Robust Hopfield Decision Transformer

Andreia Podasca
Department of Electrical and Computer Engineering
Drexel University
Philadelphia, PA 19104
andreia.podasca@drexel.edu

Anup Das*
Department of Electrical and Computer Engineering
Drexel University
Philadelphia, PA 19104
anup.das@drexel.edu

Abstract

Decision Transformer (DT) formulates offline reinforcement learning as conditional sequence modeling, predicting actions by attending over past states, actions, and returns-to-go. However, the softmax attention in DT computes each context representation in a single step and has no explicit mechanism to limit the effect of observations corrupted by sensor noise or measurement errors, which are common in real-world deployment. Prior methods improve the robustness of DT by changing how it is trained, for example by reweighting the training loss or relabeling returns-to-go with worst-case estimates, but leave the attention mechanism itself unchanged. We introduce Robust Hopfield Decision Transformer (RHDT), which replaces softmax attention with modern Hopfield layers that update each query toward stored patterns acting as attractors, with each update decreasing a log-sum-exp energy function. The stored patterns are the attention keys of the context window, and overlapping attractor basins can cause interference between them. We therefore regularize training with an orthogonality penalty on the keys and a gradient penalty that limits the sensitivity of the predicted actions to the input states. On six D4RL locomotion tasks, RHDT matches or exceeds robust baselines in average return on clean observations. Under Gaussian observation noise scaled to 30% of the dataset standard deviation in each dimension, RHDT achieves the highest average and worst-10% (CVaR-10) normalized returns on all six tasks. At this noise level, it improves the task-averaged return by 14% and the task-averaged CVaR-10 by 62% over the stronger baseline for each metric, and its task-averaged return varies less across training seeds than those of the evaluated baselines. Code is available at https://github.com/drexel-DISCO/rhdt-public.

1 Introduction

Offline reinforcement learning (RL) learns policies from fixed datasets without environment interaction, enabling deployment in domains where online exploration is costly or dangerous [Levine et al., 2020]. Decision Transformer (DT) [Chen et al., 2021] casts this problem as conditional sequence modeling, treating RL as supervised learning on trajectories and achieving competitive performance with Q-learning methods on standard benchmarks [Fujimoto et al., 2019, Kumar et al., 2020, Kostrikov et al., 2021]. However, deploying offline-trained policies in real-world environments

*https://www.anupkdas.com.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

exposes them to observation corruption from sensor noise, measurement errors, missing sensor readings, and outlier measurements [Paster et al., 2022, Zhang et al., 2020, Shen et al., 2020, Zhang et al., 2022]. The softmax attention mechanism in DT computes outputs in a single step [Vaswani et al., 2017], offering no explicit mechanism to correct errors. Therefore, small corruptions can shift attention weights, leading to erroneous actions that compound over the episode.

To address this, Robust Decision Transformer (RDT) [Xu et al., 2025] regularizes training using embedding dropout and a Gaussian-weighted loss to improve average performance under corruption. Yet focusing on average return obscures a vulnerability: we find that while mean performance degrades by up to 27%, tail performance (CVaR-10: worst 10% of episodes) can drop by up to 65%, more than twice the decline. For reliable deployment, these worst-case episodes matter most. Adversarially Robust Decision Transformer (ARDT) [Tang et al., 2024] takes a different approach, optimizing for worst-case returns through conservative return relabeling.[2] Yet like RDT, it retains softmax attention, which offers no explicit mechanism to correct errors at test time.

We take an architectural approach to robustness, grounded in associative memory. Modern Hopfield networks (MHNs) [Ramsauer et al., 2020] generalize classical associative memory [Hopfield, 1982] to continuous states with exponentially large storage capacity, where queries iteratively converge to fixed points near stored patterns via energy minimization. These attractor dynamics provide a natural mechanism for error correction: corrupted inputs are drawn toward stable attractors. However, the robustness of retrieval depends on the radius of each attractor basin; corruptions exceeding this radius may converge to incorrect attractors or to metastable states that mix several patterns.

To address these challenges, we propose Robust Hopfield Decision Transformer (RHDT), which replaces softmax attention with multi-head modern Hopfield layers as illustrated in Figure 1. First, unlike most prior MHN applications, which use bidirectional attention [Ramsauer et al., 2020, Widrich et al., 2020, Hoover et al., 2023, Podasca and Das, 2026b], we introduce causal masking to enable autoregressive action prediction from historical states, actions, and returns-to-go. Second, we employ multiple attention heads to provide robustness by combining several independent retrievals. Third, we apply gradient penalty regularization to reduce output sensitivity to observation corruptions. Fourth, we encourage attention key orthogonality to increase pattern separation, expanding attractor basins and helping corrupted queries move toward correct attractors.

Empirically, we evaluate RHDT on D4RL locomotion tasks [Fu et al., 2020] under varying levels of observation corruption. At the highest evaluated corruption level, where the noise standard deviation is 30% of the dataset standard deviation, RHDT outperforms RDT and ARDT on CVaR-10 across all tasks, with improvements up to $+13.9$ points over RDT and $+14.8$ points over ARDT (62% higher than the stronger baseline when averaged over tasks). On mean return, RHDT matches or exceeds the baselines on clean observations and outperforms them on all tasks at this level, with gains up to $+11.7$ points over RDT and $+19.6$ points over ARDT (14% higher than the stronger baseline when averaged over tasks). RHDT maintains lower variance than baselines in most settings. Ablation studies validate the contribution of each architectural component to robustness.

2 Related Works

Transformers for Offline RL. Decision Transformer (DT) [Chen et al., 2021] recasts offline RL as conditional sequence modeling using a GPT-style architecture [Radford et al., 2018], inspiring extensions in planning [Janner et al., 2021], online finetuning [Zheng et al., 2022], trajectory stitching [Wu et al., 2023], stochasticity [Paster et al., 2022], goal conditioning [Furuta et al., 2021], value guidance [Hu et al., 2024], and token mixing [Kim et al., 2023]. However, these methods rely on standard token mixing, in most cases softmax attention, and provide no explicit recovery mechanism under observation corruption.

Robustness in Decision Transformers. RDT [Xu et al., 2025] improves robustness through training-time regularization, while ARDT [Tang et al., 2024] uses expectile regression to estimate minimax returns-to-go for worst-case adversaries. However, both operate entirely at the training level, leaving attention as a single-step softmax operation vulnerable to corruption at deployment.

[2] While our corruption model is stochastic rather than adversarial, we include ARDT as a baseline because, like ours, it focuses on tail risk and worst-case performance.

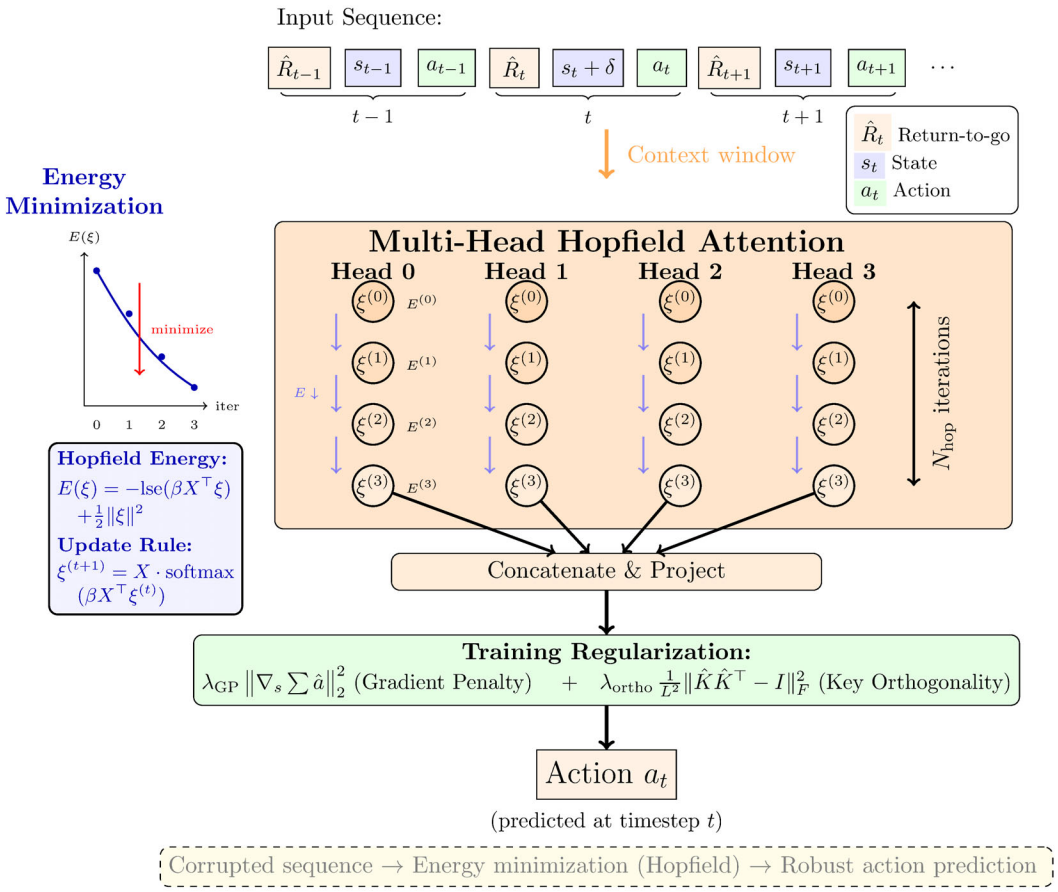

Figure 1: Robust Hopfield Decision Transformer (RHDT) architecture with energy minimization dynamics. The model processes sequences of $(\hat{R}_t, s_t, a_t)$ tuples across timesteps. Multi-head Hopfield attention performs N_{hop} iterative updates per head, each decreasing the Hopfield energy $E(\boldsymbol{\xi})$ (shown by darker node colors, perpendicular arrows indicating $E \downarrow$, and left-side energy plot). Each iteration refines queries toward stable attractors.

Our approach instead replaces softmax attention with Hopfield updates that move each query toward stable attractors by decreasing the energy (Figure 1).

Modern Hopfield Networks. Dense associative memory [Krotov and Hopfield, 2016] increases the storage capacity of classical Hopfield networks [Hopfield, 1982] by using higher-order interactions between neurons. Modern Hopfield networks (MHNs) [Ramsauer et al., 2020] extend it to continuous states with exponentially large storage capacity, where one iteration of the update rule corresponds to softmax attention and multiple iterations enable refinement toward stable attractors. MHNs have been applied to immune repertoire classification [Widrich et al., 2020], contrastive vision-language pretraining [Fürst et al., 2022], and time series prediction [Wu et al., 2024]. Modern Hopfield attention has also been used to make language models more robust, both to corrupted input embeddings [Podasca and Das, 2026b] and to adversarial attacks [Podasca and Das, 2026a]. To our knowledge, we are the first to use MHN attention in Decision Transformers, and we study how its attractor dynamics affect robustness to observation corruption.

3 Preliminaries

Offline Reinforcement Learning. We consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability $P(s'|s,a)$, reward function $R(s,a)$, and discount factor $\gamma \in [0,1)$. At each timestep t, an agent observes s_t, takes action a_t, receives reward $r_t = R(s_t, a_t)$,

and transitions to $s_{t+1} \sim P(\cdot|s_t, a_t)$. In offline RL, we learn a policy π from a fixed dataset $\mathcal{D} = \{\tau_i\}$ of trajectories from behavior policy π_β to maximize $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ without environment interaction.

Decision Transformer. Decision Transformer (DT) frames offline RL as conditional sequence modeling [Chen et al., 2021], processing trajectories as sequences of returns-to-go, states, and actions: $(\hat{R}_0, s_0, a_0, \hat{R}_1, s_1, a_1, \ldots)$, where $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ and T is the episode horizon. Each element is embedded via modality-specific linear layers with positional encodings, forming token embeddings $\mathbf{h}_i$ stacked into $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_L]^\top$. A GPT-style transformer [Radford et al., 2018] with causal masking processes $\mathbf{H}$ using self-attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V, \tag{1}$$

where $Q = \mathbf{H}W_Q$, $K = \mathbf{H}W_K$, $V = \mathbf{H}W_V$ are the query, key, and value projections, d_k is the key dimension, and the causal mask is omitted for brevity. This computes a weighted average over values in a single step. At test time, DT generates actions by conditioning on target return $\hat{R}_0$ and history.

Modern Hopfield Networks. An MHN [Ramsauer et al., 2020] defines an energy over a query $\boldsymbol{\xi}$ and stored patterns $\{\boldsymbol{x}_i\}_{i=1}^N$:

$$E(\boldsymbol{\xi}) = -\,\text{lse}(\beta, \boldsymbol{X}^\top \boldsymbol{\xi}) + \frac{1}{2}\boldsymbol{\xi}^\top \boldsymbol{\xi} + \frac{1}{\beta}\log N + \frac{1}{2}M^2, \tag{2}$$

where $\text{lse}(\beta, \boldsymbol{z}) = \beta^{-1}\log(\sum_i \exp(\beta z_i))$, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ is the pattern matrix, β is the inverse temperature, and $M = \max_i \|\boldsymbol{x}_i\|$. The update rule, which decreases this energy at every step, is:

$$\boldsymbol{\xi}^{(t+1)} = \boldsymbol{X}\,\text{softmax}(\beta \boldsymbol{X}^\top \boldsymbol{\xi}^{(t)}). \tag{3}$$

Iterating this update moves the query toward a fixed point that lies near a stored pattern when the patterns are well separated. A single iteration with $\beta = 1/\sqrt{d_k}$ recovers the attention weights in Eq. (1), with keys K as stored patterns $\boldsymbol{X}$ and queries Q as retrieval probes $\boldsymbol{\xi}$; in standard attention, these weights are applied to values V, whereas in the Hopfield formulation they weight the patterns $\boldsymbol{X}$ directly. Standard attention applies this update once, whereas iterative refinement applies it multiple times [Podasca and Das, 2026b], moving corrupted queries toward stable attractors. We use this refinement for robust decision-making.

4 Robust Hopfield Decision Transformer

We propose Robust Hopfield Decision Transformer (RHDT), replacing DT's softmax attention with Hopfield attention. We first describe this mechanism and analyze its sensitivity to observation perturbations. Next, we present gradient penalty and attention key orthogonality, two regularization techniques to reduce this sensitivity and expand attractor basins, respectively.

4.1 Hopfield Attention for Decision Making

Given input embeddings, we compute queries, keys, and values via linear projections, then apply N_{hop} iterations of the Hopfield update (Eq. 3), where the keys K serve as the stored patterns. After the N_{hop} updates move the query toward its fixed point, the final attention output is computed as $V\,\text{softmax}(\beta K^\top \boldsymbol{\xi}^{(N_{\text{hop}})})$, applying the refined attention weights to the value matrix V. To adapt MHNs for autoregressive decision-making in RL, we introduce two architectural modifications.

Causal Masking. Unlike most prior MHN applications [Ramsauer et al., 2020, Podasca and Das, 2026b] that use bidirectional attention, we apply causal masking so each position only attends to previous positions, ensuring actions are predicted without access to future states.

Multi-Head Attention. We employ multi-head Hopfield attention, where each head independently performs N_{hop} iterations with separate learned projections $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}$. Concatenated head outputs are linearly projected, which is equivalent to a learned weighted sum of the independent retrievals across heads and includes ensemble averaging as a special case; this can diversify error patterns and reduce sensitivity to individual perturbations. While these modifications enable MHNs for RL, perturbations exceeding the basin radius may still move queries toward incorrect attractors or toward metastable states that mix several patterns, which we analyze theoretically.

Theorem 1 (Hopfield Attention Sensitivity Bound)**.** *Let $\boldsymbol{\xi}^{(N_{hop})}$ denote the query state after N_{hop} Hopfield iterations with inverse temperature β and keys K (stored patterns), defined by the recurrence $\boldsymbol{\xi}^{(t+1)} = K\,\mathrm{softmax}(\beta K^\top \boldsymbol{\xi}^{(t)})$ with $\boldsymbol{\xi}^{(0)} = \boldsymbol{\xi}$. For a perturbation $\boldsymbol{\delta}$ to the initial query, the output perturbation is bounded by:*

$$\|\boldsymbol{\xi}^{(N_{hop})}(\boldsymbol{\xi}+\boldsymbol{\delta}) - \boldsymbol{\xi}^{(N_{hop})}(\boldsymbol{\xi})\|_2 \le (\beta\|K\|_2^2)^{N_{hop}}\|\boldsymbol{\delta}\|_2. \tag{4}$$

Proof. The softmax function with inverse temperature β satisfies $\|\,\mathrm{softmax}(\beta\boldsymbol{z}) - \mathrm{softmax}(\beta\boldsymbol{z}')\| \le \beta\|\boldsymbol{z}-\boldsymbol{z}'\|$. Each Hopfield iteration multiplies by $K^\top$, applies the softmax, and multiplies by K, yielding a Lipschitz constant of $\beta\|K\|_2^2$ per iteration. Over N_{hop} iterations, these constants multiply, giving the bound $(\beta\|K\|_2^2)^{N_{\text{hop}}}$ (full proof in Appendix A). □

Theorem 1 shows that this bound grows exponentially with N_{hop} when $\beta\|K\|_2^2 > 1$. We mitigate this by (1) using a small fixed N_{hop}, (2) introducing gradient penalty to reduce the sensitivity of the predicted actions to the input states, and (3) encouraging key orthogonality to expand attractor basins so that perturbations are more likely to remain within the basin.

4.2 Gradient Penalty Regularization

We penalize the squared gradient of the predicted actions with respect to the state observations:

$$\mathcal{L}_{\text{GP}} = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\frac{1}{C}\sum_{t=1}^{C}\left\|\frac{\partial}{\partial \boldsymbol{s}_t}\sum_{t'=1}^{C}\sum_{j=1}^{d_a}\hat{a}_{t',j}\right\|_2^2\right], \tag{5}$$

where $\hat{a}_{t',j}$ is the j-th dimension of the predicted action at step t', $\boldsymbol{s}_t \in \mathbb{R}^{d_s}$ is the state observation at step t, C is the context length, and the expectation is over trajectory segments from $\mathcal{D}$. Because of the causal mask, the gradient for $\boldsymbol{s}_t$ measures the sensitivity of the summed actions at steps $t' \ge t$, and computing the penalty requires one additional backward pass.

Proposition 1 (Gradient Penalty Controls Average Sensitivity)**.** *Let $\boldsymbol{\theta}^*$ minimize $\mathcal{L}_{action} + \lambda_{GP}\mathcal{L}_{GP}$ with $\mathcal{L}_{action} \ge 0$, and let ℓ_0 be the action loss of the best constant policy. Then $\mathcal{L}_{GP}(\boldsymbol{\theta}^*) \le \ell_0/\lambda_{GP}$, so the root-mean-square penalized gradient over the training data is $\mathcal{O}(1/\sqrt{\lambda_{GP}})$.*

Derivation. A constant policy has zero gradient penalty, so the total loss at $\boldsymbol{\theta}^*$ is at most ℓ_0, which gives $\lambda_{\text{GP}}\mathcal{L}_{\text{GP}}(\boldsymbol{\theta}^*) \le \ell_0$. Other bounded, non-negative loss terms only change the constant (full derivation in Appendix A).

The penalty is related to Lipschitz continuity as follows. Let J_t denote the Jacobian of all predicted actions in the context with respect to $\boldsymbol{s}_t$; the local Lipschitz constant of the policy at $\boldsymbol{s}_t$ is $\|J_t\|_2 \le \|J_t\|_F$. Our penalty evaluates $\|J_t^\top \mathbf{1}\|_2^2$, the Jacobian projected onto the all-ones vector, so it is small whenever the local Lipschitz constant is small but does not bound it. Replacing $\mathbf{1}$ with a random vector $\boldsymbol{v}$ satisfying $\mathbb{E}[\boldsymbol{v}\boldsymbol{v}^\top] = I$, such as a Rademacher vector, gives $\mathbb{E}_{\boldsymbol{v}}\|J_t^\top \boldsymbol{v}\|_2^2 = \|J_t\|_F^2$ [Hutchinson, 1989] at the same cost of one backward pass, a variant we leave for future work. With this variant, Proposition 1 bounds the mean squared local Lipschitz constant over the training data by $\ell_0/\lambda_{\text{GP}}$. A bound that holds for all inputs, including corrupted observations outside the data, would additionally require constraining the network itself, for example with spectral normalization [Miyato et al., 2018].

Gradient penalty complements key orthogonality (Section 4.3). Since keys are computed as $K = \mathbf{H}W_K$, the states affect the actions partly through the keys:

$$\frac{\partial \hat{\boldsymbol{a}}}{\partial \boldsymbol{s}} = \frac{\partial \hat{\boldsymbol{a}}}{\partial K}\cdot\frac{\partial K}{\partial \boldsymbol{s}} + \cdots \tag{6}$$

where $\cdot$ denotes tensor contraction (since K is a matrix, the partial derivatives are higher-order tensors). The penalty limits the total effect of the states on the actions, which includes this path through the stored patterns, although it does not bound $\|\partial K/\partial \boldsymbol{s}\|_F$ on its own. The two regularizers work in tandem: orthogonality expands attractor basins, while gradient penalty reduces how much the predicted actions change under corruption.

4.3 Key Orthogonality Regularization

Even with reduced sensitivity, robustness requires corrupted queries to move toward the correct attractor. This depends on attractor basin geometry, determined by separation between patterns.

Theorem 2 (Attractor Basin Radius)**.** *Consider an MHN with stored patterns $\{\boldsymbol{x}_i\}_{i=1}^N$ of equal norm, in the limit of large β. Every query within distance*

$$r_i = \frac{1}{2} \min_{j \neq i} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \tag{7}$$

of pattern $\boldsymbol{x}_i$ is mapped to $\boldsymbol{x}_i$ by the update in Eq. (3)*.*

Proof. As $\beta \to \infty$, the softmax in Eq. (3) becomes the argmax, so one update maps $\boldsymbol{\xi}$ to the pattern with the largest similarity $\boldsymbol{\xi}^\top \boldsymbol{x}_j$. For equal-norm patterns, $\boldsymbol{\xi}^\top \boldsymbol{x}_i > \boldsymbol{\xi}^\top \boldsymbol{x}_j$ holds if and only if $\|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2 < \|\boldsymbol{\xi} - \boldsymbol{x}_j\|_2$. If $\|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2 < r_i$, the triangle inequality gives $\|\boldsymbol{\xi} - \boldsymbol{x}_j\|_2 \geq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 - \|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2 > r_i$ for all $j \neq i$, so $\boldsymbol{x}_i$ is the nearest pattern (see Appendix A and [Ramsauer et al., 2020]). For finite β, the softmax mixes patterns, so the basin is smaller and the update can settle at metastable states. □

To enlarge basin radii, we encourage separation among the key vectors, which are the stored patterns ($\boldsymbol{x}_i = \boldsymbol{k}_i$). For unit-norm keys, $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = 2 - 2\cos\theta_{ij}$, where θ_{ij} is the angle between keys i and j, so the radius in Theorem 2 grows as the largest cosine similarity between keys decreases. We therefore penalize the cosine similarities between the key vectors of each head:

$$\mathcal{L}_{\text{ortho}} = \frac{1}{L^2} \|\hat{K}\hat{K}^\top - I\|_F^2, \tag{8}$$

where $\hat{K} \in \mathbb{R}^{L \times d_k}$ is the key matrix of a head with each row normalized to unit norm, and the loss is averaged over heads, layers, and batches. Exact orthogonality of the L key vectors is infeasible when $L > d_k$, as in our setting ($L = 3C = 60$, $d_k = 32$), so the penalty cannot reach zero; it instead reduces the pairwise cosine similarities and thereby increases the separation between keys. Our models use $\beta = 1/\sqrt{d_k}$, so Theorem 2 describes the geometry that the penalty targets rather than a guarantee. For finite β, Ramsauer et al. [2020] show that retrieval near $\boldsymbol{x}_i$ succeeds when the separation $\Delta_i = \boldsymbol{x}_i^\top \boldsymbol{x}_i - \max_{j \neq i} \boldsymbol{x}_i^\top \boldsymbol{x}_j$ exceeds a threshold that depends on β, the number of patterns, and the pattern norms. For unit-norm keys, $\Delta_i = 1 - \max_{j \neq i} \cos\theta_{ij}$, so the penalty directly increases the separation that this condition requires. The penalty also tightens the sensitivity bound of Theorem 1. Writing each key as its norm times a unit vector, Gershgorin's theorem gives $\|K\|_2^2 \leq \max_i \|\boldsymbol{k}_i\|_2^2 \, (1 + (L-1) \max_{i \neq j} |\cos\theta_{ij}|)$, so reducing the cosine similarities between keys lowers the per-iteration factor $\beta\|K\|_2^2$ for fixed key norms. In the ablation (Section 5.3), removing it lowers both metrics under corruption on both tasks, by at most 1.7 points.

4.4 Training Objective

The RHDT training objective combines action prediction with both regularization terms:

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \lambda_{\text{GP}} \mathcal{L}_{\text{GP}} + \lambda_{\text{ortho}} \mathcal{L}_{\text{ortho}}. \tag{9}$$

Following the Gaussian-weighted learning of RDT [Xu et al., 2025], $\mathcal{L}_{\text{action}}$ is a weighted mean squared error that down-weights poorly fit samples:

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{\tau \sim \mathcal{D}} \left[\frac{1}{C d_a} \sum_{t=1}^{C} w_t \|\hat{\boldsymbol{a}}_t - \boldsymbol{a}_t\|_2^2 \right], \quad w_t = \exp\left(-\operatorname{sg}\left(\|\hat{\boldsymbol{a}}_t - \boldsymbol{a}_t\|_2^2 / d_a \right) \right), \tag{10}$$

where $\operatorname{sg}(\cdot)$ denotes the stop-gradient operator.

5 Experiments

We evaluate RHDT on the D4RL benchmark [Fu et al., 2020] under observation corruption, comparing against Decision Transformer (DT) [Chen et al., 2021], Robust Decision Transformer (RDT) [Xu et al., 2025], and Adversarially Robust Decision Transformer (ARDT) [Tang et al., 2024]. DT is evaluated without corruption as a reference for clean performance.

5.1 Experimental Setup

Tasks and Datasets. We evaluate on six MuJoCo locomotion tasks from D4RL (v2 datasets): HalfCheetah, Hopper, and Walker2d, each with `medium` and `medium-replay` datasets. The `medium` datasets contain 1M transitions from policies trained to approximately one-third of expert performance, while `medium-replay` datasets use replay buffers collected during training. These datasets represent realistic offline RL scenarios with suboptimal, diverse trajectories.

Corruption Protocol. At test time, we inject Gaussian noise into state observations: $\tilde{s}_t = s_t + \delta$, where $\delta \sim \mathcal{N}(0, (\sigma \cdot \text{std}(s))^2)$ with $\sigma \in \{0.0, 0.1, 0.2, 0.3\}$, scaled element-wise by the standard deviation of each state dimension in the dataset to simulate realistic sensor noise. Corruption is applied independently at each timestep during evaluation, including the initial observation.

Evaluation Metrics. We report two primary metrics: (1) *Mean Score*: average normalized return measuring overall performance, where 0 represents a random policy and 100 represents expert performance on the D4RL scale; (2) *CVaR*-10: conditional value at risk, the average normalized return of the worst 10% of episodes, measuring tail performance important for reliable deployment.

Training and Evaluation Protocol. We train each method with 6 random seeds. For each trained model, we evaluate with 10 evaluation seeds and 50 episodes per seed, yielding 500 episodes per training seed. For each training seed, we pool its 500 episodes to compute the mean score and CVaR-10, and we report the average and standard deviation over training seeds. This protocol provides 3,000 evaluation episodes per task and method.

Implementation Details. All methods use a 3-layer transformer with embedding dimension 128, context length 20, and batch size 64, trained for 100K steps with AdamW (learning rate 10^{-4}, weight decay 10^{-4}, 10K-step linear warmup, gradient clipping at 0.25). The baselines use a single attention head, while RHDT uses $H = 4$ heads and $N_{\text{hop}} = 1$, i.e., one Hopfield update of each query before the readout. RHDT uses per-task regularization weights $\lambda_{\text{GP}} \in [0.0005, 0.02]$ and $\lambda_{\text{ortho}} \in [0.025, 0.2]$ (Table 5 in Appendix B with a sensitivity analysis in Appendix D). RDT uses per-task loss weights and embedding dropout, without iterative data correction since the training data are uncorrupted. Before its 100K training steps, ARDT relabels returns-to-go with a minimax return estimator trained for 5K warmup steps and 10K to 20K further steps. DT is trained with an unweighted action loss. Per-task baseline settings are listed in Appendix B.

Appendix C shows the training dynamics of RHDT, including the convergence of each loss term and its stability across training seeds.

5.2 Main Results

Table 1 compares RHDT with RDT and ARDT across corruption levels.

Tail performance. RHDT achieves the highest CVaR-10 in 22 of 24 task and corruption settings; the exceptions are Walker2d Med-Replay at $\sigma \leq 0.1$, where ARDT is at most 0.4 points higher. Its task-averaged CVaR-10 is 63% higher than RDT's at $\sigma = 0$, and at $\sigma = 0.3$ it is 64% higher than RDT's and 62% higher than ARDT's; the largest gains at $\sigma = 0.3$ are on Hopper Med-Replay over RDT ($+214\%$) and on HalfCheetah Medium over ARDT ($+151\%$).

Mean score. RHDT achieves the highest mean score on every task and corruption level, although without corruption its lead is below 0.5 points on HalfCheetah Medium and Walker2d Medium. At $\sigma = 0.3$, its task-averaged mean score is 14% higher than RDT's, the stronger baseline on this metric.

Variance. RHDT's results vary least across training seeds: it has the lowest or tied-lowest seed standard deviation in 17 of 24 task and corruption settings for both mean score and CVaR-10 (e.g., HalfCheetah Med-Replay at $\sigma = 0$: 37.4 ± 0.8 versus 26.2 ± 16.4 for RDT).

Increasing corruption. RHDT's task-averaged CVaR-10 gain over ARDT rises from $+33\%$ at $\sigma = 0$ to $+62\%$ at $\sigma = 0.3$, peaking at $+72\%$ at $\sigma = 0.2$. In contrast, ARDT's CVaR-10 relative to RDT changes sign with corruption (e.g., Walker2d Med-Replay: $+91\%$ at $\sigma = 0$ but -35% at

$\sigma = 0.3$), suggesting that worst-case return relabeling during training does not by itself provide robustness to observation corruption at test time.

Overall, RHDT improves both mean and tail performance under observation corruption.

Task	Method	DT	Mean Score				CVaR-10			
			Corruption Level (σ)				Corruption Level (σ)			
		0.0	0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
HalfCheetah Medium	RDT	42.2 ±0.1	42.0 ± 0.3	40.9 ± 0.6	37.9 ± 0.6	33.7 ± 0.4	36.2 ± 2.6	30.7 ± 5.7	22.6 ± 4.8	17.4 ± 1.7
	ARDT		41.7 ± 0.4	40.0 ± 0.4	36.1 ± 0.6	29.7 ± 1.5	33.5 ± 2.5	24.3 ± 2.7	14.4 ± 2.3	9.8 ± 2.8
	RHDT (ours)		**42.1 ± 0.4**	**41.6 ± 0.3**	**39.6 ± 0.4**	**35.9 ± 0.7**	**40.0 ± 0.4**	**38.2 ± 1.1**	**33.2 ± 1.4**	**24.6 ± 3.2**
	Δ_R / Δ_A		+0.1/+0.4	+0.7/+1.6	+1.7/+3.5	+2.2/+6.2	+3.8/+6.5	+7.5/+13.9	+10.6/+18.8	+7.2/+14.8
HalfCheetah Med-Replay	RDT	37.7 ±0.7	26.2 ± 16.4	27.5 ± 13.2	28.7 ± 8.0	27.8 ± 1.9	12.8 ± 13.9	11.0 ± 11.0	6.8 ± 4.0	4.5 ± 1.1
	ARDT		32.0 ± 3.0	30.1 ± 3.0	25.0 ± 3.7	19.2 ± 3.7	9.5 ± 4.4	8.1 ± 3.9	4.8 ± 1.7	3.2 ± 1.2
	RHDT (ours)		**37.4 ± 0.8**	**36.5 ± 0.9**	**33.8 ± 0.9**	**28.9 ± 1.5**	**32.2 ± 3.0**	**27.6 ± 4.6**	**17.6 ± 6.5**	**7.9 ± 3.1**
	Δ_R / Δ_A		+11.2/+5.4	+9.0/+6.4	+5.1/+8.8	+1.1/+9.7	+19.4/+22.7	+16.6/+19.5	+10.8/+12.8	+3.4/+4.7
Hopper Medium	RDT	62.3 ±10.1	61.9 ± 10.3	59.6 ± 7.7	52.9 ± 6.0	45.2 ± 4.7	35.4 ± 19.2	29.0 ± 8.9	21.2 ± 7.2	16.3 ± 4.2
	ARDT		58.1 ± 11.4	51.1 ± 3.9	47.9 ± 3.8	42.8 ± 5.4	46.1 ± 7.0	33.1 ± 9.0	23.7 ± 8.5	19.8 ± 7.1
	RHDT (ours)		**67.5 ± 5.2**	**64.0 ± 3.1**	**55.7 ± 3.1**	**50.5 ± 3.0**	**56.3 ± 4.7**	**43.9 ± 4.3**	**33.3 ± 3.1**	**26.7 ± 4.8**
	Δ_R / Δ_A		+5.6/+9.4	+4.4/+12.9	+2.8/+7.8	+5.3/+7.7	+20.9/+10.2	+14.9/+10.8	+12.1/+9.6	+10.4/+6.9
Hopper Med-Replay	RDT	59.0 ±22.6	45.8 ± 26.9	39.6 ± 8.7	41.5 ± 6.9	41.8 ± 5.8	17.1 ± 14.9	8.3 ± 9.2	7.1 ± 8.0	6.5 ± 6.5
	ARDT		49.5 ± 24.1	39.5 ± 12.0	34.6 ± 6.7	32.6 ± 5.6	29.3 ± 12.4	19.8 ± 8.2	15.1 ± 5.5	13.4 ± 3.1
	RHDT (ours)		**58.3 ± 15.2**	**60.0 ± 14.4**	**59.5 ± 7.3**	**52.2 ± 5.1**	**31.8 ± 5.8**	**27.8 ± 4.9**	**24.5 ± 3.4**	**20.4 ± 2.8**
	Δ_R / Δ_A		+12.5/+8.8	+20.4/+20.5	+18.0/+24.9	+10.4/+19.6	+14.7/+2.5	+19.5/+8.0	+17.4/+9.4	+13.9/+7.0
Walker2d Medium	RDT	61.5 ±10.6	56.6 ± 6.6	51.4 ± 11.4	48.3 ± 11.9	44.5 ± 10.7	21.0 ± 10.0	18.3 ± 12.4	14.7 ± 10.9	11.9 ± 8.5
	ARDT		67.0 ± 3.4	63.1 ± 5.8	57.9 ± 3.9	53.6 ± 3.5	23.5 ± 11.0	17.7 ± 8.8	14.9 ± 3.7	15.3 ± 2.5
	RHDT (ours)		**67.4 ± 5.5**	**65.1 ± 4.1**	**60.3 ± 4.9**	**56.2 ± 5.9**	**36.5 ± 13.1**	**25.5 ± 7.2**	**19.8 ± 3.0**	**19.5 ± 0.7**
	Δ_R / Δ_A		+10.8/+0.4	+13.7/+2.0	+12.0/+2.4	+11.7/+2.6	+15.5/+13.0	+7.2/+7.8	+5.1/+4.9	+7.6/+4.2
Walker2d Med-Replay	RDT	60.5 ±8.1	43.9 ± 18.4	48.2 ± 10.3	52.1 ± 6.5	55.4 ± 5.5	11.9 ± 6.0	11.7 ± 6.4	11.3 ± 6.2	11.1 ± 4.1
	ARDT		55.2 ± 11.7	51.1 ± 11.1	45.2 ± 16.2	41.5 ± 19.9	**22.7 ± 12.4**	**12.1 ± 6.7**	8.8 ± 5.7	7.2 ± 3.2
	RHDT (ours)		**66.8 ± 3.8**	**64.1 ± 3.4**	**61.9 ± 3.7**	**60.3 ± 2.0**	22.3 ± 12.9	12.0 ± 4.4	**12.0 ± 4.6**	**12.2 ± 3.2**
	Δ_R / Δ_A		+22.9/+11.6	+15.9/+13.0	+9.8/+16.7	+4.9/+18.8	+10.4/−0.4	+0.3/−0.1	+0.7/+3.2	+1.1/+5.0

Table 1: Performance on D4RL locomotion tasks under observation corruption. Mean scores and CVaR-10 are computed per training seed over 500 evaluation episodes and averaged over 6 training seeds; ± denotes the standard deviation across training seeds. All values are normalized scores (0=random, 100=expert). The DT column shows baseline Decision Transformer performance at $\sigma = 0$. Δ_R / Δ_A rows show the difference between RHDT and RDT / ARDT. Bold indicates the best performance among RDT, ARDT, and RHDT.

5.3 Ablation Study

We ablate each component by removing it from the full model, using the protocol of Section 5.1, on HalfCheetah Medium and Hopper Med-Replay (Table 2). We remove (1) multi-head attention ($H = 4 \to H = 1$), (2) the gradient penalty ($\lambda_{\text{GP}} \to 0$), and (3) key orthogonality ($\lambda_{\text{ortho}} \to 0$), keeping all other settings at each task's values (Table 5).

Method	HalfCheetah Medium		Hopper Med-Replay	
	Mean	CVaR-10	Mean	CVaR-10
Clean ($\sigma = 0.0$)				
RHDT	42.1 ± 0.4	**40.0 ± 0.4**	58.3 ± 15.2	31.8 ± 5.8
w/o Multi-Head	42.0 ± 0.1	39.1 ± 0.4	54.8 ± 10.5	29.6 ± 9.1
w/o Grad Penalty	**42.4 ± 0.1**	39.7 ± 0.3	60.7 ± 8.7	31.0 ± 12.8
w/o Key Ortho	41.8 ± 0.5	**40.0 ± 0.5**	**62.9 ± 11.6**	**41.4 ± 15.2**
Corrupted ($\sigma = 0.3$)				
RHDT	**35.9 ± 0.7**	**24.6 ± 3.2**	**52.2 ± 5.1**	**20.4 ± 2.8**
w/o Multi-Head	35.0 ± 0.6	19.3 ± 4.4	45.6 ± 8.6	19.6 ± 4.8
w/o Grad Penalty	34.2 ± 0.3	21.0 ± 2.3	46.9 ± 5.2	17.0 ± 1.2
w/o Key Ortho	35.1 ± 1.1	22.9 ± 7.2	51.8 ± 2.7	18.8 ± 3.7

Table 2: Ablation study on HalfCheetah Medium and Hopper Med-Replay. Each ablation removes one component from RHDT; values are mean ± standard deviation across the 6 training seeds. Bold indicates the best value per column.

Under Corruption. At $\sigma = 0.3$, removing any component lowers both the mean score and CVaR-10 on both tasks, and the full model is best in every column. The gradient penalty contributes most consistently: removing it lowers the mean score by 1.7 points on HalfCheetah (on all seeds) and by 5.3 on Hopper, and CVaR-10 by 3.6 and 3.4. Removing multi-head attention mainly lowers CVaR-10

on HalfCheetah ($24.6 \rightarrow 19.3$) and the mean score on Hopper ($52.2 \rightarrow 45.6$, with large variation across seeds), while removing key orthogonality has the smallest effect (at most 1.7 points).

Without Corruption. On HalfCheetah, the components cost little clean performance: all differences are below one point. On Hopper Med-Replay, removing key orthogonality raises clean CVaR-10 ($31.8 \rightarrow 41.4$), while its benefit under corruption on this task is small (1.6 points of CVaR-10), so orthogonality costs clean tail performance here.

5.4 Computational Cost Analysis

Table 3 reports training and inference costs on HalfCheetah Medium, measured on a single NVIDIA RTX PRO 5000 Blackwell GPU.

Method	Params (K)	Training			Inference	
		Time (ms/step)	Full Run (min)	Memory (MiB)	Action Sel. (ms)	Memory (MiB)
DT	859	4.53	7.5	174	1.25	40
RDT	859	4.96	8.3	174	1.28	40
ARDT†	1,135	5.01	8.3	174	1.25	40
RHDT	728	13.91	23.2	401	1.91	36

Table 3: Computational cost on HalfCheetah Medium, measured over 1,000 training steps and 1,000 action selections per training phase. All methods use a 3-layer transformer with embedding dimension 128, context length 20, and batch size 64. †ARDT includes its two value networks (276K parameters), trained for 5K warm-up and 20K minimax steps before its 100K DT steps; its time per step is its total training time divided by its 100K DT steps.

Model Size. RHDT has 15% fewer parameters than the baselines' transformers (728K vs. 859K) because it uses only the learned timestep embedding, without GPT-2's additional position embeddings (131K parameters).

Training Cost. RHDT takes $2.8\times$ longer per gradient step than RDT (13.91 ms vs. 4.96 ms), because the gradient penalty requires a second backward pass through the network and the Hopfield update adds one attention computation per layer. Its peak training memory is 401 MiB vs. 174 MiB, mainly because the gradient penalty keeps the computation graph for double backpropagation.

Inference Cost. Action selection takes 1.91 ms for RHDT vs. 1.28 ms for RDT ($1.5\times$), about 520 actions per second, which is sufficient for MuJoCo control. RHDT uses the least inference memory (36 MiB vs. 40 MiB), since it has fewer parameters.

5.5 Analysis

Figure 2 analyzes RHDT on HalfCheetah Medium over 6 training seeds.

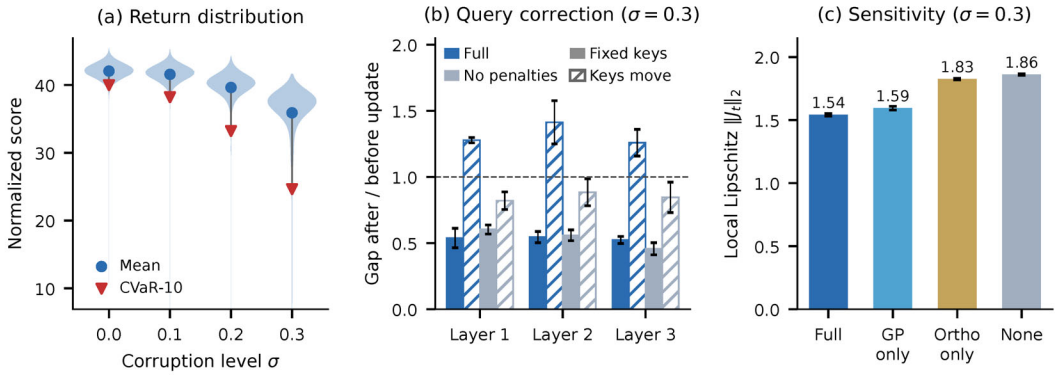

Figure 2: Analysis of RHDT on HalfCheetah Medium (6 training seeds). (a) Normalized scores over 3,000 episodes per corruption level, with the mean score and CVaR-10. (b) Deviation of corrupted queries from clean ones after the Hopfield update relative to before, with both queries updated against the same keys (fixed keys) or in the actual forward pass (keys move); values below 1 mean the update reduces the deviation. (c) Local Lipschitz constant of the policy for the four combinations of the two penalties. Error bars show the standard deviation across training seeds.

Return Distribution. Corruption mainly lowers the tail of the return distribution: from $\sigma = 0$ to $\sigma = 0.3$, the mean score drops by 15% (42.1 to 35.9), while CVaR-10 drops by 38% (40.0 to 24.6), which motivates our focus on worst-case robustness (Figure 2a).

Query Correction. When clean and corrupted queries are updated against the same keys, the Hopfield update halves the deviation of corrupted queries from clean ones in every layer (ratio 0.52 to 0.55 at $\sigma = 0.3$ on all seeds; Figure 2b), consistent with the update pulling corrupted queries toward the same attractors as the clean ones. Because noise also perturbs the keys, and with them the attractors, the deviation after the actual update grows in the full model (ratio 1.26 to 1.41), consistent with the result that removing the update leaves returns unchanged on this task (Appendix D.4).

Sensitivity. The gradient penalty lowers the policy's local Lipschitz constant $\|J_t\|_2$ at corrupted states by about 15% on all seeds (Figure 2c), limiting how strongly corruption reaches the actions.

6 Conclusions, Limitations and Future Work

We introduced RHDT, which replaces the softmax attention of Decision Transformer with modern Hopfield attention, regularized by gradient penalty and key orthogonality, to improve robustness under observation corruption. On six D4RL tasks, RHDT has the highest mean score in all 24 combinations of task and corruption level and the highest CVaR-10 in 22, trailing by at most 0.4 points in the other two. At $\sigma = 0.3$, it improves the task-averaged mean score by 14% and CVaR-10 by 62% over the stronger baseline, and its task-averaged mean score varies less across training seeds than those of the evaluated baselines.

Computational Overhead. RHDT takes about $2.8\times$ longer per training step than the evaluated baselines, because the gradient penalty requires an additional backward pass and the Hopfield update adds an attention step per layer, and about $1.5\times$ longer per action at inference (Table 3). Its rate of about 520 actions per second suffices for MuJoCo control, and applying the gradient penalty only every few steps could reduce the training cost.

Mechanism and Tuning. The Hopfield update adds a modest, task-dependent gain: without it, RHDT's task-averaged scores at $\sigma = 0.3$ still exceed those of the evaluated baselines (Appendix D.4), and a single update moves the queries toward their fixed points without reaching them (Appendix E).

Corruption Types. Our evaluation focuses on Gaussian observation noise, which models sensor measurement errors. Under impulse noise, RHDT keeps its advantage in CVaR-10, but under observation dropout it gains only at the lowest rate and falls behind the baseline on Hopper Med-Replay (Appendix H), and beyond $\sigma = 0.3$ its margins narrow (Appendix F.7). Robustness to adversarial perturbations or systematic calibration drift remains unexplored.

State Space and Observations. The main tasks have low-dimensional proprioceptive states (at most 17 dimensions). Appendix G adds Ant (111 dimensions), Kitchen with sparse rewards, Pen-Human with 25 human demonstrations, and a preliminary Atari Qbert experiment with pixel observations; RHDT has the highest mean score at the strongest corruption level in all five settings, although its CVaR-10 advantage does not carry over to Kitchen. A broader evaluation with pixel observations, where Hopfield attention operates on learned image features, remains future work.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers and the area chair for their constructive feedback. This material is based upon work supported by the U.S. National Science Foundation under Grant No. CCF-1942697 and the Department of Energy under Award No. DE-SC0022014.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:15084–15097, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, pages 2052–2062. PMLR, 2019.

Andreas Fürst, Elisabeth Rumetshofer, Johannes Lehner, Viet T Tran, Fei Tang, Hubert Ramsauer, David Kreil, Michael Kopp, Günter Klambauer, Angela Bitto, et al. Cloob: Modern hopfield networks with infoloob outperform clip. *Advances in neural information processing systems*, 35: 20450–20468, 2022.

Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021. International Conference on Learning Representations (ICLR) 2022.

Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:27532–27559, 2023.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

Shengchao Hu, Ziqing Fan, Chaoqin Huang, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Q-value regularized transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.17098*, 2024. International Conference on Machine Learning (ICML) 2024.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:1273–1286, 2021.

Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Decision convformer: Local filtering in metaformer is sufficient for decision making. *arXiv preprint arXiv:2310.03022*, 2023. International Conference on Learning Representations (ICLR) 2024.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. International Conference on Learning Representations (ICLR) 2022.

Dmitry Krotov and John J Hopfield. Dense associative memory for pattern recognition. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. ICLR 2018.

Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:38966–38979, 2022.

Andreia Podasca and Anup Das. Agatha: Gated hopfield attention for adversarially robust natural language processing. In Hung-Yi Lee, Sunipa Dev, Jessy Li, and Desmond Elliott, editors, *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, pages 16437–16482, Budapest, Hungary, October 2026a. Association for Computational Linguistics. ISBN 979-8-89176-459-0. doi: 10.18653/2026.emnlp-main.709. URL `https://aclanthology.org/2026.emnlp-main.709/`.

Andreia Podasca and Anup Das. Robertha: Eigenspectrum regularized attention for robust natural language understanding. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15229–15264, San Diego, California, United States, July 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.696. URL `https://aclanthology.org/2026.acl-long.696/`.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020. International Conference on Learning Representations (ICLR) 2021.

Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning (ICML)*, pages 8707–8718. PMLR, 2020.

Xiaohang Tang, Afonso Marques, Parameswaran Kamalaruban, and Ilija Bogunovic. Adversarially robust decision transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 37: 29697–29722, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Michael Widrich, Bernhard Schäfl, Milena Pavlović, Hubert Ramsauer, Lukas Gruber, Markus Holzleitner, Johannes Brandstetter, Geir Kjetil Sandve, Victor Greiff, Sepp Hochreiter, et al. Modern hopfield networks and attention for immune repertoire classification. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:18832–18845, 2020.

Yu-Hsuan Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. Stanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *International Conference on Learning Representations*, volume 2024, pages 30886–30925, 2024.

Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:18532–18550, 2023.

Jiawei Xu, Rui Yang, Shuang Qiu, Feng Luo, Meng Fang, Baoxiang Wang, and Lei Han. Tackling data corruption in offline reinforcement learning via sequence modeling. *arXiv preprint arXiv:2407.04285*, 2025. International Conference on Learning Representations (ICLR) 2025.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21024–21037, 2020.

Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. Corruption-robust offline reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTAT)*, pages 5757–5773. PMLR, 2022.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *International Conference on Machine Learning (ICML)*, pages 27042–27059. PMLR, 2022.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Section 5, Appendices A-H.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 6

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Theorems 1–2 and Proposition 1 are stated in Section 4 with proof sketches; full proofs are in Appendix A.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 5.1 and Appendix B.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We use the publicly available D4RL datasets. The code, with instructions to reproduce the main results, is available at `https://github.com/drexel-DISCO/rhdt-public`.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5.1 and Appendix B.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We train each method with 6 training seeds and evaluate each trained model with 10 evaluation seeds and 50 episodes per seed, giving 500 episodes per training seed and 3,000 per task and method. For each training seed we compute the mean score and CVaR-10 over its 500 episodes; tables report the average over training seeds, and $\pm$ denotes the standard deviation across training seeds (Section 5.1).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Section 5.4 and Table 3 report GPU type, training time, memory, and inference cost.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: This work uses publicly available D4RL benchmark datasets and standard MuJoCo simulation environments; no human subjects, private data, or dual-use concerns are involved.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: This work improves robustness of offline RL policies in simulated locomotion tasks and does not present direct negative societal impacts.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks. RHDT is a robustness method for simulated locomotion control and does not involve generative models, scraped datasets, or other high-risk artifacts.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the original papers for all assets used, including D4RL [Fu et al., 2020], MuJoCo, Decision Transformer [Chen et al., 2021], RDT [Xu et al., 2025], and ARDT [Tang et al., 2024]. D4RL is released under the Apache 2.0 license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    The paper does not introduce new datasets or pretrained models; the code is released at `https://github.com/drexel-DISCO/rhdt-public` with instructions to reproduce the results.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects research.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

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

A Proofs and Additional Theoretical Results

A.1 Proof of Theorem 1

Theorem 1 (Hopfield Attention Sensitivity Bound, Restated)**.** *Let $\boldsymbol{\xi}^{(N_{hop})}$ denote the query state after N_{hop} Hopfield iterations with inverse temperature β and keys (stored patterns) $K \in \mathbb{R}^{d\times N}$, defined by the recurrence $\boldsymbol{\xi}^{(t+1)} = K\,\mathrm{softmax}(\beta K^\top \boldsymbol{\xi}^{(t)})$ with $\boldsymbol{\xi}^{(0)} = \boldsymbol{\xi}$. For a perturbation $\boldsymbol{\delta}$ to the initial query, the output perturbation is bounded by:*

$$\|\boldsymbol{\xi}^{(N_{hop})}(\boldsymbol{\xi}+\boldsymbol{\delta}) - \boldsymbol{\xi}^{(N_{hop})}(\boldsymbol{\xi})\|_2 \le (\beta\|K\|_2^2)^{N_{hop}}\|\boldsymbol{\delta}\|_2. \tag{11}$$

Proof. We analyze the Lipschitz constant of a single Hopfield iteration and then compose across N_{hop} iterations.

Step 1: Lipschitz constant of softmax. For a vector $\boldsymbol{z} \in \mathbb{R}^N$, the softmax function $\sigma(\boldsymbol{z})_i = \exp(z_i)/\sum_j \exp(z_j)$ has Jacobian:

$$\frac{\partial \sigma(\boldsymbol{z})}{\partial \boldsymbol{z}} = \mathrm{diag}(\sigma(\boldsymbol{z})) - \sigma(\boldsymbol{z})\sigma(\boldsymbol{z})^\top. \tag{12}$$

Gao and Pavel [2017] show that the softmax with inverse temperature β is β-Lipschitz. A tighter bound follows from Gershgorin's theorem: writing $p = \sigma(\boldsymbol{z})$, row i of this symmetric Jacobian has diagonal entry $p_i - p_i^2$ and off-diagonal absolute sum $p_i(1-p_i)$, so every eigenvalue is at most $2p_i(1-p_i) \le 1/2$, and the spectral norm of the Jacobian is at most $1/2$. For the scaled softmax $\sigma(\beta\boldsymbol{z})$, the Jacobian becomes $\beta[\mathrm{diag}(\sigma(\beta\boldsymbol{z})) - \sigma(\beta\boldsymbol{z})\sigma(\beta\boldsymbol{z})^\top]$, with spectral norm bounded by $\beta/2$.[3]

Step 2: Lipschitz constant of one Hopfield iteration. A single Hopfield iteration is:

$$f(\boldsymbol{\xi}) = K\,\mathrm{softmax}(\beta K^\top \boldsymbol{\xi}). \tag{13}$$

By the chain rule, the Jacobian is:

$$\frac{\partial f}{\partial \boldsymbol{\xi}} = K \cdot \frac{\partial \sigma(\beta K^\top \boldsymbol{\xi})}{\partial(\beta K^\top \boldsymbol{\xi})} \cdot \beta K^\top. \tag{14}$$

Taking spectral norms and bounding the softmax Jacobian by $1/2 \le 1$ from Step 1:

$$\left\|\frac{\partial f}{\partial \boldsymbol{\xi}}\right\|_2 \le \|K\|_2 \cdot \beta \cdot \|K^\top\|_2 = \beta\|K\|_2^2. \tag{15}$$

Since this bound holds at every $\boldsymbol{\xi} \in \mathbb{R}^d$, the mean value inequality gives $\|f(\boldsymbol{\xi}) - f(\boldsymbol{\xi}')\|_2 \le \beta\|K\|_2^2\|\boldsymbol{\xi} - \boldsymbol{\xi}'\|_2$.

Step 3: Composition across N_{hop} iterations. Let $\boldsymbol{\xi}^{(0)} = \boldsymbol{\xi}$ and $\boldsymbol{\xi}^{(t+1)} = f(\boldsymbol{\xi}^{(t)})$. For two initial queries $\boldsymbol{\xi}$ and $\boldsymbol{\xi} + \boldsymbol{\delta}$, we have:

$$\|\boldsymbol{\xi}^{(t+1)} - \tilde{\boldsymbol{\xi}}^{(t+1)}\|_2 \le \beta\|K\|_2^2\|\boldsymbol{\xi}^{(t)} - \tilde{\boldsymbol{\xi}}^{(t)}\|_2, \tag{16}$$

where $\tilde{\boldsymbol{\xi}}^{(0)} = \boldsymbol{\xi} + \boldsymbol{\delta}$ and $\tilde{\boldsymbol{\xi}}^{(t)}$ denotes the trajectory starting from $\boldsymbol{\xi} + \boldsymbol{\delta}$. By induction:

$$\|\boldsymbol{\xi}^{(N_{\text{hop}})} - \tilde{\boldsymbol{\xi}}^{(N_{\text{hop}})}\|_2 \le (\beta\|K\|_2^2)^{N_{\text{hop}}}\|\boldsymbol{\delta}\|_2. \tag{17}$$

□

[3] We use the looser bound of β in Step 2 to simplify the final expression; propagating $\beta/2$ would yield a tighter bound of $(\beta\|K\|_2^2/2)^{N_{\text{hop}}}\|\boldsymbol{\delta}\|_2$.

Corollary 1 (Key Orthogonality Tightens the Sensitivity Bound)**.** *Let $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_N$ be the columns of K and θ_{ij} the angle between $\boldsymbol{k}_i$ and $\boldsymbol{k}_j$. Then*

$$\|K\|_2^2 \leq \max_i \|\boldsymbol{k}_i\|_2^2 \left(1 + (N-1) \max_{i \neq j} |\cos \theta_{ij}|\right). \tag{18}$$

The per-iteration factor $\beta\|K\|_2^2$ in Theorem 1 therefore decreases as the cosine similarities between keys decrease, for fixed key norms.

Proof. Write $K = \hat{K}D$, where $\hat{K}$ has unit-norm columns $\hat{\boldsymbol{k}}_i = \boldsymbol{k}_i / \|\boldsymbol{k}_i\|_2$ and $D = \operatorname{diag}(\|\boldsymbol{k}_1\|_2, \ldots, \|\boldsymbol{k}_N\|_2)$. Then $\|K\|_2 \leq \|\hat{K}\|_2 \|D\|_2 = \|\hat{K}\|_2 \max_i \|\boldsymbol{k}_i\|_2$. The Gram matrix $G = \hat{K}^\top \hat{K}$ has unit diagonal and off-diagonal entries $\cos \theta_{ij}$, and $\|\hat{K}\|_2^2 = \lambda_{\max}(G)$. By Gershgorin's theorem, $\lambda_{\max}(G) \leq \max_i \left(1 + \sum_{j \neq i} |\cos \theta_{ij}|\right) \leq 1 + (N-1) \max_{i \neq j} |\cos \theta_{ij}|$. Combining the two bounds gives the result. □

Corollary 2 (Sensitivity of the Attention Output)**.** *Let $\boldsymbol{y}(\boldsymbol{\xi}) = V \operatorname{softmax}(\beta K^\top \boldsymbol{\xi}^{(N_{hop})})$ be the attention output after N_{hop} iterations, with values $V \in \mathbb{R}^{d_v \times N}$. Then*

$$\|\boldsymbol{y}(\boldsymbol{\xi} + \boldsymbol{\delta}) - \boldsymbol{y}(\boldsymbol{\xi})\|_2 \leq \frac{\beta}{2} \|V\|_2 \|K\|_2 \left(\beta \|K\|_2^2\right)^{N_{hop}} \|\boldsymbol{\delta}\|_2. \tag{19}$$

For $N_{hop} = 0$ this is the corresponding bound for standard softmax attention, so each Hopfield iteration multiplies the worst-case bound by at most $\beta\|K\|_2^2$.

Proof. The readout $g(\boldsymbol{\xi}) = V \operatorname{softmax}(\beta K^\top \boldsymbol{\xi})$ has Jacobian $V \cdot \frac{\partial \sigma}{\partial \boldsymbol{z}} \cdot \beta K^\top$, whose spectral norm is at most $\frac{\beta}{2}\|V\|_2\|K\|_2$ by the $1/2$ bound in Step 1 of the proof of Theorem 1, so g is $\frac{\beta}{2}\|V\|_2\|K\|_2$-Lipschitz by the mean value inequality. Composing g with the N_{hop} iterations bounded in Theorem 1 gives the result. □

Theorem 1 and Corollary 2 hold the keys and values fixed; since $K = \mathbf{H}W_K$ and $V = \mathbf{H}W_V$, observation noise also perturbs them, and a joint bound would add terms proportional to $\|W_K\|_2$ and $\|W_V\|_2$.

A.2 Proof of Proposition 1

Proposition 1 (Gradient Penalty Controls Average Sensitivity, Restated)**.** *Let $\boldsymbol{\theta}^*$ minimize $\mathcal{L}_{action} + \lambda_{GP}\mathcal{L}_{GP}$ with $\mathcal{L}_{action} \geq 0$, and let ℓ_0 be the action loss of the best constant policy. Then $\mathcal{L}_{GP}(\boldsymbol{\theta}^*) \leq \ell_0 / \lambda_{GP}$, so the root-mean-square penalized gradient over the training data is $\mathcal{O}(1/\sqrt{\lambda_{GP}})$.*

Proof. **Step 1: A constant policy has zero penalty.** Let $\boldsymbol{\theta}_0$ be obtained from $\boldsymbol{\theta}^*$ by setting the weights of the final linear layer of the action head to zero and choosing its bias to minimize the action loss among constant outputs. The predicted actions are then independent of the states, so all derivatives $\partial \hat{a}_{t',j} / \partial \boldsymbol{s}_t$ vanish, $\mathcal{L}_{\text{GP}}(\boldsymbol{\theta}_0) = 0$, and $\mathcal{L}_{\text{action}}(\boldsymbol{\theta}_0) = \ell_0$.

Step 2: Comparison. Since $\boldsymbol{\theta}^*$ minimizes the objective,

$$\mathcal{L}_{\text{action}}(\boldsymbol{\theta}^*) + \lambda_{\text{GP}}\mathcal{L}_{\text{GP}}(\boldsymbol{\theta}^*) \leq \mathcal{L}_{\text{action}}(\boldsymbol{\theta}_0) + \lambda_{\text{GP}}\mathcal{L}_{\text{GP}}(\boldsymbol{\theta}_0) = \ell_0. \tag{20}$$

Because $\mathcal{L}_{\text{action}}(\boldsymbol{\theta}^*) \geq 0$, this gives $\mathcal{L}_{\text{GP}}(\boldsymbol{\theta}^*) \leq \ell_0 / \lambda_{\text{GP}}$, and taking the square root gives a root-mean-square penalized gradient of at most $\sqrt{\ell_0 / \lambda_{\text{GP}}}$.

Step 3: Additional loss terms. For the full objective with $\lambda_{\text{ortho}}\mathcal{L}_{\text{ortho}}$, note that $\boldsymbol{\theta}_0$ differs from $\boldsymbol{\theta}^*$ only in the final layer of the action head, which does not affect the attention keys. Hence $\mathcal{L}_{\text{ortho}}(\boldsymbol{\theta}_0) = \mathcal{L}_{\text{ortho}}(\boldsymbol{\theta}^*)$, the orthogonality terms cancel in the comparison, and the bound holds unchanged. □

The weights w_t in $\mathcal{L}_{\text{action}}$ use a stop-gradient, so training follows a semi-gradient of the objective; Proposition 1 therefore describes the trade-off controlled by λ_{GP} rather than a guarantee for the trained network.

Proposition 2 (Relation to Local Lipschitz Constants)**.** *Let $J_t \in \mathbb{R}^{Cd_a \times d_s}$ be the Jacobian of all predicted actions in the context with respect to $\boldsymbol{s}_t$, so that the local Lipschitz constant of the policy at $\boldsymbol{s}_t$ is $\|J_t\|_2$.*

1. *The penalized gradient satisfies $\|J_t^\top \mathbf{1}\|_2^2 \le Cd_a \|J_t\|_2^2$, so a small local Lipschitz constant implies a small penalty.*
2. *For a random vector $\boldsymbol{v}$ with $\mathbb{E}[\boldsymbol{v}\boldsymbol{v}^\top] = I$, $\mathbb{E}_{\boldsymbol{v}}\|J_t^\top \boldsymbol{v}\|_2^2 = \|J_t\|_F^2 \ge \|J_t\|_2^2$.*
3. *If $\boldsymbol{\theta}^*$ minimizes $\mathcal{L}_{action} + \lambda_{GP}\mathcal{L}_{GP}^{\boldsymbol{v}}$, where $\mathcal{L}_{GP}^{\boldsymbol{v}}$ replaces $\mathbf{1}$ with $\boldsymbol{v}$ and takes the expectation over $\boldsymbol{v}$, then $\mathbb{E}_{\tau\sim\mathcal{D}}\big[\frac{1}{C}\sum_{t=1}^{C}\|J_t\|_2^2\big] \le \ell_0/\lambda_{GP}$.*

Proof. For (1), $\|J_t^\top \mathbf{1}\|_2 \le \|J_t^\top\|_2 \|\mathbf{1}\|_2 = \sqrt{Cd_a}\,\|J_t\|_2$. For (2), $\|J_t^\top \boldsymbol{v}\|_2^2 = \boldsymbol{v}^\top J_t J_t^\top \boldsymbol{v}$, whose expectation is $\operatorname{tr}(J_t J_t^\top \mathbb{E}[\boldsymbol{v}\boldsymbol{v}^\top]) = \operatorname{tr}(J_t J_t^\top) = \|J_t\|_F^2$ [Hutchinson, 1989], and $\|J_t\|_F \ge \|J_t\|_2$ holds for any matrix. For (3), a constant policy has $J_t = 0$ and hence $\mathcal{L}_{\text{GP}}^{\boldsymbol{v}} = 0$, so the argument of Proposition 1 gives $\mathcal{L}_{\text{GP}}^{\boldsymbol{v}}(\boldsymbol{\theta}^*) \le \ell_0/\lambda_{\text{GP}}$, and by (2), $\mathcal{L}_{\text{GP}}^{\boldsymbol{v}} = \mathbb{E}_{\tau\sim\mathcal{D}}\big[\frac{1}{C}\sum_t \|J_t\|_F^2\big] \ge \mathbb{E}_{\tau\sim\mathcal{D}}\big[\frac{1}{C}\sum_t \|J_t\|_2^2\big]$. □

These bounds hold on average over training states and are not global: a bound for all inputs, including corrupted observations outside the data, would additionally require constraining the network itself, for example with spectral normalization [Miyato et al., 2018].

A.3 Proof and Discussion of Theorem 2

Theorem 2 (Attractor Basin Radius, Restated)**.** *Consider an MHN with stored patterns $\{\boldsymbol{x}_i\}_{i=1}^N$ of equal norm, in the limit of large β. Every query within distance*

$$r_i = \frac{1}{2}\min_{j\neq i}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \tag{21}$$

of pattern $\boldsymbol{x}_i$ is mapped to $\boldsymbol{x}_i$ by the update in Eq. (3)*.*

Proof. **Step 1: Large-β limit.** As $\beta \to \infty$, $\operatorname{softmax}(\beta \boldsymbol{X}^\top \boldsymbol{\xi})$ converges to the one-hot vector of the largest entry of $\boldsymbol{X}^\top \boldsymbol{\xi}$ whenever that entry is unique, so one update maps $\boldsymbol{\xi}$ to $\boldsymbol{x}_{i^*}$ with $i^* = \arg\max_j \boldsymbol{\xi}^\top \boldsymbol{x}_j$.

Step 2: Similarity and distance. Since $\|\boldsymbol{\xi} - \boldsymbol{x}_j\|_2^2 = \|\boldsymbol{\xi}\|_2^2 - 2\boldsymbol{\xi}^\top \boldsymbol{x}_j + \|\boldsymbol{x}_j\|_2^2$ and all patterns have equal norm, $\boldsymbol{\xi}^\top \boldsymbol{x}_i > \boldsymbol{\xi}^\top \boldsymbol{x}_j$ holds if and only if $\|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2 < \|\boldsymbol{\xi} - \boldsymbol{x}_j\|_2$.

Step 3: The ball of radius r_i. If $\|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2 < r_i$, then for every $j \neq i$ the triangle inequality gives

$$\|\boldsymbol{\xi} - \boldsymbol{x}_j\|_2 \ge \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 - \|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2 > 2r_i - r_i = r_i > \|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2. \tag{22}$$

By Step 2, $\boldsymbol{x}_i$ is then the unique most similar pattern, and by Step 1 the update maps $\boldsymbol{\xi}$ to $\boldsymbol{x}_i$. □

Geometry of the basins. For equal-norm patterns, the boundary between the basins of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the large-β limit is the hyperplane $\boldsymbol{\xi}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j) = 0$, which contains the midpoint $(\boldsymbol{x}_i + \boldsymbol{x}_j)/2$. At the midpoint, $\boldsymbol{\xi}^\top \boldsymbol{x}_i = \boldsymbol{\xi}^\top \boldsymbol{x}_j$, so the query is equally similar to both patterns. The basin of $\boldsymbol{x}_i$ is therefore its Voronoi cell, and r_i is the radius of the largest ball around $\boldsymbol{x}_i$ that lies inside it.

Finite β. For finite β, the softmax assigns weight to several patterns, so the update maps queries near a basin boundary toward a weighted average of patterns. For small β or similar patterns, such averages can be stable metastable states [Ramsauer et al., 2020], so the basins are smaller than the Voronoi cells. Ramsauer et al. [2020] show that retrieval near $\boldsymbol{x}_i$ succeeds when the separation

$\Delta_i = \boldsymbol{x}_i^\top \boldsymbol{x}_i - \max_{j \neq i} \boldsymbol{x}_i^\top \boldsymbol{x}_j$ exceeds a threshold that depends on β, the number of patterns, and the pattern norms; for unit-norm patterns, $\Delta_i = 1 - \max_{j \neq i} \cos\theta_{ij}$, which the orthogonality penalty tends to increase by pushing positive cosine similarities toward zero.

Orthonormal patterns. For orthonormal patterns, $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = \|\boldsymbol{x}_i\|_2^2 + \|\boldsymbol{x}_j\|_2^2 - 2\boldsymbol{x}_i^\top \boldsymbol{x}_j = 2$, so $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = \sqrt{2}$ and $r_i = \sqrt{2}/2$ for all pairs. Other configurations, such as unit-norm patterns at the vertices of a regular simplex, achieve pairwise distances exceeding $\sqrt{2}$, so orthogonality provides a tractable regularization target rather than the theoretical maximum. When the number of patterns exceeds the dimension, as in our setting ($N = 60$ keys in $d_k = 32$ dimensions per head), exact orthogonality is not attainable, and the penalty instead reduces the pairwise cosine similarities.

Proposition 3 (Limits of Key Orthogonality)**.** *Let $\hat{\boldsymbol{k}}_1, \ldots, \hat{\boldsymbol{k}}_N \in \mathbb{R}^d$ be unit vectors with $N > d$, and let G be their Gram matrix. Then*

1. $\frac{1}{N^2}\|G - I\|_F^2 \geq \frac{N-d}{Nd}$*, so the orthogonality penalty is bounded away from zero;*
2. $\max_{i \neq j} |\cos\theta_{ij}| \geq \sqrt{\frac{N-d}{d(N-1)}}$*;*
3. $\lambda_{\max}(G) \geq N/d$*.*

In our setting ($N = 60$, $d = 32$), the orthogonality penalty is at least 0.0146, some pair of keys in each head has $|\cos\theta_{ij}| \geq 0.122$, and $\|\hat{K}\|_2^2 \geq 1.875$ in Corollary 1.

Proof. G is positive semidefinite with unit diagonal, so $\mathrm{tr}(G) = N$, and its rank is at most d. Its at most d nonzero eigenvalues therefore sum to N, and the Cauchy–Schwarz inequality gives $\|G\|_F^2 = \sum_k \lambda_k^2 \geq N^2/d$. Since the diagonal of G is one, $\|G - I\|_F^2 = \sum_{i \neq j} \cos^2\theta_{ij} = \|G\|_F^2 - N \geq N(N-d)/d$, which gives (1) after dividing by N^2. Bounding each of the $N(N-1)$ off-diagonal terms by $\max_{i \neq j} \cos^2\theta_{ij}$ gives (2), and (3) follows because the largest of at most d nonzero eigenvalues summing to N is at least N/d. □

A.4 Remark on Multi-Head Ensemble

Remark 1 (Multi-Head Ensemble)**.** *Concatenating the head outputs and applying the output projection W_O gives $\sum_{h=1}^H W_O^{(h)} \boldsymbol{o}_h$, a learned weighted sum of the heads' retrievals, where $W_O^{(h)}$ is the block of W_O acting on head h. In the special case $W_O^{(h)} = P/H$ for a shared matrix P, this is an average, and under idealized assumptions of independent errors across heads, the standard deviation of the output error decreases by a factor of $\sqrt{H}$.*

Derivation. Let $\boldsymbol{e}_h$ denote the error of head h's output under an input perturbation $\boldsymbol{\delta}$, so that the output error is $\boldsymbol{e}_{\text{total}} = \sum_{h=1}^H W_O^{(h)} \boldsymbol{e}_h$. With $W_O^{(h)} = P/H$, this becomes

$$\boldsymbol{e}_{\text{total}} = P \frac{1}{H} \sum_{h=1}^{H} \boldsymbol{e}_h. \tag{23}$$

If the errors $\boldsymbol{e}_h$ are independent with $\mathbb{E}[\boldsymbol{e}_h] = \boldsymbol{\mu}$ and $\mathrm{Cov}(\boldsymbol{e}_h) = \sigma^2 I$, then $\frac{1}{H}\sum_h \boldsymbol{e}_h$ has mean $\boldsymbol{\mu}$ and covariance $\frac{\sigma^2}{H} I$, so

$$\mathrm{Cov}(\boldsymbol{e}_{\text{total}}) = \frac{\sigma^2}{H} P P^\top, \tag{24}$$

which is a factor of H smaller than for a single head with the same projection P, while the mean error $P\boldsymbol{\mu}$ is unchanged. The standard deviation of the error therefore scales as $\sigma/\sqrt{H}$. This is a reduction in average-case statistical noise, not in the worst-case deterministic Lipschitz constant: an average of H functions, each with Lipschitz constant L, still has Lipschitz constant L, and for the learned weighted sum the bound is $\sum_h \|W_O^{(h)}\|_2 L_h$, where L_h is the Lipschitz constant of head h. □

In practice, attention heads share the same input and are trained jointly, so their errors are not independent, and W_O is learned rather than fixed to an average. The orthogonality penalty acts on the keys within each head and does not directly encourage diversity across heads. The single-head ablation on HalfCheetah Medium and Hopper Med-Replay in Section 5.3 measures the net effect of using multiple heads.

B Implementation Details

All methods use a 3-layer transformer with embedding dimension 128, an MLP of width 512 with ReLU, and a context of 20 timesteps (60 tokens). They are trained for 100K steps on batches of 64 trajectory segments with AdamW (learning rate 10^{-4}, weight decay 10^{-4} on all parameters, default $\beta_1, \beta_2, \epsilon$), a linear warm-up over the first 10K steps followed by a constant learning rate, and gradient clipping at a global norm of 0.25, and every method uses its final weights without checkpoint selection. Table 4 lists the settings that differ between methods.

	DT	RDT	ARDT	RHDT
Attention	softmax, 1 head	softmax, 1 head	softmax, 1 head	Hopfield, 4 heads
Parameters[a]	858,758	859,143	858,758[b]	727,558
Dropout	0.1	0.1, embedding per task	0.1	0.1, embedding 0.05
Action loss	MSE	weighted (β_a per task)	MSE	weighted ($\beta_a = 1$)
Additional losses	none	reward prediction	none	GP, key orthogonality
Returns-to-go	dataset	dataset	minimax relabelled	dataset
Evaluated at	$\sigma = 0$	all levels	all levels	all levels

[a]HalfCheetah and Walker2d; for Hopper: 857,219 (DT, ARDT), 857,604 (RDT), and 726,019 (RHDT). [b]Plus two value networks of 137,985 parameters each (135,681 for Hopper), used only for relabelling.

Table 4: Settings that differ between methods. All other settings are shared (see text).

RHDT (ours). Returns-to-go, states, and actions are embedded by linear layers to 128 dimensions, with a learned timestep embedding, followed by LayerNorm and dropout 0.05. Each of the 3 pre-norm blocks applies Hopfield attention with 4 heads of dimension 32, one Hopfield update ($N_{\text{hop}} = 1$) with $\beta = 1/\sqrt{32}$, a causal and padding mask, and attention dropout 0.1 on the readout weights only, followed by the MLP with output dropout 0.1. The action is predicted from the state token through a final LayerNorm, a linear layer, and tanh. The loss is the weighted action loss of Section 4.4 plus the gradient penalty and the key orthogonality penalty, with the per-task weights in Table 5. All three terms are computed from the same forward pass, and padded timesteps count in the denominators.

DT [Chen et al., 2021]. DT uses the Hugging Face `GPT2Model` with one head, GPT-2's learned position embeddings in addition to the timestep embedding, and dropout 0.1 on embeddings, attention, and residual connections. It is trained with the unweighted squared action error, summed over action dimensions and averaged over unpadded timesteps.

RDT [Xu et al., 2025]. RDT uses the same backbone as DT, with embedding dropout before the transformer, a Gaussian-weighted action loss with weights $w = \exp(-\beta_a \cdot \text{mean squared action error})$ computed without gradient, and a reward-prediction head on the action token whose loss is scaled by a per-task weight w_r (Table 5). The reward loss is an unweighted mean squared error ($\beta_r = 0$) on rewards divided by 1000.

ARDT [Tang et al., 2024]. ARDT first trains two value networks $Q(s, a)$, each an MLP with 3 hidden layers of 256 units and ReLU, using AdamW (learning rate 3×10^{-4}, weight decay 0.01) on batches of 256 transitions. Both networks first regress the dataset returns-to-go for 5K warm-up steps; they are then trained by alternating expectile regression on $r/1000 + 0.99\,(1 - d)\,Q'(s', a')$, using the other network's prediction Q', with expectile α for one network and $1 - \alpha$ for the other, for the per-task number of minimax steps in Table 5. Every return-to-go in the dataset is then replaced by the prediction of the $(1 - \alpha)$-expectile network, and a DT with the same backbone and loss as above is trained on the relabelled data.

Seeds, software, and hardware. All methods use the same 6 training seeds (907561, 48213, 2024887, 739104, 26451, and 42) and the same 10 evaluation seeds (11, 13, 17, 19, 23, 29, 31, 37, 41, and 43), with 50 episodes per evaluation seed. Training and evaluation fix all random seeds and run PyTorch in deterministic mode. The training data are the D4RL v2 datasets, loaded from the Hugging Face Hub (`edbeeching/decision_transformer_gym_replay`), and evaluation uses the corresponding D4RL environments in Gym (e.g., `halfcheetah-medium-v2`). The experiments use Python 3.10, PyTorch 2.11.0 with CUDA 12.8, NumPy 1.26.4, Gym 0.23.1, D4RL 1.1, `mujoco-py` 2.1.2.14 with MuJoCo 2.1.0, and Hugging Face Transformers 4.44.2 and Datasets 2.21.0. All training and evaluation runs use NVIDIA RTX PRO 5000 Blackwell GPUs. The code is available at `https://github.com/drexel-DISCO/rhdt-public`.

B.1 Hyperparameter Selection

For each method and task, we selected the hyperparameter setting with the best clean score ($\sigma = 0$). For every method, these clean scores come from the same 10 evaluation seeds as the reported results. Selection used only uncorrupted evaluations, so no setting was chosen based on performance under corruption. Table 5 lists the setting selected for each method and task.

	RHDT		RDT			ARDT	
Task	λ_{GP}	λ_{ortho}	β_a	w_r	p_{emb}	α	Minimax steps
HalfCheetah Medium	0.005	0.05	0.1	0	0.05	0.1	20K
HalfCheetah Med-Replay	0.02	0.025	2.0	0.1	0.50	0.8	10K
Hopper Medium	0.001	0.2	0.1	0.1	0.05	0.8	10K
Hopper Med-Replay	0.0005	0.025	1.0	0	0.35	1.0	20K
Walker2d Medium	0.001	0.2	0.3	0.1	0.35	0.8	10K
Walker2d Med-Replay	0.0005	0.025	0.3	0.1	0.05	1.0	10K

Table 5: Per-task hyperparameters. RHDT: gradient-penalty and key-orthogonality weights. RDT: action-loss weighting β_a, reward-loss weight w_r, and embedding dropout p_{emb}. ARDT: expectile level α and minimax steps.

C Training Dynamics

Figure 3 shows the three loss terms of RHDT during training on HalfCheetah Medium and Hopper Med-Replay, averaged over every 500 steps, for the 6 training seeds. Training is stable across seeds: the final action loss is 0.0278 ± 0.00005 on HalfCheetah Medium and 0.0752 ± 0.00015 on Hopper Med-Replay. The orthogonality term decreases throughout training (from 0.096 to 0.032 and from 0.134 to 0.074) and stays above the lower bound of Proposition 3 (0.0146 for 60 keys in 32 dimensions). The gradient-penalty term rises during the learning-rate warm-up on HalfCheetah Medium, peaks at 0.72 after 7K steps, and falls to 0.30; on Hopper Med-Replay, where λ_{GP} is ten times smaller, it grows throughout training to 2.62, so the weaker penalty does not stop the input gradient from growing as the policy fits the data. At the end of training, the action loss makes up 90% and 96% of the total loss on the two tasks, with the weighted gradient-penalty term contributing 4.8% and 1.7% and the weighted orthogonality term 5.2% and 2.3%.

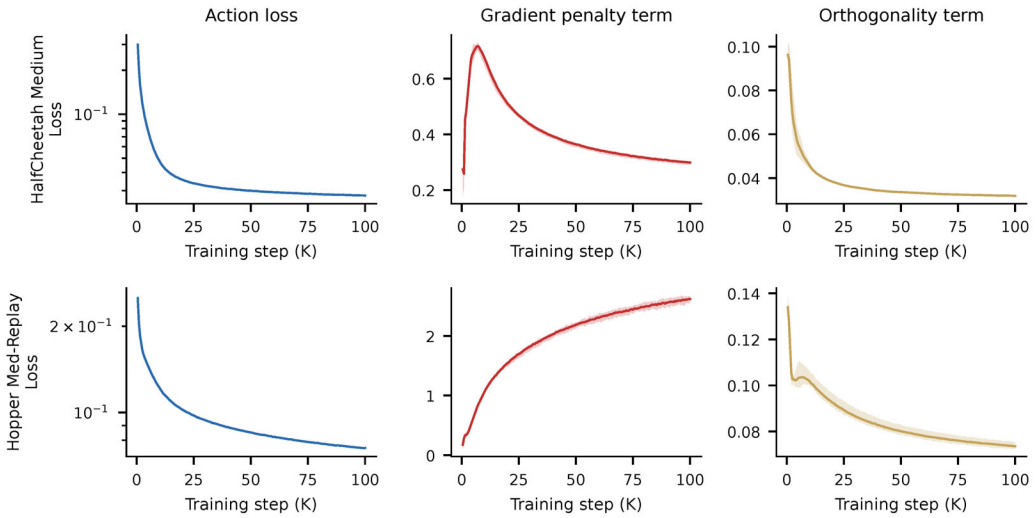

Figure 3: Training dynamics of RHDT on HalfCheetah Medium (top) and Hopper Med-Replay (bottom). Each panel shows one loss term, averaged over every 500 training steps; lines show the mean over 6 training seeds and shaded areas the range across seeds. The gradient-penalty and orthogonality terms are shown before weighting by λ_{GP} and λ_{ortho}.

D Hyperparameter Sensitivity

This section measures how sensitive RHDT is to the four hyperparameters it adds to DT: the inverse temperature β of the Hopfield attention, the gradient penalty weight λ_{GP}, the key orthogonality weight λ_{ortho}, and the number of Hopfield updates N_{hop}. Starting from the chosen configuration of each task (Table 5), we change one hyperparameter at a time and keep all others fixed. We use HalfCheetah Medium and Hopper Med-Replay, the two tasks of the ablation study, and vary β from 0.5 to 5 times its default value $1/\sqrt{d_k}$, the two penalty weights from 0 to 4 times their chosen values, and N_{hop} from 0 to 3. Because N_{hop} defines the architecture, we also compare $N_{\text{hop}} \in \{0, 1, 2\}$ on all six tasks. Every setting is trained with the same 6 training seeds and evaluated with the protocol of the main results (Section 5.1), so the settings with a penalty weight of 0 coincide with the ablation variants without that penalty. Table 6 reports all settings at every corruption level, Table 7 reports N_{hop} on all six tasks, Figure 4 plots the mean score and CVaR-10 against each hyperparameter, and Figure 5 shows measurements of the trained models that explain the main trends.

Task	Setting	Mean Score				CVaR-10			
		Corruption Level (σ)				Corruption Level (σ)			
		0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
HalfCheetah Medium	**Chosen**	42.1 ± 0.4	41.6 ± 0.3	39.6 ± 0.4	35.9 ± 0.7	40.0 ± 0.4	38.2 ± 1.1	33.2 ± 1.4	24.6 ± 3.2
	$\beta \times 0.5$	42.2 ± 0.3	41.7 ± 0.3	39.6 ± 0.4	35.9 ± 0.6	39.8 ± 0.4	38.7 ± 0.7	32.2 ± 2.3	24.8 ± 2.9
	$\beta \times 1.5$	41.9 ± 0.2	41.5 ± 0.2	39.4 ± 0.4	35.5 ± 0.9	39.7 ± 0.5	38.6 ± 1.2	31.2 ± 4.1	21.4 ± 6.1
	$\beta \times 2$	41.9 ± 0.4	41.4 ± 0.3	39.2 ± 0.7	35.5 ± 1.1	$39.8 \pm 0.4^*$	38.3 ± 0.7	29.9 ± 6.6	23.2 ± 5.8
	$\beta \times 5$	42.0 ± 0.4	41.5 ± 0.3	$39.3 \pm 0.5^*$	35.5 ± 0.7	$39.4 \pm 0.9^*$	38.1 ± 0.9	30.0 ± 5.2	22.7 ± 5.1
	$\lambda_{\text{GP}} \times 0$	$42.4 \pm 0.1^*$	41.5 ± 0.1	$38.8 \pm 0.1^*$	$34.2 \pm 0.3^*$	39.7 ± 0.3	$36.3 \pm 1.3^*$	$29.3 \pm 1.4^*$	21.0 ± 2.3
	$\lambda_{\text{GP}} \times 0.25$	$42.3 \pm 0.2^*$	$41.8 \pm 0.1^*$	39.3 ± 0.4	35.3 ± 0.2	$39.3 \pm 0.6^*$	37.8 ± 1.1	$29.6 \pm 3.4^*$	23.0 ± 1.1
	$\lambda_{\text{GP}} \times 4$	$26.8 \pm 15.4^*$	$25.1 \pm 14.5^*$	$20.0 \pm 11.8^*$	$13.2 \pm 7.7^*$	$11.3 \pm 11.4^*$	$8.5 \pm 9.0^*$	$5.5 \pm 6.2^*$	$3.9 \pm 4.3^*$
	$\lambda_{\text{ortho}} \times 0$	41.8 ± 0.5	41.3 ± 0.3	$39.1 \pm 0.4^*$	35.1 ± 1.1	40.0 ± 0.5	38.3 ± 1.1	31.3 ± 5.6	22.9 ± 7.2
	$\lambda_{\text{ortho}} \times 0.25$	40.8 ± 1.9	40.3 ± 1.9	38.5 ± 1.6	35.0 ± 1.4	38.8 ± 2.1	37.6 ± 1.7	33.7 ± 2.3	26.8 ± 3.8
	$\lambda_{\text{ortho}} \times 4$	42.1 ± 0.3	$41.7 \pm 0.2^*$	39.8 ± 0.4	36.0 ± 0.5	40.1 ± 0.3	38.7 ± 0.7	33.4 ± 3.7	24.6 ± 3.0
	$N_{\text{hop}} = 0$	42.1 ± 0.1	41.7 ± 0.1	39.6 ± 0.2	36.0 ± 0.1	39.7 ± 0.6	38.5 ± 0.7	31.6 ± 2.6	24.4 ± 1.9
	$N_{\text{hop}} = 2$	42.0 ± 0.5	41.5 ± 0.4	39.5 ± 0.4	36.0 ± 0.1	39.8 ± 0.5	38.2 ± 0.9	32.6 ± 3.1	25.8 ± 2.3
	$N_{\text{hop}} = 3$	$41.8 \pm 0.3^*$	$41.4 \pm 0.2^*$	$39.2 \pm 0.4^*$	35.5 ± 0.4	39.3 ± 1.2	38.4 ± 0.9	30.6 ± 3.1	22.9 ± 2.3
Hopper Med-Replay	**Chosen**	58.3 ± 15.2	60.0 ± 14.4	59.5 ± 7.3	52.2 ± 5.1	31.8 ± 5.8	27.8 ± 4.9	24.5 ± 3.4	20.4 ± 2.8
	$\beta \times 0.5$	60.3 ± 9.9	61.4 ± 4.0	59.4 ± 3.9	52.3 ± 3.9	35.3 ± 12.3	28.5 ± 9.4	25.6 ± 3.9	21.7 ± 3.0
	$\beta \times 1.5$	57.4 ± 14.9	57.3 ± 15.8	58.0 ± 12.1	52.3 ± 7.7	37.1 ± 11.0	26.8 ± 8.1	25.6 ± 8.2	19.9 ± 4.0
	$\beta \times 2$	52.7 ± 13.0	59.1 ± 13.0	59.3 ± 7.3	53.0 ± 5.2	31.4 ± 10.3	27.6 ± 10.3	23.5 ± 5.2	19.7 ± 3.7
	$\beta \times 5$	53.5 ± 16.8	$54.2 \pm 14.5^*$	53.9 ± 11.2	47.6 ± 8.1	33.2 ± 10.5	27.4 ± 10.2	23.8 ± 6.0	20.1 ± 3.0
	$\lambda_{\text{GP}} \times 0$	60.7 ± 8.7	59.9 ± 11.1	55.8 ± 7.5	$46.9 \pm 5.2^*$	31.0 ± 12.8	26.9 ± 11.2	22.8 ± 6.4	$17.0 \pm 1.2^*$
	$\lambda_{\text{GP}} \times 0.25$	61.2 ± 16.1	58.8 ± 14.9	56.6 ± 10.2	$48.9 \pm 4.7^*$	$40.0 \pm 8.3^*$	$29.5 \pm 6.6^*$	23.7 ± 5.4	17.6 ± 2.3
	$\lambda_{\text{GP}} \times 4$	55.1 ± 22.7	57.4 ± 14.9	59.2 ± 9.8	55.3 ± 6.2	28.3 ± 13.9	22.6 ± 9.2	23.9 ± 8.5	20.3 ± 5.4
	$\lambda_{\text{ortho}} \times 0$	62.9 ± 11.6	62.2 ± 9.6	59.0 ± 3.8	51.8 ± 2.7	$41.4 \pm 15.2^*$	30.5 ± 8.0	22.7 ± 3.9	18.8 ± 3.7
	$\lambda_{\text{ortho}} \times 0.25$	64.0 ± 17.5	62.3 ± 14.2	59.2 ± 10.3	53.7 ± 7.6	39.9 ± 15.5	28.8 ± 8.7	22.8 ± 7.4	19.2 ± 3.9
	$\lambda_{\text{ortho}} \times 4$	57.2 ± 23.9	60.6 ± 14.4	59.0 ± 10.6	53.1 ± 9.9	38.3 ± 23.5	28.7 ± 14.5	25.1 ± 7.1	19.7 ± 3.4
	$N_{\text{hop}} = 0$	57.9 ± 14.4	57.3 ± 12.9	54.2 ± 8.9	47.0 ± 7.3	36.4 ± 12.8	26.6 ± 7.0	22.5 ± 3.2	18.3 ± 2.3
	$N_{\text{hop}} = 2$	$71.6 \pm 7.9^*$	65.1 ± 11.4	64.3 ± 10.3	55.0 ± 7.5	35.6 ± 11.1	28.9 ± 7.0	27.5 ± 5.0	21.4 ± 2.8
	$N_{\text{hop}} = 3$	56.4 ± 23.2	61.0 ± 17.3	62.2 ± 10.8	54.6 ± 6.0	36.4 ± 14.0	32.0 ± 10.7	28.5 ± 6.4	22.6 ± 1.3

Table 6: Sensitivity of RHDT to its hyperparameters on HalfCheetah Medium and Hopper Med-Replay. Each row changes one hyperparameter of the chosen configuration (Table 5) and keeps the others fixed; multipliers are relative to the chosen value, and $\beta \times 1$ is the default $\beta = 1/\sqrt{d_k}$. Values are averaged over 6 training seeds, each evaluated on 500 episodes, and $\pm$ denotes the standard deviation across training seeds. * marks a mean difference from the chosen configuration larger than two standard errors of the per-seed differences.

D.1 Inverse Temperature

The inverse temperature β scales the query-key similarities in both the Hopfield update and the readout, so a larger β concentrates attention on fewer keys and raises the factor $\beta\|K\|^2$ in the sensitivity bound of Appendix A.

Scores. On HalfCheetah Medium, every value of β from 0.5 to 5 times the default stays within 0.4 points of the chosen setting in mean score at every corruption level. Its CVaR-10 at $\sigma = 0.3$ ranges from 21.4 to 24.8 without a trend in β, and these differences are smaller than the across-seed standard deviations (2.9 to 6.1). On Hopper Med-Replay, values from 0.5 to 2 times the default are

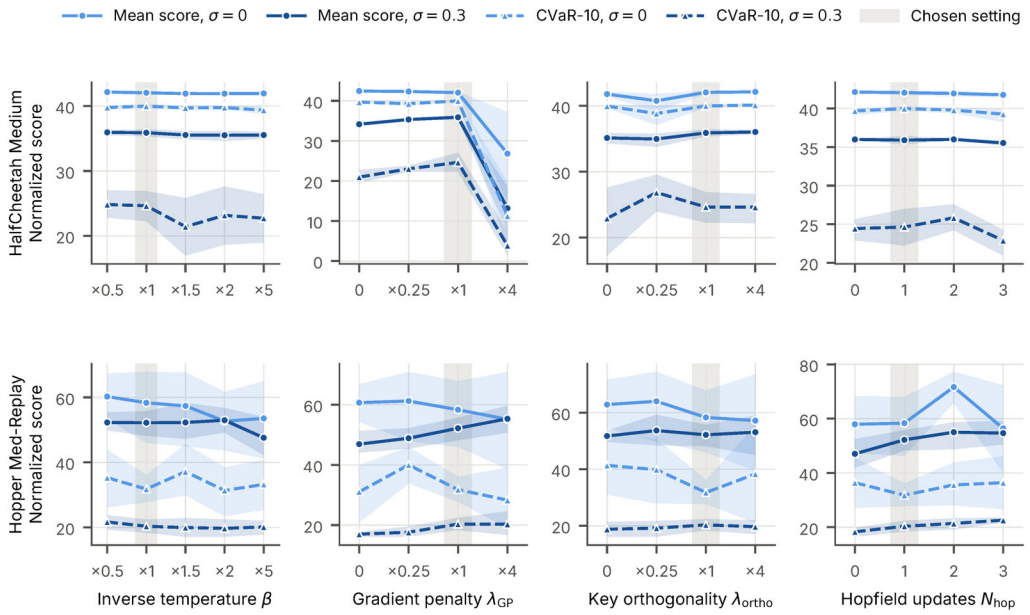

Figure 4: Mean score and CVaR-10 of RHDT as each hyperparameter is varied around the chosen configuration (gray column), on HalfCheetah Medium (top) and Hopper Med-Replay (bottom). Multipliers are relative to the chosen value, and $\beta \times 1$ is the default $\beta = 1/\sqrt{d_k}$. Lines show the average over 6 training seeds at $\sigma = 0$ (light) and $\sigma = 0.3$ (dark), and shaded bands show 95% bootstrap confidence intervals over training seeds.

within seed noise of the chosen setting, while $\beta \times 5$ lowers the mean score at every corruption level, by 4.6 to 5.8 points.

The learned norms compensate for β. Figure 5(a, b) shows why β matters little. As β increases from 0.5 to 5 times the default, the trained models shrink their keys and queries: the mean key norm decreases from 4.9 to 2.0 on HalfCheetah Medium and from 7.1 to 2.6 on Hopper Med-Replay, and the mean query norm decreases from 12.9 to 3.7 and from 9.5 to 3.8. As a result, the attention logit scale $\beta\|q\|\|k\|$ increases by only 12% and 48% while β increases tenfold, and $\beta\|k\|^2$ grows by a factor of 1.6 and 1.3. Because queries and keys are learned linear projections, the network can absorb most of a change in β into their scale, so the attention it learns depends little on the value of β. The Hopfield update also corrects query perturbations to a similar degree at every β: with the keys fixed, it reduces the distance between clean and corrupted queries to between 51% and 58% of its value before the update on both tasks (Figure 5(c)). The default $\beta = 1/\sqrt{d_k}$ is therefore a reasonable choice, and only the largest value we tested lowers performance, on one of the two tasks.

D.2 Gradient Penalty Weight

The gradient penalty limits the sensitivity of the predicted actions to the input states, so its weight λ_{GP} trades robustness against the ability to fit the actions in the dataset.

The penalty reduces sensitivity monotonically. The local Lipschitz constant $\|J_t\|_2$ decreases with λ_{GP} on both tasks (Figure 5(d)), from 1.83 without the penalty to 1.64, 1.55, and 1.47 at 0.25, 1, and 4 times the chosen weight on HalfCheetah Medium, and from 1.78 to 1.70, 1.57, and 1.37 on Hopper Med-Replay. The penalized quantity $\|J_t^\top \mathbf{1}\|_2^2$ decreases much more, by 98% and 76% from no penalty to 4 times the chosen weight, consistent with the gap between the penalized quantity and $\|J_t\|_2$ discussed in Appendix A.

Up to the chosen weight, a larger penalty improves robustness. Between no penalty and the chosen weight, the mean score and CVaR-10 at $\sigma = 0.3$ increase with λ_{GP} on both tasks: from 34.2

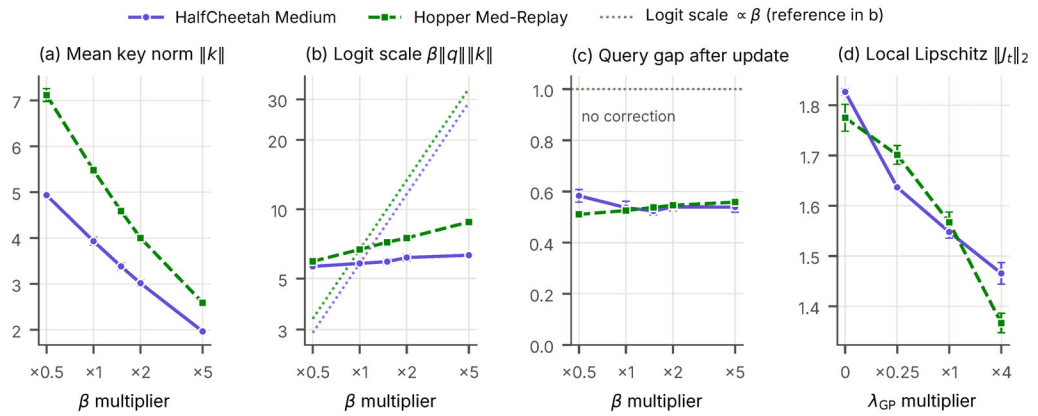

Figure 5: Measurements of trained RHDT models that explain the sensitivity results. (a) Mean norm of the attention keys and (b) attention logit scale $\beta\|q\|\|k\|$ as functions of β, averaged over layers; the dotted lines in (b) show the logit scale that would result if the norms did not change with β. (c) Distance between the clean and corrupted queries after the Hopfield update, relative to the distance before it, with both queries updated against the same corrupted keys at $\sigma = 0.3$. (d) Local Lipschitz constant $\|J_t\|_2$ of the policy at dataset states as a function of λ_{GP}. Error bars show the standard deviation across 6 training seeds.

to 35.3 and 35.9 in mean score and from 21.0 to 23.0 and 24.6 in CVaR-10 on HalfCheetah Medium, and from 46.9 to 48.9 and 52.2 and from 17.0 to 17.6 and 20.4 on Hopper Med-Replay. Removing the penalty lowers the mean score at $\sigma = 0.3$ by more than two standard errors on both tasks, and on HalfCheetah Medium it lowers the mean score on all six training seeds. The cost without corruption is small: on HalfCheetah Medium, the clean mean score is 0.3 points lower with the chosen weight than without the penalty.

Too large a penalty prevents learning on HalfCheetah Medium. At 4 times the chosen weight, HalfCheetah Medium degrades at every corruption level, to a clean mean score of 26.8 ± 15.4 and a mean score of 13.2 at $\sigma = 0.3$, lower than the chosen setting on all six training seeds. Its clean mean score drops by 15.3 points, so the penalty prevents the policy from fitting the dataset actions rather than trading clean performance for robustness, even though its Lipschitz constant is only 5% lower than with the chosen weight. On Hopper Med-Replay, 4 times the chosen weight performs similarly to the chosen setting under corruption (55.3 versus 52.2 in mean score at $\sigma = 0.3$, within seed noise). The suitable range of λ_{GP} is therefore task-dependent.

D.3 Key Orthogonality Weight

The orthogonality penalty separates the stored keys within each context window, and its weight λ_{ortho} sets how far the keys are pushed apart. Under corruption, the results depend little on this weight: between 0 and 4 times the chosen value, the mean score and CVaR-10 at $\sigma = 0.3$ stay within 2.2 points of the chosen setting on both tasks, and none of these differences exceeds two standard errors. Without corruption, the weight also matters little on HalfCheetah Medium, where 4 times the chosen value matches the chosen setting and the lower clean mean score at 0.25 times the chosen value (40.8 ± 1.9) comes mainly from one training seed. On Hopper Med-Replay, the chosen weight gives the lowest clean CVaR-10 of the four settings (31.8, against 38.3 to 41.4), but clean results on this task vary widely across training seeds, with standard deviations up to 23.5.

D.4 Number of Hopfield Updates

Table 7 compares standard softmax attention ($N_{\text{hop}} = 0$), one Hopfield update ($N_{\text{hop}} = 1$, the configuration of the main results), and two updates on all six tasks. Each update adds one softmax attention step per layer, so a larger N_{hop} also increases the training and inference cost.

Task	N_{hop}	Mean Score				CVaR-10			
		Corruption Level (σ)				Corruption Level (σ)			
		0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
HalfCheetah Medium	0	42.1 ± 0.1	41.7 ± 0.1	39.6 ± 0.2	36.0 ± 0.1	39.7 ± 0.6	38.5 ± 0.7	31.6 ± 2.6	24.4 ± 1.9
	1 (chosen)	42.1 ± 0.4	41.6 ± 0.3	39.6 ± 0.4	35.9 ± 0.7	40.0 ± 0.4	38.2 ± 1.1	33.2 ± 1.4	24.6 ± 3.2
	2	42.0 ± 0.5	41.5 ± 0.4	39.5 ± 0.4	36.0 ± 0.1	39.8 ± 0.5	38.2 ± 0.9	32.6 ± 3.1	25.8 ± 2.3
HalfCheetah Med-Replay	0	$38.0 \pm 0.4^*$	36.9 ± 0.6	33.6 ± 0.9	28.7 ± 1.1	30.4 ± 2.8	24.8 ± 6.8	13.1 ± 4.5	6.9 ± 1.7
	1 (chosen)	37.4 ± 0.8	36.5 ± 0.9	33.8 ± 0.9	28.9 ± 1.5	32.2 ± 3.0	27.6 ± 4.6	17.6 ± 6.5	7.9 ± 3.1
	2	$36.7 \pm 1.3^*$	$36.0 \pm 1.0^*$	$32.8 \pm 0.7^*$	$27.6 \pm 1.7^*$	$26.3 \pm 7.6^*$	$24.0 \pm 6.3^*$	$13.2 \pm 5.4^*$	6.9 ± 2.1
Hopper Medium	0	68.8 ± 3.9	63.3 ± 3.3	57.6 ± 3.7	53.4 ± 3.8	56.7 ± 3.4	45.1 ± 2.8	35.0 ± 4.0	27.9 ± 5.7
	1 (chosen)	67.5 ± 5.2	64.0 ± 3.1	55.7 ± 3.1	50.5 ± 3.0	56.3 ± 4.7	43.9 ± 4.3	33.3 ± 3.1	26.7 ± 4.8
	2	66.4 ± 1.9	$60.4 \pm 1.4^*$	$52.6 \pm 1.7^*$	$46.4 \pm 2.9^*$	55.0 ± 2.2	$41.0 \pm 3.3^*$	33.2 ± 2.3	27.1 ± 3.2
Hopper Med-Replay	0	57.9 ± 14.4	57.3 ± 12.9	54.2 ± 8.9	47.0 ± 7.3	36.4 ± 12.8	26.6 ± 7.0	22.5 ± 3.2	18.3 ± 2.3
	1 (chosen)	58.3 ± 15.2	60.0 ± 14.4	59.5 ± 7.3	52.2 ± 5.1	31.8 ± 5.8	27.8 ± 4.9	24.5 ± 3.4	20.4 ± 2.8
	2	$71.6 \pm 7.9^*$	65.1 ± 11.4	64.3 ± 10.3	55.0 ± 7.5	35.6 ± 11.1	28.9 ± 7.0	27.5 ± 5.0	21.4 ± 2.8
Walker2d Medium	0	63.8 ± 9.0	59.1 ± 8.6	54.2 ± 7.5	51.1 ± 6.2	31.4 ± 3.7	20.3 ± 5.0	17.8 ± 3.4	$17.3 \pm 1.1^*$
	1 (chosen)	67.4 ± 5.5	65.1 ± 4.1	60.3 ± 4.9	56.2 ± 5.9	36.5 ± 13.1	25.5 ± 7.2	19.8 ± 3.0	19.5 ± 0.7
	2	68.2 ± 7.3	65.3 ± 4.8	60.8 ± 4.8	57.7 ± 5.1	37.9 ± 17.6	26.9 ± 10.3	20.5 ± 4.3	18.6 ± 2.9
Walker2d Med-Replay	0	61.6 ± 8.1	63.7 ± 2.6	61.9 ± 4.0	58.0 ± 3.1	17.1 ± 10.8	15.3 ± 5.8	$15.7 \pm 5.6^*$	13.3 ± 3.4
	1 (chosen)	66.8 ± 3.8	64.1 ± 3.4	61.9 ± 3.7	60.3 ± 2.0	22.3 ± 12.9	12.0 ± 4.4	12.0 ± 4.6	12.2 ± 3.2
	2	67.8 ± 4.0	62.7 ± 3.8	61.3 ± 2.9	59.9 ± 0.6	25.0 ± 8.2	12.5 ± 3.6	11.3 ± 3.8	10.6 ± 1.3
Average over tasks	0	55.4	53.7	50.2	45.7	35.3	28.4	22.6	18.0
	1 (chosen)	56.6	55.2	51.8	47.3	36.5	29.2	23.4	18.5
	2	58.8	55.2	51.9	47.1	36.6	28.6	23.1	18.4

Table 7: Effect of the number of Hopfield updates N_{hop} on all six tasks, with all other hyperparameters fixed at the chosen configuration. $N_{\text{hop}} = 0$ is standard softmax attention, and $N_{\text{hop}} = 1$ is the configuration used in the main results. Values are averaged over 6 training seeds, each evaluated on 500 episodes, and $\pm$ denotes the standard deviation across training seeds; the last block averages each column over the six tasks. * marks a mean difference from $N_{\text{hop}} = 1$ larger than two standard errors of the per-seed differences.

One update compared with none. Averaged over tasks, one Hopfield update improves the mean score from 45.7 to 47.3 and the CVaR-10 from 18.0 to 18.5 at $\sigma = 0.3$, and the clean mean score from 55.4 to 56.6. The gains under corruption come from Hopper Med-Replay ($+5.2$ in mean score), Walker2d Medium ($+5.1$ in mean score and $+2.2$ in CVaR-10), and Walker2d Med-Replay ($+2.3$ in mean score), while the two HalfCheetah tasks are unaffected and Hopper Medium scores higher without the update (53.4 versus 50.5).

Two updates compared with one. A second update does not improve the task-averaged results under corruption (47.1 versus 47.3 in mean score and 18.4 versus 18.5 in CVaR-10 at $\sigma = 0.3$), and it lowers the mean score at $\sigma = 0.3$ by more than two standard errors on Hopper Medium (-4.1) and HalfCheetah Med-Replay (-1.3). It raises the clean mean score on Hopper Med-Replay from 58.3 to 71.6, mostly through two training seeds whose clean scores increase from 30.7 to 61.1 and from 54.2 to 79.6. On the two tasks of Table 6, a third update lowers the mean score on HalfCheetah Medium by at most 0.4 points and changes the results on Hopper Med-Replay by amounts within seed noise. We therefore use $N_{\text{hop}} = 1$, which performs as well as two updates on average at a lower cost.

E Convergence of the Hopfield Update

Each Hopfield update decreases the energy of the modern Hopfield network, and iterating the update moves a query toward a fixed point. RHDT applies a single update ($N_{\text{hop}} = 1$). This section measures how far that update moves the queries, whether the energy decreases in practice, what the fixed points look like, and how the trained model responds when more updates are applied at evaluation. We analyze HalfCheetah Medium and Hopper Med-Replay.

E.1 Measurement

After a model's N Hopfield updates, we apply $\xi^{(N+1)} = K \operatorname{softmax}(\beta K^\top \xi^{(N)})$ to every query of every layer and head at every environment step, excluding padded positions. Its relative change $\|\xi^{(N+1)} - \xi^{(N)}\| / \|\xi^{(N)}\|$ is zero exactly when $\xi^{(N)}$ is a fixed point, so we use it as the residual of the update. A query has converged at tolerance ϵ if its residual is at most ϵ, and an episode has converged if all its queries have. We also record the Hopfield energy of every query before and after each update. To locate the fixed points themselves, at every 20th environment step of 10 episodes per training seed, we iterate the update on every query that can attend to more than its own key until its relative change is at most 10^{-6}, for at most 200 updates, and take the result as its fixed point ξ^*.

E.2 One Update Decreases the Energy but Does Not Converge

Table 8 reports the residual, the share of converged queries, and the energy for models trained with $N_{\text{hop}} \in \{1, 2, 3\}$.

Task	N_{hop}	Residual		Converged queries (%)		Energy at $\sigma = 0$		
		$\sigma = 0$	$\sigma = 0.3$	$\sigma = 0$	$\sigma = 0.3$	Before	After N_{hop}	After one more
HalfCheetah Medium	1 (chosen)	0.423	0.427	2.8	2.3	11.69	−19.09	−20.21
	2	0.246	0.249	6.6	5.8	9.57	−18.99	−19.32
	3	0.136	0.137	15.4	13.7	12.21	−19.10	−19.27
Hopper Med-Replay	1 (chosen)	0.395	0.398	1.9	1.9	−2.72	−22.02	−23.42
	2	0.226	0.230	3.2	3.0	−0.49	−21.89	−22.55
	3	0.133	0.141	7.1	5.6	1.99	−21.47	−21.74

Table 8: Convergence of the Hopfield updates for models trained with $N_{\text{hop}} \in \{1, 2, 3\}$. Each cell pools 6 training seeds with 500 episodes each.

The energy decreases with every update with the first one producing most of the decrease. On HalfCheetah Medium, the first update lowers the mean energy from 11.7 to -19.1, while one more update lowers it only to -20.2. At $\epsilon = 10^{-4}$, the share of converged queries is 1.7% on both tasks, which matches the share of queries that can attend only to their own key (one of the 60 tokens of a full context), for which the update returns that key at once. Training with more updates lowers the residual, to 0.25 for $N_{\text{hop}} = 2$ and 0.14 for $N_{\text{hop}} = 3$ on HalfCheetah Medium. The inverse temperature has little effect: across the models with β scaled from 0.5 to 5 times its default, the residual stays between 0.35 and 0.43 on both tasks. Corruption at $\sigma = 0.3$ changes the residual by less than 0.02 in every model.

E.3 Fixed Points

Figure 6(a) shows the relative distance of the queries from their fixed points after t updates for the models of the main results. At least 99.6% of the queries reach a fixed point within 200 updates, after a median of 7 updates on HalfCheetah Medium and 8 on Hopper Med-Replay at $\epsilon = 0.01$, with a 90th percentile of 16 on both tasks. The single update of RHDT reduces the distance to the fixed point by 71% on HalfCheetah Medium (from 3.89 to 1.14) and by 55% on Hopper Med-Replay (from 1.49 to 0.67), and ten updates reduce it to 0.13 and 0.08. Most fixed points combine several keys rather than retrieve one: at the fixed point, the largest attention weight averages 0.60 on HalfCheetah Medium and 0.50 on Hopper Med-Replay, and it reaches 0.9 for 52% and 22% of the queries. These mixtures correspond to the metastable states that modern Hopfield networks form at finite β. Models trained with more updates have fixed points that mix more keys, with the share of queries whose largest weight reaches 0.9 falling to 31% and 23% on HalfCheetah Medium for $N_{\text{hop}} = 2$ and $N_{\text{hop}} = 3$.

Corruption leaves the fixed points of HalfCheetah Medium nearly unchanged (50.7% with a largest weight of at least 0.9), while on Hopper Med-Replay it makes them more concentrated (36.5%, with a mean largest weight of 0.63).

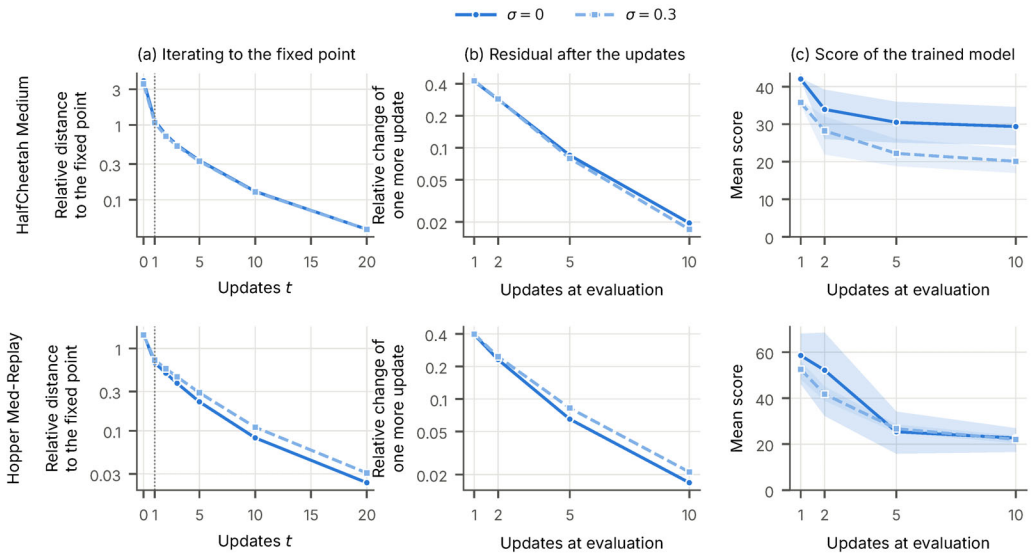

Figure 6: Convergence of the Hopfield update on HalfCheetah Medium (top) and Hopper Med-Replay (bottom). (a) Relative distance of the queries from their fixed points after t updates, for the models of the main results; the dotted line marks the single update of RHDT. (b) Residual after applying 1, 2, 5, or 10 updates at evaluation to the models trained with one update. (c) Mean score of these models, with 95% bootstrap confidence intervals over training seeds as shaded bands.

E.4 More Updates at Evaluation

Figure 6(b, c) applies 2, 5, or 10 updates at evaluation to the models trained with one update. More updates bring the queries closer to their fixed points: on HalfCheetah Medium, the residual falls from 0.42 after one update to 0.29, 0.085, and 0.020, and after ten updates 75% of the queries have converged at $\epsilon = 0.01$ (68% on Hopper Med-Replay), although no episode converges entirely. The score falls at the same time. On HalfCheetah Medium, it falls from 42.0 to 33.9, 30.5, and 29.4 without corruption and from 35.8 to 28.2, 22.2, and 20.1 at $\sigma = 0.3$. On Hopper Med-Replay, it falls from 58.5 to 52.2, 25.4, and 22.7 without corruption and from 52.5 to 41.7, 26.7, and 22.1 at $\sigma = 0.3$. The later layers are trained on the output of exactly one update, so moving the queries further changes their inputs away from what the network was trained on.

F Extended Analysis

This section provides additional analysis of the results in Table 1. Appendix F.1 plots performance against the corruption level, and Appendix F.2 shows the full distribution of episode scores. Appendix F.3 reports two location metrics that are less sensitive to individual episodes than the mean, the interquartile mean (IQM) and the median. Appendix F.4 separates the variability of returns within a training seed from the variability across training seeds. Appendix F.5 measures the fraction of its clean performance that each method retains under corruption. Appendix F.6 aggregates the results over tasks with bootstrap confidence intervals and reports the probability that RHDT outperforms each baseline, following Agarwal et al. [2021]. Appendix F.7 extends the evaluation to $\sigma = 0.5$ and $\sigma = 1.0$ on two tasks.

F.1 Performance Across Corruption Levels

Figure 7(a, b) plots the task-averaged mean score and CVaR-10 against the corruption level, and Figure 8 shows the same curves for each task. The shaded bands are 95% confidence intervals from a stratified bootstrap over training seeds (Appendix F.6), and the gray marker shows DT, which is evaluated only at $\sigma = 0$.

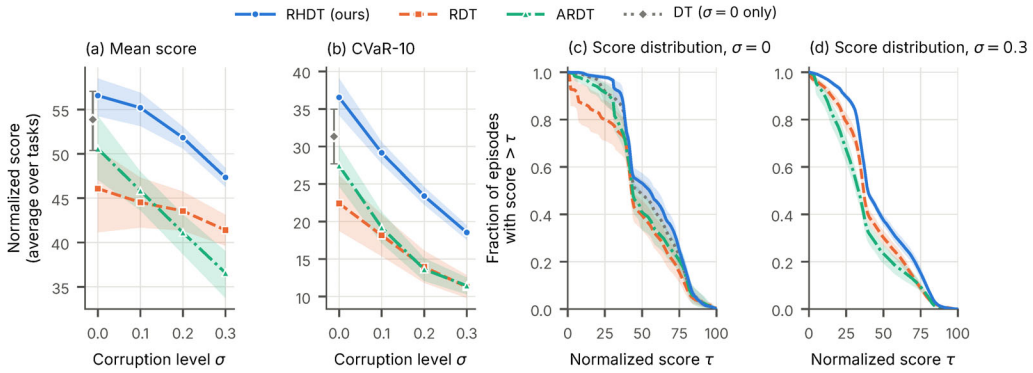

Figure 7: Task-averaged performance and score distributions. (a, b) Mean score and CVaR-10 averaged over the six tasks at each corruption level, with 95% stratified bootstrap confidence intervals over training seeds as shaded bands; the gray marker shows DT at $\sigma = 0$ with its interval. (c, d) Fraction of evaluation episodes with normalized score above τ, pooled over the six tasks and 6 training seeds (18,000 episodes per method), at $\sigma = 0$ and $\sigma = 0.3$, with 95% bootstrap bands.

Task-averaged curves. RHDT's task-averaged mean score and CVaR-10 lie above those of both baselines at every corruption level, and for $\sigma \geq 0.1$ its bands do not overlap theirs. The order of the two baselines changes with corruption: ARDT has the higher mean score at $\sigma = 0$ and $\sigma = 0.1$, but falls below RDT from $\sigma = 0.2$ onward (36.6 versus 41.4 at $\sigma = 0.3$), while their CVaR-10 values are within about one point of each other at every $\sigma \geq 0.1$. RDT's task-averaged mean score declines the least, from 46.1 to 41.4, partly because its clean score includes training runs that fail without corruption (Appendix F.5). RHDT declines from 56.6 to 47.3 and remains the highest at every level.

Per-task curves. The per-task curves show the same order on most tasks. On Hopper Med-Replay, RHDT's mean score stays nearly constant up to $\sigma = 0.2$ (58.3, 60.0, and 59.5), while both baselines drop to about 40 already at $\sigma = 0.1$. RDT's mean score is higher at $\sigma = 0.3$ than at $\sigma = 0$ on HalfCheetah Med-Replay and Walker2d Med-Replay, and its bands at $\sigma = 0$ on these tasks are wide. On Walker2d Med-Replay, RHDT's CVaR-10 drops from 22.3 at $\sigma = 0$ to 12.0 at $\sigma = 0.1$ and then stays near 12, and the three methods are within 0.4 points of each other at $\sigma = 0.1$.

F.2 Score Distributions

Figure 7(c, d) shows, for each threshold τ, the fraction of evaluation episodes whose normalized score exceeds τ, pooled over all tasks and training seeds. This follows the performance profiles

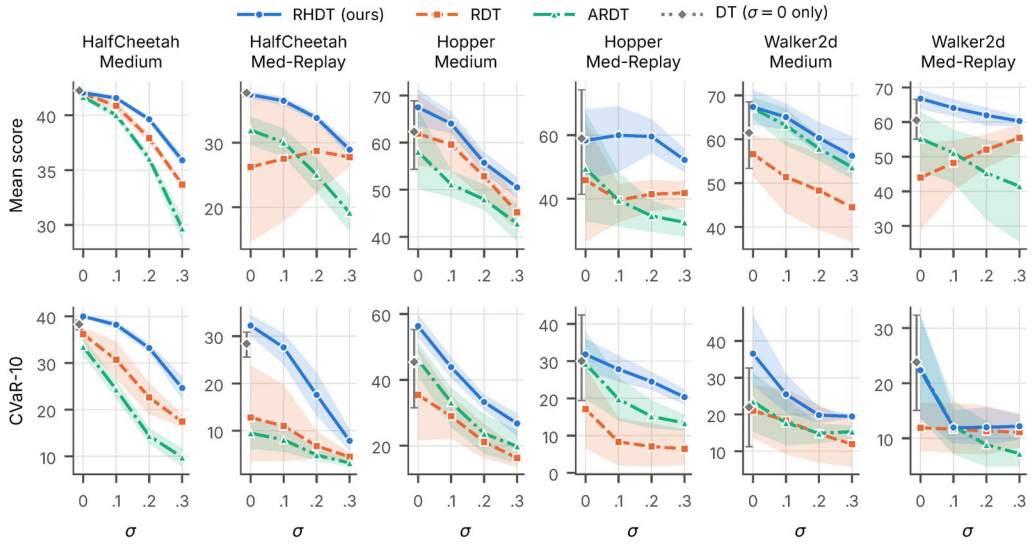

Figure 8: Mean score (top) and CVaR-10 (bottom) for each task as a function of the corruption level σ. Lines show the average over 6 training seeds, and shaded bands show 95% bootstrap confidence intervals over training seeds. The gray marker shows DT, which is evaluated only at $\sigma = 0$.

of Agarwal et al. [2021], computed over episodes instead of runs so that it shows the full return distribution of each method. A higher curve has more episodes above the threshold, the left part of each curve reflects the low-return episodes that determine CVaR-10, and the right part reflects the best episodes. The steep drop of every curve near $\tau = 40$ comes from the two HalfCheetah tasks, on which all methods score close to 40.

Without corruption. RHDT's curve lies above those of RDT, ARDT, and DT for every threshold below 85. Above 85, the curves differ by less than 0.015, since only between 4.3% and 5.7% of the episodes of each method score that high. The largest differences are at low thresholds: 13.2% of RDT's episodes score at most 10, against 2.8% for ARDT, 1.1% for DT, and 0.35% for RHDT.

Under corruption. At $\sigma = 0.3$, RHDT's curve lies at or above that of ARDT at every threshold, and above that of RDT for every threshold below 88, beyond which the two curves differ by less than 0.003. For example, 85.5% of RHDT's episodes score above 30, against 73.5% for RDT and 57.5% for ARDT. Only 2.2% of RHDT's episodes score at most 10, against 6.0% for RDT and 9.1% for ARDT. Under corruption, RHDT therefore shifts the return distribution upward, with the largest gains just below the median score and fewer episodes at the low scores that determine CVaR-10.

F.3 Interquartile Mean and Median

Table 9 reports the IQM and the median of the normalized scores for each task and corruption level. For each training seed, the IQM is the mean of the middle 50% of its 500 episode scores, so it discards the lowest and highest 25%, and the median is the score of the middle episode. As in Table 1, both metrics are averaged over the 6 training seeds, with $\pm$ denoting the standard deviation across training seeds, and the last block averages each column over the six tasks.

RHDT has the highest IQM and median in most settings. RHDT achieves the highest IQM in 22 of 24 task and corruption settings and the highest median in 21 of 24, counting ties. All exceptions occur at $\sigma \leq 0.1$. On HalfCheetah Medium at $\sigma = 0$, both baselines reach 42.4 in IQM and median against 42.1 for RHDT, a difference smaller than the across-seed standard deviation of RHDT (± 0.4). On Walker2d Medium, ARDT has a higher IQM at $\sigma = 0$ (74.4 versus 71.2) and a higher median at $\sigma = 0$ (77.2 versus 72.2) and $\sigma = 0.1$ (73.2 versus 71.3). At $\sigma \geq 0.2$, RHDT has the highest IQM and median on every task, with one tie in the median on HalfCheetah Med-Replay at $\sigma = 0.3$.

Task	Method	DT	IQM				Median			
			Corruption Level (σ)				Corruption Level (σ)			
		0.0	0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
HalfCheetah Medium	RDT	42.5 ± 0.1	$\mathbf{42.4 \pm 0.1}$	$\mathbf{41.8 \pm 0.1}$	39.6 ± 0.0	35.6 ± 0.2	$\mathbf{42.4 \pm 0.1}$	$\mathbf{41.8 \pm 0.1}$	39.7 ± 0.0	35.7 ± 0.1
	ARDT		$\mathbf{42.4 \pm 0.2}$	41.6 ± 0.1	39.1 ± 0.2	33.0 ± 1.4	$\mathbf{42.4 \pm 0.2}$	41.6 ± 0.1	39.1 ± 0.2	34.4 ± 0.6
	RHDT (ours)		42.1 ± 0.4	$\mathbf{41.8 \pm 0.3}$	$\mathbf{40.2 \pm 0.3}$	$\mathbf{37.1 \pm 0.4}$	42.1 ± 0.4	$\mathbf{41.8 \pm 0.4}$	$\mathbf{40.2 \pm 0.3}$	$\mathbf{37.2 \pm 0.5}$
	Δ_R / Δ_A		$-0.3/-0.3$	$+0.0/+0.2$	$+0.6/+1.1$	$+1.5/+4.1$	$-0.3/-0.3$	$+0.0/+0.2$	$+0.5/+1.1$	$+1.5/+2.8$
HalfCheetah Med-Replay	RDT	38.6 ± 0.6	27.8 ± 18.5	28.5 ± 16.8	30.5 ± 11.4	31.9 ± 2.1	27.8 ± 18.5	28.4 ± 16.9	29.3 ± 14.2	$\mathbf{33.0 \pm 1.4}$
	ARDT		34.9 ± 2.9	32.8 ± 3.2	27.2 ± 4.9	20.3 ± 5.3	35.4 ± 2.5	33.3 ± 3.0	27.8 ± 5.1	20.6 ± 5.8
	RHDT (ours)		$\mathbf{37.9 \pm 0.8}$	$\mathbf{37.3 \pm 0.9}$	$\mathbf{35.7 \pm 0.8}$	$\mathbf{32.4 \pm 0.7}$	$\mathbf{37.9 \pm 0.8}$	$\mathbf{37.4 \pm 0.9}$	$\mathbf{35.7 \pm 0.8}$	$\mathbf{33.0 \pm 0.6}$
	Δ_R / Δ_A		$+10.1/+3.0$	$+8.8/+4.5$	$+5.2/+8.5$	$+0.5/+12.1$	$+10.1/+2.5$	$+9.0/+4.1$	$+6.4/+7.9$	$+0.0/+12.4$
Hopper Medium	RDT	60.4 ± 8.5	61.4 ± 11.3	59.0 ± 8.3	52.0 ± 5.7	43.8 ± 4.2	61.5 ± 10.1	58.8 ± 8.1	51.9 ± 5.4	43.6 ± 3.7
	ARDT		55.8 ± 11.5	49.1 ± 4.1	46.5 ± 4.0	41.3 ± 5.3	55.1 ± 10.6	50.1 ± 3.7	46.9 ± 3.9	40.6 ± 4.5
	RHDT (ours)		$\mathbf{64.7 \pm 5.6}$	$\mathbf{62.5 \pm 3.2}$	$\mathbf{54.3 \pm 2.8}$	$\mathbf{48.8 \pm 2.4}$	$\mathbf{63.7 \pm 6.5}$	$\mathbf{62.4 \pm 2.6}$	$\mathbf{54.4 \pm 2.3}$	$\mathbf{48.4 \pm 2.3}$
	Δ_R / Δ_A		$+3.3/+8.9$	$+3.5/+13.4$	$+2.3/+7.8$	$+5.0/+7.5$	$+2.2/+8.6$	$+3.6/+12.3$	$+2.5/+7.5$	$+4.8/+7.8$
Hopper Med-Replay	RDT	61.2 ± 25.5	45.0 ± 29.3	35.5 ± 11.1	38.3 ± 10.2	39.4 ± 8.1	46.7 ± 29.5	33.0 ± 10.5	35.7 ± 11.2	37.4 ± 8.7
	ARDT		50.4 ± 29.1	36.4 ± 13.3	28.7 ± 6.0	28.1 ± 5.6	49.5 ± 29.7	31.8 ± 9.6	27.3 ± 5.7	27.2 ± 5.5
	RHDT (ours)		$\mathbf{58.7 \pm 17.4}$	$\mathbf{61.7 \pm 17.6}$	$\mathbf{60.6 \pm 10.1}$	$\mathbf{51.6 \pm 6.4}$	$\mathbf{58.7 \pm 19.7}$	$\mathbf{62.7 \pm 18.4}$	$\mathbf{60.7 \pm 10.7}$	$\mathbf{50.8 \pm 7.1}$
	Δ_R / Δ_A		$+13.7/+8.3$	$+26.2/+25.3$	$+22.3/+31.9$	$+12.2/+23.5$	$+12.0/+9.2$	$+29.7/+30.9$	$+25.0/+33.4$	$+13.4/+23.6$
Walker2d Medium	RDT	66.3 ± 13.2	59.7 ± 9.5	51.1 ± 15.4	47.0 ± 16.0	42.4 ± 13.7	61.1 ± 10.7	50.4 ± 18.0	44.9 ± 19.1	40.3 ± 15.0
	ARDT		$\mathbf{74.4 \pm 3.8}$	70.0 ± 7.3	62.8 ± 5.7	55.3 ± 5.7	$\mathbf{77.2 \pm 1.7}$	$\mathbf{73.2 \pm 9.0}$	64.9 ± 7.9	54.3 ± 5.7
	RHDT (ours)		71.2 ± 6.3	$\mathbf{70.1 \pm 5.8}$	$\mathbf{64.6 \pm 8.6}$	$\mathbf{58.1 \pm 9.9}$	72.2 ± 7.8	71.3 ± 7.0	$\mathbf{65.5 \pm 10.2}$	$\mathbf{58.7 \pm 11.9}$
	Δ_R / Δ_A		$+11.5/-3.2$	$+19.0/+0.1$	$+17.6/+1.8$	$+15.7/+2.8$	$+11.1/-5.0$	$+20.9/-1.9$	$+20.6/+0.6$	$+18.4/+4.4$
Walker2d Med-Replay	RDT	63.3 ± 9.0	46.8 ± 20.6	50.6 ± 15.2	56.1 ± 8.7	60.5 ± 7.7	48.6 ± 21.7	52.0 ± 16.8	57.7 ± 9.2	61.3 ± 8.1
	ARDT		55.8 ± 15.8	53.1 ± 17.8	47.4 ± 23.4	43.8 ± 24.1	56.4 ± 16.0	52.4 ± 21.9	49.9 ± 25.6	44.7 ± 25.5
	RHDT (ours)		$\mathbf{73.4 \pm 1.4}$	$\mathbf{72.8 \pm 2.2}$	$\mathbf{70.4 \pm 2.6}$	$\mathbf{68.0 \pm 1.8}$	$\mathbf{74.4 \pm 1.5}$	$\mathbf{73.7 \pm 2.4}$	$\mathbf{72.1 \pm 1.7}$	$\mathbf{70.3 \pm 1.4}$
	Δ_R / Δ_A		$+26.6/+17.6$	$+22.2/+19.7$	$+14.3/+23.0$	$+7.5/+24.2$	$+25.8/+18.0$	$+21.7/+21.3$	$+14.4/+22.2$	$+9.0/+25.6$
Average over tasks	RDT	55.4	47.2	44.4	43.9	42.3	48.0	44.1	43.2	41.9
	ARDT		52.3	47.2	42.0	37.0	52.6	47.1	42.7	37.0
	RHDT (ours)		**58.0**	**57.7**	**54.3**	**49.3**	**58.2**	**58.2**	**54.8**	**49.7**
	Δ_R / Δ_A		$+10.8/+5.7$	$+13.3/+10.5$	$+10.4/+12.3$	$+7.0/+12.3$	$+10.2/+5.6$	$+14.1/+11.1$	$+11.6/+12.1$	$+7.8/+12.7$

Table 9: Interquartile mean (IQM) and median of normalized scores under observation corruption. Both are computed per training seed over its 500 evaluation episodes and averaged over 6 training seeds; $\pm$ denotes the standard deviation across training seeds. The IQM is the mean of the middle 50% of episode scores. The DT column shows the IQM of DT at $\sigma = 0$. The last block averages each column over the six tasks. Δ_R / Δ_A rows show the difference between RHDT and RDT / ARDT, and bold indicates the best value among RDT, ARDT, and RHDT.

The gains hold for typical episodes. Averaged over tasks, RHDT's IQM at $\sigma = 0.3$ is 49.3, against 42.3 for RDT and 37.0 for ARDT, a gain of 7.0 and 12.3 points. The corresponding gains in median are 7.8 and 12.7 points. Both are larger than RHDT's gains in mean score at the same corruption level in Table 1, so its advantage under corruption is not driven by a small number of high-return episodes. Without corruption, RHDT's task-averaged IQM (58.0) is also above that of DT (55.4).

Typical and low-return episodes. Walker2d Medium shows how the metrics complement each other. Without corruption, ARDT's median exceeds its mean by 10.2 points (77.2 versus 67.0), while RHDT's median exceeds its mean by 4.8 points (72.2 versus 67.4). ARDT therefore has a better typical episode but more low-return episodes, which also appears in its lower CVaR-10 (23.5 versus 36.5 for RHDT in Table 1). On this task, RHDT trades a lower typical return for fewer low-return episodes, which gives it the higher mean score.

F.4 Within-Seed and Across-Seed Variability

Table 10 separates two sources of variability in the episode scores. The within-seed standard deviation measures how much the returns of a single trained policy vary from episode to episode, due to different initial states and, under corruption, different noise samples; it is the standard deviation of the 500 episode scores of a training seed, averaged over the 6 training seeds. The across-seed standard deviation measures how much the result changes when the model is trained with a different seed; it is the standard deviation of the 6 per-seed mean scores and equals the $\pm$ of the mean score in Table 1. The seed share combines the two: by the law of total variance, the variance of all 3,000 episode scores of a method on a task equals the average within-seed variance plus the variance of the per-seed means, and the seed share is the fraction contributed by the second term. A small seed share

means that a single training run is representative of the method, while a large seed share means that conclusions depend on which training seeds are used.

Task	Method	Within-Seed Std				Across-Seed Std				Seed Share (%)			
		0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
HalfCheetah Medium	DT	2.5				0.1				0.1			
	RDT	3.2	4.6	6.1	6.5	**0.3**	0.6	0.6	**0.4**	0.7	1.3	0.7	0.4
	ARDT	4.3	6.2	8.2	9.4	0.4	0.4	0.6	1.5	0.8	0.3	0.5	2.1
	RHDT (ours)	**1.3**	**2.2**	**3.4**	**4.7**	0.4	**0.3**	**0.4**	0.7	6.5	1.5	0.9	2.0
HalfCheetah Med-Replay	DT	4.4				0.7				1.9			
	RDT	7.9	9.3	10.4	10.6	16.4	13.2	8.0	1.9	73.4	58.8	31.5	2.6
	ARDT	9.4	9.6	9.8	9.1	3.0	3.0	3.7	3.7	7.6	7.4	10.4	12.1
	RHDT (ours)	**3.0**	**4.4**	**6.3**	**8.7**	**0.8**	**0.9**	**0.9**	**1.5**	4.3	3.1	1.4	2.3
Hopper Medium	DT	13.0				10.1				32.4			
	RDT	16.9	17.7	19.2	18.7	10.3	7.7	6.0	4.7	22.1	13.4	7.3	4.9
	ARDT	**10.5**	13.5	15.8	15.8	11.4	3.9	3.8	5.4	48.0	5.8	4.4	8.7
	RHDT (ours)	10.7	**12.9**	**14.6**	**15.7**	**5.2**	**3.1**	**3.1**	**3.0**	16.2	4.6	3.7	2.9
Hopper Med-Replay	DT	14.7				22.6				62.1			
	RDT	18.2	26.5	25.8	25.0	26.9	**8.7**	6.9	5.8	59.7	8.1	5.7	4.3
	ARDT	**13.5**	**17.8**	**18.9**	**17.3**	24.1	12.0	**6.7**	5.6	63.6	20.7	8.7	7.8
	RHDT (ours)	16.9	18.3	20.6	20.9	**15.2**	14.4	7.3	**5.1**	35.3	33.1	9.4	4.8
Walker2d Medium	DT	20.7				10.6				16.7			
	RDT	21.2	21.9	22.0	21.6	6.6	11.4	11.9	10.7	7.4	18.0	19.1	16.9
	ARDT	19.4	22.1	23.9	23.3	**3.4**	5.8	**3.9**	**3.5**	2.5	5.4	2.2	1.9
	RHDT (ours)	**15.6**	**19.2**	**21.6**	**21.4**	5.5	**4.1**	4.9	5.9	8.8	3.7	4.2	5.9
Walker2d Med-Replay	DT	19.5				8.1				12.3			
	RDT	**18.9**	**22.2**	**21.7**	21.2	18.4	10.3	6.5	5.5	41.1	15.1	6.8	5.3
	ARDT	19.3	23.3	23.5	**20.9**	11.7	11.1	16.2	19.9	21.9	15.6	28.1	40.8
	RHDT (ours)	19.5	22.8	23.1	22.5	**3.8**	**3.4**	**3.7**	**2.0**	2.9	1.8	2.0	0.7
Average over tasks	DT	12.5				8.7				20.9			
	RDT	14.4	17.0	17.5	17.3	13.2	8.7	6.6	4.9	34.1	19.1	11.8	5.7
	ARDT	12.7	15.4	16.7	16.0	9.0	6.0	5.8	6.6	24.1	9.2	9.0	12.2
	RHDT (ours)	**11.2**	**13.3**	**14.9**	**15.6**	**5.1**	**4.4**	**3.4**	**3.0**	12.3	7.9	3.6	3.1

Table 10: Within-seed and across-seed variability of normalized scores. Within-seed std is the standard deviation of the 500 episode scores of one training seed, averaged over the 6 training seeds. Across-seed std is the standard deviation of the 6 per-seed mean scores, as reported in Table 1. Seed share is the fraction of the variance of all 3,000 episode scores that is explained by differences between training seeds. DT is evaluated only at $\sigma = 0$, and the last block averages each column over the six tasks. Bold indicates the lowest standard deviation among RDT, ARDT, and RHDT.

RHDT varies least across training seeds. RHDT has the lowest task-averaged across-seed standard deviation at every corruption level (5.1, 4.4, 3.4, and 3.0 from $\sigma = 0$ to $\sigma = 0.3$, against 13.2, 8.7, 6.6, and 4.9 for RDT and 9.0, 6.0, 5.8, and 6.6 for ARDT). Its seed share is below 10% in 21 of 24 settings, against 13 for RDT and 15 for ARDT, and never exceeds 35.3%, while RDT reaches 73.4% and ARDT reaches 63.6%. For RHDT, most of the variability therefore comes from the episode-to-episode variation of a trained policy rather than from differences between training runs.

Where training seeds matter. Hopper Med-Replay without corruption has a large seed share for every method, 35.3% for RHDT and between 59.7% and 63.6% for DT, RDT, and ARDT, so results on this task depend most on the training seed. For RHDT, one of the six training seeds reaches a clean mean score of 30.7 on this task, while the other five reach between 54.2 and 72.6. RDT's largest seed shares, on HalfCheetah Med-Replay at $\sigma = 0$ (73.4%) and $\sigma = 0.1$ (58.8%) and on Walker2d Med-Replay at $\sigma = 0$ (41.1%), come from training runs that fail without corruption (Appendix F.5). These runs also appear in the IQM, where RDT reaches 27.8 ± 18.5 on HalfCheetah Med-Replay at $\sigma = 0$, against 37.9 ± 0.8 for RHDT (Table 9). ARDT's seed share on Walker2d Med-Replay is 40.8% at $\sigma = 0.3$, against 21.9% at $\sigma = 0$, so its training seeds respond differently to observation noise on this task.

Within-seed variability. The task-averaged within-seed standard deviation is higher at $\sigma = 0.3$ than at $\sigma = 0$ for every method, as observation noise adds variation between episodes. RHDT has the lowest task-averaged within-seed standard deviation at every corruption level (11.2 at $\sigma = 0$ and 15.6 at $\sigma = 0.3$, against 14.4 and 17.3 for RDT and 12.7 and 16.0 for ARDT) and the lowest value in 15 of 24 settings. On the HalfCheetah tasks without corruption, its within-seed standard deviation is less than half that of either baseline (1.3 versus 3.2 and 4.3 on Medium, and 3.0 versus 7.9 and 9.4 on Med-Replay). The exceptions are Hopper Med-Replay, where ARDT has the lowest value at

every corruption level, Walker2d Med-Replay, where RDT or ARDT is lower by at most 1.6 points, and Hopper Medium at $\sigma = 0$, where ARDT is lower by 0.2 points.

F.5 Score Retention Under Corruption

Table 11 reports the score of each method at each corruption level as a percentage of its own score without corruption, for the mean score and for CVaR-10. Retention separates the effect of corruption from the clean performance level: a method can rank first under corruption because it starts higher, because it degrades less, or both. Retention should be read together with the absolute scores, since a method with a low clean score can show a high retention, and a retention above 100% means that the method scores higher under corruption than without it.

Task	Method	Mean Score			CVaR-10		
		Corruption Level (σ)			Corruption Level (σ)		
		0.1	0.2	0.3	0.1	0.2	0.3
HalfCheetah Medium	RDT	97.3	90.3	80.2	84.8	62.6	48.1
	ARDT	96.1	86.5	71.3	72.6	42.9	29.2
	RHDT (ours)	98.8	94.2	85.4	95.5	83.1	61.6
HalfCheetah Med-Replay	RDT†	104.8	109.4	105.8	86.0	52.8	34.9
	ARDT	94.1	78.3	60.0	85.4	51.2	33.5
	RHDT (ours)	97.5	90.4	77.3	85.7	54.5	24.4
Hopper Medium	RDT	96.3	85.4	73.0	81.8	59.7	46.0
	ARDT	88.1	82.6	73.7	71.9	51.4	43.0
	RHDT (ours)	94.9	82.5	74.8	77.8	59.0	47.5
Hopper Med-Replay	RDT†	86.3	90.5	91.3	48.5	41.3	37.7
	ARDT	79.9	69.9	65.9	67.6	51.7	45.7
	RHDT (ours)	102.8	102.1	89.5	87.5	77.1	64.0
Walker2d Medium	RDT	90.8	85.3	78.6	87.1	70.0	56.7
	ARDT	94.2	86.4	80.0	75.3	63.4	65.3
	RHDT (ours)	96.6	89.5	83.5	69.8	54.3	53.2
Walker2d Med-Replay	RDT†	109.8	118.5	126.2	97.9	94.9	93.4
	ARDT	92.6	81.9	75.1	53.3	38.8	31.7
	RHDT (ours)	96.0	92.8	90.3	53.5	53.7	54.5
Average over tasks	RDT	96.6	94.5	89.8	81.0	62.2	50.3
	ARDT	90.6	81.3	72.3	70.0	49.7	41.7
	RHDT (ours)	97.6	91.6	83.7	79.8	64.0	50.7

Table 11: Score retention (%) under observation corruption, defined as the seed-averaged score at corruption level σ divided by the seed-averaged score at $\sigma = 0$. For the average over tasks, both scores are first averaged over the six tasks. Values above 100% indicate a higher score under corruption than without it. †RDT has one or two training seeds on this task whose mean score at $\sigma = 0$ is below 10, which lowers its clean score and raises its retention.

Retention compared with ARDT. At $\sigma = 0.3$, RHDT retains more of its mean score than ARDT on all six tasks, for example 85.4% versus 71.3% on HalfCheetah Medium and 89.5% versus 65.9% on Hopper Med-Replay, and 83.7% versus 72.3% on average. For CVaR-10 at $\sigma = 0.3$, RHDT retains more than ARDT on four of six tasks and on average (50.7% versus 41.7%), with HalfCheetah Med-Replay and Walker2d Medium as the exceptions. This is consistent with RHDT's CVaR-10 gain over ARDT increasing with corruption in Table 1.

Retention compared with RDT. Averaged over tasks, RHDT and RDT retain a similar fraction of their CVaR-10 at $\sigma = 0.3$ (50.7% and 50.3%), while RDT retains more of its mean score (89.8% versus 83.7%). RDT's average is raised by the three Med-Replay tasks, where one or two of its training seeds have clean mean scores between 3.4 and 7.2 but mean scores between 24.6 and 46.7 at $\sigma = 0.3$. These runs lower RDT's clean score, raise its retention, and explain its retention above 100% on HalfCheetah Med-Replay and Walker2d Med-Replay. On the three tasks without such runs, RHDT retains more of its mean score than RDT at $\sigma = 0.3$ (85.4% versus 80.2% on HalfCheetah Medium, 74.8% versus 73.0% on Hopper Medium, and 83.5% versus 78.6% on Walker2d Medium). Overall, RHDT's advantage over RDT under corruption comes mostly from a higher score at every corruption level, with a slower relative decline at $\sigma = 0.3$ on the tasks where all RDT training runs succeed, while its advantage over ARDT comes from both a higher score and a slower decline. RHDT's own retention exceeds 100% on Hopper Med-Replay at $\sigma = 0.1$ and 0.2 (102.8% and 102.1%), a small increase compared with the across-seed standard deviation of its clean mean score on this task (± 15.2).

Tail returns degrade faster than mean returns. For every method, the average CVaR-10 retention is lower than the average mean score retention at every corruption level. At $\sigma = 0.3$, the average CVaR-10 retention is between 41.7% and 50.7%, against between 72.3% and 89.8% for mean score.

F.6 Aggregate Performance and Probability of Improvement

Table 12 summarizes the results over all tasks with the interval estimates recommended by Agarwal et al. [2021]. We call one training seed on one task a run, which gives 36 runs per method and corruption level, and score each run by its mean score or CVaR-10 over 500 evaluation episodes. Part (a) reports the average over the 36 runs, which equals the task-averaged values used in the main results, and the IQM over runs, which discards the lowest and highest 25% of runs and is therefore less affected by a few failed or unusually good training runs. This IQM is taken over runs, whereas the IQM in Table 9 is taken over the episodes of each run. Part (b) reports the probability of improvement, the probability that a randomly chosen RHDT run outperforms a randomly chosen baseline run on the same task, averaged over tasks, where a value of 0.5 means that neither method tends to outperform the other. All intervals are 95% percentile intervals from a stratified bootstrap with 10,000 resamples, in which the training seeds of each method are resampled within each task.

Metric	Method	Corruption Level (σ)			
		0.0	0.1	0.2	0.3
(a) Aggregate scores over 36 runs (6 tasks × 6 training seeds) with 95% confidence intervals					
Mean Score (average)	DT	53.9 [50.4, 57.0]			
	RDT	46.1 [41.2, 50.5]	44.5 [41.7, 47.3]	43.6 [41.3, 45.7]	41.4 [39.6, 43.1]
	ARDT	50.6 [47.1, 54.3]	45.8 [43.6, 48.0]	41.1 [38.7, 43.2]	36.6 [33.7, 39.0]
	RHDT (ours)	**56.6 [54.3, 58.5]**	**55.2 [53.1, 56.9]**	**51.8 [50.6, 53.1]**	**47.3 [46.3, 48.4]**
Mean Score (IQM over runs)	DT	53.4 [48.6, 58.2]			
	RDT	47.6 [44.5, 50.6]	44.5 [42.1, 47.0]	43.4 [40.7, 46.1]	40.7 [38.2, 43.2]
	ARDT	50.0 [45.4, 54.7]	45.9 [42.4, 48.9]	41.2 [38.5, 43.8]	36.4 [33.4, 39.2]
	RHDT (ours)	**58.7 [55.6, 61.5]**	**57.4 [54.9, 59.6]**	**53.3 [51.9, 55.1]**	**48.9 [47.2, 50.5]**
CVaR-10 (average)	DT	31.3 [27.7, 35.0]			
	RDT	22.4 [18.8, 26.1]	18.2 [15.4, 20.9]	13.9 [11.8, 16.1]	11.3 [9.8, 12.8]
	ARDT	27.4 [24.7, 30.2]	19.2 [17.1, 21.3]	13.6 [12.1, 15.1]	11.4 [10.3, 12.5]
	RHDT (ours)	**36.5 [34.2, 39.1]**	**29.2 [27.7, 30.6]**	**23.4 [22.2, 24.6]**	**18.5 [17.6, 19.5]**
(b) Probability that a random RHDT run outperforms a random baseline run, with 95% confidence intervals					
RHDT vs RDT	Mean Score	0.75 [0.63, 0.87]	0.82 [0.71, 0.93]	0.86 [0.76, 0.94]	0.85 [0.75, 0.93]
	CVaR-10	0.84 [0.73, 0.93]	0.83 [0.74, 0.92]	0.84 [0.75, 0.92]	0.87 [0.77, 0.95]
RHDT vs ARDT	Mean Score	0.75 [0.63, 0.86]	0.92 [0.85, 0.97]	0.92 [0.85, 0.98]	0.92 [0.84, 0.98]
	CVaR-10	0.78 [0.67, 0.87]	0.82 [0.72, 0.91]	0.89 [0.80, 0.96]	0.94 [0.87, 0.99]

Table 12: Aggregate performance and probability of improvement, following Agarwal et al. [2021]. A run is one training seed on one task, scored by its mean score or CVaR-10 over 500 evaluation episodes. In (a), the average is taken over all 36 runs, and the IQM over runs is the mean of the middle 50% of the 36 run scores. In (b), the probability of improvement is computed on each task over all 6×6 pairs of RHDT and baseline runs, counting ties as one half, and then averaged over tasks. Confidence intervals are percentile intervals from a stratified bootstrap with 10,000 resamples, which resamples training seeds within each task and method. DT is evaluated only at $\sigma = 0$, and bold indicates the best value among RDT, ARDT, and RHDT.

Aggregate scores. Among the three robust methods, RHDT has the highest average mean score, IQM over runs, and average CVaR-10 at every corruption level. For $\sigma \geq 0.1$, its intervals for all three metrics lie above those of both baselines, for example 47.3 $[46.3, 48.4]$ in average mean score at $\sigma = 0.3$, against 41.4 $[39.6, 43.1]$ for RDT and 36.6 $[33.7, 39.0]$ for ARDT. Without corruption, RHDT's intervals also lie above those of both baselines for the IQM over runs and the average CVaR-10, while its interval for the average mean score meets that of ARDT at 54.3.

Probability of improvement. The probability that RHDT outperforms RDT is between 0.75 and 0.87 across metrics and corruption levels, and the probability that it outperforms ARDT is between 0.75 and 0.94. All lower confidence bounds are at least 0.63, so every interval excludes 0.5. Against ARDT, the probability is higher under corruption: for the mean score it rises from 0.75 at $\sigma = 0$ to 0.92 at every $\sigma \geq 0.1$, and for CVaR-10 it rises from 0.78 at $\sigma = 0$ to 0.94 at $\sigma = 0.3$.

F.7 Corruption Beyond the Main Range

Table 1 stops at $\sigma = 0.3$. To test the methods under stronger noise, we evaluate the RDT, ARDT, and RHDT models of Table 1 at $\sigma = 0.5$ and $\sigma = 1.0$ on HalfCheetah Medium and Hopper Med-Replay. At $\sigma = 1.0$, the noise added to each state dimension has the same standard deviation as that dimension in the dataset. Table 13 reports the results, and Figure 9 joins them with the levels of Table 1.

Task	Method	Mean Score		CVaR-10	
		$\sigma = 0.5$	$\sigma = 1.0$	$\sigma = 0.5$	$\sigma = 1.0$
HalfCheetah Medium	RDT	24.2 ± 0.6	12.0 ± 0.2	11.0 ± 0.9	6.9 ± 0.4
	ARDT	19.6 ± 2.2	11.3 ± 0.4	8.5 ± 1.7	6.9 ± 0.2
	RHDT (ours)	$\mathbf{26.0 \pm 0.9}$	$\mathbf{13.1 \pm 0.6}$	$\mathbf{12.4 \pm 1.2}$	$\mathbf{7.6 \pm 0.6}$
	Δ_{RDT}	$+1.8^*$	$+1.1^*$	$+1.4^*$	$+0.7^*$
	Δ_{ARDT}	$+6.4^*$	$+1.8^*$	$+3.9^*$	$+0.6^*$
Hopper Med-Replay	RDT	$\mathbf{31.9 \pm 5.3}$	$\mathbf{19.4 \pm 1.6}$	3.6 ± 4.2	3.5 ± 3.1
	ARDT	25.0 ± 4.1	16.1 ± 1.4	11.4 ± 1.7	7.6 ± 3.8
	RHDT (ours)	30.6 ± 3.0	$\mathbf{19.4 \pm 1.5}$	$\mathbf{13.0 \pm 0.9}$	$\mathbf{10.8 \pm 0.6}$
	Δ_{RDT}	-1.4	0.0	$+9.4^*$	$+7.3^*$
	Δ_{ARDT}	$+5.5^*$	$+3.3^*$	$+1.6^*$	$+3.1$

Table 13: Performance under Gaussian noise beyond the levels of Table 1, on the two tasks of the ablation study. Mean scores and CVaR-10 are computed per training seed over 500 evaluation episodes and averaged over 6 training seeds; $\pm$ denotes the standard deviation s across training seeds. Δ rows show RHDT minus each baseline, and * marks a difference larger than two standard errors, $2\sqrt{(s^2_{\text{RHDT}} + s^2_{\text{base}})/6}$. Bold marks the best method. All values are normalized scores (0=random, 100=expert).

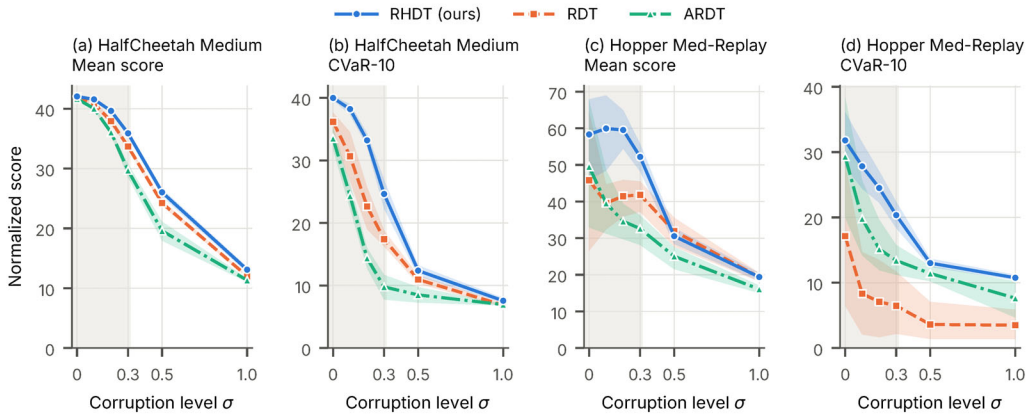

Figure 9: Mean score and CVaR-10 against the corruption level from $\sigma = 0$ to $\sigma = 1.0$ on HalfCheetah Medium and Hopper Med-Replay. The gray region marks the levels of Table 1 ($\sigma \leq 0.3$), and the colored bands are 95% bootstrap confidence intervals over training seeds.

On HalfCheetah Medium, RHDT has the highest mean score and CVaR-10 at both levels, beyond two standard errors against both baselines, with a mean score of 26.0 at $\sigma = 0.5$ against 24.2 for RDT and 19.6 for ARDT. The gaps narrow as all methods approach a common floor: the CVaR-10 lead over RDT, 10.6 points at $\sigma = 0.2$, falls to 1.4 points at $\sigma = 0.5$ and 0.7 points at $\sigma = 1.0$.

On Hopper Med-Replay, RHDT keeps the highest CVaR-10 at both levels (13.0 and 10.8, against 3.6 and 3.5 for RDT and 11.4 and 7.6 for ARDT), beyond two standard errors against RDT at both levels and against ARDT at $\sigma = 0.5$, and with the smallest standard deviation across training seeds. Its mean score, however, falls from 52.2 at $\sigma = 0.3$ to 30.6 at $\sigma = 0.5$, so its lead over RDT disappears: RDT is 1.4 points higher at $\sigma = 0.5$, within two standard errors, and the two are tied at 19.4 at $\sigma = 1.0$, while RHDT remains above ARDT by 5.5 and 3.3 points.

G Generalization Experiments

G.1 Overview

The main experiments (Section 5) cover six D4RL locomotion tasks that share dense rewards, low-dimensional states (11 to 17 dimensions), and data collected by RL-trained policies. This appendix tests whether the robustness of RHDT extends to settings that differ in state dimension, reward structure, data source, and observation type.

Ant locomotion (Section G.2) raises the state dimension to 111, with 8-dimensional actions, testing whether RHDT remains effective when the state dimension grows by about an order of magnitude. **Kitchen manipulation** (Section G.3) gives a reward only when one of four subtasks is completed, over episodes of about 280 steps, testing robustness under sparse, delayed feedback. **Adroit Pen-Human** (Section G.4) requires dexterous control of a 24-DoF hand from 25 human demonstrations recorded with motion capture, testing robustness with a small dataset of human rather than RL-trained behavior. **Atari Qbert** (Section G.5) replaces state vectors with image observations, where a convolutional encoder maps each frame to features on which Hopfield attention operates, giving a preliminary test of robustness to pixel noise on one game. Section G.6 summarizes the results across the four settings.

G.2 Ant Tasks: Generalizing to High-Dimensional State Spaces

G.2.1 Task Description

The Ant environment controls a quadruped whose four legs each have two actuated joints, a hip and an ankle, giving an 8-dimensional action space. Its observations have 111 dimensions, 27 joint positions and velocities and 84 external contact forces, compared with 11 to 17 dimensions for the main tasks. The quadruped must move forward while staying upright, with several legs in contact with the ground at once. We evaluate on ant-medium-v2 (1M transitions from a policy trained to about one third of expert performance) and ant-medium-replay-v2 (the replay buffer collected while training that policy).

G.2.2 Results

Table 14 reports performance on the Ant tasks under observation corruption.

Task	Method	Mean Score				CVaR-10			
		Corruption Level (σ)				Corruption Level (σ)			
		0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
Ant Medium	RDT	84.8 ± 23.2	82.5 ± 24.3	78.0 ± 25.3	64.9 ± 28.3	25.8	22.8	19.8	13.0
	ARDT	$\mathbf{91.5 \pm 20.8}$	$\mathbf{89.6 \pm 21.8}$	82.0 ± 25.0	71.8 ± 27.4	34.0	**31.7**	22.8	16.1
	RHDT (ours)	89.3 ± 19.9	86.1 ± 22.1	$\mathbf{82.2 \pm 22.9}$	$\mathbf{74.6 \pm 25.1}$	**35.6**	28.3	**24.6**	**18.6**
	Δ_{RDT}	$+4.5$	$+3.6$	$+4.1$	$+9.7$	$+9.8$	$+5.5$	$+4.8$	$+5.5$
	Δ_{ARDT}	-2.1	-3.4	$+0.2$	$+2.8$	$+1.6$	-3.3	$+1.8$	$+2.5$
Ant Med-Replay	RDT	80.1 ± 20.5	81.9 ± 18.6	75.5 ± 22.2	68.1 ± 23.2	30.5	33.6	21.7	17.3
	ARDT	67.5 ± 25.9	63.9 ± 25.0	59.2 ± 25.7	54.2 ± 23.3	18.4	21.0	16.4	16.9
	RHDT (ours)	$\mathbf{83.8 \pm 16.4}$	$\mathbf{82.3 \pm 17.0}$	$\mathbf{77.9 \pm 18.8}$	$\mathbf{71.6 \pm 19.9}$	**39.2**	**35.1**	**27.2**	**22.1**
	Δ_{RDT}	$+3.7$	$+0.4$	$+2.4$	$+3.5$	$+8.7$	$+1.5$	$+5.5$	$+4.8$
	Δ_{ARDT}	$+16.3$	$+18.4$	$+18.7$	$+17.4$	$+20.8$	$+14.1$	$+10.8$	$+5.1$

Table 14: Performance on D4RL Ant tasks under observation corruption, averaged over six training seeds. $\pm$ denotes the standard deviation across training seeds. Δ rows show RHDT minus each baseline, and bold marks the best method. All values are normalized scores (0=random, 100=expert).

On **Ant Med-Replay**, RHDT has the highest mean score and CVaR-10 at every corruption level. Its mean score exceeds that of RDT by 0.4 to 3.7 points and that of ARDT by 16.3 to 18.7 points. Its CVaR-10 exceeds that of RDT by 1.5 to 8.7 points and that of ARDT by 5.1 to 20.8 points; without corruption, it is 39.2 against 30.5 for RDT and 18.4 for ARDT. The CVaR-10 gap to ARDT narrows as corruption increases, from 20.8 points at $\sigma = 0$ to 5.1 points at $\sigma = 0.3$. RHDT also has the smallest $\pm$ values at every corruption level (16.4 to 19.9, against 18.6 to 23.2 for RDT and 23.3 to 25.9 for ARDT).

On **Ant Medium**, ARDT has the highest mean score without corruption (91.5, against 89.3 for RHDT and 84.8 for RDT) and at $\sigma = 0.1$, while RHDT has the highest mean score at $\sigma = 0.2$ and $\sigma = 0.3$. At $\sigma = 0.3$, the mean score of RHDT exceeds those of ARDT and RDT by 2.8 and 9.7 points. RHDT has the highest CVaR-10 at every level except $\sigma = 0.1$, where ARDT is higher (31.7 against 28.3), and its CVaR-10 exceeds that of RDT by 4.8 to 9.8 points at every level.

G.3 Kitchen Task: Generalizing to Sparse Rewards and Long Horizons

G.3.1 Task Description

The Kitchen environment controls a 9-DoF Franka robot arm that must complete four subtasks in sequence: opening the microwave, moving the kettle to a target location, flipping the light switch, and sliding open a cabinet door. Unlike the dense-reward locomotion tasks, Kitchen gives a reward of 1 only when a subtask is completed, over episodes of about 280 timesteps, so the policy must act for many steps without feedback. Every demonstration in the kitchen-complete-v0 dataset completes all four subtasks in order, so the task does not require stitching partial trajectories; it tests whether robustness holds under sparse, delayed rewards.

G.3.2 Results

Table 15 reports performance on Kitchen-Complete-v0 under observation corruption.

Method	Mean Score				Seed Median			
	Corruption Level (σ)				Corruption Level (σ)			
	0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
RDT	68.6 ± 21.6	65.1 ± 26.3	62.1 ± 23.7	55.2 ± 22.5	67.3	59.7	45.2	43.7
ARDT	82.0 ± 15.5	78.9 ± 18.3	69.2 ± 22.6	51.1 ± 26.3	77.1	77.1	77.0	42.6
RHDT (ours)	$\mathbf{88.9 \pm 21.6}$	$\mathbf{85.3 \pm 22.6}$	$\mathbf{74.5 \pm 25.3}$	$\mathbf{57.0 \pm 28.1}$	**77.4**	**77.3**	**77.2**	**59.8**
Δ_{RDT}	+20.3	+20.1	+12.4	+1.8	+10.1	+17.6	+31.9	+16.1
Δ_{ARDT}	+6.9	+6.3	+5.2	+5.9	+0.4	+0.2	+0.2	+17.1

Method	CVaR-10			
	Corruption Level (σ)			
	0.0	0.1	0.2	0.3
RDT	42.2	42.4	**41.7**	**27.1**
ARDT	**63.2**	**54.9**	31.2	0.0
RHDT (ours)	58.1	53.8	31.4	8.7
Δ_{RDT}	+16.0	+11.4	−10.3	−18.5
Δ_{ARDT}	−5.1	−1.1	+0.2	+8.7

Table 15: Performance on Kitchen-Complete-v0 under observation corruption, averaged over six training seeds. $\pm$ denotes the standard deviation across training seeds. Δ rows show RHDT minus each baseline, and bold marks the best method. All values are normalized scores (100 = all four subtasks completed). The seed median is the median across training seeds of each seed's mean score.

RHDT has the highest mean score at every corruption level: 88.9 without corruption, against 82.0 for ARDT and 68.6 for RDT, and 57.0 at $\sigma = 0.3$, against 51.1 and 55.2. Its lead over ARDT stays between 5.2 and 6.9 points, while its lead over RDT shrinks from 20.3 points at $\sigma = 0$ to 1.8 points at $\sigma = 0.3$.

Table 15 also reports the median across training seeds of each seed's mean score, which is less affected than the mean by a single outlying seed. RHDT and ARDT have similar medians up to $\sigma = 0.2$ (77.2 to 77.4 and 77.0 to 77.1), while the median of RDT falls from 67.3 to 45.2. At $\sigma = 0.3$, the median of ARDT drops to 42.6 and that of RDT to 43.7, while that of RHDT is 59.8.

CVaR-10 shows a different pattern. ARDT has the highest CVaR-10 at $\sigma = 0$ and $\sigma = 0.1$ (63.2 and 54.9, against 58.1 and 53.8 for RHDT), and RDT has the highest at $\sigma = 0.2$ and $\sigma = 0.3$ (41.7 and 27.1). At $\sigma = 0.3$, the CVaR-10 of RHDT falls to 8.7 and that of ARDT to 0.0, so the worst 10% of ARDT's episodes complete no subtask, while that of RDT falls less, from 42.2 to 27.1.

G.4 Adroit Pen-Human Task: Generalizing to Dexterous Manipulation with Human Demonstrations

G.4.1 Task Description

The Pen-Human task controls a 24-DoF simulated hand that must rotate a pen to match a randomly sampled target orientation. Its 45-dimensional observations contain the hand joint angles, the position, velocity, and orientation of the pen, and the target orientation, and its 24-dimensional actions control the individual joints. The reward is dense: it rewards alignment between the pen and the target orientation, adds large bonuses at timesteps where the two are close, and penalizes dropping the pen. Unlike the locomotion data, which come from RL-trained policies, the pen-human-v1 dataset contains only 25 human demonstrations recorded with motion capture. The task tests whether robustness holds for high-dimensional dexterous control learned from a small set of human demonstrations.

G.4.2 Results

Table 16 reports performance on Pen-Human-v1 under observation corruption. Because the scores vary widely across training seeds on this task, the table also reports the median across training seeds of each seed's mean score.

Method	Mean Score				Median Score			
	Corruption Level (σ)				Corruption Level (σ)			
	0.0	0.1	0.2	0.3	0.0	0.1	0.2	0.3
RDT	170.4 ± 144.6	167.0 ± 145.6	167.6 ± 145.1	167.4 ± 146.0	109.6	107.0	105.9	111.7
ARDT	176.4 ± 142.6	175.6 ± 142.9	176.3 ± 143.8	175.5 ± 142.4	141.4	142.5	139.8	141.0
RHDT (ours)	$\mathbf{181.3 \pm 143.5}$	$\mathbf{183.3 \pm 142.6}$	$\mathbf{181.5 \pm 142.8}$	$\mathbf{178.1 \pm 143.7}$	**157.5**	**173.5**	**158.1**	**144.5**
Δ_{RDT}	+10.9	+16.3	+13.9	+10.6	+47.9	+66.4	+52.2	+32.7
Δ_{ARDT}	+4.8	+7.7	+5.2	+2.6	+16.1	+30.9	+18.3	+3.5

Method	CVaR-10			
	Corruption Level (σ)			
	0.0	0.1	0.2	0.3
RDT	**3.8**	3.7	3.7	**3.5**
ARDT	3.4	3.3	3.5	**3.5**
RHDT (ours)	**3.8**	**3.8**	**3.8**	**3.5**
Δ_{RDT}	+0.0	+0.1	+0.1	+0.0
Δ_{ARDT}	+0.5	+0.5	+0.3	+0.0

Table 16: Performance on Pen-Human-v1 under observation corruption, averaged over six training seeds. $\pm$ denotes the standard deviation across training seeds. Δ rows show RHDT minus each baseline, and bold marks the best method. All values are normalized scores (0=random, 100=expert); scores above 100 exceed the D4RL expert reference. The seed median is the median across training seeds of each seed's mean score.

RHDT has the highest mean score at every corruption level, exceeding ARDT by 2.6 to 7.7 points and RDT by 10.6 to 16.3 points (181.3 against 176.4 and 170.4 without corruption). These differences are small relative to the $\pm$ values, which exceed 140 for every method. RHDT also has the highest median score at every level, exceeding that of RDT by 32.7 to 66.4 points and that of ARDT by 16.1 to 30.9 points up to $\sigma = 0.2$ and by 3.5 points at $\sigma = 0.3$. CVaR-10 lies between 3.3 and 3.8 for every method and corruption level, close to the score of a random policy.

G.5 Atari Qbert: A Preliminary Test with Pixel Observations

G.5.1 Task Description

We test whether the robustness of RHDT carries over to pixel observations on the Atari game Qbert. Following the Atari setup of DT [Chen et al., 2021], all three methods are trained on the same 1% subset of the DQN Replay dataset, take 4 stacked 84×84 grayscale frames as input, encode them with the convolutional encoder of DT, and use a context of 30 timesteps. In RHDT, Hopfield attention therefore operates on features learned from the frames instead of on embedded state vectors. Each method is trained with 6 training seeds and evaluated on 500 episodes per training seed, and scores are human-normalized (0 = random policy, 100 = human player). At test time, we add Gaussian noise

with standard deviation σ to the pixel intensities after scaling them to $[0, 1]$, so σ corresponds to 255σ gray levels, and we evaluate σ up to 0.1, about 26 gray levels.

G.5.2 Results

Table 17 reports performance on Qbert.

Method	Corruption Level (σ)					
	0	0.005	0.01	0.02	0.05	0.1
DT, reported by Chen et al. [2021]	15.4 ± 11.4					
CQL, reported by Chen et al. [2021]	104.2					
DT	6.3 ± 7.2	6.3 ± 6.9	5.7 ± 5.9	4.8 ± 4.8	2.2 ± 1.4	0.8 ± 0.3
RDT	13.6 ± 10.2	13.6 ± 9.9	13.5 ± 10.0	12.1 ± 8.7	6.5 ± 5.3	0.8 ± 0.6
RHDT (ours)	$\mathbf{68.7 \pm 3.2}$	$\mathbf{66.8 \pm 3.0}$	$\mathbf{65.4 \pm 3.6}$	$\mathbf{62.8 \pm 4.1}$	$\mathbf{41.7 \pm 7.6}$	$\mathbf{3.6 \pm 2.7}$
Δ_{DT}	$+62.4^*$	$+60.5^*$	$+59.7^*$	$+58.0^*$	$+39.5^*$	$+2.8^*$
Δ_{RDT}	$+55.1^*$	$+53.2^*$	$+51.9^*$	$+50.7^*$	$+35.2^*$	$+2.8^*$

Table 17: Performance on Atari Qbert under Gaussian pixel noise with standard deviation σ, with pixel intensities scaled to $[0, 1]$. Scores are human-normalized (0 = random policy, 100 = human player). DT, RDT, and RHDT are averaged over 6 training seeds with 500 evaluation episodes each, and $\pm$ denotes the standard deviation across training seeds. The first two rows are reported by Chen et al. [2021] without corruption; their DT result uses 3 training seeds with 20 evaluation episodes each. Δ rows show RHDT minus each baseline, * marks a difference larger than two standard errors, and bold marks the best of DT, RDT, and RHDT.

Without corruption, RHDT reaches 68.7, against 13.6 for RDT and 6.3 for DT under the same protocol. Chen et al. [2021] report 15.4 ± 11.4 for DT over 3 training seeds and 60 evaluation episodes, consistent with our estimate, and 104.2 for CQL on the same 1% subset. This dataset therefore supports scores well above those of DT, and the score of RHDT lies between those of DT and CQL. Under corruption, RHDT has the highest score at every level. At $\sigma = 0.02$ and $\sigma = 0.05$, it also retains the largest share of its clean score, 91% and 61%, against 89% and 48% for RDT and 76% and 35% for DT. At $\sigma = 0.1$, the scores of all three methods fall to 3.6 or below.

G.6 Summary

The additional experiments test RHDT on Ant, Kitchen, Pen-Human, and Atari Qbert, which differ from main locomotion tasks in state dimension, reward structure, data source, and observation type.

Higher-Dimensional States. On Ant (111 state dimensions), RHDT has the highest mean score and CVaR-10 on both datasets at $\sigma = 0.2$ and $\sigma = 0.3$, and on Ant Med-Replay at every corruption level. On Ant Medium, ARDT has the higher mean score without corruption and at $\sigma = 0.1$.

Sparse Rewards. On Kitchen, RHDT has the highest mean and median score at every corruption level, but its CVaR-10 falls below that of RDT at $\sigma = 0.2$ and $\sigma = 0.3$. Under sparse rewards, the worst episodes are those that complete few subtasks, so tail performance behaves differently.

Human Demonstrations. On Pen-Human, trained on 25 human demonstrations, RHDT has the highest mean and median score at every corruption level, with larger gaps in the median. CVaR-10 stays near the random-policy score for all methods.

Pixel Observations. On Atari Qbert, a single game with a convolutional encoder, RHDT scores 68.7 without corruption, against 13.6 for RDT and 6.3 for DT and below the 104.2 that CQL reaches on the same data, and it has the highest score at every pixel-noise level, keeping 61% of its clean score at $\sigma = 0.05$, against 48% for RDT and 35% for DT.

H Robustness to Alternative Observation Corruptions

Section 5 evaluates RHDT under Gaussian observation noise. To test whether its robustness extends to other corruptions, we evaluate two further types: observation dropout and impulse noise.

H.1 Corruption Models

H.1.1 Gaussian Noise (Main Paper)

Gaussian noise (Section 5.1) adds zero-mean noise to every state dimension at every timestep:

$$\tilde{s}_t = s_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \mathrm{diag}((\sigma \cdot \boldsymbol{\sigma}_s)^2)), \tag{25}$$

where σ sets the noise level and $\boldsymbol{\sigma}_s \in \mathbb{R}^{d_s}$ holds the per-dimension standard deviations of the training dataset.

H.1.2 Observation Dropout

Observation dropout sets each state dimension to zero independently with probability p at every timestep, modeling sensor failures:

$$\tilde{s}_t = s_t \odot m_t, \quad m_t \sim \text{Bernoulli}(1-p)^{d_s}, \tag{26}$$

where $\odot$ denotes element-wise multiplication and $m_t \in \{0,1\}^{d_s}$ is the mask. We evaluate $p \in \{0.1, 0.2, 0.3\}$. Dropout replaces values instead of perturbing them, so a masked value can only be inferred from the remaining dimensions and from earlier timesteps in the context. Because states are standardized with the dataset mean μ_i and standard deviation σ_i before entering the model, a masked dimension does not appear as missing: the model receives $-\mu_i/\sigma_i$, a fixed offset that is large for dimensions whose mean is large relative to their spread.

H.1.3 Impulse Noise

Impulse noise adds a large Gaussian perturbation to the whole state at a random subset of timesteps, modeling sensor glitches:

$$\tilde{s}_t = \begin{cases} s_t + \eta_t, & \text{with probability } p, \\ s_t, & \text{with probability } 1-p, \end{cases} \qquad \eta_t \sim \mathcal{N}(0, \mathrm{diag}((\alpha\boldsymbol{\sigma}_s)^2)), \tag{27}$$

with $\alpha = 5$. We evaluate $p \in \{0.05, 0.1, 0.15\}$. Most timesteps stay clean, but a corrupted timestep is perturbed with 5 times the per-dimension standard deviation, compared with at most 0.3 times under the Gaussian noise of Section 5.1.

H.2 Observation Dropout Results

Table 18 reports the results under observation dropout. At $p = 0.3$, the task-averaged mean score falls to 10.9 for RHDT and 11.8 for RDT. At $p = 0.1$, RHDT has the higher mean score on five of six tasks, and its task-averaged mean score and CVaR-10 exceed those of RDT by 2.7 and 2.4 points. At $p = 0.2$ and $p = 0.3$, the two methods are close on average, with task-averaged differences of at most 1.2 points in mean score and 0.3 points in CVaR-10.

H.3 Impulse Noise Results

Table 19 reports the results under impulse noise. Averaged over tasks, RHDT's CVaR-10 is higher by 7.2, 6.4, and 5.3 points at $p = 0.05$, 0.1, and 0.15, and its mean score by 6.6, 4.3, and 2.7 points. The gap in mean score narrows as impulses become more frequent while gap in tail performance remains.

Task	Method	Mean Score			CVaR-10		
		Dropout Rate (p)			Dropout Rate (p)		
		0.1	0.2	0.3	0.1	0.2	0.3
HalfCheetah Medium	RDT	32.9 ± 0.6	$\mathbf{25.5 \pm 0.7}$	$\mathbf{20.4 \pm 0.5}$	$\mathbf{16.1 \pm 2.7}$	$\mathbf{10.4 \pm 1.1}$	$\mathbf{8.3 \pm 1.5}$
	RHDT (ours)	$\mathbf{33.2 \pm 0.9}$	25.3 ± 1.9	19.6 ± 2.0	15.3 ± 2.5	8.8 ± 1.5	6.7 ± 1.7
	Δ_{RDT}	$+0.3$	-0.2	-0.8	-0.8	-1.7^*	-1.6
HalfCheetah Med-Replay	RDT	20.6 ± 13.8	15.0 ± 10.1	8.4 ± 6.9	5.1 ± 3.0	3.6 ± 1.5	2.7 ± 0.8
	RHDT (ours)	$\mathbf{27.1 \pm 2.0}$	$\mathbf{20.0 \pm 2.9}$	$\mathbf{11.7 \pm 4.2}$	$\mathbf{6.8 \pm 2.4}$	$\mathbf{5.2 \pm 1.8}$	$\mathbf{4.3 \pm 1.6}$
	Δ_{RDT}	$+6.5$	$+5.0$	$+3.2$	$+1.7$	$+1.6$	$+1.6^*$
Hopper Medium	RDT	30.0 ± 2.7	21.1 ± 2.1	$\mathbf{15.0 \pm 1.6}$	11.5 ± 0.8	$\mathbf{9.3 \pm 1.3}$	$\mathbf{7.0 \pm 3.0}$
	RHDT (ours)	$\mathbf{34.7 \pm 3.5}$	$\mathbf{21.5 \pm 1.5}$	$\mathbf{15.0 \pm 1.3}$	$\mathbf{15.3 \pm 4.0}$	8.3 ± 2.9	5.6 ± 2.9
	Δ_{RDT}	$+4.8^*$	$+0.4$	-0.1	$+3.8^*$	-1.0	-1.5
Hopper Med-Replay	RDT	$\mathbf{30.7 \pm 6.3}$	$\mathbf{21.6 \pm 6.9}$	$\mathbf{10.6 \pm 4.2}$	9.5 ± 6.4	$\mathbf{6.4 \pm 2.8}$	$\mathbf{2.6 \pm 0.7}$
	RHDT (ours)	24.1 ± 8.2	8.7 ± 4.6	3.9 ± 1.9	$\mathbf{10.6 \pm 4.4}$	3.4 ± 1.5	1.7 ± 0.9
	Δ_{RDT}	-6.5	-12.9^*	-6.7^*	$+1.0$	-3.0^*	-0.9
Walker2d Medium	RDT	40.9 ± 2.8	21.8 ± 2.9	$\mathbf{9.3 \pm 1.1}$	9.5 ± 4.3	3.5 ± 1.6	0.0 ± 0.3
	RHDT (ours)	$\mathbf{50.2 \pm 6.5}$	$\mathbf{23.9 \pm 4.5}$	9.1 ± 1.3	$\mathbf{17.0 \pm 1.7}$	$\mathbf{6.3 \pm 1.9}$	$\mathbf{0.4 \pm 0.2}$
	Δ_{RDT}	$+9.3^*$	$+2.1$	-0.2	$+7.5^*$	$+2.8^*$	$+0.4^*$
Walker2d Med-Replay	RDT	39.9 ± 5.6	$\mathbf{15.5 \pm 1.6}$	$\mathbf{7.1 \pm 1.4}$	5.8 ± 4.1	1.2 ± 1.8	0.3 ± 0.9
	RHDT (ours)	$\mathbf{41.6 \pm 3.2}$	13.8 ± 2.4	5.9 ± 3.1	$\mathbf{7.1 \pm 3.3}$	$\mathbf{2.0 \pm 3.0}$	$\mathbf{0.7 \pm 2.5}$
	Δ_{RDT}	$+1.7$	-1.7	-1.2	$+1.3$	$+0.8$	$+0.4$

Table 18: Performance under observation dropout, where each state dimension is set to zero with probability p at every timestep. Mean scores and CVaR-10 are computed per training seed over 500 evaluation episodes and averaged over 6 training seeds; $\pm$ denotes the standard deviation s across training seeds. Δ_{RDT} is RHDT minus RDT, and * marks a difference larger than two standard errors. Bold marks the higher of the two methods. All values are normalized scores (0=random, 100=expert).

Task	Method	Mean Score			CVaR-10		
		Impulse Probability (p)			Impulse Probability (p)		
		0.05	0.1	0.15	0.05	0.1	0.15
HalfCheetah Medium	RDT	38.0 ± 0.2	34.3 ± 0.2	30.6 ± 0.2	26.2 ± 1.4	19.4 ± 1.4	13.6 ± 1.8
	RHDT (ours)	$\mathbf{38.1 \pm 0.5}$	$\mathbf{34.8 \pm 0.8}$	$\mathbf{31.4 \pm 0.9}$	$\mathbf{28.5 \pm 1.5}$	$\mathbf{24.0 \pm 1.4}$	$\mathbf{19.2 \pm 1.4}$
	Δ_{RDT}	$+0.1$	$+0.5$	$+0.8$	$+2.3^*$	$+4.7^*$	$+5.6^*$
HalfCheetah Med-Replay	RDT	29.7 ± 5.0	25.5 ± 3.2	21.5 ± 3.1	7.8 ± 2.8	4.9 ± 1.8	4.4 ± 2.4
	RHDT (ours)	$\mathbf{32.6 \pm 0.6}$	$\mathbf{28.8 \pm 0.6}$	$\mathbf{25.0 \pm 1.1}$	$\mathbf{16.7 \pm 4.6}$	$\mathbf{12.4 \pm 3.9}$	$\mathbf{9.6 \pm 4.2}$
	Δ_{RDT}	$+2.9$	$+3.2^*$	$+3.5^*$	$+8.9^*$	$+7.5^*$	$+5.3^*$
Hopper Medium	RDT	47.0 ± 3.9	39.4 ± 2.4	34.9 ± 2.4	22.5 ± 5.6	17.9 ± 3.8	15.3 ± 2.4
	RHDT (ours)	$\mathbf{51.7 \pm 1.9}$	$\mathbf{43.7 \pm 1.6}$	$\mathbf{39.5 \pm 1.4}$	$\mathbf{31.3 \pm 3.4}$	$\mathbf{25.9 \pm 3.5}$	$\mathbf{21.6 \pm 2.9}$
	Δ_{RDT}	$+4.7^*$	$+4.3^*$	$+4.7^*$	$+8.8^*$	$+8.1^*$	$+6.3^*$
Hopper Med-Replay	RDT	37.0 ± 12.6	34.6 ± 7.5	33.0 ± 4.7	9.1 ± 7.4	6.9 ± 6.3	6.2 ± 6.1
	RHDT (ours)	$\mathbf{51.9 \pm 11.1}$	$\mathbf{44.7 \pm 8.2}$	$\mathbf{39.6 \pm 6.3}$	$\mathbf{23.3 \pm 2.8}$	$\mathbf{20.0 \pm 2.7}$	$\mathbf{17.7 \pm 3.3}$
	Δ_{RDT}	$+14.9^*$	$+10.1^*$	$+6.6^*$	$+14.2^*$	$+13.1^*$	$+11.5^*$
Walker2d Medium	RDT	48.2 ± 5.7	41.8 ± 3.3	$\mathbf{35.3 \pm 2.8}$	16.1 ± 8.5	14.7 ± 7.0	12.2 ± 4.8
	RHDT (ours)	$\mathbf{55.8 \pm 6.5}$	$\mathbf{45.3 \pm 4.8}$	34.9 ± 3.9	$\mathbf{23.9 \pm 3.7}$	$\mathbf{18.6 \pm 1.4}$	$\mathbf{14.6 \pm 2.0}$
	Δ_{RDT}	$+7.6^*$	$+3.4$	-0.4	$+7.9^*$	$+3.9$	$+2.3$
Walker2d Med-Replay	RDT	48.2 ± 12.7	45.3 ± 9.7	37.9 ± 7.7	11.2 ± 5.5	9.5 ± 3.8	8.2 ± 2.4
	RHDT (ours)	$\mathbf{57.7 \pm 3.1}$	$\mathbf{49.6 \pm 5.7}$	$\mathbf{39.1 \pm 5.7}$	$\mathbf{12.7 \pm 2.6}$	$\mathbf{10.9 \pm 1.6}$	$\mathbf{9.2 \pm 1.3}$
	Δ_{RDT}	$+9.6$	$+4.3$	$+1.3$	$+1.4$	$+1.4$	$+1.0$

Table 19: Performance under impulse noise, where with probability p at every timestep the state receives Gaussian noise with 5 times the per-dimension standard deviation. Metrics, $\pm$, Δ_{RDT}, *, and bold are as in Table 18. All values are normalized scores (0=random, 100=expert).