Appendices

A Additional Visualizations and Experiments for DR3

In this section, we provide visualizations and diagnostic experiments evaluating various aspects of feature co-adaptation and the DR3 regularizer. We first provide more empirical evidence showing the presence of feature co-adaptation in modern deep offline RL algorithms. We will also visualize DR3 inspired from the implicit regularizer term in TD-learning alleviates rank collapse discussed in Kumar et al. (2021). We will compare the efficacies of the explicit regularizer induced for different choices of the noise covariance matrix M (Equation 4), understand the effect of dropping the stop gradient term in ou practical regularizer and finally, perform diagnostic experiments visualizing if the Q-networks learned with DR3 resemble more like neural networks trained via supervised learning, measured in terms of sensitivity and robustness to layer reinitialization (Zhang et al., 2019).

A.1 More Empirical Evidence of Feature Co-Adaptation

In this section, we provide more empirical evidence demonstrating the existence of the feature co-adaptation issue in modern offline RL algorithms such as DQN and CQL. As shown below in Figure A.1, while the average dataset Q-value for both CQL and DQN exhibit a flatline trend, the dot product similarity for consecutive state-action tuples generally continues to increase throughout training and does not flatline. While DQN eventually diverges in Seaquest, the dot products increase with more gradient steps even before divergence starts to appear.

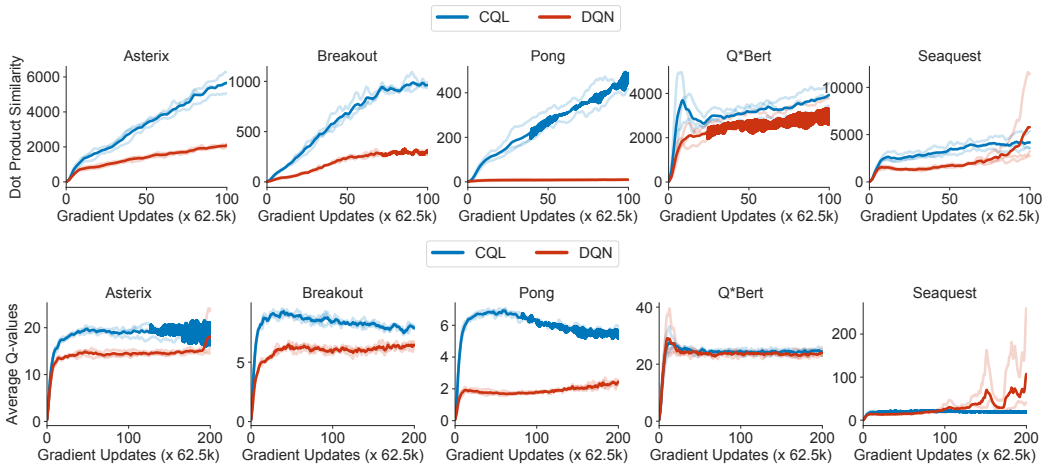

Figure A.1: **Demonstrating feature co-adaptation on five Atari games with standard offline DQN and CQL, averaged over 3 seeds.** Observe that the feature dot products continue to rise with more training for both CQL and DQN, indicating the presence of co-adaptation. On the other hand, the average Q-values exhibit a converged trend, except on Seaquest. Further, note that the dot products continue to increase for CQL even though CQL explicitly corrects for out-of-distribution action inputs.

A.2 Layer-wise structure of a Q-network trained with DR3

To understand if DR3 indeed makes Q-networks behave as if they were trained via supervised learning, utilizing the empirical analysis tools from Zhang et al. (2019), we test the robustness/sensitivity of each layer in the learned network to re-initialization, while keeping the other layers fixed. This tests if a particular layer is *critical* to the predictions of the learned neural network and enables us to reason about generalization properties (Zhang et al., 2019; Chatterji et al., 2019). We ran CQL and REM and saved all the intermediate checkpoints. Then, as shown in Figure A.2, we first loaded a checkpoint (x-axis), and computed policy performance (shaded color; colorbar) by re-initializing a given layer (y-axis) of the network to its initialization value before training for the same run.

Note in Figure A.2, that while almost all layers are absolutely critical for the base CQL algorithm, utilizing DR3 substantially reduces sensitivity to the latter layers in the Q-network over the course of

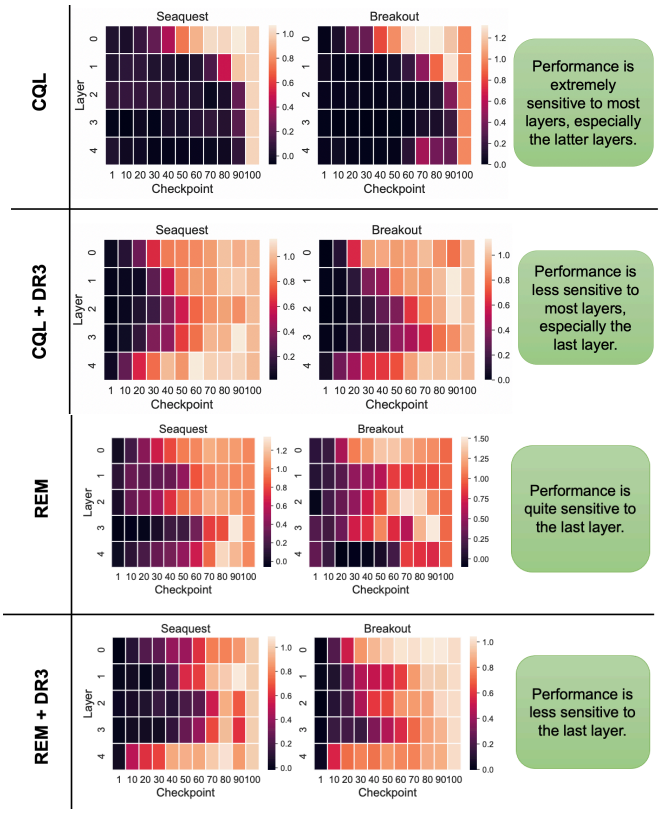

Figure A.2: **CQL vs CQL + DR3 and REM vs REM + DR3.** Average robustness of the learned Q-function to re-initialization of all layers to different checkpoints over the course of training created based on the protocol from Zhang et al. (2019). The colors in the heatmap indicate performance of the reinitialized checkpoint, normalized w.r.t. the checkpoint without any change to layers. Note that while CQL and REM are more sensitive (i.e., less robust) to reinitialization of all the layers especially the last layer, CQL + DR3 and REM + DR3 behave closer to supervised learning, in the sense that they are more robust to reinitialization of layers of the network, especially the last layer.

training. This is similar to what Zhang et al. (2019) observed for supervised learning, where the initial layers of a network were the most critical, and the latter layers primarily performed near-random transformations without affecting the performance of the network. This indicates that utilizing DR3 alters the internal layers of a Q-network trained with TD to behave closer to supervised learning.

A.3 Rank Collapse is Alleviated With DR3

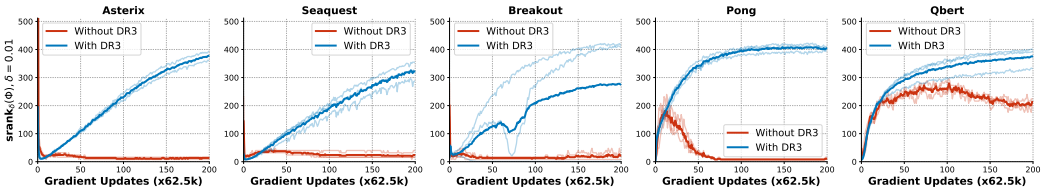

Figure A.3: **Comparing the feature ranks for CQL and CQL + DR3.** Observe that utilizing DR3 successfully alleviates the rank collapse issue noted in prior work without explicitly correcting for it.

Prior work (Kumar et al., 2021) has shown that implicit regularization in TD-learning can lead to a feature rank collapse phenomenon in the Q-function, which hinders the Q-function from using its full representational capacity. Such a phenomenon is absent in supervised learning, where the feature rank does not collapse. Since DR3 is inspired by mitigating the effects of the term in the implicit regularizer (Equation 4) that only appears in the case of TD-learning, we wish to understand if utilizing DR3 also alleviates rank collapse. To do so, we compute the effective rank $\mathrm{srank}_\delta(\phi)$

Table A.1: Normalized interquartile mean performance with 95% stratified bootstrap CIs (Agarwal et al., 2021) across 17 Atari games of REM, REM + $\Delta'(\Phi)$ (Stop gradient in DR3), REM + DR3 after 6.5M gradient steps for the 1% setting and 12.5M gradient steps for the 5%, 10% settings. Observe that REM + $\Delta'(\phi)$ also improves over the base REM method significantly, by about 130%, even though $\Delta'(\phi)$ is generally comparable and somewhat worse than the DR3 regularizer used in the main paper.

Data	REM	REM + $\Delta'(\Phi)$	REM+DR3
1%	4.0 (3.3, 4.8)	15.0 (13.4, 16.6)	16.5 (14.5, 18.6)
5%	25.9 (23.4, 28.8)	55.5 (50.8, 59.8)	60.2 (55.8, 65.1)
10%	53.3 (51.4, 55.3)	67.7 (64.7, 71.3)	73.8 (69.3, 78)

metric of the features learned by Q-functions trained via standard Q-learning, standard Q-learning with DR3 explicit regularizer and offline SARSA. As shown in Figure A.3, for the case of five Atari games, utilizing DR3 alleviates the rank collapse issue completely. This is potentially surprising, because no term in the practical DR3 regularizer explicitly aims to increase rank: feature dot products can be made smaller while retaining low ranks by simply controlling the magnitude. But, as we observe, utilizing DR3 enables learning high-rank features, thus likely indicating that correcting for appropriate terms in $\mathrm{R_{TD}}(\theta)$ can address some of the previously observed pathologies in TD-learning.

A.4 Induced Implicit Regularizer: Theory And Practice

In this section, we compare the performance of our practical DR3 regularizer to the regularizers (Equation 4) obtained for different choices of M, such as M induced by noise, studied in previous work, and also evaluate the effect of dropping the stop gradient function from the practical version of our regularizer.

Empirically comparing the explicit regularizers for different noise covariance matrices, M. The theoretically derived regularizer (Equation 4) suggests that for a given choice of M, the following equivalent of feature dot products should increase over the course of training:

$$\Delta_M(\theta) := \sum_{\mathbf{s},\mathbf{a}\in\mathcal{D}} \mathrm{trace}\left[\Sigma^*_M \nabla Q(\mathbf{s},\mathbf{a}) \nabla Q(\mathbf{s}',\mathbf{a}')^\top\right]. \quad \text{(Generalized dot products)} \quad \text{(A.1)}$$

We evaluate the efficacy of the explicit regularizer that penalizes the generalized dot products, $\Delta_M(\theta)$, in improving the performance of the policy. While Σ^*_M must be explicitly computed by running fixed point iteration for every parameter iterate θ found during TD-learning – which makes this method significantly computationally expensive, we evaluated it on five Atari games. As shown in Figure A.4, the DR3 penalty with the choice of M which corresponds to label noise, and the dot product DR3 penalty used in the paper generally perform similarly on these domains, attaining almost identical learning curves on **4/5 games**, and clearly improving over the base algorithm. This hints at the possibility of utilizing other noise covariance matrices to derive an explicit regularizer.

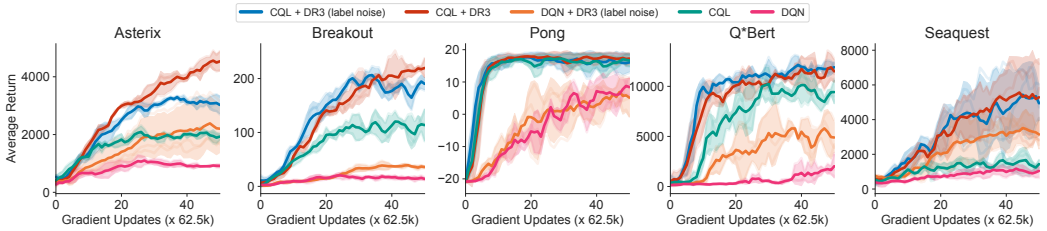

Figure A.4: **Comparing the performance of explicit penalties for two different choices of the covariance matrix M.** Observe that in all the five games the DR3 regularizer derived for the choice of M from Blanc et al. (2020) also leads to a substantial increase in performance over the base algorithm, and in four of five games, DR3 (label-noise) works just as well as DR3.

Effect of stop gradient. Finally, we investigate the effect of utilizing a stop gradient in the DR3 regularizer. We run a variant of DR3: $\Delta'(\phi) = \sum_{\mathbf{s},\mathbf{a},\mathbf{s}'} \phi(\mathbf{s},\mathbf{a})^\top [[\phi(\mathbf{s}',\mathbf{a}')]]$, with the stop gradient on the second term $(\mathbf{s}',\mathbf{a}')$ and present a comparison to the one without the stop gradient in Table A.1

for REM as the base offline method, averaged over 17 games. Note that this version of DR3, with the stop gradient, also improves upon the baseline offline RL method (i.e., REM) by **130%**. While this performs largely similar, but somewhat worse than the complete version without the stop gradient, these results do indicate that utilizing $\Delta'(\phi)$ can also lead to significant gains in performance.

B Extended Related Work

In this section, we briefly review some extended related works, and in particular, try to connect feature co-adaptation and implicit regularization to various interesting results pertaining to RL lower-bounds with function approximation and self-supervised learning.

Lower-bounds for offline RL. Zanette (2020) identifies hard instances for offline TD learning of linear value functions when the provided features are "aliased". Note that this work does not consider feature learning or implicit regularization, but their hardness result relies heavily on the fact the given linear features are aliased in a special sense. Aliased features utilized in the hard instance inhibit learning along certain dimensions of the feature space with TD-style updates, necessitating an exponential sample size for near-accurate value estimation, even under strong coverage assumptions. A combination of Zanette (2020)'s argument, which provides a hard instance given aliased features, and our analysis, which studies the emergence of co-adapted/similar features in the offline deep RL setting, could imply that the co-adaptation can lead to failure modes from the hard instance, even on standard Offline RL problems, when provided with limited data.

Connections to self-supervised learning (SSL). Several modern self-supervised learning methods (Grill et al., 2020; Chen & He, 2020) can be viewwed as utilizing some form of bootstrapping where different augmentations of the same input $(\mathbf{x} + \text{Aug}_1, \mathbf{x} + \text{Aug}_2)$ serve as consecutive state-action tuples that appear on two sides of the backup. If we may extrapolate our reasoning of feature co-adaptation to this setting, it would suggest that performing noisy updates on a self-supervised bootstrapping loss will give us feature representations that are highly similar for consecutive state-action tuples, i.e., the representations for $\phi(\mathbf{x}+\text{Aug}_1)^\top \phi(\mathbf{x}+\text{Aug}_2)$ will be high. Intuitively, an easy way for obtaining high feature dot products is for $\phi(\cdot)$ to capture only that information in $\cdot$, which is agnostic to data augmentation, thus giving rise to features that are invariant to transformations. This aligns with what has been shown in self-supervised learning (Tian et al., 2020; 2021). Another interesting point to note is that while such an explanation would indicate that highly co-adapted features are beneficial in SSL, such features can be adverse in value-based RL as discussed in Section 3.

Preventing divergence in deep TD-learning. Finally, we discuss Achiam et al. (2019) which proposes to pre-condition the TD-update using the inverse the neural tangent kernel (Jacot et al., 2018) matrix so that the TD-update is always a contraction, for every θ_k found during TD-learning. Intuitively, this can be overly restrictive in several cases: we do not need to ensure that TD always contracts, but that is eventually stabilizes at good solution over long periods of running noisy TD updates, Our implicit regularizer (Equation'4) derives this condition, and our theoretically-inspired DR3 regularizer shows that empirically, it suffices to penalize the dot product similarity in practice.

C Proof of Theorem 3.1

In this section, we will derive our implicit regularizer $R_{\text{TD}}(\theta)$ that emerges when performing TD updates with a stochastic noise model with covariance matrix M. We first introduce our notation that we will use throughout the proof, then present our assumptions and finally derive the regularizer. Our proof utilizes the analysis techniques from Blanc et al. (2020) and Damian et al. (2021), which analyze label-noise SGD for supervised learning, however key modifications need to be made to their arguments to account for non-symmetric matrices that emerge in TD learning. As a result, the form of the resulting regularizer is very different. To keep the proof concise, we will appeal to lemmas from these prior works which will allow us to bound certain concentration terms.

C.1 Notation

The noisy TD-learning update for training the Q-function is given by:

$$\theta_{k+1} = \theta_k - \eta \underbrace{\left(\sum_i \nabla_\theta Q(\mathbf{s}_i, \mathbf{a}_i) \left(Q_\theta(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma Q_\theta(\mathbf{s}'_i, \mathbf{a}'_i)) \right) \right)}_{:= g(\theta)} + \eta \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}(0, M) \quad \text{(C.1)}$$

where $g(\theta)$ denotes the parameter update. Note that $g(\theta)$ is not a full gradient of a scalar objective, but it is a form of a "pseudo"-gradient or "semi"-gradient. Let ε_k denote an i.i.d.random noise that is added to each update. This noise is sampled from a zero-mean Gaussian random variable with covariance matrix M, i.e., $\mathcal{N}(0, M)$.

Let θ^* denote a point in the parameter space such that in the vicinity of θ^*, $g(\theta) \leq \mathscr{C}$, for a small enough $\mathscr{C}$. Let $G(\theta)$ denote the derivative of $g(\theta)$ w.r.t. θ: $G(\theta) = \nabla_\theta g(\theta)$ and let $\nabla G(\theta)$ denote the third-order tensor $\nabla^2_\theta g(\theta)$. For notation clarity, let $G = G(\theta^*), \nabla G = \nabla G(\theta^*)$. Let e_i denote the signed TD error for a given transition $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i) \in \mathcal{D}$ at θ^*:

$$e_i = Q_{\theta^*}(\mathbf{s}_i, \mathbf{a}_i) - (r_i + \gamma Q_{\theta^*}(\mathbf{s}'_i, \mathbf{a}'_i)). \tag{C.2}$$

Since θ^* is a fixed point of the training TD error, $e_i = 0$. Following Blanc et al. (2020), we will assume that the learning rate in gradient descent, η, is small and we will ignore terms that scale as $\mathcal{O}(\eta^{1+\delta})$, for $\delta > 0$. Our proof will rely on using a reference Ornstein-Uhlenbeck (OU) process which the TD parameter iterates will be compared to. Let ζ_k denote the k-th iterate of an OU process, which is defined as:

$$\zeta_{k+1} = (I - \eta G)\zeta_k + \eta\varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}(0, M) \tag{C.3}$$

We will drop θ from ∇_θ to indicate that the gradient is being computed at θ^*, and drop $(\mathbf{s}_i, \mathbf{a}_i)$ from $Q(\mathbf{s}_i, \mathbf{a}_i)$ and instead represent it as Q_i for brevity; we will represent $Q(\mathbf{s}'_i, \mathbf{a}'_i)$ as Q'_i. We assume that $\nabla^2 Q_i$ is $\mathscr{L}_2$-Lipschitz and $\nabla^3 Q_i$ is $\mathscr{L}_3$-Lipschitz throughout the parameter space Θ.

C.2 Proof Strategy

For a given point θ^* to be an attractive fixed point of TD-learning, our proof strategy would be to derive the condition under which it mimics a given OU noise process, which as we will show stays close to the parameter θ^*. This condition would then be interpreted as the gradient of a "induced" implicit regularizer. If the point θ^* is not a stationary point of this regularizer, we will show that the movement θ is large when running the noisy TD updates, indicating that the regularizer, atleast in part guides the dynamics of TD-learning. To show this, we would write out the gradient update, isolate some terms that will give rise to the implicit regularizer, and bound the remaining terms using contraction and concentration arguments. The contraction arguments largely follow prior work (though with key exceptions in handling contraction with asymmetric and complex eigenvalue matrices), while the form of the implicit regularizer is different. Finally, we will interpret the resulting update over large timescales to show that learning is indeed guided by the implicit regularizer.

C.3 Assumptions and Conditions

Next, we present some key assumptions we will need for the proof. Our first assumption is that the matrix $G \in \mathbb{R}^{d\times d}$ is of maximal rank possible, which is equal to the number of datapoints n and $n \ll d$, the dimensionality of the parameter space. Crucially, this assumption do not imply that G is of full rank – it cannot be, because we are in the overparameterized regime.

Assumption A1 (G spans an n-dimensional basis.)**.** *Assume that the matrix G spans n-possible directions in the parameter space and hence, attains the maximal possible rank it can.*

The second condition we require is that the matrices $\sum_i \nabla Q_i \nabla Q_i^\top$ and M share the same n-dimensional basis as matrix G:

Assumption A2. *$\sum_i \nabla Q_i \nabla Q_i^\top$, M, and G span identical n-dimensional subspaces.*

This is a technical condition that is required. If this condition is not met, as we will show the learning dynamics of noisy TD will not be a contraction in certain direction in the parameter space and TD-learning will not stabilize at such a solution θ^*. In fact, we will utilize a stronger version of this statement for TD-learning to converge, and we will discuss this shortly.

C.4 Lemmas Used In The Proof

Next, we present some lemmas that would be useful for proving the theoretical result.

Lemma C.1 (Expressions for the first and and second-order derivatives of $g(\theta).$)**.** *The following definitions and expansions apply to our proof:*

$$G(\theta^*) = \sum_i \nabla^2 Q_i e_i + \sum_i \nabla Q_i (\nabla Q_i - \gamma \nabla Q_i')^\top$$
$$\nabla G(\theta^*)[\mathbf{v}, \mathbf{v}] = 2 \sum_i \nabla^2 Q_i \mathbf{v}\mathbf{v}^\top (\nabla Q_i - \gamma \nabla Q_i') + \sum_i \mathrm{tr}\left((\nabla^2 Q_i - \gamma \nabla^2 Q_i')\mathbf{v}\mathbf{v}^\top\right) \nabla Q_i + \nabla^3 Q_i e_i$$

Lemma C.1 presents a decomposition of the matrix G and the directional derivative of the third order tensor $\nabla G[\mathbf{v}, \mathbf{v}]$ in directions $\mathbf{v}$ and $\mathbf{v}$, which will appear in the Taylor expansion layer. Note that at θ^* since $e_i = 0$, the first term in $G(\theta^*)$ and the third term in $\nabla G(\theta^*)[\mathbf{v}, \mathbf{v}]$ vanish. Lemma C.2 derives a fixed-point recursion for the covariance matrix of the total noise accumulated in the OU-process with covariance matrix M and this will appear in our proof.

Lemma C.2 (Covariance of the random noise process ζ_k)**.** *Let ζ_k denote the OU process satisfying: $\zeta_{k+1} = (I - \eta G)\zeta_k + \eta \varepsilon_k$, where $\varepsilon_k \sim \mathcal{N}(0, M)$, where $M \succcurlyeq 0$. Then, $\zeta_{k+1} \sim \mathcal{N}(0, \Sigma)$, where Σ satisfies the discrete Lyapunov equation:*

$$\Sigma_M^* = (I - \eta G)\Sigma_M^* (I - \eta G)^\top + \eta^2 M.$$

Proof. For the OU process, $\zeta_{k+1} = (I - \eta G)\zeta_k + \eta \varepsilon_k$, since ε_k is a Gaussian random variable, by induction so is ζ_{k+1}, and therefore the covariance matrix of ζ_{k+1} is given by:

$$\Sigma_{k+1} := (I - \eta G)\Sigma_k (I - \eta G^\top) + \eta^2 M. \tag{C.4}$$

Solving for the fixed point for Σ_k gives the desired expression. □

In our proofs, we will require the following contraction lemmas to tightly bound the magnitude of some zero-mean terms that will appear in the noisy TD update under certain scenarios. Unlike the analysis in Damian et al. (2021) and Blanc et al. (2020) for supervised learning with label noise, where the contraction terms like $(I - \eta G)^k G$ are bounded by $\approx \frac{1}{k\eta}$ intuitively because $I - \eta G$ is a contraction in the subspace spanned by matrix G. However, this is not true for TD-learning directly since terms like $(I - \eta G)^k S$ appear for a different matrix S. Therefore, TD-learning will diverge from θ^* unless matrices G and M have their corresponding eigenvectors assigned to the top eigenvalues be approximately "aligned". We formalize this definition next, and then provide a proof of the concentration guarantee.

Definition 1 ((ω, C_0)-alignment)**.** *Given a positive semidefinite matrix A, let $A = U_A \Lambda_A U_A^\top$ denote its eigendecomposition. Without loss of generality assume that the eiegenvalues are arranged in decreasing order, i.e., $\forall i > j, \Lambda_A(i) \leq \Lambda_A(j)$. Given another matrix B, let $B = U_B \Lambda_B U_B^H$ denote its complex eigendecomposition, where eigenvalues in Λ_B are arranged in decreasing order of their complex magnitudes, i.e., $\forall i > j, |\Lambda_B(i)| \leq |\Lambda_B(j)|$. Then the matrix pair (A, B) is said to be (ω, C_0)-aligned if $|U_B^H(i) U_A(i)| \leq \omega$ and if $\forall\, i, \Lambda_A(i) \leq C_0 |\Lambda_B(i)|$ for a constant C_0.*

If two matrices are (ω, C_0)-aligned, this means that the corresponding eigenvectors when arranged in decreasing order of eigenvalue magnitude roughly align with each other. This condition would be crucial while deriving the implicit regularizer as it will quantify the rate of contraction of certain terms that define the neighborhood that the iterates of noisy TD-learning will lie in with high probability. We will operate in the setting when the matrix G and $\sum_i \nabla Q_i \nabla Q_i^\top$ are (ω, C_0)-aligned with each other, and matrix M and G are also (ω, C_0)-aligned (note that we can consider ω', C_0'), which will not change our bounds and therefore we go for less notational clutter). Next we utilize this notion of alignment to show a particular contraction bound that extends the weak contraction bound in Damian et al. (2021).

Lemma C.3. *Assume we are given a matrix G such that $|\lambda_i(I - \eta G)| \leq \rho_0 < 1$ for all λ_i such that $\lambda_i \neq 0$. Let $G = U \Lambda U^H$ be the complex eigenvalue decomposition of G (since almost every matrix is complex-diagonalizable). For a positive semi-definite matrix S that is (ω, C_0)-aligned with G, if $S = U_S \Lambda_S U_S^\top$ is its eigenvalue decomposition, the following contraction bound holds:*

$$||(I - \eta G)^k S|| = \mathcal{O}\left(\frac{\omega C_0}{\eta k}\right)$$

Proof. To prove this statement, we can expand $(I - \eta G)$ using its eigenvalue decomposition only in the subspace that is jointly shared by G and M, and then utilize the definition of ω-alignment to bound the terms.

$$||(I - \eta G)^k S|| = ||(I - \eta U \Lambda U^H)^k U_S \Lambda_S U_S^\top|| \tag{C.5}$$

$$= \left|\left|(UU^H - \eta U \Lambda_U U^H)^k U_S \Lambda_S U_S^\top\right|\right| \tag{C.6}$$

$$= \left|\left|U\left(I - \eta\Lambda\right)^k U^H U_S \Lambda_S U_S^\top\right|\right| \tag{C.7}$$

$$\leq \omega \cdot ||\left(I - \eta\Lambda\right)^k|| \cdot \Lambda_S \tag{C.8}$$

$$\leq \omega \cdot C_0 \cdot \left(\max_i \ |1 - \eta\Lambda(i)|^k |\Lambda(i)|\right) \tag{C.9}$$

Now we need to solve for the inner maximization term. When $\Lambda(i)$ is not complex for any i, the term above is $\lesssim 1/\eta k$ using the result from Damian et al. (2021), but when $\Lambda(i)$ is complex, this bound can only hold under certain conditions. To note when this quantity is bounded, we expand $|1 - \eta x|^k$ for some complex number $x = r(\cos\theta + \iota \sin\theta)$:

$$|1 - \eta x|^k = |(1 - \eta r\cos\theta) + \iota\eta r \sin\theta| \tag{C.10}$$

$$= \left[\sqrt{(1 - \eta r\cos\theta)^2 + \eta^2 r^2 \sin^2\theta}\right]^k = \left(1 + \eta^2 r^2 - 2\eta r\cos\theta\right)^{k/2} \tag{C.11}$$

$$\implies |1 - \eta x|^k |x| = \left(1 + \eta^2 r^2 - 2\eta r\cos\theta\right)^{k/2} r \tag{C.12}$$

$$\lesssim \frac{1}{\eta k} \quad \text{if } \eta \leq \min_i \frac{\mathrm{Re}(\Lambda(i))}{|\Lambda(i)|} \quad \text{and} \quad \infty \text{ otherwise.} \tag{C.13}$$

Plugging back the above expression in the bound above completes the proof. □

The proof of Lemma C.3 indicates that unless the learning rate η and the matrix G are such that the $|\lambda_i(I - \eta G)| \leq \rho < 1$ in directions spanned by matrix S, such an expression may not converge. This is expected since the matrix $I - \eta G$ will not contract in directions of non-zero eigenvalues if the real part $r\cos\theta$ is negative or zero. Additionally, we note that under Definition 1, we can extend several weak-contraction bounds from Damian et al. (2021) (Lemmas 9-14 in Damian et al. (2021)) to our setting.

Next, Lemma C.4 shows that the OU noise iterates are bounded with high probability when Definition 1 holds:

Lemma C.4 (ζ_k is bounded with high probability)**.** *With probability atleast* $1 - \delta$ *and under Definition 1,* $||\zeta_k|| \leq n\omega\sqrt{\eta C_0}\log\frac{1}{\delta} = \mathcal{O}(\sqrt{\eta})$.

Proof. To prove this lemma, we first bound the trace of the covariance matrix Σ_{k+1} and then apply high probability bounds on the Martingale norm concentration. The trace of the covariance matrix Σ_{k+1} can be bounded as follows (all the equations below are restricted to the dimensions of non-zero eigenvalues of G):

$$\mathrm{tr}\left[\Sigma_{k+1}\right] = \sum_{j\leq k} \mathrm{tr}\left[(I - \eta G)^j M (I - \eta G^\top)^j\right] \tag{C.14}$$

$$= \sum_{j\leq k} \mathrm{tr}\left[(UU^H - \eta U\Lambda U^H)^j M (UU^H - \eta U\Lambda U^H)^j\right] \tag{C.15}$$

$$= \sum_{j\leq k} \mathrm{tr}\left[U(I - \eta\Lambda)^j U^H U_M \Lambda_M U_M^\top U (I - \eta\Lambda)^j U^H\right] \tag{C.16}$$

$$= \sum_{j\leq k} n\omega^2 C_0 \mathrm{tr}\left[|I - \eta\Lambda|^j \cdot |\Lambda| \cdot |I - \eta\Lambda|^j\right] \tag{C.17}$$

$$\leq n\omega^2 C_0 \sum_{j\leq k} n \cdot \max_\lambda(|1 - \eta\lambda|^{2j} \cdot |\lambda|) \leq \eta n^2 C_0 \omega^2 \tag{C.18}$$

Now, we can apply Corollary 1 from Damian et al. (2021) to obtain a bound on $||\zeta_k||$ as with high probability, atleast $1 - \delta$, $||\zeta_k|| \leq \sqrt{2\mathrm{tr}(\Sigma)\log\frac{1}{\delta}} = n\omega\sqrt{\eta C_0}\log\frac{1}{\delta}$. □

C.5 Main Proof of Theorem 3.1

In this section, we present the main proof of Theorem 3.1. The proof involves two components: **(1)** the part where we derive the regularizer, and **(2)** bounding additional terms via concentration inequalities. Part **(1)** is specific to TD-learning, while a lot of the machinery for part **(2)** is directly taken from prior work (Damian et al., 2021) and Blanc et al. (2020). We focus on part **(1)** here.

Our strategy is to analyze the learning dynamics of noisy TD updates that originate at θ^*. In a small neighborhood around θ^*, we can expand the noisy TD update (Equation 3) using Taylor's expansion around θ^* which gives:

$$\theta_{k+1} = \theta_k - \eta g(\theta_k) + \eta\varepsilon_k, \;\; \varepsilon_k \sim \mathcal{N}(0, M) \tag{C.19}$$

$$\implies \theta_{k+1} = \theta_k - \eta\left(g + G(\theta_k - \theta^*) - \frac{\eta}{2}G[\theta_k - \theta^*, \theta_k - \theta^*]\right) + \eta\varepsilon_k + \mathcal{O}(\eta||\theta_k - \theta^*||^3). \tag{C.20}$$

Denoting $\nu_k := \theta_k - \theta^*$, using the fact that $||g(\theta^*)|| \leq \mathscr{C}$, we find that ν_k can be written as:

$$\nu_{k+1} = (I - \eta G)\nu_k + \varepsilon_k + \frac{\eta}{2}G[\nu_k, \nu_k] + \mathcal{O}(\eta||\nu_k||^3 + \eta\mathscr{C}) \tag{C.21}$$

Since the OU process ζ_k stays in the vicinity of the point θ^*, and follows a similar recursion to the one above, our goal would be to design a regularizer so that Equation C.21 closely follows the OU process. Thus, we would want to bound the difference between the variable ν_k and the variable ζ_k, denoted as r_k to be within a small neighborhood:

$$r_{k+1} = \nu_{k+1} - \zeta_{k+1} = (I - \eta G)\underbrace{(\nu_k - \zeta_k)}_{r_k} + \frac{1}{2}G[\nu_k, \nu_k] + \mathcal{O}(\eta||\nu_k||^3 + \eta\mathscr{C}).$$

We can write down an expression for r_k summing over all the terms:

$$r_{k+1} = \underbrace{-\frac{\eta}{2}\sum_{j\leq k}(I - \eta G)^{k-j}\nabla G[\nu_k, \nu_k]}_{\text{term (a)}} + \underbrace{\sum_{j\leq k}(I - \eta G)^j\left[\mathcal{O}(\eta||\nu_k||^3 + \eta\mathscr{C})\right]}_{\text{term (b)}}. \tag{C.22}$$

Term (a) in the above equation is the one that can induce a displacement in r_k as k increases and would be used to derive the regularizer, whereas term (b) primarily consists of terms that concentrate to 0. We first analyze term (a) and then we will analyze the concentration terms later.

To analyze term (a), note that the term $\nabla G[\nu_k, \nu_k]$, by Lemma C.1, only depends on ν_k via the covariance matrix $\nu_k\nu_k^\top$. So we will partition this term into two terms: **(i)** a term that utilizes the asymptotic covariance matrix of the OU process and **(ii)** errors due to a finite k and stochasticity that will concentrate.

$$2 \times \text{(a)} \; = \eta\sum_{j\leq k}(I - \eta G)^{k-j}\nabla G[\nu_k, \nu_k] \tag{C.23}$$

$$= \sum_{j\leq k}(I - \eta G)^{k-j}\nabla G[\zeta^*, \zeta^*] + \sum_{j\leq k}(I - \eta G)^{k-j}\nabla G([\nu_k, \nu_k] - [\zeta^*, \zeta^*]), \tag{C.24}$$

The first term is a "bias" term and doesn't concentrate to 0, and will give rise to the regularizer. We can break this term using Lemma C.1 as:

$$\nabla G[\zeta^*, \zeta^*] = 2\sum_i \nabla^2 Q_i \Sigma_M^*(\nabla Q_i - \gamma\nabla Q_i') + \sum_i \text{tr}\left[(\nabla^2 Q_i - \gamma\nabla^2 Q_i')\Sigma_M^*\right]\nabla Q_i \tag{C.25}$$

The regularizer $R_{\text{TD}}(\theta)$ is the function such that:

$$\nabla_\theta R_{\text{TD}}(\theta) = \sum_i \nabla^2 Q_i \Sigma_M^*(\nabla Q_i - \gamma\nabla Q_i') \tag{C.26}$$

$$\implies R_{\text{TD}}(\theta) = \sum_i \nabla Q_i \Sigma_M^* \nabla Q_i^\top - \gamma\sum_i \text{trace}\left(\Sigma_M^*\nabla Q_i[[\nabla Q_i']]^\top\right), \tag{C.27}$$

where $[[\cdot]]$ denotes the stop gradient operator. If the point θ^* is a stationary point of the regularizer $R_{\text{TD}}(\theta)$, then Equations C.26 and C.27 imply that the first term of Equation C.25 must be 0. Therefore in this case to show that θ^* is attractive, we need to show that the other terms in Equations C.25, C.24 and term (b) in Equation C.22 concentrate around 0 and are bounded in magnitude. The remaining part of the proof shown in Appendix C.7 provides these details, but we first summarize the main takeaways in the proof to conclude the argument.

C.6 SUMMARY OF THE ARGUMENT

We will show how to concentrate terms in Equation C.26 besides the regularizer largely following the techniques from prior work, but we first summarize the entire proof. The overall update to the vector r_k which measures the displacement between the parameter vector $\theta_k - \theta^*$ and the OU-process ζ_k can be written as follows, and it is governed by the derivative of the implicit regularizer (modulo error terms):

$$r_{k+1} = -\frac{\eta}{2}\sum_{j\leq k}(I-\eta G)^{k-j}\nabla_\theta R_{\text{TD}}(\theta^*) + \mathcal{O}\left(\sqrt{\eta t}\cdot \text{poly}(\mathscr{C}, \mathscr{L}_2, \mathscr{L}_3, \omega, C_0)\right). \quad \text{(C.28)}$$

An important detail to note here is that since the regularizer consists of Σ_M^* and the size of Σ_M^* (i.e, its eigenvalues), as shown in Lemma C.4 depends on one factor of η. So, effectively the first term in Equation C.28 does depend on two factors of η. Using Equation C.28, we can write the deviation between θ^* and θ_k as:

$$\nu_{k+1} = \zeta_{k+1} - \frac{\eta}{2}\sum_{j\leq k}(I-\eta G)^{k-j}\nabla_\theta R_{\text{TD}}(\theta^*) + \mathcal{O}\left(\sqrt{\eta t}\cdot \text{poly}(\mathscr{C}, \mathscr{L}_2, \mathscr{L}_3, \omega, C_0)\right). \quad \text{(C.29)}$$

The OU process ζ_k converges to θ^* in the subspace spanned by G, since the condition $\rho(I-\eta G) < 1$ is active in this subspace (if the condition that $\rho(I-\eta G) < 1$ in the subspace spanned by G is not true, then as Ghosh & Bellemare (2020) show, TD can diverge). Now, given G satisfies this spectral radius condition, ζ_k would converge to θ^* within a timescale of $\mathcal{O}\left(\frac{1}{\eta}\right)$ within this subspace, which as Blanc et al. (2020) put it is the strength of the "mean-reversion" term. On the remaining directions (note that $d \gg n$), the dynamics is guided by the regularizer, although with a smaller weight of η^2.

C.7 ADDITIONAL PROOF DETAILS: CONCENTRATING OTHER TERMS

We first concentrate the terms in Equation C.25. The cumulative effect of the second term in Equation C.25 is given by:

$$\eta\sum_{j\leq k}(I-\eta G)^{j-k}\nabla Q_i \text{tr}\left[(\nabla^2 Q_i - \gamma\nabla^2 Q_i')\Sigma_M^*\right] \quad \text{(C.30)}$$

$$\leq \eta\sum_{j\leq k}(I-\eta G)^{j-k}\nabla Q_i\cdot\mathcal{O}\left(\mathscr{L}_2(1+\gamma)\sigma\right) \leq \mathcal{O}\left(\eta\sqrt{\frac{k}{\eta}}\omega_0 C_0\mathscr{L}_2(1+\gamma)\sigma\right), \quad \text{(C.31)}$$

which follows from the fact that $\nabla^2 Q_i$ is $\mathscr{L}_2$-Lipschitz, and using Lemma C.3 for contracting the remaining terms.

Next, we turn to concentrating the second term in Equation C.24. This term corresponds to the contribution of difference between the empirical covariance matrix $\nu_k\nu_k^\top$ and the asymptotic covariance matrix $\zeta^*\zeta^{*\top}$. We expand this term below using the form of G from Lemma C.1, and bound it one by one.

$$\sum_{j\leq k}(I-\eta G)^{k-j}\nabla G([\nu_k,\nu_k]-[\zeta^*,\zeta^*]) \quad \text{(C.32)}$$

$$= \sum_{j\leq k}\sum_i (I-\eta G)^{k-j}\nabla^2 Q_i\left(\nu_k\nu_k - \zeta^*\zeta^{*\top}\right)(\nabla Q_i - \gamma\nabla Q_i') + \mathcal{O}\left(\sqrt{\eta k}\omega_0 C_0\mathscr{L}_2(1+\gamma)\sigma\right) \quad \text{(C.33)}$$

Now, we note that the term $\Delta_{k+1} := \nu_{k+1}\nu_{k+1}^\top - \zeta^*\zeta^{*\top}$ can itself be written as a recursion:

$$\Delta_{k+1} = (I-\eta G)(\Delta_k)(I-\eta G)^\top + \underbrace{(I-\eta G)\zeta_k\varepsilon^\top + \varepsilon\zeta_k^\top(I-\eta G)^\top}_{A_k} + \underbrace{\varepsilon\varepsilon^\top - \eta M}_{B_k} \quad \text{(C.34)}$$

Expanding the term Δ_{k+1} in terms of a summation over k, and plugging it into the expression from Equation C.35 we get

$$\sum_i \sum_{j \le k} (I - \eta G)^{k-j} \nabla^2 Q_i (I - \eta G)^j \Delta_0 (I - \eta G^\top)^j \tag{C.35}$$
$$+ \sum_i \sum_{j \le k} \sum_{p \le j} (I - \eta G)^{k-j} \nabla^2 Q_i (I - \eta G)^{j-p-1} (A_p + B_p) (I - \eta G^\top)^{j-p-1}$$

Now by noting that if G and ∇Q_i are (ω, C_0)-aligned, then so are $G^\top$ and ∇Q_i, we can finish the proof by repeating the calculations used by Damian et al. (2021) (Appendix B, Equations 67-73) to bound the terms in Equation C.35 by $\mathcal{O}(\sqrt{\eta k})$, but with an additional factor of $\omega^2 C_0^2$.

Term (b) in Equation C.22. When $\mathscr{C}$ is small enough, we can bound the term (b) using $\mathcal{O}(\sqrt{\eta k})$, similar to Damian et al. (2021).

D Proof of Proposition 3.2

In this section, we will prove Proposition 3.2. First, we refer to Proposition 3.1 in Ghosh & Bellemare (2020), which shows that TD-learning is stable and converges if and only if the matrix $M_\phi = \Phi^\top (\Phi - \gamma \Phi')$ has eigenvalues with all positive real entries. Now note that if,

$$\sum_{\mathbf{s},\mathbf{a}} \phi(\mathbf{s},\mathbf{a})^\top \phi(\mathbf{s},\mathbf{a}) \le \gamma \sum_{\mathbf{s},\mathbf{a},\mathbf{s}'} \phi(\mathbf{s}',\mathbf{a}')^\top \phi(\mathbf{s},\mathbf{a}) \tag{D.1}$$
$$\implies \text{trace}\left(\Phi^\top \Phi\right) \le \gamma \text{trace}\left(\Phi^\top \Phi'\right) \tag{D.2}$$
$$\implies \text{trace}\left[\Phi^\top \left(\Phi - \gamma \Phi'\right)\right] \le 0. \tag{D.3}$$

Since the trace of a real matrix is the sum of real components of eigenvalues, if for a given matrix M, $\text{trace}(M) \le 0$, then there exists atleast one eigenvalue λ_i such that $\text{Re}(\lambda_i) \le 0$. If $\lambda_i < 0$, then the learning dynamics of TD would diverge, while if $\lambda_i = 0$ for all i, then learning will not contract towards the TD fixed point. This concludes the proof of this result.

E Experimental Details of Applying DR3

In this section, we discuss the practical experimental details and hyperparameters in applying our method, DR3 to various offline RL methods. We first discuss an overview of the offline RL methods we considered in this paper, and then provide a discussion of hyperparameters for DR3.

E.1 Background on Various Offline RL Algorithms

In this paper, we consider four base offline RL algorithms that we apply DR3 on. These methods are detailed below:

REM. Random ensemble mixture (Agarwal et al., 2020) is an uncertainty-based offline RL algorithm uses multiple parameterized Q-functions to estimate the Q-values. During the Bellman backup, REM computes a random convex combination of the target Q-values and then trains the Q-function to match this randomized target estimate. The randomized target value estimate provides a robust estimate of target values, and delays unlearning and performance degradation that we typically see with standard DQN-style algorithms in the offline setting. For instantiating REM, we follow the instantiation provided by the authors and instantiate a multi-headed Q-function with 200 heads, each of which serves as an estimate of the target value. These multiple heads branch off the last-but-one layer features of the base Q-network. The objective for REM is given by:

$$\min_\theta \mathbb{E}_{\mathbf{s},\mathbf{a},r,\mathbf{s}' \sim \mathcal{D}} \left[\mathbb{E}_{\alpha_1,\ldots,\alpha_K \sim \Delta} \left[\ell_\lambda \left(\sum_k \alpha_k Q^k_\theta(\mathbf{s},\mathbf{a}) - r - \gamma \max_{\mathbf{a}'} \sum_k \alpha_k Q^k_{\theta'}(\mathbf{s}',\mathbf{a}') \right) \right] \right] \tag{E.1}$$

where l_λ denotes the Huber loss while P_Δ denotes the probability distribution over the standard (K 1)-simplex.

CQL. Conservative Q-learning (Kumar et al., 2020b) is an offline RL algorithm that learns a conservative value function such that the estimated performance of the policy under this learned value function lower-bounds its true value. CQL modifies the Q-function training to incorporate a term that minimizes the overestimated Q-values in expectation, while maximizing the Q-values observed in the dataset, in addition to standard TD error. This CQL regularizer is typically multiplied by a coefficient α, and we pick $\alpha = 0.1$ for all our Atari experiments following Kumar et al. (2021) and $\alpha = 5.0$ for all our kitchen and antmaze D4RL experiments. Using $\overline{y}_k(\mathbf{s}, \mathbf{a})$ to denote the target values computed via the Bellman backup (we use actor-critic backup for D4RL experiments and the $\max_{\mathbf{a}'}$ backup for standard Q-learning in our Atari experiments following Kumar et al. (2020b)), the objective for training CQL is given by:

$$\min_Q \alpha \left(\mathbb{E}_{\mathbf{s} \sim \mathcal{D}} \left[\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s}, \mathbf{a})) \right] - \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}} \left[Q(\mathbf{s}, \mathbf{a}) \right] \right) + \frac{1}{2} \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim \mathcal{D}} \left[\left(Q(\mathbf{s}, \mathbf{a}) - \overline{y}_k(\mathbf{s}, \mathbf{a}) \right)^2 \right].$$

The deep Q-network utilized by us is a ReLU network with four hidden layers of size $(256, 256, 256, 256)$ for the D4RL experiments, while for Atari we utilize the standard convolutional neural network from Agarwal et al. (2020); Kumar et al. (2021) with 3 convolutional layers borrowed from the nature DQN network and then a hidden feedforward layer of size 512.

BRAC. Behavior-regularized actor-critic (Wu et al., 2019) is a policy-constraint based actor-critic offline RL algorithm which regularizes the policy to stay close to the behavior policy π_β to prevent the selection of "out-of-distribution" actions. In addition, BRAC subtracts this divergence estimate from the target Q-values when performing the backup, to specifically penalize target values that come from out-of-distribution action inputs at the next state $(\mathbf{s}', \mathbf{a}')$.

$$\begin{aligned} \text{Q-function:} \quad & \min_\theta \; \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}} \left[\left(r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_\phi(\cdot|\mathbf{s}')} [\bar{Q}_\theta(\mathbf{s}', \mathbf{a}') + \beta \log \hat{\pi}_\beta(\mathbf{a}'|\mathbf{s}')] - Q_\theta(\mathbf{s}, \mathbf{a}) \right)^2 \right]. \\ \text{Policy:} \quad & \max_\phi \; \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_\phi(\cdot|\mathbf{s})} \left[Q_\theta(\mathbf{s}, \mathbf{a}) + \beta \log \hat{\pi}_\beta(\mathbf{a}|\mathbf{s}) - \alpha \log \pi_\phi(\mathbf{a}|\mathbf{s}) \right]. \end{aligned} \tag{E.2}$$

COG. COG (Singh et al., 2020) is an algorithmic framework for utilizing large, unlabeled datasets of diverse behavior to learn generalizable policies via offline RL. Similar to real-world scenarios where large unlabeled datasets are available alongside limited task-specific data, the agent is provided with two types of datasets. The task-specific dataset consists of behavior relevant for the task, but the prior dataset can consist of a number of random or scripted behaviors being executed in the same environment/setting. The goal in this task is to actually stitch together relevant and overlapping parts of different trajectories to obtain a good policy that can work from a new initial condition that was not seen in a trajectory that actually achieved the reward. COG utilizes CQL as the base offline RL algorithm, and following Singh et al. (2020), we fix the hyperparameter $\alpha = 1.0$ in the CQL part for both base COG and COG + DR3. All other hyperparameters including network sizes, etc are kept fixed as the prior work Singh et al. (2020) as well.

E.2 Tasks and Environments Used

Atari 2600 games used. For all our experiments, we used the same set of 17 games utilized by Kumar et al. (2021) to test rank collapse. In the case of Atari, we used the 5 standard games (Asterix, Qbert, Pong, Seaquest, Breakout) for tuning the hyperparameters, a strategy followed by several prior works (Gulcehre et al., 2020; Agarwal et al., 2020; Kumar et al., 2021). The 17 games we test on are: Asterix, Qbert, Pong, Seaquest, Breakout, Double Dunk, James Bond, Ms. Pacman, Space Invaders, Zaxxon, Wizard of Wor, Yars' Revenge, Enduro, Road Runner, BeamRider, Demon Attack, Ice Hockey.

Following Agarwal et al. (2021), we report interquartile mean (IQM) normalized scores across all runs as mean scores can be dominated by performance on a few outlier tasks while median is independent of performance on all except 1 task – zero score on half of the tasks would not affect the median. IQM which corresponds to 25% trimmed mean and considers the performance on middle 50% of the runs. IQM interpolates between mean and median, which correspond to 0% and almost 50% trimmed means across runs.

D4RL tasks used. For our experiments on D4RL, we utilize the Gym-MuJoCo-v0 environments for evaluating BRAC, since BRAC performed somewhat reasonably on these domains (Fu et al., 2020), whereas we use the harder AntMaze and Franka Kitchen domains for evaluating CQL, since these domains are challenging for CQL (Kumar et al., 2020b).

Table E.1: **Hyperparameters used by the offline RL Atari agents in our experiments.** Following Agarwal et al. (2020), the Atari environments used by us are stochastic due to sticky actions, *i.e.*, there is a 25% chance at every time step that the environment will execute the agents previous action again, instead of the new action commanded. We report offline training results with same hyperparameters over 5 random seeds of the offline dataset, game simulator and network initialization.

Hyperparameter	Setting (for both variations)
Sticky actions	Yes
Sticky action probability	0.25
Grey-scaling	True
Observation down-sampling	(84, 84)
Frames stacked	4
Frame skip (Action repetitions)	4
Reward clipping	[-1, 1]
Terminal condition	Game Over
Max frames per episode	108K
Discount factor	0.99
Mini-batch size	32
Target network update period	every 2000 updates
Training environment steps per iteration	250K
Update period every	4 environment steps
Evaluation ϵ	0.001
Evaluation steps per iteration	125K
Q-network: channels	32, 64, 64
Q-network: filter size	8×8, 4×4, 3×3
Q-network: stride	4, 2, 1
Q-network: hidden units	512

Robotic manipulation tasks from COG (Singh et al., 2020). These tasks consist of a 6-DoF WidowX robot, placed in front of two drawers and a larger variety of objects. The robot can open or close a drawer, grasp objects from inside the drawer or on the table, and place them anywhere in the scene. The task here consists of taking an object out of a drawer. A reward of +1 is obtained when the object has been taken out, and zero otherwise. There are two variants of this domain: **(1)** in the first variant, the drawer starts out closed, the top drawer starts out open (which blocks the handle for the lower drawer), and an object starts out in front of the closed drawer, which must be moved out of the way before opening, and **(2)** in the second variant, the drawer is blocked by an object, and this object must be removed before the drawer can be opened and the target object can be grasped from the drawer. The prior data for this environment is collected from a collection of scripted randomized policies. These policies are capable of opening and closing both drawers with 40-50% success rates, can grasp objects in the scene with about a 70% success rate, and place those objects at random places in the scene (with a slight bias for putting them in the tray).

E.3 The DR3 Regularizer Coefficient

We utilize identical hyperparameters of the base offline RL algorithms when DR3 is used, where the base hyper-parameters correspond to the ones provided in the corresponding publications. DR3 requires us to tune the additional coefficient c_0, that weights the DR3 explicit regularizer term. In order to find this value on our domains, we followed the tuning strategy typically followed on Atari, where we evaluated four different values of $c_0 \in \{0.001, 0.01, 0.03, 0.3\}$ on 5 games (Asterix, Seaquest, Breakout, Pong and SpaceInvaders) on the 5% replay dataset settings, picked c_0 that wprked best on just these domains, and used it to report performance on all 17 games, across all dataset settings (1% replay and 10% initial replay) in Section 6. This protocol is standard in Atari and has been used previously in Agarwal et al. (2020); Gulcehre et al. (2020); Kumar et al. (2021) in the context of offline RL. The value of the coefficient found using this strategy was $c_0 = 0.001$ for REM and $c_0 = 0.03$ for CQL.

For CQL on D4RL, we ran DR3 with multiple values of $c_0 \in \{0.0001, 0.001, 0.01, 0.5, 1.0, 10.0\}$, and picked the smallest value of c_0 which did not lead to eventually divergent (either negatively diverging or positively diverging) Q-values, in average. For the antmaze domains, this corresponded to $c_0 = 0.001$ and for the FrankaKitchen domains, this corresponded to $c_0 = 1.0$.

F Complete Results on All Domains

In this section, we present the results obtained by running DR3 on the Atari and D4RL domains which were not discussed in the main paper due to lack of space. We first understand the effect of applying DR3 on BRAC (Wu et al., 2019), which was missing from the main paper, and then present the per-game Atari results.

Table F.1: Normalized interquartile mean (IQM) final performance (last iteration return) of CQL, CQL + DR3, REM and REM + DR3 after 6.5M gradient steps for the 1% setting and 12.5M gradient steps for the 5%, 10% settings. Intervals in brackets show 95% CIs computed using stratified percentile bootstrap (Agarwal et al., 2021)

Data	CQL	CQL + DR3	REM	REM + DR3
1%	44.4 (31.0, 54.3)	**61.6** (39.1, 71.5)	0.0 (-0.7, 0.1)	**13.1** (9.9, 18.3)
5%	89.6 (67.9, 98.1)	**100.2** (90.6, 102.7)	3.9 (3.1, 7.6)	**74.8** (59.6, 84.4)
10%	57.4 (53.2, 62.4)	**67.0** (62.8, 73.0)	24.9 (15.0, 29.1)	**72.4** (65.7, 81.7)

Table F.2: **Performance of DR3 when applied in conjunction with BRAC (Wu et al., 2019).** Note that DR3 attains a larger final performance (at the end of 2M steps of training) as well as a higher average performance (i.e. stability score) across all iterations of training.

Task	Average Performance across Iterations		Final Performance	
	BRAC	BRAC + DR3	BRAC	BRAC + DR3
halfcheetah-expert-v0	1.7 ± 1.9	49.9 ± 16.7	2.1 ± 3.3	71.5 ± 24.9
halfcheetah-medium-v0	43.5 ± 0.2	43.2 ± 0.2	45.1 ± 0.8	44.9 ± 0.6
halfcheetah-medium-expert-v0	17.0 ± 5.4	6.0 ± 5.5	24.8 ± 9.3	6.7 ± 7.3
halfcheetah-random-v0	24.4 ± 0.4	18.4 ± 0.3	24.9 ± 0.8	18.2 ± 1.0
halfcheetah-medium-replay-v0	44.9 ± 0.3	44.1 ± 0.4	45.0 ± 1.4	44.9 ± 0.5
hopper-expert-v0	15.7 ± 1.5	21.8 ± 3.2	16.6 ± 6.0	20.8 ± 5.3
hopper-medium-v0	32.8 ± 1.4	46.3 ± 7.1	36.2 ± 1.7	58.3 ± 13.7
hopper-medium-expert-v0	40.2 ± 5.7	37.0 ± 2.9	31.7 ± 11.8	21.8 ± 4.9
hopper-random-v0	11.7 ± 0.0	11.2 ± 0.0	12.2 ± 0.0	11.1 ± 0.0
hopper-medium-replay-v0	31.6 ± 0.3	30.3 ± 0.8	31.3 ± 1.2	36.1 ± 5.7
walker2d-expert-v0	25.5 ± 14.4	33.6 ± 11.8	54.0 ± 31.0	60.6 ± 20.2
walker2d-medium-v0	81.3 ± 0.3	80.8 ± 0.2	83.8 ± 0.2	83.4 ± 0.3
walker2d-medium-expert-v0	5.8 ± 5.2	6.4 ± 3.4	22.4 ± 22.0	39.5 ± 23.3
walker2d-random-v0	1.4 ± 0.8	1.7 ± 0.9	0.0 ± 0.1	2.9 ± 2.1
walker2d-medium-replay-v0	26.1 ± 6.4	47.4 ± 4.1	11.7 ± 7.0	38.7 ± 9.6

Table F.4: **Mean evaluation returns per Atari game across 5 runs with standard deviations for 1% dataset**. The coefficient for DR3 is 0.03 with a CQL coefficient of 1.0. The average performance is computed over 20 checkpoints spaced uniformly over training for 100 iterations where 1 iteration corresponds to 62,500 gradient updates.

Game	Final Performance		Average Performance across Iterations	
	CQL	CQL + DR3	CQL	CQL + DR3
Asterix	656.9 ± 91.0	821.4 ± 75.1	650.2 ± 65.3	814.1 ± 25.1
Breakout	23.9 ± 3.8	32.0 ± 3.2	23.8 ± 0.5	32.8 ± 3.1
Pong	16.7 ± 1.7	14.2 ± 3.3	15.7 ± 2.0	15.1 ± 2.3
Seaquest	449.0 ± 11.0	446.6 ± 26.9	474.5 ± 30.3	456.1 ± 17.0
Qbert	8033.8 ± 1513.2	9162.7 ± 993.6	7980.0 ± 379.9	9000.7 ± 225.2
SpaceInvaders	386.0 ± 123.2	351.9 ± 77.1	371.7 ± 47.5	440.6 ± 29.6
Zaxxon	829.4 ± 813.3	1757.4 ± 879.4	834.6 ± 504.0	1634.0 ± 673.9
YarsRevenge	11848.2 ± 2977.7	16011.3 ± 1409.0	15077.9 ± 1301.9	17741.6 ± 613.6
RoadRunner	37000.7 ± 1148.5	24928.7 ± 7484.5	35899.9 ± 653.1	32063.3 ± 1011.4
MsPacman	1869.8 ± 167.2	2245.7 ± 193.8	1991.9 ± 55.1	2224.1 ± 80.8
BeamRider	780.3 ± 64.5	617.9 ± 25.1	782.0 ± 36.1	619.9 ± 20.9
Jamesbond	558.5 ± 124.8	460.5 ± 102.0	524.6 ± 118.5	484.2 ± 89.4
Enduro	198.4 ± 34.2	253.5 ± 14.2	259.8 ± 16.4	276.1 ± 16.9
WizardOfWor	771.1 ± 358.2	904.6 ± 343.7	833.7 ± 168.4	935.2 ± 174.4
IceHockey	-8.7 ± 1.3	-7.8 ± 0.9	-8.8 ± 0.9	-7.9 ± 0.7
DoubleDunk	-15.1 ± 1.9	-14.0 ± 2.8	-15.3 ± 0.9	-14.5 ± 1.0
DemonAttack	1970.2 ± 161.3	386.2 ± 75.3	1338.8 ± 298.4	414.0 ± 46.0

Table F.5: **Mean evaluation returns per Atari game across 5 runs with standard deviations for 5% dataset.** The coefficient for DR3 is 0.03 with a CQL coefficient of 0.1. The average performance is computed over 20 checkpoints spaced uniformly over training for 200 iterations where 1 iteration corresponds to 62,500 gradient updates.

Game	Final Performance		Average Performance across Iterations	
	CQL	CQL + DR3	CQL	CQL + DR3
Asterix	1798.2 ± 168.6	3318.5 ± 301.7	1812.7 ± 64.0	3790.5 ± 218.0
Breakout	94.1 ± 44.4	166.0 ± 23.1	105.1 ± 10.4	196.5 ± 4.4
Pong	13.1 ± 4.2	17.9 ± 1.1	15.2 ± 1.3	17.4 ± 1.2
Seaquest	1815.9 ± 722.8	2030.7 ± 822.8	1382.3 ± 258.1	3722.3 ± 969.5
Qbert	10595.7 ± 1648.5	9605.6 ± 1593.5	9552.0 ± 925.6	10830.7 ± 783.1
SpaceInvaders	758.9 ± 56.9	1214.6 ± 281.8	662.0 ± 58.1	1323.7 ± 94.4
Zaxxon	1501.0 ± 1165.7	4250.1 ± 626.2	1508.8 ± 437.5	3556.5 ± 531.3
YarsRevenge	24036.7 ± 3370.6	17124.7 ± 2125.6	22733.1 ± 1175.3	18339.8 ± 1299.7
RoadRunner	40728.4 ± 3318.9	38432.6 ± 1539.7	42338.4 ± 471.4	41260.2 ± 1008.6
MsPacman	2975.9 ± 522.1	2790.6 ± 353.1	2923.6 ± 251.3	3101.2 ± 381.6
BeamRider	1897.6 ± 473.7	785.8 ± 43.5	2218.5 ± 242.4	775.9 ± 12.5
Jamesbond	108.8 ± 49.1	96.8 ± 43.2	76.5 ± 4.6	106.1 ± 34.8
Enduro	764.3 ± 168.7	938.5 ± 63.9	797.7 ± 47.8	923.2 ± 40.3
WizardOfWor	943.2 ± 380.3	612.0 ± 343.3	1004.3 ± 314.7	1007.4 ± 313.2
IceHockey	-17.3 ± 0.6	-15.0 ± 0.7	-16.6 ± 0.5	-12.0 ± 0.3
DoubleDunk	-18.1 ± 1.5	-16.2 ± 1.7	-17.3 ± 1.0	-16.0 ± 1.6
DemonAttack	4055.8 ± 499.7	8517.4 ± 1065.9	4062.4 ± 465.8	8396.7 ± 689.4

Table F.6: **Mean returns per Atari game across 5 runs with standard deviations for initial 10% dataset.** The coefficient for DR3 is 0.03 with a CQL coefficient of 0.1. The average performance is computed over 20 checkpoints spaced uniformly over training for 200 iterations.

Game	Final Performance		Average Performance across Iterations	
	CQL	CQL + DR3	CQL	CQL + DR3
Asterix	2803.9 ± 294.6	3906.2 ± 521.3	2903.2 ± 217.7	4692.2 ± 377.0
Breakout	64.7 ± 7.3	70.8 ± 5.5	65.6 ± 5.7	75.4 ± 6.0
Pong	5.3 ± 6.8	5.5 ± 6.2	7.3 ± 5.0	8.1 ± 5.2
Seaquest	222.3 ± 219.5	1313.0 ± 220.0	704.9 ± 254.5	1327.9 ± 250.0
Qbert	4803.2 ± 489.5	5395.3 ± 1003.6	4492.5 ± 240.8	4708.5 ± 463.0
SpaceInvaders	704.9 ± 121.5	938.1 ± 80.3	737.8 ± 23.8	902.1 ± 60.0
Zaxxon	231.6 ± 450.9	836.8 ± 434.7	394.4 ± 385.1	725.7 ± 370.3
YarsRevenge	13076.2 ± 2427.0	12413.9 ± 2869.7	12493.2 ± 543.6	12395.6 ± 1044.2
RoadRunner	45063.5 ± 1749.7	45336.9 ± 1366.7	45522.7 ± 1068.1	44808.0 ± 911.7
MsPacman	2459.5 ± 381.3	2427.5 ± 191.3	2528.1 ± 149.2	2488.3 ± 109.8
BeamRider	4200.7 ± 470.2	3468.0 ± 238.0	4729.5 ± 94.8	3344.3 ± 289.0
Jamesbond	84.6 ± 25.4	89.7 ± 15.6	108.7 ± 34.1	111.7 ± 10.9
Enduro	946.7 ± 289.7	1160.2 ± 81.5	1013.9 ± 29.7	1136.2 ± 32.5
WizardOfWor	520.4 ± 451.2	764.7 ± 250.0	499.8 ± 238.5	792.2 ± 101.3
IceHockey	-18.1 ± 0.7	-16.0 ± 1.3	-17.6 ± 0.5	-15.2 ± 1.0
DoubleDunk	-21.2 ± 1.1	-20.6 ± 1.0	-20.6 ± 0.3	-19.7 ± 0.5
DemonAttack	4145.2 ± 400.6	7152.9 ± 723.2	4839.4 ± 586.7	7278.5 ± 701.3

Table F.7: **Mean returns per Atari game across 5 runs with standard deviations for 1% dataset.** The coefficient for DR3 is 0.001 while we use a multi-headed REM with 200 Q-heads (Agarwal et al., 2020). The average performance is computed over 20 checkpoints spaced uniformly over training for 100 iterations.

Game	Final Performance		Average Performance across Iterations	
	REM	REM + DR3	REM	REM + DR3
Asterix	240.4 ± 29.1	405.7 ± 46.5	304.4 ± 9.3	413.7 ± 39.6
Breakout	0.7 ± 0.7	14.3 ± 2.8	6.3 ± 1.0	10.3 ± 1.1
Pong	-14.2 ± 1.7	-7.7 ± 6.3	-14.1 ± 2.2	-15.3 ± 3.0
Seaquest	81.0 ± 78.5	293.3 ± 191.5	246.6 ± 49.5	489.9 ± 128.6
Qbert	239.6 ± 133.2	436.3 ± 111.5	255.5 ± 76.0	471.0 ± 116.5
SpaceInvaders	152.8 ± 27.5	206.6 ± 77.6	188.6 ± 5.8	262.7 ± 22.4
Zaxxon	534.9 ± 731.3	2596.4 ± 1726.4	1807.9 ± 478.2	707.7 ± 577.4
YarsRevenge	1452.6 ± 1631.0	5480.2 ± 962.3	4018.8 ± 987.8	7352.0 ± 574.7
RoadRunner	0.0 ± 0.0	3872.9 ± 1616.4	1601.2 ± 637.9	14231.9 ± 2406.0
MsPacman	698.8 ± 129.5	1275.1 ± 345.6	690.4 ± 69.7	860.4 ± 57.1
BeamRider	703.0 ± 97.4	522.9 ± 42.2	745.5 ± 30.7	592.2 ± 27.7
Jamesbond	41.0 ± 27.0	157.6 ± 65.0	53.3 ± 12.1	88.8 ± 27.2
Enduro	0.5 ± 0.4	132.4 ± 16.1	21.7 ± 4.0	197.5 ± 19.1
WizardOfWor	362.5 ± 321.8	1663.7 ± 417.8	552.1 ± 253.1	1460.8 ± 194.8
IceHockey	-16.7 ± 0.9	-9.1 ± 5.1	-12.1 ± 0.8	-4.8 ± 1.8
DoubleDunk	-21.8 ± 1.0	-17.6 ± 1.5	-20.4 ± 0.6	-17.1 ± 1.6
DemonAttack	102.0 ± 17.3	162.0 ± 34.7	124.0 ± 10.7	145.6 ± 27.2

Table F.8: **Mean returns per Atari game across 5 runs with standard deviations for the 5% dataset.** The coefficient for DR3 is 0.001 while we use a multi-headed REM with 200 Q-heads (Agarwal et al., 2020). The average performance is computed over 20 checkpoints spaced uniformly over training for 200 iterations.

Game	Final Performance		Average Performance across Iterations	
	REM	REM + DR3	REM	REM + DR3
Asterix	876.8 ± 201.1	2317.0 ± 838.1	958.9 ± 50.9	1252.6 ± 395.1
Breakout	15.2 ± 4.9	33.4 ± 4.0	16.3 ± 3.4	17.7 ± 2.4
Pong	7.5 ± 5.2	-0.7 ± 9.9	-4.7 ± 3.0	-12.0 ± 3.2
Seaquest	1276.0 ± 417.3	2753.6 ± 1119.7	1484.3 ± 367.7	1602.0 ± 603.7
Qbert	2421.4 ± 1841.8	7417.0 ± 2106.7	1330.7 ± 431.0	4045.8 ± 898.9
SpaceInvaders	431.5 ± 23.3	443.5 ± 67.4	349.5 ± 22.6	362.1 ± 33.6
Zaxxon	6738.2 ± 966.6	1609.7 ± 1814.1	3630.7 ± 751.4	346.1 ± 512.1
YarsRevenge	14454.2 ± 1644.4	16930.4 ± 2625.8	14628.3 ± 1945.1	12936.5 ± 1286.0
RoadRunner	15570.9 ± 12795.6	46601.6 ± 2617.2	22740.3 ± 1977.2	33554.1 ± 1880.4
MsPacman	1272.2 ± 215.3	2303.1 ± 202.7	1147.7 ± 126.1	1438.7 ± 140.4
BeamRider	1922.5 ± 589.1	674.8 ± 21.4	886.9 ± 82.1	698.3 ± 21.5
Jamesbond	189.6 ± 77.0	130.5 ± 45.7	120.2 ± 9.3	88.6 ± 41.5
Enduro	172.7 ± 55.9	583.9 ± 108.7	236.8 ± 11.3	457.7 ± 39.3
WizardOfWor	838.4 ± 670.0	2661.6 ± 371.4	1281.3 ± 66.7	1863.7 ± 261.2
IceHockey	-9.7 ± 4.2	-6.5 ± 3.1	-8.1 ± 0.7	-4.1 ± 1.5
DoubleDunk	-18.4 ± 0.9	-17.6 ± 2.6	-19.6 ± 1.0	-17.8 ± 1.9
DemonAttack	507.7 ± 120.1	5602.3 ± 1855.5	581.6 ± 207.0	1452.3 ± 765.0

Table F.9: **Mean returns per Atari game across 5 runs with standard deviations for initial 10% dataset.** The coefficient for DR3 is 0.001 while we use a multi-headed REM with 200 Q-heads (Agarwal et al., 2020). The average performance is computed over 20 checkpoints spaced uniformly over training for 200 iterations.

Game	Final Performance		Average Performance across Iterations	
	REM	REM + DR3	REM	REM + DR3
Asterix	2254.7 ± 403.6	5122.9 ± 328.9	2684.6 ± 184.4	3432.1 ± 257.5
Breakout	81.2 ± 13.9	96.8 ± 21.2	63.5 ± 4.6	62.4 ± 6.1
Pong	8.8 ± 3.1	7.6 ± 11.1	2.6 ± 2.1	-2.5 ± 5.6
Seaquest	1540.2 ± 354.6	981.3 ± 605.9	1029.5 ± 260.6	836.2 ± 234.3
Qbert	4330.7 ± 250.2	4126.2 ± 495.7	3478.0 ± 248.0	3494.7 ± 380.3
SpaceInvaders	895.2 ± 68.3	799.0 ± 28.3	699.7 ± 31.4	653.1 ± 21.5
Zaxxon	950.7 ± 897.4	0.0 ± 0.0	490.2 ± 306.6	0.0 ± 0.0
YarsRevenge	10913.1 ± 1519.1	11924.8 ± 2413.8	11508.5 ± 290.0	10977.7 ± 1026.9
RoadRunner	45521.7 ± 2502.1	49129.4 ± 1887.9	37997.4 ± 638.6	41995.2 ± 1482.1
MsPacman	2177.4 ± 393.0	2268.8 ± 455.0	1930.5 ± 141.7	2126.6 ± 147.6
BeamRider	2921.7 ± 308.7	4154.9 ± 357.2	3727.5 ± 304.3	2871.0 ± 44.3
Jamesbond	197.8 ± 73.8	149.3 ± 304.5	149.0 ± 120.5	83.3 ± 162.4
Enduro	529.5 ± 200.7	832.5 ± 65.5	584.6 ± 85.3	801.8 ± 39.3
WizardOfWor	606.5 ± 823.2	920.0 ± 497.0	838.3 ± 343.7	926.3 ± 318.5
IceHockey	-4.3 ± 0.6	-5.9 ± 5.1	-7.0 ± 1.1	-5.4 ± 3.7
DoubleDunk	-17.7 ± 3.9	-19.5 ± 2.5	-16.9 ± 0.5	-16.7 ± 1.0
DemonAttack	6097.9 ± 1251.3	9674.7 ± 1600.6	4649.1 ± 514.6	5141.9 ± 361.4

Table F.10: Average returns across 5 runs for the random agent and the average performance of the trajectories in the DQN (Nature) dataset. For Atari normalized scores reported in the paper, the random agent is assigned a score of 0 while the average DQN replay is assigned a score of 100. Note that the random agent scores are also evaluated on Atari 2600 games with sticky actions.

Game	Random	Average DQN-Replay
Asterix	279.1	3185.2
Breakout	1.3	104.9
Pong	-20.3	14.5
Seaquest	81.8	1597.4
Qbert	155.0	8249.7
SpaceInvaders	149.5	1529.8
Zaxxon	10.6	1854.1
YarsRevenge	3147.7	21015.0
RoadRunner	15.5	38352.3
MsPacman	248.0	3108.8
BeamRider	362.0	4576.4
Jamesbond	27.6	560.3
Enduro	0.0	671.9
WizardOfWor	686.6	1128.5
IceHockey	-9.8	-8.5
DoubleDunk	-18.4	-11.3
DemonAttack	166.0	4407.5