}$ also leads to subpar performance (refer to Fig. 4 in Appendix C). To address this, we build upon prior work in guided RL [15] and offline RL [9], and regularize the training of $\pi^{ROT}$ by combining it with a BC loss as seen in Eq. 2.

$$\pi^{ROT} = \underset{\pi}{\operatorname{argmax}}\left[(1-\lambda(\pi)))\mathbb{E}_{(s,a)\sim\mathcal{D}_\beta}[Q(s,a)] - \alpha\lambda(\pi)\mathbb{E}_{(s^e,a^e)\sim\mathcal{T}^e}\|a^e - \pi^{BC}(s^e)\|^2\right] \tag{2}$$

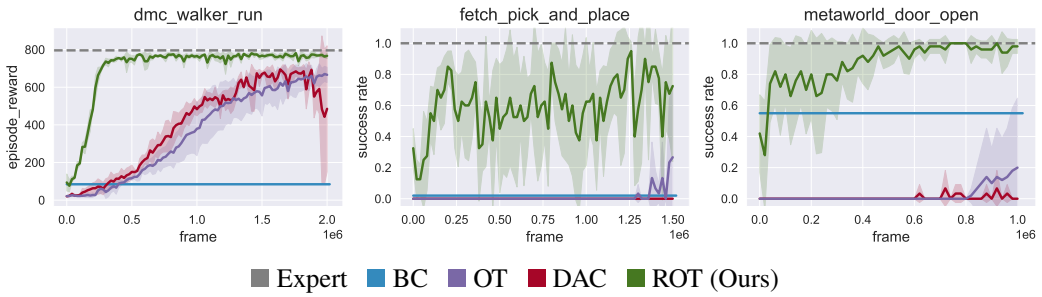


Figure 1: Pixel-based continuous control learning on 3 selected environments. Shaded region represents $\pm 1$ standard deviation across 5 seeds. We notice that ROT is significantly more sample efficient compared to prior work.

Here, $Q(s, a)$ represents the Q-value from the critic which is optimized using OT-based rewards during the actor-critic policy optimization. $\alpha$ is a fixed weight, while $\lambda(\pi)$ is a policy-dependent adaptive weight that controls the contributions of the two loss terms. $\mathcal{D}_\beta$ refers to the replay buffer for online rollouts.

**Adaptive Regularization with Soft Q-filtering** While prior work [15, 17] use hand-tuned schedules for $\lambda(\pi)$, we propose a new adaptive scheme that removes the need for tuning. This is done by comparing the performance of the current policy $\pi^{ROT}$ and the pretrained policy $\pi^{BC}$ on a batch of data sampled from the replay buffer for online rollouts $\mathcal{D}_\beta$. More precisely, given a behavior policy $\pi^{BC}(s)$, the current policy $\pi^{ROT}(s)$, the Q-function $Q(s, a)$ and the replay buffer $\mathcal{D}_e$, we set $\lambda$ as:

$$\lambda(\pi^{ROT}) = \mathbb{E}_{(s,\cdot)\sim\mathcal{D}_\beta}\left[\mathbb{1}_{Q(s,\pi^{BC}(s))>Q(s,\pi^{ROT}(s))}\right] \tag{3}$$

The strength of the BC regularization hence depends on the performance of the current policy with respect to the behavior policy. This filtering strategy is inspired by Nair et al. [18], where instead of a binary hard assignment we use a soft continuous weight. Experimental comparisons with hand-tuned decay strategies are presented in Appendix I.2.

**Considerations for image-based observations** Since we are interested in using ROT with high-dimensional visual observations, additional machinery is required to ensure compatibility. Following prior work in image-based RL and imitation [8, 11], we perform data augmentations on visual observations and then feed it into a CNN encoder. Similar to Cohen et al. [11], we use a target encoder with Polyak averaging to obtain representations for OT reward computation. This is necessary to reduce the non-stationarity caused by learning the encoder alongside the ROT imitation process. Further implementation details and the training procedure can be found in Appendix D.

# 3 Experiments

Our experiments are designed to answer the following questions: (a) How efficient is ROT for imitation learning? (b) How does ROT perform on real-world tasks? Additional results and details about simulated and robot tasks have been provided in Appendix I.

## 3.1 How efficient is ROT for imitation learning?

Performance of ROT for image-based imitation is depicted on select environments in Fig. 1. To reach $90\%$ of expert performance, ROT is on average $8.7\times$ faster on DeepMind Control tasks, $2.1\times$ faster on OpenAI Robotics tasks, and $8.9\times$ faster on Meta-world tasks. We also find that the improvements of ROT are most apparent on the harder tasks, which are shown in Fig. 1. Appendix I.1 shows results on all 20 simulated tasks, along with experiments that exhibit similar improvements in state-based settings.

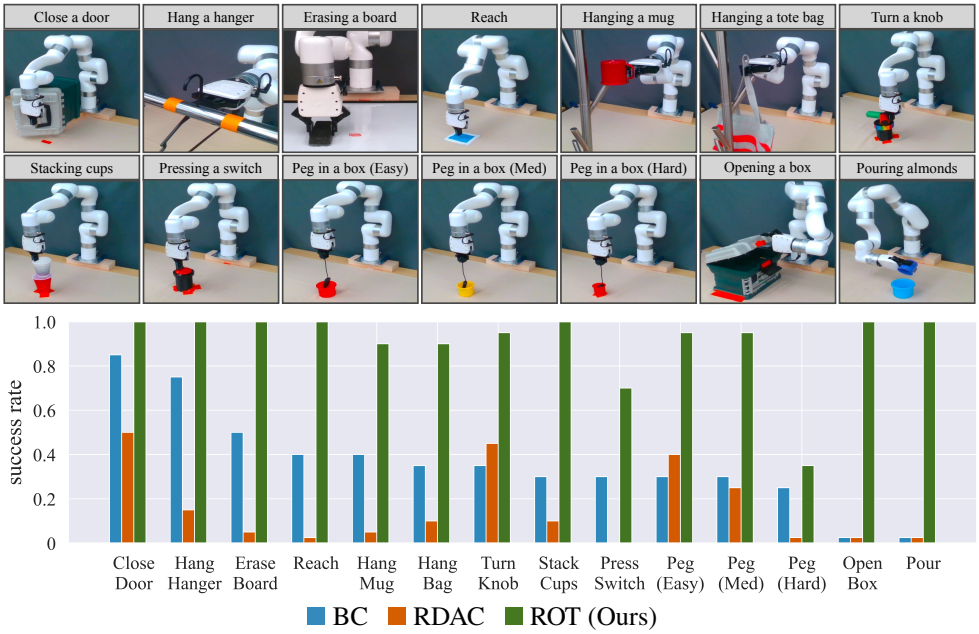


Figure 2: **(Top)** ROT is evaluated on a set of 14 robotic manipulation tasks. **(Bottom)** Success rates for each task is computed by running 20 trajectories from varying initial conditions on the robot.

### 3.2 How does ROT perform on real-world tasks?

We devise a set of 14 manipulation tasks on our xArm robot to compare the performance of ROT with BC and our strongest baseline RDAC, an adversarial IRL method that combines DAC [7] with our pretraining and regularization scheme. The BC policy is trained using supervised learning on a single expert demonstration collected by a human operator. ROT and RDAC finetune the pretrained BC policy through 1 hour of online training, which amounts to $\sim$ 6k environment steps. Since there is just one demonstration, our tasks are designed to have random initializations but fixed goals. Note that a single demonstration only demonstrates solving the tasks from one initial condition. Evaluation results across 20 different initial conditions can be seen in Fig. 2. We observe that ROT has an average success rate of 90.1% over 20 evaluation trajectories across all tasks as compared to 36.1% for BC and 14.6% for RDAC. The poor performance of BC can be attributed to distributional mismatch due to accumulation of error in online rollouts and different initial conditions. The poor performance of RDAC can be attributed to slow learning during the initial phase of training. More detailed evaluations of RDAC on simulated environments is present in Appendix I.4.

## 4 Conclusion and Limitations

In this work, we have proposed a new imitation learning algorithm, ROT, that demonstrates improved performance compared to prior state-of-the-art work on a variety of simulated and robotic domains. However, we recognize a few limitations in this work: (a) Since our OT-based approach aligns agents with demonstrations without task-specific rewards, it relies on the demonstrator being an 'expert'. Extending ROT to suboptimal, noisy and multimodal demonstrations would be an exciting problem to tackle. (b) Performing BC pretraining and BC-based regularization requires access to expert actions, which may not be present in some real-world scenarios particularly when learning from humans. Recent work on using inverse models to infer actions given observational data could alleviate this challenge [19]. (c) On robotic tasks such as *Peg in box (hard)* and *Pressing a switch* from Fig. 2, we find that ROT's performance drops substantially compared to other tasks. This might be due to the lack of visual features corresponding to the task success. For example, in the 'Peg' task, it is visually difficult to discriminate if the peg is in the box or behind the box. Similarly for the 'Switch' task, it is difficult to discern if the button was pressed or not. This limitation can be addressed by integrating more sensory modalities such as additional cameras, and tactile sensors in the observation space.

# References


[1] D. Pomerleau. An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems*, 1, 1998.

[2] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[3] P. N. Kolm and G. Ritter. Modern perspectives on reinforcement learning in finance. *Modern Perspectives on Reinforcement Learning in Finance (September 6, 2019). The Journal of Machine Learning in Finance*, 1(1), 2020.

[4] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[5] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[6] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[7] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

[8] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[9] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[10] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao, et al. Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*, 2022.

[11] S. Cohen, B. Amos, M. P. Deisenroth, M. Henaff, E. Vinitsky, and D. Yarats. Imitation learning from pixel observations for continuous control, 2022. URL https://openreview.net/forum?id=JLbXkHkLCG6.

[12] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[14] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[15] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[17] R. Jena, C. Liu, and K. Sycara. Augmenting gail with bc for sample efficient imitation learning. *arXiv preprint arXiv:2001.07798*, 2020.

[18] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[19] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871. IEEE, 2020.

[20] R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[21] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[23] F. Torabi, G. Warnell, and P. Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

[24] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[25] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy. *arXiv preprint arXiv:2008.04899*, 2020.

[26] R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

[27] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[28] G. Papagiannis and Y. Li. Imitation learning with sinkhorn distances. *arXiv preprint arXiv:2008.09167*, 2020.

[29] P. A. Knight. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

[30] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[32] C. Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[33] G. Peyré, M. Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[34] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[35] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[36] H. Xiao, M. Herman, J. Wagner, S. Ziesche, J. Etesami, and T. H. Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

[37] F. Torabi, G. Warnell, and P. Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[38] E. Cetin and O. Celiktutan. Domain-robust visual imitation learning with mutual information constraints. *arXiv preprint arXiv:2103.05079*, 2021.

[39] S. Toyer, R. Shah, A. Critch, and S. Russell. The magical benchmark for robust imitation. *Advances in Neural Information Processing Systems*, 33:18284–18295, 2020.

[40] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn. Visual adversarial imitation learning using variational models. *Advances in Neural Information Processing Systems*, 34, 2021.

[41] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[42] G. Peyré, M. Cuturi, and J. Solomon. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, pages 2664–2672. PMLR, 2016.

[43] S. Cohen, G. Luise, A. Terenin, B. Amos, and M. Deisenroth. Aligning time series on incomparable spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 1036–1044. PMLR, 2021.

[44] I. Redko, T. Vayer, R. Flamary, and N. Courty. Co-optimal transport. *arXiv preprint arXiv:2002.03731*, 2020.

[45] M. Cuturi and M. Blondel. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, pages 894–903. PMLR, 2017.

[46] A. Fickinger, S. Cohen, S. Russell, and B. Amos. Cross-domain imitation learning via optimal transport. *arXiv preprint arXiv:2110.03684*, 2021.

[47] 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.

[48] S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[49] Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[50] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. {OPAL}: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

[51] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[53] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[54] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[55] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[56] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[57] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

[58] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[59] S. K. S. Ghasemipour, R. Zemel, and S. Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pages 1259–1277. PMLR, 2020.

[60] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

# Appendix

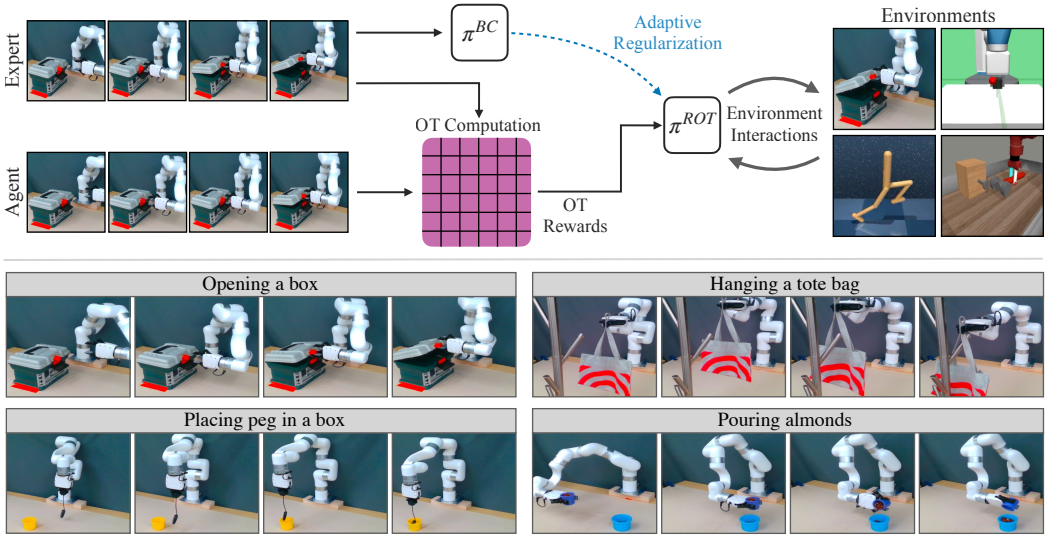


Figure 3: **(Top)** Regularized Optimal Transport (ROT) is a new imitation learning algorithm that adaptively combines offline behavior cloning with online trajectory-matching based rewards. This enables significantly faster imitation across a variety of simulated and real robotics tasks, while being compatible with high-dimensional visual observation. **(Bottom)** On our xArm robot, ROT can learn visual policies with only a single human demonstration and under an hour of online training.

## A Background

**Reinforcement Learning (RL)** We study RL as a discounted infinite-horizon Markov Decision Process (MDP) [20, 21]. For pixel observations, the agent's state is approximated as a stack of consecutive RGB frames [22]. The MDP is of the form $(\mathcal{O}, \mathcal{A}, P, R, \gamma, d_0)$ where $\mathcal{O}$ is the observation space, $\mathcal{A}$ is the action space, $P : \mathcal{O} \times \mathcal{A} \to \Delta(\mathcal{O})$ is the transition function that defines the probability distribution over the next state given the current state and action, $R : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma$ is the discount factor and $d_0$ is the initial state distribution. The goal is to find a policy $\pi : \mathcal{O} \to \Delta(\mathcal{A})$ that maximizes the expected discount sum of rewards $\mathbb{E}_\pi[\Sigma_{t=0}^{\infty} \gamma^t R(\boldsymbol{o}_t, \boldsymbol{a}_t)]$, where $\boldsymbol{o}_0 \sim d_0$, $a_t \sim \pi(\boldsymbol{o}_t)$ and $\boldsymbol{o}_{t+1} \sim P(.|\boldsymbol{o}_t, \boldsymbol{a}_t)$.

**Imitation Learning (IL)** The goal of imitation learning is to learn a behavior policy $\pi^b$ given access to either the expert policy $\pi^e$ or trajectories derived from the expert policy $\mathcal{T}^e$. While there are a multitude of settings with differing levels of access to the expert [23], this work operates in the setting where the agent only has access to observation-based trajectories, i.e. $\mathcal{T}^e \equiv \{(o_t, a_t)_{t=0}^T\}_{n=0}^N$. Here $N$ and $T$ denotes the number of trajectory rollouts and episode timesteps respectively. We choose this specific setting since obtaining observations and actions from expert or near-expert demonstrators is feasible in real-world settings [24, 25] and falls in line with recent work in this area [26, 6, 7].

**Inverse Reinforcement Learning (IRL)** IRL [4, 27] tackles the IL problem by inferring the reward function $r^e$ based on expert trajectories $\mathcal{T}^e$. Then given the inferred reward $r^e$, policy optimization is used to derive the behavior policy $\pi^b$. Prominent algorithms in IRL [7, 6] requires alternating the inference of reward and optimization of policy in an iterative manner, which is practical for restricted model classes [27]. For compatibility with more expressive deep networks, techniques such as adversarial learning [6, 7] or optimal-transport [28, 26, 11] are needed. Adversarial learning based approaches tackle this problem by learning a discriminator that models the gap between the expert trajectories $\mathcal{T}^e$ and behavior trajectories $\mathcal{T}^b$. The behavior policy $\pi^b$ is then optimized to minimize this gap through gap-minimizing rewards $r^e$. Such a training procedure is prone to instabilities since $r^e$ is updated at every iteration and is hence non-stationary for the optimization of $\pi^b$.

**Optimal Transport for Imitation Learning (OT)** To alleviate the non-stationary reward problem with adversarial IRL frameworks, a new line of OT-based approaches have been recently proposed [28, 26, 11]. Intuitively, the closeness between expert trajectories $\mathcal{T}^e$ and behavior trajectories $\mathcal{T}^b$ can be computed by measuring the optimal transport of probability mass from $\mathcal{T}^b \rightarrow \mathcal{T}^e$. During policy learning, the policy $\pi_\phi$ encompasses a feature preprocessor $f_\phi$ which transforms observations into informative state representations. Some examples of a preprocessor function $f_\phi$ are an identity function, a mean-variance scaling function and a parametric neural network. In this work, we use a parametric neural network as $f_\phi$. Given a cost function $c : \mathcal{O} \times \mathcal{O} \rightarrow \mathbb{R}$ defined in the preprocessor's output space and an OT objective $g$, the optimal alignment between an expert trajectory $\mathbf{o}^e$ and a behavior trajectory $\mathbf{o}^b$ can be computed as

$$\mu^* \in \arg\min_{\mu \in \mathcal{M}} g(\mu, f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e), c) \tag{4}$$

where $\mathcal{M} = \{\mu \in \mathbb{R}^{T \times T} : \mu \mathbf{1} = \mu^T \mathbf{1} = \frac{1}{T}\mathbf{1}\}$ is the set of coupling matrices and the cost $c$ can be the Euclidean or Cosine distance. In this work, inspired by [11], we use the entropic Wasserstein distance with cosine cost as our OT metric, which is given by the equation

$$\begin{aligned} g(\mu, f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e), c) &= \mathcal{W}^2(f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e)) \\ &= \sum_{t,t'=1}^{T} C_{t,t'} \mu_{t,t'} \end{aligned} \tag{5}$$

where the cost matrix $C_{t,t'} = c(f_\phi(\mathbf{o}^b), f_\phi(\mathbf{o}^e))$. Using Eq. 5 and the optimal alignment $\mu^*$ obtained by optimizing Eq. 4, a reward signal can be computed for each observation using the equation

$$r^{OT}(o_t^{\boldsymbol{b}}) = -\sum_{t'=1}^{T} C_{t,t'} \mu^*_{t,t'} \tag{6}$$

Intuitively, maximizing this reward encourages the imitating agent to produce trajectories that closely match demonstrated trajectories. Since solving Eq. 4 is computationally expensive, approximate solutions such as the Sinkhorn algorithm [29, 28] are used instead.

**Actor-Critic based reward maximization** Given rewards obtained through OT computation, efficient maximization of the reward can be achieved through off-policy learning [7]. In this work, we use Deep Deterministic Policy Gradient (DDPG) [16] as our base RL optimizer which is an actor-critic algorithm that concurrently learns a deterministic policy $\pi_\phi$ and a Q-function $Q_\theta$. However, instead of minimizing a one step Bellman residual in vanilla DDPG, we use the recent n-step version of DDPG from Yarats et al. [8] that achieves high performance on visual control problems.

# B Related Work

**Imitation Learning (IL)** IL [30] refers to the setting where agents learn from demonstrations without access to environment rewards. IL can be broadly categorized into Behavior Cloning (BC) [1, 23] and Inverse Reinforcement Learning (IRL) [4, 27]. BC solely learns from offline demonstrations but suffers on out-of-distributions samples [5] whereas IRL focuses on learning a robust reward function through online interactions but suffers from sample inefficiency [7]. Deep IRL methods can be further divided into two categories: $(1)$ adversarial learning [31] based methods, and $(2)$ state-matching [32, 33] based methods. GAIL [6] is an adversarial learning based formulation inspired by maximum entropy IRL [34] and GANs [31]. There has been a significant body of work built up on GAIL proposing alternative losses [35, 36, 37], and enhancing its sample efficiency by porting it to an off-policy setting [7]. There have also been visual extensions of these adversarial learning approaches [38, 39, 40, 11]. However, although adversarial methods produce competent policies, they are inefficient due to the non-stationarity associated with iterative reward inference [11].

**Optimal Transport (OT)** OT [32, 33] is a tool for comparing probability measures while including the geometry of the space. In the context of IL, OT computes an alignment between a set of agent

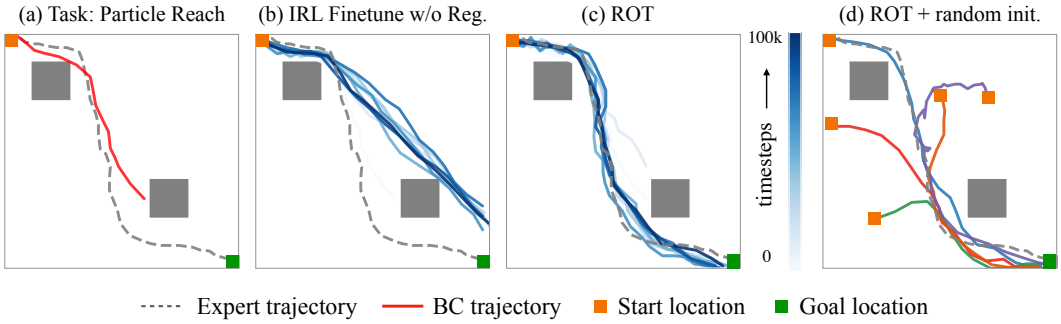


Figure 4: Given a single demonstration to avoid the grey obstacle and reach the goal location, BC is unable to solve the task (a). Finetuning from this BC policy with OT-based reward also fails to solve the task (b). ROT, with adaptive regularization of OT-based IRL with BC successfully solves the task (c). Even when the ROT agent is initialized randomly, it is able to solve the task (d).

and expert observations using distance metrics such as Sinkhorn [41], Gromov-Wasserstein [42], GDTW [43], CO-OT [44] and Soft-DTW [45]. For many of these distance measures, there is an associated IL algorithm, with SIL [28] using Sinkhorn, PWIL [26] using greedy Wasserstein, GDTW-IL [43] using GDTW, and GWIL [46] using Gromov-Wasserstein. Recent work from Cohen et al. [11] demonstrates that the Sinkhorn distance [28] produces the most efficient learning among the discussed metrics. They further show that SIL is compatible with high-dimensional visual observations and encoded representations. Inspired by this, ROT adopts the Sinkhorn metric for its OT reward computation, and improves upon SIL through adaptive behavior regularization.

**Behavior Regularized Control** Behavior regularization is a widely used technique in offline RL [47] where explicit constraints are added to the policy improvement update to avoid bootstrapping on out-of-distribution actions [48, 49, 50, 51, 52, 53]. In an online setting with access to environment rewards, prior work [15, 10] has shown that behavior regularization can be used to boost sample efficiency by finetuning a pretrained policy via online interactions. For instance, Jena et al. [17] demonstrates the effectiveness of behavior regularization to enhance sample efficiency in the context of adversarial IL. ROT builds upon this idea by extending to visual observations, OT-based IL, and adaptive regularization, which leads to improved performance (see Appendix I.4). We also note that the idea of using adaptive regularization has been previously explored in RL [18]. However, ROT uses a soft, continuous adaptive scheme, which on initial experiments provided significantly faster learning compared to hard assignments.

# C Challenges in Online Finetuning from a Pretrained Policy

In this section, we study the challenges with finetuning a pretrained policy with online interactions in the environment. Fig. 4 illustrates a task where an agent is supposed to navigate the environment from the top left to the bottom right, while dodging obstacles in between. The agent has access to a single expert demonstration, which is used to learn a BC policy for the task. Fig. 4 (a) shows that this BC policy, though close to the expert demonstration, performs suboptimally due to accumulating errors on out-of-distribution states during online rollouts [5]. Further, Fig. 4 (b) uses this BC policy as an initialization and naively finetunes it with OT rewards (described in Section **??**). Such naive finetuning of a pretrained policy (or actor) with an untrained critic in an actor-critic framework exhibits a forgetting behavior in the actor, resulting in performance degradation as compared to the pretrained policy. This phenomenon has also been reported by Nair et al. [9] and we provide a detailed discussion in Appendix C.1. In this paper, we propose ROT which addresses this issue by adaptively keeping the policy close to the behavior data during the initial phase of finetuning and reduces this dependence over time. Fig. 4 (c) demonstrates the performance of our approach on such finetuning. It can be clearly seen that even though the BC policy is suboptimal, our proposed adaptive regularization scheme quickly improves and solves the task by driving it closer to the expert demonstration. In Fig. 4 (d), we demonstrate that even if the agent was initialized at points outside the expert trajectory, the agent is still able to learn quickly and complete the task. This generalization to starting states would not be possible with regular BC.

### C.1 Issue with Fine-tuning Actor-Critic Frameworks

In this paper, we use $n$-step DDPG proposed by Yarats et al. [8] as our RL optimizer for actor-critic based reward maximization. DDPG [16] concurrently learns a deterministic policy $\pi_\phi$ using deterministic policy gradients (DPG) [54] and a Q-function $Q_\theta$ by minimizing a n-step Bellman residual (for n-step DDPG). For a parameterized actor network $\pi_\phi(s)$ and a critic function $Q_\theta(s, a)$, the deterministic policy gradients (DPG) for updating the actor weights is given by

$$\begin{aligned} \nabla_\phi J &\approx \mathbb{E}_{s_t \sim \rho_\beta} \left[ \nabla_\phi \, Q_\theta(s, a)|_{s=s_t, a=\pi_\phi(s_t)} \right] \\ &= \mathbb{E}_{s_t \sim \rho_\beta} \left[ \nabla_a \, Q_\theta(s, a)|_{s=s_t, a=\pi_\phi(s_t)} \, \nabla_\phi \, \pi_\phi(s)|_{s=s_t} \right] \end{aligned} \tag{7}$$

Here, $\rho_\beta$ refers to the state visitation distribution of the data present in the replay buffer at time $t$. From Eq. 7, it is clear that the policy gradients in this framework depend on the gradients with respect to the critic value. Hence, as mentioned in [9, 10], naively initializing the actor with a pretrained policy while using a randomly initialized critic results in the untrained critic providing an exceedingly poor signal to the actor network during training. As a result, the actor performance drops immediately and the good behavior of the informed initialization of the policy gets forgotten. In this paper, we propose an adaptive regularization scheme that permits finetuning a pretrained actor policy in an actor-critic framework. As opposed to Rajeswaran et al. [15], Jena et al. [17] which employ on-policy learning, our method is off-policy and aims to leverage the sample efficient characteristic of off-policy learning as compared to on-policy learning [7].

## D Algorithmic Details

**Algorithm 1** ROT: Regularized Optimal Transport

**Require:**
Expert Demonstrations $\mathcal{T}^e \equiv \{(o_t, a_t)_{t=0}^T\}_{n=0}^N$
Pretrained policy $\pi^{BC}$
Replay buffer $\mathcal{D}$, Training steps $T$, Episode Length $L$
Task environment $env$
Parametric networks for RL backbone (e.g., the encoder, policy and critic function for DrQ-v2)
A discriminator $D$ for adversarial baselines

**Algorithm:**
$\pi^{ROT} \leftarrow \pi^{BC}$ ▷ Initialize with pretrained policy
**for** each timestep $t$ = $1...T$ **do**
  **if** done **then**
    $r_{1:L}$ = rewarder$_{OT}$(episode) ▷ OT-based reward computation
    Update episode with $r_{1:L}$ and add $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1}, r_t)$ to $\mathcal{D}$
    $\mathbf{o}_t = env.reset(),$ done $=$ False, episode $= [\,]$
  **end if**
  $\mathbf{a}_t = \pi^{ROT}(\mathbf{o}_t)$
  $\mathbf{o}_{t+1},$ done $= env.step(\mathbf{a}_t)$
  episode.append$([\mathbf{o}_t, \mathbf{a}_t, \mathbf{o}_{t+1}])$
  Update backbone-specific networks and reward-specific networks using $\mathcal{D}$
**end for**

### D.1 Implementation

Algorithm 1 describes our proposed algorithm, Regularized Optimal Transport (ROT), for sample efficient imitation learning for continuous control tasks. Further implementation details are as follows:

**Algorithm and training procedure** Our model consists of 3 primary neural networks - the encoder, the actor and the critic. During the BC pretraining phase, the encoder and the actor are trained using a mean squared error (MSE) on the expert demonstrations. Next, for finetuning, weights of the

pretrained encoder and actor are loaded from memory and the critic is initialized randomly. We observed that the performance of the algorithm is not very sensitive to the value of $\alpha$ and we set it to $0.03$ for all experiments in this paper. A copy of the pretrained encoder and actor are stored with fixed weights to be used for computing $\lambda(\pi)$ for soft Q-filtering.

**Actor-critic based reward maximization** We use a recent n-step DDPG proposed by Yarats et al. [8] as our RL backbone. The deterministic actor is trained using deterministic policy gradients (DPG) [54] given by Eq. 7. The critic is trained using clipped double Q-learning similar to Yarats et al. [8] in order to reduce the overestimation bias in the target value. This is done using two Q-functions, $Q_{\theta 1}$ and $Q_{\theta 2}$. The critic loss for each critic is given by the equation

$$\mathcal{L}_{\theta_k} = \mathbb{E}_{(s,a)\sim D_\beta}\left[(Q_{\theta_k}(s,a) - y)^2\right] \forall\, k \in \{1,2\} \tag{8}$$

where $\mathcal{D}_\beta$ is the replay buffer for online rollouts and $y$ is the target value for n-step DDPG given by

$$y = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \min_{k=1,2} Q_{\bar{\theta}_k}(s_{t+n}, a_{t+n}) \tag{9}$$

Here, $\gamma$ is the discount factor, $r$ is the reward obtained using OT-based reward computation and $\bar{\theta}_1$, $\bar{\theta}_2$ are the slow moving weights of target Q-networks.

**Target feature processor to stabilize OT rewards** The OT rewards are computed on the output of the feature processor $f_\phi$ which is initialized with a parametric neural network. Hence, as the weights of $f_\phi$ change during training, the rewards become non-stationary resulting in unstable training. In order to increase the stability of training, the OT rewards are computed using a target feature processor $f_{\phi'}$ [11] which is updated with the weights of $f_\phi$ every $T_{update}$ environment steps. For state-based observations, $f_\phi$ corresponds to a 'trunk' network which is a single layer neural network. For pixel-based observations, $f_\phi$ includes DrQ-v2's encoder followed by the 'trunk' network.

### D.2 Hyperparameters

The complete list of hyperparameters is provided in Table 1. Similar to Yarats et al. [8], there is a slight deviation from the given setting for the Walker Stand/Walk/Run task from the DeepMind Control suite where we use a mini-batch size of 512 and a $n$-step return of 1.

## E Environments

Table 2 lists the different tasks that we experiment with from the DeepMind Control suite [12, 55], OpenAI Robotics suite [56] and the Meta-world suite [57] along with the number of training steps and the number of demonstrations used. For the tasks in the OpenAI Robotics suite, we fix the goal while keeping the initial state randomized. No modifications are made in case of the DeepMind Control suite and the Meta-world suite. The episode length for all tasks in DeepMind Control is 1000 steps, for OpenAI Robotics is 50 steps and Meta-world is 125 steps (except bin picking which runs for 175 steps).

## F Demonstrations

For DeepMind Control tasks, we train expert policies using pixel-based DrQ-v2 [8] and collect 10 demonstrations for each task using this expert policy. The expert policy is trained using a stack of 3 consecutive RGB frames of size $84 \times 84$ with random crop augmentation. Each action in the environment is repeated 2 times. For OpenAI Robotics tasks, we train a state-based DrQ-v2 with hindsight experience replay [58] and collect 50 demonstrations for each task. The state representation comprises the observation from the environment appended with the desired goal location. For this, we did not do frame stacking and action repeat was set to 2. For Meta-World tasks, we use a single expert demonstration obtained using the task-specific hard-coded policies provided in their open-source implementation [57].

| Method | Parameter | Value |
|---|---|---|
| Common | Replay buffer size | 150000 |
| | Learning rate | $1e^{-4}$ |
| | Discount $\gamma$ | 0.99 |
| | $n$-step returns | 3 |
| | Action repeat | 2 |
| | Seed frames | 12000 |
| | Mini-batch size | 256 |
| | Agent update frequency | 2 |
| | Critic soft-update rate | 0.01 |
| | Feature dim | 50 |
| | Hidden dim | 1024 |
| | Optimizer | Adam |
| ROT | Exploration steps | 0 |
| | DDPG exploration schedule | 0.1 |
| | Target feature processor update frequency(steps) | 20000 |
| | Reward scale factor | 10 |
| | Fixed weight $\alpha$ | 0.03 |
| | Linear decay schedule for $\lambda(\pi)$ | linear(1,0.1,20000) |
| OT | Exploration steps | 2000 |
| | DDPG exploration schedule | linear(1,0.1,500000) |
| | Target feature processor update frequency(steps) | 20000 |
| | Reward scale factor | 10 |
| DAC | Exploration steps | 2000 |
| | DDPG exploration schedule | linear(1,0.1,500000) |
| | Gradient penalty coefficient | 10 |

Table 1: List of hyperparameters.

# G Robot Tasks

In this section, we describe the suite of manipulation experiments carried out on a xArm robot in this paper.

(a) **Door Close:** Here, the robot arm is supposed to close an open door by pushing it to the target.

(b) **Hang Hanger:** While holding a hanger between the grippers, the robot arm is initialized at a random position and is tasked with putting the hanger at a goal region on a closet rod.

(c) **Erase Board:** While holding a board duster between the grippers, the robot arm is tasked with erasing markings drawn on the board while being initialized at a random position.

(d) **Reach:** The robot arm is required to reach a specific goal after being initialized at a random position.

(e) **Hang Mug:** While holding a mug between the grippers, the robot arm is initialized at a random position and is tasked with hanging the mug on a specific hook.

| Suite | Tasks | Allowed Steps | # Demonstrations |
|---|---|---|---|
| DeepMind Control | Acrobot Swingup | $2 \times 10^6$ | 10 |
| | Cartpole Swingup | | |
| | Cheetah Run | | |
| | Finger Spin | | |
| | Hopper Stand | | |
| | Hopper Hop | | |
| | Quadruped Run | | |
| | Walker Stand | | |
| | Walker Walk | | |
| | Walker Run | | |
| OpenAI Robotics | Fetch Reach | $1.5 \times 10^6$ | 50 |
| | Fetch Push | | |
| | Fetch Pick and Place | | |
| Meta-World | Hammer | $1 \times 10^6$ | 1 |
| | Drawer Close | | |
| | Door Open | | |
| | Bin Picking | | |
| | Button Press Topdown | | |
| | Door Unlock. | | |
| xArm Robot | Close Door | $6 \times 10^3$ | 1 |
| | Hang Hanger | | |
| | Erase Board | | |
| | Reach | | |
| | Hang Mug | | |
| | Hang Bag | | |
| | Turn Knob | | |
| | Stack Cups | | |
| | Press Switch | | |
| | Peg (Easy) | | |
| | Peg (Medium) | | |
| | Peg (Hard) | | |
| | Open Box | | |
| | Pour | | |

Table 2: List of tasks used for evaluation.

(f) **Hang Bag:** While holding a tote between the grippers, the robot arm is initialized at a random position and is tasked with hanging the tote bag on a specific hook.

(g) **Turn Knob:** The robot arm is tasked with rotating a knob placed on the table by a certain angle after being initialized at a random position. We consider a 90 degree rotation as success.

Figure 5: Examples of randomized initializations for the real robot tasks.

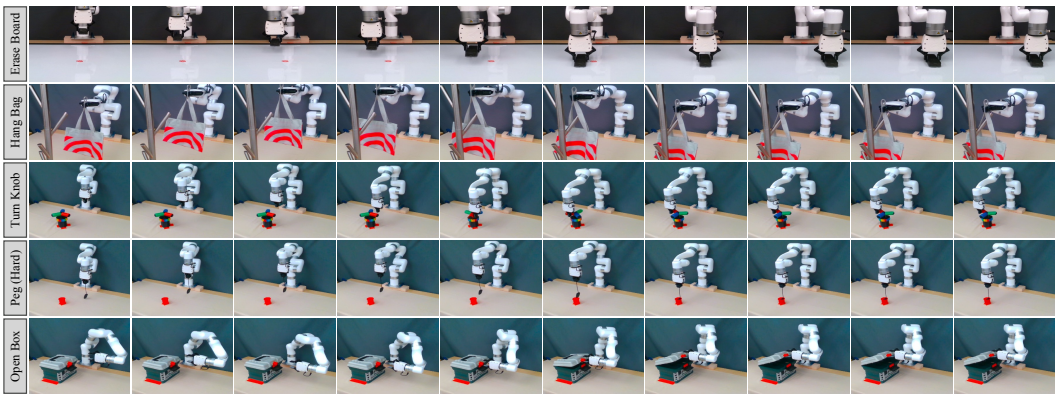


Figure 6: An example of trajectories for selected real robot tasks.

(h) **Stack Cups:** While holding a cup between the gripper, the robot arm is required with stacking it on another cup placed on the table.

(i) **Press Switch:** With the gripper kept closed, the robot arm is required to press a switch (with an LED light) placed on the table.

(j) **Peg (Easy, Medium, Hard):** The robot arm is tasked with inserting a peg, hanging by a wire, into a bucket placed on the table. This task has 3 variants - Easy, Medium, Hard - with the size of the bucket decreasing from Easy to Hard.

(k) **Box Open:** In this task, the robot arm is supposed to open the lid of a box placed on the table by lifting a handle provided in the front of the box.

(l) **Pour:** While holding a cup containing some item (in our case, almonds), the robot arm is supposed to move towards another cup placed on the table and pour the item into this cup.

**Evaluation procedure** For each task, we obtained a set of 20 random initializations and evaluate all of the methods (BC, RDAC and ROT) over 20 trajectories from the same set of initializations. These initializations are different for each task based on the limits of the observation space for the task.

# H Baselines

Throughout the paper, we compare ROT with several prominent imitation learning and reinforcement learning methods. Here, we give a brief description of each of the baseline models that have been used.

(a) **Expert:** For each task, the expert refers to the expert policy used to generate the demonstrations for the task (described in Appendix F).

(b) **Behavior Cloning (BC):** This refers to the behavior cloned policy trained on expert demonstrations.

(c) **Adversarial IRL (DAC):** Discriminator Actor Critic [7] is a state-of-the-art adversarial imitation learning method [6, 37, 7]. Since DAC outperforms prior work such as GAIL[6] and AIRL[35], it serves as our primary adversarial imitation baseline.

(d) **State-matching IRL (OT):** Sinkhorn Imitation Learning [28, 26] is a state-of-the-art state-matching imitation learning method [59] that approximates OT matching through the Sinkhorn Knopp algorithm. Since ROT is derived from similar OT-based foundations, we use SIL as our primary state-matching imitation baseline.

(e) **RDAC:** This is the same as ROT, but instead of using state-matching IRL (OT), adversarial IRL (DAC) is used.

(f) **Finetune with fixed weight:** This is similar to ROT where instead of using a time-varying adaptive weight $\lambda(i)$, only the fixed weight $\lambda_0$ is used. $\lambda_0$ is set to a fixed value of $0.03$.

(g) **Finetune with fixed schedule:** This is similar to ROT that uses both the fixed weight $\lambda_0$ and the time-varying adaptive weight $\lambda_1(i)$. However, instead of using Soft Q-filtering to compute $\lambda_1(i)$, a hand-coded linear decay schedule is used.

(h) **DrQ-v2 (RL):** DrQ-v2 [8] is a state-of-the-art algorithm for pixel-based RL. DrQ-v2 is assumed to have access to environment rewards as opposed to ROT which computes the reward using OT-based techniques.

(i) **Demo-DrQ-v2:** This refers to DrQ-v2 but with access to both environment rewards and expert demonstrations. The model is initialized with a pretrained BC policy followed by RL finetuning with an adaptive regularization scheme like ROT. During RL finetuning, this baseline has access to environment rewards.

(j) **BC+OT:** This is the same as the OT baseline but the policy is initialized with a pretrained BC policy. No adaptive regularization scheme is used while finetuning the pretrained policy.

(k) **OT+BC Reg.:** This is the same as the OT baseline with randomly initialized networks but during training, the adaptive regularization scheme is added to the objective function.

# I Experimental Results

Our experiments are designed to answer the following questions: (a) How efficient is ROT for imitation learning? (b) How does ROT perform on real-world tasks? (c) How important is the choice of IRL method in ROT? (d) Does soft Q-filtering improve imitation? (e) How does ROT compare to standard reward-based RL?

**Simulated tasks** We experiment with 10 tasks from the DeepMind Control suite [12, 55], 3 tasks from the OpenAI Robotics suite [56], and 7 tasks from the Meta-world suite [57]. For DeepMind Control tasks, we train expert policies using DrQ-v2 [8] and collect 10 demonstrations for each task using this policy. For OpenAI Robotics tasks, we train a state-based DrQ-v2 with hindsight experience replay [58] and collect 50 demonstrations for each task. For Meta-world tasks, we use a single hard-coded expert demonstration from their open-source implementation [57]. Full environment details can be found in Appendix E and details about the variations in demonstrations and initialization conditions can be found in Appendix F.

**Robot tasks** Our real world setup for each of the 14 manipulation tasks can be seen in Fig. 2. We use an Ufactory xArm 7 robot with a xArm Gripper as the robot platform for our real world experiments. However, our method is agnostic to the specific robot hardware. The observations are RGB images from a fixed camera. In this setup, we only use a single expert demonstration collected by a human operator with a joystick and limit the online training to a fixed period of 1 hour. Descriptions of each task and the evaluation procedure is in Appendix G.

**Primary baselines** We compare ROT with baselines against several prominent imitation learning methods. While a full description of our baselines are in Appendix H, a brief description of the two strongest ones are as follows:

1. **Adversarial IRL (DAC):** Discriminator Actor Critic [7] is a state-of-the-art adversarial imitation learning method [6, 37, 7]. DAC outperforms prior work such as GAIL [6] and AIRL [35], and thus it serves as our primary adversarial imitation baseline.
2. **Trajectory-matching IRL (OT):** Sinkhorn Imitation Learning [28, 26] is a state-of-the-art trajectory-matching imitation learning method [59] that approximates OT matching through the Sinkhorn Knopp algorithm [60, 41]. Since ROT is derived from similar OT-based foundations, we use SIL as our primary state-matching imitation baseline.

## I.1 How efficient is ROT for imitation learning?

In addition to the results provided in Sec. 3.1, Fig. 7 and Fig. 8 shows the performance of ROT for pixel-based imitation on 10 tasks from the DeepMind Control suite, 3 tasks from the OpenAI Robotics suite and 7 tasks from the Meta-world suite. On all but one task, ROT is significantly more sample efficient than prior work. Finally, the improvements from ROT hold on state-based

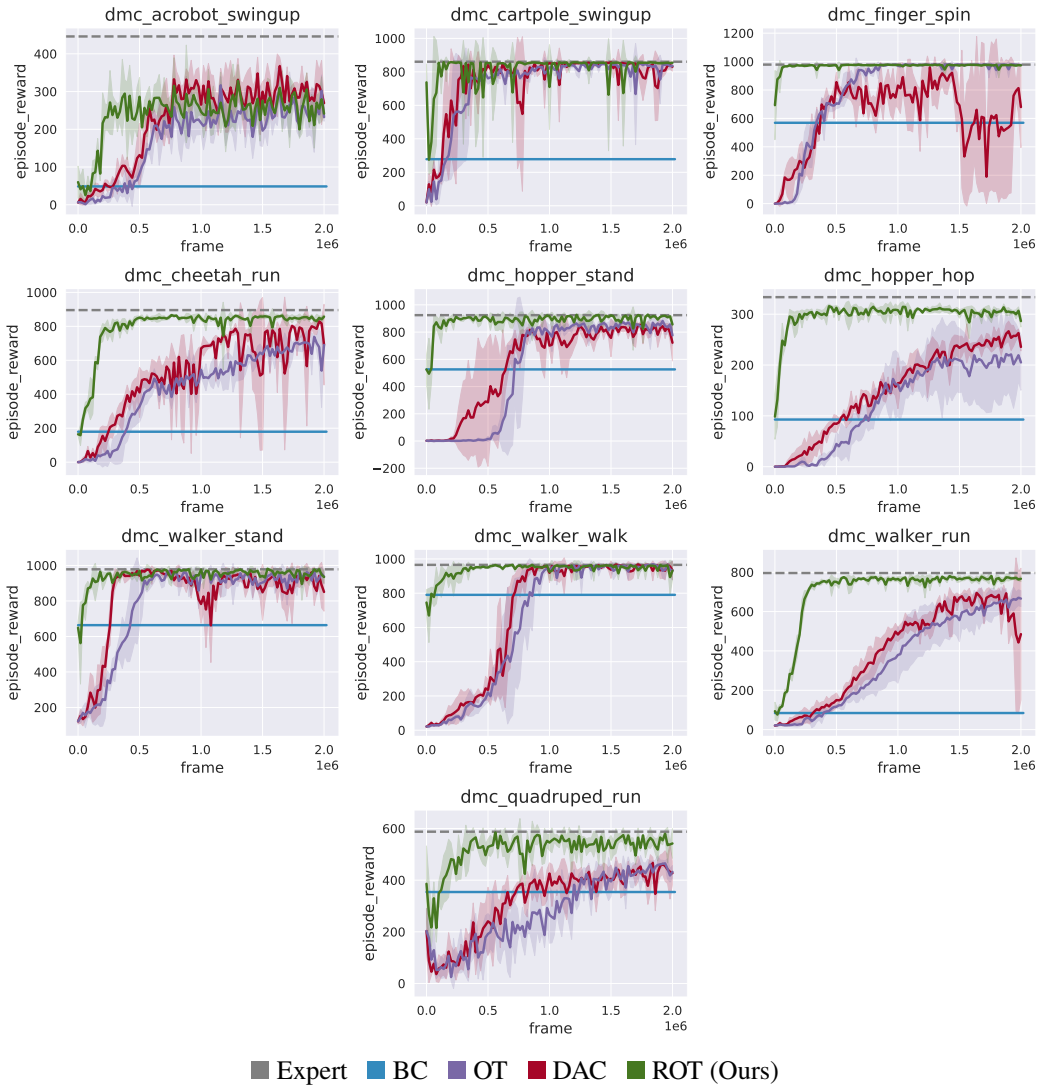


Figure 7: Pixel-based continuous control learning on 10 DMC environments. Shaded region represents $\pm 1$ standard deviation across 5 seeds. We notice that ROT is significantly more sample efficient compared to prior work.

observations as well(see Fig. 9). Table 3 provides a comparison between the factor of speedup of ROT to reach 90% of expert performance compared to prior state-of-the-art [7, 11] methods.

### I.2 Does soft Q-filtering improve imitation?

To understand the importance of soft Q-filtering, we compare ROT against two variants of our proposed regularization scheme: (a) A tuned fixed BC regularization weight (ignoring $\lambda(\pi)$ in Eq. 2); (b) A carefully designed linear-decay schedule for $\lambda(\pi)$, where it varies from $1.0$ to $0.0$ in the first 20k environment steps [15]. As demonstrated in Fig. 10, ROT is on par and in some cases exceeds the efficiency of a hand-tuned decay schedule, while not having to hand-tune its regularization weights. We hypothesize this improvement is primarily due to the better stability of adaptive weighing as seen in the significantly smaller standard deviation on the Meta-world tasks.

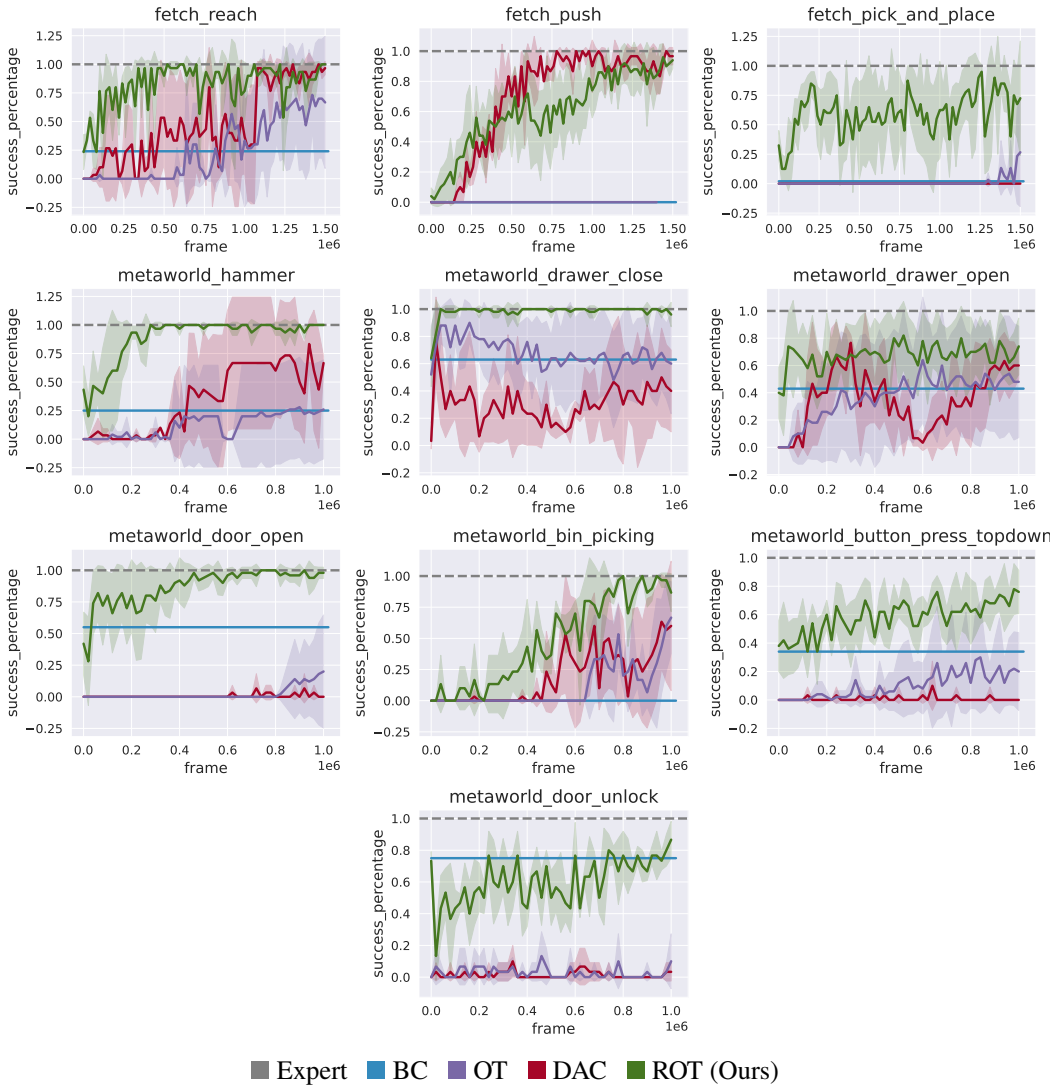


Figure 8: Pixel-based continuous control learning on 3 OpenAI Gym Robotics and 7 Meta-World tasks. Shaded region represents $\pm 1$ standard deviation across 5 seeds. We notice that ROT is significantly more sample efficient compared to prior work.

### I.3 How does ROT compare to standard reward-based RL?

We compare the performance of ROT against DrQ-v2 [8], a state-of-the-art algorithm for image-based RL. As opposed to the reward-free setting ROT operates in, DrQ-v2 has access to environments rewards. The results in Fig. 11 show that ROT handily outperforms DrQ-v2. This clearly demonstrates the usefulness of imitation learning in domains where expert demonstrations are available over reward-based RL. We also compare against a demo-assisted variant of DrQ-v2 agent using the same pretraining and regularization scheme as ROT. Interestingly, we find that our soft Q-filtering based regularization can accelerate learning of RL with task rewards, which can be seen in the high performance of the demo-assisted variant of DrQ-v2.

### I.4 How important are the design choices in ROT?

**Importance of pretraining and regularizing the IRL policy** Fig. 12 compares the following variants of ROT on set of pixel-based tasks: (a) Training the IRL policy from scratch (OT); (b) Finetuning a pretrained BC policy without BC regularization (BC+OT); (c) Training the IRL policy

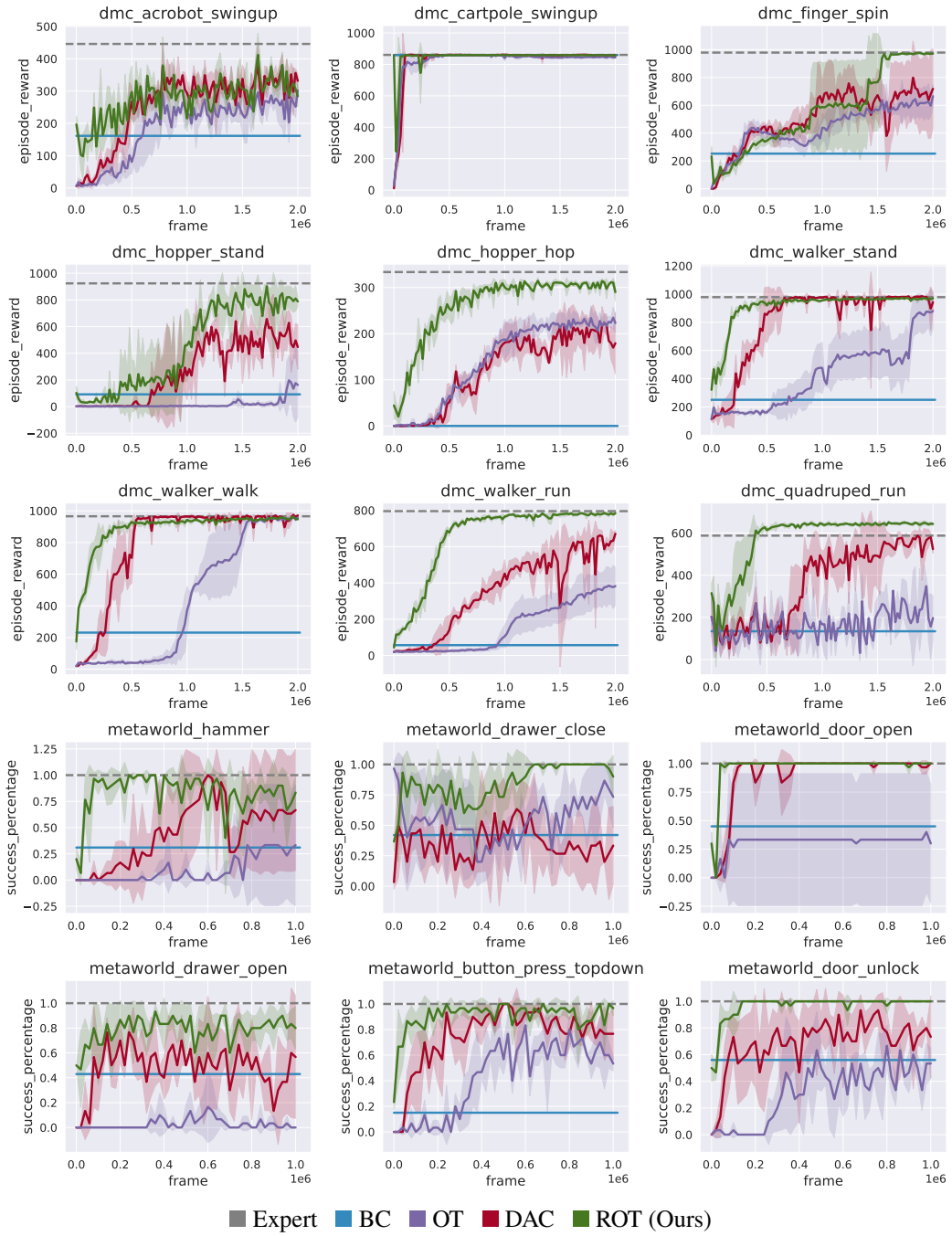


Figure 9: State-based continuous control learning on DMC and Meta-World tasks. We notice that ROT is significantly more sample efficient compared to prior work.

from scratch with BC regularization (OT+BC Reg.). We observe that pretraining the IRL policy (BC+OT) does not provide a significant difference without regularization. This can be attributed to the 'forgetting behavior' of pre-trained policies, studied in Nair et al. [9]. Interestingly, we see that even without BC pretraining, keeping the policy close to a behavior distribution (OT+BC Reg.) can yield improvements in efficiency over vanilla training from scratch. Our key takeaway from these experiments is that both pretraining and BC regularization are required to obtain sample-efficient imitation learning.

| Suite | Tasks | ROT | 2nd Best Model | Speedup Factor |
|---|---|---|---|---|
| DeepMind Control | Acrobot Swingup | 200k | 600k (OT) | 3 |
| | Cartpole Swingup | 100k | 350k (OT) | 3.5 |
| | Finger Spin | 20k | 700k (OT) | 35 |
| | Cheetah Run | 400k | 2M (DAC) | 5 |
| | Hopper Stand | 60k. | 750k (OT) | 12.5 |
| | Hopper Hop | 200k | >2M (DAC) | 10 |
| | Walker Stand | 80k | 400k (DAC) | 5 |
| | Walker Walk | 200k | 750k (DAC) | 3.75 |
| | Walker Run | 320k | >2M (OT) | 6.25 |
| | Quadruped Run | 400k | >2M (DAC) | 5 |
| OpenAI Robotics | Fetch Reach | 300k | 1.1M (DAC) | 3.67 |
| | Fetch Push | 1.1M | 600k (DAC) | 0.54 |
| | Fetch Pick and Place | 750k | >1.5M (OT) | 2 |
| Meta-World | Hammer | 200k | >1M (DAC) | 5 |
| | Drawer Close | 20k | >1M (OT) | 50 |
| | Drawer Open | >1M | >1M (OT) | 1 |
| | Door Open | 400k | >1M (OT) | 2.5 |
| | Bin Picking | 700k | >1M (OT) | 1.43 |
| | Button Press Topdown | >1M | >1M (OT) | 1 |
| | Door Unlock | 1M | >1M (OT) | 1 |

Table 3: Task-wise comparison between environment steps required to reach 90% of expert performance for pixel-based ROT compared to the strongest baseline for each task.

**Choice of IRL method** In ROT, we build on OT-based IRL instead of adversarial IRL. This is because adversarial IRL methods require iterative reward learning, which produces a highly non-stationary reward function for policy optimization. In Fig. 13, we compare ROT with adversarial IRL methods that use our pretraining and adaptive BC regularization technique (RDAC). We find that our soft Q-filtering method does improve prior state-of-the-art adversarial IRL (RDAC vs. DAC in Fig. 13). However, our OT-based approach (ROT) is more stable and on average leads to more efficient learning.

**Choice of Q-filtering method** In ROT, we adopt a soft Q-filtering method as opposed to the hard assignment strategy proposed by Nair et al. [18]. Fig. 14 shows a comparison between the performance of soft Q-filtering and hard Q-filtering. We observe that though the two strategies have comparable performance in most cases, soft Q-filtering exhibits better sample efficiency and more stable training in some tasks. This justifies our choice of opting for soft Q-filtering as opposed to hard assignment.

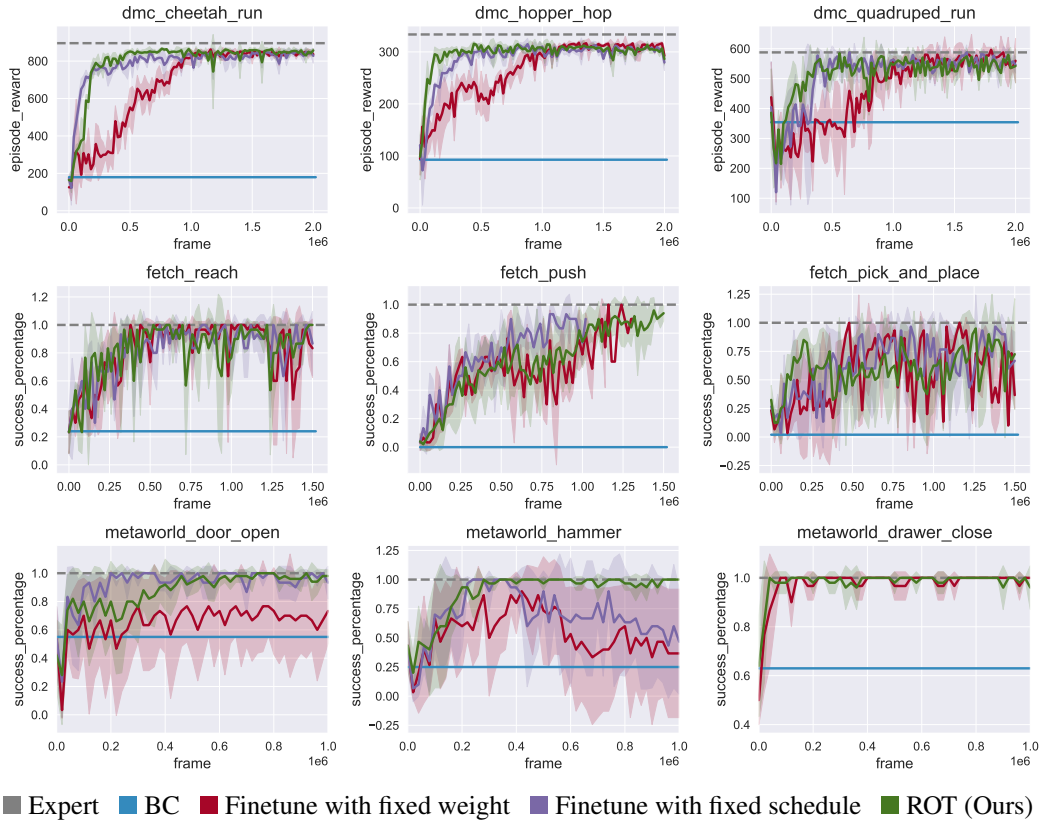


Figure 10: Pixel-based ablation analysis on the effect of varying BC regularization schemes. We observe that our adaptive soft-Q filtering regularization is more stable compared to prior hand-tuned regularization schemes.

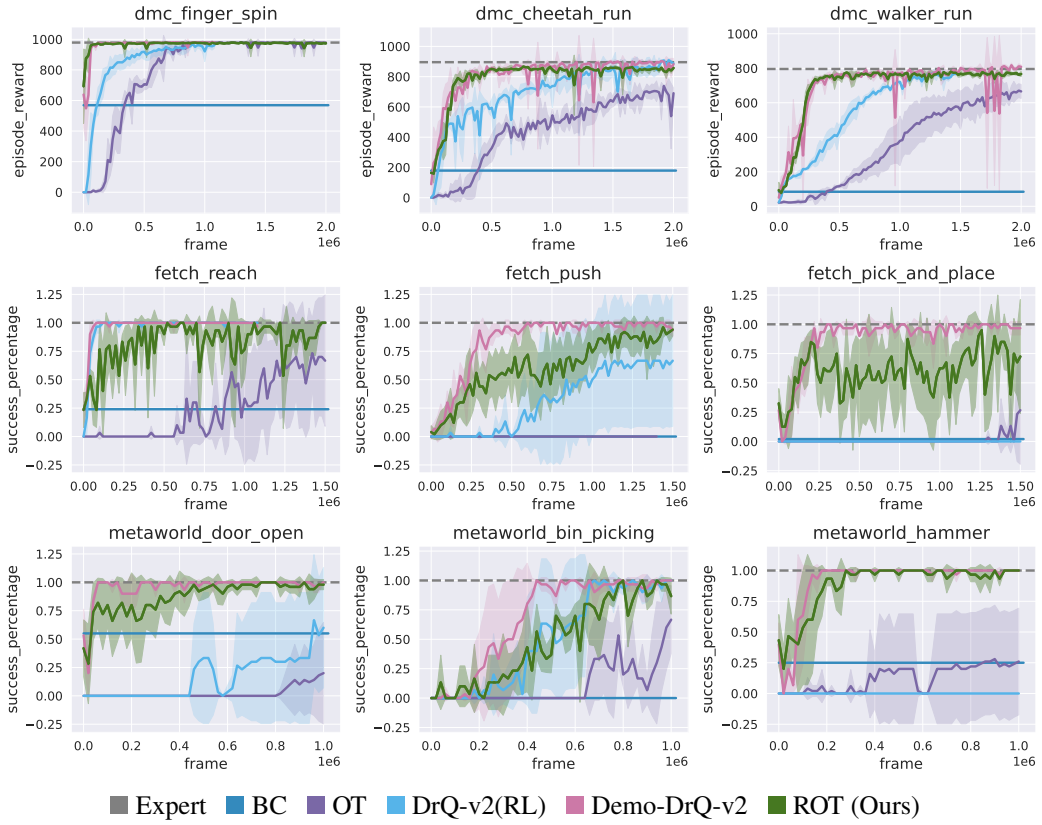


Figure 11: Pixel-based ablation analysis on the performance comparison of ROT against DrQ-v2, a reward-based RL method. Here we see that ROT can outperform plain RL that requires explicit task-reward. However, we also observe that this RL method combined with our regularization scheme provides strong results.

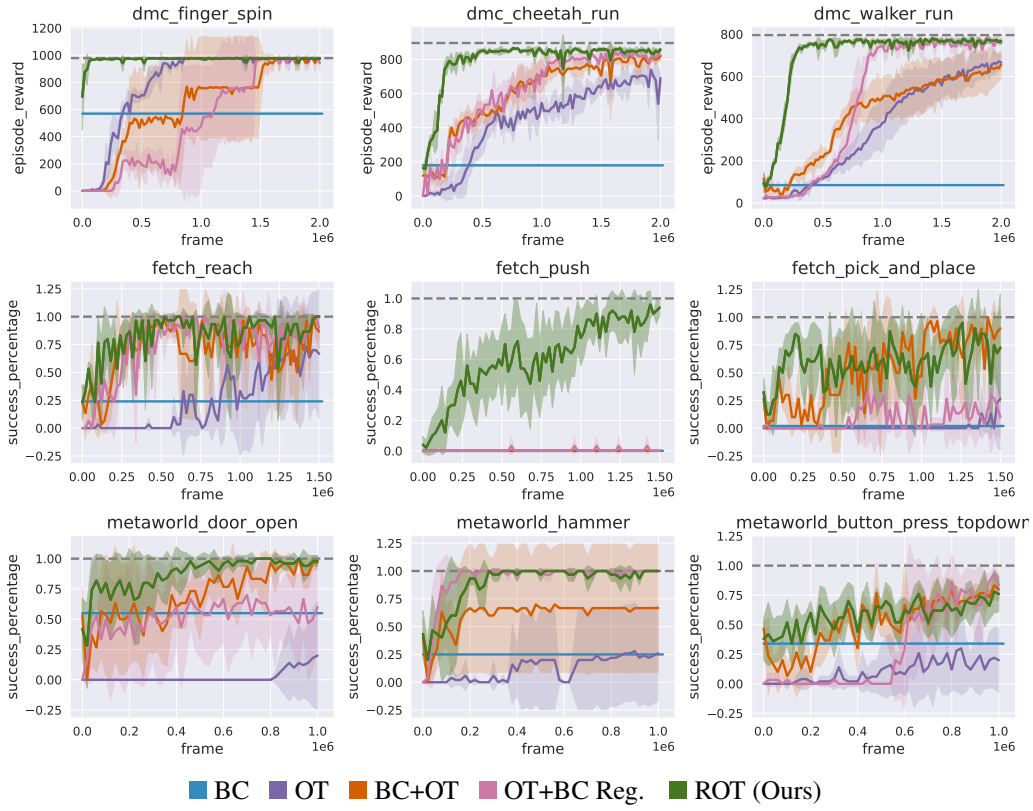


Figure 12: Pixel-based ablation analysis on the importance of pretraining and regularizing the IRL policy. The key takeaway from these experiments is that both pretraining and BC regularization are required to obtain sample-efficient imitation learning.

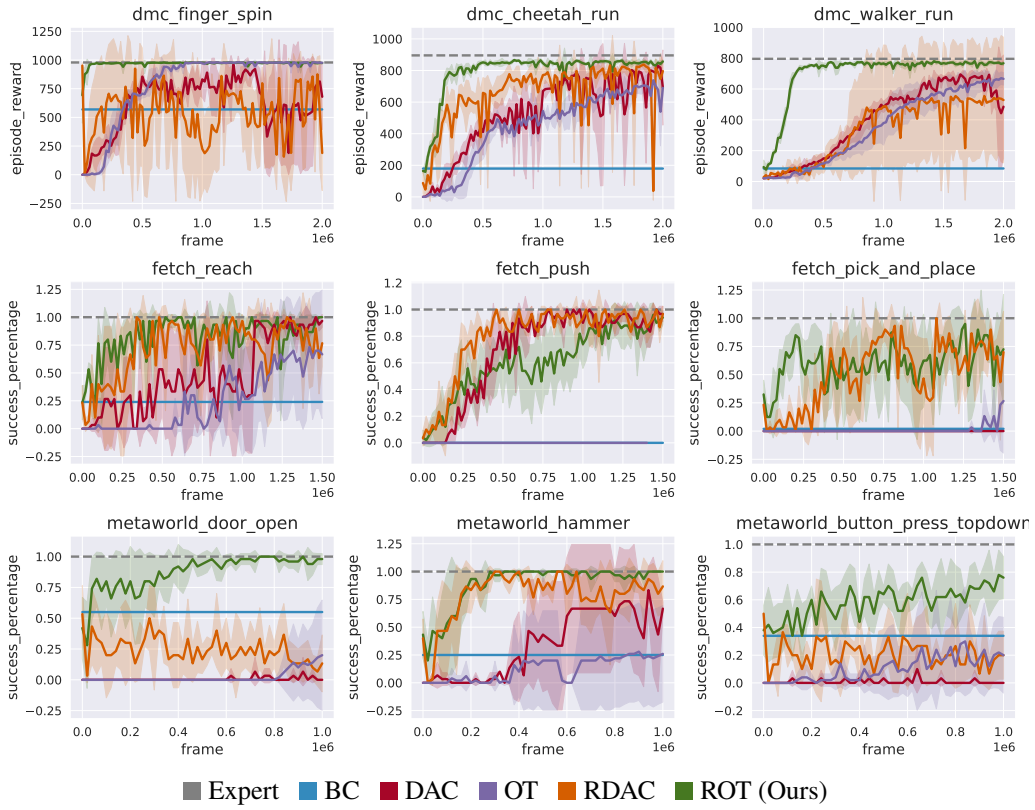


Figure 13: Pixel-based ablation analysis on the choice of base IRL method. We find that although adversarial methods benefit from regularized BC, the gains seen are smaller compared to ROT.

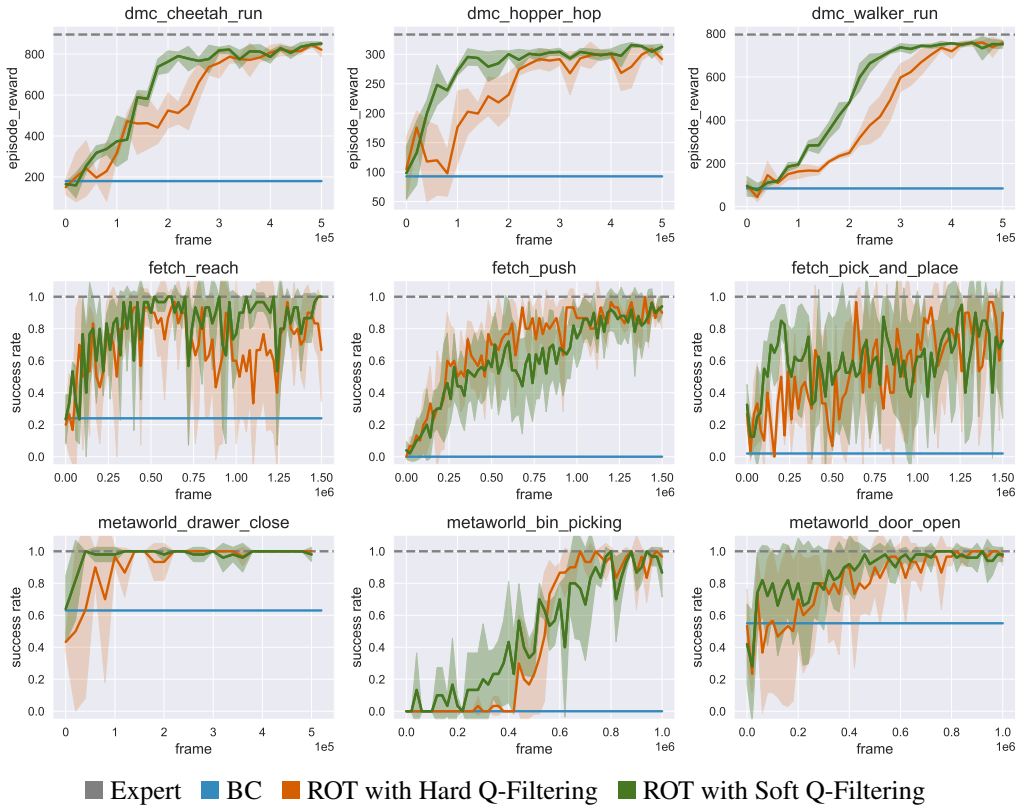


Figure 14: Pixel-based ablation analysis on the choice of Q-filtering method. We find that although the two strategies have comparable performance in most cases, soft Q-filtering exhibits better sample efficiency and more stable training in some tasks.