A Closer Look at Advantage-Filtered Behavioral Cloning in High-Noise Datasets

Jake Grigsby
University of Virginia
`jcg6dn@virginia.edu`

Abstract

Recent Offline Reinforcement Learning methods have succeeded in learning high-performance policies from fixed datasets of experience. A particularly effective approach learns to first identify and then mimic optimal decision-making strategies. Our work evaluates this method's ability to scale to vast datasets consisting almost entirely of sub-optimal noise. A thorough investigation on a custom benchmark helps identify several key challenges involved in learning from high-noise datasets. We re-purpose prioritized experience sampling to locate expert-level demonstrations among millions of low-performance samples. This modification enables offline agents to learn state-of-the-art policies in benchmark tasks using datasets where expert actions are outnumbered nearly $65:1$.

1 Introduction

Reinforcement Learning (RL) promises to automate decision-making and control tasks across a wide range of applications. However, there are many real-world problems for which the typical learning and exploration process is too expensive (e.g., robotics, finance) or too dangerous (e.g., healthcare, self-driving) to make RL a viable solution. Fortunately, there is hope that we can trade RL's reliance on exploration and online feedback for a new reliance on large fixed datasets, much like those that have driven progress in supervised learning over the last decade [40, 30, 29]. In many situations, we can collect experience by recording the decisions of human demonstrators or existing control systems. For example, we can collect data for self-driving systems by mounting cameras and other sensors onto human-driven vehicles. However, as the scale of the dataset collection process grows, it becomes increasingly impractical to verify the quality of the data we are collecting. Even worse: the task may be challenging enough that no quality demonstrations could ever exist; instead, fleeting moments of optimal decision-making occur by chance in a sea of noise collected by a large number of agents.

Our goal is to build agents that can learn to pick out valuable information in large, unfiltered datasets containing experience from a mixture of sub-optimal policies. Recent work has proposed new ways to mimic the best demonstrations in a dataset [32, 45, 44, 36, 6]. In this paper, we analyze and address some of the problems that arise in determining which demonstrations are worth learning. We build a custom dataset of experience from popular continuous control tasks to demonstrate the challenge of learning from datasets with few or zero successful trajectories and provide an easy-to-implement solution that lets us learn expert policies from millions of low-performance samples.

2 Background and Related Work

In the interest of space, we assume the reader is familiar with Deep RL, continuous control tasks, and the general off-policy actor-critic framework. For an introduction, please refer to [31] and [13].

2.1 Offline Reinforcement Learning

Offline (or "Batch") RL is a subfield that deals with the special case of learning from static demonstration datasets. In offline RL, our goal is to discover the best policy given a dataset of fixed experience without the ability to explore the environment. The naive application of off-policy RL methods to the fully offline setting often fails due to distribution shift between the dataset and test environment [30]. Solutions based on approximate dynamic programming are also vulnerable to exploding Q-values due to the propagation of overestimation error with no opportunity for correction in the online environment [24].

Modern approaches to offline RL broadly involve modifications to standard off-policy algorithms to reduce overestimation and minimize distributional shift. REM uses an ensemble of critics to reduce overestimation error [2]. UWAC [46] uses the uncertainty of value predictions to reduce error propagation. CQL [23] penalizes out-of-distribution Q-values to encourage in-distribution actions. MOPO [49] is a model-based method that penalizes actions that leave the data distribution that the model was trained on.

2.2 Advantage-Filtered Behavioral Cloning

While much progress has been made in adapting off-policy RL to the constraints of the offline setting, these methods often struggle to outperform simple Behavioral Cloning (BC) [37, 12]. In BC, we train an actor network to replicate the actions taken in the demonstration dataset:

$$\mathcal{L}_{actor} = \underset{(s,a)\sim\mathcal{D}}{E} \left[-\log\pi_\theta(a|s)) \right] \tag{1}$$

Behavioral Cloning's inability to outperform its demonstrations becomes a major shortcoming when the dataset contains sub-optimal experience. Ideally, we would discard trajectories that contain low-quality demonstrations and only clone the best behavior available. This is the core idea behind Filtered Behavioral Cloning (FBC). FBC compares demonstrations to discard or down-weight the advice of sub-optimal policies. A reasonable metric for comparison is the advantage function, defined as $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, which represents the change in expected return when taking action a instead of following the current policy. Filtering experience by advantage is essentially a counterfactual query - we compute how much better action a *would have been* than the action chosen by our policy. We will refer to BC methods that filter experience by advantage as Advantage-Filtered Behavioral Cloning (AFBC). Because we do not have access to the true Q^π or V^π functions, we need to estimate the advantage. There are two main approaches:

1. **Monte Carlo Advantages** define an estimate of the advantage $\hat{A}^\pi(s,a) = \hat{\eta}(s,a) - V_\phi(s)$, where $\hat{\eta}(s,a)$ represents the empirical expected return of trajectories in the buffer that contain this (s,a) pair and $V_\phi(s)$ represents a learned state value function parameterized by a neural network with weights ϕ.

2. **Q-Based Advantages** define an estimate of the advantage $\hat{A}^\pi(s,a) = Q_\phi(s,a) - E_{a'\sim\pi(s)}[Q_\phi(s,a')] \approx Q_\phi(s,a) - \frac{1}{k}\Sigma_{i=0}^k Q_\phi(s,a' \sim \pi_\theta(s))$. We typically learn Q_ϕ as in standard off-policy actor-critics. The BC loss prevents the actor policy from diverging from the demonstration policy and reduces the risk of overestimation error.

Once we have a reliable estimate of the advantage function, we add a "filter" to the BC loss (Eq 1):

$$\mathcal{L}_{actor} = \underset{(s,a)\sim\mathcal{D}}{E} \left[-f(\hat{A}^\pi(s,a))\log\pi_\theta(a|s)) \right] \tag{2}$$

There are two common choices of f:

1. **Binary Filters** define $f(\hat{A}(s,a)) = \mathbb{1}_{\{\hat{A}^\pi(s,a)>0\}}$. This creates a boolean mask that eliminates samples which are thought to be worse than the current policy [45].

2. **Exponential Filters** define $f(\hat{A}(s,a)) = \exp(\beta\hat{A}^\pi(s,a))$, where β is a hyperparameter. The advantage values are often clipped or normalized for numerical stability [44, 32].

The methods implemented and expanded upon in this paper are most directly related to CRR [45] and AWAC [32], as discussed in Section 4.1. AWAC and CRR are roughly concurrent publications
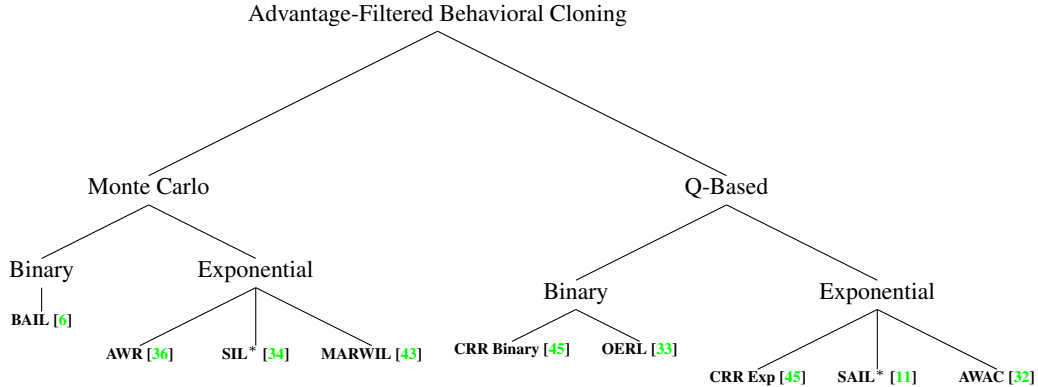
Figure 1: A rough classification of the recent Advantage-Filtered Behavioral Cloning literature, which contains a surprising amount of overlap due to concurrent publication and changes in motivation/experimental focus. * SIL and SAIL use a clipped rather than an exponential advantage, but the loss is still decreasing in the magnitude of the advantage.

that arrive at a very similar method. CRR is focused on offline RL in the RL Unplugged benchmark [12], while AWAC is more concerned with accelerating online fine-tuning by pre-training on smaller offline datasets. These methods are the latest iteration of the core AFBC idea that has appeared many times in recent literature. This section is partly an attempt to unify the literature and highlight a common theme that has been somewhat under-recognized. To the best of our knowledge, the first deep-learning-era AFBC implementation is MARWIL [43], which performed BC on samples re-weighted by their Monte Carlo advantage estimates and applied the technique to environments like robot soccer and multiplayer online battle arenas. AWR [36] is a very similar method that focuses on continuous control tasks. BAIL [6] learns an upper-envelope value function that encourages optimistic value estimates, and therefore conservative advantage estimates. Samples are filtered based on heuristics of their advantage relative to the rest of the dataset (e.g. with advantage larger than x or larger than $x\%$ of all samples, where x becomes a hyperparameter). In [33], demonstration datasets accelerate off-policy learning by providing an expert action when its value exceeds that of the agent's policy (according to the critic network) - essentially creating a deterministic binary filter. SIL [34] combines an AFBC-style loss with the standard policy gradient in an online setting, where the BC step greatly improves sample efficiency. SAIL [11] adds Q-based advantages to SIL and corrects outdated MC estimates by replacing them with parameterized estimates when they become too pessimistic. [14] gives another approach to self-imitating policy gradients for sample-efficient (online) continuous control. SQIL [38] and ORIL [50] use standard off-policy optimization with a modified reward function that encourages the agent to stay in (or return to) (s, a) pairs that are covered by the dataset, thereby helping it recover from distributional shift. We provide a rough classification of the most relevant modern literature according to the tree diagram in Figure 1.

Imitation Learning [35] also deals with learning from sub-optimal demonstrations. While the goals and experiments are similar to the offline RL setting, the methods are typically quite different. See VILD [42], IC-GAIL, and 2IWIL [47] for recent examples.

3 Datasets for High-Noise Offline RL

Our goal is to investigate the performance of AFBC as the quality of the dataset decreases, and useful demonstrations are hidden in sub-optimal noise. The first step is to gather those fixed datasets and create a systematic approach to varying the quantity and quality of demonstrations provided to our agents. Learning from sub-optimal data is a widely recognized challenge for offline RL, and as such it is a key component of recent benchmarks, including D4RL [12], RL Unplugged [15], and NeoRL [37]. However, these benchmarks do not give us enough control over the demonstration dataset for our purposes. Of these, NeoRL and D4RL are closest to what we are looking for - both offer low, medium, and expert performance datasets of varying sizes but are not large enough for some of our experiments and only offer those three tiers of quality. In contrast, RL Unplugged contains enormous datasets dumped from agent replay buffers throughout training but gives us less control over the quality of any given subset of that data. In an effort to get the best of both worlds, we collect our own offline RL dataset in the following manner:

| Dataset Name	Very Bad (0-199)	Bad (200-399)	Okay (400-699)	Great (700-899)	Expert (900+)	Total Samples:
Great-Expert	0	0	0	750,000	750,000	1,500,000
Ok-Expert	0	0	500,000	500,000	500,000	1,500,000
Bad-Expert	0	375,000	375,000	375,000	375,000	1,500,000
VeryBad-Expert	300,000	300,000	300,000	300,000	300,000	1,500,000
Signal 600k	600,000	450,000	200,000	150,000	100,000	1,500,000
Signal 1M	1,000,000	600,000	200,000	150,000	100,000	2,050,000
Signal 3M	3,000,000	1,000,000	200,000	150,000	100,000	4,450,000
Signal 4.5M	4,500,000	2,000,000	200,000	150,000	100,000	6,950,000
Stitching	4,500,000	2,000,000	0	0	0	6,500,000
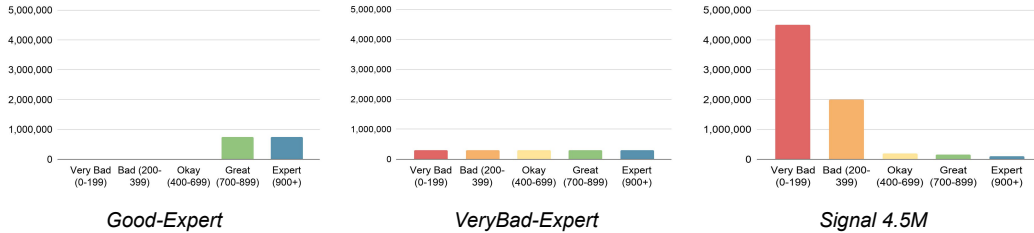
Figure 2: **Distribution of samples in all** 9 **dataset types.** We begin with near-optimal BC data and slowly shift the distribution towards low-performance noise.

1. We train online agents in 5 tasks from the DeepMind Control Suite. The D4RL and RL Unplugged authors stress the importance of diverse policies and state-space coverage. With that in mind, we train multiple policies per environment using a mixture of algorithms and hyperparameters. More specifically, we use (deterministic) TD3 [13] and several variants of (stochastic) SAC [17], implemented in [3].

2. We pause at regular intervals during training and record the policies' behavior for 50 episodes of environment interaction. To broaden our state-action space coverage, we sample from stochastic policies (rather than taking the mean, as is standard for test-time policy evaluation). The average undiscounted return of the policy is saved alongside the (s, a, r, s', d) tuples.

3. The saved experience is split into 5 levels of performance. The DMC tasks have returns in the range $0 - 1000$, and we divide the dataset according to the average return of the policy at the time they were recorded. The final result is a large dataset with millions of samples and five levels of performance. Full size and quality information is listed in Appendix A Figure 8.

From this raw data, we create a series of 9 offline RL datasets for each task. The size and makeup of each dataset is listed in Figure 2. *Great-Expert* splits $1,500,000$ samples evenly across expert and high-performance demonstrations. This represents a near best-case scenario for behavioral cloning methods in which very little data needs to be actively ignored. *Okay-Expert* begins a trend where the same $1,500,000$ sample budget is allocated to lower quality data, making vanilla BC less and less effective. This concludes with *VeryBad-Expert*, where samples are split evenly across all 5 performance bins. The next 4 datasets limit expert experience to just $100,000$ samples, which our experiments show is typically enough to train successful policies without overfitting. The challenge is finding those $100,000$ samples in a buffer of millions of poor demonstrations. The most difficult of these is *Signal 4.5M*, where sub-optimal data crowds the dataset at a ratio of almost $65 : 1$. Finally, we test the agent's ability to extract knowledge from pure noise. In the *Stitching* task, the agent is given $6,500,000$ samples of experience from agents that perform only slightly better than random policies and must learn to identify the few moments of optimal decision making and combine them into a single policy.

We benchmark the challenge of Behavioral Cloning in continuous control tasks by training BC agents on our high-noise and multi-policy datasets. The results are shown in Appendix D Figure 12. BC often fails to make progress, even with the *Great-Expert* datasets, presumably because of distribution shift or mixed-policy learning. We see a sharp decline in BC performance as the dataset fills with noisy demonstrations.

4 Challenges in AFBC

4.1 AFBC Baseline and Implementation Details

We implement a custom AFBC baseline combining elements of AWAC and CRR. Monte Carlo advantage estimates can work well in practice but limit our ability to learn from low-performance data by creating pessimistic advantage estimates. Therefore, we adopt the Q-Based estimates from CRR and AWAC, using critic networks trained similarly to SAC [17] and TD3 [13].

$$\mathcal{L}_{critic} = \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}} \left[\frac{1}{2} \sum_{i=1}^{2} (Q_{\phi,i}(s,a) - (r + \gamma(\min_{j=1,2} Q_{\phi',j}(s',\tilde{a}')))^2 \right], \tilde{a}' \sim \pi_\theta(s') \quad (3)$$

The advantage estimate is computed using 4 action samples:

$$\hat{A}^\pi(s,a) = \frac{1}{2} \sum_{j=1,2} Q_{\phi,j}(s,a) - \frac{1}{4} \sum_{i=0}^{4} (\frac{1}{2} \sum_{j=1,2} Q_{\phi,j}(s, a' \sim \pi_\theta(s))) \quad (4)$$

Note that we do not use distributional critics as in CRR - nor do we use the the critic weighted policy technique at test time. The policy is a tanh-squashed Gaussian distribution, implemented as in [48]. We use a single codebase for all experiments in order to control for small implementation details [19], and have extensively benchmarked our code against existing results in the literature [12, 37]. See Appendix B for more implementation details and a full list of hyperparameters.

4.2 Binary vs. Exponential Filters

While the exponential filter has been used with great success in prior work, it comes with two non-trivial implementation challenges. First, we need to deal with the magnitude of advantages across different environments. Advantage estimates are dependent on the scale of rewards, which can vary widely even across similar tasks in the same benchmark (e.g., Gym MuJoCo [5]). There are plenty of reasonable approaches to solving this. The simplest is to clip advantages in a numerically stable range, but this runs the risk of losing the ability to differentiate between high-advantage actions. MARWIL keeps a running average and normalizes the advantages inside the filter. The AWAC codebase considers several alternatives, including softmax normalization. An interesting alternative is PopArt [18, 20]; by standardizing the output of our critic networks we rescale advantages and get the benefits of PopArt's stability and hyperparameter insensitivity for free. The second challenge is the temperature hyperparameter β. Prior work demonstrates significant changes in performance across similar β values [43], and is often forced to use different settings in each domain [32].

We demonstrate the extent of the problem by implementing 9 reasonable variants and evaluating them on tasks from the D4RL benchmark. We use D4RL instead of our custom datasets in order to validate our implementation and compare against previously published results. The results are shown in Figure 3. There is very little correlation between a setting's relative return in one task and its performance in the others, making it difficult to set a high-performance default a priori. A change in β is enough to take us from near-failure to state-of-the-art performance. We argue that this could be considered a major shortcoming when benchmarking research and dealing with real-world problems, especially in a field that already suffers from significant implementation issues [10, 19]. For this reason, all following experiments use a binary filter.

4.3 Advantage Distributions and Effective Batch Size

In an effort to get a better understanding of the dynamics of the AFBC algorithm, we track the distribution of advantages in the dataset throughout training. Example results on the *VeryBad-Expert* datasets are shown in Figure 4. Using data from our experiments, we identify three challenges with current AFBC methods:

Effective Batch Size. Regardless of dataset quality, there are many samples with negative advantage. In a typical task, the critic networks assign a negative advantage to more than half of the dataset - and that number can rise as high as 90% in some cases. This means that we rely on a small fraction of each batch to compute our gradients and improve the actor. The effect is most apparent when using

walker2d-medium-expert-v0		hopper-medium-expert-v0		halfcheetah-medium-expert-v0	
PopArt, β=10	4509	PopArt, β=3	3622	Clip, β=3	10280
Clip, β=3	4446	Clip, β=1/2	3577	Clip, β=1/2	7942
Clip, β=1	4203	Clip, β=1	3575	Clip, β=1	6109
Clip, β=1/2	4138	PopArt, β=5	3496	PopArt, β=3	5158
PopArt, β=5	1817	Clip, β=3	3326	Norm, β=1	4252
Norm, β=1	1281	Norm, β=1	458	Norm, β=10	4135
PopArt, β=3	1046	Norm, β=100	393	Norm, β=100	4091
Norm, β=10	899	Norm, β=10	250	PopArt, β=10	3852
Norm, β=100	639	PopArt, β=10	26	PopArt, β=5	3575
D4RL		*D4RL*		*D4RL*	
Reference	*3747*	*Reference*	*3621*	*Reference*	*7750*

Figure 3: **The implementation challenges of exponential filters.** We test 9 exponential-filter variants on three tasks from the D4RL benchmark. The scores are listed in decreasing order; there is little correlation between a filter's rank in one task and its rank in the others. The 'D4RL Reference' row lists the highest reported score of any algorithm from the D4RL results [12], to provide some context.
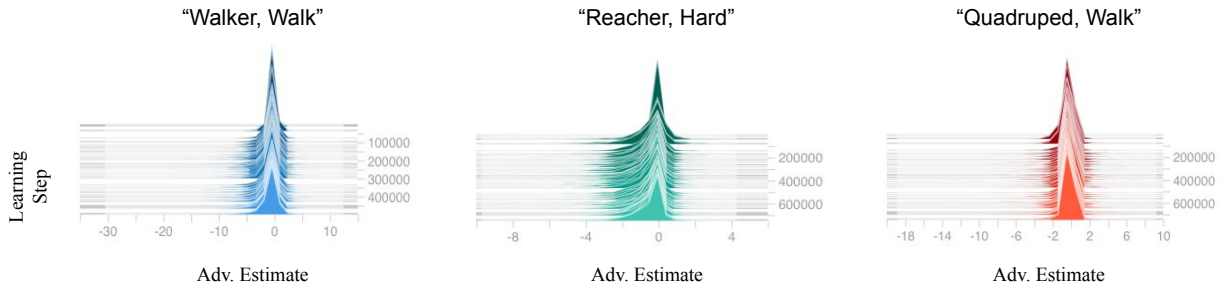

Figure 4: **Distribution of advantage (adv.) estimates throughout training.** We estimate the advantage of a large sample of demonstrations from the replay buffer on the *VeryBad-Expert* datasets for three control tasks where AFBC performs well. These figures were generated for every task and dataset combination with similar results.

binary filters, where we may be entirely disregarding a large portion of each minibatch. Exponential filters attribute at least some learning signal to each sample, but we are likely to be down-weighting most of our batch. The AFBC baselines show that learning is still possible at low signal-to-noise ratios, but we are sacrificing stability by reducing our network's effective batch size. In noisy datasets (*Signal-4.5M*), advantageous samples become so rare that a uniform sample of data yields batches with prohibitively high variance.

Noisy Labels. Advantage estimates are highly concentrated at low absolute values. Many of the demonstrations have advantage estimates that oscillate close enough to zero that they can be labeled as positive or negative based on randomness in the estimator.

Static Advantage Distributions. The advantage distribution does not change very much over time. The estimator spends the first few thousand learning steps adapting to the task's reward scale but makes few adjustments over the remaining steps. A priori, we might expect the agent to clone most or all of the dataset before becoming more confident in its ability to identify the best strategies. Instead, the estimator quickly learns to clone a small percentage of the dataset.

4.4 Addressing Effective Batch Size with Prioritized Sampling

As discussed above, learning from sub-optimal datasets with low-advantage actions reduces our actor network's batch size. We can improve by sampling batches that are more likely to be accepted by the advantage filter. A straightforward way to implement this is to re-purpose Prioritized Experience Relay (PER) [41] to sample high-advantage actions rather than high-error bellman backups. The critic update proceeds as usual - sampling uniformly from the replay buffer and minimizing temporal

difference error across the entire dataset. This gives us the opportunity to re-compute advantages and identify new useful demonstrations. During the actor update, we sample transitions from the buffer proportional to the advantage we computed when they were last used to train the critic. We can still filter low-performance samples, but this is much less likely to be necessary (see Figure 5). PER also seems to offer a partial solution to the noisy labels problem; for a sample to be presented to the filter inside of the actor update, it must have been assigned a positive advantage at some point in the recent past. Before we clone the action, it then has to be labeled positive *again* - reducing our ability to be fooled by the large group of near-zero-advantage actions.

Prioritized Experience Replay Details: The Prioritized Experience Replay implementation is based on OpenAI Baselines [9]. The replay is given the option of sampling uniformly from the underlying buffer or using prioritized sampling. We sample uniformly during the critic update and then update the samples' priorities according to $\max(\hat{A}^\pi(s,a), \epsilon)$, where ϵ is a small positive constant. We also experiment with binary priority weights ($\mathbb{1}_{\{\hat{A}^\pi(s,a) \geq 0\}} + \epsilon$). See Appendix D Figure 10 for a brief comparison. We do not use any importance sampling weights. This is a key difference between our use-case and the traditional use of PER. Normally, we prioritize samples based on a value that is highly correlated with their effect on the gradient (e.g., absolute TD error) and need to use importance weights to compensate for sampling the highest priorities. This priority system decreases our chances of sampling actions that our filter will discard but does not suffer from the same skewed gradient values. As the actor takes gradient steps in the direction of the approved experience, its action probabilities center around the provided action until the advantage drops to zero. In order to keep track of this effect, we re-compute the priorities of the sampled actions after each actor update.
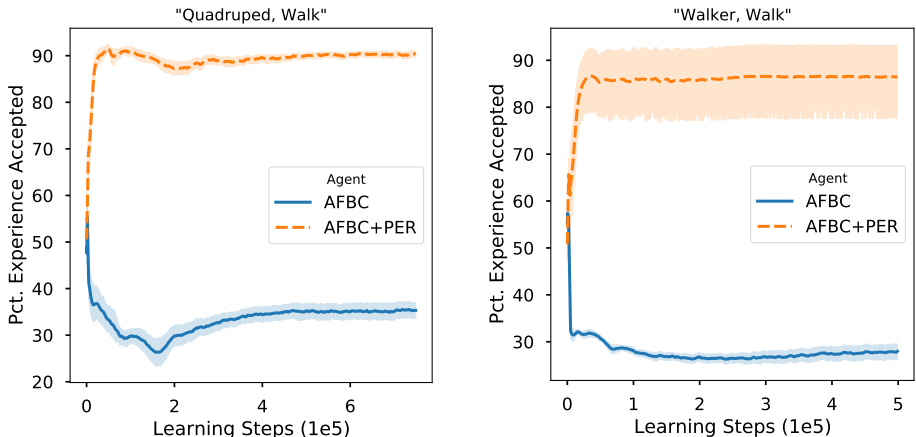

Figure 5: **Effective batch size of AFBC agents**. We measure the percentage of actor update batches that are approved by a binary filter throughout training. Curves represent the mean and 95% confidence interval across 5 random seeds in all 8 datasets that contain expert experience. PER samples actions that are more likely to have positive advantage, meaning that fewer samples are masked or down-weighted by our advantage filter.

This simple trick leads to a dramatic improvement in performance on the *Signal* datasets. Results are listed in Figures 6 and 11. Simply put: if AFBC can learn a high-quality policy, AFBC with the PER trick can maintain that performance despite an enormous amount of noise. This technique also increases performance in the *Stitching* task, thanks to its enhanced ability to identify the rare optimal decisions of random or near-random policies. It also risks overfitting and instability on the high-quality datasets; this can be corrected by adjusting the α prioritization parameter of the experience replay. Low α values reduce PER to uniform sampling. These experiments use $\alpha = .6$.

4.5 Learning from Worst-Case Experience

One question that arises when working with large sub-optimal datasets is what may happen in the worst-case scenario where much of the dataset is intentionally misleading. In this setting, we lose the ability to stitch together random policies and must instead learn to isolate the expert data and ignore everything else. To research this situation, we need a domain where humans can easily intuit about

7

		Great-Expert	Ok-Expert	Bad-Expert	VeryBad-Expert	Signal 600k	Signal 1M	Signal 3M	Signal 4.5M	Stitching
"Walker, Walk"	*BC*	795 ± 24	616 ± 45	422 ± 36	341 ± 37	112 ± 15	87 ± 17	62 ± 8	69 ± 10	44 ± 10
	AFBC	834 ± 37	843 ± 39	**828 ± 33**	839 ± 74	461 ± 69	586 ±137	672 ± 121	502 ± 23	467 ± 21
	AFBC+PER	**859 ± 36**	**898 ± 36**	812 ± 98	**856 ± 54**	**625 ± 180**	**826 ± 90**	**917 ± 20**	**828 ± 51**	**538 ± 42**
"Reacher, Hard"	*BC*	134 ± 44	86 ± 39	49 ± 19	33 ± 17	19 ± 11	20 ± 12	15 ± 10	17 ± 12	26 ± 11
	AFBC	**451 ± 347**	**593 ± 296**	**916 ± 28**	**947 ± 16**	944 ± 15	954 ± 10	952 ± 9	947 ± 13	936 ± 18
	AFBC+PER	137 ± 215	157 ± 164	905 ± 30	917 ± 38	**946 ± 22**	**964 ± 11**	**956 ± 19**	**961 ± 12**	**945 ± 22**
"Quadruped, Walk"	*BC*	689 ± 75	287 ±28	244 ± 18	212 ± 20	125 ± 18	101 ± 21	107 ± 15	99 ± 16	101 ± 17
	AFBC	**729 ± 66**	**679 ± 63**	570 ±82	482 ± 39	471 ± 32	455 ± 28	482 ± 27	540 ± 52	393 ± 174
	AFBC+PER	658 ± 144	676 ± 201	**886 ± 33**	**825 ± 99**	**828 ± 62**	**878 ± 37**	**841 ± 38**	**728 ± 194**	**746 ± 70**

Figure 6: **AFBC in High-Noise Continuous Control Tasks.** Prioritized sampling allows AFBC to continue to learn high-performance policies when uniform sampling is overwhelmed by noise. We use a consistent set of hyperparameters for each task, and report the mean and 95% confidence interval across 5 random seeds. Results on more difficult tasks in Figure 11.

the optimal strategy and identify the worst possible policy. Therefore, we will take a brief detour from the high-dimensional robotic control tasks of the DMC Suite and consider the classic "Mountain Car" task in which an under-powered car learns to gain momentum by going backward to summit a large hill. The environment is pictured in Figure 7a. The agent receives a large positive reward $+100$ for reaching the goal flag on top of the mountain, with a small penalty for fuel expenditure along the way. The worst possible solution is to gain the speed necessary to climb up the mountain before deciding to turn around and return to the starting position, thereby wasting as much fuel as possible. The default state space is a $(position, velocity)$ tuple, but we compress this information to a scalar representing the current position and direction of movement to plot the Q and A functions as a 3D surface. The action space is bounded in $[-1, 1]$, where positive actions accelerate the cart to the right, and negative actions accelerate it to the left.

The compressed state space makes the problem more difficult by hiding the velocity information necessary to manage fuel expenditure, but it is still possible to reach the goal flag. We collect a dataset of expert TD3 actions, alongside actions from a random policy. We also isolate random actions that display worst-case behavior where the cart is making progress up the mountain but reverses to move downhill. We create three offline datasets: expert demonstrations, a $9 : 1$ ratio of random and expert demonstrations, and a $9 : 1$ ratio of worst-case (or 'adversarial') actions. Figure 7b shows the results of BC, AFBC and AFBC+PER on these datasets. BC performs as expected: it learns from expert data but is distracted by random actions and is confused by the adversarial demonstrations. Default AFBC can ignore the random actions but still succumbs to the adversarial advice. We attribute this to Q-value inflation - a hypothesis discussed in the appendix of the CRR [45] paper; in short: the adversarial advice is so concentrated in a small region of (s, a) space that the bias of consistent Q-updates causes the critic to overestimate the advantage of the adversarial actions. AFBC with the PER trick is better equipped to handle this situation. By replaying experience proportional to its advantage, PER ignores actions that have a small positive advantage due to bias and clones more of the true expert data. AFBC+PER can solve the task based on the adversarial dataset, although the sparse reward has inherently high variance. The critic networks correctly minimize the advantage of the adversarial data (Figure 7c) and learn an accurate Q-function (Figure 7d).

5 Discussions & Conclusion

The ability to automatically learn high-performance decision-making systems from large datasets will open up exciting opportunities to safely and effectively apply Reinforcement Learning to the real world. There are many domains where we can find enough data to train large neural network policies but cannot verify the demonstrated actions' quality. Success hinges on our ability to answer counterfactual questions about the data: are the decisions made in the dataset the correct ones, or can we find a way to do better? Advantage-Filtered Behavioral Cloning offers an intuitive way to formalize and answer this question. In this paper, we have conducted a thorough empirical
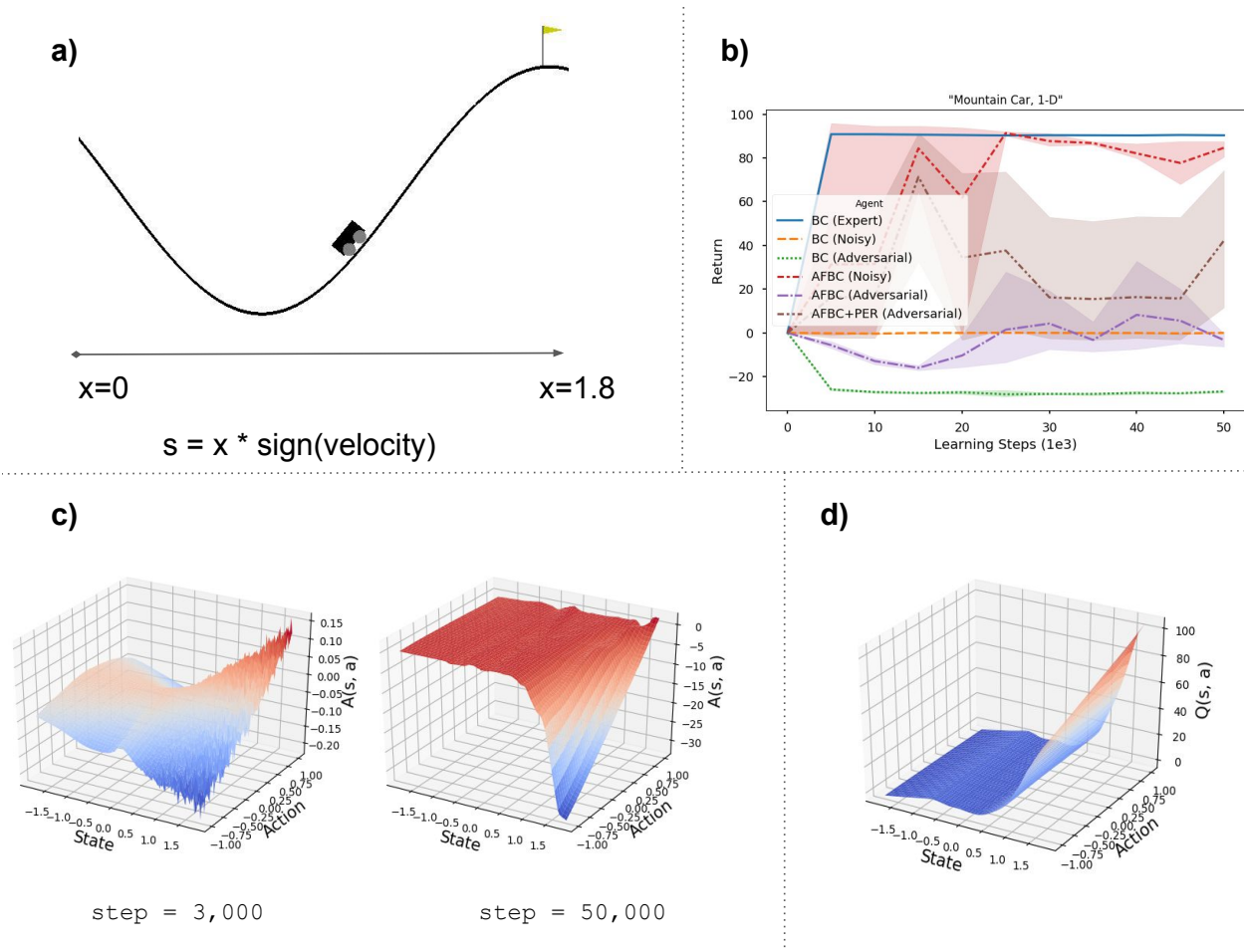
Figure 7: **Adversarial Demonstrations in Modified Mountain Car.** a) We create a version of the Mountain Car task that combines state and velocity information into a scalar state that is convenient for plotting. b) AFBC can solve the task despite a $9 : 1$ ratio of random actions and can even learn despite a dataset of worst-case demonstrations. c) We show the evolution of the learned advantage function across the full (s, a) space. d) The final Q-function.

study that attempts to unify existing techniques, identify critical obstacles, and provide assurance that this method can learn from unfiltered datasets of any size. However, the prioritized sampling method does not fully address the noisy label and static distribution problems discussed in Section 4.3. We experimented with several theoretical solutions that provided somewhat underwhelming improvements on our baseline tasks. Please see Appendix C for a thorough discussion of future directions.

Methods that can ignore or even improve using low-quality data are valuable because they simplify offline RL dataset collection by reducing the risk that additional data will damage the system. Adding more data is rarely unhelpful and is likely to increase performance. This creates an engineering situation similar to Deep Supervised Learning, where more data and a bigger model are never the wrong answer. Moving forward, we hope to combine this approach with the kinds of large network architectures and high-dimensional datasets that have spurred progress in Deep Supervised Learning to solve complicated tasks beyond simulated control benchmarks.

References

[1] Joshua Achiam. "Spinning Up in Deep Reinforcement Learning". In: (2018).

[2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. *An Optimistic Perspective on Offline Reinforcement Learning*. 2020. arXiv: `1907.04543 [cs.LG]`.

[3]   Anonymous. *RL Library Name Removed for Author Anonymity*. https://github.com/. 2020.

[4]   Oron Anschel, Nir Baram, and Nahum Shimkin. *Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning*. 2017. arXiv: 1611.01929 [cs.AI].

[5]   Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG].

[6]   Xinyue Chen et al. *BAIL: Best-Action Imitation Learning for Batch Deep Reinforcement Learning*. 2020. arXiv: 1910.12179 [cs.LG].

[7]   Xinyue Chen et al. *Randomized Ensembled Double Q-Learning: Learning Fast Without a Model*. 2021. arXiv: 2101.05982 [cs.LG].

[8]   Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. "Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 834–843. URL: http://proceedings.mlr.press/v70/chou17a.html.

[9]   Prafulla Dhariwal et al. *OpenAI Baselines*. https://github.com/openai/baselines. 2017.

[10]   Logan Engstrom et al. *Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO*. 2020. arXiv: 2005.12729 [cs.LG].

[11]   Johan Ferret, Olivier Pietquin, and Matthieu Geist. "Self-Imitation Advantage Learning". In: *arXiv preprint arXiv:2012.11989* (2020).

[12]   Justin Fu et al. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2021. arXiv: 2004.07219 [cs.LG].

[13]   Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1587–1596.

[14]   Tanmay Gangwani, Qiang Liu, and Jian Peng. *Learning Self-Imitating Diverse Policies*. 2019. arXiv: 1805.10309 [stat.ML].

[15]   Caglar Gulcehre et al. *RL Unplugged: A Suite of Benchmarks for Offline Reinforcement Learning*. 2021. arXiv: 2006.13888 [cs.LG].

[16]   Chuan Guo et al. *On Calibration of Modern Neural Networks*. 2017. arXiv: 1706.04599 [cs.LG].

[17]   Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[18]   Hado van Hasselt et al. *Learning values across many orders of magnitude*. 2016. arXiv: 1602.07714 [cs.LG].

[19]   Peter Henderson et al. *Deep Reinforcement Learning that Matters*. 2019. arXiv: 1709.06560 [cs.LG].

[20]   Matteo Hessel et al. *Multi-task Deep Reinforcement Learning with PopArt*. 2018. arXiv: 1809.04474 [cs.LG].

[21]   Eric Jones, Travis Oliphant, et al. "SciPy: Open source scientific tools for Python". In: ().

[22]   Aviral Kumar, Abhishek Gupta, and Sergey Levine. *DisCor: Corrective Feedback in Reinforcement Learning via Distribution Correction*. 2020. arXiv: 2003.07305 [cs.LG].

[23]   Aviral Kumar et al. *Conservative Q-Learning for Offline Reinforcement Learning*. 2020. arXiv: 2006.04779 [cs.LG].

[24]   Aviral Kumar et al. "Stabilizing off-policy q-learning via bootstrapping error reduction". In: *arXiv preprint arXiv:1906.00949* (2019).

[25]   Arsenii Kuznetsov et al. *Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics*. 2020. arXiv: 2005.04269 [cs.LG].

[26]   Michail G Lagoudakis and Ronald Parr. "Reinforcement learning as classification: Leveraging modern classifiers". In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 424–431.

[27]   Qingfeng Lan et al. *Maxmin Q-learning: Controlling the Estimation Bias of Q-learning*. 2020. arXiv: 2002.06487 [cs.LG].

[28]   Kimin Lee et al. *SUNRISE: A Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning*. 2020. arXiv: 2007.04938 [cs.LG].

[29] Sergey Levine. "Deep Reinforcement Learning in the Real World". Workshop on New Directions in Reinforcement Learning and Control. 2019. URL: https://www.youtube.com/watch?v=b97H5uz8xkI.

[30] Sergey Levine et al. *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. 2020. arXiv: 2005.01643 [cs.LG].

[31] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. 2015. arXiv: 1509.02971 [cs.LG].

[32] Ashvin Nair et al. *Accelerating Online Reinforcement Learning with Offline Datasets*. 2020. arXiv: 2006.09359 [cs.LG].

[33] Ashvin Nair et al. *Overcoming Exploration in Reinforcement Learning with Demonstrations*. 2018. arXiv: 1709.10089 [cs.LG].

[34] Junhyuk Oh et al. *Self-Imitation Learning*. 2018. arXiv: 1806.05635 [cs.LG].

[35] Takayuki Osa et al. "An Algorithmic Perspective on Imitation Learning". In: *Foundations and Trends in Robotics* 7.1-2 (2018), pp. 1–179. ISSN: 1935-8261. DOI: 10.1561/2300000053. URL: http://dx.doi.org/10.1561/2300000053.

[36] Xue Bin Peng et al. *Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning*. 2019. arXiv: 1910.00177 [cs.LG].

[37] Rongjun Qin et al. *NeoRL: A Near Real-World Benchmark for Offline Reinforcement Learning*. 2021. arXiv: 2102.00714 [cs.LG].

[38] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. *SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards*. 2019. arXiv: 1905.11108 [cs.LG].

[39] Mamshad Nayeem Rizve et al. *In Defense of Pseudo-Labeling: An Uncertainty-Aware Pseudo-label Selection Framework for Semi-Supervised Learning*. 2021. arXiv: 2101.06329 [cs.LG].

[40] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV].

[41] Tom Schaul et al. *Prioritized Experience Replay*. 2016. arXiv: 1511.05952 [cs.LG].

[42] Voot Tangkaratt et al. "Variational Imitation Learning with Diverse-quality Demonstrations". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 9407–9417. URL: http://proceedings.mlr.press/v119/tangkaratt20a.html.

[43] Qing Wang et al. "Exponentially Weighted Imitation Learning for Batched Historical Data". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/4aec1b3435c52abbdf8334ea0e7141e0-Paper.pdf.

[44] Qing Wang et al. "Exponentially Weighted Imitation Learning for Batched Historical Data." In:

[45] Ziyu Wang et al. *Critic Regularized Regression*. 2020. arXiv: 2006.15134 [cs.LG].

[46] Yue Wu et al. "Uncertainty Weighted Offline Reinforcement Learning". In: 2020. URL: https://offline-rl-neurips.github.io/pdf/27.pdf.

[47] Yueh-Hua Wu et al. "Imitation learning from imperfect demonstration". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6818–6827.

[48] Denis Yarats and Ilya Kostrikov. *Soft Actor-Critic (SAC) implementation in PyTorch*. https://github.com/denisyarats/pytorch_sac. 2020.

[49] Tianhe Yu et al. *MOPO: Model-based Offline Policy Optimization*. 2020. arXiv: 2005.13239 [cs.LG].

[50] Konrad Zolna et al. *Offline Learning from Demonstrations and Unlabeled Experience*. 2020. arXiv: 2011.13885 [cs.LG].

A   Dataset Details

We provide a listing of the quantity of available Q-learning samples for each environment in Figure 8. Figure 2 shows the breakdown of how samples are distributed to generate datasets of varying quality.

	Transitions Per Performance Level				
Task Name	"Very Bad" (0-199)	"Bad" (200-399)	"Okay" (400-699)	"Great" (700-899)	"Expert" (900+)
Walker, Walk	4,800,000	2,200,000	2,750,000	1,550,000	3,700,000
Cheetah, Run	5,400,000	2,850,000	3,650,000	7,900,000	2,450,000
Fish, Swim	6,900,000	2,750,000	6,500,000	3,900,000	0
Reacher, Hard	10,300,000	2,350,000	2,650,000	2,650,000	5,800,000
Quadruped, Walk	5,750,000	3,350,000	3,750,000	4,600,000	3,050,000

Figure 8: **Available dataset sizes.** Quantity of Q-learning samples available for each performance range across the 5 control tasks studied.

B   Implementation Details

B.1   Advantage Weighted Actor-Critic and Critic Regularized Regression Baselines

Hyperparameters and implementation details are listed in Table 1.

Policy Log Std Range	[-10, 2]
Target Delay	2
Weight Decay	None
Gradient Clipping	None
Actor LR	1e-4
Critic LR	1e-4
Eval Interval	5000
Eval Episodes	10
Buffer Size	Size of Dataset
Gamma	0.99
Tau	0.005
Batch Size	512
Gradient Steps	750k
Max Episode Steps	1000
Action Bound	[-1, 1]
Network Architecture	(256, ReLU, 256, ReLU)

Table 1: Default hyperparameters for the main DMC experiments.

We use a tanh-squashed Gaussian distribution for the actor, implemented as in [48]. We also ran trials on every dataset with the implementation in [1], a custom implementation of the Beta distribution described in [8], as well as some shorter runs with several other publicly available implementations. The results were surprisingly mixed. The distribution parameterization can be a critical implementation detail because unlike standard online approaches, the AFBC algorithm often requires us to compute log probabilities of foreign actions far outside the center of our actor's own distribution. Numerically stable log prob computations are challenging, and implementations designed for online algorithms may have had no reason to test for stability in this context. Even successful implementations typically return large log prob values early in training. This is not an issue as long as they quickly stabilize to a reasonable range, and we use some protective clipping (e.g. at clearly unstable values like $\log \pi_\theta(a) \in (-1000, 1000)$) to minimize the damage.

B.2   PopArt

PopArt is implemented as described in [18] and [20]. We use an adaptive step size when computing the normalization statistics in order to reduce reliance on initialization. The ν value is initialized to a large positive constant to improve stability (see [18] Pg 13).

B.3 Evaluation

The scores displayed in the tables above are computed by:

1. Smoothing the learning curve with a polyak coefficient of .65.

2. Determining the mean and standard deviation of several (usually 5) smoothed learning curves from different random seeds. This creates a lower-variance learning curve.

3. Reporting the average return and two standard deviations of our low-variance learning curve over the last 10 evaluations of training.

C Alternative Binary Filters, Enhanced Critic Updates, and Future Directions

While prioritized sampling is an effective solution to small effective batch sizes, we would still like to learn more accurate advantage estimates and dynamic acceptance curves. In this section, we describe several enhancements in hopes of furthering future research.

Annealed Binary Filters with Statistical T-Tests. Binary filters appear to rush towards an experience approval percentage that is overly pessimistic early in training. One way to address this is to introduce an overly optimistic filter and adjust its approval criteria during the learning process. A naive way to approach this is to approve samples above some low advantage threshold a_0 and increase that threshold over time, or to approve samples with an advantage higher than $x_0\%$ of the dataset and increase that threshold as learning progresses. However, it is challenging to set those hyperparameters across different tasks. We design a binary filter that uses a pairwise statistical T-test to create a much more task-invariant hyperparameter. We estimate the advantage of a sample k times and then estimate the advantage of actions recommended by the current policy k times, and only approve the dataset action for cloning if its advantage is greater than that of the current policy with statistical confidence p_t, where p_t can be annealed from 1.0 to high confidence .05. Experiments show that this does smooth the experience approval curve, but the final performance only matches that of the standard binary filter in the control tasks we study. This approach is inspired by [26], and implemented using the statistics functions in [21].

Annealed Binary Filters with Advantage Classifier Networks. A second candidate solution to the same problem is to train an ensemble of networks that attempt to classify the advantage of actions as positive or negative. We can then use our ensemble's mean confidence and uncertainty to make informed decisions about experience approval - inspired by techniques in self-supervised learning and pseudo-labeling [39, 16]. This helps us deal with the noise surrounding the large fraction of samples near zero advantage because pseudo-labels have to be positive for several gradient updates before the ensemble will agree on their approval. We implement this using a similar network architecture to existing critic networks but with a sigmoid output. Once again, performance is essentially identical to the AFBC+PER baseline; the DMC tasks studied are simple enough that techniques either perform very well or fail. We think it is likely that this technique could be successfully applied to a different domain.

Accurate and Sample-Efficient Critics with REDQ and Weighted Bellman Updates. The improvement of AFBC over standard BC rests on our ability to accurately estimate the advantage of (s, a) pairs, which makes the method vulnerable to bias in the critic learning process. Critic overestimation and error propagation is a thoroughly investigated problem in recent work [13, 27, 25, 4, 22, 2]. REDQ [7] trains an ensemble of critic networks on target values generated by a random subset of that ensemble and provides an effective bias-variance trade-off. It is also shown to allow for many more gradient updates per sample, something that could be useful in the offline RL context where the size of our dataset is fixed. Our main experiments are focused on high-noise datasets containing millions of samples, which makes overfitting a secondary concern, but this could be a key feature when working with small datasets. Another approach to managing overestimation error is to minimize the impact of uncertain target values in the temporal difference update. We can use REDQ's critic ensemble to estimate target uncertainty and down-weight bellman backups when s' is out-of-distribution. We implement a modified version of the SUNRISE [28] critic loss, where we

weight samples in the critic update according to a normalized uncertainty metric:

$$\mathcal{L}_{critic} = \mathcal{L}_{critic} * \text{softmax}(\tau \text{Std}(Q_{\{\theta_i | i \in \{0,1,...n\}\}}(s', a' \sim \pi(s')))) \tag{5}$$

Where τ is a temperature hyperparameter. Figure 9 displays the mean Q value of the critic networks in two sample-restricted "Walker, Walk" datasets. Interestingly, critic bias typically results in uncontrollable *under*-estimation in the AFBC context because actors are restricted to in-distribution actions, and the bias is caused by the min function in the target computation rather than over-exploitation of positive bias by a policy gradient update. REDQ prevents Q-function collapse and allows for more stable learning despite a very small offline dataset.
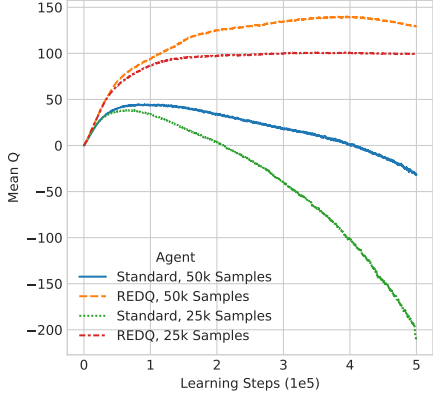

Figure 9: **Critic Ensembles in Low-Sample Datasets.** We compare the Q-function learned by a REDQ ensemble with weighted updates to that of standard clipped double-q learning in a version of *"Walker, Walk" VeryBad-Expert* with just 25k and 50k transitions. The smaller dataset can cause runaway under-estimation, but the more advanced critic techniques minimize this effect.

D   Additional Results

This sections contains additional figures referenced in the main text that were deferred to the appendix due to the page limit.

	Reacher GE	Reacher OGE	Reacher BOGE	Reacher VBOGE	Reacher Signal 600k	Reacher Signal 1M	Reacher Signal 3M	Reacher Signal 4.5M	Walker GE	Walker OGE	Walker Signal-1M	Walker Signal 3M	Walker Signal 4.5M	Walker Stitching
Advantage Priorities	137 ± 215	157± 164	**905 ± 30**	917 ± 38	946 ± 22	**964 ± 11**	956 ± 19	**961 ± 12**	**859 ± 36**	**898 ± 36**	826 ± 90	**917 ± 20**	828 ± 51	**538 ± 42**
Binary Priorities	**318 ± 389**	**376 ± 385**	844 ± 90	**950 ± 14**	**948 ± 20**	958 ± 11	**961 ± 8**	950 ± 15	803	661	792 ± 78	659 ± 83	**838 ± 38**	528 ± 30

Figure 10: **A comparison of binary and clipped advantage prioritization schemes.** The returns are very comparable. We suspect this is because the advantage values across the dataset are often small and tightly distributed; as a result, the binary priorities are actually quite similar to the true advantage weighting. Reported scores are the mean and 95% confidence interval across 5 random seeds. "VBOGE" stands for "Very Bad, Bad, Okay, Good, Expert" - referring to the distribution of dataset samples. In the main text this dataset is abbreviated "*VeryBad-Expert*". The same pattern applies to "BOGE", "OGE", and "GE".

Fish, Swim	Okay Great	Bad Great	VeryBad Great	Signal 600k	Signal 1M	Signal 3M
BC	86 ± 4	81 ± 2	78 ± 2	78 ± 3	79 ± 3	79 ± 3
AFBC	210 ± 48	87 ± 13	77 ± 9	76 ± 7	76 ± 9	78 ± 7
AFBC+PER	**319 ± 81**	**321 ± 124**	**183 ± 43**	**199 ± 72**	**171 ± 71**	**241 ± 19**

Cheetah, Run	Great Expert	Okay Expert	Bad Expert	VeryBad Expert	Signal 600k	Signal 1M	Signal 3M	Signal 4.5M
BC	**349 ± 89**	**252 ± 34**	**299 ± 21**	**241 ± 25**	**218 ± 24**	200 ± 19	165 ± 25	185 ± 23
AFBC	100 ± 24	101 ± 26	169 ± 28	155 ± 28	209 ± 56	183 ± 23	249 ± 22	283 ± 34
AFBC+PER	84 ± 36	75 ± 26	192 ± 41	221 ± 60	178 ± 53	**253 ± 28**	**320 ± 54**	**346 ± 53**

Figure 11: **AFBC in Challenging Environments where BC Fails.** There are some environments where the offline datasets appear insufficient for learning. These results use the default hparams in Table 1. Reported scores are the mean and 95% confidence interval across 5 random seeds.
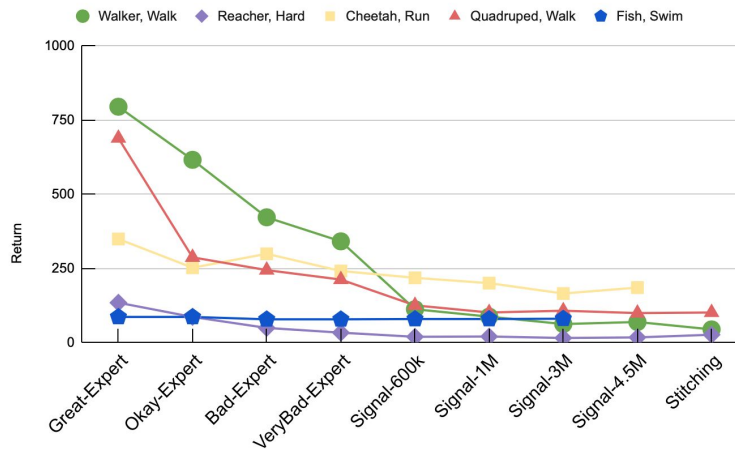

Figure 12: **Behavioral Cloning in High-Noise Datasets.** We train a BC agent on 9 datasets across each task. The reported return is the average of 5 random seeds. The horizontal axis is sorted by decreasing dataset quality. If BC can learn a successful policy from high-quality demonstrations, it fails to maintain its performance as the dataset fills with sub-optimal noise.