Efficient Offline Reinforcement Learning: The Critic is Critical

Adam Jelley
University of Edinburgh
adam.jelley@ed.ac.uk

Trevor McInroe
University of Edinburgh
trevor.mcinroe@ed.ac.uk

Sam Devlin
Microsoft Research Cambridge
sam.devlin@microsoft.com

Amos Storkey
University of Edinburgh
a.storkey@ed.ac.uk

Abstract

Recent work has demonstrated both benefits and limitations from using supervised approaches (without temporal-difference learning) for offline reinforcement learning. While off-policy reinforcement learning provides a promising approach for improving performance beyond supervised approaches, we observe that training is often inefficient and unstable due to temporal difference bootstrapping. In this paper we propose a best-of-both approach by first learning the behavior policy and critic with supervised learning, before improving with off-policy reinforcement learning. Specifically, we demonstrate improved efficiency by pre-training with a supervised Monte-Carlo value-error, making use of commonly neglected downstream information from the provided offline trajectories. We find that we are able to more than halve the training time of the considered offline algorithms on standard benchmarks, and surprisingly also achieve greater stability. We further build on the importance of having consistent policy and value functions to propose novel hybrid algorithms, TD3+BC+CQL and EDAC+BC, that regularize *both* the actor and the critic towards the behavior policy. This helps to more reliably improve on the behavior policy when learning from limited human demonstrations. Code is available at: `https://github.com/AdamJelley/EfficientOfflineRL`

1 Introduction

Recent work has highlighted the effectiveness of supervised learning approaches (without temporal difference learning) for offline reinforcement learning (Emmons et al., 2022; Chen et al., 2021; Brandfonbrener et al., 2021; Peng et al., 2019). Other work has analysed in detail the limitations of these supervised approaches and when off-policy reinforcement learning techniques should be favoured (Kumar et al., 2022b; Brandfonbrener et al., 2022; Paster et al., 2022). Given these seemingly opposing approaches, it is natural to ask: can we get the best of both supervised learning and temporal difference learning for offline reinforcement learning? Specifically, can we provide an approach that provides the training efficiency and stability of supervised learning, while still gaining the performance benefits of multi-step temporal difference learning?

In this work we investigate such an approach, by supervised pre-training off-policy reinforcement algorithms to obtain approximate behavior policy and values before attempting improvement. Our core contribution is to demonstrate that pre-training the critic in addition to the policy is crucial to achieve efficient and stable improvement beyond the performance of the behavior policy.

As a motivational example, we can consider offline tabular Q-learning on the simple 4 state MDP illustrated below, where we are provided with a single offline trajectory from an unknown policy.

Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2024).

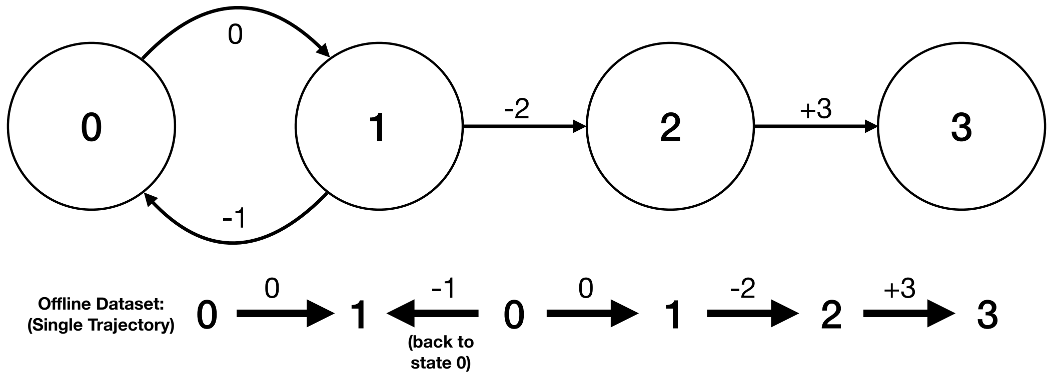

Figure 1: A motivational tabular MDP. In offline reinforcement learning, we are provided with a dataset of trajectories. In this paper we utilize information from the entire trajectory to help initialize a self-consistent critic for off-policy reinforcement learning, which eliminates much of the initial inefficiency and instability associated with bootstrapping in temporal difference losses.

Tabular Q-learning initializes all Q-values to zero (or equivalently Q-networks are randomly initialized such that all initial Q-values are close to zero for deep Q-learning), and then performs temporal difference updates using the following update rule (Sutton et al., 2018):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{1}$$

For simplicity in this minimal motivational example, let us take the discount factor γ and the learning rate α to be 1. Performing updates using the provided offline trajectory leads to the Q-value updates shown in the left column of Table 1. We find that the policy converges to the optimal policy after 2 epochs, and the values converge to the correct optimal values after 3 epochs.

However, in the case of offline reinforcement learning, we have access to better initial Q-value estimates in the form of Monte-Carlo samples from the trajectory. This follows since in addition to Bellman's optimally equation in Equation 1, we also have the definition of the Q-function for the behavior policy π_B for which we can compute a sample-based expectation from our offline data:

$$Q^{\pi_B}(s_t, a_t) = \mathbb{E}_{\tau \sim \pi_B} \left[\sum_{n=0}^{\infty} \gamma^n r_{t+n} \middle| s_t, a_t \right] \tag{2}$$

Initializing the Q-values with Monte-Carlo estimates sampled from the provided trajectory and using the same update rule given in Equation 1 leads to the Q-values shown in the right column of Table 1. We now find that the policy converges to the optimal policy immediately after initialization, and the values converge to the optimal values after a single epoch.

Table 1: State-action values for each epoch of Q-learning to convergence for the motivational MDP and offline trajectory provided in Figure 1 for both zero and Monte-Carlo value initializations.

Epoch	Zero Initialization				MC Value Initialization			
	$Q(0, \rightarrow)$	$Q(1, \leftarrow)$	$Q(1, \rightarrow)$	$Q(2, \rightarrow)$	$Q(0, \rightarrow)$	$Q(1, \leftarrow)$	$Q(1, \rightarrow)$	$Q(2, \rightarrow)$
0	0	0	0	0	0 or +0.5[1]	**0**	**+1**	**+3**
1	0	-1	-2	**+3**	**+1**	**0**	**+1**	**+3**
2	-2	-2	**+1**	**+3**	**+1**	**0**	**+1**	**+3**
3	**+1**	**0**	**+1**	**+3**	**+1**	**0**	**+1**	**+3**

This minimal example demonstrates that the benefit of pre-training values on offline data arises even in the absence of function approximation, near-optimal behavior policies, or complex MDPs. However, the inefficiency of boostrapping from randomly initialized temporal difference (TD) targets still exists in the case of uninformed neural network initializations, and in sparse-reward environments where rewards received at the end of long trajectories may take many TD updates to propagate.

[1]The MC initialization value for $Q(0, \rightarrow)$ depends on the choice of first-visit or every-visit Monte-Carlo (Sutton et al., 2018). In this example it doesn't matter which is used after initialization.

In this paper, we provide similar approaches for pre-training a critic or value network, which can be combined with most existing off-policy reinforcement learning algorithms, leading to improved training efficiency and stability. We first consider return-maximising reinforcement learning, and apply our approach to a minimal offline reinforcement learning baseline, TD3+BC (Fujimoto and Gu, 2021). We then generalize our procedure for entropy-regularized reinforcement learning, and apply our approach to a state-of-the-art offline baseline, EDAC (An et al., 2021). Finally, we build on our insight into the importance of having consistent policy and value functions by introducing novel hybrid algorithms, TD3+BC+CQL and EDAC+BC, that regularize *both* the actor and the critic towards the behavior policy and values to enable smooth performance improvement over the pre-training performance on the challenging Adroit environments (Rajeswaran et al., 2018).

2 Preliminaries and Related Work

Online reinforcement learning (RL) involves an agent taking actions according to a policy π to interact with a Markov Decision Process (MDP). An MDP can be defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, d_0, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(s'|s,a)$ is the transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, d_0 is the distribution of initial states, and $\gamma \in (0, 1]$ is a discount factor. The goal is generally to learn the policy π^* that maximises the expected discounted returns: $\pi^* = \arg\max_\pi \mathbb{E}_{\pi,\mathcal{T}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. Offline RL poses the same goal, but the policy π must be learned from a fixed dataset of interactions from a behavior policy π_B, without any additional data collection. This behavior policy π_B is generally unknown and arbitrarily optimal, and may be a human policy, a hardcoded policy, another agent's policy, or some mixture of policies.

Perhaps the most straightforward approach for learning a policy from the offline data is behavior cloning (Pomerleau, 1991). Since the training is supervised, convergence is relatively stable and efficient, but the learned policy can at best match the performance of the behavior policy since behavior cloning does not utilize reward information, and online performance may be brittle due to accumulating errors taking the agent out-of-distribution (OOD) of known states (Ross et al., 2011).

What if we want to improve on the behavior policy? Behaviour cloning variants such as BC-$k\%$ (Levine et al., 2020) and Advantage-Weighted Regression (Peng et al., 2019; Peters and Schaal, 2007) utilize reward information to selectively clone the behavior data. More recently, conditioning the policy on desired returns or goals (Srivastava et al., 2021; Ma et al., 2022) has seen some success with transformer based approaches (Chen et al., 2021; Janner et al., 2021; Carroll et al., 2022). While these behavior cloning variants are sometimes able to achieve generalization to greater returns at test-time than observed in the dataset (Brandfonbrener et al., 2022; Kumar et al., 2022b), in general their performance is more limited than true reinforcement learning approaches which can more effectively use mechanisms such as trajectory stitching (Paster et al., 2022; Yang et al., 2022).

A more promising approach for improving on the behavior policy is to use off-policy reinforcement learning, usually in the form of actor-critic algorithms (Lillicrap et al., 2019; Silver et al., 2014). However, naively taking the offline dataset as the replay buffer for an off-policy algorithm usually leads to policy collapse (Fujimoto et al., 2019). This occurs because as the policy (actor) and values (critic) are optimized for maximal return, they necessarily go out-of-distribution of the data (Chen and Jiang, 2019). Since there are no additional interactions to provide correcting feedback on these actions and values as in the online case, growing extrapolation errors cause erroneous values and actions that lead to performance equivalent to random. Most modifications of off-policy reinforcement learning algorithms for the offline setting involve regularization of either the actions *or* the values towards the provided dataset to prevent this out-of-distribution extrapolation (Fu et al., 2022). TD3+BC (Fujimoto and Gu, 2021) modifies TD3 (Fujimoto et al., 2018) by introducing a behavior cloning term to regularize the policy towards the behavior policy. Alternatively, CQL (Kumar et al., 2020) modifies Q-learning to regularize the values for out-of-distribution actions to prevent positive extrapolation error. However, since regularization towards the behavior policy or values limits performance improvement (Moskovitz et al., 2022), recent approaches instead aim to capture out-of-distribution uncertainty (Wu et al., 2021). SAC-N and EDAC (An et al., 2021) use the minimum of an ensemble of critics to obtain value estimates that minimise positive extrapolation error (with EDAC introducing additional diversification loss over SAC-N to reduce the required ensemble size), such that policy optimization is less likely to lead to policy collapse.

While these off-policy RL approaches can lead to better performance than modified imitation approaches, their convergence can be inefficient due to the bootstrapping in the Bellman update used for learning values. In this work we attempt to address these difficulties by pre-training with supervised objectives, utilizing Monte-Carlo (MC) return estimates. The use of MC values has a long history in reinforcement learning, for which additional related work is discussed in Appendix A.

3 Theoretical Motivation

In this section we provide a more formal analysis of the intuition provided for the minimal tabular Q-learning example in the introduction. We aim to investigate the functional form with which the Q-function initialization can affect the convergence to optimal Q-values. We consider the case of approximate fitted Q-iteration (Riedmiller, 2005), in which:

$$\hat{Q}_{k+1} \leftarrow \arg\min_{\hat{Q}} ||\hat{Q} - \hat{T}\hat{Q}_k||_\infty \tag{3}$$

where we have defined $\hat{Q}_k$ as the estimate of the $Q(s, a)$ function after k iterations of Equation 3, and $\hat{T}$ as the approximate Bellman operator $\hat{T}Q = \hat{r} + \gamma\hat{P}\max_a Q$, where $\hat{r}(s, a)$ and $\hat{P}(s'|s, a)$ are the sampled reward and environment transition probabilities respectively. The exact equivalents are denoted without the circumflex. We consider minimizing the infinite norm for each iteration since there is no convergence guarantee for the squared norm commonly used for empirical loss functions (Agarwal et al., 2022). In general, the error due to each Q-iteration (between successive iterations of the $\hat{Q}$ function) can be written as the sum of an 'approximation' error, and a 'sampling' error:

$$\begin{aligned} ||\hat{Q}_k - T\hat{Q}_{k-1}||_\infty &= ||\hat{Q}_k - \hat{T}\hat{Q}_{k-1} + \hat{T}\hat{Q}_{k-1} - T\hat{Q}_{k-1}||_\infty & (4) \\ &\leq \underbrace{||\hat{Q}_k - \hat{T}\hat{Q}_{k-1}||_\infty}_{\text{approximation}} + \underbrace{||\hat{T}\hat{Q}_{k-1} - T\hat{Q}_{k-1}||_\infty}_{\text{sampling}} & (5) \end{aligned}$$

Since the offline data is provided for offline reinforcement learning, the samples from which the Bellman operator can be approximated is fixed. We additionally assume here that the entire dataset is sampled uniformly to convergence in each iteration, such that the sampling error is constant and unaffected by Q-function initialization. We therefore focus on the approximation error, and assume this is finite at each iteration, $||\hat{Q}_{k+1} - T\hat{Q}_k||_\infty \leq ||\hat{Q}_{k+1} - \hat{T}\hat{Q}_k||_\infty \leq \epsilon_k$. Since we are interested in the effect of Q-function initialization on final optimality, we write (Agarwal et al., 2022):

$$\begin{aligned} ||\hat{Q}_k - Q^*||_\infty &= ||\hat{Q}_k - T\hat{Q}_{k-1} + T\hat{Q}_{k-1} - Q^*||_\infty & (6) \\ &= ||(\hat{Q}_k - T\hat{Q}_{k-1}) + (T\hat{Q}_{k-1} - TQ^*)||_\infty & (7) \\ &\leq ||\hat{Q}_k - T\hat{Q}_{k-1}||_\infty + ||T\hat{Q}_{k-1} - TQ^*||_\infty & (8) \\ &\leq \epsilon_{k-1} + ||T\hat{Q}_{k-1} - TQ^*||_\infty & (9) \\ &\leq \epsilon_{k-1} + \gamma||\hat{Q}_{k-1} - Q^*||_\infty & (10) \\ &\leq \sum_{i=0}^{k-1} \gamma^i \epsilon_{k-i-1} + \gamma^k ||\hat{Q}_0 - Q^*||_\infty & (11) \end{aligned}$$

using the fact that Q^* is a fixed point of T in Equation 7 and that T is a $\gamma-$contraction in Equation 10.

This provides a measure of the (inifite-norm) Bellman error after k iterations of fitted Q-iteration. We assume that for final convergence to the optimal policy, we would like to achieve an error over the offline dataset of order δ. Therefore for two different intializations of Q, Q_0 and Q_0', we assume that it takes k and k' iterations respectively to reach this error level, and make the bound tight, so we have:

$$\delta = \sum_{i=0}^{k-1} \gamma^i \epsilon_{k-i-1} + \gamma^k ||\hat{Q}_0 - Q^*||_\infty \tag{12}$$

$$\delta = \sum_{i=0}^{k'-1} \gamma^i \epsilon'_{k'-i-1} + \gamma^{k'} ||\hat{Q}'_0 - Q^*||_\infty \tag{13}$$

We now make the strong assumption that the approximation error is in fact independent of the initialization and iteration step, such that $\epsilon_i = \epsilon'_j = \epsilon \; \forall \; i, j \in \mathbb{N}$, so that we can proceed to combine Equations 12 and 13 as follows (assuming without loss of generality that $k > k'$):

$$\sum_{i=k'}^{k-1} \gamma^i \epsilon = \gamma^{k'} ||\hat{Q}'_0 - Q^*||_\infty - \gamma^k ||\hat{Q}_0 - Q^*||_\infty \tag{14}$$

$$\frac{\epsilon(\gamma^{k'} - \gamma^k)}{1-\gamma} = \gamma^{k'} ||\hat{Q}'_0 - Q^*||_\infty - \gamma^k ||\hat{Q}_0 - Q^*||_\infty \tag{15}$$

$$k - k' = \frac{1}{\ln \gamma} \ln \left(\frac{\frac{\epsilon}{1-\gamma} - ||\hat{Q}'_0 - Q^*||_\infty}{\frac{\epsilon}{1-\gamma} - ||\hat{Q}_0 - Q^*||_\infty} \right) \tag{16}$$

We see that $\hat{Q}'_0 \to \hat{Q}_0 \implies k' \to k$, i.e. a given initialization requires a given number of fitted Q-iterations to converge, as expected. We also see that as $\hat{Q}'_0$ becomes closer to Q^* (in terms of the infinite norm over all states and actions), the numerator grows with respect to the denominator and so $k - k'$ increases logarithmically, demonstrating that the number of fitted Q-iterations k' decreases as the Q'_0 initialization improves.

In this work, we parameterize the Q function with a network and assume this can be initialized to the approximate behavior value function Q_B by pre-training the network with a supervised Monte-Carlo value error, so $\hat{Q}'_0 = \hat{Q}_B$. If we assume that the behavior policy π_B that generated the offline data achieves returns better than random (zero), we expect that $||\hat{Q}_B - Q^*||_\infty < ||Q^*||_\infty$. If this holds, then Equation 16 demonstrates that the training efficiency (number of Q-iterations to convergence) should improve by initializing $\hat{Q}'_0 = \hat{Q}_B$ compared to $\hat{Q}_0 = 0$. However, there are many ways in which this theory could be deviated from in practice, such as the use of generalized value iteration (rather than fitted Q-iteration) used by modern deep actor-critic methods, the use of a squared- rather than infinite-norm loss function, the strong assumption that the error in each fitted Q-iteration is constant for any initialization, and the fact that variance in Monte-Carlo value targets may be such that $||\hat{Q}_B - Q^*||_\infty \not< ||Q^*||_\infty$ (especially given the infinite norm). Despite these limitations, this theoretical analysis provides clear motivation for our practical implementation outlined below.

4 Pretraining Off-Policy Reinforcement Learning Algorithms in Practice

4.1 Outline Procedure

We now explain our pre-training procedure for the standard return maximising case. In this setting we first compute the discounted return-to-go R from each state-action pair until the end of the trajectory for all timesteps in the dataset. We can then pre-train the actor with behavior cloning and the critic (Q-network) with the pre-computed discounted return-to-go, both using supervised mean-squared error (MSE) minimisation (or cross-entropy for discrete actions). We note that for more complex datasets the behavior policy and returns may be asymmetric or multi-modal, in which case the implicit Gaussianity assumption for this loss may be limiting and other pre-training losses may be required, but find MSE is sufficient for most standard environments. This procedure provides initial actor and critic networks consistent with the behavior data. Finally, a suitable off-policy reinforcement learning algorithm (utilizing a temporal difference loss) can be applied to these pre-trained actor and critic networks to efficiently increase the policy return. Our pre-training procedure is outlined in pseudocode in Algorithm 1. As we will see in Section 5, this increase in training efficiency more than makes up for the time and computational expense associated with the pre-training.

4.2 Bias-Variance and Optimism-Pessimism Tradeoffs

Under the behavior policy, this discounted return to go provides a Monte-Carlo (MC) sample of the expectation that the critic or Q-network aims to predict. In the case of deterministic environments and a single deterministic behavior policy, this Monte Carlo sample will equal the expectation exactly. For stochastic behavior policies and environments, this Monte Carlo sample may become high variance, which can lead to performance drops after pre-training. In this case it would be possible to use an

n-step or λ-return to reduce this variance at the cost of bias introduced by bootstrapping (a well known case of the bias-variance tradeoff) (Sutton et al., 2018). However, since computing the λ-return would require inferring the current critic value for every downstream state in the trajectory, a more practical way of controlling this tradeoff is simply to compute the $TD(0)$ return and combine it with the MC return using a tradeoff parameter $\lambda \in [0, 1]$:

$$\tilde{R} = (1 - \lambda)R + \lambda(r + \gamma Q(s', a')) \tag{17}$$

For large offline datasets with high environment coverage where the greatest training efficiency gains are possible, there should be sufficient Monte Carlo samples to reduce this variance to a manageable level for pre-training, so λ can be small in order to utilize more information from the offline data (i.e. the full return to go). However, for smaller datasets capturing stochastic policies and environments, larger λ may be beneficial. We investigate the empirical effect of varying λ in Appendix E.

Additionally, by pre-training on sampled returns with a symmetric error, the critic is equally likely to under- or over-estimate the values of out-of-distribution actions when optimizing the policy after pre-training, even in the deterministic case where the returns are exact. This overestimation can lead to policy collapse as discussed in Section 2. Therefore it can be helpful to add some value regularization $\mathcal{R}(Q(s, a))$ during pre-training, such as that introduced in CQL (Kumar et al., 2020) to effectively lower-bound the Q function. We will see in Section 6 that including some value regularization can be beneficial when the offline data is limited. This is reflected in Algorithm 1.

Algorithm 1 Pre-training Hard Off-Policy RL

Require: Dataset D for use as replay buffer
  Initialize π_θ and Q_ϕ parameters, θ and ϕ
  for each transition $(s_t, a_t, s_{t+1}, r_t) \in D$ **do**
    Compute $R_t = \sum_{n=0}^{T-t} \gamma^n r_{t+n}$
      ▷ Or n-step/λ-return/$\tilde{R}$ (eq. 17)
    Append R_t to transition:
      $(s_t, a_t, s_{t+1}, r_t, R_t)$
  end for
  while not converged **do** ▷ **Pre-Training**
    Sample batch $B = (s, a, s', r, R) \sim D$
    Update θ with behavior cloning:
      $\mathcal{L}_\theta = \mathbb{E}_B\left[(\pi(s) - a)^2\right]$
    Update ϕ with (generalized) return:
      $\mathcal{L}_\phi = \mathbb{E}_B\left[(Q(s, a) - R)^2\right] (+\mathcal{R}(Q(s, a))$
        ▷ Optional regularization $\mathcal{R}(Q(s, a))$
    Update target networks (Polyak update):
      $\phi' \leftarrow \tau\phi' + (1 - \tau)\phi$
  end while
  while $t < T$ **do** ▷ **Off-Policy RL**
    Sample batch $(s, a, s', r, R) \sim D$
    Apply hard offline RL update to
      pre-trained π and Q to improve returns
  end while

Algorithm 2 Pre-training Soft Off-Policy RL

Require: Dataset D for use as replay buffer
  Initialize π_θ and Q_ϕ parameters, θ and ϕ
  while not converged **do** ▷ **Actor Pre-Training**
    Sample batch $B = (s, a, s', r) \sim D$
    Update θ with soft behavior cloning:
      $\mathcal{L}_\theta = \mathbb{E}_{B, \tilde{a} \sim \pi}\left[\alpha \log(\pi(\tilde{a}|s)) - \log(\pi(a|s))\right]$
  end while
  for transition $(s_t, a_t, s_{t+1}, r_t) \in D$ **do**
    $R_t = \sum_{n=0}^{T-t} \gamma^n r_{t+n} + \alpha \sum_{n=1}^{T-t} \gamma^n \mathcal{H}(\pi(\cdot|s_{t+n}))$
      ▷ Or n-step/λ-return/$\tilde{R}$ (eq. 17)
    Append R to transition: (s, a, s', r, R)
  end for
  while not converged **do** ▷ **Critic Pre-Training**
    Sample batch $B = (s, a, s', r, R) \sim D$
    Update ϕ with (generalized) soft return:
      $\mathcal{L}_\phi = \mathbb{E}_B\left[(Q(s, a) - R)^2\right] (+\mathcal{R}(Q(s, a))$
    Update target networks: $\phi' \leftarrow \tau\phi' + (1 - \tau)\phi$
  end while
  while $t < T$ **do** ▷ **Off-Policy RL**
    Sample batch $(s, a, s', r, R) \sim D$
    Apply soft RL updates to pre-trained π and Q
  end while

4.3 Generalisation to Maximum Entropy Off-Policy RL Algorithms

The maximum entropy RL framework, and particularly Soft Actor-Critic (Haarnoja et al., 2019) along with offline variants such as SAC-N and EDAC (An et al., 2021), have recently become popular for their improved robustness and sample efficiency relative to the 'hard' return maximisation considered above. This 'soft' RL framework involves maximising the expected return alongside the entropy of the policy, balanced by a temperature parameter α (Ziebart et al., 2010):

$$\pi^* = \arg\max_\pi \mathbb{E}_{\tau \sim \pi, \mathcal{T}}\left[\sum_{t=0}^{\infty} \gamma^t r_t + \alpha\mathcal{H}(\pi(\cdot|s_t))\right] \tag{18}$$

where $\mathcal{H}(\pi(\cdot|s_t)) = \mathbb{E}_{\tilde{a} \sim \pi(\cdot|s_t)}\left[-\log(\pi(\tilde{a}|s))\right]$ is the entropy of the policy π in state s_t.

This also modifies the definition of the state-action value functions as follows:

$$Q(s_t, a_t) = \mathbb{E}_{\tau\sim\pi,\mathcal{T}}\left[\sum_{n=0}^{\infty}\gamma^n r_{t+n} + \alpha\sum_{n=1}^{\infty}\gamma^n\mathcal{H}(\pi(\cdot|s_{t+n}))\Big|s_t, a_t\right] \tag{19}$$

Therefore, in order to pre-train value functions for the behavior policy as before we must modify the return-to-go to incorporate these entropy bonuses from every future timestep except the first. However, since these entropy bonuses depend on the current policy, we now separate our pre-training procedure above into two phases. First, we pretrain our policy with soft behavior cloning:

$$\mathcal{L}_{\pi_\theta} = \mathbb{E}_{s,a\sim D,\tilde{a}\sim\pi}\left[\alpha\log(\pi_\theta(\tilde{a}|s)) - \log(\pi_\theta(a|s))\right] \tag{20}$$

This provides an approximate behavior policy with which we can compute Monte Carlo samples of Equation 19 as soft returns-to-go, which can be used to augment the offline dataset as in Section 4.1. These soft returns-to-go can then be used as the targets to pre-train the critic to achieve consistency with the soft behavior cloned policy, and providing a springboard initialization for a soft off-policy RL algorithm to efficiently improve the policy. The full pseudocode for this procedure is outlined in Algorithm 2, and a further discussion of the rational for this procedure is included in Appendix B.

5 Initial Experiments on D4RL MuJoCo

5.1 Implementation Details

We begin our investigation into the benefits of pre-training off-policy RL algorithms by considering the D4RL MuJoCo benchmark (Fu et al., 2021). We utilize the standard HalfCheetah, Hopper and Walker2d environments and the medium dataset (a suboptimal policy with approximately $1/3$ of the performance of the expert) of 1M transitions, since this provides a meaningful behavior policy to learn from at initialization, with room for improvement with off-policy reinforcement learning. We also consider the medium-replay and full-replay datasets with an identical approach in Appendix F.

For our implementations we utilize the Clean Offline Reinforcement Learning codebase (CORL) (Tarasov et al., 2022), that provides algorithm implementations benchmarked to match published performance measures. During our investigation into improving the efficiency of off-policy reinforcement learning algorithms, we found that introducing LayerNorm (Ba et al., 2016) into both the actor and critic networks significantly improved training efficiency and stability, independently verifying the findings of Ball et al. (2023), as demonstrated in Figure 2. Therefore for all results demonstrated in this paper, we use the benchmarked implementations and default hyperparameters found in the CORL codebase, with the new addition of a LayerNorm after every linear layer except the final one for each network. We investigate the effect of LayerNorm in more detail in Appendix C.

We use TD3+BC (Fujimoto and Gu, 2021) as a ‘hard’ off-policy algorithm, and EDAC (An et al., 2021) as a ‘soft’ entropy-regularized algorithm to apply after pre-training. For EDAC, we include the auxiliary ensemble diversification loss as value regularization during pre-training, to prevent the collapse of the ensemble. We pre-train until convergence in each case, which can be determined by monitoring the convex supervised loss functions. We find this corresponds to just 10-50k updates and is a small proportion of the total updates required for offline RL convergence. The online performance as a function of offline updates, including pre-training, is demonstrated below in Figure 2.

5.2 Results and Analaysis

We find that pre-training as described in Algorithms 1, 2 leads to much more efficient training, both for TD3+BC (a hard RL algorithm with actor regularization) and for EDAC (a soft RL algorithm with critic regularization), even when taking the cost of pre-training into account. In particular, we find that on the more difficult Hopper and Walker2d environments, **the inclusion of pre-training generally reaches expert-level performance in less than $1/2$ of the training steps and computation time required without pre-training** (and in less than $1/5$ of the training steps and computation time of the currently used implementations without LayerNorm or pre-training). Surprisingly, we notice that in many cases the final performance is also more stable, which we discuss further in Appendix D.

However, in some cases, we find that the performance drops after critic pre-training. Fundamentally, this arises because at the end of pre-training we change the objective of the actor from imitation

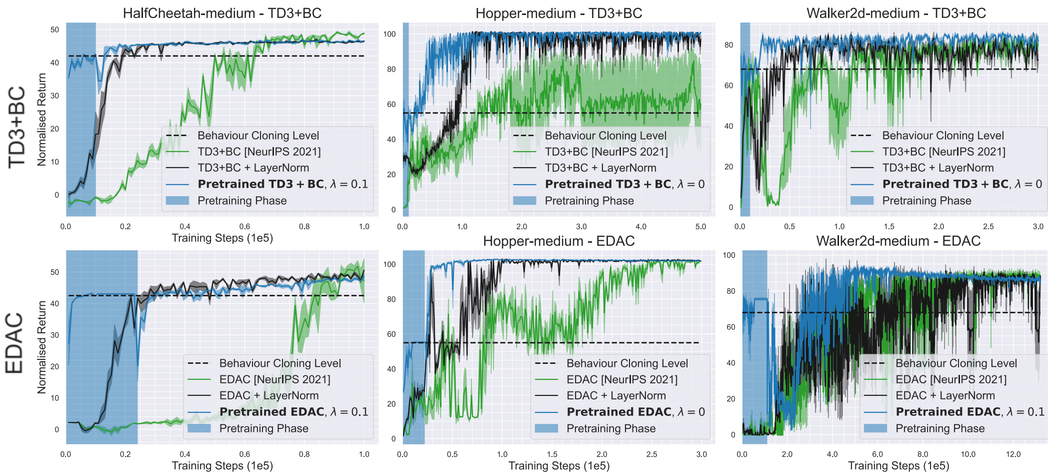

Figure 2: Performance mean $\pm$ standard deviation at each training step over 3 independent seeds. Supervised pre-training before offline reinforcement learning is more efficient than offline reinforcement learning from scratch. Surprisingly, performance is also more stable long after pre-training.

learning (predicting the action that would have been chosen in the dataset), to off-policy reinforcement learning (predicting the action that will maximise the critic prediction). If the values predicted by the critic are sufficiently accurate for the behavior policy as a result of our proposed critic pre-training, then the performance should smoothly improve, but if values are inaccurate due to sample variance then the performance may drop and take some time for Bellman updates to reduce this variance. The dynamics of these environments are deterministic, but the D4RL datasets are collected with a stochastic policy (SAC, Haarnoja et al. (2018)) and from stochastic initial states, which leads to variance in the MC value estimates. Since the action spaces are larger for HalfCheetah and Walker2d than for Hopper (6 dimensional rather than 3), there will be more variance in the policy at each timestep and therefore also in the return. Furthermore, since all but one of the medium-level Hopper trajectories end in termination, this increases the return signal-to-noise ratio for Hopper relative to the HalfCheetah and Walker2d environments which end with timeouts. To mitigate the performance drop off in these environments, we incorporate a small amount of TD target using Equation 17, with $\lambda = 0.1$ for HalfCheetah, and Walker2d for EDAC. We investigate the inclusion of λ in Appendix E.

As an ablation, we also investigate pre-training the actor alone (BC only pre-training) in Appendix D. We find that performance quickly deteriorates after pre-training due to the inconsistent random critic values, verifying our hypothesis on the performance drop, and the findings of Orsini et al. (2021).

6 Extension to Data-Limited Adroit Environments

We now consider the Adroit tasks (Rajeswaran et al., 2018). These are more complex and realistic environments that require controlling a 24-DoF robotic hand to perform tasks such as aligning a pen, hammering a nail, opening a door, or relocating a ball. The human datasets provided for these environments are very limited, consisting of just 25 trajectories of human demonstrations. The cloned datasets augment these trajectories with a behavior cloned policy to get a 50-50 mixture. These demonstrations can be improved upon by acting more efficiently, but are much better than random behavior, so the optimal policy and values should be close to those of the behavior data.

6.1 Motivation for Actor and Critic Regularization With Pre-Training

In such data-limited settings however, off-policy algorithms often suffer from policy collapse as discussed in Section 4.1, since the actor or critic erroneously extrapolate out-of-distribution (OOD) of the offline data. Indeed, when we applied our pre-training approach from Section 5 to TD3+BC and EDAC on the Adroit environments, we found that the human-level pre-training performance often rapidly collapsed after pre-training. As we saw in Section 5.2 and Appendix D, consistency of both actor and critic are crucial for performance improvement. However, most offline RL methods only apply regularization to only one of either the actor or the critic (as also noted by Tarasov et al. (2023)). Motivated by this insight, we introduce two new algorithms that incorporate regularization on *both* the actor and the critic to enable smooth performance improvement after pre-training in data-limited domains. First, we introduce TD3+BC+CQL, which combines the existing behavior cloning on the actor, with additional CQL regularization on the critic, to penalise large OOD Q-values. Second, we introduce EDAC+BC, which combines the existing uncertainty-based regularization on the critic, with additional behavior cloning on the actor, to penalise OOD actions.

6.2 Implementation Details

For all baselines we use their benchmarked CORL implementations (Tarasov et al., 2022) and previously published hyperparameters where possible. Crucially, for the TD3+BC and EDAC baselines, we use the same regularization as TD3+BC+CQL and EDAC+BC for the existing regularization components. We train all algorithms for 300k updates (corresponding to up to 4 hours training time on our RTX2080 GPUs). For our novel algorithms, we pre-train for 200k steps, to provide ample time for supervised convergence. To fairly measure performance we then average online performance evaluated every 10000 offline updates between 200k and 300k updates, over 4 independent seeds. Full details of our hyperparameters and procedure are provided in Appendicies G and H.

6.3 Results and Analysis

Env-Dataset	BC	CQL	TD3+BC	EDAC	**Pre-Trained TD3+BC+CQL (Ours)**	**Pre-Trained EDAC+BC (Ours)**
pen-human	68.3 ± 12.7	51.1 ± 3.8	64.0 ± 2.1	9.2 ± 1.6	$\mathbf{74.1 \pm 15.1}$	71.7 ± 15.5
door-human	0.6 ± 0.7	6.6 ± 6.4	0.1 ± 0.1	-0.2 ± 0.1	0.6 ± 0.6	$\mathbf{13.2 \pm 3.8}$
hammer-human	9.6 ± 4.4	6.7 ± 1.4	5.6 ± 2.5	0.4 ± 0.3	10.6 ± 6.1	$\mathbf{15.8 \pm 5.3}$
relocate-human	2.2 ± 0.6	0.6 ± 0.2	0.4 ± 0.1	0.5 ± 0.1	1.9 ± 0.5	$\mathbf{3.6 \pm 0.5}$
pen-cloned	53.2 ± 12.3	50.3 ± 3.9	21.7 ± 4.9	13.6 ± 12.5	$\mathbf{60.5 \pm 13.0}$	52.6 ± 18.3
door-cloned	0.3 ± 0.5	$\mathbf{6.6 \pm 3.6}$	0.1 ± 0.5	-0.1 ± 0.0	0.0 ± 0.3	0.2 ± 0.5
hammer-cloned	3.6 ± 1.4	4.0 ± 1.5	2.8 ± 2.0	0.7 ± 0.1	1.4 ± 0.6	$\mathbf{8.3 \pm 6.4}$
relocate-cloned	0.1 ± 0.0	0.2 ± 0.1	-0.1 ± 0.1	0.2 ± 0.1	0.1 ± 0.0	0.1 ± 0.1

Table 2: Normalized average returns ($\pm$ standard deviation) on Adroit, by averaging performance between 200k and 300k updates over 4 independent seeds. Our combined algorithms ensure that *both* the actor and critic are regularized to stay close to the behavior policy after pre-training, often giving greater performance than the component algorithms when learning from limited demonstrations.

We find that behavior cloning (BC) provides a strong baseline, notably greatly outperforming previously quoted BC performances on these environments due to our addition of LayerNorm. We see that CQL and EDAC (with LayerNorm) perform reasonably with this evaluation approach using the same hyperparameters as specified in the original papers (Kumar et al., 2020; An et al., 2021), although may benefit from a greater training budget. The original TD3+BC paper did not consider the Adroit environments, but we see comparable performances with relatively strong behavior cloning regularization (tuned with $\alpha = 1$ for the pen environments and $\alpha = 0.1$ for other environments). However, our additions of CQL regularization to TD3+BC and BC regularization to EDAC both generally lead to improved performance in these environments. In fact, we find that our new hybrid algorithms perform similarly using this evaluation procedure even without pre-training, due to their strong inherent regularization, as shown and discussed further in Appendix I. Performance plots in these environments for the human datasets are provided in Appendix J.

7 Conclusion

We have demonstrated that pre-training policies and value functions to first be consistent with the provided offline dataset can improve the efficiency and stability of subsequent off-policy reinforcement learning. In particular, this consistency can reduce subsequent inefficiency and instability associated with bootstrapped temporal difference learning, and can more than halve the number of updates (and therefore computation time) required for state-of-the-art offline algorithms to converge on standard environments. Building on our insight into the importance of policy and value consistency, we demonstrated that combining regularization on both components can enable greater performance than policy or value regularization alone when learning from limited human demonstrations. We hope our research inspires further work towards bridging the gap between classic and deep algorithms to improve the efficiency and stability of offline reinforcement learning as scale continues to increase.

Acknowledgments and Disclosure of Funding

Thank you to Eloi Alonso, Lukas Schäfer and Tom Lee for insightful discussions. Adam Jelley was supported by Microsoft Research and EPSRC through Microsoft's PhD Scholarship Programme.

References

Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. (2022). Reinforcement Learning: Theory and Algorithms.

Agarwal, R., Schuurmans, D., and Norouzi, M. (2020). An Optimistic Perspective on Offline Reinforcement Learning. arXiv:1907.04543 [cs, stat].

An, G., Moon, S., Kim, J.-H., and Song, H. O. (2021). Uncertainty-Based Offline Reinforcement Learning with Diversified Q-Ensemble. arXiv:2110.01548 [cs] version: 2.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer Normalization. arXiv:1607.06450 [cs, stat].

Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. (2023). Efficient Online Reinforcement Learning with Offline Data. arXiv:2302.02948 [cs].

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying Count-Based Exploration and Intrinsic Motivation. arXiv:1606.01868 [cs, stat].

Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., and Bruna, J. (2022). When does return-conditioned supervised learning work for offline reinforcement learning? arXiv:2206.01079 [cs].

Brandfonbrener, D., Whitney, W. F., Ranganath, R., and Bruna, J. (2021). Offline RL Without Off-Policy Evaluation. arXiv:2106.08909 [cs, stat].

Carroll, M., Lin, J., Paradise, O., Georgescu, R., Sun, M., Bignell, D., Milani, S., Hofmann, K., Hausknecht, M., Dragan, A., and Devlin, S. (2022). Towards Flexible Inference in Sequential Decision Problems via Bidirectional Transformers. arXiv:2204.13326 [cs].

Chen, J. and Jiang, N. (2019). Information-Theoretic Considerations in Batch Reinforcement Learning. arXiv:1905.00360 [cs, stat].

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision Transformer: Reinforcement Learning via Sequence Modeling. *arXiv:2106.01345 [cs]*. arXiv: 2106.01345.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. (2022). RvS: What is Essential for Offline RL via Supervised Learning? arXiv:2112.10751 [cs, stat].

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2021). D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219 [cs, stat].

Fu, Y., Wu, D., and Boulet, B. (2022). A Closer Look at Offline RL Agents.

Fujimoto, S. and Gu, S. S. (2021). A Minimalist Approach to Offline Reinforcement Learning. arXiv:2106.06860 [cs, stat].

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2052–2062. PMLR. ISSN: 2640-3498.

Fujimoto, S., Meger, D., Precup, D., Nachum, O., and Gu, S. S. (2022). Why Should I Trust You, Bellman? The Bellman Error is a Poor Replacement for Value Error. *arXiv:2201.12417 [cs, stat]*. arXiv: 2201.12417.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *arXiv:1802.09477 [cs, stat]*. arXiv: 1802.09477.

Goecks, V. G., Gremillion, G. M., Lawhern, V. J., Valasek, J., and Waytowich, N. R. (2020). Integrating Behavior Cloning and Reinforcement Learning forImproved Performance in Dense and Sparse Reward Environments. *New Zealand*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv:1801.01290 [cs, stat]*. arXiv: 1801.01290.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2019). Soft Actor-Critic Algorithms and Applications. arXiv:1812.05905 [cs, stat].

He, F. S., Liu, Y., Schwing, A. G., and Peng, J. (2016). Learning to Play in a Day: Faster Deep Reinforcement Learning by Optimality Tightening. arXiv:1611.01606 [cs, stat].

Janner, M., Li, Q., and Levine, S. (2021). Offline Reinforcement Learning as One Big Sequence Modeling Problem. arXiv:2106.02039 [cs] version: 4.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. (2022a). Offline Q-Learning on Diverse Multi-Task Data Both Scales And Generalizes. arXiv:2211.15144 [cs].

Kumar, A., Hong, J., Singh, A., and Levine, S. (2022b). When Should We Prefer Offline Reinforcement Learning Over Behavioral Cloning? arXiv:2204.05618 [cs].

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative Q-Learning for Offline Reinforcement Learning. arXiv:2006.04779 [cs, stat].

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. arXiv:2005.01643 [cs, stat].

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*. arXiv: 1509.02971.

Ma, Y. J., Yan, J., Jayaraman, D., and Bastani, O. (2022). How Far I'll Go: Offline Goal-Conditioned Reinforcement Learning via f-Advantage Regression. Technical Report arXiv:2206.03023, arXiv. arXiv:2206.03023 [cs] version: 1 type: article.

Moskovitz, T., Parker-Holder, J., Pacchiano, A., Arbel, M., and Jordan, M. I. (2022). Tactical Optimism and Pessimism for Deep Reinforcement Learning. arXiv:2102.03765 [cs].

Oh, J., Guo, Y., Singh, S., and Lee, H. (2018). Self-Imitation Learning. arXiv:1806.05635 [cs, stat].

Orsini, M., Raichuk, A., Hussenot, L., Vincent, D., Dadashi, R., Girgin, S., Geist, M., Bachem, O., Pietquin, O., and Andrychowicz, M. (2021). What Matters for Adversarial Imitation Learning? arXiv:2106.00672 [cs].

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. (2017). Count-Based Exploration with Neural Density Models. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2721–2730. PMLR. ISSN: 2640-3498.

Paster, K., McIlraith, S., and Ba, J. (2022). You Can't Count on Luck: Why Decision Transformers Fail in Stochastic Environments. arXiv:2205.15967 [cs].

Patterson, A., Neumann, S., White, M., and White, A. (2023). Empirical Design in Reinforcement Learning. arXiv:2304.01315.

Patterson, A., White, A., and White, M. (2022). A Generalized Projected Bellman Error for Off-policy Value Estimation in Reinforcement Learning. *Journal of Machine Learning Research*, 23.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. (2021). Carbon Emissions and Large Neural Network Training. arXiv:2104.10350 [cs].

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. (2019). Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. arXiv:1910.00177 [cs, stat].

Peters, J. and Schaal, S. (2007). Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, Corvalis Oregon USA. ACM.

Pomerleau, D. A. (1991). Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation*, 3(1):88–97. Conference Name: Neural Computation.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. (2018). Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. arXiv:1709.10087 [cs].

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. (2022). A Generalist Agent. Technical Report arXiv:2205.06175, arXiv. arXiv:2205.06175 [cs] type: article.

Riedmiller, M. A. (2005). Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*.

Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. arXiv:1011.0686 [cs, stat].

Schulman, J., Chen, X., and Abbeel, P. (2018). Equivalence Between Policy Gradients and Soft Q-Learning. *arXiv:1704.06440 [cs]*. arXiv: 1704.06440.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. page 9.

Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. (2021). Training Agents using Upside-Down Reinforcement Learning. arXiv:1912.02877 [cs].

Sutton et al. (2018). *Reinforcement Learning: An Introduction*. Second edition.

Tang, Y. (2021). Self-Imitation Learning via Generalized Lower Bound Q-learning. arXiv:2006.07442 [cs, stat].

Tarasov, D., Kurenkov, V., Nikulin, A., and Kolesnikov, S. (2023). Revisiting the Minimalist Approach to Offline Reinforcement Learning. arXiv:2305.09836 [cs].

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. (2022). CORL: Research-oriented deep offline reinforcement learning library. In *3rd offline RL workshop: Offline RL as a "launchpad"*.

Togelius, J. and Yannakakis, G. N. (2023). Choose Your Weapon: Survival Strategies for Depressed AI Academics. arXiv:2304.06035 [cs].

Wilcox, A., Balakrishna, A., Dedieu, J., Benslimane, W., Brown, D. S., and Goldberg, K. (2022). Monte Carlo Augmented Actor-Critic for Sparse Reward Deep Reinforcement Learning from Suboptimal Demonstrations. arXiv:2210.07432 [cs].

Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Behram, F. A., Huang, J., Bai, C., Gschwind, M., Gupta, A., Ott, M., Melnikov, A., Candido, S., Brooks, D., Chauhan, G., Lee, B., Lee, H.-H. S., Akyildiz, B., Balandat, M., Spisak, J., Jain, R., Rabbat, M., and Hazelwood, K. (2022). Sustainable AI: Environmental Implications, Challenges and Opportunities. arXiv:2111.00364 [cs].

Wu, Y., Zhai, S., Srivastava, N., Susskind, J., Zhang, J., Salakhutdinov, R., and Goh, H. (2021). Uncertainty Weighted Actor-Critic for Offline Reinforcement Learning. arXiv:2105.08140 [cs].

Yang, M., Schuurmans, D., Abbeel, P., and Nachum, O. (2022). Dichotomy of Control: Separating What You Can Control from What You Cannot. arXiv:2210.13435 [cs].

Zhang, X. and Ma, H. (2018). Pretraining Deep Actor-Critic Reinforcement Learning Algorithms With Expert Demonstrations. arXiv:1801.10459 [cs, stat].

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. (2010). Modeling Interaction via the Principle of Maximum Causal Entropy.

A Additional Related Work on Use of Monte-Carlo Values and Pre-Training in Reinforcement Learning

Recent work (Fujimoto et al., 2022; Patterson et al., 2022; Chen and Jiang, 2019) has demonstrated that the Bellman error can be a poor proxy for the real value error, particularly when used for incomplete, off-policy datasets as in the offline setting, causing significant issues with utilising the Bellman error as the objective for training value functions offline. Monte-Carlo (MC) return estimates have previously been used successfully in online reinforcement learning to improve the sample efficiency of online exploration (Bellemare et al., 2016; He et al., 2016; Ostrovski et al., 2017; Oh et al., 2018; Tang, 2021; Wilcox et al., 2022), but none of these approaches consider how to use MC returns in offline reinforcement learning, which is becoming of increasing importance for scaling reinforcement learning (Kumar et al., 2022a).

In the context of offline RL, Brandfonbrener et al. (2021) similarly recognise the effectiveness of learning the behaviour policy and value function before improvement, but do not consider the use of supervised learning to improve efficiency, and only take one step of TD improvement to prevent out-of-distribution extrapolation of this value function, rather than the more general and controllable combination of actor and critic regularization we propose. Pre-training policies with imitation learning for offline RL was recently investigated by Orsini et al. (2021), but they found that the gain from pre-training is generally insignificant due to policy updates from randomly initialised critic networks causing the policy to rapidly deteriorate (as we find in Section 5), motivating our work on pre-training the critic. Other work has considered pre-training off-policy algorithms using expert demonstrations (Goecks et al., 2020; Zhang and Ma, 2018), but only consider the online setting and do not consider efficiency. Our work provides the first analysis of the benefits of pre-training with a supervised *value-error* objective, leading to more efficient and stable subsequent off-policy reinforcement learning.

B Rational for Separation of Actor and Critic Pre-training for Entropy regularized Reinforcement Learning

In section 3.3 and algorithm 2 of the main text, we propose separating the pre-training of the policy and value network into two separate phases for entropy-regularized reinforcement learning algorithms. By first pre-training the policy with soft behaviour cloning, an approximate behaviour policy can be learned which then enables approximate behaviour entropy bonuses to be included in the subsequent pre-training of the critic. However, an alternative approach could be to pre-train the policy and critic in parallel, as in the pure return maximisation framework. In theory this would require updating the returns-to-go for each policy update to incorporate the changing entropy bonuses as the variance of the policy is updated. Since this requires a complete forward pass of the policy for all subsequent states in the trajectories of the states sampled for an update, this makes the pre-training infeasibly expensive.

Another potential approach is to only train the mean of the standard Gaussian policy to match the behaviour policy, and keep the variance constant such that all entropy bonuses could be caluclated an intialisation and would be unaffected by policy pre-training. However, we note that the standard tanh squashing applied to the policy to keep the sampled action within the environment action bounds leads to a changing entropy of the resulting policy, even with the Gaussian variance kept constant.

A final approach we considered was to compute the soft returns-to-go based on the initialisation policy, and then only pre-train the values (no behaviour cloning). While this approach was successful and led to training efficiency gains, the rapid updating of the actor at the beginning of training (and particularly the rapidly changing policy entropy) quickly leads to inconsistent values, so we found that the investment in pre-training the policy with soft behaviour cloning first was worth the computational time in most cases.

C Investigation Into Affect of LayerNorm

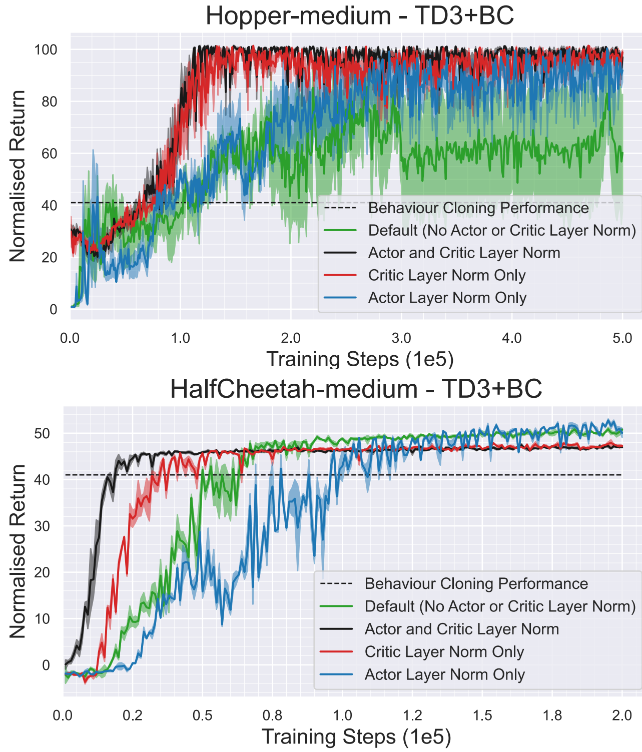

Figure 3: Investigation into the affect of adding LayerNorm to both the actor and critic networks for TD3+BC on Hopper and HalfCheetah-medium. All lines show mean and standard deviation in normalised return at each timestep over 3 seeds.

We investigate the effect of the addition of LayerNorm (Ba et al., 2016) to both the actor and critic networks for TD3+BC on the Hopper-medium and HalfCheetah-medium datasets. The standard author implementation of TD3+BC (along with that of SAC-N, EDAC and most other off-policy reinforcement learning algorithms (Tarasov et al., 2022)) does not include any form of representational normalisation, and is shown in green. We consider the addition of LayerNorm after every linear layer in the network (before activation) except the final linear layer. We find that adding LayerNorms to the critic network leads to significant improvement in training efficiency and stability. This independently verifies the findings of Ball et al. (2023), who hypothesise that this occurs because the normalisation prevents severe value extrapolation for out-of-distribution actions, leading to overestimation error. Surprisingly, we find that the addition of LayerNorm to the actor (without addition to the critic) leads to worse efficiency and stability than the default (no LayerNorm) for the HalfCheetah environment. However, the addition of LayerNorm to both the actor *and* the critic leads to greater training efficiency and stability than the default or applying either normalisation alone, for both environments.

We find that these insights generally hold across investigated environments and datasets. However, we notice this this addition comes at the cost of a few percent in final performance for HalfCheetah. Since this is the only environment for which this was found to occur and we still see significant improvements in efficiency and stability, we apply LayerNorm to both the actor and the critic for all experiments in this paper (except where explicitly stated otherwise) towards our goal of improving training efficiency. We also expect the addition of LayerNorm to be universally introduced to off-policy reinforcement learning algorithms going forwards.

D Ablation of Critic Pre-Training

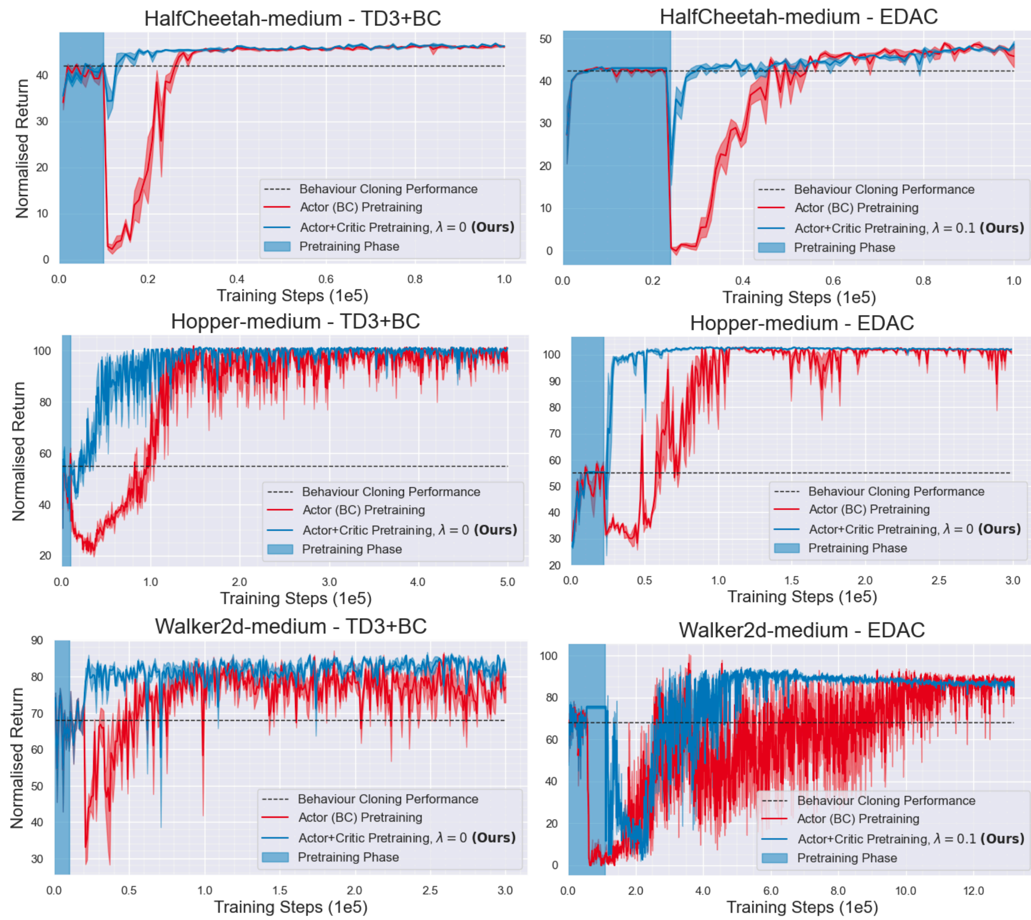

Figure 4: Ablation of critic pre-training (imitation only pre-training shown in red). This demonstrates the benefit of our proposed critic pre-training. Both implementations utilise LayerNorm. All lines show mean and standard deviation in normalised return at each timestep over 3 seeds.

We see that when pre-training the actor with behaviour cloning, the initial performance matches our proposed actor and critic pre-training, but quickly declines after pre-training due to the randomly initialised critic, matching the findings of Orsini et al. (2021). Therefore we see that the improved efficiency and stability from pre-training arises due to our combined actor and critic pre-training. This demonstrates that having a consistent actor and critic is essential for performance improvement.

Surprisingly, we also notice that in many cases the final performance is also more stable, even hundreds of thousands of updates after pre-training. As analysed in Fujimoto et al. (2022) and Chen and Jiang (2019), for finite data regimes such as the offline setting, the Bellman equation can be satisfied by infinitely many suboptimal solutions. Additionally, Q-values trained by minimising the Bellman error are often empirically found be to be inaccurate (Schulman et al., 2018). We hypothesise that this additional stability could be occurring because the initial pre-training using the value error reduces the subset of possible solutions to those with lower value error when subsequently minimising the Bellman error on the finite offline dataset. However, since this benefit is auxiliary to our central focus of improving efficiency, we leave investigation of this effect to further work.

E Investigation into Empirical Bias Variance Tradeoff

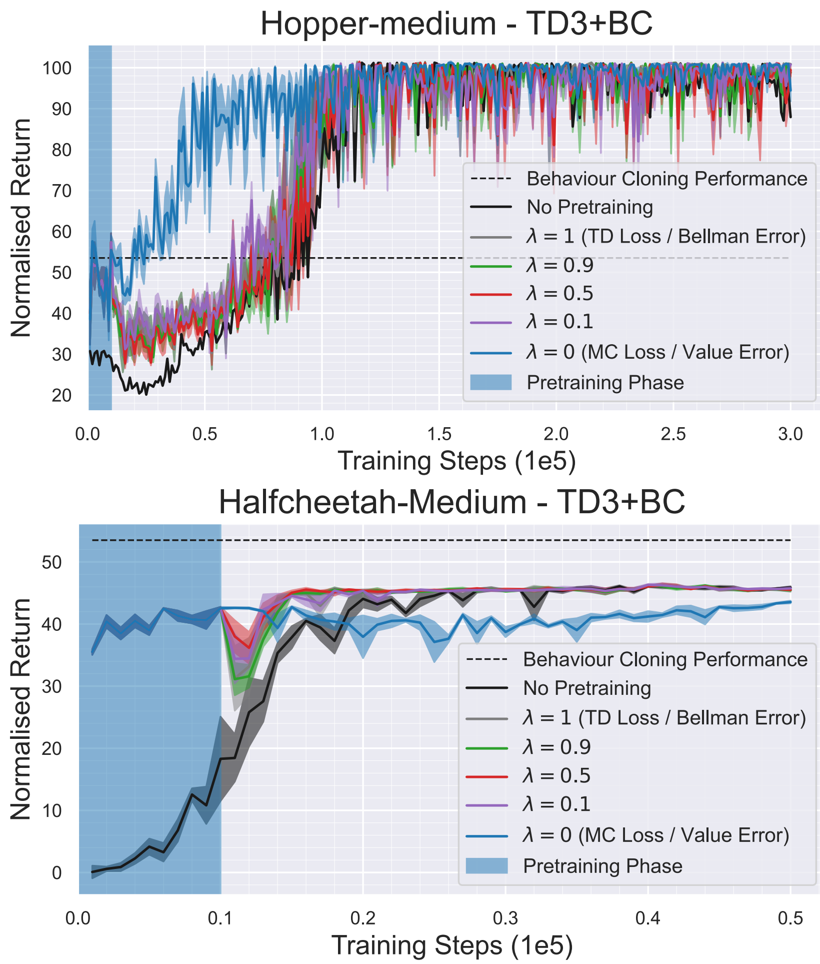

Figure 5: Investigation into the empirical bias-variance tradeoff by varying λ defined in equation 3 for TD3+BC on Hopper and HalfCheetah medium. All lines show mean and standard deviation in normalised return at each timestep over 3 seeds.

We investigate the bias-variance tradeoff described in section 3.2 in practice by empirically varying λ for TD3+BC on the Hopper and HalfCheetah medium datasets. We find that while all values of $\lambda \in [0, 1]$ provide efficiency benefits over no pre-training, the benefit is more significant for $\lambda = 0$ (corresponding to the originally proposed value-error pre-training), likely because even for $\lambda = 0.1$ the temporal difference component to the loss can have significant impact on the training dynamics, and the pre-training duration is not long enough for this bootstrapping loss to reach consistency. However, including some temporal difference component ($\lambda > 0$) can help to smooth the transition with off-policy reinforcement learning as we see for HalfCheetah, while the inclusion of some Monte-Carlo component helps to improve the training efficiency.

We also see that stability on convergence for the Hopper environment is greater for smaller λ and particularly apparent for $\lambda = 0$, supporting our hypothesis in Section 4 and Appendix D that this additional stability follows from the use of the value error (rather than the temporal difference error) in pre-training.

F MuJoCo Medium-Replay and Full-Replay Dataset Experiments

We apply our pre-training approach proposed in Section 4 with an identical experimental implementation to that described in Section 5.1 to the medium-replay and full-replay datasets, shown below in Figures 6 and 7 respectively.

F.1 Medium-replay

The medium-replay dataset consists of 1M transitions from the replay buffer of an agent trained from random to medium performance. The influence of our pre-training approach is shown below.

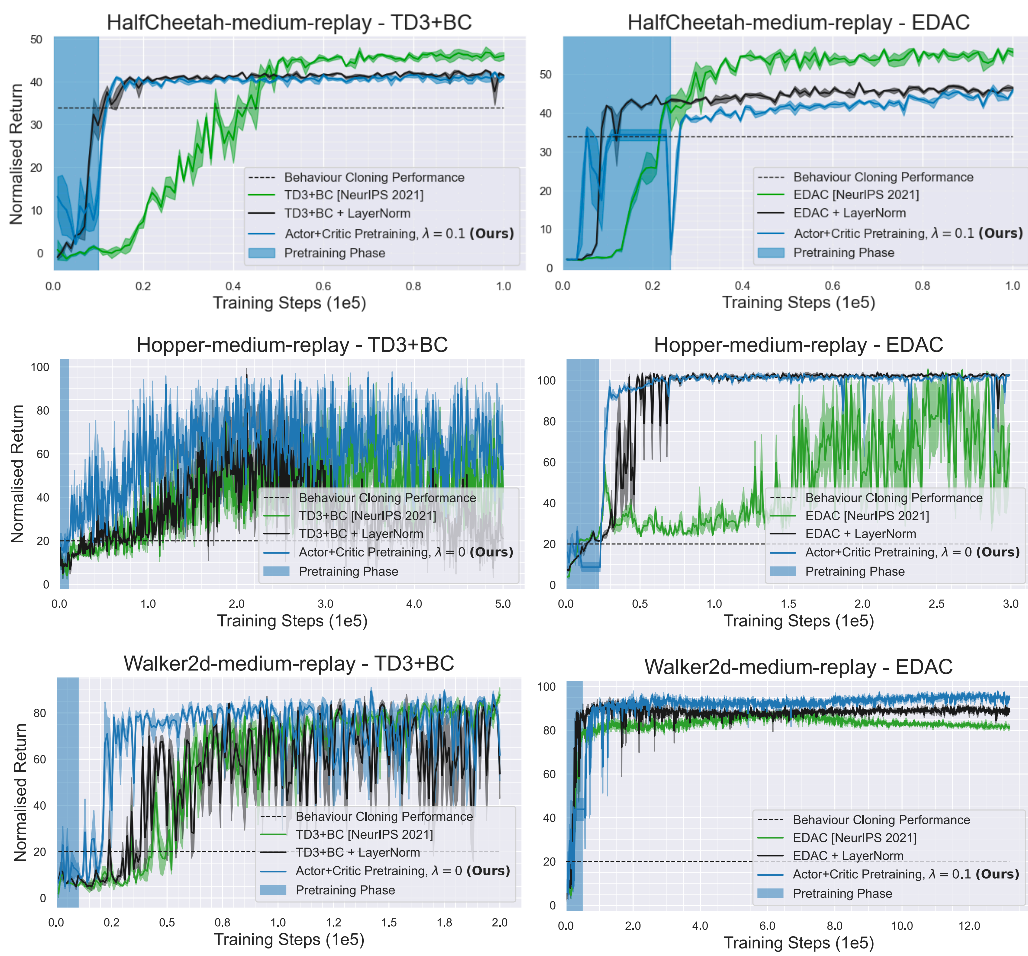

Figure 6: Application of supervised pre-training to the *medium-replay* MuJoCo datasets. We generally see training efficiency gains similar to those observed in Figure 2. Plots show mean and standard deviation at each timestep for 3 independent runs.

We generally see similar efficiency and stability gains to those observed for the medium datasets in Figure 2. However, we see there are no efficiency gains to be made on the HalfCheetah environments given the baselines with our addition of LayerNorm converge so quickly. We also notice that the performance of Walker2d for EDAC is much cleaner, likely due to greater diversity of data helping to stabilise performance.

F.2 Full-replay

The full-replay dataset consists of 1M transitions from the replay buffer of an agent trained from random to expert performance. The influence of our pre-training approach is shown below.

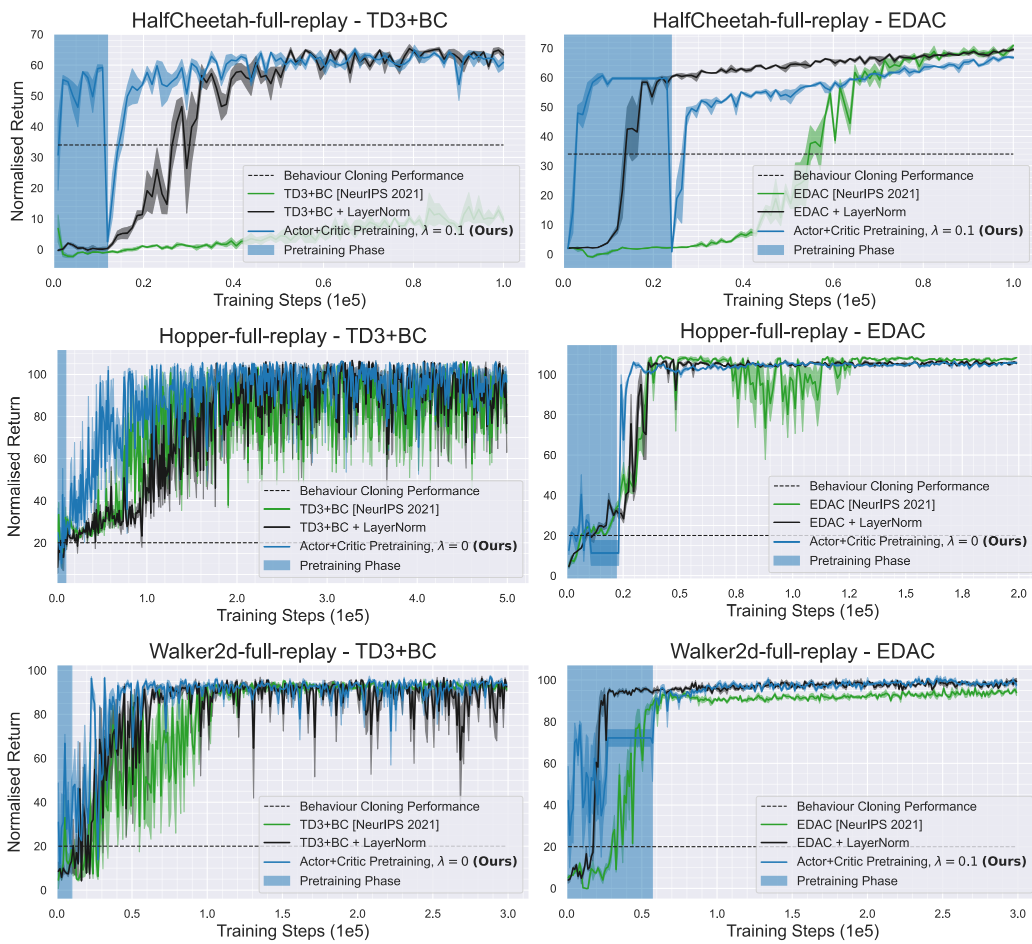

Figure 7: Application of supervised pre-training to the *full-replay* MuJoCo datasets. We generally see training efficiency gains similar to those observed in Figure 2. Plots show mean and standard deviation at each timestep for 3 independent runs.

Yet again we see similar efficiency and stability gains to those observed for the medium datasets in Figure 2 (with the exception of the HalfCheetah environments). In some cases, the efficiency gains may be further improved by optimisation of the pre-training duration and temporal difference component λ from those that were originally selected for the medium datasets.

G Key Hyperparameters for Adroit Experiments

The Adroit experiments were performed and evaluated as described in Section 5 of the main paper, with more detail provided below in Appendix H. Key hyperparameters for each algorithm are provided in Table 3 below. Full hyperparameter configurations (including those not provided below) are available in the config files of the provided CORL codebase (Tarasov et al., 2022). Where possible, hyperparameters were chosen to match previously published values for the Adroit environments, and otherwise their default implementation values. Crucially, where hyperparameters are shared between algorithms (such as between TD3+BC and TD3+BC+CQL) they were chosen to be equal, to investigate whether additional regularization on the critic/actor can improve over the exisiting tuned regularization on the actor/critic alone.

To incorporate our additional regularization losses which may be of different scales to the existing losses, we utilise the normalisation strategy described in TD3+BC (Fujimoto and Gu, 2021). Namely for primary loss α and additional auxiliary loss β which may be of a different scale, we combine them as follows to more evenly balance the losses throughout training:

$$\mathcal{L} = \alpha/|\alpha| + c\,\beta/|\beta| \tag{21}$$

where $|\cdot|$ denotes the magnitude of the gradient-detached loss and c is the regularization coefficient referred to as CQL/BC-regularizer provided in Table 3.

Finally, we note that for the behaviour cloning baseline and for the behaviour cloning regularization in both TD3+BC(+CQL) and EDAC+BC we use 'hard' behaviour cloning (using a mean-squared error objective). In the case of the BC baseline and EDAC+BC it would be possible to use 'soft' behaviour cloning as in Equation 6, but we found in both cases 'hard' behaviour cloning (using the sampled action from the Gaussian policy for EDAC) performed much better. However we still use 'soft' behaviour cloning for pre-training EDAC+BC to maintain the policy entropy in pre-training.

Table 3: Adroit Experiments Key Hyperparameters

ALGORITHM	TASK	PARAMETER	VALUE
BC	All	BC Objective	MSE
CQL	All	n-actions	10
CQL	All	Temperature	1.0
TD3+BC(+CQL)	Pen	α	1.0
TD3+BC(+CQL)	Door/Hammer/Relocate	α	0.1
TD3+BC+CQL	Pen	CQL-regularizer	1.0
TD3+BC+CQL	Door/Hammer/Relocate	CQL-regularizer	10.0
TD3+BC+CQL	All	n-actions	10
TD3+BC+CQL	All	Temperature	1.0
TD3+BC+CQL	All	Pre-training λ	0
EDAC(+BC)	Pen	N (num critics)	20
EDAC(+BC)	Pen (Human)	η	1000
EDAC(+BC)	Pen (Cloned)	η	10
EDAC(+BC)	Door/Hammer/Relocate	N (num critics)	50
EDAC(+BC)	Door/Hammer/Relocate	η	200
EDAC+BC	All	BC Objective	MSE
EDAC+BC	All	BC-regularizer	1.0
EDAC+BC	All	Pre-training λ	0

H Evaluation Procedure for Adroit Experiements

Evaluation for the Adroit experiments is carried out as described in Section 5. In particular, we train all algorithms using the standard hyperparameters provided above for 300k steps, and evaluate their performance by evaluating the agent performance every 10k steps between 200k and 300k steps. For the algorithms including pre-training, we pre-train for 200k steps to allow more than sufficient convergence of the pre-training stage (using $\lambda = 0$). There are two motivations for this evaluation procedure. Firstly, we would like to measure the performance of the algorithms after a relatively short training time (i.e. reduced number of training steps) to provide a quantitative measure of the performance efficiency. Secondly, since the performance of the offline RL algorithms considered on this benchmark (and in general) are very unstable, online performance varies significantly between random and human performance during training. Therefore, to reduce the variance of the results and to incorporate performance stability, we do not take a single (or best) checkpoint, but rather average the performance of checkpoints taken every 10k steps between 200k and 300k training steps, each evaluated online for 10 episodes. This is more representative of real world performance off offline RL algorithms where the online performance may not be possible to evaluate during training.

One additional consideration for evaluation resulted from the fact that we noticed that, aside from the *pen* environment, the human demonstrations in the human datasets were much longer than the truncation limit of the online truncation limit of the environments. The truncation limit for all environments (for the v1 environments other than *pen*) is set to 200 timesteps, while the maximum demonstration lengths are 300, 624 and 527 timesteps for the *Door*, *Hammer* and *Relocate* environments respectively. This partially explains the poor performance of previous algorithms on the non-*pen* environments in this benchmark (An et al., 2021), since the environment does not allow sufficient time to receive reward for successfully imitating the demonstrated behaviour, such as opening the door or hammering the nail, before truncating the epsiode! Therefore we adjust the truncation limit in these environments to match the maximum demonstration lengths. To compute the standard human normalized score (defined as $(\text{agent\_score} - \text{random\_score})/(\text{human\_score} - \text{random\_score})$, we maintain the same maximal human scores as provided in the d4rl benchmark, but adjust the minimal random scores by running the independent uniform random policy in the environments for the new truncation time limits. This gives rise to the new environment evaluation variables provided below in Table 4. Crucially, despite this improved evaluation procedure, we evaluate all algorithms considered using this procedure in an identical manner, to provide a fair comparison of algorithm performance that takes into account both efficiency and stability.

Table 4: Adroit Environment Evaluation Parameters

ENV	**NEW TIMESTEP LIMIT**	**NEW MIN (RANDOM) SCORES**	**MAX (HUMAN) SCORES**
pen	100	-162.09	3076.83
door	300	-84.52	2880.57
hammer	624	-856.83	12794.13
relocate	527	-37.95	4233.88

I Pre-Training Ablation on Adroit Environments

We ablate the pre-training stage of our new hybrid algorithms, TD3+BC+CQL and EDAC+BC, on the Adroit environment below in Table 5.

Table 5: Pre-training Ablation for Our Hybrid Algorithms on Adroit Environments.

	TD3-BC-CQL	Pre-trained TD3-BC-CQL	EDAC-BC	Pre-trained EDAC-BC
pen-human	79.7 ± 10.1	74.1 ± 15.1	$\mathbf{83.8 \pm 14.8}$	71.7 ± 15.5
door-human	0.6 ± 0.3	0.6 ± 0.6	0.9 ± 1.0	$\mathbf{13.2 \pm 3.8}$
hammer-human	13.2 ± 9.7	10.6 ± 6.1	9.3 ± 3.8	$\mathbf{15.8 \pm 5.3}$
relocate-human	2.9 ± 1.4	1.9 ± 0.5	2.1 ± 0.7	$\mathbf{3.6 \pm 0.5}$
pen-cloned	$\mathbf{63.7 \pm 12.5}$	60.5 ± 13.0	56.3 ± 21.3	52.6 ± 18.3
door-cloned	0.1 ± 0.4	0.0 ± 0.3	$\mathbf{0.8 \pm 0.7}$	0.2 ± 0.5
hammer-cloned	2.2 ± 0.7	1.4 ± 0.6	4.9 ± 2.6	$\mathbf{8.3 \pm 6.4}$
relocate-cloned	0.0 ± 0.1	$\mathbf{0.1 \pm 0.0}$	0.0 ± 0.0	0.1 ± 0.1

We see that both versions of each algorithm (with and without pre-training) perform similarly, although there appears to be a slight benefit from pre-training for EDAC+BC in some environments. This is likely partially just because there is little efficiency gain to be made from pre-training in this data-limited setting, and partly because the combined regularization is sufficient to keep the actor and critic consistent with each other and with the data distribution. Also both of the additional regularization components (behaviour cloning and CQL-style regularization) do not rely on temporal difference bootstrapping and therefore have similar efficiency to supervised learning and reduce the benefit of pre-training. However, a shorter pre-training period may demonstrate greater benefits from pre-training (since 200k steps is really more than required, providing additional time for non-pre-trained versions to learn), along with increased dataset size. Importantly, these hybrid algorithms, motivated by our pre-training approach, still demonstrate promising improvements in performance relative to the component algorithms in this data-limited setting, as demonstrated in Table 2.

We also note that even with our substantial efforts to make our evaluation procuedure as ‘fair’ as possible between algorithms (as described in Appendicies G and H) the variances (and therefore confidence intervals) of these Adroit results are still non-negligible due to the nature of the limited human data, the stochastic starting states of the environment, the high variance algorithms used, and our limited computation resources. However, our aim is not to show that any one algorithm is ‘best’, as this is dependent on a wide range of factors and is often entirely infeasible in general (Patterson et al., 2023). Indeed, the performance of these algorithms is generally comparable (as intuitively might be expected given the same limited behaviour data), and significantly improving on the behaviour policy is challenging. Rather, our results on this benchmark aim to demonstrate the idea that if the performance of an algorithm collapses after pre-training (or more generally, imitation learning gives rise to non-negligible performance but off-policy RL does not), this can be mitigated by introducing additional regularization towards the behaviour policy, and it is often more effective to regularize *both* the actor and the critic rather than just one of these components, as in Table 2.

J Performance Plots for Adroit Experiments

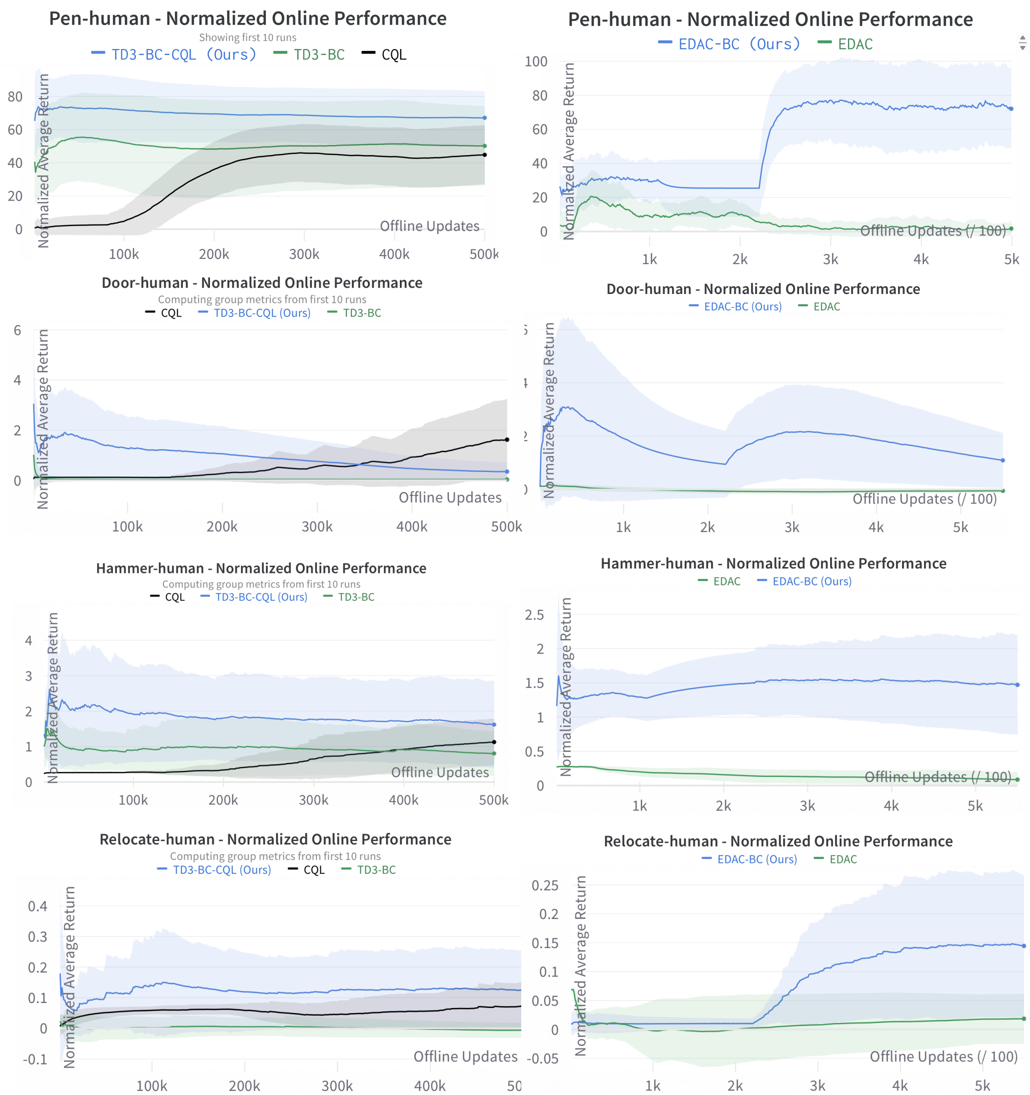

Figure 8: Training plots for Adroit environments using the human datasets. As described in Section 5, we introduce additional combined regularization, giving rise to novel hybrid algorithms TD3+BC+CQL and EDAC+BC, to prevent performance collapse after pre-training. We find that combining actor and critic regularization leads to better performance than equivalent actor or critic regularization alone (regularization hyperparameters provided in Table 3). However, we see that in many of these data-limited environments, subsequent off-policy reinforcement learning is not able to improve upon the initial pre-training performance corresponding to imitation learning (with LayerNorm). Plots show mean and standard deviation at each timestep for 4 independent runs.

K Outlook and Discussion

For academics and RL practitioners with a modest computational budget, the application of the proposed pre-training approach could significantly speed up research and development time, enabling more ideas to be investigated (Togelius and Yannakakis, 2023). For larger computational budgets and datasets, the proposed pre-training approach could save many thousands of GPU hours spent on un-initialised bootstrapping with associated cost and emissions, which is becoming of increasing importance for training large models (Patterson et al., 2021; Wu et al., 2022). Given that offline reinforcement learning currently appears to be a promising avenue to scaling reinforcement learning and achieving associated emergent properties witnessed in related domains (Kumar et al., 2022a; Reed et al., 2022; Agarwal et al., 2020), we anticipate the scale of offline reinforcement learning to only increase. Additionally, for human datasets where the available offline data is generally high quality, our combined regularization approaches, TD3+BC+CQL and EDAC+BC, may additionally help to improve the stability of offline RL and lead to safer online deployment of policies learned entirely offline.