Distance Weighted Supervised Learning for Offline Interaction Data

Joey Hejna [1] **Jensen Gao** [1] **Dorsa Sadigh** [1]

Abstract

Sequential decision making algorithms often struggle to leverage different sources of unstructured offline interaction data. Imitation learning (IL) methods based on supervised learning are robust, but require optimal demonstrations, which are hard to collect. Offline goal-conditioned reinforcement learning (RL) algorithms promise to learn from sub-optimal data, but face optimization challenges especially with high-dimensional data. To bridge the gap between IL and RL, we introduce Distance Weighted Supervised Learning or DWSL, a supervised method for learning goal-conditioned policies from offline data. DWSL models the entire distribution of time-steps between states in offline data with only supervised learning, and uses this distribution to approximate shortest path distances. To extract a policy, we weight actions by their reduction in distance estimates. Theoretically, DWSL converges to an optimal policy constrained to the data distribution, an attractive property for offline learning, without any bootstrapping. Across all datasets we test, DWSL empirically maintains behavior cloning as a lower bound while still exhibiting policy improvement. In high-dimensional image domains, DWSL surpasses the performance of both prior goal-conditioned IL and RL algorithms. Visualizations and code can be found at https://sites.google.com/view/dwsl/home.

1. Introduction

Many advancements in deep learning are underpinned by a markedly similar formula: collect a large, diverse dataset and train a model with supervised learning. Thus, for sequential decision making problems it is naturally desirable to apply the same formula yet again. This is especially true when learning multi-task policies that operate on high-dimensional image observations, which is often the case for real-world robotics applications. Training generalizable robot policies will likely require scaling dataset size and model capacity, which supervised learning algorithms have excelled at.

One promising paradigm for scalable robot learning is offline goal-conditioned learning, where policies are trained to reach arbitrary goals from interaction data without dense reward annotation. This can be done via supervised learning with goal-conditioned imitation learning (IL) (Ghosh et al., 2021; Emmons et al., 2022), where dataset actions are cloned by a policy. IL methods tend to excel when given access to broad data in the form of expert demonstrations (Jang et al., 2022) or unstructured play (Lynch et al., 2019). However, collecting vast quantities of this type of data has proven to be tedious at best (Akgun et al., 2012), and any sub-optimalities within the data can be detrimentally materialized by the learned policy.

Value-based offline reinforcement learning (RL) algorithms, on the other hand, promise to address these challenges by learning policies that can improve upon the behavior found in sub-optimal datasets. However, RL methods can be difficult to get working in practice due the optimization challenges of temporal difference learning, known as the *deadly triad*, which consist of function approximation, off-policy learning, and bootstrapping (Van Hasselt et al., 2018; Sutton & Barto, 2018). These issues are further exacerbated in offline image-based settings where additional data cannot be collected to correct modeling errors, and high-dimensional visual observations make stitching together distinct experience difficult. As a result, despite complicated tricks and careful hyperparameter tuning, the performance of offline RL methods can vary dramatically from dataset to dataset. For example, while offline RL algorithms perform well on data collected by other RL agents, they have been known to fail on data collected by humans (Mandlekar et al., 2021) or exploratory strategies (Yarats et al., 2022).

Ideally, we want scalable algorithms that perform well on the widest variety of datasets. To make progress towards this goal, we propose an offline goal-conditioned algorithm that maintains the optimization robustness of supervised learning, but still performs value estimation to improve

[1] Department of Computer Science, Stanford University. Correspondence to: Joey Hejna <jhejna@cs.stanford.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

upon sub-optimal data. Prior work (Peng et al., 2019) have attempted to do the same with regression-based objectives, but were not designed for the goal-conditioned setting and in practice *still used boostrapping*.

Our key insight is to view goal-conditioned value prediction — which enables policy improvement beyond imitation — as *a supervised classification problem*. Instead of learning optimal values directly, we model the entire empirical distribution of discrete distances between states in offline data. Then, we compute meaningful statistics of this distribution to extract shortest path estimates between any two states without the optimization challenges of bootstrapping. Specifically, we use the LogSumExp as a smooth estimate of the minimum distance. Finally, we re-weight actions by how closely they reduce the estimated distance to the goal. We call our approach Distance Weighted Supervised Learning or DWSL. Unlike prior supervised approaches to RL, DWSL theoretically converges to an optimal policy constrained to the behavior distribution of the dataset, instead of doing only a single step of policy improvement.

Empirically, DWSL has several attractive properties that we believe contribute to its practical performance and broad applicability. DWSL does not require access to a dense reward function, goal labels, or optimal demonstrations. We find DWSL to be extremely robust to hyperparameters and datasets, which we posit is due to the usage of only supervised learning methods as subroutines. In our experiments, DWSL exceeds the performance of imitation learning on 23 of the 27 datasets we test *using the same algorithm-specific hyperparameters*. DWSL matches IL on 3 of the remaining 4. On 4 of 5 human-collected datasets and 6 of 8 image datasets, DWSL surpasses the performance of the best offline GCRL baseline. Because DWSL inherits the properties of supervised learning while still exhibiting policy improvement, we believe it is a prime candidate for scaling to larger, more realistic datasets comprised of high-dimensional image observations.

2. Related Work

Learning to reach goals is a long studied problem in both imitation learning (IL) and reinforcement learning (RL). IL approaches to goal-reaching often clone the actions of an expert conditioned on a future goal. While IL has shown promising results on expert demonstrations (Argall et al., 2009; Brohan et al., 2022; Duan et al., 2017) and play (Lynch et al., 2019; Belkhale & Sadigh, 2022; Pertsch et al., 2021), its performance directly depends on dataset optimality. To address these limitations, past works have used additional online data (Gupta et al., 2020; Ding et al., 2019), which is difficult to collect, or attempt to directly correct for the ability of experts (Beliaev et al., 2022; Zhang et al., 2021; Cao & Sadigh, 2021).

Other approaches leverage goal-conditioned RL (GCRL) (Kaelbling, 1993) to improve upon offline sub-optimal data. GCRL algorithms learn policies with sparse goal-reaching reward functions, which can be difficult to optimize, particularly when there is little data of the agent actually reaching its goal. To make datasets appear more optimal, most GCRL algorithms leverage hindsight relabeling (Andrychowicz et al., 2017), or other strategies (Eysenbach et al., 2020a; Li et al., 2020), where achieved outcomes are designated as desired goals. Nevertheless, traditional GCRL methods (Eysenbach et al., 2020b; Kaelbling, 1993; Schaul et al., 2015; Eysenbach et al., 2019) are based on dynamic programming, which can be difficult in high-dimensional spaces with function approximation (Van Hasselt et al., 2018).

In the offline setting we consider, these problems are amplified by out-of-distribution (OOD) errors when sampling actions that cannot be corrected with new data. Offline GCRL algorithms have attempted to address OOD errors by being pessimistic about actions outside of the dataset (Chebotar et al., 2021; Rosete-Beas et al.; Kumar et al., 2020), using weighted imitation learning objectives (Ma et al., 2022; Yang et al., 2022), using model-based planning (Tian et al., 2020), or by estimating implicit quantities (Fang et al.) to avoid the need for sampling (Kostrikov et al., 2022; Garg et al., 2023). Despite these efforts, offline RL methods do not always outperform IL (Mandlekar et al., 2021).

In between IL and RL, some methods optimize policy improvement objectives using only supervised learning. Ghosh et al. (2021) do so for goal reaching using only hindsight optimality. In settings with access to reward, Peters & Schaal (2007); Wang et al. (2018) optimize for a single step of trust region policy improvement (Schulman et al., 2015) using regression. Peng et al. (2019) also solves for the same objective with regression, but their practical algorithm still uses bootstrapping. Hartikainen et al. (2020) consider GCRL with distances, but do so online and use regression. Unlike all these methods, DWSL is designed for offline learning, fits a discrete distribution, and solves for an optimal constrained policy, not a single-step improvement. Other “upside-down” approaches to RL condition on reward to replicate the best performance found in offline datasets (Emmons et al., 2022; Srivastava et al., 2019). However, all of these methods require reward labels and do not guarantee policy improvement.

Recent works have also scaled existing approaches to larger transformer models. Cui et al. (2022) does so for goal-conditioned IL, Decision Transformer (Chen et al., 2021) does so for upside-down RL, and Trajectory Transformer (Janner et al., 2021) does so for model-based planning. Even though the latter two of are not goal-conditioned, we view all of their general insights as complementary to DWSL. While we focus on evaluating DWSL’s algorithmic approach by holding architectures constant, like the aforementioned transformer-based approaches, DWSL uses only supervised objectives, and thus could be combined with similar sequence models in practice.

3. Distance Weighted Supervised Learning

We divide the description of our method across three sections. In Sec. 3.1, we describe the "learning from offline interaction data" setting. Next, in Sec. 3.2 we provide a high-level overview of DWSL in terms of learning distance estimates. In Sec. 3.3, we then formalize and translate this intuition into the offline goal-conditioned reinforcement learning (GCRL) paradigm. A full algorithm block for our method can be found in Appendix B.

3.1. Learning from Offline Interaction Data

We seek to develop methods that can leverage the broadest set of training data. Consequently, we assume the agent acts in a deterministic Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, f(s,a), \mathcal{G}, r(s,a,g), \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, deterministic dynamics $s_{t+1} = f(s_t, a_t)$ where s_{t+1} is the resulting state from taking action a_t at state s_t, goal space $\mathcal{G}$, sparse goal-conditioned reward function $r(s,a,g)$, and discount factor γ. The goal space $\mathcal{G}$ is a subspace of the state space $\mathcal{S}$ admitted by a goal-extraction function $g = \phi(s)$, which is often the identity $\phi(s) = s$. Our objective is to learn a goal-conditioned policy $\pi(a|s,g)$ that has mastery over its environment by being able to reach and remain at goals. To capture this, we maximize the expected discounted return of a reward function $r(s,a,g)$ given a goal distribution $p(g)$:

$$\max_{\pi} \mathbb{E}_{g \sim p(g), a \sim \pi(\cdot|s,g)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g) \right]. \quad (1)$$

While this exact setup differs from prior work, it shares strong connections with two common problem settings: the Stochastic Shortest Path (SSP) problem (Bertsekas & Tsitsiklis, 1991) and GCRL. If we choose ϕ to be the identity, $p(g)$ to be uniform, and the reward function to be $r(s_t, a_t, g) = -1\{s_t \neq g\}$ under known, stochastic dynamics, we recover the SSP problem. However, SSP assumes the ground-truth dynamics to be known, which is not the case when learning purely from offline trajectories.

Alternatively, if we choose $r(s_t, a_t, g) = p(\phi(s_{t+1}) = g|s_t, a_t)$ and have access to a known $p(g)$, we recover GCRL. Note that works in GCRL assume that each trajectory is labeled with the policy's intended goal, providing information about the test-time goal distribution $p(g)$. While seemingly innocuous, this assumption limits the data that offline GCRL can learn from. Many offline data sources, like unstructured play (Lynch et al., 2019) or unsupervised exploration (Yarats et al., 2022), do not contain goal labels along with each trajectory. Moreover, goals can be hard to obtain. In image domains for example, collecting a goal label would require constructing a scene by hand where the desired task has been solved.

To learn from the broadest set of offline data, we consider a more general setting where we do not assume access to ground-truth dynamics, reward labels, or the test-time goal distribution a priori. At training time, we are only given a dataset of state-action trajectories of an arbitrary level of optimality. We denote trajectories of horizon T as $\tau = (s_0, a_0, s_1, a_1, \dots, a_{T-1}, s_T)$, and the entire dataset of trajectories as $\mathcal{D} = \{(\tau)\}$. We take $p(g)$ to be the distribution of goals induced by applying the goal extraction function ϕ over all states in the dataset. While we could try to use a uniform distribution like in SSP, we posit that for most practical datasets goals around the data distribution are likely closer to those for tasks of interest. Our method can use any sparse indicator reward function that can be computed purely from state-action sequences, but in practice we find empirically estimating $r(s_t, a_t, g)$ as $-1\{\phi(s_{t+1}) \neq g\}$ to work well.

3.2. An Intuitive Explanation of DWSL

In this section, we describe the DWSL algorithm using a distance-based interpretation. Intuitively, the best goal-reaching policy is one that reaches goal g from current state s in the fewest number of time steps, or by the shortest path. However, trajectories within the offline dataset $\mathcal{D}$ do not necessarily follow shortest paths. As a result, imitation learning techniques like GCSL (Ghosh et al., 2021), which clone the action taken at each state, may exhibit sub-optimal behavior when trained on $\mathcal{D}$. To improve upon imitation policies, prior works in offline GCRL have estimated shortest paths using approximate dynamic programming with a sparse reward function. Such approaches predict a state-action value function $Q^\pi(s,a,g) = \mathbb{E}_\pi[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_t, a_t, g) | s_t = s, a_t = a]$, or the expected discounted return when following policy π after taking action a at state s. A parameterized Q-function, Q^π_θ, is then iteratively updated to follow the argmax policy $\pi^*(a|s,g) = \arg\max_a Q_\theta(s,a,g)$ via the following temporal difference (TD) update:

$$\min_\theta \mathbb{E}[(Q_\theta(s_t, a_t, g) - y)^2],$$
$$y = r(s_t, a_t, g) + \gamma \max_{a_{t+1}} Q_\theta(s_{t+1}, a_{t+1}, g).$$

This objective, however, is difficult to optimize in practice. First, TD updates are bootstrapped from next state Q-function predictions. Without high coverage of actions a_t and resulting next states s_{t+1}, we can end up with inaccurate or even divergent estimates using Q_θ. This is particularly problematic for high-dimensional data like images. Second, the max operation over a_{t+1} can result in out-of-distribution actions which cause the Q-function to exhibit

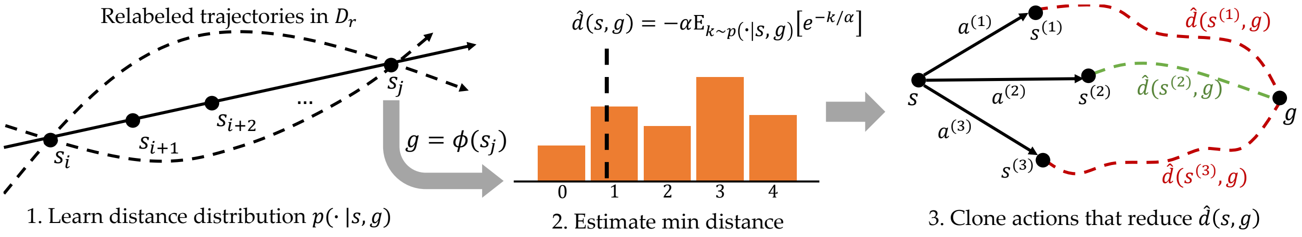

Figure 1. A depiction of the three phases of DWSL. In the first phase, we learn a distribution of distances over the number of time-steps between states. We then compute the LogSumExp of this distribution in order to estimate the minimum distances to the goal . Finally, we extract a policy by imitating actions that follow reduce the learned distance metric.

significant errors. Prior works have used policy constraints (Wu et al., 2019; Fujimoto & Gu, 2021) and value conservatism (Kumar et al., 2020) to mitigate this phenomena. More recently, implicit estimation techniques (Kostrikov et al., 2022; Garg et al., 2023) have been used to eliminate model evaluation on out-of-distribution actions, but still rely on bootstrapping.

DWSL eliminates both of these issues. To do so, we 1) only estimate distances with supervised learning and 2) only evaluate our learned models under the support of the data distribution during training. We learn the entire distribution of pairwise distances between states in $\mathcal{D}$, use this distribution to estimate the minimum goal distance contained within the dataset at each state, and then learn a policy to follow these paths. Our entire approach is outlined in Figure 1.

Learning Distances. For any two states s_i, s_j in a trajectory τ, where $i < j$, we know that goal $\phi(s_j)$ can be reached from s_i in $j - i$ time steps. Using this *hindsight relabeling* technique (Andrychowicz et al., 2017), we generate a relabeled dataset $\mathcal{D}_r = \{(s_i, a_i, s_{i+1}, \phi(s_j))\}$ that contains all the pairwise distances between states and goals. For a given state and goal pair sampled from $\mathcal{D}_r$, we model the discrete distribution over the number of time-steps k from s to g, or $p^r(k|s,g)$ as shown on the left-most side of Figure 1. We obtain a parameterized estimate p^r_θ of this distribution via maximum likelihood under the relabeled dataset:

$$\max_\theta \mathbb{E}_{\mathcal{D}_r}[\log p^r_\theta(j - i - 1|s_i, \phi(s_j))]. \tag{2}$$

In practice, p^r_θ is modeled as a discrete classifier over possible distances. The shortest path between s and g contained within $\mathcal{D}_r$ is then the smallest value of k such that $p^r(k|s,g) > 0$. However, because p^r_θ is learned using function approximation, estimating the minimum distance in this manner with p^r_θ will likely exploit modeling errors. Instead, we compute the LogSumExp over the distribution to obtain a soft estimate of the minimum distance:

$$\hat{d}(s,g) = -\alpha \log \mathbb{E}_{k\sim p^r_\theta(\cdot|s,g)}\left[e^{-k/\alpha}\right]. \tag{3}$$

Note that we multiply the distances by -1 to obtain the minimum estimate, instead of the maximum. α is a temperature hyperparameter such that as $\alpha \to 0$, $\hat{d}(s,g)$ approaches the minimum distance k that has $p^r_\theta(k|s,g) > 0$.

Leveraging Distances for Policy Learning. After learning minimum distance estimates, we want to follow the path they induce at each state. For example, assume we are in state s and want to reach goal g. We can take one of two actions, $a^{(1)}$ and $a^{(2)}$, which result in states $s^{(1)}$ and $s^{(2)}$ respectively. We would prefer to take action $a^{(1)}$ if $\hat{d}(s^{(1)}, g) < \hat{d}(s^{(2)}, g)$, e.g., it has a smaller estimated distance to the goal. Thus, we want to weight the likelihood of different actions by their resulting distance estimates (right of Figure 1). However, naïvely weighting actions this way would apply a larger weighting to all datapoints closer to g, as any state far away from g will naturally have a larger distance. We instead weight the likelihood of actions according to their *reduction* in estimated distance to the goal, which we refer to as the advantage. This gives the following imitation learning objective over the relabeled dataset for learning a parameterized policy π_ψ:

$$\max_\psi \mathbb{E}_{\mathcal{D}_r}\left[e^{\text{adv}/\beta} \log \pi_\psi(a_i|s_i, \phi(s_j))\right], \tag{4}$$
$$\text{adv} = \hat{d}(s_i, \phi(s_j)) - 1 - \hat{d}(s_{i+1}, \phi(s_j)).$$

If taking action a_i to state s_{i+1} follows the shortest path to $\phi(s_j)$, then $\hat{d}(s_i, \phi(s_j)) = 1 + \hat{d}(s_{i+1}, \phi(s_j))$, and a_i will be given a weight of 1 in Equation 4. If a_i results in a sub-optimal s_{i+1} that leads to a longer path, $\hat{d}(s_i, \phi(s_j)) < 1 + \hat{d}(s_{i+1}, \phi(s_j))$, and the assigned weight will be less than 1. We use exponentiated advantages to ensure all weights are positive, as done in prior work (Peng et al., 2019; Nair et al., 2020; Yang et al., 2022; Ma et al., 2022; Kostrikov et al., 2022; Garg et al., 2023), but we learn these advantage weights using only supervised learning. β is a temperature hyperparameter that controls weighting.

To summarize, our algorithm first learns p^r_θ via Equation 2, estimates distances according to Equation 3, and then learns a policy via Equation 4. In practice we truncated the distance distribution p^r_θ to predict only over a finite number of bins B. As predicting the distribution over a long horizon T may be challenging, we take a similar approach to N-step returns and use $B = T//N$ discrete bins. When the task horizon T is sufficiently small, it suffices to set $B = T$. We normalize the distance values by the number of bins B so that they always lie in $[0, 1]$.

Following this algorithm theoretically approximates the optimal solution to a constrained version of the objective in Eq. 1. In the next section, we show this connection by demonstrating that distances are equivalent to expected returns.

3.3. Policy Improvement with DWSL

In this section, we show that the DWSL algorithm theoretically solves for an optimal constrained policy under mild assumptions. Here, we develop a lower bound for the infinite horizon MDPs traditional used in GCRL. In the Appendix, we show that the DWSL objective recovers the exact optimal policy for fininte horizon MDPs which we use in practice. We begin by outlining how distances can be translated to the total sum of rewards, or return an agent receives. Using this insight, we show how distances can be substituted into common policy learning objectives.

Equivalence of Distances and Returns. For sparse reward functions, the number of time-steps it takes a policy to reach a goal can be directly mapped to its return. Specifically, we consider the relabeling policy $\pi_r(a|s, g)$ that produces the relabeled experience present in $\mathcal{D}_r$ and make two assumptions. First, we assume the existence of a stationary action $a^{(s)}$ for each state $s \in \mathcal{S}$ such that $f(s, a^{(s)}) = s$ meaning that once a goal is reached, it is possible to stay there indefinitely. Second, we assume that the relabeling policy $\pi_r(a|s, g)$ always takes the stationary action to stay at an achieved the goal, or $\pi_r(s, \phi(s)) = a^{(s)}$. Because sparse reward functions only depend on whether or not $\phi(s) = g$, a policy's return depends only on how many time-steps it was at the goal. As the relabeling policy π_r always remains at the goal after reaching it, its return can be computed from just its time-step distance to the goal. For example, consider the reward function $r(s_t, a_t, g) = -1\{\phi(s_{t+1}) \neq g\}$ used in prior work in GCRL, which is 0 if the goal is reached after taking action a_t from s_t, and -1 otherwise. If π_r takes k time steps to reach goal g from state s_t, it receives a reward of -1 for $k-1$ steps, and then a reward of 0 forever. Per the geometric series formula, this equates to a total discounted return of $-(1-\gamma^{k-1})/(1-\gamma)$. Thus, we establish a one-to-one mapping between the distances and returns of π_r. This implies that modeling the distribution of time-steps it takes π_r to reach goals is equivalent to modeling the distribution of returns. As shown in Section 3.2, the distribution of time-steps $p^r(k|s, g)$ is easily learned in a supervised manner. Next, we show that this distribution over time-steps can be substituted into constrained RL objectives.

Bounding KL-Constrained Values. In this subsection, we present the KL-constrained RL objective and its optimal solution. Then, we show that we can use the distribution of returns of π_r to bound the corresponding optimal state-value function.

KL-constrained RL adds a KL-divergence penalty $D_{KL}(\pi||\pi_r)$ between the learned policy π and dataset behavior policy π_r to the objective from Equation 1:

$$\max_{\pi} \mathbb{E}_{\pi,p(g)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g) - \alpha \log \frac{\pi(a_t|s_t, g)}{\pi_r(a_t|s_t, g)} \right]. \tag{5}$$

In the above, α weights the KL-penalty. Similar objectives have proven to be useful for preventing excessive extrapolation in offline RL (Kumar et al., 2020).

As shown by Garg et al. (2023); Pertsch et al. (2021) using principles first derived in maximum entropy RL (Haarnoja et al., 2017; Ziebart, 2010), the optimal state value V^* and state-action value Q^* to Equation 5 satisfy a unique fixed point. In the goal-conditioned setting, this gives

$$Q^*(s_t, a_t, g) = r(s_t, a_t, g) + \gamma V^*(s_{t+1}, g), \tag{6}$$

$$V^*(s_t, g) = \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t,g)} \left[e^{Q^*(s_t,a_t,g)/\alpha} \right]. \tag{7}$$

Notice how the value function V^* is written in terms of the expectation under π_r. By repeatedly substituting Equation 6 into 7 we remove the dependence on Q^*, which we do not know, and obtain an expectation as a function of the discounted returns of π_r. This is summarized by the following Proposition, for which the full proof is in Appendix A.1.

Proposition 3.1. *Assuming deterministic dynamics, we can bound the optimal value function for the objective in Equation 5 using empirical returns of π_r as*

$$V^*(s_t, g) \geq \alpha \log \mathbb{E}_{\pi_r}[e^{(\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}, g))/\alpha}]. \tag{8}$$

The lower bound is a result of applying Jensen's inequality to the exponentiated discount factor, $\mathbb{E}[X]^\gamma \geq E[X^\gamma]$. As $\gamma \to 1$, as is often the case for RL in the real-world, this bound becomes tighter. In Appendix A.2 we show that this relationship holds with equality for finite horizon MDPs.

The right hand side is simply the LogSumExp over the discounted returns of π_r. As the LogSumExp converges to the maximum, we interpret this result as stating that the optimal value function is lower bounded by the best behavior exhibited by π_r. As shown before, we can equate the distribution over discounted returns with the distribution over time-step distances. Thus, via a simple change of variables, we arrive at the following corollary.

Corollary 3.2. *Assume there exists an onto mapping from discounted returns to time-step distances k such that $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g) = \mathcal{R}_k$ for some $k \in \mathbb{N}$, then*

$$V^*(s_t, g) \geq \alpha \log \mathbb{E}_{k \sim p^r(\cdot|s_t,g)} \left[e^{\mathcal{R}_k/\alpha} \right]. \tag{9}$$

The full proof is in Appendix A.3. Notice that the form of this bound is identical to the distance estimator used in Equation (2) when $r(s_t, a_t, g) = -1\{\phi(s_{t+1}) = g\}$, implying that the distance learning component of DWSL does indeed extract optimal value function estimates from just the distance distribution $p^r(\cdot|s, g)$. While most works depend on bootstrapping to estimate the optimal value function, we have shown that we can obtain similar estimates using only supervised learning by taking advantage of the structure of reward functions in offline GCRL.

Policy Extraction. The final step of the DWSL algorithm is to extract the optimal policy from distance, or value, estimates. Given the optimal Q^* and V^*, Peng et al. (2019) showed that the optimal policy π^* for the KL-constrained RL objective can be written in proportion to the advantage function $A^*(s, a, g) = Q^*(s, a, g) - V^*(s, g)$ and the behavior distribution π_r. Given that we assume deterministic dynamics, we can remove the dependence of A^* on Q^* via Equation 6, and arrive at the following statement:

$$\pi^*(a_t|s_t, g) \propto \pi_r(a_t|s_t, g) \exp\left(A^*(s_t, a_t, g)/\alpha\right), \quad (10)$$
$$A^*(s_t, a_t, g) = r(s_t, a_t, g) + \gamma V^*(s_{t+1}, g) - V^*(s_t, g).$$

As is common practice (Nair et al., 2020), we project this result to the parameterized space of a learned policy π_ψ by solving $\psi^* = \arg\min_\psi E_{\mathcal{D}_r}[D_{KL}(\pi^*||\pi_\psi)]$. Doing so exactly recovers the policy learning objective in Equation 4 in Section 3.2 when using the reward function $r(s_t, a_t, g) = -1\{\phi(s_{t+1}) = g\}$. Thus, in the finite horizon case where Lemma 3.1 is tight (see Appendix A.2), DWSL recovers the optimal policy corresponding to the objective in Equation 5.

We note that when using the expectation instead of the LogSumExp, DWSL recovers the advantage-weighted regression (AWR) objective (Peng et al., 2019) and corresponds to one-step of KL-constrained policy improvement. We provide derivations of this connection in Appendix A.4 and ablate our choice of the LogSumExp statistic in Sec 4.2.

4. Experiments

In this section we seek to answer the following questions: 1) How does DWSL perform across a broad selection of robotic datasets? 2) Does DWSL use the right objective? 3) When does DWSL fail? 4) How robust is DWSL?

4.1. How does DWSL Perform?

We extensively evaluate the performance of DWSL on offline interaction datasets across a variety of simulated robotics environments. In this section we present results on the following domains, but include more details, additional results, and learning curves in Appendix C.

Visual Gym Robotics. The Gym robotics environments from Plappert et al. (2018), including Fetch and Hand, are designed to test GCRL algorithms. We construct image-based offline datasets by applying a large amount of action noise to ground-truth policies trained with TD3-HER as in Yang et al. (2022). We use these datasets to evaluate learning from high-dimensional, sub-optimal data.

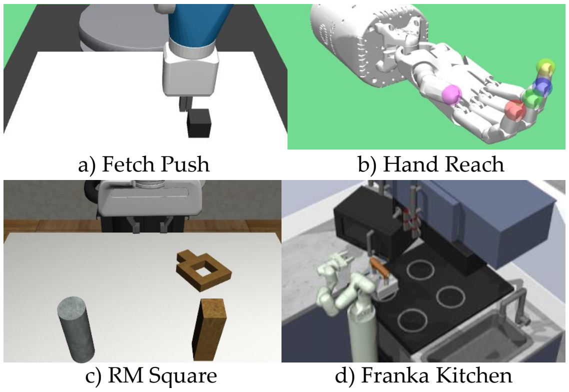

Figure 2. Environment depictions. For Hand and Fetch we show (cropped) image observations provided to agents.

Franka Kitchen. The Franka Kitchen dataset from Lynch et al. (2019) is comprised of 566 human demonstrations and tests how well different methods can learn from play data. At test time, we sample goals from the end of a held-out set of validation trajectories and measure how many of four sequential tasks are completed.

Robomimic. We take the *Square* and *Can* datasets from Mandlekar et al. (2021) with both proficient and multi-human demonstrations and make them goal-conditioned. At test time, we condition on success states from the validation set. RL algorithms in the past have exhibited poor performance on this benchmark. Consequently, these datasets test DWSL's ability to maintain IL as a lower bound.

We compare the performance of DWSL to baselines suitable for learning goal-conditioned policies from offline interaction data without reward or goal labels. GCSL Ghosh et al. (2021) is an imitation learning method that leverages hindsight relabeling. We also compare to state-of-the-art offline GCRL algorithms that perform TD-learning. WGCSL (Yang et al., 2022) uses Q-Learning to estimate shortest paths. GoFAR (Ma et al., 2022) performs value iteration over a dual form of GCRL. GCIQL is a goal conditioned version of IQL (Kostrikov et al., 2022) which uses expectile regression to avoid sampling OOD actions. Finally, we compare to a variant of our algorithm that uses bootstrapping via distributional RL to fit the distance distribution, which we denote DWSL-B. For our method, we use the same hyperparameters (α, β) for every single dataset. We vary the number of bins B with the task horizon. Our full results can be found in Table 1.

Overall, we find that DWSL surpasses or matches the performance of our supervised baseline, GCSL, on *all* of these datasets. This is not the case for any offline GCRL

Environment	Dataset	GCSL	WGCSL	GoFAR	GCIQL	DWSL-B	DWSL
Franka Kitchen	566 Play Demos	**2.98 ± .10**	2.62 ± .19	2.43 ± .06	2.81 ± .19	**2.97 ± .16**	**2.92 ± .04**
RM Can	100 PH Demos	67 ± 17%	50 ± 19%	44 ± 8%	58 ± 16%	70 ± 9%	**77 ± 5%**
	300 MH Demos	31 ± 13%	27 ± 6%	35 ± 2%	28 ± 20%	**38 ± 10%**	36 ± 8%
RM Square	100 PH Demos	**48 ± 9%**	4 ± 4%	18 ± 5%	44 ± 4%	38 ± 3%	**47 ± 12%**
	300 MH Demos	10 ± 5%	7 ± 3%	**17 ± 7%**	13 ± 4%	15 ± 9%	14 ± 5%
Fetch Push	250K Noise 1	18.53 ± .5	22.94 ± .4	14.15 ± 1.8	20.62 ± .5	**24.78 ± .4**	22.88 ± .6
	250K Noise 2	10.29 ± .2	12.89 ± .4	7.34 ± .4	11.76 ± .4	15.23 ± .7	**16.57 ± .2**
Fetch Pick	250K Noise 1	26.55 ± 1.1	27.97 ± .7	26.23 ± .5	25.33 ± 1.2	**29.86 ± .1**	**30.46 ± .2**
	250K Noise 2	11.65 ± .6	12.32 ± 1.2	10.32 ± .3	11.08 ± .5	14.26 ± .3	**17.45 ± .4**
Fetch Slide	250K Noise 0.5	**3.61 ± .2**	3.08 ± .1	2.19 ± .2	3.46 ± .2	**3.53 ± .1**	**3.67 ± .1**
	250K Noise 1	2.23 ± .1	2.02 ± .1	1.55 ± .1	2.40 ± .2	2.44 ± .2	**2.60 ± .1**
Hand Reach	1M 90% R 10% E	1.74 ± 1.7	5.41 ± 3.5	5.99 ± 1.4	3.97 ± 3.8	5.91 ± 1.9	**9.53 ± 3.8**
	500K Noise 0.2	4.07 ± 2.0	9.46 ± 3.7	**16.70 ± 2.3**	5.40 ± 2.4	10.49 ± 2.2	10.79 ± 1.2

Table 1. This table includes results across the primary environments we test. Bolded numbers are within 95% of the best score. We run four seeds in state-based domains and three seeds in image domains. The dataset column provides details on the type of demonstrations. In Robomimic (RM), "PH" indicates that the demos came from a single proficient user and "MH" indicates that the demos came from multiple users of varying abilities. For Gym Image datasets, the first value gives the number of transitions in the dataset. For datasets with "Noise", we provide the standard deviation of the Gaussian noise applied to the oracle policy when collecting the dataset. "90% R 10% E" refers to 90% random, 10% expert data.

methods. Moreover, DWSL outperforms prior TD-learning based offline GCRL methods on Franka Kitchen and 3 of 4 Robomimic datasets, supporting past evidence (Mandlekar et al., 2021) that TD-learning often struggles with play or human collected data. Notably, TD-learning approaches also perform significantly worse in image domains, despite the fact that the data comes from an oracle RL policy. DWSL outperforms or matches all offline GCRL algorithms on 6 of 8 image-based Gym Robotics datasets, while converging faster. We show a sample learning curve in Figure 3. The gap between DWSL and bootstrapping approaches is largest in the more sub-optimal, higher noise image datasets. This is potentially because dynamic programming in high-dimensions is harder with sparser coverage. Offline GCRL algorithms beat DWSL only in the low-dimensional state-based Gym Robotics domains, but DWSL still exhibits improvement over pure imitation. These results are provided in Appendix 4. Finally, DWSL outperforms or matches DWSL-B on 7 of 8 image-based Gym Robotics datasets, while performing similarly in other domains, suggesting that with our distributional learning formulation and LogSumExp distance estimation, bootstrapping is not critical for performance.

4.2. Does DWSL use the right objective?

Unlike other approaches, DWSL fits the KL-constrained optimal policy by learning a discrete distribution and using the LogSumExp for distance estimation. We analyze if this is the correct choice by comparing DWSL to other supervised variants that use different objectives. In Figure 3, we compare DWSL's use of the LogSumExp and classification objective to other options in both state and image domains. We first compare to replacing the LogSumExp with the expectation. We also compare to using continuous regression rather than classification, for modeling the expectation as proposed by AWR (Peng et al., 2019), and modeling expectiles as proposed by (Kostrikov et al., 2022). To exaggerate the ability of these approaches to improve behavior, we use datasets comprised of 90% random and 10% expert trajectories and for the state dataset, decreased α. Overall, we find that our method performs the best, suggesting that our decision to model entire discrete distributions and use the LogSumExp for distance estimation are important for best performance when only using supervised learning for advantage estimation.

4.3. When does DWSL fail?

We run experiments on a goal-conditioned version of the AntMaze benchmark from Fu et al. (2020) to find failure modes of DWSL. This toy benchmark requires value propagation to stitch together different parts of sub-optimal experience present in the data. We expect DWSL to perform worse on stitching because it does not use approximate dynamic programming. Instead, DWSL can be thought of as performing greedy search for the best action supported by the dataset at each state. Here, DWSL consistently improves upon GCSL and other supervised methods (results in Appendix C) but performs worse than GoFAR and GCIQL. This indicates that a lack of bootstrapping is a fundamental drawback to all supervised algorithms. When dealing with more realistic high-dimensional robotics data, however, we believe the benefits of supervised learning can outweigh this downside.

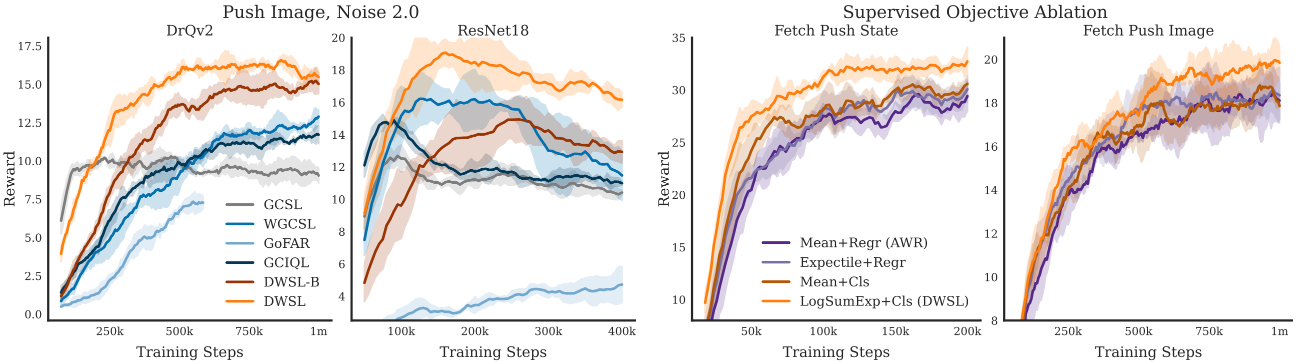

Figure 3. Here we present ablation results on Fetch Push. From left to right, the first two plots show learning curves for image datasets with different encoders. The first encoder is adapted from DrQv2 (Yarats et al., 2021) (used in main experiments), and the second encoder is adapted from Mandlekar et al. (2021) and uses a ResNet-18. The right two plots show the performance of different supervised objectives on state and image datasets with 90% random and 10% expert data. "Cls" stands for classification and "Regr" for regression. Our LogSumExp statistic is better able to separate expert from random data than objectives proposed by prior work, like Peng et al. (2019).

Dataset	**GCSL**	**WGCSL**	**GoFar**	**GCIQL**	**DWSL**
Umaze	64 ± 2	83 ± 3	$\mathbf{91 \pm 1}$	85 ± 1	74 ± 4
Umaze Diverse	59 ± 1	47 ± 6	$\mathbf{86 \pm 3}$	$\mathbf{86 \pm 2}$	67 ± 3
Med Play	56 ± 6	35 ± 22	70 ± 1	$\mathbf{74 \pm 5}$	69 ± 8
Med Diverse	60 ± 3	27 ± 13	63 ± 4	$\mathbf{72 \pm 7}$	68 ± 6
Large Play	17 ± 5	0 ± 0	$\mathbf{40 \pm 7}$	31 ± 8	15 ± 5
Large Diverse	12 ± 3	3 ± 3	$\mathbf{45 \pm 8}$	22 ± 4	18 ± 2

Table 2. Success rates on the AntMaze-v2 benchmark from Fu et al. (2020). Learning curves in Appendix C.

4.4. How robust is DWSL?

In this section, we investigate how the performance of DWSL is affected by changes in architecture, augmentation, hyperparameters, and access to ground-truth goals.

To test how well DWSL scales, we run experiments on the Gym Robotics image datasets with ResNet-18 architectures from Mandlekar et al. (2021). An example learning curve on the "Fetch Push Noise 2.0" dataset can be found in Figure 3, and all learning curves can be found in Appendix C.4. Overall performance increases for most methods with this additional network capacity, though overfitting starts earlier. Interestingly, the gap between DWSL and DWSL-B increases with the ResNet architecture, which we believe indicates that DWSL's supervised objectives are more capable of scaling with model size. We train encoders with gradients from the policy instead of the value function because it performed better for all methods (Appendix C.4), suggesting that supervised learning leads to stronger representation learning in high-dimensions.

DWSL's supervised value function function is also more robust to image-based perturbations than dynamic programming based GCRL algorithms. In Appendix C.4, we measure the pearson correlation of estimated values on expert demonstrations in Fetch Push with and without random shift augmentations. DWSL's pearson correlation coefficient drops by only 0.006 under augmentation, while other methods can drop by more than ten times that amount.

We test the sensitivity of DWSL to different values of α and β in state domains in Appendix C.4. Varying α from 0.1 to 10 only changes return by 0.2 and 2.86 on Fetch Push state with noisy expert and 90% random data respectively. Varying β from 0.01 to 0.25 changes return by 5.87 and 1.52 on the same benchmarks. This is unlike many offline RL algorithms, where temperature parameters have been shown to drastically affect performance (Garg et al., 2023).

Finally, in Appendix C.4 we consider how performance might change if we give algorithms access to the test-time goal distribution $p(g)$ by testing DWSL and WGCSL with different goal relabeling ratios, which controls how often goals are relabeled with achieved states during training. When the relabeling ratio is 1, we recover the *learning from offline interaction* setting we consider. We find that the performance of WGCSL is affected dramatically when the relabeling ratio is 0.5 – it substantially degrades in Fetch, but improves in AntMaze – while DWSL is largely unaffected.

5. Conclusion

We propose Distance Weighted Supervised Learning, a general method for learning goal-conditioned policies from offline interaction data. Theoretically, we show that by exchanging continuous value estimates with statistics of the empirical discrete distance distribution, we can fit an optimal constrained policy using only supervised learning. Empirically, we demonstrate that DWSL outperforms prior work on image domains and unlike previous offline GCRL methods always performs the same or better than IL baselines. This makes DWSL suitable for scaling to large, real-world applications with heterogeneous and high-dimensional data. Future work can apply DWSL to larger architectures and datasets and explore avenues for pretraining the learned distance distribution on action-free datasets such as internet video.

Acknowledgements

This work was supported by ONR, DARPA YFA, Ford, and NSF Awards #1941722 and #2218760. JH is graciously supoported by the National Defense Science Engineering Graduate (NDSEG) Fellowship Program. We would also like to thank members of the ILIAD lab and our anonymous reviewers for fruitful discussions.

References

Akgun, B., Cakmak, M., Jiang, K., and Thomaz, A. L. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355, 2012.

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

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Beliaev, M., Shih, A., Ermon, S., Sadigh, D., and Pedarsani, R. Imitation learning by estimating expertise of demonstrators. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1732–1748. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/beliaev22a.html.

Belkhale, S. and Sadigh, D. PLATO: Predicting latent affordances through object-centric play. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=UAA5bNospA0.

Bertsekas, D. P. and Tsitsiklis, J. N. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

Cao, Z. and Sadigh, D. Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters*, 6(3):5231–5238, 2021.

Chebotar, Y., Hausman, K., Lu, Y., Xiao, T., Kalashnikov, D., Varley, J., Irpan, A., Eysenbach, B., Julian, R. C., Finn, C., et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In *International Conference on Machine Learning*, pp. 1518–1528. PMLR, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Cui, Z. J., Wang, Y., Muhammad, N., Pinto, L., et al. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

Ding, Y., Florensa, C., Abbeel, P., and Phielipp, M. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

Duan, Y., Andrychowicz, M., Stadie, B., Jonathan Ho, O., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline RL via supervised learning? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=S874XAIpkR-.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Eysenbach, B., Geng, X., Levine, S., and Salakhutdinov, R. R. Rewriting history with inverse rl: Hindsight inference for policy improvement. *Advances in neural information processing systems*, 33:14783–14795, 2020a.

Eysenbach, B., Salakhutdinov, R., and Levine, S. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2020b.

Fang, K., Yin, P., Nair, A., Walke, H. R., Yan, G., and Levine, S. Generalization with lossy affordances: Leveraging broad offline data for learning visuomotor tasks. In *6th Annual Conference on Robot Learning*.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519. IEEE, 2016.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Fujimoto, S. and Gu, S. A minimalist approach to offline reinforcement learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Q32U7dzWXpc.

Garg, D., Hejna, J., Geist, M., and Ermon, S. Extreme q-learning: Maxent reinforcement learning without entropy. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2301.02328.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C. M., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=rALA0Xo6yNJ.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*, pp. 1025–1037. PMLR, 2020.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Hartikainen, K., Geng, X., Haarnoja, T., and Levine, S. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1lmhaVtvr.

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., and Finn, C. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pp. 991–1002. PMLR, 2022.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Kaelbling, L. P. Learning to achieve goals. In *IJCAI*, volume 2, pp. 1094–8. Citeseer, 1993.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Li, A., Pinto, L., and Abbeel, P. Generalized hindsight for reinforcement learning. *Advances in neural information processing systems*, 33:7754–7767, 2020.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1903.01973.

Ma, Y. J., Yan, J., Jayaraman, D., and Bastani, O. Offline goal-conditioned reinforcement learning via \$f\$-advantage regression. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_h29VprPHD.

Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Martín-Martín, R. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

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

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *CoRR*, abs/1910.00177, 2019. URL https://arxiv.org/abs/1910.00177.

Pertsch, K., Lee, Y., and Lim, J. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

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

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018. URL http://arxiv.org/abs/1802.09464.

Rosete-Beas, E., Mees, O., Kalweit, G., Boedecker, J., and Burgard, W. Latent plans for task-agnostic offline reinforcement learning. In *6th Annual Conference on Robot Learning*.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

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

Tian, S., Nair, S., Ebert, F., Dasari, S., Eysenbach, B., Finn, C., and Levine, S. Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*, 2020.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Wang, Q., Xiong, J., Han, L., Liu, H., Zhang, T., et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Yang, R., Lu, Y., Li, W., Sun, H., Fang, M., Du, Y., Li, X., Han, L., and Zhang, C. Rethinking goal-conditioned supervised learning and its connection to offline RL. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KJztlfGPdwW.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., and Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10674–10681, 2021.

Yarats, D., Brandfonbrener, D., Liu, H., Laskin, M., Abbeel, P., Lazaric, A., and Pinto, L. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Zhang, S., Cao, Z., Sadigh, D., and Sui, Y. Confidence-aware imitation learning from demonstrations with varying optimality. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 12340–12350. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/670e8a43b246801ca1eaca97b3e19189-Paper.pdf.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

Appendix

We divide the Appendix into four different sections as follows.

A. In Appendix A we provide full derivations of mathematical results from 3 and additional theoretical results.

B. In Appendix B we provide a detailed algorithm block for DWSL.

C. Appendix B serves as an extended results section, where we provide full learning curves for experiments and ablations mentioned in Section 4.

D. Finally, in Appendix D we provide hyperparameter and implementation details.

A. Theory

A.1. Proof of Proposition 3.1

Assume deterministic dynamics and discount factor $\gamma < 1$. Garg et al. (2023) gives us that for objective 5, the following equations hold for optimal value functions:

$$\begin{aligned} Q^*(s_t, a_t, g) &= r(s_t, a_t, g) + \gamma V^*(s_{t+1}, g) \\ V^*(s_t, g) &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{Q^*(s_t, a_t)/\alpha} \right] \end{aligned}$$

We proceed by taking the optimal value function, substituting in the optimal Q-value, and subsequently the next-timestep value definition. Then, we apply Jensen's Inequality and the Markov property to compute the expectation over both a_t and a_{t+1}. Repeating this process recursively, we attain the sum of discounted returns.

$$\begin{aligned} V^*(s_t, g) &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{Q^*(s_t, a_t)/\alpha} \right] \\ &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{(r(s_t, a_t, g) + \gamma V^*(s_{t+1}, g))/\alpha} \right] \text{ substitute } Q^* \\ &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{r(s_t, a_t, g)/\alpha} \left(e^{V^*(s_{t+1}, g)/\alpha} \right)^{\gamma} \right] \text{ isolate } V^* \\ &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{r(s_t, a_t, g)/\alpha} \left(\mathbb{E}_{a_{t+1} \sim \pi_r(\cdot|s_{t+1}, g)} \left[e^{Q^*(s_{t+1}, a_{t+1})/\alpha} \right] \right)^{\gamma} \right] \text{ substitute } V^* \\ &\geq \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{r(s_t, a_t, g)/\alpha} \mathbb{E}_{a_{t+1} \sim \pi_r(\cdot|s_{t+1}, g)} \left[e^{\gamma Q^*(s_{t+1}, a_{t+1})/\alpha} \right] \right] \text{ apply Jensen's to } \gamma \\ &= \alpha \log \mathbb{E}_{a_t, a_{t+1} \sim \pi_r} \left[e^{r(s_t, a_t, g)/\alpha} e^{\gamma Q^*(s_{t+1}, a_{t+1})/\alpha} \right] \text{ apply the Markov Property} \end{aligned}$$

We then notice that we have written the value function for time t in terms of the Q function for time $t + 1$. We can repeatedly apply the same steps, shown once and then infinitely below, to obtain the final result.

$$\begin{aligned} V^*(s_t, g) &\geq \alpha \log \mathbb{E}_{a_t, a_{t+1} \sim \pi_r} \left[e^{r(s_t, a_t, g)/\alpha} e^{\gamma Q^*(s_{t+1}, a_{t+1})/\alpha} \right] \\ &= \alpha \log \mathbb{E}_{a_t, a_{t+1}, a_{t+2} \sim \pi_r} \left[e^{r(s_t, a_t, g)/\alpha} e^{\gamma r(s_{t+1}, a_{t+1})/\alpha} e^{\gamma^2 Q(s_{t+2}, a_{t+2})/\alpha} \right] \text{ repeat once} \\ &= \alpha \log \mathbb{E}_{\tau \sim \pi_r} \left[e^{r(s_t, a_t, g)/\alpha} e^{\gamma r(s_{t+1}, a_{t+1})/\alpha} e^{\gamma^2 r(s_{t+2}, a_{t+2})/\alpha} \ldots \right] \text{ repeat infinitely} \\ &= \alpha \log \mathbb{E}_{\tau \sim \pi_r} \left[e^{\left(\sum_{t'=t}^{\infty} \gamma^{t-t'} r(s_{t'}, a_{t'}, g) \right)/\alpha} \right] \text{ reduce} \end{aligned}$$

Note that the only step that is not true with equality was the application of Jensen's Inequality to the discount factor γ, as $\mathbb{E}[X]^{\gamma} \geq E[X^{\gamma}]$. This bound becomes tight as the function is near linear for $\gamma \to 1$.

A.2. Proof of Proposition 3.1 for Finite Horizons

The bound DWSL computes on the optimal KL-constrained policy is tight in the finite horizon case with $\gamma = 1$ which we use in practice. Here we present an analogous proof of Proposition 3.1 for the finite case. Consider the optimal finite horizon fixed point equations, indexed by time t:

$$\begin{aligned} Q_t^*(s_t, a_t, g) &= r(s_t, a_t, g) + V_{t+1}^*(s_{t+1}, g) \\ V_t^*(s_t, g) &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{Q_t^*(s_t, a_t)/\alpha} \right] \end{aligned}$$

Note that we have removed γ as $\gamma = 1$ for the finite horizon case. We can proceed to follow the same substitution steps. We can then follow the same steps as in Proposition 3.1. Below we do so, skipping a few steps as the log is the same:

$$\begin{aligned} V^*(s_t, g) &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{(r(s_t, a_t, g) + V_{t+1}^*(s_{t+1}, g))/\alpha} \right] \\ &= \alpha \log \mathbb{E}_{a_t \sim \pi_r(\cdot|s_t, g)} \left[e^{r(s_t, a_t, g)/\alpha} \mathbb{E}_{a_{t+1} \sim \pi_r(\cdot|s_{t+1}, g)} \left[e^{Q_{t+1}^*(s_{t+1}, a_{t+1})/\alpha} \right] \right] \text{ substitute } V^* \\ &= \alpha \log \mathbb{E}_{a_t, a_{t+1} \sim \pi_r} \left[e^{r(s_t, a_t, g)/\alpha} e^{Q_{t+1}^*(s_{t+1}, a_{t+1})/\alpha} \right] \\ &= \alpha \log \mathbb{E}_{\tau \sim \pi_r} \left[e^{\left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}, g)\right)/\alpha} \right] \end{aligned}$$

In the final line we repeat the the first three steps up to the max horizon T.

A.3. Proof of Corollary 3.2

Corollary 3.2 substitutes the distribution over trajectories from π_r with the distribution of distances $p^r(\cdot|s_t, g)$ by assuming there exists a mapping between possible returns and a function of the distance k. Specifically, $\sum_{t=0}^{\infty} \gamma^t r(s, a, g) = \mathcal{R}_k$ for some $k \in \mathbb{N}$. This is possible for reward functions $r(s, a, g) = -1\{\phi(s') = g\}$ because we assume π_r to stay at goals after reaching them. Thus, the valid discounted returns of π_r directly correspond to distances. In the below derivation, we use summations to expand the expectation, but they could be exchanged for integrals. We start with Proposition 3.1 and expand the expectation.

$$\begin{aligned} V^*(s_t, g) &\geq \alpha \log \mathbb{E}_{\tau \sim \pi_r} \left[e^{\left(\sum_{t'=t}^{\infty} \gamma^{t-t'} r(s_{t'}, a_{t'}, g)\right)/\alpha} \right] \\ &= \alpha \log \sum_{\tau} \pi_r(\tau|s_t, g) e^{\left(\sum_{t'=t}^{\infty} \gamma^{t-t'} r(s_{t'}, a_{t'}, g)\right)/\alpha} \text{ Expand expectation over trajectories} \\ &= \alpha \log \sum_{\tau} \sum_{k} \pi_r(\tau, \phi(s_{t+k}) = g|s_t, g) e^{\left(\sum_{t'=t}^{\infty} \gamma^{t-t'} r(s_{t'}, a_{t'}, g)\right)/\alpha} \text{ law of total probability} \\ &= \alpha \log \sum_{\tau} \sum_{k} \pi_r(\tau, \phi(s_{t+k}) = g|s_t, g) e^{\mathcal{R}_k/\alpha} \text{ sub in } \mathcal{R}_k \\ &= \alpha \log \sum_{k} e^{-\frac{1-\gamma^k}{(1-\gamma)\alpha}} \sum_{\tau} \pi_r(\tau, \phi(s_{t+k}) = g|s_t, g) \text{ group like terms, swap sum} \\ &= \alpha \log \sum_{k} e^{-\frac{1-\gamma^k}{(1-\gamma)\alpha}} p^{\pi_r}(k|s_t, g) \text{ marginalize to distance distribution} \\ &= \alpha \log \mathbb{E}_{k \sim p^{\pi_r}(\cdot|s_t, g)} \left[e^{-\frac{1-\gamma^k}{(1-\gamma)\alpha}} \right] \text{ re-write expectation.} \end{aligned}$$

This corollary also holds for the finite horizon case, by assuming $k \in 0, 1, ...T$.

A.4. From DWSL to AWR

When using the expectation instead of the LogSumExp, DWSL solves for the same objective as AWR (Peng et al., 2019), which is similar to Wang et al. (2018). We show this below. The key theoretical difference between DWSL and our approach is that AWR exponentially weights the learned policy by the behavior policy's advantage function, instead of the optimal KL-constrained policy's advantage function. Using our notation for the relabeled policy and goal-conditioned setting, the AWR objective is:

$$\max_{\pi} \mathbb{E}_{\pi_r} \left[e^{(\mathcal{R}^{\pi_r}_{s,a,g} - V^{\pi_r}(s,g)} \log \pi(a|s,g) \right]$$

where $\mathcal{R}^{\pi_r}_{s,a}$ is the Monte-Carlo estimate of the empirical returns of π_r taking action a from state s. In practice, AWR uses bootstrapping to compute the advantage estimate as $r(s,a,g) + V^{\pi_r}(s',g) - V^{\pi_r}(s,g)$ instead of pure supervised learning. This corresponds to solving one step of the policy improvement objective from (Schulman et al., 2015):

$$\eta(\pi) = \mathbb{E}_{\pi_r} \left[\mathcal{R}^{\pi_r}_{s,a,g} - V^{\pi_r}(s,g) \right] - \beta \mathbb{E}_{s \sim \pi_r} [D_{KL}(\pi || \pi_r)]$$

Note that with DWSL we can reconstruct estimates of $V^{\pi_r}(s,g)$ by computing the expectation of the discrete distance distribution as follows:

$$V^{\pi_r}(s,g) = \mathbb{E}_{k \sim p^r(\cdot|s,g)} [\mathcal{R}_k]$$

We then recover a version of AWR using classification which performs a single-step of policy improvement.

B. DWSL Algorithm

Below we provide a detailed outline of the DWSL algorithm.

Algorithm 1 Distance Weighted Supervised Learning

```
Input: Dataset $\mathcal{D}$, Goal function $\phi$, N-Step $N$, Horizon $T$, Temperatures $\alpha, \beta$
Initialize: Distribution $p^r_\theta$ with $B = T//N$ bins, Policy $\pi_\psi$
for distribution training steps... do
    Sample relabeled data $(s_i, \phi(s_j)), j > i$
    Update $\theta$ via $\max_\theta \mathbb{E}_{\mathcal{D}_r} [\log p^r_\theta ((j - i - 1)//N | s_i, \phi(s_j))]$
end for
for policy training steps... do
    Sample relabeled data $(s_i, s_{i+1}, \phi(s_j)), j > i$
    Compute $c(s_i, \phi(s_j)) = 1\{\phi(c_{i+1}) \neq \phi(s_j)\}/B$
    Compute $\hat{d}(s_i, \phi(s_j)) = -\alpha \log \mathbb{E}_{k \sim p^r_\theta(\cdot|(s_i, \phi(s_j))} \left[ e^{-k/(B\alpha)} \right]$
    Compute $\hat{d}(s_{i+1}, \phi(s_j)) = -\alpha \log \mathbb{E}_{k \sim p^r_\theta(\cdot|(s_{i+1}, \phi(s_j))} \left[ e^{-k/(B\alpha)} \right]$
    Compute adv $= \hat{d}(s_i, \phi(s_j)) - c(s_i, \phi(s_j)) - \hat{d}(s_{i+1}, \phi(s_j))$
    Update $\psi$ via $\max_\psi \mathbb{E}_{\mathcal{D}_r} \left[ e^{\text{adv}/\beta} \log \pi_\psi(a_i | s_i, \phi(s_j)) \right]$
end for
Return: $\theta, \psi$
```

C. Experiments

We were unable to fit all experiments, ablations and learning curves within the body of the main paper. In this section, we include full learning curves for all experiments featured in the main paper, and ablations on more environments. This section is designed to mimic the main body of the paper.

C.1. How does DWSL perform?

We primarily evaluated the performance of DWSL on the following environments. In the main body of the paper, we omit state results for the Gym Robotics benchmarks from the main body of the paper for space. As expected, bootstrapping-based RL methods perform best on these benchmarks.

Datasets and Environments

Gym Robotics. In the Fetch environments, the agent is tasked with manipulating a block in different ways, namely pushing, pick and place, and sliding like in air hockey. The goal extraction function ϕ yields the location of the cube from the state

Dataset	Mean	Mediam	75th %ile	90th %ile
Fetch Push Image, 250K Noise 1.0	13.43	8	25	35
Fetch Pick Image, 250K Noise 1.0	3.34	0	2	11
Fetch Slide Image, 250K Noise 0.5	4.13	0	5	15
Fetch Push Image, 250K Noise 2.0	3.28	0	1	13
Fetch Pick Image, 250K Noise 2.0	0.7	0	0	0
Fetch Slide Image, 250K Noise 1.0	1.73	0	1	5

Table 3. Here we show different return statistics of the noisy Fetch Image datasets that we create. As can be seen, the median trajectory return in 5 of 6 datasets is zero, showing that the datasets we test on are indeed extremely sub-optimal.

vector. In the Hand Reach environment, a shadow hand robot is commanded to reach a specific hand configuration, and ϕ gives the locations of the fingers. For state-based experiments, we take the offline GCRL datasets from Yang et al. (2022) which are either collected randomly, or from an expert policy with Gaussian noise of standard deviation 0.2. We use the same 90% random, 10% expert split from Ma et al. (2022). However, this type of dataset inherently disadvantages imitation learning methods, as one could expect to do better by just discarding the random data. Thus, we also create our own suboptimal datasets by training expert agents on each environment with TD3+HER, and add larger amounts of independent Gaussian noise, usually standard deviation 1 or 2. Note that the action space for these environments range from -1 to 1, thus this amount of noise results in very sub-optimal data. For the visual Gym Robotics environments, we also collect offline datasets using TD3+HER agents, but render the environment to RGB images of resolution 64x64. In all visual experiments ϕ is the identity. At test time we construct goal images by moving agents into a state deemed "successful" by the underlying state environment. Vision datasets are smaller, consisting of 250K transitions, except for Hand Reach where we used 500K transitions from a noisy expert, and 1 million transitions comprised of 90% random interactions and 10% expert. When constructing our own 90% random, 10% expert datasets, our expert policy also has Gaussian noise of standard deviation 0.2 as in Yang et al. (2022). In Table 3 we provide return statistics of the Fetch datasets we created.

Franka Kitchen. We take the Franka Kitchen demo dataset from Gupta et al. (2020) and make it goal-conditioned as done in Cui et al. (2022) by choosing ϕ to be the identity. We use the same train/test split as done in Cui et al. (2022), and use the final states of demo trajectories as goal states.

Robomimic. The Robomimic datasets from Mandlekar et al. (2021) are intended for unconditional behavior cloning. We make them goal-conditioned by setting ϕ to be the position of the object of interest, either the can for bin moving, or the square for peg insertion. We also discard the relative positions of the object to the robot end effector as we found including these components in the state hurt performance in the goal-conditioned setting. This corresponds to the latter seven components of the "object" key in robomimic. We use the same datasets, and set the evaluation horizon to 500.

Baselines

We compare our method to the following baselines on all environments:

GCSL. This purely supervised technique from Ghosh et al. (2021) uses hindisght relabeling with imitation learning losses.

WGCSL. This method from Yang et al. (2022) modifies GCSL by computing advantage weights using Q-Learning. WGCSL contains three weighting components: DRW which weights by the discount factor, GAEW which weights by the advantage from the learned Q function (this is the most important component), and BAW which retains only the best advantages over time. Following Ma et al. (2022), we implement WGCSL with BAW and GAEW, but fix an error in the advantage calculation used in Ma et al. (2022). This leads to WGCSL performing better than reported in Ma et al. (2022).

GoFAR. Ma et al. (2022) propose using a state-matching objective for GCRL. They learn a reward discriminator and a bootstrapped value function for weighting the imitation learning loss. We found that this method was unstable on image datasets when using only actor gradients, causing some runs to crash. When reporting results we considered only up until the first seed crashed. This still resulted in better results than using gradients from the value function.

GCIQL. We modify IQL from Kostrikov et al. (2022) to be goal-conditioned. IQL is a state-of-the-art offline RL algorithm that uses expectile regression to implicitly estimate maximal values. Unlike all other methods, GCIQL uses an actor, critic, and value network.

DWSL-B. Instead of using supervised learning as in DWSL, in this variant we learn distance distributions using bootstrapping, similar to the distributional Q-learning approach used in Eysenbach et al. (2019). However, unlike in Eysenbach et al. (2019), we only learn the behavioral distance distribution of our dataset, rather than perform Q-learning. Thus, to perform additional policy improvement, we still extract distance estimates using the LogSumExp as in DWSL.

Below we include results for state-based Gym Robotics experiments in Table 4, and full learning curves for all other experiments on different datasets.

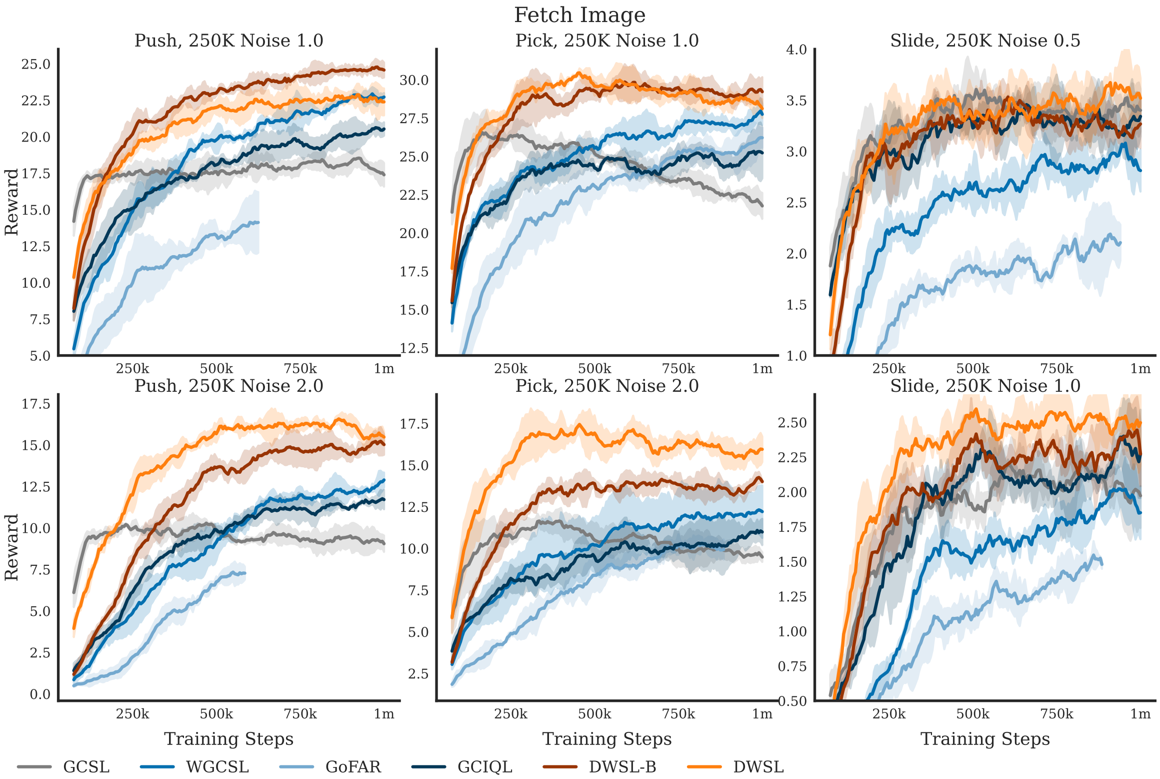

Figure 4. Learning curves for the Fetch image datasets.

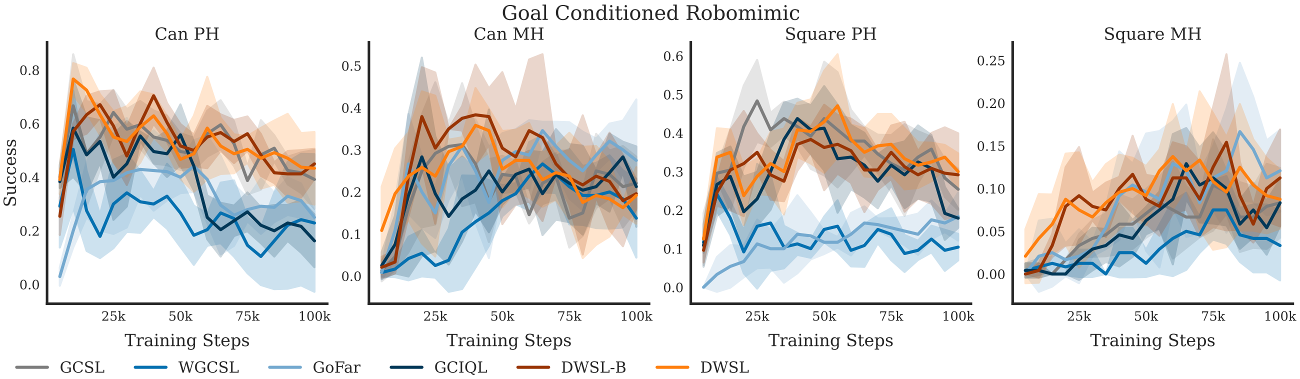

Figure 5. Learning curves for Robomimic.

Environment	Dataset	GCSL	WGCSL	GoFAR	GCIQL	DWSL-B	DWSL
Fetch Push	2M 90% R 10% E	27.97 ± 1.17	**37.14** ± **0.68**	**37.56** ± **0.50**	**36.65** ± **0.59**	34.57 ± 0.96	31.60 ± 0.99
	500K Noise 2	17.88 ± 0.29	**29.87** ± **1.56**	27.85 ± 1.41	**29.67** ± **1.12**	25.51 ± 1.80	24.23 ± 1.63
Fetch Pick	2M 90% R 10% E	23.81 ± 1.00	**34.90** ± **1.03**	**33.36** ± **1.63**	**34.23** ± **1.61**	32.37 ± 0.91	30.51 ± 1.61
	500K Noise 2	11.37 ± 0.98	**19.76** ± **3.57**	17.03 ± 2.09	**19.37** ± **0.70**	18.50 ± 1.68	17.44 ± 2.12
Fetch Slide	2M 90% R 10% E	4.73 ± 0.56	**9.50** ± **0.48**	6.82 ± 0.94	**9.13** ± **0.49**	8.14 ± 0.58	8.06 ± 0.68
	500K Noise 0.2	3.95 ± 0.40	**11.85** ± **0.50**	7.88 ± 0.42	11.25 ± 0.77	9.90 ± 0.52	10.03 ± 0.33
2M 90% R 10% E	1M 90% R 10% E	6.39 ± 0.69	**26.98** ± **2.51**	18.82 ± 3.68	17.55 ± 2.67	19.10 ± 0.80	18.13 ± 2.17
	500K Noise 0.2	0.95 ± 0.54	**30.30** ± **0.55**	28.57 ± 2.47	**30.08** ± **1.14**	27.34 ± 1.56	26.41 ± 1.31

Table 4. Results for state-based Gym Robotics datasets. Bolded numbers are within 95% of the best score. We run four seeds in state-based domains. While bootstrapping-based RL methods perform best on these datasets, DWSL still exhibits policy improvement and consistently outperforms GCSL.

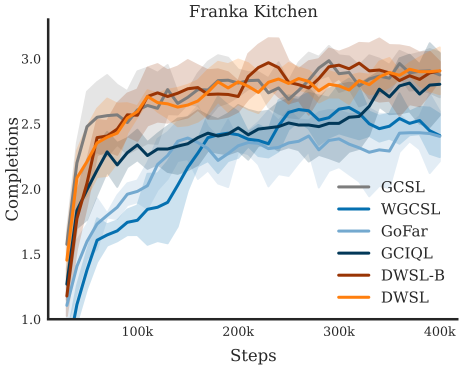

Figure 6. Learning curve for Franka Kitchen

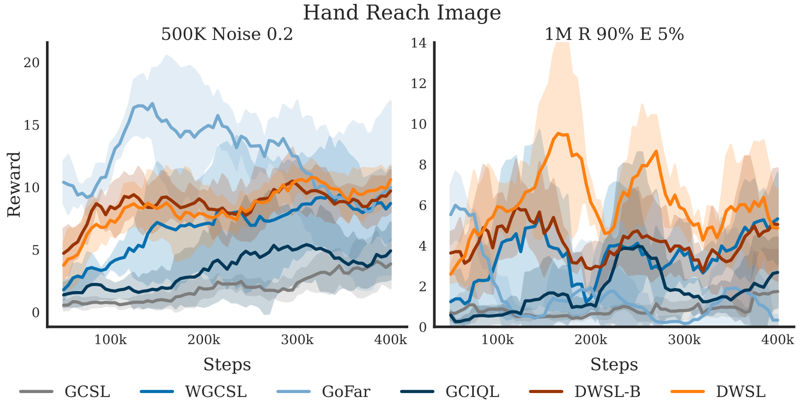

Figure 7. Learning curves for the Hand Reach image datasets.

C.2. Does DWSL use the right objective?

In this section we compare DWSL's supervised objective to different possibilities. We find the most impact on datasets with more biased actions, i.e. where the mean action for each state is less optimal. For datasets comprised of noisy rollouts of a trained policy, where actions are less biased, we suspect that the average computed distance is highly correlated with the optimal distance. When we consider datasets with random data as well, which introduces more action bias, the advantage of DWSL's objective becomes more clear. Specifically, these datasets contain 90% Random data, and 10% Expert data with 0.2 standard deviation Gaussian noise. For the state based dataset with contain 90% Random data, and 10% Expert data, we decrease the value of α from 1 to 0.1. This lets DWSL be less constrained to the random data. Like in our main paper, the image environments stay at $\alpha = 1$. Results can be found in Figures 9 and 8. In the next section, we also include results for different supervised learning methods on AntMaze.

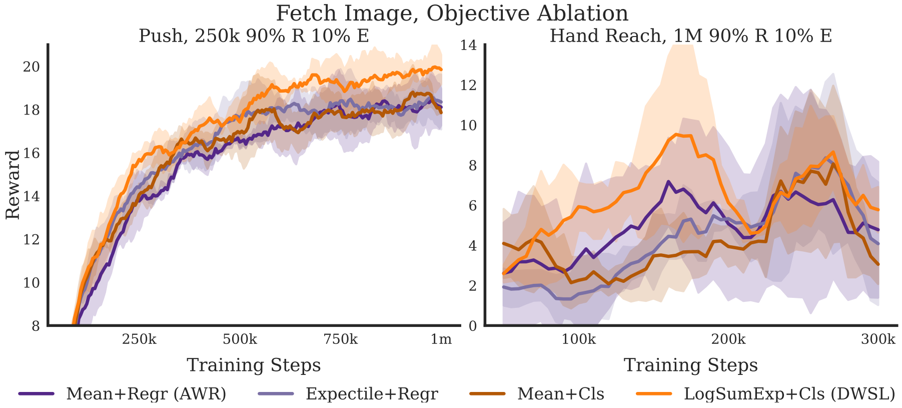

Figure 8. Objective ablation on image datasets.

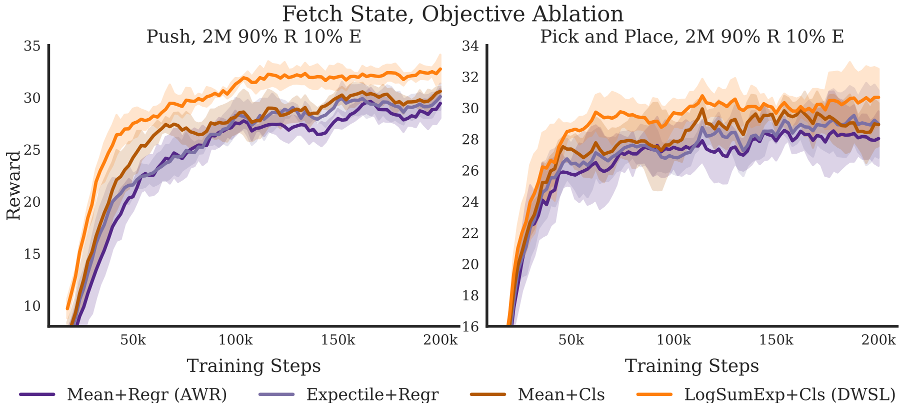

Figure 9. Objective ablation on state datasets.

C.3. When does DWSL fail?

In this section we consdier the AntMaze task from Fu et al. (2020). In AntMaze, the agent learn to navigate mazes of varying sizes by stitching together offline trajectories. While this task is usually learned from rewards, we make it goal-conditioned by setting ϕ to be the (x, y) position of the agent in the maze. Below, we include full learning curves for all methods on AntMaze. We also include DWSL-B, which was omitted from the main paper for space. In Table 5, we also include results for different supervised objectives on the AntMaze benchmark. These results, which show that all supervised methods suffer on the AntMaze stitching data. This indicates that our use of classification over regression is not inhibiting DWSL's performance in these settings.

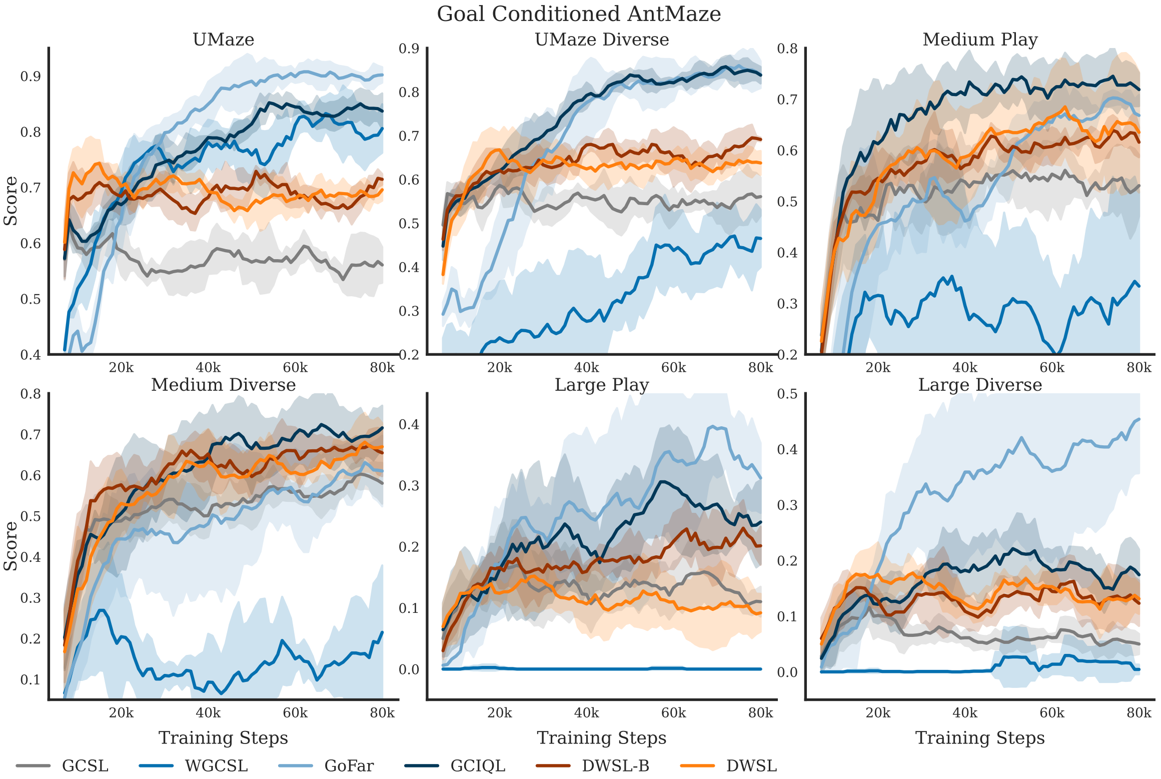

Figure 10. Learning curves for the AntMaze datasets.

Dataset	GCSL	AWR	Expectile+Regr	DWSL
UMaze	64 ± 2	68 ± 4	72 ± 2	**74 ± 4**
UMaze Diverse	59 ± 1	61 ± 5	60 ± 6	**67 ± 3**
Medium Play	56 ± 6	59 ± 7	61 ± 4	**69 ± 8**
Medium Diverse	60 ± 3	61 ± 4	59 ± 12	**68 ± 6**
Large Play	**17 ± 5**	15 ± 3	**17 ± 7**	15 ± 5
Large Diverse	12 ± 3	15 ± 6	17 ± 4	**18 ± 2**

Table 5. Performance on the AntMaze benchmark with different supervised learning algorithms. DWSL maintains the highest performance overall.

C.4. How robust is DWSL?

In this section we evaluate the robustness of DWSL to various properties, such as architecture size, hyperparameters, and relabeling ratio. Due to space limitations, we did not include all of these ablations in the main body of the paper. We include them below.

Larger Architectures. We experiment with using larger ResNet18-based encoders with the Robomimic architecture (Mandlekar et al., 2021). We concatenate current and goal images along the channel axis. Overall, we find improved performance and faster convergence (or overfitting) with larger architectures. Full results with ResNet architectures on the Gym Fetch Image datasets are shown in Figure 11.

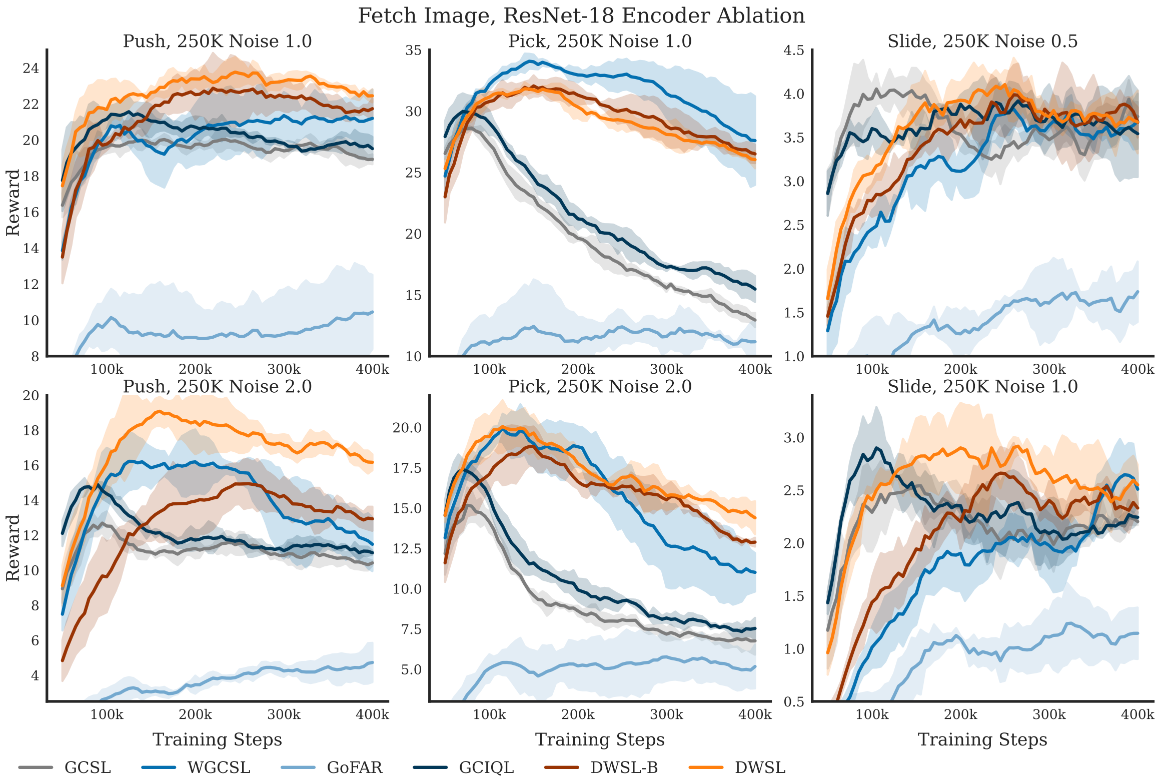

Figure 11. Learning curves for the Fetch image datasets using the ResNet18 Architecture from RoboMimic.

Image Gradients. We tried training our image encoders using gradients from both the actor and value/distance function for different offline GCRL algorithms. Interestingly, we got better performance when using only gradients from the actor network. We believe that this is because supervised learning and action prediction provide a stronger representation learning signal. We present results ablating this decision on the "Fetch Push Noise 2.0" dataset below, where actor-only gradients outperform or match "both" gradients for all methods. We therefore use actor-only gradients on all methods in image domains for consistency.

Algorithm	**Actor**	**Both**
WGCSL	**12.89** ± 0.36	11.27 ± 1.68
GoFAR	**7.34 ± 0.44**	5.71 ± 0.93
GCIQL	**11.76 ± 0.38**	**11.76 ± 0.48**
DWSL-B	**15.23 ± 0.70**	**15.51 ± 0.47**
DWSL	**16.57 ± 0.18**	**16.34 ± 0.02**

Table 6. Comparing learning the image encoder using actor-only gradients, or "both" gradients (where a learned value/distance function also provides gradients), on the "Fetch Push Noise 2.0" dataset.

Robustness to Image Augmentations. Value functions learned by DWSL exhibit more robustness to random image shifting and cropping augmentations than those learned by temporal difference based methods. We test this by taking ten expert trajectories on the Fetch Push Image task and plotting the value function learned from the "Fetch Push Noise 2.0" dataset at each state sequentially. A perfect value function would almost linearly increase from the start of the trajectory to the end. To quantitatively measure this, we compute the Pearson correlation coefficient of predicted values with the timesteps in each demonstration. Our results are summarized in Table 7 and depicted in Figures 12 and 13. DWSL has the highest correlation coefficient both with and without augmentations. When we add image augmentations, the correlation coefficient of temporal difference based methods drops significantly. Qualitatively, we see a much worse linear relationship. This again indicates the added robustness gained by using supervised learning instead of bootstrapping.

	WGCSL	**GoFar**	**GCIQL**	**DWSL**
No Aug	0.921	0.937	0.927	**0.963**
Random Crop	0.808	0.828	0.899	**0.957**
Difference	0.113	0.109	0.028	**0.006**

Table 7. Correlation coefficients with and without image augmentations for ten expert trajectories on the Fetch Push Image Noise 2.0 dataset.

Robustness to Hyperparameters. Because DWSL uses only supervised learning, we posit that it will exhibit a higher level of robustness to hyperparameters than Q-Learning approaches. We try $\alpha \in [0.1, 1, 10]$ and $\beta = [0.01, 0.05, 0.25]$ in state-based Gym Robotics environemnts. Different values of α tradeoff how conservative DWSL is. Thus, for some datasets lower values of α may work better, particularly when there is more random data. We include full results on all tasks with $\alpha = 1$ in Table 8.

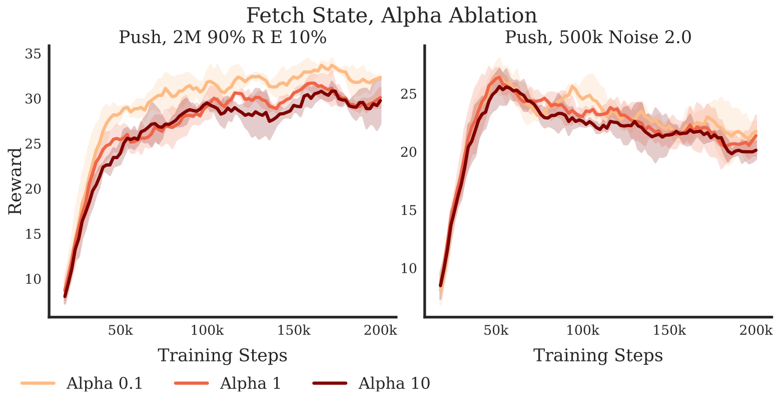

Figure 14. α ablation on Fetch Push state datasets.

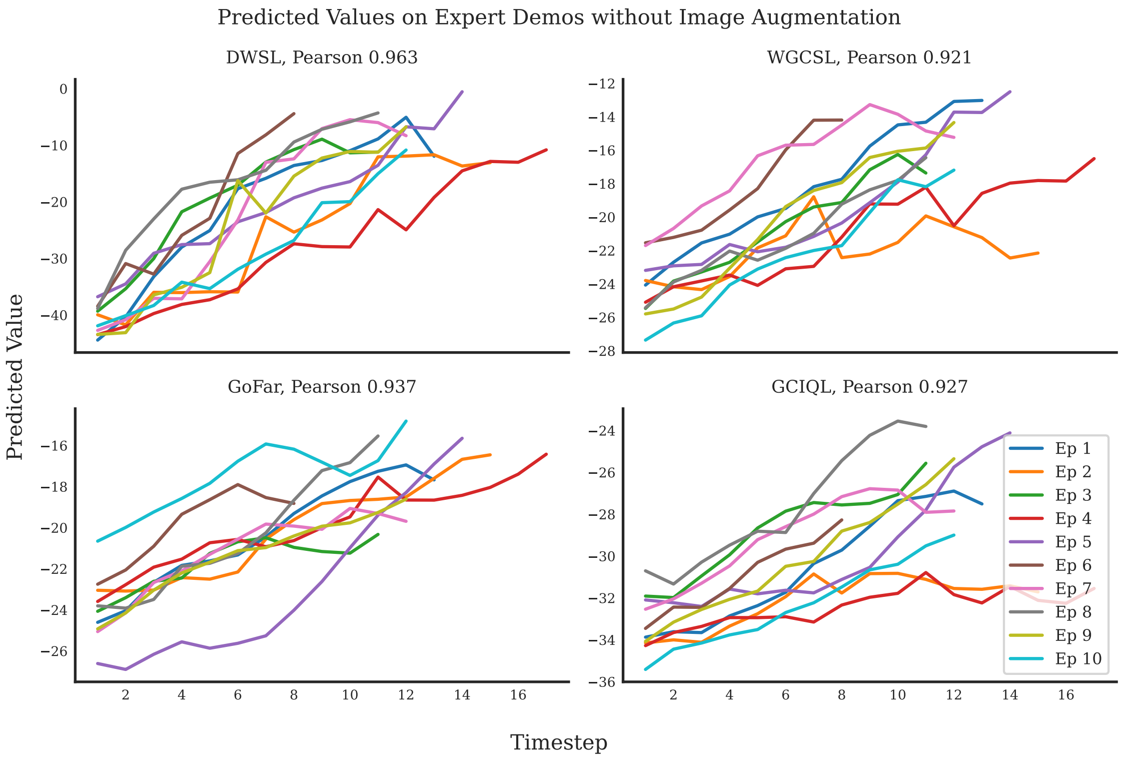

Figure 12. Plots of learned values versus timesteps on ten expert trajectories without any image augmentation.

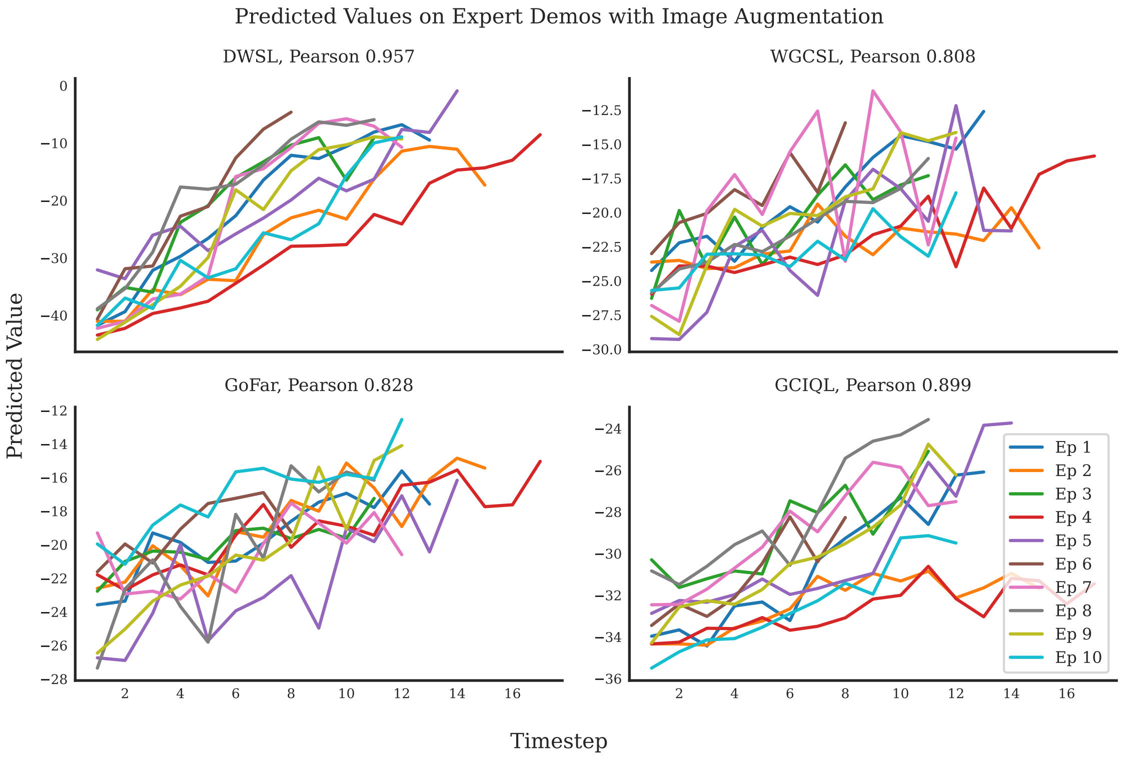

Figure 13. Plots of learned values versus timesteps on ten expert trajectories with random image cropping augmentation.

Dataset	$\alpha = 1$	$\alpha = 0.1$
Fetch Push Image, 250K Noise 1.0	22.88 ± 0.61	24.07 ± 0.33
Fetch Pick Image, 250K Noise 1.0	30.46 ± 0.23	29.53 ± 0.16
Fetch Slide Image, 250K Noise 0.5	3.67 ± 0.07	3.61 ± 0.16
Fetch Push Image, 250K Noise 2.0	16.57 ± 0.18	14.70 ± 0.23
Fetch Pick Image, 250K Noise 2.0	17.45 ± 0.47	15.84 ± 0.59
Fetch Slide Image, 250K Noise 1.0	2.60 ± 0.13	2.72 ± 0.12
Hand Reach Image, 500K Noise 0.2	10.79 ± 1.20	10.20 ± 2.74
Hand Reach Image, 1M 90% 10% E	9.53 ± 3.82	6.57 ± 0.98
Franka Kitchen	2.92 ± 0.04	2.88 ± 0.10
RoboMimic Can, 100 PH	77 ± 5	70 ± 11
RoboMimic Can, 300 MH	36 ± 8	32 ± 7
RoboMimic Square, 100 PH	47 ± 12	47 ± 5
RoboMimic Square, 300 MH	14 ± 5	13 ± 5
Antmaze UMaze	74 ± 4	76 ± 4

Table 8. Comparison between $\alpha = 1$ and $\alpha = 0.1$ for all the tasks we study

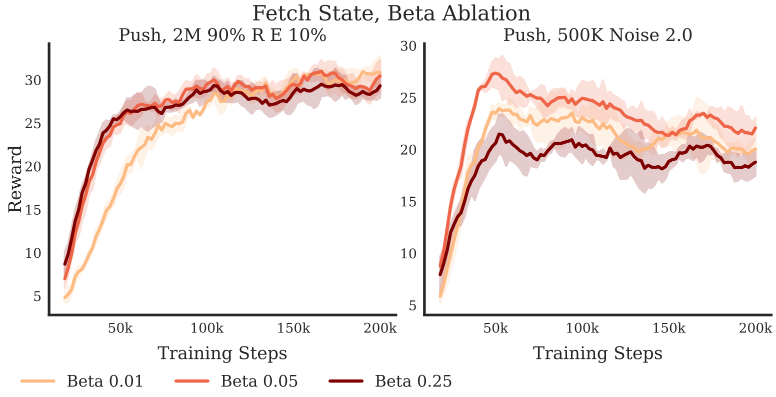

Figure 15. β ablation on Fetch Push state datasets.

Relabeling Ratio. While the *learning from offline interaction* setting we deal with assumes data does not come with goal labels, here we investigate the effect of having goal labels on performance. We find that DWSL is relatively robust to the relabeling ratio. On the other hand, Q-learning methods like WGCSL seem to be more sensitve. We provide analysis of this in the learning curves below.

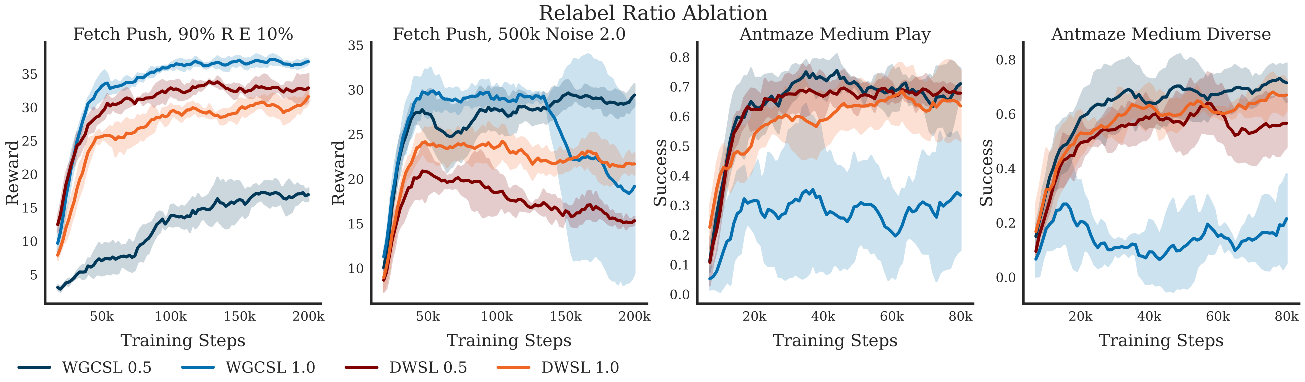

Figure 16. Relabel ratio ablation on Fetch Push and AntMaze Medium Diverse and Play.

Performance Versus Dataset Size. We test how the performance of different methods vary when changing the size of the dataset for Visual Fetch Push with noise standard deviation 2.0 and include the results in Figure 17. We find that the performance of all methods significantly drops as the size of the dataset is reduced. DWSL maintains the highest performance at all dataset sizes.

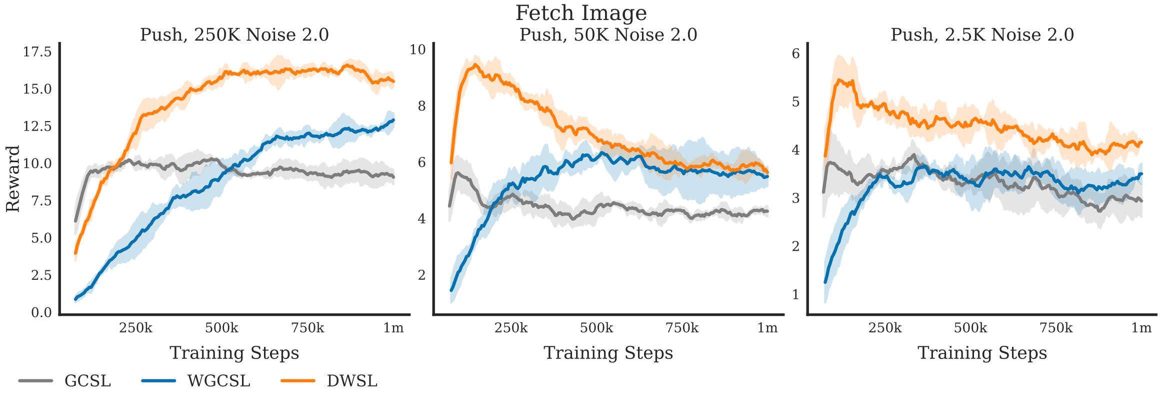

Figure 17. Dataset size comparison on the Fetch Push task from Pixels.

D. Implementation Details

D.1. Architectures

State. For all state-based experiments, we use MLPs with ReLU activations. For the policy network, we include a final tanh activation to normalize the outputs to each environment's action space of [-1, 1]. For all Gym Robotics environments, we use three hidden layers of size 256 as done in prior work (Yang et al., 2022; Ma et al., 2022). For all other state environments, we use two hidden layers of size 512, as larger networks have been shown to perform better for BC on Franka Kitchen, AntMaze, and Robomimic (Emmons et al., 2022; Mandlekar et al., 2021).

Image. For our main image-based experiments, we adapt the network architectures used in Yarats et al. (2021). The main difference is because we operate in the goal-conditioned setting, the input to our convolutional encoder is the concatenation of the current image observation and goal image along the channel dimension. Consequently, we adjust the size of the initial convolutional filters to be compatible with a larger channel dimension. We share the visual encoder across all networks used for each algorithm. Representations from the encoder are first fed to a trunk module (specific to each network) consisting of a single fully-connected layer normalized by LayerNorm, and then tanh applied to the 50 dimensional output of this layer. All downstream networks consist of 2-layer MLPs with ReLU activations, with hidden dimension 1024. The policy networks again use tanh activations for output normalization.

For our ablation experiments on encoder architecture, we replace the previously described convolutional encoder with the ResNet-18 based encoder used in Mandlekar et al. (2021), which consists of ResNet-18 followed by a spatial-softmax layer (Finn et al., 2016) with 64 keypoints. We similarily adjust the initial convolutional filters of the ResNet-18 to support the concatenated image observations, and the rest of its parameters are initialized from ImageNet pre-training. We remove the trunk module, but otherwise the other components of each network remain identical.

D.2. Model Selection

Model selection is a challenging problem in offline RL (Mandlekar et al., 2021; Emmons et al., 2022), particularly as validation loss does not always correlate with performance. As we compare supervised algorithms, like our method, with bootstrapping based methods, model selection is even more complicated. This is because supervised methods tend to train faster but can also overfit the data more easily, while RL approaches usually take longer to converge. Prior works in IL (Mandlekar et al., 2021) compute the maximum evaluation performance of each seed, and average the maximums. This type of evaluation can be too optimistic about perfect model selection. Prior works in RL simply report average peformance after a fixed number of steps. If we trained RL methods to convergence, the performance of supervised methods, like DWSL, could deteriorate from overfitting, giving an unfair advantage to RL approaches. To balance both of these interests, we make learning curves for all methods by averaging results across multiple seeds. We then report the mean and standard deviation

Algorithm	Hyperparameter	Value
WGCSL	β	1
	Max clip	10
	Target update frequency	20
	Polyak coefficient	0.05 (0.1 on images)
GoFAR	β	1
	Max clip	10
	Target update frequency	20
	Polyak coefficient	0.05
GCIQL	β	1
	Max clip	10
	Target update frequency	20
	Polyak coefficient	0.05
	Expectile	0.7
DWSL-B	β	0.05
	Max clip	10
	Target update frequency	20
	Polyak coefficient	0.05
	α	1
DWSL	β	0.05
	Max clip	10
	α	1

Table 9. Algorithm Specific Hyperparameters

of the highest point on the curves. This is equivalent to early-stopping for each method at the point where it performs best. Because of this, the evaluation frequency and number of evaluation episodes can influence the results. We keep this the same across all methods per dataset, and report the values we used for each environment in Table 10.

D.3. Hyperparameters

We use the Adam optimizer for all experiments. For all methods, we relabel goals for each state by sampling uniformly from all future states in its trajectory. For baselines that use discount factors, we use $\gamma = 0.98$ for Gym Robotics environments, and $\gamma = 0.99$ for the remaining environments. We ran four seeds for all state experiments, and 3 seeds for all image experiments. For algorithm specific hyperparameters, we include them in Table 9. “Max clip” refers to the maximum exponentiated advantage weight we clip to. For other hyperparamters common to all methods, we include them in Table 10.

For DWSL-B and DWSL, we tuned β using state-based Fetch Push. We also tuned the expectile value of GCIQL using state-based Fetch Push. We did not tune α for DWSL-B or DWSL. Other algorithm specific hyperparameters are default values used in past work (Yang et al., 2022; Ma et al., 2022).

D.4. Rewards

State. For methods that use rewards (namely offline GCRL), we assume access to a sparse reward function $r(s, a, g) = -1\{||\phi(s) - g||_2 < d\}$ that detects whether a state is within some distance threshold d of a goal. These distance thresholds d are based on the default values used to determine success in each environment. For fair comparison with methods that learn distances, including DWSL, we relabel the distance between states s_i and s_j as 1, rather than the default value of $j - i$, if $r(s_i, a, \phi(s_j))$ = 0. Robomimic and Franka Kitchen, however, did not originally support goal-conditioning. Thus, we use the strict equality scheme as described below for images.

Image. For image domains, we do not assume access to a ground-truth sparse reward function, because it is non-trivial to assess if a state is considered to have reached a goal given only image observations. Instead, we assign rewards to relabeled trajectories based only on transitions in the data, e.g. $r(s, a, \phi(s')) = 0$ if and only if s' is the state that immediately follows s in its trajectory, and -1 otherwise.

Environment	Horizon	Learning Rate	Batch Size	Network Arch	Train Steps	Eval Freq	Eval Ep	Plot Window	DWSL N	DWSL B
Fetch State	50	0.0005	512	[256, 256, 256]	200k	2000	20	10	1	50
Fetch Image	50	0.0003	256	DrQv2	1m	5000	100	15	1	50
Encoder ablation	50	0.0003	256	ResNet18+[1024, 1024]	400k	5000	100	10	1	50
Hand State	50	0.0005	512	[256, 256, 256]	200k	2000	20	10	1	50
Hand Image	50	0.0003	256	DrQv2	400k	5000	50	10	1	50
Franka Kitchen	280	0.0005	512	[512, 512]	400k	10000	20	4	2	140
Robomimic PH	500	0.0005	512	[512, 512]	100k	5000	25	2	2	80
Robomimic MH	500	0.0005	512	[512, 512]	100k	5000	25	2	2	160
Antmaze Umaze	700	0.0005	512	[512, 512]	80k	1000	50	8	3	100
Antmaze Medium	900	0.0005	512	[512, 512]	80k	1000	50	8	3	150
Antmaze Large	1000	0.0005	512	[512, 512]	80k	1000	50	8	3	200

Table 10. Domain Specific Hyperparameters

D.5. Image Augmentation

For our image-based experiments, we apply random shift augmentation as done in Yarats et al. (2021). For each training transition (state, next state, goal), we apply the same sampled augmentation to each image, and randomize augmentations across transitions.