A General Pipeline for Autonomous Robotic Reinforcement Learning

Archit Sharma*, **Ahmed M. Ahmed***, **Rehaan Ahmed, Chelsea Finn**
Stanford University
{architsh, ahmedah, rehaan, cbfinn}@stanford.edu

Abstract: Learning via trial-and-error, reinforcement learning (RL) can enable robotic agents to learn robust policies. However, practical realizations of RL algorithms require extensive human supervision: primarily for engineering reward functions and repeated resetting of the environment between episodes of interactions. In this work, we propose a general pipeline requiring minimal human supervision throughout training: starting with a small set of expert demonstrations, the agent autonomously practices the task by learning to both *do* and *undo* the task, simultaneously inferring the reward function from the provided demonstrations as well. We validate our proposed pipeline on EARL, a non-episodic RL benchmark from visual inputs with minimal environment resets. We demonstrate our pipeline on a real-world Franka Panda robot arm, where we find that autonomous training improves the robustness of a policy by 50% compared to behavior cloning on a grasping task.

Keywords: reinforcement learning, autonomous systems, inverse RL

1 Introduction

Being robustly performant in situations that a robot may practically encounter is a prerequisite for robotics to become useful in the real world. Imitation learning methods such as behavior cloning (BC), though competent, do not generalize well when the agent goes outside its training distribution [1]. More general robots encounter broader set of states, anticipating and providing expert demonstrations becomes increasingly expensive and untenable. On the other hand, reinforcement learning can iteratively learn by trial-and-error, leading to more widely robust policies. However, practical instantiations of RL on robotic setups require extensive supervision for engineering task-specific *reward functions* and *repeatedly resetting* the environments between episodes. Enabling robots to train autonomously would allow robots to train and improve longer leading to more robust policies, and make RL algorithms more broadly applicable.

Autonomous operation (i.e. minimal human supervision for resetting environments) is an understated requirement for successful applications of RL to robot learning [2, 3, 4, 5, 6, 7]. Unfortunately, these prior works design environments and/or reset mechanisms to enable autonomous operation that is time-intensive, task-specific and often brittle. Zhu et al. [8] outline a set of conditions for a general autonomous robotic system: no state estimation, minimal/no reward engineering and no repeated interventions for resetting environments. They introduce a perturbation controller to randomly explore the environment and continue improving the policy autonomously. However, the random nature of the perturbation controller does not scale well as the size of state-action increases. Gupta et al. [9] demonstrate an autonomous system to learn a high-dimensional dexterous manipulation task by leveraging multi-task RL. However, the setup require state-estimation, reward engineering and a manually designed task-graph.

*These authors contributed equally to this work

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

In addition to the requirements laid down in Zhu et al. [8], we require that human supervision for an autonomous robotic system to be *front-loaded*, i.e., before the robot begins training. This enables the robot to train and improve without requiring further supervision. In view of these requirements, we propose the following general pipeline: a human collects a small set of demonstrations prior to training, which is used for both *task-specification* and *autonomous policy learning* directly from high-dimensional visual observations. Why demonstrations? Demonstrations can allow robotic systems to scale to larger state-action spaces, alleviating exploration challenges [10] while allowing task-specification via inverse RL [11, 12, 13, 14]. Additionally, demonstrations can also accelerate autonomous policy learning via MEDAL [15, 16], where a backward policy learns to match the distribution of states visited by an expert, such that the forward policy can learn to solve the task from a wide set of states. The complete pipeline and how it works from visual inputs is detailed in Section 3. We evaluate our experiments on EARL [17], a simulated non-episodic RL benchmark, where we validate the performance of our proposed pipeline. In the real world, we evaluate our proposed pipeline on the task of grasping a cube using Franka Panda robot arm, directly from visual inputs. We observe that our pipeline learns a policy that improves the grasping robustness by over 50% compared to behavior cloning.

2 Preliminaries

The formal problem setup for autonomous RL can be found in Sharma et al. [17]. Briefly, the agent operates in Markov Decision Process $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \rho_0)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ denotes the transition dynamics of the environment, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as the reward function, and ρ_0 denotes the initial state distribution. The challenge is to learn a competent and robust policy in a non-episodic environment, i.e., the environment does **NOT** reset to a state $s_0 \sim \rho_0$ repeatedly after a fixed number of steps, as is the case in episodic RL.

Our proposed pipeline extensively uses implicit distribution matching via adversarial learning, as popularized by generative adversarial networks [18]. Adversarial learning enables us to match $p_\theta(\cdot)$ to a target distribution $p^*(\cdot)$ that can only be sampled to generate a dataset $\{x_i \in \mathcal{X}\}_{i=0}^N$. The problem of minimizing the Jensen-Shannon divergence $\text{JS}(p_\theta \;||\; p^*)$ can be written as a minimax optimization: $\min_{p_\theta} \max_D \mathbb{E}_{x \sim p_\theta} [\log(1 - D(x))] + \mathbb{E}_{x \sim p^*} [\log D(x)]$, where $D : \mathcal{X} \mapsto (0, 1)$ is a discriminator trying to classify real samples from p^* from fake ones generated by p_θ, whereas p_θ is trying to fool the discriminator into predicting the samples are from p^*. The idea has been extensively used in reinforcement learning: adversarial imitation learning [11, 19], reward inference from goals [12, 13] and autonomous learning [16]. In context of RL, p^* is the target distribution of states $\subset \mathcal{S}$ and p_θ is the stationary distribution induced by a policy π_θ. Noting that $\min_{p_\theta} \mathbb{E}_{s \sim p_\theta} [\log(1 - D(s))] = \max_{\pi_\theta} -\mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi_\theta(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)} [\sum_{t=0}^{\infty} \gamma^t \log(1 - D(s_{t+1}))]$, the optimization with respect to the generator can be re-written as a RL problem with $-\log(1-D(\cdot))$ as the reward function. The classification for D is to discriminate between the states sampled from the target distribution p^* from the states visited by the policy π_θ. In this work, we will be building on VICE [13, 20] and MEDAL [16], where VICE uses a set of goal images as the target distribution and MEDAL uses the set of the states visited by an expert policy π^* as the target distribution.

3 Pipeline for Autonomous RL

We have the following desiderata for an autonomous learning robotic system [8]: No state estimation, i.e., the system should be able to learn from high-dimensional visual inputs, minimal/no reward engineering and minimal human supervision for resetting environments between trials. Additionally, our system will assume access to a small set of expert demonstrations, collected prior to training. In view of these considerations, we propose the following pipeline:

High-dimensional visual observations. We use convolutional neural networks for encoding the observations, borrowing the architecture from DrQ-v2 [21]. We follow the prescribed data augmen-

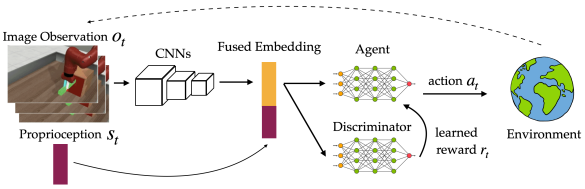

Figure 1: Overview of the architecture used by the agents. Image observations are embed by a CNN into a low-dim space and passed to the policy, critic and discriminator after concatenating with proprioception. Discriminator outputs the reward for the agent, while being trained to distinguish between states visited by the agent and target states. The forward and backward agents differ only in the choice of target distributions.

Algorithm 1: Training Loop Overview

initialize $\mathbb{F}, \mathbb{B}$; // forward, backward agents
$s \sim \rho_0$; $\mathbb{A} \leftarrow \mathbb{F}$; // initialize env + agent
while *not done* **do**
  $a \sim \mathbb{A}.\texttt{act}(s)$; $s' \sim \mathcal{T}(\cdot \mid s, a)$;
  $\mathbb{A}.\texttt{update\_buffer}(\{s, a, s'\})$;
  $\mathbb{A}.\texttt{update\_discriminator}()$;
  $\mathbb{A}.\texttt{update\_agent}()$;
  // switch agents after a fixed interval
  if *switch* **then**
    $\mathbb{A} \leftarrow \texttt{switch}(\mathbb{A}, (\mathbb{F}, \mathbb{B}))$
  $s \leftarrow s'$;

tations of random cropping and shifts when embedding visual observations. The embeddings are fused with proprioceptive information from the robot before being passed to the agent.

Minimal reward engineering. We learn the reward function in line with the minimax optimization in Section 2. We learn a discriminator, with a convolutional backbone (including data augmentations) and late fusion embeddings similar to the one described earlier. The positive observations come from the target distribution and the negative observations are collected by the policy while interacting with the environment. The discriminator is trained with a binary cross-entropy loss to solve this classification problem. To regularize the discriminator, we use mixup [22] on embeddings generated *after* fusion with proprioception, and spectral norm [23]. Overall, only a small set of samples from the target distribution is required to learn the reward function.

Minimal human supervision for resets. To reduce the requirement of resetting environments, we build on MEDAL [16]. MEDAL instantiates a forward agent π_f that learns how to solve the task and a backward agent π_b that learns how to undo the task. Both the agents solve the minimax optimization described in Section 2, with the only difference being the target distribution. Specifically, the forward agent optimizes the VICE reward, where the target distribution is created by choosing the last H_{goal} steps of the every expert demonstration (interpreted as the goal distribution of the task). The backward agent optimizes the MEDAL reward, where the objective for the policy is to match the state-distribution of the expert agent. This enables the agent to re-try the task from a wide variety of relevant states. Therefore, the target distribution are *all* the states in expert demonstrations. Both the forward and backward agent are trained using REDQ [24]. Given the access to demonstrations, we leverage techniques from Nair et al. [10], specifically oversampling transitions from expert data and regularizing policy using the behavior cloning loss.

An overview of our proposed learning pipeline and a high-level pseudocode can be found in Figure 1. The exact hyperparameters can be found in Appendix A.

4 Experiments

In this section, we empirically analyze our proposal: (a) we evaluate the our proposed modifications to MEDAL on the EARL benchmark [17] and (b) we validate our proposed pipeline in the real-world on a Franka Panda robot on a cube grasping task.

Simulated experiments. For our simulated environments, we evaluate MEDAL on three tasks from the EARL benchmark: `Tabletop Organization` is a manipulation task where a simplified gripper has to relocate the mug to one of four goal locations, `Door Closing` tasks a sawyer robot with closing the door and `Peg Insertion` requires the sawyer robot to pick up the peg from different positions and insert it into the hole. All the environments are sparse reward environments that come with 10 forward demonstrations and 10 backward demonstrations. EARL environments are non-episodic in nature, i.e the environment is reset infrequently (every 10K steps instead of every 100 steps), and the policy is evaluated by averaging the return over 10 trials starting from the initial state

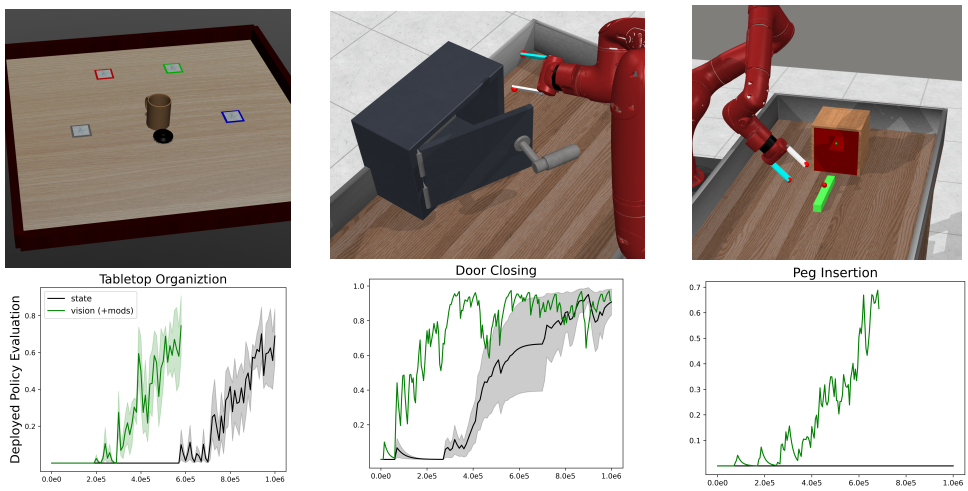

Figure 2: Deployed policy evaluation on the EARL benchmark. We evaluate our proposed modifications to MEDAL, including working from visual observations, oversampling transitions from expert demonstrations and BC regularization for the policy loss. Our proposed pipeline and modifications (labeled *vision (+mods)*, in green) result in substantial improvements to sample-efficiency of MEDAL (labeled *state*, in black).

distribution, every 5K training steps (*Deployed Policy Evaluation*). We modify the environments to return visual observations instead of state information to test our pipeline. The non-episodic nature of the environment requires the agent to *reset* the environment itself to enable autonomous practicing.

We evaluate the following changes to MEDAL: switching from SAC to REDQ [24] as the base algorithm, learning from images instead of state, reward learned by VICE, using BC regularization on the policy loss and oversampling transitions from demonstrations when training the agents [10]. We observe in Figure 2 that our proposed pipeline substantially improves the performance over MEDAL, getting to 80% success rate in 600K steps on `Tabletop Organization` and `Door Closing`, and 70% on `Peg Insertion` where MEDAL has a performance of 0%.

Franka Panda cube grasping. In a real world experiment, we task the Franka robot arm with grasping a red cube. The setup includes a wrist camera [25] and fixed third person camera for image observations, and the end-effector position and gripper width as proprioceptive information (Figure 3). We collect 50 expert demonstrations, with randomized orientations, with some noise around the center of the arena. We run the setup autonomously for 300K samples, amounting to a wall-clock time of approximately 1 day. Such a large data collection is only because the system was learning autonomously, not requiring any explicit human monitoring.

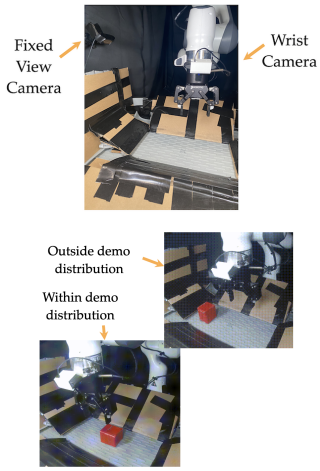

Figure 3: (*top*) An overview of the robot learning setup. (*bottom*) Samples showing test states in-and-outside demo distribution

We compare the performance of the policy learned after autonomous learning to that learned by behavior cloning (BC). We evaluate the learned policies on grasping the cube starting from (a) *in-distribution* (ID) states and (b) *out-of-distribution* (OOD) states where the position and orientation is varied outside the demo distribution. For ID performance, both the polices get 100% success rate on 20 trials. However, on OOD states, BC learned policy gets a 20% success rate whereas autonomously learned policy gets a success rate of 75% on 20 trials. This experiment provides evidence for how reinforcement learning when learning autonomously can train for long times, and learn a more robust policy in the process.

References

[1] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2010. URL https://arxiv.org/abs/1011.0686.

[2] F. Ebert, S. Dasari, A. X. Lee, S. Levine, and C. Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 983–993. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/ebert18a.html.

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

[4] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost, 2018. URL https://arxiv.org/abs/1810.06045.

[5] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation, 2019. URL https://arxiv.org/abs/1909.11652.

[6] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics, 2019. URL https://arxiv.org/abs/1903.11239.

[7] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale, 2021. URL https://arxiv.org/abs/2104.08212.

[8] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine. The ingredients of real-world robotic reinforcement learning, 2020. URL https://arxiv.org/abs/2004.12570.

[9] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention, 2021. URL https://arxiv.org/abs/2104.11203.

[10] 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.

[11] J. Ho and S. Ermon. Generative adversarial imitation learning, 2016. URL https://arxiv.org/abs/1606.03476.

[12] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine. Variational inverse control with events: A general framework for data-driven reward definition, 2018. URL https://arxiv.org/abs/1805.11686.

[13] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine. End-to-end robotic reinforcement learning without reward engineering. *ArXiv*, abs/1904.07854, 2019.

[14] S. Haldar, V. Mathur, D. Yarats, and L. Pinto. Watch and match: Supercharging imitation with regularized optimal transport, 2022. URL https://arxiv.org/abs/2206.15469.

[15] A. Sharma, A. Gupta, S. Levine, K. Hausman, and C. Finn. Autonomous reinforcement learning via subgoal curricula, 2021. URL https://arxiv.org/abs/2107.12931.

[16] A. Sharma, R. Ahmad, and C. Finn. A state-distribution matching approach to non-episodic reinforcement learning, 2022. URL https://arxiv.org/abs/2205.05212.

[17] A. Sharma, K. Xu, N. Sardana, A. Gupta, K. Hausman, S. Levine, and C. Finn. Autonomous reinforcement learning: Formalism and benchmarking, 2021. URL https://arxiv.org/abs/2112.09605.

[18] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[19] 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, 2018. URL https://arxiv.org/abs/1809.02925.

[20] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine. Variational inverse control with events: A general framework for data-driven reward definition, 2018. URL https://arxiv.org/abs/1805.11686.

[21] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning, 2021. URL https://arxiv.org/abs/2107.09645.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization, 2017. URL https://arxiv.org/abs/1710.09412.

[23] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks, 2018. URL https://arxiv.org/abs/1802.05957.

[24] X. Chen, C. Wang, Z. Zhou, and K. Ross. Randomized ensembled double q-learning: Learning fast without a model, 2021. URL https://arxiv.org/abs/2101.05982.

[25] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn. Vision-based manipulators need to also see from their hands, 2022. URL https://arxiv.org/abs/2203.12677.

A Appendix

In this section, we provide additional details of our experimental setup and hyperparameters. As mentioned earlier, we use the encoder architecture and augmentations from DrQ-v2 [21]. We share the encoder for the actor, critic, and VICE discriminator but use a separate encoder for the MEDAL discriminator. To stabilize training for all discriminators we use mixup and spectral norm regularization [22, 23]. We also utilize REDQ to improve sample complexity with an ensemble of 5 critics and a UTD ratio of 1 for all tasks [24].

A.1 Simulation Experimental Details

For simulation tasks, we provide 10 updates to the forward discriminator every 500 steps, and to the backward discriminator every 100 steps. We also measure performance by evaluating the agent for 10 episodes at an interval of 10000 steps, and make use of oversampling, i.e. sampling state-action pairs from the demo buffer which replace a subset of the batch sampled from the replay buffer before updating the critic. More details can be found in the table below.

Name	Description	Value
Seed Frames	Number of random actions taken before training	10,000
Oversampling Count	Sampled transitions from demos for critic	16
Episode Length	Number of timesteps per rollout	100
Entropy Bonus	Entropy loss coefficient	0.0
Learning Rate	Learning rate for Adam	3×10^{-4}
Optimizer	Optimizer for agent and discriminator	Adam
Frame Stack	Frame stack X Env frames	1
Hidden Dim	Hidden dimension for shared encoder	256
Agent Batch Size	Number of samples per minibatch	256
Discriminator Batch Size	Number of samples per minibatch for discrimators	512
γ	Discount factor	0.99
τ	update weight for target critic	0.005

Table 1: Simulation parameters

A.2 Robot Experimental Details

For the robot experiments, we mostly adhere to the parameters used for simulation in the table with a few modifications. We reduce the number of seed frames from 10,000 to 2000 as the wall-clock time per step for the robot is much higher than in simulation, and we find that we are still able to produce strong results. We also only provide **one** update for the forward discriminator every 1000 steps, and also one for the backward discriminator every 500 steps. We also add a wrist camera to improve overall performance [25], and provide **only** the wrist-camera view to both discriminators as we empirically find that this serves as an additional regularisation. Additionally, we increase the oversampling count to 32 and make use of a regularization schedule that provides an auxiliary BC loss for the actor which decays linearly from 1 to 0.0 over the first 10,000 steps for both discriminators. Finally, we provide no proprioceptive information for the VICE discriminator, but we give MEDAL discriminator the proprioceptive information, as it needs a stronger notion of the robot's localization to adequately reset to a varied number of initial positions for improved robustness.