XQCfD: Accelerating Fast Actor-Critic Algorithms with Prior Data and Prior Policies

Daniel Palenicek[*1,2] **Florian Vogt**[*3] **Joe Watson**[*4] **Ingmar Posner**[4]
Danica Kragic[3] **Jan Peters**[1,2,5,6]
[1]Technical University of Darmstadt [2]hessian.AI [3]KTH Royal Institute of Technology
[4]University of Oxford [5]German Research Center for AI (DFKI)
[6]Robotics Institute Germany (RIG)
{palenicek,joe}@robot-learning.de

Abstract

For reinforcement learning in the real world, online exploration is expensive. A common practice in robotic reinforcement learning is to incorporate additional data to improve sample efficiency. Expert demonstration data is often crucial for solving hard exploration tasks with sparse rewards. While prior data is used to augment experience and pre-train models, we show that the design of existing algorithms fails to achieve the sample efficiency that is possible in this setting due to a failure to use pretrained policies effectively. We propose `XQCfD`, which extends the sample-efficient `XQC` actor-critic to learn from demonstrations, using augmented replay buffers, pre-trained policies and stationary policy architectures, designed to avoid rapidly 'unlearning' the strong initial policy like prior works. We show our stationary network architecture enables policy improvement out-of-distribution better than standard network architectures due to its higher entropy predictions. `XQCfD` achieves state of the art performance across a range of complex manipulation tasks with sparse rewards from the popular Adroit, Robomimic and MimicGen benchmarks — notably, with a low update-to-data ratio and no ensemble networks.

1 Introduction

Reinforcement learning (RL) provides a general framework for sequential decision-making, but high sample complexity remains the central barrier to real-world deployment [39]. Model-free deep actor-critic methods have made remarkable strides in sample efficiency through careful algorithmic and architectural choices: batch normalization, weight normalization, categorical critics, high update-to-data ratios, culminating in methods such as `CrossQ` [4] and `XQC` [25, 24]. In many practical settings, prior data is readily available before online interaction begins: expert demonstrations, previously collected offline datasets, or trajectories from related tasks. Leveraging such data promises to accelerate online learning significantly. Several research threads have combined RL with this additional data: *offline-to-online RL* pre-trains a policy offline and finetunes it online; *RL from demonstrations* (RLfD, [41, 30]) incorporates expert data into the agent's experience; *RL from prior data*, e.g., RLPD [2], mixes sub-optimal offline and online data in the agent's replay buffer. *Imitation-bootstrapping* [12] interleaves on-policy and behavioral cloning actions during data collection to incorporate prior knowledge.

A recurring limitation in RLfD is failing to properly leverage the performance of *behavioral cloning* (BC [28]) at initialization [2, 12]. Many current RLfD algorithms require millions of

[*]denotes equal contribution.

ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning

interactions to match BC performance, even though they eventually surpass it. This phenomenon occurs because the pre-trained policy is rapidly unlearned due to the learning dynamics of deep actor-critic algorithms and their use of function approximation. The BC policy is not optimal for the randomly-initialized critic, and therefore the BC policy is rapidly unlearned in favor of a policy that is. This phenomenon has been observed in inverse reinforcement learning, where it is more severe due to the additional learned reward [23, 43].

We propose `XQCfD` (`XQC` from demonstrations), which extends the `XQC` actor-critic implementation with a set of algorithmic changes to achieve state-of-the art sample efficiency for RLfD:

- We perform policy pretraining on the demonstration data to start learning at the BC-level of performance, rather than learn from scratch.
- We also perform critic pretraining to ensure the initial actor and critic are coherent.
- To mitigate unlearning due to stochastic optimization, we use KL regularization between the policy and BC policy as an additional regularizer, instead of a maximum entropy one.
- To bridge KL and maximum entropy regularization, we adopt a 'stationary' parametric policies which returns a constant, maximum entropy action distribution when out of distribution. This property enables exploration and policy improvement in states outside of the expert demonstration data, so the KL regularization does not limit policy improvement over BC.

While none of these design decisions are independently novel, this combination has not been used for RLfD, and we show empirically across a range of sparse manipulation tasks that this approach prevents BC unlearning and therefore enables dramatic sample efficiency compared to baselines when combined with XQC. In summary, our contributions are

1. We combine the actor-critic algorithm `XQC` with prior data and pre-trained policies for a sample-efficient RLfD algorithm, highlighting the crucial implementation details.
2. We motivate and demonstrate the utility of stationary network architectures in RLfD.
3. We provide thorough ablation experiments to validate our design decisions.

2 Related Work

Reinforcement learning from demonstrations or prior data is a long-established idea, especially for domains where exploration is challenging and expensive. Currently, using prior data comes in three distinct settings, based on how the data is used and what is pre-trained.

Finetuning behavioral cloning with reinforcement learning. For challenging domains such as real-world robotics and strategy games, a practical solution for sample efficiency is to finetune BC policies [35, 26, 14, 16, 37]. More modern practices combine an on- or off-policy deep RL algorithm with an additional BC-based loss or policy regularization in the actor objective to prevent unlearning [30, 33]. We build on this work by incorporating a more sample-efficient off-policy algorithm that is able to finetune a BC policy without a significant performance drop, which is observed when using randomly initialized critics, e.g., the RLFT baseline in [12].

Online reinforcement learning with additional data. With the advent of replay buffers in deep reinforcement learning, prior data can be easily integrated by merely augmenting the replay buffer. Algorithm design decisions concern whether the replay buffer is simply augmented or multiplexed[2] [41, 2]. Imitation-bootstrapped RL [12] improves sample efficiency by incorporating BC action during interactions, and identifying dropout in the policy as an effective implementation detail and `REDQ` [6] as an effective RL algorithm in this setting. However, by not utilizing BC initialization effectively, buffer-based methods forfeit potentially significant sample efficiency gains, which we show in our experimental results.

Offline-to-online reinforcement learning. A related setting where the prior data is suboptimal considers continuing offline RL with online interactions and learning [21]. This setting is challenging due to the difficulty of performing continual learning with function approximation with severe performance degradation, characterized as 'catastrophic forgetting' or a lack of network 'plasticity'.

[2] Multiplexing refers to combining samples from several replay buffers within a single minibatch.

Solutions include BC regularization [44], careful sampling of offline and online samples from the replay buffer [15] and calibrating the Q values during offline learning to mitigate policy unlearning during online interactions [22]. Our RLfD setting corresponds to offline-to-online RL with expert offline data, and we do not consider using suboptimal data as we desire strong initial BC policies.

Kullback-Leibler (KL)-regularized RL. While `XQC` builds on `SAC` (soft actor-critic, [10]), `XQCfD` uses a KL-regularized realization rather than a maximum entropy one, where the BC policy is a prior. Siegel et al. [36] use the same regularization for offline RL, using `MPO` [1] as the actor-critic algorithm. Rudner et al. [33] use this regularization for RLfD with `SAC`, but use a nonparametric Gaussian process as the BC policy and a standard network architecture for the RL policy and minimizes their KL during pre-training. Watson et al. [43] use this formulation with `SAC`, but in an inverse RL setting.

3 Sample-Efficient Actor-Critic with Prior Data and Prior Policies

We consider the standard reinforcement learning setting, using a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ are the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each timestep h, an agent observes state $\boldsymbol{s}_h \in \mathcal{S}$, selects action $\boldsymbol{a}_h \sim \pi(\cdot \mid \boldsymbol{s}_h)$ according to a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, transitions to $\boldsymbol{s}_{h+1} \sim P(\cdot \mid \boldsymbol{s}_h, \boldsymbol{a}_h)$, and receives reward $r_h = r(\boldsymbol{s}_h, \boldsymbol{a}_h)$. The objective is to find a policy $\pi^\star$ that maximizes the expected discounted return $\mathcal{J}(\pi) = \mathbb{E}\big[\sum_{h=0}^{\infty} \gamma^h r_h\big]$. A central quantity in reinforcement learning is the state-action value function $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}\big[\sum_{k=0}^{\infty} \gamma^k r_{h+k} \mid \boldsymbol{s}_h = \boldsymbol{s},\, \boldsymbol{a}_h = \boldsymbol{a}\big]$, which satisfies the Bellman equation $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \gamma\, \mathbb{E}_{\boldsymbol{s}' \sim P(\cdot \mid \boldsymbol{s}, \boldsymbol{a}),\, \boldsymbol{a}' \sim \pi(\cdot \mid \boldsymbol{s}')}[Q^\pi(\boldsymbol{s}', \boldsymbol{a}')]$. Actor-critic algorithms maintain two function approximators: an actor $\pi_{\boldsymbol{\phi}}$ parameterized by $\boldsymbol{\phi}$ that represents the policy, and a critic $Q_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ that estimates the value function. Deep actor-critic algorithms maintain a replay buffer $\mathcal{B}$ of past interactions. When learning from demonstrations, we also have access to a dataset of demonstrations $\mathcal{D}$ of state-action trajectories capturing expert behavior.

Given a sample-efficient RL algorithm, incorporating prior expert data is a way of alleviating the pressure to explore and therefore achieve greater sample efficiency. While prior RLfD works adopt effective RL algorithms with Q ensembles, layer norm, and high update-to-data ratios, the recently proposed `XQC` [25] has matched or outperformed these architectures w.r.t. parameter count, sample efficiency and wall-clock speed by combining batch norm [13], weight norm [17] and a distributional critic [3]. This combination ensures the critic's 'effective' learning rate [40] does not decrease and the critic's loss landscape is smooth and well-conditioned [34, 9], which enables sample-efficient learning without any great computational overhead [25]. To leverage prior data, we multiplex the replay buffers during online learning and pre-train the policy using BC. The open research question concerns the tension between preventing unlearning and facilitating finetuning. Can we finetune a parametric policy without unlearning the strong initialization? To tackle this challenge, we use a policy architecture that has maximum entropy action predictions, as we adopt the entropy-regularized RL formulation, which regularizes the policy against a prior distribution using the KL divergence. While online RL uses a uniform action prior for exploration[3] with temperature $\alpha \geq 0$, in the RLfD setting we replace this uninformative prior with the pre-trained BC policy $\pi_{\text{BC}}(\boldsymbol{a} \mid \boldsymbol{s})$, which also initializes $\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})$, resulting in actor loss

$$\boldsymbol{\phi}_\star = \arg\max_{\boldsymbol{\phi}} \mathcal{L}_\pi(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{s} \sim \mathcal{B}}\left[\mathbb{E}_{\boldsymbol{a} \sim \pi_{\boldsymbol{\phi}}(\cdot \mid \boldsymbol{s})}[Q_{\boldsymbol{\theta}}(\boldsymbol{s}, \boldsymbol{a})] - \alpha\, \mathbb{D}_{\text{KL}}[\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s}) \mid\mid \pi_{\text{BC}}(\boldsymbol{a} \mid \boldsymbol{s})]\right]. \quad (1)$$

Adopting the KL divergence provides strong policy improvement guarantees. The solution to Equation 1 has monotonic improvement guarantees over the BC policy (Lemma 1).

Lemma 1. *Monotonic policy improvement [32]. Given a prior policy $p(\boldsymbol{a}|\boldsymbol{s})$, the pseudo-posterior policy update $\pi(\boldsymbol{a}|\boldsymbol{s})$ improves on the prior in a monotonic fashion for critic Q due to the change-of-measure inequality [8],*

$$\mathbb{E}_{\boldsymbol{a} \sim \pi(\cdot \mid \boldsymbol{s})}[Q(\boldsymbol{s}, \boldsymbol{a})] \geq \mathbb{E}_{\boldsymbol{a} \sim p(\cdot \mid \boldsymbol{s})}[Q(\boldsymbol{s}, \boldsymbol{a})]\ \forall \boldsymbol{s} \in \mathcal{S},\ \pi(\boldsymbol{a} \mid \boldsymbol{s}) \propto \exp\left(\frac{1}{\alpha} Q(\boldsymbol{s}, \boldsymbol{a})\right) p(\boldsymbol{a} \mid \boldsymbol{s}). \quad (2)$$

However, in moving our objective from maximum entropy regularization with a uniform prior to KL regularization with the BC prior, we may have compromised our deep reinforcement learning

[3] A maximum entropy objective is equivalent to a KL minimization objective against a uniform distribution.

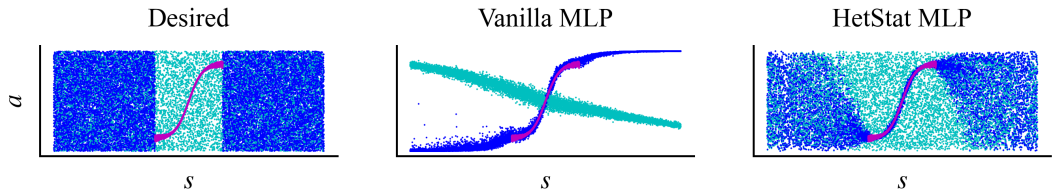

Figure 1: (Left) In the entropy-regularized RLfD setting, we desire a BC policy that fits the data (●) while reflecting a maximum entropy prior out of distribution. (Centre) The typical multi-layer perceptron (MLP) architecture used in deep actor-critic methods does not have this property. The out-of-distribution predictive behavior is undefined. Due to this behavior, KL regularization encourages the policy to stay close to suboptimal actions. (Right) By incorporating stationarity into the policy architecture, the policy is maximum entropy at initialization (●) and maximum entropy out-of-distribution after training (●). In practice, there is a smooth transition between modeling the data and maximum entropy prior due to the smoothness of the function approximation.

algorithm. In states outside of the demonstration distribution where we wish to perform online RL, $\pi_{\text{BC}}(\boldsymbol{a} \mid \boldsymbol{s})$ may be very different from the uniform prior $\mathcal{U}(\boldsymbol{a})$, especially as the predictive behavior of neural networks outside of the data distribution (OOD) is undefined, as illustrated in Figure 1. Therefore, during RL finetuning in out-of-distribution states, Equation 1 may encourage the agent to perform suboptimal actions rather than random exploration, and reducing α as a result also encourages unlearning the expert behavior. To fix this issue, we desire a policy network that predicts a uniform action distribution when out of distribution. Fortunately, this is possible by using policies with *stationary* feature spaces.

3.1 Maximum Entropy Policies using Stationary Features

To finetune the BC policy in out-of-distribution states with maximum entropy RL using Equation 1, we require a BC policy with the following characteristic,

$$\pi_{\text{BC}}(\boldsymbol{a} \mid \boldsymbol{s}) = \begin{cases} p_{\mathcal{D}}(\boldsymbol{a} \mid \boldsymbol{s}) \text{ if } \boldsymbol{s} \in \mathcal{D}, \text{ where } p_{\mathcal{D}} \text{ is a conditional generative model of } \mathcal{D}, \\ \mathcal{U}(\boldsymbol{a}), \text{ the uniform prior, otherwise.} \end{cases} \tag{3}$$

A consequence of this characteristic is that, in the absence of any demonstration data, we desire a policy architecture where $\pi_{\text{BC}}(\boldsymbol{a} \mid \boldsymbol{s}) = \mathcal{U}(\boldsymbol{a}) \;\forall\; \boldsymbol{s} \in \mathcal{S}$. Following soft actor-critic [10], we use a Gaussian policy with a tanh transformation to bound the support of the action distribution to match the uniform prior. Therefore, this property is approximated by a latent action $\boldsymbol{a} = \tanh(\boldsymbol{z})$ with fixed zero-mean Gaussian predictions. The desired out-of-distribution behavior described above can be achieved with a latent Gaussian model with stationary features $\boldsymbol{\varphi}$. Stationary features have constant self-similarity: the inner product $\varphi_{\boldsymbol{\theta}}(\boldsymbol{s}')^\top \varphi_{\boldsymbol{\theta}}(\boldsymbol{s})$ is a function of a distance metric between $\boldsymbol{s}'$ and $\boldsymbol{s}$, which means that the variance is a constant when $\boldsymbol{s}' = \boldsymbol{s}$, $p(z \mid \boldsymbol{s}) = \mathcal{N}(0, \sigma^2)$ as desired when $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ due to the predictive distribution of Gaussian models shown below. Our latent Gaussian policy is implemented using stationary last layer features, on top of MLP features, with Gaussian last layer weights $\boldsymbol{w}$ such that $p(\boldsymbol{z} \mid \boldsymbol{s}) = \int p(\boldsymbol{z} \mid \boldsymbol{s}, \boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}$,

$$z = \boldsymbol{w}^\top \boldsymbol{\varphi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \quad p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad p(z \mid \boldsymbol{s}) = \mathcal{N}(\boldsymbol{\mu}^\top \boldsymbol{\varphi}_{\boldsymbol{\theta}}(\boldsymbol{s}), \boldsymbol{\varphi}_{\boldsymbol{\theta}}(\boldsymbol{s})^\top \boldsymbol{\Sigma}\, \boldsymbol{\varphi}_{\boldsymbol{\theta}}(\boldsymbol{s})).$$

This predictive behavior is preserved for out-of-distribution states after training, subject to the smoothness of the features. Unfortunately, this stationary property is only achieved exactly when $\varphi(\boldsymbol{s})$ is an infinitely large feature space and $\varphi_{\boldsymbol{\theta}}(\boldsymbol{s})^\top \varphi_{\boldsymbol{\theta}}(\boldsymbol{s}')$ is expressed through its tractable kernel function $k(\boldsymbol{s}, \boldsymbol{s}')$ [31]. Fortunately, we can construct a *finite* approximate to these feature spaces using *random Fourier features* [29] $\boldsymbol{f}_k$ for kernel $k(\cdot, \cdot)$,

$$\boldsymbol{\varphi}_{\boldsymbol{\theta}, \boldsymbol{V}}(\boldsymbol{s}) = \boldsymbol{f}_k(\boldsymbol{V}\boldsymbol{s}), \quad V_{ij} \sim \mathcal{N}(0, 1), \tag{4}$$

where $\boldsymbol{f}_k$ are periodic functions required to approximate stationary kernel k and $\boldsymbol{V}$ is a fixed random weight matrix that constructs the random features with $\boldsymbol{f}_k$. In practice, we apply the random features to neural network features rather than $\boldsymbol{s}$ for greater function approximation capability, i.e., $\boldsymbol{f}_k(\boldsymbol{V}\varphi_{\text{NN}}(\boldsymbol{s}))$, which preserves stationarity [20]. We refer to this architecture as HetStat, as it is heteroscedastic[4] and stationary. HetStat policies were first proposed by Watson et al. [43] in an

[4]Heteroscedastic predictions have input-varying variance, whereas homoscedastic predictions have constant variance.

Algorithm 1: XQC from Demonstrations (`XQCfD`).

Data: Expert demonstrations $\mathcal{D}$, temperature α, parametric policy class $\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})$, parametric critic class $Q_{\boldsymbol{\theta}}(\boldsymbol{s}, a)$, total steps T, update-to-data ratio N, policy delay D
Result: $\pi_{\boldsymbol{\phi}_T}(\boldsymbol{a} \mid \boldsymbol{s})$, matching or improving the initial policy $\pi_{\boldsymbol{\phi}_1}(\boldsymbol{a} \mid \boldsymbol{s})$

```
// pre-train policy and critic
```

Train initial policy from demonstrations using Equation 5, $\boldsymbol{\phi}_1 = \arg\min_{\boldsymbol{\phi}} \mathcal{L}_{\text{BC}}(\boldsymbol{\phi})$.
Pre-train critic on $\mathcal{D}$ using $\boldsymbol{\phi}_1$ and Equation 6.
for $h = 1 \to T$ **do** `// finetune policy with reinforcement learning`
  Interact with environment $\boldsymbol{s}_{h+1} \sim P(\cdot \mid \boldsymbol{s}_h, \boldsymbol{a}_h)$, $\boldsymbol{a}_h \sim \pi_{\boldsymbol{\phi}_h}(\cdot \mid \boldsymbol{s}_h)$, store in buffer $\mathcal{B}$;
  for $n = 1 \to N$ **do**
    Update critic using $\mathcal{L}_Q(\boldsymbol{\theta})$ (Equation 6) on minibatch from $\mathcal{B}$ and $\mathcal{D}$;
    if $i \% D = 0$ **then**
      Update policy optimizing $\mathcal{L}_\pi(\boldsymbol{\phi})$ (Equation 1) on minibatch from $\mathcal{B}$ and $\mathcal{D}$;
    end
  end
end

inverse RL setting, combining it with SAC and the same actor objective (Equation 1). While this policy architecture was required for their learned reward, they also benefited from regularization properties of stationary priors observed by Rudner et al. [33]. However, while Rudner et al. [33] used a (stationary) homoscedastic, nonparametric Gaussian process BC prior and an MLP policy, we and Watson et al. [43] adopt a stationary network policy for both to benefit from parameter initialization and avoid the scaling issues that arise with nonparametric GPs [31].

3.2 Off-policy reinforcement learning from demonstrations

We now outline the design of `XQC` and our extensions for `XQCfD`.

Actor-critic algorithm. `XQC` builds on SAC and adopts three design decisions: batch norm, weight norm, and a categorical critic. Following [4], during critic learning we apply batch norm to the states and next states combined, i.e., computing $\boldsymbol{q}, \boldsymbol{q}' = Q_{\boldsymbol{\theta}}([\boldsymbol{s}, \boldsymbol{s}'], [\boldsymbol{a}, \boldsymbol{a}'])$ to ensure the batch statistics correctly capture the state-action distribution seen by the critic. In practice a target network is used to predict $\boldsymbol{q}'$. Secondly, we apply weight norm to the critic network to constrain them to the unit sphere, i.e., $\tilde{\boldsymbol{W}} = \boldsymbol{W}/||\boldsymbol{W}||_2$. This normalization keeps the effective learning rate constant [17], which describes the phenomena where learning slows down as the weight norms of the parameters grow due to gradients scaling inversely proportional to the weights when batch norm is used [24]. The categorical critic has 101 atoms and a cross entropy loss following the C51 architecture (Appendix B, Bellemare et al. [3]). For the actor, applying BN and WN is optional, but often beneficial and therefore used in our experiments.

For `XQCfD`, we make three algorithmic changes: initializing the policy via pre-training, replacing the entropy term in the actor loss with KL regularization (Equation 1), and maintaining both the demonstration and replay buffer separately. The algorithm is summarized in Algorithm 1.

Model pre-training. For BC pre-training, we use a loss function that combines mean-squared error (MSE) and negative log likelihood (NLLH) metrics. As our stochastic policy predicts both mean and variance, it is a heteroscedastic regression model. In deep learning, these models can suffer from ambiguity between signal and noise, resulting in regression fits that under-fits in the mean and overestimates the variance rather than accurately modeling complex functions. The *faithful* loss [38] resolves this issue by training the mean prediction $\boldsymbol{\mu}(\boldsymbol{s}) = \boldsymbol{f}_{\boldsymbol{\mu}}(\boldsymbol{\varphi}(\boldsymbol{s}))$ with the MSE objective, while training only the variance head with the NLLH. Crucially, gradients are cut between the variance head and the shared feature space $\boldsymbol{\varphi}(\boldsymbol{s})$, so that $\boldsymbol{\Sigma}(\boldsymbol{s}) = \boldsymbol{f}_{\boldsymbol{\Sigma}}(\text{sg}(\boldsymbol{\varphi}(\boldsymbol{s})))$, where $\text{sg}(\cdot)$ denotes the stop-gradient operator, to ensure the features are optimized for the mean fit rather than uncertainty quantification. The resulting loss is

$$\mathcal{L}_{\text{BC}}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{a},\boldsymbol{s} \sim \mathcal{D}}[\|\boldsymbol{a} - \boldsymbol{l}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{s}))\|_2^2 - \log p(\boldsymbol{z} \mid \boldsymbol{\mu}(\boldsymbol{s}), \boldsymbol{\Sigma}(\boldsymbol{s})^2) - \log |\nabla_{\boldsymbol{z}} \boldsymbol{l}(\boldsymbol{z})|], \tag{5}$$

where $\boldsymbol{l}(\cdot)$ is the action transform, e.g., $\tanh$, of the latent Gaussian action $\boldsymbol{z} \sim p(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$. We also pre-train the critic on the demonstration data using the same critic objective used for RL.

Policy regularization. For the KL divergence between policies in Equation 1, we use the closed-form KL between latent Gaussian actions, as the KL divergence between the tanh-squashed Gaussian has no closed-form expression and would require a Monte Carlo KL approximation. Since the KL divergence is invariant under shared bijections [27], the closed-form Gaussian KL is in fact equal to our desired quantity. Applying the closed-form KL to regularize predictions has also been used in supervised learning [42]. For the KL scaling term α, we find a constant value is sufficient to regularize the update [43] and did not jointly optimize the temperature like Haarnoja et al. [10]. The BC parameters initialize the RL policy, so the KL starts at zero.

Experience replay. When combining data buffers, we combine the minibatches equally throughout training, coined as 'symmetric sampling' in Ball et al. [2]. We also use the expert demonstration data to pre-compute the reward statistics for normalization and avoid online estimators.

4 Experiments

We now evaluate the empirical performance of `XQCfD` across a diverse set of sparse-reward manipulation tasks drawn from three established benchmarks, comparing `XQCfD` with HetStat and MLP policy architectures. This section provides an overview of the experimental setup, including environments, implementation details and evaluation protocol. Section 4.1 presents our main results, comparing against relevant baselines in terms of sample efficiency and asymptotic performance. We then ablate key design decisions to isolate their individual contributions (Section 4.3).

Environments. We evaluate `XQCfD` on a total of 11 sparse-reward manipulation tasks spanning 3 popular benchmarks. From Adroit [30], we use four dexterous hand tasks: `pen`, `door`, `hammer`, `relocate`. From Robomimic [18], we consider three tasks: `Lift`, `NutAssemblySquare`, and `PickPlaceCan`. From MimicGen [19], we include four tasks: `StackThree`, `Coffee`, `HammerCleanup`, `MugCleanup`. These environments are characterized by high-dimensional action spaces, contact-rich dynamics, and binary sparse rewards, making learning particularly challenging for agents relying solely on online exploration.

Demonstration data. For each task, we assume access to the standard dataset of expert demonstrations. For most environments this is 200 demonstrations per task, apart from `StackThree_D0` which provides 1000 demonstrations to variety of initial cube positions. This data regime is reasonable to be collected manually by experts and provides performant BC policies. We use the same expert datasets for BC pre-training, replay buffer augmentation and reward normalization statistics.

Baselines. We compare `XQCfD` against several baselines, including BC [28], `RLPD` [2], `XQC` [25], and `SAC` [10]. As a means of separating the contribution of the actor-critic algorithm from the RLfD, we consider three variants of `XQC`, reflecting several RLfD approaches. These are (i) using BC pre-training (`XQC` + BC), (ii) combining expert offline data with the online replay buffer (`XQC` + OD) like `RLPD`, and (iii) combining (i) and (ii) (`XQC` + BC + OD). For `SAC`, we use a HetStat policy.

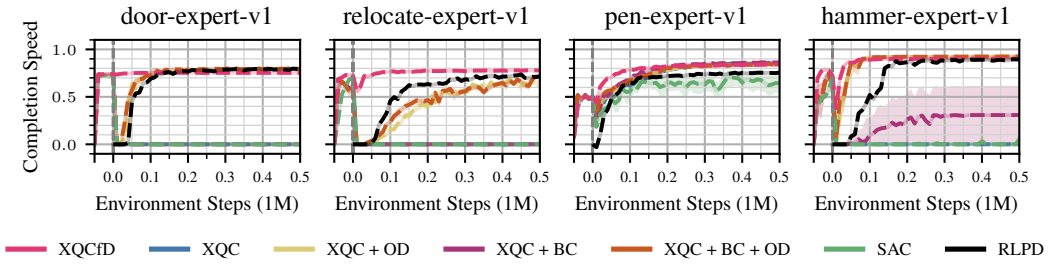

Figure 2: Performance of `XQCfD` and baselines on Adroit over 10 seeds, showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals. BC pretraining performance is shown before the vertical dashed line. Notably, in two of the tasks, BC achieves expert performance. The benefit of improving upon BC is clear in this scenario, and baselines then require 100K–1M interactions to recover BC-level performance after unlearning. In the pen task, where BC is only $\sim$60% successful, `XQCfD` only achieves a speed-up of 50K interactions, which supports our Hypothesis 2.

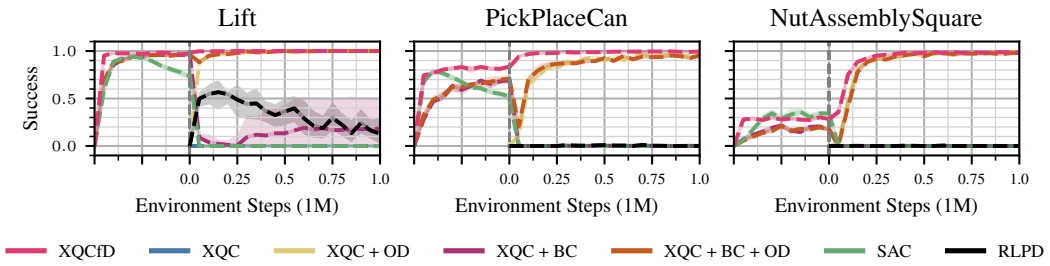

Figure 3: Performance results on Robomimic over 10 seeds showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals. BC pretraining performance is shown before the vertical dashed line. OD refers to adding expert offline data. Robomimic includes three sparse manipulation tasks that are challenging to solve without leveraging demonstrations. For `Lift` and `PickPlaceCan`, the benefit of `XQCfD` is clear, and BC performs close-to-expert performance and therefore `XQCfD` is dramatically more stable than baselines. However, we see in the `Lift` task (simply lifting a block) `XQC` + offline data learns extremely rapidly, and `XQC` + offline data + BC unlearns very little. However, for the harder `PickPlaceCan` task, unlearning is much more severe for our strong `XQC`-based baselines and `XQCfD` achieves a sample efficiency of 1M steps. For the harder `NutAssemblySquare` task, BC performs worse, therefore the performance gap between `XQCfD` and `XQC` + offline data is initially small and rapidly negligible, matching our Hypothesis 2.

Hypotheses. Our experimental design is built upon the following hypotheses.

1. `XQCfD` will improve upon BC performance without unlearning.
2. The sample efficiency of `XQCfD` will depend on the performance of the BC policy.
3. `XQC` will still outperform `SAC` and LayerNorm (`RLPD`) in the sparse RLfD setting.

If these three hypotheses are true, `XQCfD` will be an effective RLfD implementation, subject to the quality of the demonstration data.

4.1 Experimental results

In Figures 2, 3, 4, we benchmark `XQCfD` against baselines on Adroit, Robomimic and MimicGen respectively, looking at 11 tasks in total. Across all tasks, Hypothesis 1 holds, with little observable unlearning at the transition to BC. Improving on BC rather than starting from scratch yields notable gains in sample efficiency. In `PickPlaceCan` the strongest baselines require $\sim$1M additional samples to reach `XQCfD` performance and $\sim$500K in `relocate-expert-v1` and `HammerCleanup-D0`. Moreover, `XQCfD` is the only method to achieve non-zero success rate in the harder MimicGen tasks `StackThree-D0` and `Coffee-D0`. For tasks `NutAssemblySquare` and `MugCleanup-D0`, the BC success rate was only in the region of 20-30%, and there was a much smaller performance gap

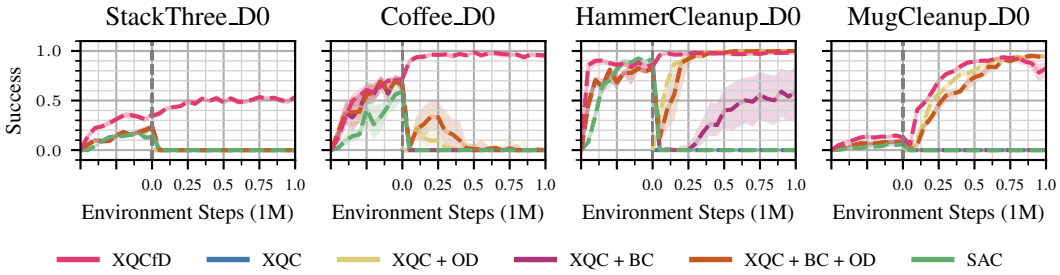

Figure 4: Performance results of `XQCfD` and baselines on MimicGen over 10 seeds, showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals. BC pretraining performance is shown before the vertical dashed line. OD refers to adding expert offline data. Compared to Robomimic, these tasks are slightly more complex due to a broader initial state distribution and longer task horizon. For this suite, `XQCfD` excels in tasks where the initial BC policy performs well. In the Mug task, where the BC policy achieves a low success rate, `XQCfD` is comparable to baselines without the benefit of pretraining. For the hardest task, StackThree, `XQCfD` is the only algorithm to achieve greater-than-zero success, but only improve the BC policy by 10%. This suggests there are still improvements to be made before harder tasks can be mastered.

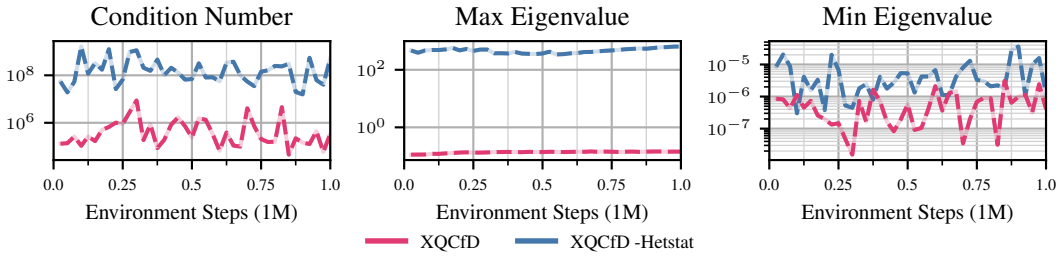

Figure 5: An empirical analysis of the optimization landscape of the actor with and without the HetStat policy for `PickPlaceCan`. The HetStat policy improves the condition number by around 100×, which we attribute to the inherent regularization from the stationarity of the HetStat policy [33]. The condition number analysis technique follows Behrooz et al. [9] and Palenicek et al. [25].

between `XQCfD` and `XQC` + OD. We find a lower temperature was better in these environments, as the agent benefited from a greater trust region to improve on the worse-performing BC policy. However, this result confirms our Hypothesis 2 that the value in `XQCfD` relies on initial BC performance. We found that this temperature parameter was also highly coupled to the unlearning phenomena, as `NutAssemblySquare` and `MugCleanup-D0` both exhibit a small drop in performance due to a lack of regularization. Figure 7 shows that a lower temperature can prevent premature convergence.

Regarding Hypothesis 3, Figure 2 shows that while `SAC` and `RLPD` can solve several tasks, similar `XQC` variants outperform them for all but one environment. This result reflects previous findings on the qualities of `XQC` [24, 25]. Moreover, Ball et al. [2] tune the hyperparameters of `RLPD` per environment. We were not able to tune `RLPD` to solve Robomimic, as observed by Biza et al. [5].

4.2 Actor conditioning analysis

Palenicek et al. [25] have shown that critic architectures with better conditioning can be linked to improved RL training performance. In Figure 5, we see that the HetStat policy improves the condition number of the actor loss by around 100×, supporting the insight that stationary policies are provide a better loss landscape for learning due to their consistent predictions when out of distribution [33].

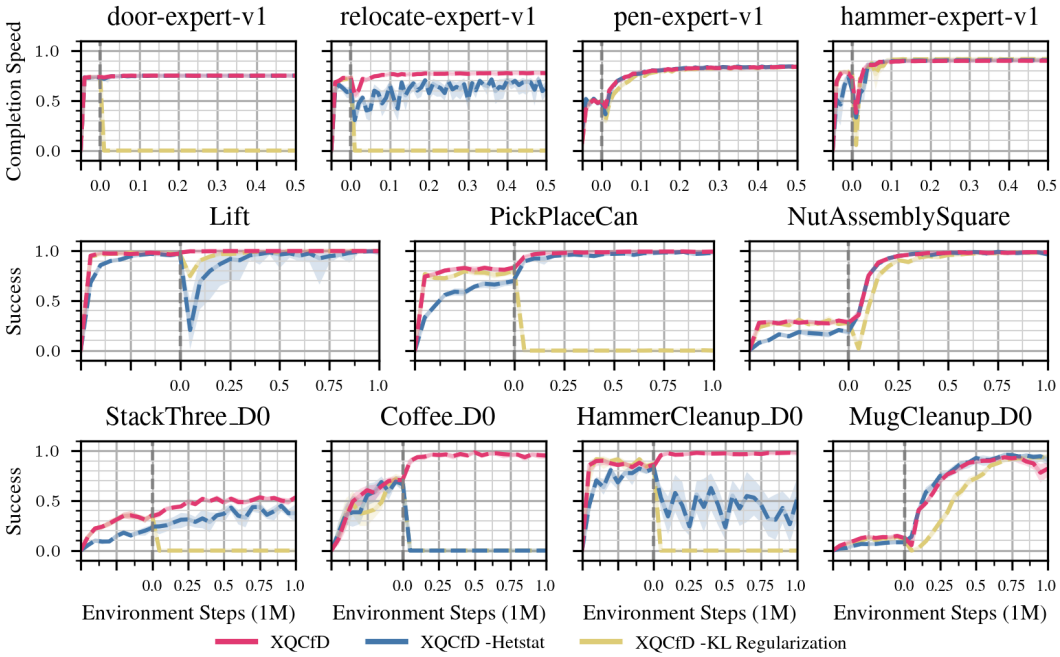

Figure 6: Ablation studies on `XQCfD`, replacing the HetStat MLP with a standard MLP and replacing KL regularization with standard entropy regularization for the Adroit (top), Robomimic (middle) and MimicGen (bottom) environments over 10 seeds, showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals.

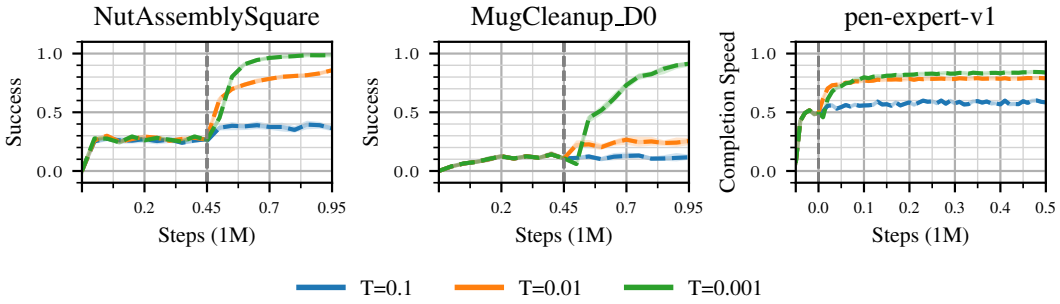

Figure 7: Sensitivity study of `XQCfD`'s temperature α that controls KL regularization against the BC policy. Evaluated over one task for Adroit, Robomimic and MimicGen over 10 seeds, showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals. Lower temperatures result in a larger performance drop on the transition from BC to RL due to unlearning, but lower temperatures also facilitate greater RL finetuning due to the smaller proximal regularization.

4.3 Ablation studies and hyperparameter sensitivity

In Figure 6, we perform an ablation study on the HetStat policy and KL regularization. For 6 out of 11 environments, the HetStat policy has clear benefits over the standard MLP for stable finetuning.

Regarding KL regularization, for 6 out of the 11 environments, removing KL regularization resulted in complete unlearning and failure to converge. For the 5 environments where KL regularization is not needed, these are environments where BC has a low success rate, so standard RL with auxiliary data is sufficient and unlearning does not need to be prevented. While HetStat policies and KL regularization are needed under the same conditions, only one environment (`Coffee`) required both. Our hypothesis is that the HetStat architecture is beneficial for stable finetuning, hence replacing it results in poor convergence (`relocate`, `Lift`, `PickPlaceCan`, `HammerCleanup`). Removing KL regularization appears to frequently trigger rapid unlearning, although it is unclear why the agent cannot recover with additional experience, as the baselines appear to do when learning from scratch. The baselines that do not have the additional data to aid exploration (`SAC` and `XQC +BC`) consistently fail after unlearning.

In Figure 7, we sweep across different values of the fixed temperature. Higher temperatures correspond to stronger BC regularization. When the BC policy has weak performance, online policy improvement struggles with stronger regularization, so lowering the temperature increases the policy improvement during online training.

Lastly, in Figures 8 to 10 in Appendix A, we provide a hyperparameter sensitivity experiment w.r.t. the number of expert demonstrations, which affects both the BC pretraining and auxiliary replay buffer. The trend we observe is that if fewer demonstrations result in a weaker BC policy, RL finetuning can be weaker, which suggests the KL regularization should be tuned to account for a suboptimal BC policy when using fewer demonstrations.

5 Conclusion

We present `XQC` from demonstrations (`XQCfD`), an effective RLfD implementation that combines a sample-efficient RL algorithm, BC pretraining and regularization, and stationary policies. `XQCfD` solves an persistent limitation of prior deep off-policy RLfD approaches that are incapable of efficiently finetuning BC policies due to the randomly initialized critic causing the policy performance to drop below BC. By finetuning BC policies instead of relearning from scratch, sample efficiency is significantly improved. Across 11 sparse-reward manipulation tasks from Adroit, Robomimic and MimicGen, `XQCfD` achieves state-of-the-art performance with a low update-to-data ratio and no Q-function ensembles. Our ablations confirm that our design decisions contribute to this performance in combination. For future work, we plan to extend this algorithm to the action chunking setting with generative policies, which has been shown to improve BC performance on complex tasks [7] but is currently not stationary or straightforward to finetune with off-policy RL.

Limitations. Our experimental scope is restricted to expert demonstrations; we do not evaluate on suboptimal offline data, although `XQCfD` could be applied to this setting with no adjustments.

Another current limitation is the fixed KL temperature, since RL performance can be sensitive to the BC quality when balancing regularization against premature convergence. Future work could investigate adaptive temperature schedules that adjust regularization strength during training.

Acknowledgments and Disclosure of Funding

The work was enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre. We further thanks the Swedish Research Council and the Knut and Alice Wallenberg Foundation. This research was funded by the research cluster "Third Wave of AI", funded by the excellence program of the Hessian Ministry of Higher Education, Science, Research and the Arts, hessian.AI and has benefited from the early stages of the funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy— EXC-3057; funding will begin in 2026. This work was also supported by the Knut and Alice Wallenberg Foundation, Swedish Research Council and ERC AdV BIRD 88480, as well as the UKRI/EPSRC Programme Grant [EP/V000748/1]. Ingmar Posner holds concurrent appointments as a Professor of Applied AI at the University of Oxford and as an Amazon Scholar. This paper describes work performed at the University of Oxford and is not associated with Amazon.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations (ICLR)*, 2018.

[2] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning (ICML)*, 2023.

[3] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2017.

[4] Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. CrossQ: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. *International Conference on Learning Representations (ICLR)*, 2024.

[5] Ondrej Biza, Thomas Weng, Lingfeng Sun, Karl Schmeckpeper, Tarik Kelestemur, Yecheng Jason Ma, Robert Platt, Jan-Willem van de Meent, and Lawson LS Wong. On-robot reinforcement learning with goal-contrastive rewards. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[6] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W Ross. Randomized Ensembled Double Q-Learning: Learning fast without a model. In *International Conference on Learning Representations (ICLR)*, 2021.

[7] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research (IJRR)*, 2025.

[8] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain Markov process expectations for large time. *Communications on Pure and Applied Mathematics*, 1983.

[9] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via Hessian eigenvalue density. In *International Conference on Machine Learning (ICML)*, 2019.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

[11] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

[12] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2024.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[14] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Advances in Neural Information Processing Systems (NeurIPS)*, 2008.

[15] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble. In *Conference on Robot Learning (CoRL)*, 2022.

[16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 2016.

[17] Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[18] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[19] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning (CoRL)*, 2023.

[20] Lassi Meronen, Martin Trapp, and Arno Solin. Periodic activation functions induce stationarity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[21] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[22] Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[23] Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What matters for adversarial imitation learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[24] Daniel Palenicek, Florian Vogt, Joe Watson, and Jan Peters. Scaling off-policy reinforcement learning with batch and weight normalization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[25] Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. XQC: Well-conditioned optimization accelerates deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2026.

[26] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[27] Yury Polyanskiy. Information theoretic methods in statistics and computer science: Lecture 1 — f-divergences, 2020.

[28] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 1991.

[29] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

[30] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems (RSS)*, 2018.

[31] Carl E. Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[32] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Robotics: Science and Systems (RSS)*, 2013.

[33] Tim GJ Rudner, Cong Lu, Michael A Osborne, Yarin Gal, and Yee Teh. On pathologies in KL-regularized reinforcement learning from expert demonstrations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[35] Nicol Schraudolph, Peter Dayan, and Terrence J Sejnowski. Temporal difference learning of position evaluation in the game of Go. *Advances in Neural Information Processing Systems (NeurIPS)*, 1993.

[36] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[37] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

[38] Andrew Stirn, Hans-Hermann Wessels, Megan Schertzer, Laura Pereira, Neville E. Sanjana, and David A. Knowles. Faithful heteroscedastic regression with neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

[39] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

[40] Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

[41] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[42] Joe Watson, Jihao Andreas Lin, Pascal Klink, and Jan Peters. Neural linear models with functional Gaussian process priors. In *Third Symposium on Advances in Approximate Bayesian Inference*.

[43] Joe Watson, Sandy H. Huang, and Nicolas Heess. Coherent soft imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[44] Yi Zhao, Rinu Boney, Alexander Ilin, Juho Kannala, and Joni Pajarinen. Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning. *Offline Reinforcement Learning Workshop at Neural Information Processing Systems*, 2021.

A Ablation: Sensitivity to the Number of Expert Demonstrations

Figures 8 to 10 show that `XQCfD` remains effective across a range of demonstration dataset sizes on Adroit, Robomimic, and MimicGen. As expected from Hypothesis 2, gains scale with initial BC quality, indicating that the KL temperature can be tuned to extract further performance when fewer demonstrations are available.

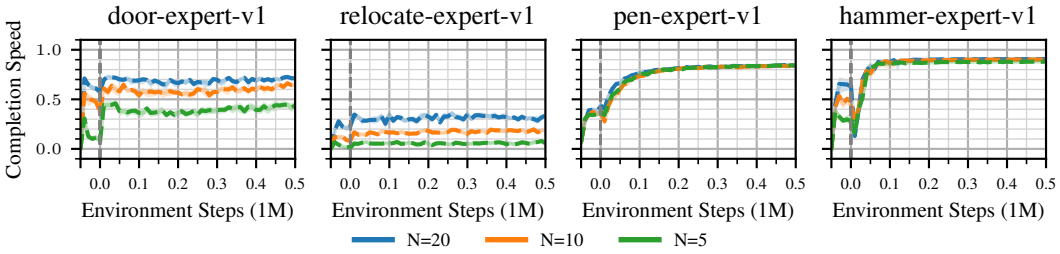

Figure 8: Sensitivity study on `XQCfD`, replacing the number of expert demonstrations for the Adroit environments over 10 seeds, showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals.

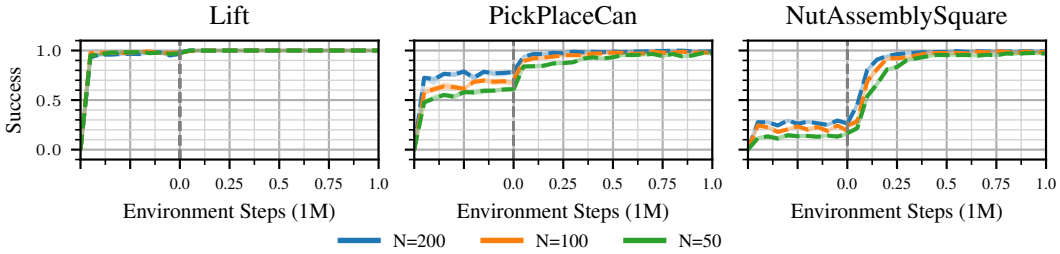

Figure 9: Sensitivity study on `XQCfD`, replacing the number of expert demonstrations for the Robomimic environments over 10 seeds, showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals.

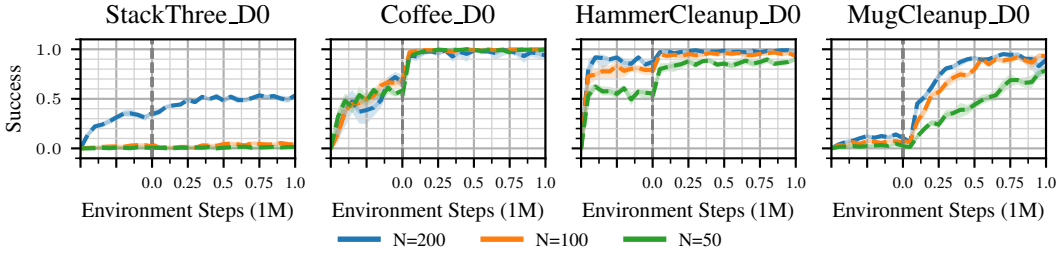

Figure 10: Sensitivity study on `XQCfD`, replacing the number of expert demonstrations for the MimicGen environments over 10 seeds, showing the IQM and 10th and 90th percentile stratified bootstrap confidence intervals.

B The Categorical Critic Objective

In C51 [3], the critic models the return distribution as a categorical distribution over N fixed atoms $\{z_i\}_{i=1}^{N}$ with probabilities $\boldsymbol{p}_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}) = (p_{\boldsymbol{\phi},1}(\boldsymbol{s}, \boldsymbol{a}), \ldots, p_{\boldsymbol{\phi},N}(\boldsymbol{s}, \boldsymbol{a}))$. The target distribution is obtained by projecting the Bellman update $r + \gamma Z$ onto the atom support, yielding projected

probabilities $\boldsymbol{m}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$. The critic is then trained by minimizing the cross-entropy loss,

$$\mathcal{L}_Q(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}'\sim\mathcal{B}}\left[\sum_{i=1}^{N} m_i(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') \log p_{\boldsymbol{\theta},i}(\boldsymbol{s}, \boldsymbol{a})\right], \tag{6}$$

where the target projection coefficients are computed as

$$m_i(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = \sum_{j=1}^{N}\left[1 - \frac{\left|[\,r + \gamma z_j\,]_{z_1}^{z_N} - z_i\right|}{z_{i+1} - z_i}\right]_0^1 p_{\bar{\boldsymbol{\theta}},j}(\boldsymbol{s}', \boldsymbol{a}'). \tag{7}$$

with $\boldsymbol{a}' \sim \pi_{\boldsymbol{\phi}}(\cdot \mid \boldsymbol{s}')$. Function $[\cdot]_a^b$ denoting clipping to $[a, b]$, and $\bar{\theta}$ denoting the parameters of a target network.

C Experimental Details

Evaluation protocol. All experiments were run using 10 random seeds. For Robomimic and MimicGen, evaluations are performed every 50000 environment steps, whereas for Adroit, evaluations are performed every 10000 environment steps. In each evaluation phase, we execute 100 rollouts using a deterministic policy. For Robomimic and MimicGen, we report the average success rate, and for Adroit, we report the completion speed, following the protocol of Ball et al. [2].

Implementation details. Our implementation is based on the publicly available `XQC` codebase [25] (https://github.com/danielpalenicek/xqc). We use two sets of hyperparameters that differ only in their fixed temperature. For the environments `MugCleanup`, `NutAssemblySquare`, `Pen`, and `Hammer`, we set the temperature to 0.001, while for all other environments, we use a temperature of 0.01. We observe that stronger BC policies perform better with a higher temperature, whereas weaker BC policies benefit from a lower temperature. Table 1 summarizes our hyperparameters alongside those used in the baseline methods.

Hyperparameter	`XQCfD`	`XQC`	`SAC`	`RLPD`
Learning rate	3e-4	3e-4	3e-4	3e-4
hidden dim	512	512	512	256
Policy Delay	3	3	3	20
Initial Temperature	0.01 / 0.001	0.01	0.01	1.0
Target entropy	-	$\lvert A\rvert/2$	$\lvert A\rvert/2$	$\lvert A\rvert/2$
Target network momentum	0.005	0.005	0.005	0.005
Update-to-data ratio	2	2	2	20
Critic ensemble size	2	2	2	10
Critic loss	Categorical	Categorical	MSE	MSE
Discount	Heuristic	Heuristic	Heuristic	0.99

Table 1: Overview of hyperparameters, including `SAC` (same parameters as `XQC`) and critic loss settings.

RLPD uses environment-specific hyperparameters, which we summarize in Table 2. Since RLPD does not provide hyperparameters for Robomimic, we manually adjusted them. We did not perform any additional tuning of these hyperparameters.

Environment	**CDQ**	**Entropy**	**MLP Size**	**MLP Layers**
Adroit	True	False	256	3
Robomimic	True	False	512	3

Table 2: Overview of RLPD environment-specific hyperparameters.

Compute resources. The experiments for this paper were conducted on a high-performance computing cluster utilizing NVIDIA A100 and H100 GPUs. A single GPU could run 5 seeds in parallel in 16 hours, leveraging JAX vmap over seeds. We estimate the total computational effort for this paper at approximately 1,000 GPU-hours. This total encompasses the primary experimental suite, ablation studies, hyperparameter tuning, and prior investigations.

Licenses. We use the following assets: XQC codebase [25] (MIT), Adroit [30] (Apache 2.0), Robomimic [18] (MIT), MimicGen [19] (NVIDIA License), RLPD [2] (MIT). All assets are used in accordance with their respective licenses.

D Discount Heuristic

We set the discount factor based on the environment's horizon, following Hansen et al. [11]:

$$\gamma = \text{clip}\left(\frac{T/5-1}{T/5}, [0.95, 0.995]\right),$$

where T denotes the episode length.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, the abstract and introduction list the contributions of `XQCfD`. We have additionally added a contributions paragraph to the end of the introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We name limitations and assumptions in the paper and provide a dedicated limitations paragraph at the end of the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Our Lemma 1 is an established result we use to motivate our algorithm design.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have taken great care to aid reproducibility. First, we descibte the proposed algorithm in detail throughout the main paper. Second, Section C in the Appendix details implementation details, evaluation protocols and lists hyperparameters.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We do not provide the code at the current time, however, we plan to release it upon acceptance of the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide details in the main paper, as well as a dedicated section in the appendix listing experimental details, evaluation protocols and hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report IQM and 10th and 90th percentile stratified bootstrap confidence intervals across 10 random seeds for all training curves, following best practices in deep RL research (Agarwal et. al. 2021).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include a paragraph in the Appendix, which details compute resources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and conform to the code of ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This work concerns a sample-efficient deep actor-critic implementation and has no broader societal impact implications we believe would require a discussion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work focuses on fundamental research on small scale deep reinforcement learning algorithms. We neither train, nor provide any large scale model checkpoints that could be misused. Therefore, this question is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite used codebases and benchmarks extensively in the paper and have added a section with references and corresponding licenses to the Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: This work does not use LLMs as part of the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.