TOWARDS GENERAL-PURPOSE MODEL-FREE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) promises a framework for near-universal problem-solving. In practice however, RL algorithms are often tailored to specific benchmarks, relying on carefully tuned hyperparameters and algorithmic choices. Recently, powerful model-based RL methods have shown impressive general results across benchmarks but come at the cost of increased complexity and slow run times, limiting their broader applicability. In this paper, we attempt to find a unifying model-free deep RL algorithm that can address a diverse class of domains and problem settings. To achieve this, we leverage model-based representations that approximately linearize the value function, taking advantage of the denser task objectives used by model-based RL while avoiding the costs associated with planning or simulated trajectories. We evaluate our algorithm, MR.Q, on a variety of common RL benchmarks with a single set of hyperparameters and show a competitive performance against domain-specific and general baselines, providing a concrete step towards building general-purpose model-free deep RL algorithms.

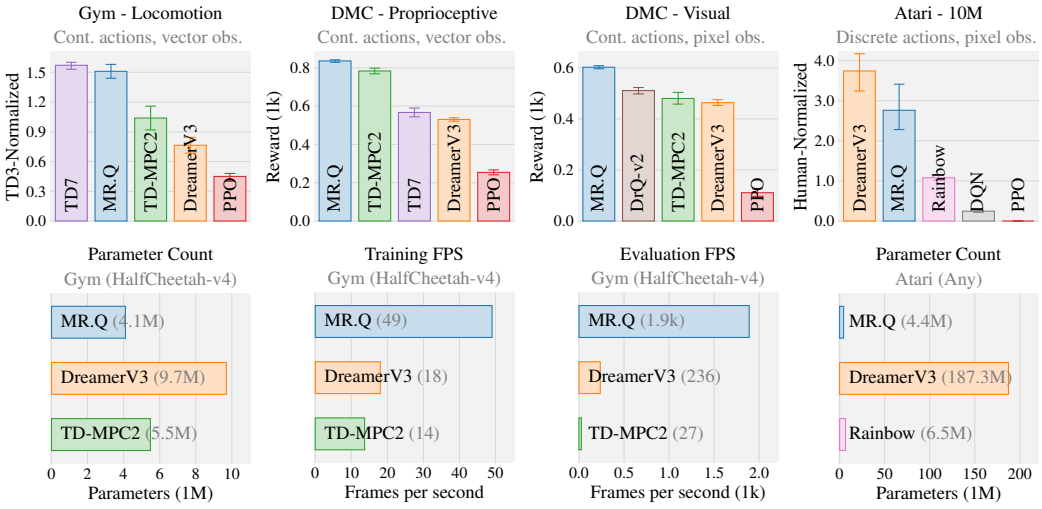

Figure 1: **Summary of results.** Aggregate mean performance across four common RL benchmarks and 118 environments featuring diverse characteristics (e.g., observation and action spaces, task types). Error bars capture a 95% stratified bootstrap confidence interval. Our algorithm, MR.Q, achieves a competitive performance against both state-of-the-art domain-specific and general baselines, while using a single set of hyperparameters. Notably, MR.Q accomplishes this with fewer network parameters and substantially faster training and evaluation speeds than general-purpose model-based methods.

1 INTRODUCTION

The conceptual premise of RL is inherently general-purpose—an RL agent can learn optimal behavior with only two basic elements: a well-defined objective and data describing its interactions with the environment. In reality, however, most RL algorithms are anything but general-purpose. Instead, RL algorithms are highly specialized and typically characterized by specific problem classes, such

as discrete versus continuous actions or vector versus pixel observations, with each category requiring its own set of algorithmic choices and hyperparameters. For example, Rainbow and TD3 (Hessel et al., 2018; Fujimoto et al., 2018), common methods for Atari and MuJoCo respectively (Bellemare et al., 2013; Todorov et al., 2012), have more differences than similarities in their shared hyperparameters (Table 1)—without accounting for further algorithmic differences.

Table 1: Hyperparameter differences between Rainbow (Hessel et al., 2018) and TD3 (Fujimoto et al., 2018). TD3 uses an expected moving average (EMA) update with an effective frequency of $\frac{1}{1-0.995} = 200$.

Hyperparameter	Rainbow	TD3
Discount factor	0.99	0.99
Optimizer	Adam	Adam
Learning Rate	$6.25 \cdot 10^{-5}$	10^{-3}
Adam ϵ	$1.5 \cdot 10^{-4}$	10^{-8}
Replay buffer size	1M	1M
Minibatch size	32	100
Target network update	Iterative	EMA
Effective target update freq.	8k	200
Initial random steps	20k	1k

To some extent, general-purpose algorithms do exist—policy gradient methods (Williams, 1992; Schulman et al., 2015; 2017) and many evolutionary approaches (Rechenberg, 1978; Back, 1996; Rubinstein, 1997; Salimans et al., 2017) require few assumptions on the underlying problem. Unfortunately, these methods often offer poor sample efficiency and asymptotic performance compared more domain-specific approaches, and in some instances, can require extensive re-tuning over numerous implementation-level details (Engstrom et al., 2020; Huang et al., 2022).

Recently, DreamerV3 (Hafner et al., 2023) and TD-MPC2 (Hansen et al., 2024), have showcased the potential of general-purpose model-based approaches, achieving impressive single-task performance on a diverse set of benchmarks without re-tuning hyperparameters. However, despite their success, model-based methods also introduce substantial algorithmic and computational complexity, making them less practical than lightweight domain-specific model-free algorithms.

This paper presents a general model-free RL algorithm that leverages model-based representations to achieve the sample efficiency and performance of model-based methods, without the computational overhead. A recent surge of high-performing model-free RL algorithms with dynamics-based representations (Guo et al., 2020; 2022; Schwarzer et al., 2020; 2023; Zhao et al., 2023; Fujimoto et al., 2024; Zheng et al., 2024; Scannell et al., 2024) has showcased the potential of this family of algorithms when tailored for a single benchmark. Recognizing the similarity between these model-based and model-free approaches, our hypothesis is that the true benefit of model-based objectives is in the implicitly learned representation, rather than the model itself, and thus prompting the question:

Can model-based representations alone enable sample-efficient general-purpose learning?

Our proposed approach is based on learning features that approximately capture a linear relationship between state-action pairs and value. To do so, we draw heavily from modern dynamics-based representation learning methods (see Related Work) as well as the work of Parr et al. (2008), who show that both model-based and model-free objectives converge to the same solution in linear space. By mapping states and actions into a single, unified embedding, we eliminate any environment-specific characteristics of the input space and allow for a standardized set of hyperparameters.

We evaluate our method, MR.Q, on four widely used RL benchmarks and 118 environments, and achieve competitive performance against state-of-the-art domain-specific and general baselines without algorithmic or hyperparameter changes between environments or benchmarks. Code will be open-sourced with the camera-ready version.

2 Related Work

General-purpose RL. Although many traditional RL methods are general-purpose in principle, practical constraints often force assumptions about the task domain. For example, algorithms like Q-learning and SARSA (Watkins, 1989; Rummery & Niranjan, 1994) can be conceptually extended to continuous spaces, but are typically implemented using discrete lookup tables. In practice, early examples of general decision-making approaches can be found in on-policy methods with function approximation. For instance, both evolutionary algorithms (Rechenberg, 1978; Back, 1996; Rubinstein, 1997; Salimans et al., 2017) and policy gradient methods (Williams, 1992; Sutton et al., 1999;

Schulman et al., 2015; 2017) offer update rules with convergence guarantees and independence to the input space. However, despite their generality, these methods are also hindered by poor sample efficiency and are prone to local minima, limiting their suitability for many practical applications.

In contrast, the design of deep RL algorithms tends to favor more specialized approaches that align closely with a single benchmark—e.g., DQN↔Atari (Bellemare et al., 2013; Mnih et al., 2015), DDPG↔MuJoCo (Todorov et al., 2012; Lillicrap et al., 2015), or AlphaGo↔Go (Silver et al., 2016). Generalizing beyond these initial benchmarks can often require significant engineering, tuning, or algorithmic discovery (Luong et al., 2019; Schrittwieser et al., 2020; Haydari & Yılmaz, 2020; Ibarz et al., 2021). In imitation learning, GATO achieved generalist behavior, but relied on large expert datasets (Reed et al., 2022). Recently, DreamerV3 (Hafner et al., 2023) demonstrated a strong capability over many benchmarks without re-tuning, but used costly large models and simulated rollouts. Our objective is to discover a lightweight model-free approach to general-purpose learning.

Dynamics-based representation learning. Building representations from system dynamics is a long-standing approach for adaptation, partial observability, and feature selection (Dayan, 1993; Littman & Sutton, 2001; Parr et al., 2008). Numerous model-free methods have been developed to learn representations by predicting future latent states (Munk et al., 2016; Van Hoof et al., 2016; Zhang et al., 2018; Gelada et al., 2019; Lee et al., 2020; Guo et al., 2020; 2022; Schwarzer et al., 2020; 2023; Zintgraf et al., 2021; Yu et al., 2021; 2022; Fujimoto et al., 2021; 2024; McInroe et al., 2021; Seo et al., 2022; Kim et al., 2022; Tang et al., 2023; Zhao et al., 2023; Zheng et al., 2024; Ni et al., 2024; Scannell et al., 2024). Unsurprisingly, these model-free approaches relate closely to model-based counterparts which learn a latent dynamics model for planning or value estimation (Watter et al., 2015; Finn et al., 2016; Karl et al., 2017; Ha & Schmidhuber, 2018; Schrittwieser et al., 2020; 2021; Ye et al., 2021; Hansen et al., 2022; 2024; Hafner et al., 2019; 2023; Wang et al., 2024). Our approach, MR.Q, is most closely related to the state-action representation learning in TD7 (Fujimoto et al., 2024). At a high level, MR.Q differs from TD7 by discarding the original input and including losses over the reward and termination. MR.Q also differs significantly in implementation, drawing inspiration from prior work to determine a set of design choices that performs well across benchmarks, including multi-step returns, unrolled dynamics, and categorical losses.

State abstraction. Our work is closely related to bisimulation metrics (Ferns et al., 2004; 2011; Castro, 2020) and MDP homomorphisms (Ravindran, 2004; van der Pol et al., 2020a;b; Rezaei-Shoshtari et al., 2022) which rely on measures of similarity in reward and dynamics for state or action abstraction. These concepts have inspired practical approximations to bisimulation metrics as a means of shaping representations in deep RL agents, particularly those using image-based observations (Zhang et al., 2020; Castro et al., 2021; Zang et al., 2022). Our research also connects to two-stage linear RL, where a non-linear function learns an embedding that is used by linear RL (Levine et al., 2017; Chung et al., 2019). Under this setting, Ghosh & Bellemare (2020) studied the representations learned by Petrik (2007) and Parr et al. (2008) and showed a connection between the capacity of a learned representation to represent the reward and dynamics and off-policy stability.

3 Background

Reinforcement learning (RL) problems are described by a Markov Decision Process (MDP) (Bellman, 1957), which we define by a tuple (S, A, p, R, γ) of state space S, action space A, dynamics function p, reward function R and discount factor γ. Value-based RL methods learn a value function $Q^\pi(s,a) := \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$ that models the expected discounted sum of rewards $r_t \sim R(s_t, a_t)$ by following a policy π which maps states s to actions a.

The true value function Q^π is estimated by an approximate value function Q_θ. We use subscripts to indicate the network parameters θ. Target networks, which are used to introduce stationarity in prediction targets, have parameters denoted by an apostrophe, e.g., $Q_{\theta'}$. These parameters are periodically synchronized with the current network parameters ($\theta' \leftarrow \theta$).

4 Model-based Representations for Q-learning

This section presents the MR.Q algorithm (Model-based Representations for Q-learning), a model-free RL algorithm that learns an approximately linear representation of the value function through

model-based objectives. Value-based RL algorithms learn a value function Q that maps state-action pairs (s, a) to values in $\mathbb{R}$ and a policy π that maps states s to actions a. Like many representation learning methods for RL, MR.Q adds an initial step that transforms states and state-action pairs into embeddings $\mathbf{z}_s$ and $\mathbf{z}_{sa}$, which serves as inputs to the downstream policy and value function.

$$f_\omega : s \to \mathbf{z}_s, \qquad g_\omega : (s, a) \to \mathbf{z}_{sa}, \tag{1}$$

$$\pi_\phi : \mathbf{z}_s \to a, \qquad Q_\theta : \mathbf{z}_{sa} \to \mathbb{R}. \tag{2}$$

While neither the value function nor policy require explicit representation learning, using intermediate embeddings has two main benefits:

1. Introducing an explicit representation learning stage can enable richer alternative learning signals that are grounded in the dynamics and rewards of the MDP, as opposed to relying exclusively on non-stationary value targets used in both value and policy learning.
2. Representation learning can transform the input into a unified, abstract space that is decoupled from the original input characteristics, e.g., images or action spaces. This abstraction allows us to filter irrelevant or spurious details and use unified downstream architectures, improving robustness to environment variations.

To learn these embeddings, we draw inspiration from linear feature selection, revisiting the work of Parr et al. (2008), as well as MDP homomorphisms (Ravindran & Barto, 2002). In Section 4.1 we highlight how model-based objectives can be used to learn features that share an approximately linear relationship with the true value function. Then in Section 4.2, we relax our theoretical motivation for a practical algorithm based on recent advances in dynamics-based representation learning.

4.1 Theoretical Motivation

Consider a linear decomposition of the value function, where the value function $Q(s, a)$ is represented by features $\mathbf{z}_{sa}$ and linear weights $\mathbf{w}$:

$$Q(s, a) = \mathbf{z}_{sa}^\top \mathbf{w}. \tag{3}$$

Our primary objective is to learn features $\mathbf{z}_{sa}$ that share an approximately linear relationship with the true value function Q^π. However, since this relationship is only approximate, we use these features as input to a non-linear function $\hat{Q}(\mathbf{z}_{sa})$, rather than relying solely on linear function approximation.

We start by exploring how to find features that can linearly represent the true value function. Given a dataset D of tuples (s, a, r, s', a'), we consider two possible approaches for learning a value function Q: A model-free update based on semi-gradient TD (Sutton, 1988; Sutton & Barto, 1998):

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbb{E}_D \left[\nabla_{\mathbf{w}} \left(\mathbf{z}_{sa}^\top \mathbf{w} - |r + \gamma \mathbf{z}_{s'a'}^\top \mathbf{w}|_{\text{sg}} \right)^2 \right]. \tag{4}$$

A model-based approach to learn $\mathbf{w}_{\text{mb}}$, based on rolling out estimates of the dynamics and reward:

$$\mathbf{w}_{\text{mb}} := \sum_{t=0}^{\infty} \gamma^t W_p^t \mathbf{w}_r, \tag{5}$$

$$\mathbf{w}_r := \operatorname*{argmin}_{\mathbf{w}} \mathbb{E}_D \left[\left(\mathbf{z}_{sa}^\top \mathbf{w} - r \right)^2 \right], \qquad W_p := \operatorname*{argmin}_{W} \mathbb{E}_D \left[\left(\mathbf{z}_{sa}^\top W - \mathbf{z}_{s'a'} \right)^2 \right]. \tag{6}$$

Closely following Parr et al. (2008) and Song et al. (2016), we can show that these approaches converge to the same solution (proofs for this section can be found in Appendix A).

Theorem 1. *The fixed point of the model-free approach (Equation 4) and the solution of the model-based approach (Equation 5) are the same.*

From the insight of Theorem 1, we can connect the value error VE, the difference between an approximate value function Q and the true value function Q^π,

$$\text{VE}(s, a) := Q(s, a) - Q^\pi(s, a) \tag{7}$$

to the accuracy of reward and dynamics components of the estimated model (Theorem 2).

Theorem 2. *The value error of the solution described by Theorem 1 is bounded by the accuracy of the estimated dynamics and reward:*

$$|\text{VE}(s, a)| \leq \frac{1}{1 - \gamma} \max_{(s,a) \in S \times A} \left(|\mathbf{z}_{sa}^\top \mathbf{w}_r - \mathbb{E}_{r|s,a}[r]| + \max_i |\mathbf{w}_i| \sum |\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'|s,a}[\mathbf{z}_{s'a'}]| \right). \tag{8}$$

Parr et al. (2008) and Song et al. (2016) use a related insight regarding the Bellman error to infer an approach for feature selection. However, with the advent of deep learning, we can instead directly learn the features $\mathbf{z}_{sa}$ by jointly optimizing them alongside the linear weights $\mathbf{w}_r$ and W_p. This is accomplished by treating the features and linear weights as a unified end-to-end model and balancing the losses in Equation 6 with a hyperparameter λ:

$$\mathcal{L}(\mathbf{z}_{sa}, \mathbf{w}_r, W_p) = \underbrace{\mathbb{E}_D\left[\left(\mathbf{z}_{sa}^\top \mathbf{w}_r - r\right)^2\right]}_{\text{Reward learning}} + \underbrace{\lambda \mathbb{E}_D\left[\left(\mathbf{z}_{sa}^\top W_p - \mathbf{z}_{s'a'}\right)^2\right]}_{\text{Dynamics learning}}. \tag{9}$$

However, the resulting Equation 9 has some notable drawbacks.

Dependency on π. The dynamics target $\mathbf{z}_{s'a'}$ depends on an action a' determined by the policy π. In policy optimization problems, this introduces non-stationarity, where the target embedding must be continually updated to reflect changes in the policy. This creates an undesirable interdependence between the policy and encoder.

Undesirable local minima. Jointly optimizing both the features $\mathbf{z}_{sa}$ and the dynamics target can lead to undesirable local minima, similar to the issues encountered with Bellman residual minimization (Baird, 1995; Fujimoto et al., 2022). This can result in collapsed or trivial solutions when the dataset does not fully cover the state and action space or when the reward is sparse.

To address these issues, we suggest relaxations on our proposed, theoretically grounded approach:

$$\mathcal{L}(\mathbf{z}_{sa}, \mathbf{w}_r, W_p) = \mathbb{E}_D\left[\left(\mathbf{z}_{sa}^\top \mathbf{w}_r - r\right)^2\right] + \lambda \mathbb{E}_D\left[\left(\mathbf{z}_{sa}^\top W_p - \underbrace{\bar{\mathbf{z}}_{s'}}_{\text{Adjustment}}\right)^2\right]. \tag{10}$$

We propose two key modifications to alleviate the aforementioned issues. Firstly, we use a state-dependent embedding $\mathbf{z}_{s'}$ as the dynamics target, rather than the state-action embedding $\mathbf{z}_{s'a'}$. This eliminates any dependency on the current policy while still capturing the environment's dynamics.

Secondly, to mitigate the issue of local minima, we use a target network $f_{\omega'}(s')$ to generate the dynamics target $\bar{\mathbf{z}}_{s'}$, where the parameters ω' are periodically updated to track the current network parameters ω. Empirical evidence from prior work suggests that this approach can yield significant performance gains (Grill et al. (2020); Assran et al. (2023), see Related Work), although it no longer guarantees convergence to a fixed point.

Due to these two changes, even if the modified objective defined by Equation 10 is minimized, we can no longer assume there is a *linear* relationship between the embedding $\mathbf{z}_{sa}$ and the value function. However, we can instead allow for a *non-linear* relationship, replacing linear weights $\mathbf{w}$ with a non-linear function $\hat{Q}(\mathbf{z}_{sa})$. We can show that this relationship exists as long as the features are sufficiently rich (i.e., such that a MDP homomorphism is satisfied (Ravindran & Barto, 2002)).

Theorem 3. *Given functions $f(s) = \mathbf{z}_s$ and $g(\mathbf{z}_s, a) = \mathbf{z}_{sa}$, then if there exists functions $\hat{p}$ and $\hat{R}$ such that for all $(s, a) \in S \times A$:*

$$\mathbb{E}_{\hat{R}}[\hat{R}(\mathbf{z}_{sa})] = \mathbb{E}_R[R(s, a)], \qquad \hat{p}(\mathbf{z}_{s'}|\mathbf{z}_{sa}) = \sum_{\hat{s}:\mathbf{z}_{\hat{s}}=\mathbf{z}_{s'}} p(\hat{s}|s, a), \tag{11}$$

then for any policy π where there exists a corresponding policy $\hat{\pi}(a|\mathbf{z}_s) = \pi(a|s)$, there exists a function $\hat{Q}$ equal to the true value function Q^π over all possible state-action pairs $(s, a) \in S \times A$:

$$\hat{Q}(\mathbf{z}_{sa}) = Q^\pi(s, a). \tag{12}$$

Furthermore, Equation 11 guarantees the existence of an optimal policy $\hat{\pi}^*(a|\mathbf{z}_s) = \pi^*(a|s)$.

Consequently, even if the features $\mathbf{z}_{sa}$ do not linearly represent the true value function, i.e., the loss in Equation 9 cannot be not exactly minimized, $\mathbf{z}_{sa}$ can still be used in a non-linear relationship to represent the value function. Furthermore, Theorem 3 outlines a similar objective as the original linear objective defined in Equation 9, in learning the reward and dynamics of the MDP.

These results motivates the practical algorithm discussed in the following section. Using the adjusted loss defined in Equation 10, we will aim to learn features with an approximately linear relationship to the true value function, but use a non-linear value function with those features to account for the error induced by our approximations.

4.2 ALGORITHM

We now present the details of MR.Q (Model-based Representations for Q-learning). Building on the insights from the previous section, our key idea is to learn a state-action embedding $\mathbf{z}_{sa}$ that is approximately linear with the true value function Q^π. To account for approximation errors, these features are used with *non-linear* function approximation to determine the value.

The state embedding vector $\mathbf{z}_s$ is obtained as an intermediate component by training end-to-end with the state-action encoder. MR.Q handles different input modalities by swapping the architecture of the state encoder. Since $\mathbf{z}_s$ is a vector, the remaining networks are independent of the observation space and use feedforward networks.

Given the transition (s, a, r, d, s') from the replay buffer:

Output MR.Q	
Trained end-to-end	
State Encoder	$\mathbf{z}_s = f_\omega(s)$
State-Action Encoder	$\mathbf{z}_{sa} = g_\omega(\mathbf{z}_s, a)$
MDP predictor	$\tilde{\mathbf{z}}_{s'}, \tilde{r}, \tilde{d} = \mathbf{z}_{sa}^\top \mathbf{m}$
Decoupled RL	
Value	$\tilde{Q}_i = Q_\theta(\mathbf{z}_{sa})$
Policy	$a_\pi = \pi_\phi(\mathbf{z}_s)$

Update MR.Q

if $t \,\%\, T_{\text{target}} = 0$ **then**
  Target networks: $\theta', \phi', \omega' \leftarrow \theta, \phi, \omega$.
  Reward scaling: $\bar{r}' \leftarrow \bar{r}$, $\bar{r} \leftarrow \text{mean}_D r$.
  for T_{target} time steps **do**
    Encoder update: Equation 14.
Value update: Equation 19.
Policy update: Equation 20.

The encoder loss is composed of three terms based on the reward, dynamics and terminal signal that are unrolled over a short horizon. The value function and policy are trained independently, using standard losses (Silver et al., 2014; Fujimoto et al., 2018). We use LAP (Fujimoto et al., 2020) to sample transitions with priority according to their TD errors (Schaul et al., 2016), the absolute difference between the predicted value and the target value in Equation 19.

The target network, reward scaling (defined in Equation 19), and the encoder are updated periodically every T_{target} time steps. This synchronized update schedule keeps the input and target output fixed for the downstream value function and policy within each iteration, thus reducing non-stationarity in the optimization (Fujimoto et al., 2024).

4.2.1 ENCODER

The encoder loss is based on unrolling the dynamics of the learned model over a short horizon. Given a subsequence of an episode $(s_0, a_0, r_1, d_1, s_1, ..., r_{H_{\text{Enc}}}, d_{H_{\text{Enc}}}, s_{H_{\text{Enc}}})$, the model is unrolled by encoding the initial state s_0, then by repeatedly applying the state-action encoder g_ω and linear MDP predictor $\mathbf{m}$:

$$\tilde{\mathbf{z}}^t, \tilde{r}^t, \tilde{d}^t := g_\omega(\tilde{\mathbf{z}}^{t-1}, a^{t-1})^\top \mathbf{m}, \qquad \text{where } \tilde{\mathbf{z}}^0 := f_\omega(s_0). \tag{13}$$

The final loss is summed over the unrolled model and balanced by corresponding hyperparameters:

$$\mathcal{L}_{\text{Encoder}}(f, g, \mathbf{m}) := \sum_{t=1}^{H_{\text{Enc}}} \lambda_{\text{Reward}} \mathcal{L}_{\text{Reward}}(\tilde{r}^t) + \lambda_{\text{Dynamics}} \mathcal{L}_{\text{Dynamics}}(\tilde{\mathbf{z}}_{s'}^t) + \lambda_{\text{Terminal}} \mathcal{L}_{\text{Terminal}}(\tilde{d}^t). \tag{14}$$

$\lambda_{\text{Terminal}}$ is set to 0 until the first terminal transition (i.e., $d = 0$) is viewed. This approach is commonly used in model-based RL (Oh et al., 2015; Hafner et al., 2023; Hansen et al., 2024), as well as dynamics-based representation learning (Schwarzer et al., 2020; 2023; Scannell et al., 2024).

Reward loss. While our theoretical analysis suggests using the mean-squared error to train the predicted reward, we find that a categorical representation of the reward is more effective in practice for predicting sparse rewards and is robust to reward magnitude. This empirical benefit is consistent with prior work (Schrittwieser et al., 2020; Hafner et al., 2023; Hansen et al., 2024; Wang et al., 2024). Our reward loss function uses the cross entropy CE between the predicted reward $\tilde{r}$ and a two-hot encoding of the reward r:

$$\mathcal{L}_{\text{Reward}}(\tilde{r}) := \text{CE}\left(\tilde{r}, \text{Two-Hot}(r)\right). \tag{15}$$

To handle a wide range of reward magnitudes without prior knowledge, the locations of the two-hot encoding are spaced at increasing non-uniform intervals, according to $\text{symexp}(x) = \text{sign}(x)(\exp(x) - 1)$ (Hafner et al., 2023).

Dynamics loss. The dynamics loss minimizes the mean-squared error between the predicted next state embedding $\tilde{\mathbf{z}}_{s'}$ and the next state embedding $\bar{\mathbf{z}}_{s'}$ from the target encoder $f_{\omega'}$:

$$\mathcal{L}_{\text{Dynamics}}(\tilde{\mathbf{z}}_{s'}) := \left(\tilde{\mathbf{z}}_{s'} - \bar{\mathbf{z}}_{s'}\right)^2. \tag{16}$$

As discussed in the previous section, using the next state embedding $\mathbf{z}_{s'}$ eliminates the dependency on the policy that would occur when using a state-action embedding target.

Terminal loss. To encode termination, the dynamics loss is repeated where the target is masked by the binary terminal signal d, i.e., the target is set to the 0-vector if the transition is terminal:

$$\mathcal{L}_{\text{Terminal}}(\tilde{d}) := (\tilde{d} - d\bar{\mathbf{z}}_{s'})^2. \tag{17}$$

This form of loss allows for a dense learning signal even when the terminal function is highly sparse.

4.2.2 Value Function

Value learning is primarily based on TD3 (Fujimoto et al., 2018). Specifically, we train two value functions and take the minimum output between their respective target networks to determine the value target. Similar to TD3, the target action is determined by the target policy $\pi_{\phi'}$, perturbed by small amount of clipped Gaussian noise:

$$a_\pi = \begin{cases} \text{argmax}\, a' & \text{for discrete } A, \\ \text{clip}(a', -1, 1) & \text{for continuous } A, \end{cases} \quad \text{where } a' = \pi_{\phi'}(s') + \text{clip}(\epsilon, -c, c), \quad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{18}$$

Discrete actions are represented by a one-hot encoding, where the Gaussian noise is added to each dimension. Action noise and the clipping is scaled according the range of the action space.

We modify the TD3 loss in a few ways. Firstly, following numerous prior work across benchmarks (Hessel et al., 2018; Barth-Maron et al., 2018; Yarats et al., 2022; Schwarzer et al., 2023), we predict multi-step returns over a horizon H_Q. Secondly, we use the Huber loss instead of mean-squared error to eliminate bias from prioritized sampling (Fujimoto et al., 2020). Finally, the target value is normalized according to the average absolute reward $\bar{r}$ in the replay buffer:

$$\mathcal{L}_{\text{Value}}(\tilde{Q}_i) := \text{Huber}\left(\tilde{Q}_i, \frac{1}{\bar{r}}\left(\sum_{t=0}^{H_Q - 1} \gamma^t r_t + \gamma^{H_Q} \tilde{Q}'_j\right)\right), \quad \tilde{Q}'_j := \bar{r}' \min_{j=1,2} Q_{\theta'_j}(\mathbf{z}_{s_{H_Q} a_{H_Q, \pi}}). \tag{19}$$

The value $\bar{r}'$ captures the *target* average absolute reward, which is the scaling factor used to the most recently copied value functions $Q_{\theta'_j}$. This value is updated simultaneously with the target networks $\bar{r}' \leftarrow \bar{r}$. Maintaining a consistent reward scale keeps the loss magnitude constant across different benchmarks, thus improving the robustness of a single set of hyperparameters.

4.2.3 Policy

For both continuous and discrete action spaces, the policy is updated using the deterministic policy gradient (Silver et al., 2014):

$$\mathcal{L}_{\text{Policy}}(a_\pi) := -0.5 \sum_{i=\{1,2\}} \tilde{Q}_i(\mathbf{z}_{s a_\pi}) + \lambda_{\text{pre-activ}} \mathbf{z}_\pi^2, \quad \text{where } a_\pi = \text{activ}(\mathbf{z}_\pi). \tag{20}$$

To make the loss universal between action spaces, we use Gumbel-Softmax (Jang et al., 2017; Lowe et al., 2017; Cianflone et al., 2019) for discrete actions, and Tanh for continuous actions. A small regularization penalty is added to the square of the pre-activations $\mathbf{z}_\pi$ before the policy's final activation to help avoid local minima when the reward, and value, is sparse (Bjorck et al., 2021).

For exploration, Gaussian noise is added to each dimension of the action (or one-hot encoding of the action). Similar to Equation 18, the resulting action vector is clipped to the range of the action space for continuous actions. For discrete actions, the final action is determined by the argmax operation.

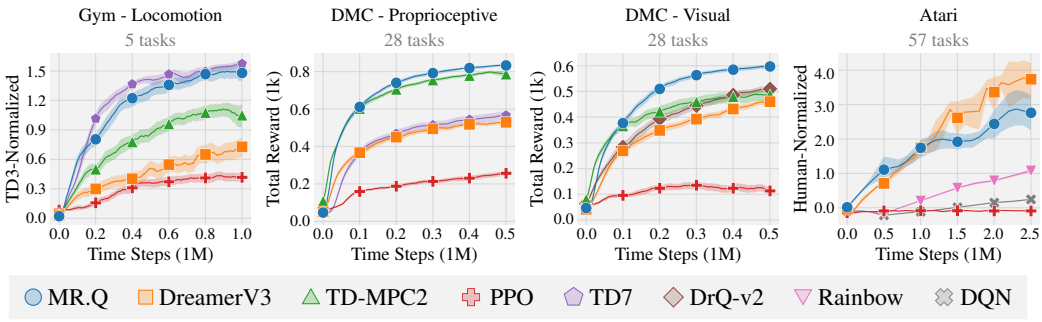

Figure 2: **Aggregate learning curves.** Average performance over each benchmark. Results are over 10 seeds. The shaded area captures a 95% stratified bootstrap confidence interval. Due to action repeat, 500k time steps in DMC correspond to 1M frames in the original environment and 2.5M time steps in Atari corresponds to 10M frames in the original environment.

5 Experiments

We evaluate MR.Q on four popular RL benchmarks and 118 environments, and compare its performance against strong domain-specific baselines, the general model-based approaches, DreamerV3 (Hafner et al., 2023) and TD-MPC2 (Hansen et al., 2024), and the general model-free algorithm, PPO (Schulman et al., 2017). Rather than establish MR.Q as the state-of-the-art approach in any particular benchmark, our objective is to demonstrate its broad applicability and effectiveness across a diverse set of tasks with a single set of hyperparameters. The baselines use author-suggested default hyperparameters and are fixed across environments.

5.1 Main Results

Aggregate learning curves are displayed in Figure 2, with full results displayed in Appendix C.

Gym - Locomotion. This subset of the Gym benchmark (Brockman et al., 2016; Towers et al., 2024) considers 5 locomotion tasks in the MuJoCo simulator (Todorov et al., 2012) with continuous actions and low level states. Agents are trained for 1M time steps without any environment preprocessing. We evaluate against three baselines: TD7 (Fujimoto et al., 2024), a state-of-the-art (or near) approach for this benchmark, as well as TD-MPC2, DreamerV3, and PPO. To aggregate results, we normalize using the performance of TD3 (Fujimoto et al., 2018).

DMC - Proprioceptive. The DeepMind Control suite (DMC) (Tassa et al., 2018) is a collection of continuous control robotics tasks built on the MuJoCo simulator. These tasks use the proprioceptive states as the observation space, meaning that the input is a vector, and limit the total reward for each episode at 1000, making it easy to aggregate results. We report results on all 28 default tasks that were used by either TD-MPC2 or DreamerV3. Agents are trained for 500k time steps, equivalent to 1M frames in the original environment due to action repeat. For comparison, we evaluate against the same three algorithms as in the Gym benchmark, with TD-MPC2 considered state-of-the-art (or near) for this benchmark. We also include TD7 due to its strong performance in the Gym benchmark.

DMC - Visual. The visual DMC benchmark includes the same 28 tasks as the proprioceptive benchmark, but uses image-based observations instead. Agents are trained for 500k time steps. For baselines, we include DrQ-v2 (Yarats et al., 2022), given its state-of-the-art (or near) performance in model-free RL, alongside TD-MPC2, DreamerV3, and PPO.

Atari. The Atari benchmark is built on the Arcade Learning Environment (Bellemare et al., 2013). This benchmark uses pixel observations and discrete actions and includes the 57 games used by DreamerV3. We follow standard preprocessing steps, including sticky actions (Machado et al., 2018) (full details in Appendix B.3). Agents are trained for 2.5M time steps (equivalent to 10M frames), a setting which has been considered by prior work (Sokar et al., 2023). For comparison, we evaluate against three baselines: the model-based approach DreamerV3, as well as model-free approaches, DQN (Mnih et al., 2015), Rainbow (Hessel et al., 2018), and PPO. Results are aggregated by normalizing scores against human performance.

Discussion. Throughout our experiments, we find the presence of "no free lunch", where the top-performing baseline in one benchmark fails to replicate its success in another. Regardless, MR.Q achieves the highest performance in both DMC benchmarks, showcasing its ability to handle different observation spaces. Although it falls slightly behind TD7 in the Gym benchmark, MR.Q is the strongest method overall across all continuous control benchmarks. In Atari, while DreamerV3 outperforms MR.Q, it relies on a model with 40 times more parameters and struggles comparatively in the remaining benchmarks. When compared to the model-free baselines, MR.Q surpasses PPO, DQN, and Rainbow, demonstrating its effectiveness with discrete action spaces.

5.2 Design Study

To better understand the impact of certain design choices and hyperparameters, we attempt variations of MR.Q, and report the aggregate results in Table 2.

Table 2: **Design study.** Average difference in normalized performance from varying design choices across each benchmark over 5 seeds. Negative changes are highlighted lightly $[-0.01, -0.2)$. Damaging changes are highlighted moderately $[-0.2, -0.5)$. Catastrophic changes are highlighted boldly (≤ -0.5). Positive changes are similarly highlighted (> 0.01).

Design	Gym - Locomotion TD3-Normalized	DMC - Proprioceptive Reward (1k)	DMC - Visual Reward (1k)	Atari - 1M Human-Normalized
		Relaxations		
Linear value function	-1.12 [-1.15, -1.08]	-0.58 [-0.59, -0.56]	-0.41 [-0.42, -0.39]	-1.56 [-1.62, -1.50]
Dynamics target	-0.15 [-0.24, -0.01]	-0.15 [-0.15, -0.15]	-0.05 [-0.05, -0.04]	-0.59 [-1.02, -0.16]
No target encoder	-0.65 [-0.72, -0.58]	-0.35 [-0.35, -0.34]	-0.15 [-0.15, -0.15]	-1.07 [-1.10, -1.04]
Revert	-1.40 [-1.46, -1.35]	-0.72 [-0.73, -0.72]	-0.52 [-0.52, -0.51]	-1.90 [-1.91, -1.88]
Non-linear model	0.06 [-0.01, 0.14]	-0.00 [-0.02, 0.01]	-0.01 [-0.02, -0.00]	-0.28 [-0.53, -0.03]
		Loss functions		
MSE reward loss	0.19 [0.07, 0.30]	-0.06 [-0.08, -0.05]	-0.05 [-0.07, -0.04]	-1.00 [-1.07, -0.93]
No reward scaling	-0.08 [-0.23, 0.03]	-0.01 [-0.02, 0.00]	-0.00 [-0.01, 0.01]	-0.03 [-0.46, 0.36]
MSE terminal loss	-0.17 [-0.33, -0.03]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.14 [-0.09, 0.33]
No min	-0.39 [-0.47, -0.31]	-0.01 [-0.02, 0.01]	0.00 [-0.01, 0.01]	-0.08 [-0.31, 0.37]
No LAP	-0.11 [-0.14, -0.08]	0.00 [-0.00, 0.01]	-0.01 [-0.02, -0.01]	-0.01 [-0.34, 0.56]
No MR	-0.48 [-0.63, -0.34]	-0.19 [-0.19, -0.18]	-0.07 [-0.09, -0.03]	-0.99 [-1.09, -0.90]
		Horizons		
1-step return	-0.28 [-0.39, -0.17]	-0.04 [-0.05, -0.02]	-0.03 [-0.03, -0.02]	-0.91 [-1.02, -0.80]
No unroll	-0.03 [-0.21, 0.13]	-0.01 [-0.01, -0.00]	-0.04 [-0.06, -0.01]	-0.54 [-0.61, -0.48]

Relaxations. In Section 4.1, we outlined a loss (Equation 9) that, if globally minimized, would provide features that are linear with the true value function. MR.Q in practice relaxes this theoretical result by modifying the loss and using a non-linear value function. In **Linear value function**, we replace the non-linear value function with a linear function. In **Dynamics target**, we replace the state embedding dynamics target with a state-action embedding $\bar{\mathbf{z}}_{s'a'}$ determined from the target state-action encoder g_ω. In **No target encoder**, we use the current encoder to generate the dynamics target $\mathbf{z}_{s'a'}$, and jointly optimize it within the encoder loss. In **Revert**, we consider all of the aforementioned changes simultaneously, using linear value functions and setting the dynamics target as a state-action embedding determined by the current encoder. In **Non-linear model**, we replace the linear MDP predictor with individual networks that predict each component separately from $\mathbf{z}_{sa}$.

Loss functions. MR.Q's loss functions use several unconventional choices. In **MSE reward loss**, we replace the categorical loss function on the predicted reward in Equation 15 with the mean-squared error (MSE). In **No reward scaling**, we remove the reward scaling in Equation 19, setting $\bar{r} = \bar{r}' = 1$. In **MSE terminal loss**, we replace the loss function in Equation 17 with the MSE with the binary terminal signal. In **No min**, we take the mean over the target value functions instead of the minimum in Equation 19. In **No LAP**, we remove prioritized sampling (Fujimoto et al., 2020) and use the MSE instead of the Huber loss in the value update. Lastly, in **No MR**, we remove model-based representation learning and train the encoder end-to-end with the value function.

Horizons. Finally, we consider the role of extended predictions. In **1-step return**, we remove multi-step value predictions and use TD learning. In **No unroll**, we remove the dynamics unrolling in Equation 14, by setting the encoder horizon $H_{\text{Enc}} = 1$.

Discussion. Our design study shows the importance of the practical relaxations to our theoretical results were necessary to achieve a strong performance. The experiments further validate the benefits of design choices and hyperparameters. We highlight two results in particular: (1) in the non-linear model experiment, increasing the model capacity did not improve performance. This result suggests that the advantages of maintaining an approximately linear relationship with the value function can outweigh the benefits of increased capacity. (2) Our study also reveals a key distinction between the Gym and Atari benchmarks—while the MSE reward loss function offers a moderate performance gain in Gym, it significantly degrades performance in Atari. This discrepancy highlights how hyperparameters can overfit to individual benchmarks.

6 Discussion and Conclusion

This paper introduces MR.Q, a general model-free deep RL algorithm that achieves strong performance across diverse benchmarks and environments. Drawing inspiration from the theory of model-based representation learning, MR.Q demonstrates that model-free deep RL is a promising avenue for building general-purpose algorithms that achieve high performance across environments, while being simpler and less expensive than model-based alternatives.

Our work also reveals insights on which design choices matter when building general-purpose model-free deep RL algorithms and how common benchmarks respond to these design choices.

Model-based and model-free RL. MR.Q integrates model-based objectives with a model-free backbone during training, effectively blurring the boundary between traditional model-based and model-free RL. While MR.Q could be extended to the model-based setting by incorporating planning or simulated trajectories with the state-action encoder, these components can add significant execution time and increase the overall complexity and tuning required by a method. Moreover, the performance of MR.Q in these common RL benchmarks demonstrates that these model-based components may be simply unnecessary—suggesting that the representation itself could be the most valuable aspect of model-based learning, even in methods that do use planning. This argument is echoed by DreamerV3 and TD-MPC2, which rely on short planning horizons and trajectory generation, while including both value functions and traditional model-free policy updates. As such, it may be necessary to examine more complex settings, to reliably see a benefit from model-based search or planning, e.g., (Silver et al., 2016).

Universality of RL benchmarks. Our results demonstrate that there is a striking lack of positive transfer between benchmarks. For example, despite the similarities in tasks and the same underlying MuJoCo simulator, the top performers in Gym and DMC fail to replicate their success on the opposing benchmark. Similarly, although DreamerV3 excels at Atari, these performance benefits do not translate to continuous control environments, underperforming TD3 in Gym and outright failing to learn the Dog and Humanoid tasks in DMC (see Appendix C). These findings show the limitations of single-benchmark evaluations, indicating that success on one benchmark may not translate easily to others, and highlights the need for more comprehensive benchmarks.

Limitations. MR.Q is only the first step towards a new generation of general-purpose model-free deep RL algorithms. Many challenges remains for a fully general algorithm. In particular, MR.Q is not equipped to handle settings such as hard exploration tasks or non-Markovian environments. Another limitation is our evaluation only considers standard RL benchmarks. Although this allows direct comparison with other methods, established algorithms such as PPO have demonstrated their effectiveness in highly unique settings, such as team video games (Berner et al., 2019), drone racing (Kaufmann et al., 2023), and large language models (Achiam et al., 2023; Touvron et al., 2023). To demonstrate similar versatility, new algorithms must undergo the same rigorous testing across a range of tasks that is beyond the scope of any single study.

As the community continues to push the boundaries of what is possible with deep RL, we believe that building simpler general-purpose algorithms has the potential to make this technology more accessible to a wider audience, ultimately enabling users to train agents with ease. Perhaps one day—with just the click of a button.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pp. 507–517. PMLR, 2020.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributional policy gradients. *International Conference on Learning Representations*, 2018.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684, 1957.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Johan Bjorck, Carla P Gomes, and Kilian Q Weinberger. Is high variance unavoidable in rl? a case study in continuous control. In *International Conference on Learning Representations*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10069–10076, 2020.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34:30113–30126, 2021.

Wesley Chung, Somjit Nath, Ajin Joseph, and Martha White. Two-timescale networks for nonlinear value function approximation. In *International Conference on Learning Representations*, 2019.

Andre Cianflone, Zafarali Ahmed, Riashat Islam, Avishek Joey Bose, and William L Hamilton. Discrete off-policy policy gradient using continuous relaxations, 2019.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pp. 162–169, 2004.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519. IEEE, 2016.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, volume 80, pp. 1587–1596. PMLR, 2018.

Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in Neural Information Processing Systems*, 33, 2020.

Scott Fujimoto, David Meger, and Doina Precup. A deep reinforcement learning approach to marginalized importance sampling with the successor representation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 3518–3529. PMLR, 2021.

Scott Fujimoto, David Meger, Doina Precup, Ofir Nachum, and Shixiang Shane Gu. Why should i trust you, bellman? The Bellman error is a poor replacement for value error. In *International Conference on Machine Learning*, volume 162, pp. 6918–6943. PMLR, 2022.

Scott Fujimoto, Wei-Di Chang, Edward J. Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For SALE: State-action representation learning for deep reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2024.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179. PMLR, 2019.

Dibya Ghosh and Marc G Bellemare. Representations for stable off-policy reinforcement learning. In *International Conference on Machine Learning*, pp. 3556–3565. PMLR, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pp. 3875–3886. PMLR, 2020.

Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35:31855–31870, 2022.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, volume 80, pp. 1861–1870. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, pp. 8387–8406. PMLR, 2022.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

Ammar Haydari and Yasin Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2020.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022.

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *stat*, 1050:3, 2017.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

Kyungsoo Kim, Jeongsoo Ha, and Yusung Kim. Self-predictive dynamics for generalization of vision-based reinforcement learning. In *IJCAI*, pp. 3150–3156, 2022.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pp. 5556–5566. PMLR, 2020.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

Nir Levine, Tom Zahavy, Daniel J Mankowitz, Aviv Tamar, and Shie Mannor. Shallow updates for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Michael Littman and Richard S Sutton. Predictive representations of state. *Advances in neural information processing systems*, 14, 2001.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE communications surveys & tutorials*, 21(4):3133–3174, 2019.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Trevor McInroe, Lukas Schäfer, and Stefano V Albrecht. Learning temporally-consistent representations for data-efficient reinforcement learning. *arXiv preprint arXiv:2110.04935*, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Jelle Munk, Jens Kober, and Robert Babuška. Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4667–4673. IEEE, 2016.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. In *The Twelfth International Conference on Learning Representations*, 2024.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, 28, 2015.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 752–759, 2008.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Marek Petrik. An analysis of laplacian methods for value function approximation in mdps. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 2574–2579, 2007.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004.

Balaraman Ravindran and Andrew G Barto. Model minimization in hierarchical reinforcement learning. In *Abstraction, Reformulation, and Approximation: 5th International Symposium, SARA 2002 Kananaskis, Alberta, Canada August 2–4, 2002 Proceedings 5*, pp. 196–211. Springer, 2002.

Ingo Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie: Workshop, Hannover, 29. Sept.–1. Okt. 1977*, pp. 83–114. Springer, 1978.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Sahand Rezaei-Shoshtari, Rosie Zhao, Prakash Panangaden, David Meger, and Doina Precup. Continuous mdp homomorphisms and homomorphic policy gradient. In *Advances in Neural Information Processing Systems*, 2022.

Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Aidan Scannell, Kalle Kujanpää, Yi Zhao, Mohammadreza Nakhaei, Arno Solin, and Joni Pajarinen. iqrl–implicitly quantized representations for sample-efficient reinforcement learning. *arXiv preprint arXiv:2406.02696*, 2024.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, Puerto Rico, 2016.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2020.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pp. 19561–19579. PMLR, 2022.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pp. 387–395, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 32145–32168. PMLR, 2023.

Zhao Song, Ronald E Parr, Xuejun Liao, and Lawrence Carin. Linear feature encoding for reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Yunhao Tang, Zhaohan Daniel Guo, Pierre Harvey Richemond, Bernardo Avila Pires, Yash Chandak, Rémi Munos, Mark Rowland, Mohammad Gheshlaghi Azar, Charline Le Lan, Clare Lyle, et al. Understanding self-predictive learning for reinforcement learning. In *International Conference on Machine Learning*, pp. 33632–33656. PMLR, 2023.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033. IEEE, 2012.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Elise van der Pol, Thomas Kipf, Frans A Oliehoek, and Max Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1431–1439, 2020a.

Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020b.

Herke Van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 3928–3934. IEEE, 2016.

Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. Efficientzero v2: Mastering discrete and continuous control with limited data. *arXiv preprint arXiv:2403.00564*, 2024.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1995–2003, 2016.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021.

Tao Yu, Cuiling Lan, Wenjun Zeng, Mingxiao Feng, Zhizheng Zhang, and Zhibo Chen. Playvirtual: Augmenting cycle-consistent virtual trajectories for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5276–5289, 2021.

Tao Yu, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. Mask-based latent reconstruction for reinforcement learning. *Advances in Neural Information Processing Systems*, 35:25117–25131, 2022.

Hongyu Zang, Xin Li, and Mingzhong Wang. Simsr: Simple distance-based state representations for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8997–9005, 2022.

Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*, 2018.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2020.

Yi Zhao, Wenshuai Zhao, Rinu Boney, Juho Kannala, and Joni Pajarinen. Simplified temporal consistency reinforcement learning. In *International Conference on Machine Learning*, pp. 42227–42246. PMLR, 2023.

Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé III, and Furong Huang. Taco: Temporal latent action-driven contrastive loss for visual reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: Variational bayes-adaptive deep rl via meta-learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.

Appendix

Table of Contents

A Proofs

Theorem 1. *The fixed point of the model-free approach (Equation 4) and the solution of the model-based approach (Equation 5) are the same.*

Proof. Let Z be a matrix containing state-action embeddings $\mathbf{z}_{sa}$ for each state-action pair $(s,a) \in S \times A$. Let Z' be the corresponding matrix of next state-action embeddings $\mathbf{z}_{s'a'}$. Let R be the vector of the corresponding rewards $r(s,a)$.

The linear semi-gradient TD update:

$$\begin{aligned}
\mathbf{w}_{t+1} &:= \mathbf{w}_t - \alpha Z^\top (Z\mathbf{w}_t - (R + \gamma Z'\mathbf{w}_t)) && (21)\\
&= \mathbf{w}_t - \alpha Z^\top Z\mathbf{w}_t + \alpha Z^\top R + \alpha\gamma Z^\top Z'\mathbf{w}_t && (22)\\
&= (I - \alpha(Z^\top Z - \gamma Z^\top Z'))\mathbf{w}_t + \alpha Z^\top R && (23)\\
&= (I - \alpha A)\mathbf{w}_t + \alpha B, && (24)
\end{aligned}$$

where $A := Z^\top Z - \gamma Z^\top Z'$ and $B := Z^\top R$.

The fixed point of the system:

$$\begin{aligned}
\mathbf{w}_{\text{mf}} &= (I - \alpha A)\mathbf{w}_{\text{mf}} + \alpha B && (25)\\
\mathbf{w}_{\text{mf}} - (I - \alpha A)\mathbf{w}_{\text{mf}} &= \alpha B && (26)\\
\alpha A\mathbf{w}_{\text{mf}} &= \alpha B && (27)\\
\mathbf{w}_{\text{mf}} &= A^{-1}B. && (28)
\end{aligned}$$

The least squares solution to W_p and $\mathbf{w}_r$

$$W_p := \left(Z^\top Z\right)^{-1} Z^\top Z' \tag{29}$$

$$\mathbf{w}_r := \left(Z^\top Z\right)^{-1} Z^\top R \tag{30}$$

By rolling out W_p and $\mathbf{w}_r$, we arrive at a model-based solution:

$$Q := Z\mathbf{w}_{\text{mb}} = Z \sum_{t=0}^{\infty} \gamma^t W_p^t \mathbf{w}_r. \tag{31}$$

Simplify $\mathbf{w}_{\text{mb}}$:

$$\mathbf{w}_{\text{mb}} := \sum_{t=0}^{\infty} \gamma^t W_p^t \mathbf{w}_r \tag{32}$$

$$\mathbf{w}_{\text{mb}} = \left(I - \gamma W_p\right)^{-1} \mathbf{w}_r \tag{33}$$

$$\mathbf{w}_{\text{mb}} = \left(I - \gamma \left(Z^\top Z\right)^{-1} Z^\top Z'\right)^{-1} \left(Z^\top Z\right)^{-1} Z^\top R \tag{34}$$

$$Z^\top Z \left(I - \gamma \left(Z^\top Z\right)^{-1} Z^\top Z'\right) \mathbf{w}_{\text{mb}} = Z^\top R \tag{35}$$

$$\left(Z^\top Z - \gamma Z^\top Z'\right) \mathbf{w}_{\text{mb}} = Z^\top R \tag{36}$$

$$\mathbf{w}_{\text{mb}} = A^{-1} B \tag{37}$$

$$\mathbf{w}_{\text{mb}} = \mathbf{w}_{\text{mf}}. \tag{38}$$

■

Theorem 2. *The value error of the solution described by Theorem 1 is bounded by the accuracy of the estimated dynamics and reward:*

$$|\text{VE}(s,a)| \leq \frac{1}{1-\gamma} \max_{(s,a)\in S\times A} \left(|\mathbf{z}_{sa}^\top \mathbf{w}_r - \mathbb{E}_{r|s,a}[r]| + \max_i |\mathbf{w}_i| \sum |\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'|s,a}[\mathbf{z}_{s'a'}]| \right). \tag{39}$$

Proof. Let $\mathbf{w}$ be the solution described in Theorem 1, i.e. $\mathbf{w} = \mathbf{w}_{\text{mb}} = \mathbf{w}_{\text{mf}}$. Let $p^\pi(s,a)$ be the discounted state-action visitation distribution according to the policy π starting from the state-action pair (s,a).

Firstly from Theorem 1, we can show that

$$\mathbf{w} = (I - \gamma W_p)^{-1} \mathbf{w}_r \tag{40}$$

$$\Rightarrow (I - \gamma W_p)\mathbf{w} = \mathbf{w}_r \tag{41}$$

$$\Rightarrow \mathbf{w} - \gamma W_p \mathbf{w} = \mathbf{w}_r. \tag{42}$$

Simplify $\text{VE}(s,a)$:

$$\begin{aligned}
\text{VE}(s,a) &:= Q(s,a) - Q^\pi(s,a) && (43)\\
&= Q(s,a) - Q^\pi(s,a) && (44)\\
&= Q(s,a) - \mathbb{E}_{r,s',a'}\left[r + \gamma Q^\pi(s',a')\right] && (45)\\
&= Q(s,a) - \mathbb{E}_{r,s',a'}\left[r + \gamma\left(Q(s',a') - \text{VE}(s',a')\right)\right] && (46)\\
&= Q(s,a) - \mathbb{E}_{r,s',a'}\left[r + \gamma Q(s',a')\right] + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right] && (47)\\
&= Q(s,a) - \mathbb{E}_{r,s',a'}\left[r - \mathbf{z}_{sa}^\top\mathbf{w}_r + \mathbf{z}_{sa}^\top\mathbf{w}_r + \gamma Q(s',a')\right] + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right] && (48)\\
&= Q(s,a) - \mathbb{E}_{r,s',a'}\left[r - \mathbf{z}_{sa}^\top\mathbf{w}_r + \mathbf{z}_{sa}^\top\mathbf{w}_r + \gamma\left(\mathbf{z}_{s'a'}^\top\mathbf{w} - \mathbf{z}_{sa}^\top W_p\mathbf{w} + \mathbf{z}_{sa}^\top W_p\mathbf{w}\right)\right] && (49)\\
&\quad + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right]\\
&= \mathbf{z}_{sa}^\top\mathbf{w} - \mathbb{E}_{r,s',a'}\left[r - \mathbf{z}_{sa}^\top\mathbf{w}_r + \mathbf{z}_{sa}^\top\mathbf{w}_r + \gamma\left(\mathbf{z}_{s'a'}^\top\mathbf{w} - \mathbf{z}_{sa}^\top W_p\mathbf{w} + \mathbf{z}_{sa}^\top W_p\mathbf{w}\right)\right] && (50)\\
&\quad + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right]\\
&= \mathbf{z}_{sa}^\top\mathbf{w} - \mathbb{E}_r\left[r - \mathbf{z}_{sa}^\top\mathbf{w}_r + \mathbf{z}_{sa}^\top\mathbf{w}_r\right] - \gamma\mathbb{E}_{s',a'}\left[\mathbf{z}_{s'a'}^\top\mathbf{w} - \mathbf{z}_{sa}^\top W_p\mathbf{w} + \mathbf{z}_{sa}^\top W_p\mathbf{w}\right] && (51)\\
&\quad + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right]\\
&= \mathbf{z}_{sa}^\top\mathbf{w} - \mathbf{z}_{sa}^\top\mathbf{w}_r - \gamma\mathbf{z}_{sa}^\top W_p\mathbf{w} - \mathbb{E}_r\left[r - \mathbf{z}_{sa}^\top\mathbf{w}_r\right] - \gamma\mathbb{E}_{s',a'}\left[\mathbf{z}_{s'a'}^\top\mathbf{w} - \mathbf{z}_{sa}^\top W_p\mathbf{w}\right] && (52)\\
&\quad + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right]\\
&= \mathbf{z}_{sa}^\top\left(\mathbf{w} - \gamma W_p\mathbf{w} - \mathbf{w}_r\right) - \mathbb{E}_r\left[r - \mathbf{z}_{sa}^\top\mathbf{w}_r\right] - \gamma\mathbb{E}_{s',a'}\left[\mathbf{z}_{s'a'}^\top\mathbf{w} - \mathbf{z}_{sa}^\top W_p\mathbf{w}\right] && (53)\\
&\quad + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right]\\
&= -\mathbb{E}_r\left[r - \mathbf{z}_{sa}^\top\mathbf{w}_r\right] - \gamma\mathbb{E}_{s',a'}\left[\mathbf{z}_{s'a'}^\top\mathbf{w} - \mathbf{z}_{sa}^\top W_p\mathbf{w}\right] + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right] && (54)\\
&= \left(\mathbf{z}_{sa}^\top\mathbf{w}_r - \mathbb{E}_r\left[r\right]\right) + \gamma\left(\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'}\left[\mathbf{z}_{s'a'}^\top\right]\right)\mathbf{w} + \gamma\mathbb{E}_{s',a'}\left[\text{VE}(s',a')\right]. && (55)
\end{aligned}$$

Then given the recursive relationship, akin to the Bellman equation (Sutton & Barto, 1998), the value error VE recursively expands to the discounted state-action visitation distribution p^π. For $(\hat{s},\hat{a}) \in S \times A$:

$$\text{VE}(\hat{s},\hat{a}) = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim p^\pi(\hat{s},\hat{a})}\left[\left(\mathbf{z}_{sa}^\top\mathbf{w}_r - \mathbb{E}_{r|s,a}\left[r\right]\right) + \gamma\left(\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'|s,a}\left[\mathbf{z}_{s'a'}^\top\right]\right)\mathbf{w}\right]. \quad (56)$$

Taking the absolute value:

$$\begin{aligned}
|\text{VE}(\hat{s},\hat{a})| &= \left|\frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim p^\pi(\hat{s},\hat{a})}\left[\left(\mathbf{z}_{sa}^\top\mathbf{w}_r - \mathbb{E}_{r|s,a}\left[r\right]\right) + \gamma\left(\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'|s,a}\left[\mathbf{z}_{s'a'}^\top\right]\right)\mathbf{w}\right]\right| && (57)\\
|\text{VE}(\hat{s},\hat{a})| &\le \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim p^\pi(\hat{s},\hat{a})}\left[\left|\mathbf{z}_{sa}^\top\mathbf{w}_r - \mathbb{E}_{r|s,a}\left[r\right]\right| + \gamma\left|\left(\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'|s,a}\left[\mathbf{z}_{s'a'}^\top\right]\right)\mathbf{w}\right|\right] && (58)\\
&= \frac{1}{1-\gamma}\max_{(s,a)\in S\times A}\left(\left|\mathbf{z}_{sa}^\top\mathbf{w}_r - \mathbb{E}_{r|s,a}\left[r\right]\right| + \gamma\left|\left(\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'|s,a}\left[\mathbf{z}_{s'a'}^\top\right]\right)\mathbf{w}\right|\right) && (59)\\
&\le \frac{1}{1-\gamma}\max_{(s,a)\in S\times A}\left(\left|\mathbf{z}_{sa}^\top\mathbf{w}_r - \mathbb{E}_{r|s,a}\left[r\right]\right| + \max_i|\mathbf{w}_i|\sum\left|\mathbf{z}_{sa}^\top W_p - \mathbb{E}_{s',a'|s,a}\left[\mathbf{z}_{s'a'}\right]\right|\right). && (60)
\end{aligned}$$

■

Theorem 3. *Given functions $f(s) = \mathbf{z}_s$ and $g(\mathbf{z}_s, a) = \mathbf{z}_{sa}$, then if there exists functions $\hat{p}$ and $\hat{R}$ such that for all $(s,a) \in S \times A$:*

$$\mathbb{E}_{\hat{R}}[\hat{R}(\mathbf{z}_{sa})] = \mathbb{E}_R\left[R(s,a)\right], \qquad \hat{p}(\mathbf{z}_{s'}|\mathbf{z}_{sa}) = \sum_{\hat{s}:\mathbf{z}_{\hat{s}}=\mathbf{z}_{s'}} p(\hat{s}|s,a), \quad (61)$$

then for any policy π where there exists a corresponding policy $\hat{\pi}(a|\mathbf{z}_s) = \pi(a|s)$, there exists a function $\hat{Q}$ equal to the true value function Q^π over all possible state-action pairs $(s,a) \in S \times A$:

$$\hat{Q}(\mathbf{z}_{sa}) = Q^\pi(s,a). \quad (62)$$

Furthermore, Equation 61 guarantees the existence of an optimal policy $\hat{\pi}^*(a|\mathbf{z}_s) = \pi^*(a|s)$.

Proof. Let

$$Q_h^\pi(s,a) = \sum_{t=0}^{h} \gamma^t \mathbb{E}_\pi[R(s_t,a_t)|s_0 = s, a_0 = a] \tag{63}$$

$$\hat{Q}_h(\mathbf{z}_{sa}) = \sum_{t=0}^{h} \gamma^t \mathbb{E}_\pi[\hat{R}(\mathbf{z}_{s_t a_t})|s_0 = s, a_0 = a] \tag{64}$$

Then

$$\begin{aligned} Q_0^\pi(s,a) &= \mathbb{E}_R[R(s,a)] && (65)\\ &= \mathbb{E}_{\hat{R}}[\hat{R}(\mathbf{z}_{sa})] && (66)\\ &= \hat{Q}_0(\mathbf{z}_{sa}). && (67) \end{aligned}$$

Assuming $Q_{n-1}^\pi(s,a) = \hat{Q}_{n-1}(\mathbf{z}_{sa})$ then noting that $\hat{p}(\mathbf{z}|\mathbf{z}_{sa}) = 0$ if $\mathbf{z}$ that is not in the image of $f(s) = \mathbf{z}_s$.

$$\begin{aligned} Q_n^\pi(s,a) &= \mathbb{E}_R[R(s,a)] + \gamma \mathbb{E}_{s',a'}[Q_{n-1}^\pi(s',a')] && (68)\\ &= \mathbb{E}_{\hat{R}}[\hat{R}(s,a)] + \gamma \mathbb{E}_{s',a'}[\hat{Q}_{n-1}(\mathbf{z}_{s'a'})] && (69)\\ &= \mathbb{E}_{\hat{R}}[\hat{R}(s,a)] + \gamma \sum_{s'}\sum_{a'} p(s'|s,a)\pi(a'|s')\hat{Q}_{n-1}(\mathbf{z}_{s'a'}) && (70)\\ &= \mathbb{E}_{\hat{R}}[\hat{R}(s,a)] + \gamma \sum_{z_{s'}}\sum_{a'} \hat{p}(\mathbf{z}_{s'}|\mathbf{z}_{sa})\hat{\pi}(a'|\mathbf{z}_{s'})\hat{Q}_{n-1}(\mathbf{z}_{s'a'}) && (71)\\ &= \hat{Q}_n(\mathbf{z}_{sa}). && (72) \end{aligned}$$

Thus $\hat{Q}(\mathbf{z}_{sa}) = \lim_{n\to\infty} \hat{Q}_n(\mathbf{z}_{sa})$ exists, as $\hat{Q}_n$ can be defined as a function of $\hat{p}$, $\hat{R}$, and $\hat{\pi}$ for all n. Similarly, let π be an optimal policy. Repeating the same arguments we see that

$$\begin{aligned} Q_n^\pi(s,a) &= \mathbb{E}_R[R(s,a)] + \gamma \mathbb{E}_{s',a'}[Q_{n-1}^\pi(s',a')] && (73)\\ &= \mathbb{E}_R[R(s,a)] + \gamma \sum_{s'} p(s'|s,a) \max_{a'} Q_{n-1}^\pi(s',a') && (74)\\ &= \mathbb{E}_{\hat{R}}[\hat{R}(s,a)] + \gamma \sum_{z_{s'}} \hat{p}(\mathbf{z}_{s'}|\mathbf{z}_{sa}) \max_{a'} \hat{Q}_{n-1}(\mathbf{z}_{s'a'}) && (75)\\ &= \hat{Q}_n(\mathbf{z}_{sa}). && (76) \end{aligned}$$

Thus there exists a function $\hat{Q}(g(\mathbf{z}_s,a)) = Q^*(s,a)$, consequently, there exists an optimal policy $\hat{\pi}^*(a|\mathbf{z}_s) = \text{argmax}_a \hat{Q}(s,a)$.

■

B EXPERIMENTAL DETAILS

B.1 HYPERPARAMETERS

Table 3: **MR.Q Hyperparameters.** Hyperparameters values are kept fixed across all benchmarks.

	Hyperparameter	Value
	Dynamics loss weight $\lambda_{\text{Dynamics}}$	1
	Reward loss weight λ_{Reward}	0.1
	Terminal loss weight $\lambda_{\text{Terminal}}$	0.1
	Pre-activation loss weight $\lambda_{\text{pre-activ}}$	1e – 5
	Encoder horizon H_{Enc}	5
	Multi-step returns horizon H_Q	3
TD3 (Fujimoto et al., 2018)	Target policy noise σ	$\mathcal{N}(0, 0.2^2)$
	Target policy noise clipping c	$(-0.3, 0.3)$
LAP (Fujimoto et al., 2020)	Probability smoothing α	0.4
	Minimum priority	1
Exploration	Initial random exploration time steps	10k
	Exploration noise	$\mathcal{N}(0, 0.2^2)$
Common	Discount factor γ	0.99
	Replay buffer capacity	1M
	Mini-batch size	256
	Target update frequency T_{target}	250
	Replay ratio	1
Encoder Network	Optimizer	AdamW (Loshchilov & Hutter, 2019)
	Learning rate	1e – 4
	Weight decay	1e – 4
	$\mathbf{z}_s$ dim	512
	$\mathbf{z}_{sa}$ dim	512
	$\mathbf{z}_a$ dim (only used within architecture)	256
	Hidden dim	512
	Activation function	ELU (Clevert et al., 2015)
	Weight initialization	Xavier uniform (Glorot & Bengio, 2010)
	Bias initialization	0
	Reward bins	65
	Reward range	$[-10, 10]$ (effective: $[-22\text{k}, 22\text{k}]$)
Value Network	Optimizer	AdamW
	Learning rate	3e – 4
	Hidden dim	512
	Activation function	ELU
	Weight initialization	Xavier uniform
	Bias initialization	0
	Gradient clip norm	20
Policy Network	Optimizer	AdamW
	Learning rate	3e – 4
	Hidden dim	512
	Activation function	ReLU
	Weight initialization	Xavier uniform
	Bias initialization	0
	Gumbel-Softmax τ (Jang et al., 2017)	10

B.2 NETWORK ARCHITECTURE

This section describes the networks used in our method using PyTorch code blocks (Paszke et al., 2019). The state encoder and state-action encoder are described as separate networks for clarity but are trained end-to-end as a single network. The value and policy networks are trained independently from the encoders.

Preamble

```
import torch.nn as nn
import torch.nn.functional as F

zs_dim = 512
za_dim = 256
zsa_dim = 512

def ln_activ(self, x):
    x = F.layer_norm(x, (x.shape[-1],))
    return self.activ(x)
```

State Encoder f Network

For image inputs, four convolutional layers are used, each with 32 output channels, kernel size of 3, strides of $(2, 2, 2, 1)$, and ELU activations (Clevert et al., 2015). The convolutional layers are followed by a linear layer taking in the flattened output followed by LayerNorm (Ba et al., 2016) and a final ELU activation.

For vector inputs, a three layer multilayer perceptron (MLP) is used, with hidden dimension 512 and LayerNorm followed by ELU activations after each layer.

The resulting state embedding $\mathbf{z}_s$ is trained end-to-end with the state-action encoder. It is also used downstream by the policy network (without propagating gradients).

```
if image_observation_space:
    self.zs_cnn1 = nn.Conv2d(state_channels, 32, 3, stride=2)
    self.zs_cnn2 = nn.Conv2d(32, 32, 3, stride=2)
    self.zs_cnn3 = nn.Conv2d(32, 32, 3, stride=2)
    self.zs_cnn4 = nn.Conv2d(32, 32, 3, stride=1)
    # Assumes 84 x 84 input
    self.zs_lin = nn.Linear(1568, zs_dim)
else:
    self.zs_mlp1 = nn.Linear(state_dim, 512)
    self.zs_mlp2 = nn.Linear(512, 512)
    self.zs_mlp3 = nn.Linear(512, zs_dim)

self.activ = F.elu

def cnn_forward(self, state):
    state = state/255. - 0.5
    zs = self.activ(self.zs_cnn1(state))
    zs = self.activ(self.zs_cnn2(zs))
    zs = self.activ(self.zs_cnn3(zs))
    zs = self.activ(self.zs_cnn4(zs))
    zs = zs.reshape(batch_size, 1568)
    return ln_activ(self.zs_lin(zs))

def mlp_forward(self, state):
    zs = self.ln_activ(self.zs_mlp1(state))
    zs = self.ln_activ(self.zs_mlp2(zs))
    return self.ln_activ(self.zs_mlp3(zs))
```

State-Action Encoder g Network

Action input is processed by a linear layer followed by an ELU activation. Afterwards, the processed action is concatenated with the state embedding and processed by a three layer MLP with hidden dimension 512, and LayerNorm followed by ELU activations after the first two layers.

The resulting state-action embedding $\mathbf{z}_{sa}$ is used by a linear layer to make predictions about reward, the next state embedding, and the terminal signal. It is also used downstream by the value network (without propagating gradients).

```
self.za = nn.Linear(action_dim, za_dim)
self.zsa1 = nn.Linear(zs_dim + za_dim, 512)
self.zsa2 = nn.Linear(512, 512)
self.zsa3 = nn.Linear(512, zsa_dim)
self.pred = nn.Linear(zsa_dim, output_dim)
self.activ = F.elu

def forward(self, zs, action):
    za = self.activ(self.za(action))
    zsa = torch.cat([zs, za], 1)
    zsa = self.ln_activ(self.zsa1(zsa))
    zsa = self.ln_activ(self.zsa2(zsa))
    zsa = self.zsa3(zsa)
    return self.pred(zsa), zsa
```

Value Q Networks

The value network is a four layer MLP with hidden dimension 512, and LayerNorm followed by ELU activations after the first three layers.

Two value networks are used with the same network and forward pass.

```
self.l1 = nn.Linear(zsa_dim, 512)
self.l2 = nn.Linear(512, 512)
self.l3 = nn.Linear(512, 512)
self.l4 = nn.Linear(512, 1)
self.activ = F.elu

def forward(self, zsa):
    q = self.ln_activ(self.l1(zsa))
    q = self.ln_activ(self.l2(q))
    q = self.ln_activ(self.l3(q))
    return self.l4(q)
```

Policy π Network

The policy network is a three layer MLP with hidden dimension 512, and LayerNorm followed by ReLU activations after the first two layers.

For discrete actions, the final activation is the Gumbel Softmax with τ = 10. For continous actions, the final activation is a tanh function.

```
self.l1 = nn.Linear(zs_dim, 512)
self.l2 = nn.Linear(hdim, 512)
self.l3 = nn.Linear(512, action_dim)
self.activ = F.relu

if discrete_action_space:
    self.final_activ = partial(F.gumbel_softmax, tau=10)
else:
    self.final_activ = torch.tanh

def forward(self, zs):
    a = self.ln_activ(self.l1(zs))
    a = self.ln_activ(self.l2(a))
    return self.final_activ(self.l3(a))
```

B.3 ENVIRONMENTS

All main experiments were run for 10 seeds (the design study is based on 5 seeds). Evaluations are based on the average performance over 10 episodes, measured every 5k time steps for Gym and DM control and every 100k time steps for Atari.

Gym - Locomotion. For the gym locomotion tasks (Todorov et al., 2012; Brockman et al., 2016; Towers et al., 2024), we choose the five most common environments that appear in prior work (Fujimoto et al., 2018; 2024; Haarnoja et al., 2018; Kuznetsov et al., 2020). We use the -v4 version. No preprocessing is applied. When aggregating scores, we use normalize with the TD3 scores obtained from TD7 (Fujimoto et al., 2024):

$$\text{TD3-Normalized}(x) := \frac{x - \text{random score}}{\text{TD3 score} - \text{random score}}. \tag{77}$$

	Random	TD3
Ant-v4	-70.288	3942
HalfCheetah-v4	-289.415	10574
Hopper-v4	18.791	3226
Humanoid-v4	120.423	5165
Walker2d-v4	2.791	3946

DM Control Suite. For the DM control suite (Tassa et al., 2018), we choose the 28 default environments that appear either in the evaluation of TD-MPC2 or DreamerV3. We omit any custom environments included by the TD-MPC2 authors. The same subset of tasks are used in the evaluation of proprioceptive and visual control. Like prior work, for both observation spaces, we use an action repeat of 2 (Hansen et al., 2024). For visual control, the state (network input) is composed of the previous 3 observations which are resized to 84×84 pixels in RGB format (Tassa et al., 2018).

Atari. For the Atari games (Bellemare et al., 2013; Brockman et al., 2016; Towers et al., 2024), we use the 57 games in the Atari-57 benchmark that appears in prior work (Hessel et al., 2018; Schrittwieser et al., 2020; Badia et al., 2020; Hafner et al., 2023). For DQN and Rainbow, two games (Defender and Surround) are missing from the Dopamine framework (Castro et al., 2018) and are omitted. We use the -v5 version. For MR.Q, we use the common preprocessing steps (Mnih et al., 2015; Machado et al., 2018; Castro et al., 2018), where an action repeat of 4 is used and the observations are grayscaled, resized to 84×84 pixels and set to the max between the 3rd and 4th frame. The state (network input) is composed of the previous 4 observations.

Consider the 16 frame sequence used by a single state, where f_i is the ith grayscaled and resized frame and o_j is the jth observation set to the max of two frames

$$\overbrace{f_0, f_1, \underbrace{f_2, f_3}_{o_0=\max(f_2,f_3)}}^{\text{action } a_0}, \overbrace{f_4, f_5, \underbrace{f_6, f_7}_{o_1=\max(f_6,f_7)}}^{\text{action } a_1}, \overbrace{f_8, f_9, \underbrace{f_{10}, f_{11}}_{o_2=\max(f_{10},f_{11})}}^{\text{action } a_2}, \overbrace{f_{12}, f_{13}, \underbrace{f_{14}, f_{15}}_{o_3=\max(f_{14},f_{15})}}^{\text{action } a_3}, \tag{78}$$

then the state is defined as follows:

$$s = \begin{bmatrix} o_0 = \max(f_2, f_3) \\ o_1 = \max(f_6, f_7) \\ o_2 = \max(f_{10}, f_{11}) \\ o_3 = \max(f_{14}, f_{15}) \end{bmatrix}. \tag{79}$$

When aggregating scores, we normalize with Human scores obtained from (Wang et al., 2016):

$$\text{Human-Normalized}(x) := \frac{x - \text{random score}}{\text{Human score} - \text{random score}}. \tag{80}$$

	Random	Human
Alien	227.8	7127.7
Amidar	5.8	1719.5
Assault	222.4	742.0
Asterix	210.0	8503.3
Asteroids	719.1	47388.7
Atlantis	12850.0	29028.1
BankHeist	14.2	753.1
BattleZone	2360.0	37187.5
BeamRider	363.9	16926.5
Berzerk	123.7	2630.4
Bowling	23.1	160.7
Boxing	0.1	12.1
Breakout	1.7	30.5
Centipede	2090.9	12017.0
ChopperCommand	811.0	7387.8
CrazyClimber	10780.5	35829.4
Defender (not used)	2874.5	18688.9
DemonAttack	152.1	1971.0
DoubleDunk	-18.6	-16.4
Enduro	0.0	860.5
FishingDerby	-91.7	-38.7
Freeway	0.0	29.6
Frostbite	65.2	4334.7
Gopher	257.6	2412.5
Gravitar	173.0	3351.4
Hero	1027.0	30826.4
IceHockey	-11.2	0.9
Jamesbond	29.0	302.8
Kangaroo	52.0	3035.0
Krull	1598.0	2665.5
KungFuMaster	258.5	22736.3
MontezumaRevenge	0.0	4753.3
MsPacman	307.3	6951.6
NameThisGame	2292.3	8049.0
Phoenix	761.4	7242.6
Pitfall	-229.4	6463.7
Pong	-20.7	14.6
PrivateEye	24.9	69571.3
Qbert	163.9	13455.0
Riverraid	1338.5	17118.0
RoadRunner	11.5	7845.0
Robotank	2.2	11.9
Seaquest	68.4	42054.7
Skiing	-17098.1	-4336.9
Solaris	1236.3	12326.7
SpaceInvaders	148.0	1668.7
StarGunner	664.0	10250.0
Surround (not used)	-10.0	6.5
Tennis	-23.8	-8.3
TimePilot	3568.0	5229.2
Tutankham	11.4	167.6
UpNDown	533.4	11693.2
Venture	0.0	1187.5
VideoPinball	16256.9	17667.9
WizardOfWor	563.5	4756.5
YarsRevenge	3092.9	54576.9
Zaxxon	32.5	9173.3

B.4 BASELINES

DreamerV3. (Hafner et al., 2023). Results for Gym and DMC were obtained by re-running the authors' code (https://github.com/danijar/dreamerv3 - Commit 251910d04c9f38dd9dc385775bb0d6efa0e57a95) over 10 seeds, using the author-suggested hyperparameters from the DMC benchmark. Code was modified slightly to match our evaluation protocol. Atari results are based on the authors' reported results.

DrQ-v2. (Yarats et al., 2022). We use the authors' reported results whenever possible. For missing any results, we re-ran the authors' code (https://github.com/facebookresearch/drqv2 - Commit c0c650b76c6e5d22a7eb5f2edffd1440fe94f8ef) for 10 seeds.

DQN. (Mnih et al., 2015). Results were obtained from the Dopamine framework (Castro et al., 2018).

PPO. (Schulman et al., 2017). Results were gathered using Stable Baselines 3 (Raffin et al., 2021) and default hyperparameters. The default MLP policy was used for Gym and DMC-proprioceptive and the default CNN policy was used for DMC-visual and Atari.

Rainbow. (Hessel et al., 2018). Results were obtained from the Dopamine framework (Castro et al., 2018).

TD-MPC2. (Hansen et al., 2024). Results for DMC were obtained by re-running the authors' code on their main branch (https://github.com/nicklashansen/tdmpc2 - Commit 5f6fadec0fec78304b4b53e8171d348b58cac486). As the Gym environments include a termination signal, results for Gym were obtained by running their episodic branch (https://github.com/nicklashansen/tdmpc2/tree/episodic-rl - Commit 3789fcd5b872079ad610fa3299ff47c3a427a04a). All experiments were run for 10 seeds and use the default author-suggested hyperparameters for all tasks.

TD7. (Fujimoto et al., 2024). Results for Gym were obtained from the authors. Results for DMC were obtained by re-running the authors' code (https://github.com/sfujim/TD7 - Commit c1c280de1513f474488061b4cf39642b75dd84bd) using our setup for DMC. All experiments use 10 seeds and use the default author-suggested hyperparameters from the Gym benchmark.

B.5 SOFTWARE VERSIONS

- Gymnasium 0.29.1 (Towers et al., 2024)
- MuJoCo 3.2.2 (Todorov et al., 2012)
- NumPy 2.1.1 (Harris et al., 2020)
- Python 3.11.8 (Van Rossum & Drake Jr, 1995)
- PyTorch 2.4.1 (Paszke et al., 2019)

C Complete Main Results

C.1 Gym

Table 4: **Gym - Locomotion final results.** Final average performance at 1M time steps over 10 seeds. The [bracketed values] represent a 95% bootstrap confidence interval. The aggregate mean, median and interquartile mean (IQM) are computed over the TD3-normalized score (see Appendix B.3).

Task	TD7	PPO	TD-MPC2	DreamerV3	MR.Q
Ant	8509 [8168, 8844]	1584 [1360, 1815]	4751 [2988, 6145]	1947 [1076, 2813]	6989 [6203, 7617]
HalfCheetah	17433 [17301, 17559]	1744 [1523, 2118]	15078 [14065, 15932]	5502 [3717, 7123]	13305 [11841, 14140]
Hopper	3511 [3236, 3736]	3022 [2633, 3339]	2081 [1197, 2921]	2666 [2106, 3210]	2684 [2154, 3269]
Humanoid	7428 [7304, 7553]	477 [436, 518]	6071 [5770, 6333]	4217 [2785, 5523]	7259 [5080, 9336]
Walker2d	6096 [5621, 6547]	2487 [1907, 3022]	3008 [1706, 4321]	4519 [3692, 5244]	6629 [5816, 7493]
Mean	1.57 [1.54, 1.60]	0.45 [0.41, 0.49]	1.04 [0.92, 1.15]	0.76 [0.67, 0.86]	1.39 [1.27, 1.49]
Median	1.55 [1.45, 1.63]	0.41 [0.36, 0.47]	1.18 [0.83, 1.22]	0.81 [0.58, 0.93]	1.42 [1.19, 1.71]
IQM	1.54 [1.50, 1.58]	0.41 [0.36, 0.46]	1.05 [0.89, 1.19]	0.72 [0.62, 0.85]	1.45 [1.27, 1.58]

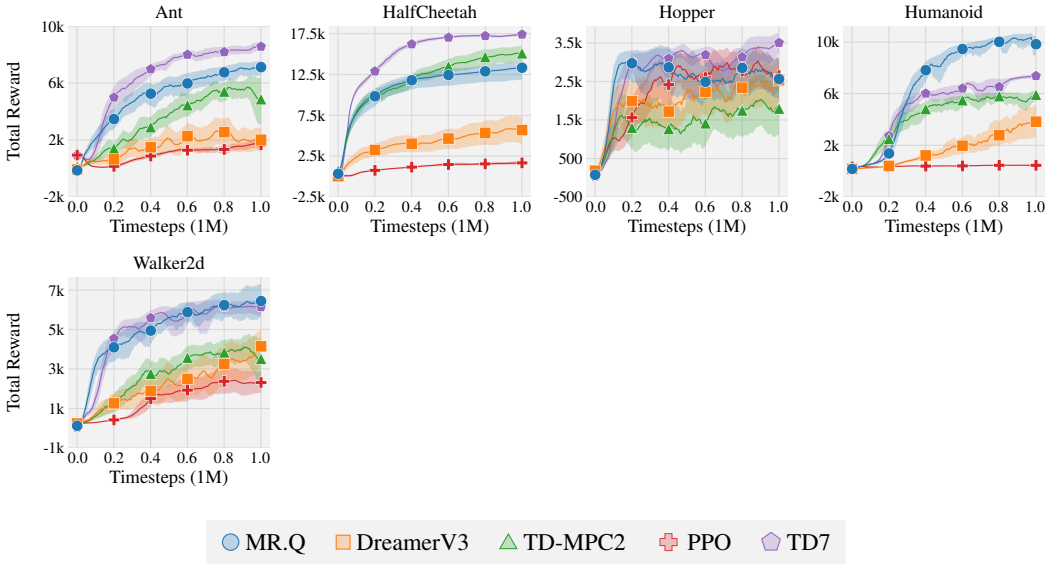

Figure 3: **Gym - Locomotion learning curves.** Results are over 10 seeds. The shaded area captures a 95% boostrap confidence interval.

C.2 DMC - PROPRIOCEPTIVE

Table 5: **DMC - Proprioceptive final results.** Final average performance at 500k time steps (1M time steps in the original environment due to action repeat) over 10 seeds. The [bracketed values] represent a 95% bootstrap confidence interval. The aggregate mean, median and interquartile mean (IQM) are computed over the default reward.

Task	TD7	PPO	TD-MPC2	DreamerV3	MR.Q
acrobot-swingup	58 [38, 75]	39 [33, 45]	584 [551, 615]	230 [193, 266]	567 [523, 616]
ball_in_cup-catch	983 [981, 985]	769 [689, 841]	984 [982, 986]	968 [965, 973]	981 [979, 984]
cartpole-balance	999 [998, 1000]	999 [1000, 1000]	996 [995, 998]	998 [997, 1000]	999 [999, 1000]
cartpole-balance_sparse	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]	999 [1000, 1000]	1000 [1000, 1000]
cartpole-swingup	869 [866, 873]	776 [661, 853]	875 [870, 880]	736 [591, 838]	866 [866, 866]
cartpole-swingup_sparse	573 [333, 806]	391 [159, 625]	845 [839, 849]	702 [560, 792]	798 [780, 818]
cheetah-run	821 [642, 913]	269 [247, 295]	917 [915, 920]	699 [655, 744]	914 [911, 917]
dog-run	69 [36, 101]	26 [26, 28]	265 [166, 342]	4 [4, 5]	569 [547, 595]
dog-stand	582 [432, 741]	129 [122, 139]	506 [266, 715]	22 [20, 27]	967 [960, 975]
dog-trot	21 [13, 30]	31 [30, 34]	407 [265, 530]	10 [6, 17]	877 [845, 898]
dog-walk	52 [19, 116]	40 [37, 43]	486 [240, 704]	17 [15, 21]	916 [908, 924]
finger-spin	335 [99, 596]	459 [420, 497]	986 [986, 988]	666 [577, 763]	937 [917, 956]
finger-turn_easy	912 [774, 983]	182 [153, 211]	979 [975, 983]	906 [883, 927]	953 [931, 974]
finger-turn_hard	470 [199, 727]	58 [35, 79]	947 [916, 977]	864 [812, 900]	950 [910, 974]
fish-swim	86 [64, 120]	103 [84, 128]	659 [615, 706]	813 [808, 819]	792 [773, 810]
hopper-hop	87 [25, 160]	10 [0, 23]	425 [368, 500]	116 [66, 165]	251 [195, 301]
hopper-stand	670 [466, 829]	128 [56, 216]	952 [944, 958]	747 [669, 806]	951 [948, 955]
humanoid-run	57 [23, 92]	0 [1, 1]	181 [121, 231]	0 [1, 1]	200 [170, 236]
humanoid-stand	317 [117, 516]	5 [5, 6]	658 [506, 745]	5 [5, 6]	868 [822, 903]
humanoid-walk	176 [42, 320]	1 [1, 2]	754 [725, 791]	1 [1, 2]	662 [610, 724]
pendulum-swingup	500 [251, 743]	115 [70, 164]	846 [830, 862]	774 [740, 802]	748 [597, 829]
quadruped-run	645 [567, 713]	144 [122, 170]	942 [938, 947]	130 [92, 169]	947 [940, 954]
quadruped-walk	949 [939, 957]	122 [103, 142]	963 [959, 967]	193 [137, 243]	963 [959, 967]
reacher-easy	970 [951, 982]	367 [188, 558]	983 [980, 986]	966 [964, 970]	983 [983, 985]
reacher-hard	898 [861, 936]	125 [40, 234]	960 [936, 979]	919 [864, 955]	977 [975, 980]
walker-run	804 [783, 825]	97 [91, 104]	854 [851, 859]	510 [430, 588]	793 [765, 815]
walker-stand	983 [974, 989]	431 [363, 495]	991 [990, 994]	941 [934, 948]	988 [987, 990]
walker-walk	977 [975, 980]	283 [253, 312]	981 [979, 984]	898 [875, 919]	978 [978, 980]
Mean	566 [544, 590]	254 [241, 267]	783 [769, 797]	530 [520, 539]	835 [829, 842]
Median	613 [548, 718]	127 [112, 145]	896 [893, 899]	700 [644, 741]	927 [914, 934]
IQM	612 [569, 657]	154 [135, 167]	868 [860, 880]	577 [557, 594]	907 [903, 914]

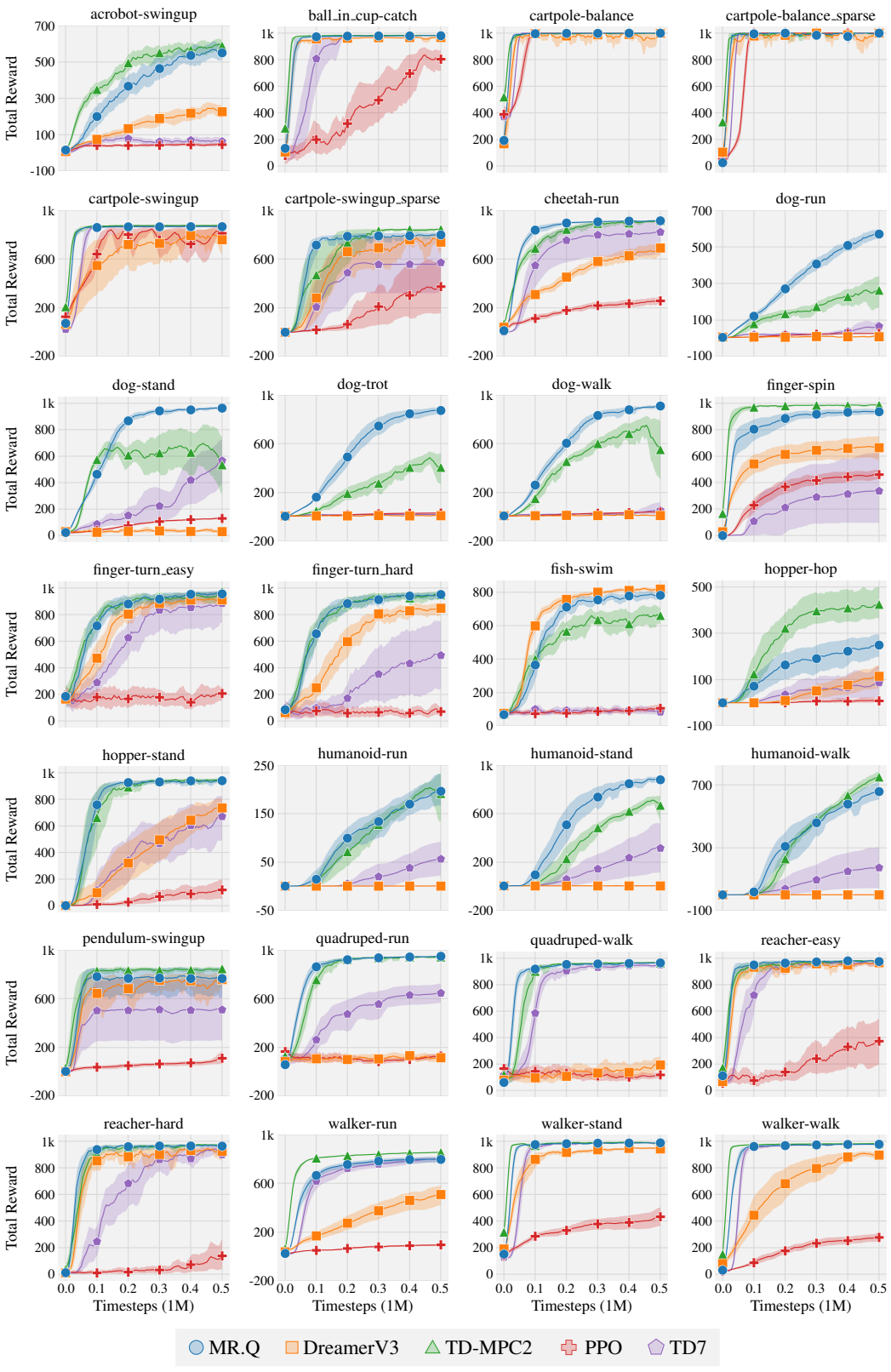

Figure 4: **DMC - Proprioceptive learning curves.** Time steps consider the number of environment interactions, where 500k time steps equals 1M frames in the original environment. Results are over 10 seeds. The shaded area captures a 95% boostrap confidence interval.

C.3 DMC - VISUAL

Table 6: **DMC - Visual final results.** Final average performance at 500k time steps (1M time steps in the original environment due to action repeat) over 10 seeds. The [bracketed values] represent a 95% bootstrap confidence interval. The aggregate mean, median and interquartile mean (IQM) are computed over the default reward.

Task	DrQ-v2	PPO	TD-MPC2	DreamerV3	MR.Q
acrobot-swingup	168 [127, 219]	2 [1, 4]	197 [179, 217]	121 [106, 145]	287 [254, 316]
ball_in_cup-catch	909 [821, 973]	105 [5, 282]	932 [899, 961]	971 [969, 973]	977 [975, 980]
cartpole-balance	993 [990, 996]	353 [231, 485]	972 [948, 991]	998 [997, 1000]	999 [999, 999]
cartpole-balance_sparse	962 [887, 1000]	487 [233, 751]	1000 [1000, 1000]	999 [999, 1000]	1000 [1000, 1000]
cartpole-swingup	864 [854, 873]	596 [437, 723]	690 [521, 813]	725 [603, 807]	868 [860, 875]
cartpole-swingup_sparse	774 [741, 805]	0 [0, 0]	636 [404, 804]	547 [351, 726]	797 [777, 816]
cheetah-run	728 [701, 753]	155 [110, 210]	431 [267, 556]	618 [576, 661]	775 [752, 807]
dog-run	10 [9, 12]	11 [9, 14]	14 [10, 18]	9 [6, 14]	60 [44, 80]
dog-stand	43 [37, 49]	51 [48, 56]	117 [72, 148]	61 [30, 92]	216 [201, 232]
dog-trot	14 [11, 18]	13 [12, 15]	20 [14, 25]	14 [13, 16]	65 [55, 79]
dog-walk	22 [18, 29]	16 [14, 18]	22 [17, 28]	11 [11, 12]	77 [71, 83]
finger-spin	860 [787, 922]	241 [107, 377]	786 [492, 984]	656 [544, 765]	965 [938, 982]
finger-turn_easy	503 [399, 615]	189 [144, 233]	562 [317, 779]	491 [447, 542]	953 [927, 974]
finger-turn_hard	223 [121, 340]	60 [1, 120]	903 [870, 940]	494 [401, 571]	932 [905, 957]
fish-swim	84 [65, 107]	77 [64, 92]	43 [21, 64]	90 [84, 96]	79 [68, 93]
hopper-hop	224 [170, 278]	0 [0, 0]	187 [119, 238]	205 [125, 287]	270 [230, 315]
hopper-stand	917 [903, 931]	1 [0, 2]	582 [321, 794]	888 [875, 900]	852 [703, 930]
humanoid-run	1 [1, 1]	1 [1, 1]	0 [1, 1]	1 [1, 1]	1 [1, 2]
humanoid-stand	6 [7, 7]	6 [6, 7]	5 [5, 7]	5 [5, 7]	7 [7, 8]
humanoid-walk	2 [2, 2]	1 [1, 1]	1 [1, 2]	1 [2, 2]	2 [2, 3]
pendulum-swingup	838 [813, 861]	0 [0, 1]	748 [574, 850]	761 [709, 807]	829 [816, 842]
quadruped-run	459 [412, 507]	118 [98, 139]	262 [184, 330]	328 [255, 397]	498 [476, 522]
quadruped-walk	750 [699, 796]	149 [113, 184]	246 [179, 310]	316 [260, 379]	833 [797, 867]
reacher-easy	938 [903, 973]	113 [55, 192]	956 [932, 978]	735 [678, 796]	979 [978, 982]
reacher-hard	705 [580, 831]	10 [0, 30]	911 [867, 946]	338 [227, 461]	965 [945, 977]
walker-run	546 [475, 612]	39 [35, 44]	665 [566, 719]	669 [615, 708]	615 [571, 655]
walker-stand	980 [977, 984]	253 [210, 310]	937 [907, 962]	969 [966, 973]	980 [977, 985]
walker-walk	766 [489, 957]	47 [40, 56]	958 [952, 965]	942 [936, 949]	970 [968, 973]
Mean	510 [497, 523]	110 [98, 125]	492 [471, 512]	463 [452, 475]	602 [595, 608]
Median	626 [528, 665]	49 [32, 53]	572 [419, 654]	493 [420, 532]	813 [779, 822]
IQM	545 [519, 564]	58 [46, 67]	501 [458, 537]	452 [430, 473]	692 [678, 703]

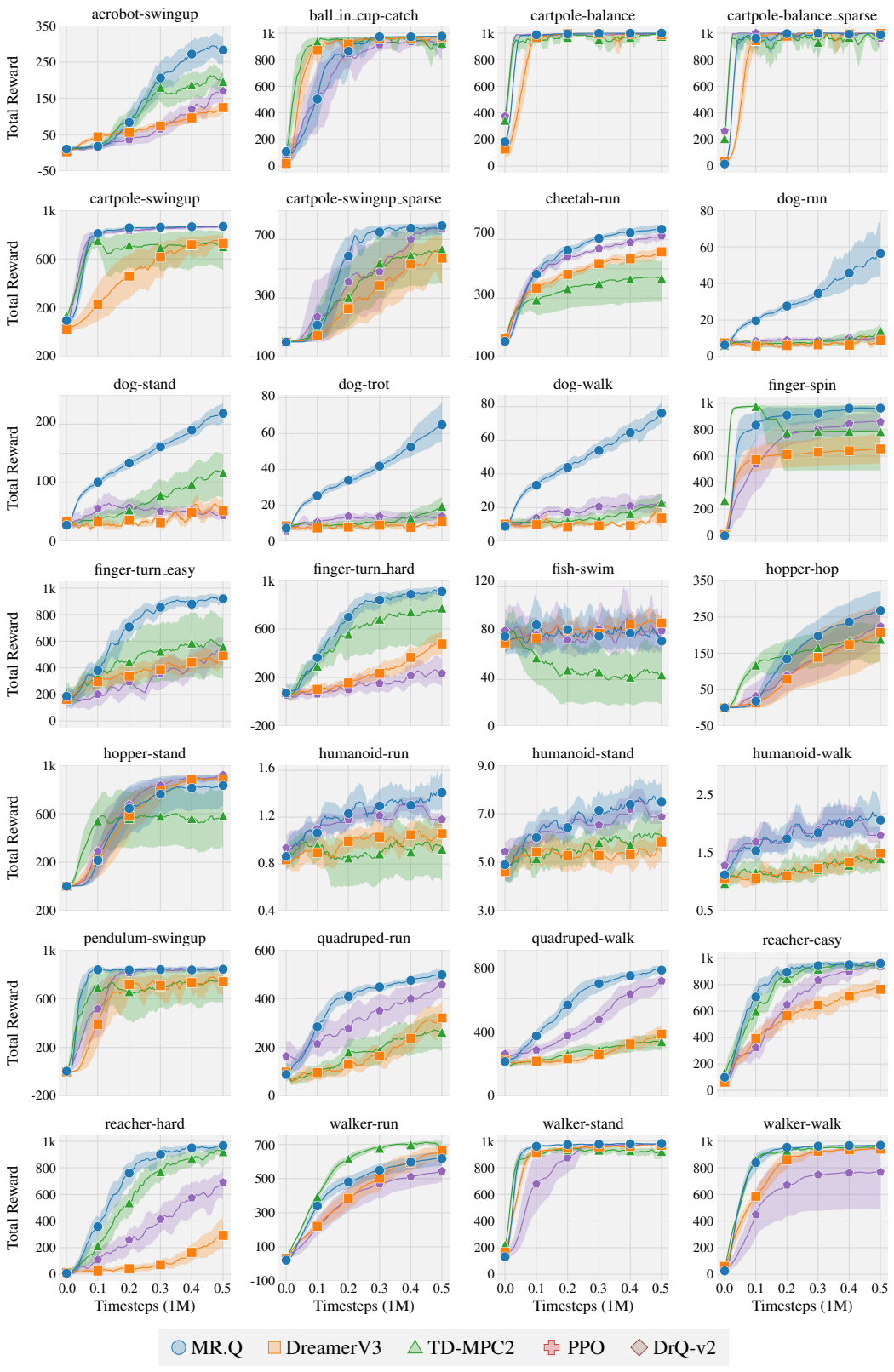

Figure 5: **DMC - Visual learning curves.** Time steps consider the number of environment interactions, where 500k time steps equals 1M frames in the original environment. Results are over 10 seeds. The shaded area captures a 95% boostrap confidence interval.

C.4 ATARI

Table 7: **Atari final results.** Final average performance at 2.5M time steps (10M time steps in the original environment due to action repeat) over 10 seeds. The [bracketed values] represent a 95% bootstrap confidence interval. The aggregate mean, median and interquartile mean (IQM) are computed over the human-normalized score.

Task	DQN	Rainbow	PPO	DreamerV3	MR.Q
Alien	925 [880, 968]	1220 [1191, 1265]	320 [257, 386]	4838 [3863, 5813]	2303 [1621, 3051]
Amidar	178 [170, 186]	301 [281, 330]	126 [86, 162]	470 [417, 524]	604 [531, 659]
Assault	988 [955, 1011]	1430 [1392, 1480]	423 [266, 567]	3518 [2966, 4179]	1256 [1194, 1363]
Asterix	2381 [2317, 2469]	2699 [2598, 2778]	296 [220, 391]	7319 [6095, 8386]	3330 [3094, 3538]
Asteroids	423 [408, 436]	754 [711, 818]	206 [182, 230]	1359 [1237, 1495]	798 [587, 997]
Atlantis	7365 [6881, 7745]	80837 [51775, 122485]	2000 [2000, 2000]	664529 [208350, 973362]	605100 [497110, 723234]
BankHeist	474 [448, 496]	895 [890, 901]	187 [42, 381]	801 [691, 1002]	589 [354, 822]
BattleZone	3598 [3254, 3878]	20209 [16966, 22358]	2200 [1430, 3040]	22599 [20973, 24554]	16360 [9480, 23720]
BeamRider	869 [716, 1065]	5982 [5690, 6268]	479 [353, 597]	5635 [3477, 7962]	1896 [1824, 1971]
Berzerk	488 [464, 508]	443 [413, 487]	384 [311, 465]	758 [691, 823]	458 [401, 517]
Bowling	29 [27, 32]	44 [37, 51]	51 [38, 60]	101 [70, 138]	66 [63, 69]
Boxing	37 [31, 44]	68 [66, 71]	-3 [-6, 0]	97 [97, 99]	97 [96, 99]
Breakout	21 [19, 25]	41 [40, 44]	9 [8, 11]	137 [110, 162]	60 [28, 100]
Centipede	2832 [2418, 3204]	4992 [4784, 5138]	4239 [2280, 6283]	20067 [17102, 22563]	15642 [13910, 17500]
ChopperCommand	997 [972, 1023]	2265 [2165, 2371]	688 [473, 884]	15172 [13336, 17153]	4690 [3650, 5852]
CrazyClimber	64611 [46520, 79414]	103539 [99685, 107101]	896 [172, 1750]	132811 [128446, 135761]	116194 [110720, 122764]
Defender	-	-	1333 [711, 1996]	34187 [29985, 39261]	40624 [36116, 45132]
DemonAttack	1503 [1282, 1690]	2477 [2268, 2658]	139 [115, 162]	4836 [3443, 6231]	7408 [5390, 9596]
DoubleDunk	-18 [-20, -18]	-18 [-19, -19]	-1 [-3, -1]	21 [20, 22]	-7 [-8, -6]
Enduro	589 [567, 617]	1601 [1557, 1635]	13 [9, 17]	476 [177, 784]	1837 [1731, 1946]
FishingDerby	-42 [-62, -17]	10 [4, 15]	-89 [-91, -87]	40 [32, 47]	12 [7, 19]
Freeway	8 [0, 19]	32 [32, 32]	15 [11, 18]	19 [6, 32]	32 [32, 33]
Frostbite	269 [239, 296]	2510 [2040, 2823]	245 [230, 258]	5183 [2075, 8291]	3654 [1674, 5567]
Gopher	1470 [1305, 1590]	4279 [4139, 4425]	126 [80, 174]	38711 [26654, 48506]	17514 [11903, 27065]
Gravitar	167 [153, 183]	202 [184, 218]	63 [33, 100]	831 [763, 904]	355 [315, 396]
Hero	2679 [2401, 2945]	9323 [7914, 10863]	1741 [1172, 2318]	20582 [19840, 21598]	13503 [13370, 13646]
IceHockey	-9 [-10, -9]	-5 [-6, -5]	-8 [-10, -8]	14 [13, 16]	0 [-1, 2]
Jamesbond	47 [43, 52]	514 [510, 520]	85 [62, 106]	836 [566, 1117]	614 [558, 686]
Kangaroo	539 [525, 553]	5501 [3862, 7151]	402 [262, 520]	8825 [5234, 12302]	10166 [7772, 12548]
Krull	4229 [3942, 4490]	5972 [5905, 6043]	421 [142, 764]	23092 [14679, 28172]	9327 [8954, 9767]
KungFuMaster	15997 [13182, 18793]	18074 [16146, 20737]	52 [15, 90]	70703 [51278, 95834]	28080 [24354, 32066]
MontezumaRevenge	0 [0, 0]	0 [0, 0]	0 [0, 0]	1310 [598, 2178]	160 [0, 320]
MsPacman	2187 [2121, 2247]	2347 [2293, 2403]	457 [352, 581]	4484 [3608, 5476]	4050 [3209, 4893]
NameThisGame	4000 [3814, 4187]	8604 [8278, 8931]	1084 [672, 1505]	15742 [14466, 17102]	10835 [9949, 11848]
Phoenix	4948 [4270, 5627]	4830 [4708, 4977]	101 [80, 120]	15827 [14924, 16429]	5043 [4668, 5400]
Pitfall	-60 [-88, -35]	-14 [-30, -6]	-16 [-37, -2]	0 [0, 0]	0 [0, 0]
Pong	-4 [-13, 3]	15 [14, 16]	-5 [-8, -3]	16 [16, 17]	18 [18, 19]
PrivateEye	118 [78, 181]	111 [78, 166]	-17 [-591, 837]	3046 [975, 5103]	117 [94, 159]
Qbert	1658 [1246, 2152]	5353 [4373, 6783]	484 [395, 575]	16807 [16060, 17515]	6073 [3630, 8126]
Riverraid	3198 [3167, 3221]	4272 [4060, 4428]	1045 [820, 1229]	9160 [8181, 10124]	10658 [8490, 13043]
RoadRunner	27980 [27269, 28692]	33412 [32326, 34449]	723 [446, 945]	66453 [40558, 103420]	49924 [45862, 53312]
Robotank	4 [4, 5]	19 [18, 20]	4 [2, 6]	51 [47, 55]	12 [12, 14]
Seaquest	299 [277, 318]	1641 [1621, 1662]	250 [216, 285]	3416 [2665, 4381]	2434 [2362, 2512]
Skiing	-19568 [-19793, -19349]	-24070 [-25320, -22735]	-27901 [-30000, -23704]	-30043 [-30433, -29756]	-30000 [-30000, -30000]
Solaris	1645 [1480, 1804]	1289 [1143, 1465]	0 [0, 2]	2340 [1882, 2861]	1769 [1075, 2598]
SpaceInvaders	663 [649, 675]	743 [721, 764]	294 [232, 350]	1433 [1044, 1943]	680 [590, 763]
StarGunner	692 [663, 719]	1488 [1472, 1504]	415 [325, 504]	2090 [1660, 2668]	1036 [1012, 1068]
Surround	-	-	-9 [-10, -10]	5 [4, 7]	-3 [-5, -2]
Tennis	-21 [-24, -19]	-1 [-2, 0]	-20 [-22, -19]	-3 [-11, 0]	1 [-5, 11]
TimePilot	1539 [1474, 1618]	2703 [2629, 2787]	548 [450, 674]	7779 [3128, 13016]	3916 [3498, 4334]
Tutankham	112 [98, 123]	179 [164, 191]	29 [17, 42]	253 [239, 269]	160 [148, 172]
UpNDown	7669 [7113, 8132]	12397 [11489, 13312]	595 [422, 734]	284807 [185845, 389509]	58944 [15055, 102834]
Venture	25 [6, 43]	19 [14, 25]	2 [0, 6]	0 [0, 0]	0 [0, 0]
VideoPinball	5129 [4612, 5649]	26245 [23476, 29027]	1005 [0, 2515]	22345 [20669, 23955]	67995 [31863, 116012]
WizardOfWor	481 [395, 542]	2213 [1827, 2568]	225 [185, 266]	7086 [6464, 7743]	2632 [1820, 3652]
YarsRevenge	9426 [9177, 9656]	10708 [10388, 11071]	1891 [1010, 2856]	62209 [57772, 66741]	30245 [28033, 32597]
Zaxxon	112 [16, 230]	3661 [3131, 4192]	0 [0, 0]	17347 [15394, 19615]	7674 [5750, 8736]
Mean	0.25 [0.22, 0.27]	1.08 [1.02, 1.13]	-0.09 [-0.11, -0.06]	3.74 [3.22, 4.17]	2.76 [2.26, 3.38]
Median	0.12 [0.10, 0.12]	0.40 [0.39, 0.48]	0.01 [0.00, 0.01]	1.25 [1.11, 1.52]	0.78 [0.62, 0.89]
IQM	0.17 [0.15, 0.18]	0.61 [0.60, 0.62]	0.02 [0.01, 0.02]	1.46 [1.35, 1.51]	0.89 [0.83, 0.92]

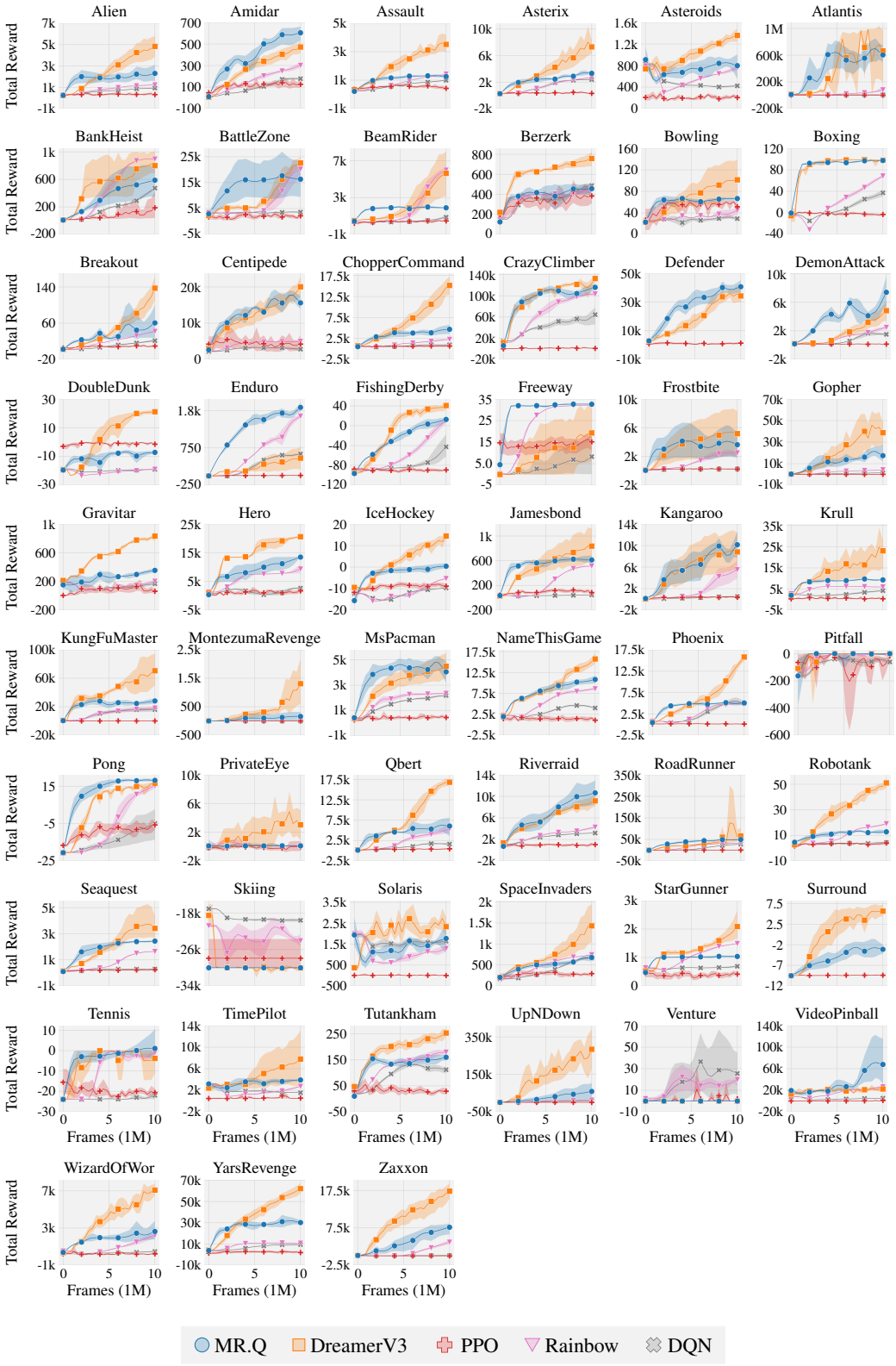

Figure 6: **Atari learning curves.** Time steps consider the number of environment interactions, where 2.5M time steps equals 10M frames in the original environment. Results are over 10 seeds. The shaded area captures a 95% boostrap confidence interval.

D COMPLETE ABLATION RESULTS

In this section, we show a per-environment breakdown of each variation in the design study in Section 5.2. Each table reports the raw score for each environment. The [bracketed values] represent a 95% bootstrap confidence interval. The aggregate mean, median and interquartile mean (IQM) are computed over the the difference in the normalized score. We use TD3 to normalize for Gym, raw scores for DMC and human scores to normalize for Atari (see Appendix B.3). Highlighting is used to designate the scale of the difference in normalized score:

- (≤ -0.5)
- $[-0.2, -0.5)$
- $[-0.01, -0.2)$
- $[0.01, 0.2)$
- $[0.2, 0.5)$
- (≥ 0.5)

D.1 GYM

Task	MR.Q	Linear value function	Dynamics target	No target encoder	Revert
Ant	6989 [6215, 7621]	1750 [1591, 1945]	6976 [6698, 7218]	4608 [3631, 5574]	-330 [-1636, 805]
HalfCheetah	13305 [11857, 14135]	3379 [2948, 3721]	14134 [13742, 14578]	12619 [12183, 12927]	-806 [-995, -640]
Hopper	2684 [2126, 3255]	710 [509, 925]	2505 [1792, 3361]	514 [263, 781]	91 [26, 158]
Humanoid	7259 [5060, 9411]	457 [422, 491]	3758 [2200, 6081]	393 [321, 472]	128 [86, 180]
Walker2d	6629 [5849, 7529]	1141 [658, 1605]	6269 [6026, 6513]	4483 [3247, 5719]	65 [-12, 171]
Mean	-	-1.12 [-1.15, -1.08]	-0.15 [-0.24, -0.01]	-0.65 [-0.72, -0.58]	-1.40 [-1.46, -1.35]
Median	-	-1.31 [-1.34, -1.26]	-0.06 [-0.13, 0.05]	-0.59 [-0.74, -0.44]	-1.41 [-1.42, -1.40]
IQM	-	-1.19 [-1.19, -1.16]	-0.05 [-0.14, 0.01]	-0.60 [-0.73, -0.47]	-1.46 [-1.47, -1.42]

Task	MR.Q	Non-linear model	MSE reward loss	No reward scaling
Ant	6989 [6215, 7621]	7077 [6309, 7686]	7308 [6676, 7757]	7258 [6837, 7684]
HalfCheetah	13305 [11857, 14135]	13225 [12703, 13730]	14442 [14156, 14704]	13426 [13187, 13591]
Hopper	2684 [2126, 3255]	2826 [2047, 3605]	2872 [2240, 3525]	2336 [1986, 2824]
Humanoid	7259 [5060, 9411]	10361 [10138, 10594]	10704 [10207, 11068]	6523 [3829, 8669]
Walker2d	6629 [5849, 7529]	5203 [4424, 5919]	6779 [5790, 7769]	5676 [5285, 6211]
Mean	-	0.06 [-0.01, 0.14]	0.19 [0.07, 0.30]	-0.08 [-0.23, 0.03]
Median	-	0.02 [-0.06, 0.13]	0.08 [-0.08, 0.21]	-0.11 [-0.20, 0.03]
IQM	-	0.02 [-0.04, 0.12]	0.08 [-0.02, 0.22]	-0.08 [-0.17, 0.04]

Task	MR.Q	MSE terminal loss	No min	No LAP
Ant	6989 [6215, 7621]	6369 [5816, 6955]	6253 [5140, 7273]	6670 [6396, 6945]
HalfCheetah	13305 [11857, 14135]	12620 [10365, 13853]	14134 [13808, 14538]	13150 [13060, 13233]
Hopper	2684 [2126, 3255]	2509 [2083, 2972]	1623 [1287, 1931]	2252 [1441, 3075]
Humanoid	7259 [5060, 9411]	5640 [3128, 7969]	4989 [3302, 7037]	7544 [5616, 8805]
Walker2d	6629 [5849, 7529]	5562 [5007, 6091]	2416 [1354, 3524]	5225 [4290, 6020]
Mean	-	-0.17 [-0.33, -0.03]	-0.39 [-0.47, -0.31]	-0.11 [-0.14, -0.08]
Median	-	-0.15 [-0.28, 0.02]	-0.33 [-0.41, -0.21]	-0.08 [-0.15, -0.01]
IQM	-	-0.16 [-0.27, -0.01]	-0.32 [-0.39, -0.24]	-0.08 [-0.15, 0.01]

Task	MR.Q	No MR	1-step return	No unroll
Ant	6989 [6215, 7621]	3451 [1772, 5132]	7791 [7766, 7827]	7792 [7661, 7957]
HalfCheetah	13305 [11857, 14135]	11174 [9161, 12527]	12959 [10186, 14766]	14253 [13754, 14754]
Hopper	2684 [2126, 3255]	2445 [1707, 3184]	3091 [2457, 3738]	2514 [1969, 3293]
Humanoid	7259 [5060, 9411]	3024 [1523, 4406]	1088 [769, 1517]	6714 [3533, 9973]
Walker2d	6629 [5849, 7529]	4993 [4684, 5322]	4794 [3395, 6315]	5506 [4507, 6302]
Mean	-	-0.48 [-0.63, -0.34]	-0.28 [-0.39, -0.17]	-0.03 [-0.21, 0.13]
Median	-	-0.41 [-0.49, -0.33]	-0.03 [-0.29, 0.12]	-0.05 [-0.21, 0.14]
IQM	-	-0.48 [-0.59, -0.34]	-0.12 [-0.30, 0.06]	-0.02 [-0.19, 0.12]

D.2 DMC - PROPRIOCEPTIVE

Task	MR.Q	Linear value function	Dynamics target	No target encoder	Revert
acrobot-swingup	567 [517, 621]	30 [15, 46]	626 [578, 684]	16 [9, 25]	11 [8, 18]
ball_in_cup-catch	981 [979, 983]	820 [658, 922]	980 [978, 983]	569 [436, 719]	301 [233, 365]
cartpole-balance	999 [999, 1000]	449 [380, 520]	999 [999, 1000]	992 [986, 997]	272 [206, 332]
cartpole-balance_sparse	1000 [1000, 1000]	183 [142, 224]	1000 [1000, 1000]	1000 [1000, 1000]	197 [177, 214]
cartpole-swingup	866 [866, 866]	267 [225, 310]	869 [866, 876]	852 [840, 861]	191 [99, 263]
cartpole-swingup_sparse	798 [779, 818]	12 [5, 22]	817 [809, 832]	0 [0, 0]	0 [0, 0]
cheetah-run	914 [911, 917]	394 [376, 411]	919 [919, 921]	904 [899, 910]	74 [31, 123]
dog-run	569 [546, 595]	11 [5, 18]	254 [202, 305]	11 [8, 16]	3 [4, 4]
dog-stand	967 [959, 975]	22 [17, 29]	672 [520, 830]	36 [27, 51]	22 [18, 29]
dog-trot	877 [845, 898]	15 [11, 20]	319 [279, 365]	12 [8, 19]	4 [3, 5]
dog-walk	916 [908, 924]	13 [11, 18]	312 [247, 396]	10 [7, 15]	4 [4, 6]
finger-spin	937 [917, 958]	736 [670, 825]	942 [916, 971]	869 [698, 963]	0 [0, 1]
finger-turn_easy	953 [928, 975]	238 [157, 319]	947 [900, 979]	624 [509, 756]	159 [60, 280]
finger-turn_hard	950 [908, 974]	23 [1, 63]	923 [878, 966]	431 [301, 563]	60 [20, 100]
fish-swim	792 [772, 811]	83 [65, 102]	410 [307, 493]	97 [65, 129]	71 [53, 90]
hopper-hop	251 [201, 295]	10 [4, 17]	199 [131, 265]	0 [0, 1]	0 [0, 1]
hopper-stand	951 [948, 955]	66 [30, 102]	639 [413, 866]	4 [3, 7]	5 [3, 8]
humanoid-run	200 [169, 236]	1 [1, 1]	1 [1, 1]	1 [1, 1]	1 [1, 1]
humanoid-stand	868 [823, 907]	6 [5, 7]	7 [6, 8]	7 [6, 9]	7 [7, 8]
humanoid-walk	662 [609, 721]	1 [1, 2]	2 [2, 3]	2 [2, 3]	1 [2, 2]
pendulum-swingup	748 [594, 830]	357 [114, 617]	826 [812, 840]	784 [706, 834]	61 [20, 103]
quadruped-run	947 [940, 954]	172 [93, 252]	942 [930, 951]	829 [757, 895]	87 [40, 145]
quadruped-walk	963 [959, 968]	91 [52, 141]	939 [935, 943]	952 [946, 960]	81 [36, 130]
reacher-easy	983 [983, 985]	802 [722, 877]	984 [983, 986]	983 [981, 986]	789 [727, 856]
reacher-hard	977 [975, 979]	853 [778, 914]	970 [965, 976]	975 [972, 979]	526 [361, 695]
walker-run	793 [766, 816]	238 [207, 274]	730 [585, 814]	776 [757, 792]	25 [22, 31]
walker-stand	988 [987, 990]	859 [780, 921]	988 [985, 991]	988 [986, 991]	221 [179, 264]
walker-walk	978 [978, 980]	504 [397, 613]	975 [975, 977]	974 [973, 976]	33 [23, 46]
Mean	-	-0.58 [-0.59, -0.56]	-0.15 [-0.15, -0.15]	-0.35 [-0.35, -0.34]	-0.72 [-0.73, -0.72]
Median	-	-0.58 [-0.64, -0.57]	-0.01 [-0.02, -0.00]	-0.22 [-0.23, -0.21]	-0.78 [-0.80, -0.76]
IQM	-	-0.62 [-0.64, -0.60]	-0.05 [-0.06, -0.03]	-0.27 [-0.29, -0.25]	-0.78 [-0.78, -0.76]

Task	MR.Q	Non-linear model	MSE reward loss	No reward scaling
acrobot-swingup	567 [517, 621]	553 [478, 629]	577 [547, 612]	593 [532, 672]
ball_in_cup-catch	981 [979, 983]	982 [982, 984]	983 [982, 985]	983 [982, 984]
cartpole-balance	999 [999, 1000]	999 [999, 1000]	999 [998, 1000]	998 [999, 999]
cartpole-balance_sparse	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]
cartpole-swingup	866 [866, 866]	866 [866, 867]	865 [865, 866]	865 [864, 866]
cartpole-swingup_sparse	798 [779, 818]	824 [809, 839]	812 [786, 834]	647 [318, 822]
cheetah-run	914 [911, 917]	909 [906, 913]	910 [907, 915]	911 [909, 914]
dog-run	569 [546, 595]	588 [516, 646]	527 [513, 545]	586 [546, 613]
dog-stand	967 [959, 975]	962 [938, 982]	964 [958, 971]	959 [940, 979]
dog-trot	877 [845, 898]	868 [816, 914]	861 [831, 886]	817 [713, 903]
dog-walk	916 [908, 924]	920 [915, 925]	724 [473, 882]	901 [890, 917]
finger-spin	937 [917, 958]	868 [767, 951]	907 [875, 947]	873 [768, 947]
finger-turn_easy	953 [928, 975]	972 [968, 978]	935 [894, 977]	977 [973, 982]
finger-turn_hard	950 [908, 974]	931 [887, 975]	947 [910, 969]	946 [905, 969]
fish-swim	792 [772, 811]	790 [754, 824]	793 [766, 821]	745 [663, 809]
hopper-hop	251 [201, 295]	288 [222, 332]	174 [119, 230]	343 [263, 477]
hopper-stand	951 [948, 955]	848 [664, 945]	854 [707, 936]	934 [912, 948]
humanoid-run	200 [169, 236]	205 [191, 221]	91 [45, 121]	184 [149, 214]
humanoid-stand	868 [823, 907]	811 [712, 878]	214 [14, 566]	810 [655, 899]
humanoid-walk	662 [609, 721]	668 [590, 734]	77 [3, 224]	665 [589, 765]
pendulum-swingup	748 [594, 830]	819 [802, 836]	827 [811, 843]	816 [790, 838]
quadruped-run	947 [940, 954]	944 [934, 954]	949 [941, 956]	951 [944, 958]
quadruped-walk	963 [959, 968]	963 [961, 967]	966 [961, 971]	966 [961, 971]
reacher-easy	983 [983, 985]	983 [982, 984]	964 [927, 984]	964 [926, 984]
reacher-hard	977 [975, 979]	953 [914, 976]	976 [974, 978]	971 [968, 975]
walker-run	793 [766, 816]	795 [784, 811]	778 [709, 821]	804 [783, 820]
walker-stand	988 [987, 990]	983 [971, 991]	988 [987, 990]	989 [988, 990]
walker-walk	978 [978, 980]	974 [972, 978]	967 [948, 979]	979 [978, 980]
Mean	-	-0.00 [-0.02, 0.01]	-0.06 [-0.08, -0.05]	-0.01 [-0.02, 0.00]
Median	-	-0.00 [-0.00, 0.00]	-0.00 [-0.01, 0.00]	-0.00 [-0.00, 0.00]
IQM	-	-0.00 [-0.01, 0.00]	-0.01 [-0.02, -0.00]	-0.00 [-0.00, 0.00]

Task	MR.Q	MSE terminal loss	No min	No LAP
acrobot-swingup	567 [517, 621]	567 [517, 621]	623 [572, 673]	566 [520, 612]
ball_in_cup-catch	981 [979, 983]	981 [979, 983]	982 [980, 984]	981 [980, 984]
cartpole-balance	999 [999, 1000]	999 [999, 1000]	999 [1000, 1000]	999 [998, 1000]
cartpole-balance_sparse	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]	992 [982, 1000]
cartpole-swingup	866 [866, 866]	866 [866, 866]	868 [866, 874]	865 [865, 866]
cartpole-swingup_sparse	798 [779, 818]	798 [779, 818]	799 [780, 812]	796 [779, 810]
cheetah-run	914 [911, 917]	914 [911, 917]	910 [893, 921]	908 [905, 913]
dog-run	569 [546, 595]	569 [546, 595]	577 [540, 610]	536 [499, 573]
dog-stand	967 [959, 975]	967 [959, 975]	946 [917, 969]	971 [966, 976]
dog-trot	877 [845, 898]	877 [845, 898]	846 [767, 906]	842 [764, 897]
dog-walk	916 [908, 924]	916 [908, 924]	747 [447, 908]	899 [886, 914]
finger-spin	937 [917, 958]	937 [917, 958]	926 [907, 950]	915 [892, 948]
finger-turn_easy	953 [928, 975]	953 [928, 975]	976 [972, 980]	975 [967, 983]
finger-turn_hard	950 [908, 974]	950 [908, 974]	894 [833, 953]	949 [909, 972]
fish-swim	792 [772, 811]	792 [772, 811]	785 [763, 810]	788 [754, 826]
hopper-hop	251 [201, 295]	251 [201, 295]	336 [322, 352]	347 [265, 431]
hopper-stand	951 [948, 955]	951 [948, 955]	935 [926, 947]	941 [935, 948]
humanoid-run	200 [169, 236]	200 [169, 236]	198 [175, 225]	202 [191, 212]
humanoid-stand	868 [823, 907]	868 [823, 907]	833 [793, 871]	880 [856, 900]
humanoid-walk	662 [609, 721]	662 [609, 721]	597 [292, 808]	697 [561, 828]
pendulum-swingup	748 [594, 830]	748 [594, 830]	825 [811, 839]	815 [792, 836]
quadruped-run	947 [940, 954]	947 [940, 954]	946 [941, 951]	937 [927, 947]
quadruped-walk	963 [959, 968]	963 [959, 968]	959 [942, 972]	955 [942, 967]
reacher-easy	983 [983, 985]	983 [983, 985]	983 [981, 986]	983 [982, 986]
reacher-hard	977 [975, 979]	977 [975, 979]	978 [974, 982]	974 [969, 981]
walker-run	793 [766, 816]	793 [766, 816]	806 [779, 821]	812 [803, 822]
walker-stand	988 [987, 990]	988 [987, 990]	989 [986, 992]	986 [985, 987]
walker-walk	978 [978, 980]	978 [978, 980]	978 [976, 980]	977 [974, 980]
Mean	-	-0.00 [-0.01, 0.01]	-0.01 [-0.02, 0.01]	0.00 [-0.00, 0.01]
Median	-	0.00 [-0.00, 0.00]	-0.00 [-0.00, 0.00]	-0.00 [-0.00, 0.00]
IQM	-	-0.00 [-0.00, 0.00]	-0.00 [-0.00, 0.00]	-0.00 [-0.00, 0.00]

Task	MR.Q	No MR	1-step return	No unroll
acrobot-swingup	567 [517, 621]	576 [483, 665]	440 [360, 528]	515 [455, 598]
ball_in_cup-catch	981 [979, 983]	981 [980, 984]	984 [983, 985]	982 [981, 984]
cartpole-balance	999 [999, 1000]	994 [991, 999]	999 [1000, 1000]	999 [999, 1000]
cartpole-balance_sparse	1000 [1000, 1000]	1000 [1000, 1000]	961 [886, 1000]	1000 [1000, 1000]
cartpole-swingup	866 [866, 866]	870 [864, 878]	881 [879, 882]	864 [861, 867]
cartpole-swingup_sparse	798 [779, 818]	684 [528, 814]	845 [845, 847]	818 [811, 831]
cheetah-run	914 [911, 917]	871 [823, 907]	922 [921, 924]	909 [908, 911]
dog-run	569 [546, 595]	68 [63, 75]	299 [196, 360]	514 [473, 554]
dog-stand	967 [959, 975]	494 [452, 530]	606 [344, 865]	955 [944, 971]
dog-trot	877 [845, 898]	65 [49, 80]	725 [679, 756]	857 [833, 883]
dog-walk	916 [908, 924]	102 [81, 122]	788 [739, 832]	920 [905, 934]
finger-spin	937 [917, 958]	888 [731, 975]	983 [977, 988]	880 [781, 940]
finger-turn_easy	953 [928, 975]	947 [913, 974]	980 [979, 982]	950 [917, 976]
finger-turn_hard	950 [908, 974]	846 [756, 926]	968 [958, 976]	947 [907, 971]
fish-swim	792 [772, 811]	706 [683, 727]	498 [323, 651]	709 [618, 783]
hopper-hop	251 [201, 295]	85 [33, 142]	364 [336, 394]	297 [169, 442]
hopper-stand	951 [948, 955]	365 [233, 491]	952 [947, 959]	949 [944, 955]
humanoid-run	200 [169, 236]	1 [1, 2]	190 [124, 241]	192 [172, 214]
humanoid-stand	868 [823, 907]	201 [9, 517]	753 [665, 838]	858 [806, 913]
humanoid-walk	662 [609, 721]	84 [3, 247]	761 [689, 827]	675 [593, 772]
pendulum-swingup	748 [594, 830]	827 [812, 842]	823 [807, 841]	819 [793, 843]
quadruped-run	947 [940, 954]	871 [793, 933]	945 [940, 950]	950 [944, 955]
quadruped-walk	963 [959, 968]	951 [943, 962]	962 [958, 968]	962 [955, 969]
reacher-easy	983 [983, 985]	980 [979, 983]	984 [984, 986]	981 [976, 986]
reacher-hard	977 [975, 979]	949 [909, 974]	980 [979, 983]	954 [913, 978]
walker-run	793 [766, 816]	780 [769, 790]	835 [827, 843]	780 [702, 825]
walker-stand	988 [987, 990]	983 [980, 988]	990 [990, 992]	988 [988, 990]
walker-walk	978 [978, 980]	976 [975, 977]	979 [977, 982]	975 [969, 981]
Mean	-	-0.19 [-0.19, -0.18]	-0.04 [-0.05, -0.02]	-0.01 [-0.01, -0.00]
Median	-	-0.05 [-0.08, -0.01]	0.00 [0.00, 0.00]	-0.00 [-0.00, 0.00]
IQM	-	-0.06 [-0.08, -0.05]	0.00 [-0.00, 0.01]	-0.00 [-0.01, -0.00]

D.3 DMC - VISUAL

Task	MR.Q	Linear value function	Dynamics target	No target encoder	Revert
acrobot-swingup	287 [253, 317]	15 [5, 22]	296 [281, 323]	16 [0, 38]	19 [13, 23]
ball_in_cup-catch	977 [975, 980]	644 [328, 904]	972 [965, 976]	605 [496, 726]	195 [91, 297]
cartpole-balance	999 [999, 999]	306 [254, 349]	998 [998, 999]	978 [947, 997]	190 [163, 212]
cartpole-balance_sparse	1000 [1000, 1000]	243 [183, 327]	1000 [1000, 1000]	1000 [1000, 1000]	346 [193, 639]
cartpole-swingup	868 [861, 875]	229 [181, 294]	861 [859, 865]	689 [487, 808]	115 [82, 175]
cartpole-swingup_sparse	797 [777, 818]	4 [0, 14]	267 [0, 801]	0 [0, 0]	0 [0, 0]
cheetah-run	775 [752, 805]	230 [159, 294]	831 [761, 875]	745 [723, 784]	69 [36, 129]
dog-run	60 [44, 80]	19 [17, 22]	36 [34, 39]	10 [6, 20]	3 [3, 4]
dog-stand	216 [201, 233]	76 [70, 82]	191 [155, 247]	60 [44, 89]	17 [16, 19]
dog-trot	65 [54, 79]	19 [16, 24]	46 [42, 53]	9 [9, 10]	5 [4, 6]
dog-walk	77 [70, 83]	30 [26, 33]	62 [57, 72]	16 [10, 21]	6 [6, 8]
finger-spin	965 [938, 982]	789 [598, 923]	786 [672, 931]	929 [893, 981]	1 [0, 2]
finger-turn_easy	953 [925, 974]	132 [98, 200]	876 [691, 969]	898 [855, 969]	133 [99, 200]
finger-turn_hard	932 [905, 957]	66 [0, 100]	859 [777, 963]	492 [385, 577]	66 [0, 100]
fish-swim	79 [67, 93]	69 [45, 109]	71 [38, 106]	65 [49, 84]	53 [47, 60]
hopper-hop	270 [229, 317]	1 [0, 2]	184 [165, 204]	2 [0, 6]	0 [0, 2]
hopper-stand	852 [705, 932]	7 [3, 11]	911 [900, 922]	5 [3, 9]	4 [3, 8]
humanoid-run	1 [1, 2]	1 [1, 1]	1 [1, 2]	1 [1, 1]	1 [1, 1]
humanoid-stand	7 [7, 8]	5 [4, 8]	6 [5, 8]	7 [6, 8]	6 [5, 7]
humanoid-walk	2 [2, 3]	1 [1, 2]	2 [1, 3]	1 [1, 2]	1 [1, 2]
pendulum-swingup	829 [815, 842]	97 [0, 192]	749 [632, 840]	191 [93, 287]	66 [0, 100]
quadruped-run	498 [474, 523]	131 [77, 187]	488 [468, 517]	575 [566, 594]	86 [69, 120]
quadruped-walk	833 [796, 868]	105 [57, 155]	717 [445, 895]	817 [790, 868]	76 [34, 100]
reacher-easy	979 [978, 982]	605 [398, 868]	977 [973, 981]	979 [970, 986]	583 [395, 684]
reacher-hard	965 [945, 977]	288 [195, 473]	975 [971, 978]	970 [963, 975]	33 [0, 100]
walker-run	615 [571, 655]	158 [133, 193]	531 [463, 577]	611 [590, 631]	35 [29, 39]
walker-stand	980 [977, 984]	707 [601, 881]	982 [981, 984]	984 [979, 988]	280 [269, 294]
walker-walk	970 [968, 973]	350 [228, 529]	904 [850, 951]	965 [957, 972]	22 [19, 25]
Mean	-	-0.41 [-0.42, -0.39]	-0.05 [-0.05, -0.04]	-0.15 [-0.15, -0.15]	-0.52 [-0.52, -0.51]
Median	-	-0.37 [-0.44, -0.37]	-0.01 [-0.01, -0.00]	-0.03 [-0.05, -0.03]	-0.68 [-0.72, -0.62]
IQM	-	-0.42 [-0.43, -0.38]	-0.02 [-0.02, -0.01]	-0.05 [-0.06, -0.04]	-0.57 [-0.58, -0.56]

Task	MR.Q	Non-linear model	MSE reward loss	No reward scaling
acrobot-swingup	287 [253, 317]	279 [235, 314]	265 [242, 294]	323 [275, 368]
ball_in_cup-catch	977 [975, 980]	973 [970, 977]	974 [969, 978]	973 [971, 977]
cartpole-balance	999 [999, 999]	999 [999, 999]	998 [998, 1000]	999 [999, 999]
cartpole-balance_sparse	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]
cartpole-swingup	868 [861, 875]	849 [819, 873]	876 [875, 878]	860 [829, 877]
cartpole-swingup_sparse	797 [777, 818]	768 [684, 824]	33 [0, 60]	813 [805, 823]
cheetah-run	775 [752, 805]	763 [757, 770]	732 [712, 757]	720 [678, 758]
dog-run	60 [44, 80]	37 [36, 40]	36 [33, 38]	61 [49, 73]
dog-stand	216 [201, 233]	200 [193, 207]	195 [187, 208]	317 [239, 387]
dog-trot	65 [54, 79]	51 [47, 56]	47 [44, 50]	65 [55, 79]
dog-walk	77 [70, 83]	69 [62, 78]	63 [58, 71]	89 [83, 96]
finger-spin	965 [938, 982]	907 [841, 971]	924 [884, 980]	903 [776, 975]
finger-turn_easy	953 [925, 974]	932 [889, 977]	844 [786, 881]	873 [775, 954]
finger-turn_hard	932 [905, 957]	938 [903, 967]	900 [867, 964]	923 [885, 962]
fish-swim	79 [67, 93]	67 [55, 79]	75 [59, 105]	73 [59, 87]
hopper-hop	270 [229, 317]	308 [257, 368]	102 [14, 151]	244 [204, 298]
hopper-stand	852 [705, 932]	935 [929, 942]	919 [914, 925]	911 [888, 926]
humanoid-run	1 [1, 2]	1 [1, 1]	1 [1, 1]	1 [1, 1]
humanoid-stand	7 [7, 8]	6 [6, 8]	7 [7, 8]	7 [6, 8]
humanoid-walk	2 [2, 3]	2 [2, 3]	1 [1, 2]	2 [2, 4]
pendulum-swingup	829 [815, 842]	820 [795, 844]	581 [94, 842]	823 [798, 846]
quadruped-run	498 [474, 523]	555 [514, 578]	516 [478, 544]	505 [471, 545]
quadruped-walk	833 [796, 868]	762 [727, 788]	835 [751, 880]	823 [781, 867]
reacher-easy	979 [978, 982]	939 [900, 979]	979 [976, 984]	962 [924, 983]
reacher-hard	965 [945, 977]	868 [753, 956]	941 [879, 976]	972 [970, 975]
walker-run	615 [571, 655]	612 [593, 633]	596 [505, 643]	600 [544, 632]
walker-stand	980 [977, 984]	982 [977, 987]	983 [984, 984]	986 [984, 989]
walker-walk	970 [968, 973]	951 [917, 972]	969 [961, 976]	970 [968, 974]
Mean	-	-0.01 [-0.02, -0.00]	-0.05 [-0.07, -0.04]	-0.00 [-0.01, 0.01]
Median	-	-0.01 [-0.01, -0.00]	-0.01 [-0.01, 0.00]	-0.00 [-0.00, 0.00]
IQM	-	-0.01 [-0.01, -0.00]	-0.01 [-0.01, -0.00]	-0.00 [-0.00, 0.00]

Task	MR.Q	MSE terminal loss	No min	No LAP
acrobot-swingup	287 [253, 317]	287 [253, 317]	332 [301, 391]	280 [235, 354]
ball_in_cup-catch	977 [975, 980]	977 [975, 980]	975 [974, 976]	971 [966, 978]
cartpole-balance	999 [999, 999]	999 [999, 999]	998 [998, 999]	998 [997, 999]
cartpole-balance_sparse	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]
cartpole-swingup	868 [861, 875]	868 [861, 875]	879 [879, 880]	798 [785, 821]
cartpole-swingup_sparse	797 [777, 818]	797 [777, 818]	805 [763, 829]	764 [736, 799]
cheetah-run	775 [752, 805]	775 [752, 805]	751 [734, 762]	706 [670, 741]
dog-run	60 [44, 80]	60 [44, 80]	42 [37, 51]	62 [45, 91]
dog-stand	216 [201, 233]	216 [201, 233]	228 [224, 232]	279 [229, 315]
dog-trot	65 [54, 79]	65 [54, 79]	50 [48, 53]	58 [56, 61]
dog-walk	77 [70, 83]	77 [70, 83]	86 [69, 106]	91 [85, 101]
finger-spin	965 [938, 982]	965 [938, 982]	870 [709, 982]	940 [864, 979]
finger-turn_easy	953 [925, 974]	953 [925, 974]	963 [952, 975]	844 [785, 879]
finger-turn_hard	932 [905, 957]	932 [905, 957]	933 [874, 976]	932 [858, 974]
fish-swim	79 [67, 93]	79 [67, 93]	54 [49, 61]	63 [49, 89]
hopper-hop	270 [229, 317]	270 [229, 317]	255 [244, 275]	186 [152, 204]
hopper-stand	852 [705, 932]	852 [705, 932]	923 [902, 945]	884 [877, 896]
humanoid-run	1 [1, 2]	1 [1, 2]	1 [1, 1]	1 [1, 2]
humanoid-stand	7 [7, 8]	7 [7, 8]	6 [5, 8]	10 [6, 18]
humanoid-walk	2 [2, 3]	2 [2, 3]	2 [2, 4]	2 [2, 3]
pendulum-swingup	829 [815, 842]	829 [815, 842]	831 [809, 843]	829 [808, 841]
quadruped-run	498 [474, 523]	498 [474, 523]	539 [500, 578]	463 [428, 485]
quadruped-walk	833 [796, 868]	833 [796, 868]	849 [745, 909]	799 [713, 905]
reacher-easy	979 [978, 982]	979 [978, 982]	953 [897, 981]	948 [885, 980]
reacher-hard	965 [945, 977]	965 [945, 977]	936 [872, 975]	973 [973, 974]
walker-run	615 [571, 655]	615 [571, 655]	666 [643, 682]	662 [629, 725]
walker-stand	980 [977, 984]	980 [977, 984]	982 [975, 988]	984 [984, 985]
walker-walk	970 [968, 973]	970 [968, 973]	970 [968, 972]	972 [964, 979]
Mean	-	-0.00 [-0.00, 0.00]	0.00 [-0.01, 0.01]	-0.01 [-0.02, -0.01]
Median	-	-0.00 [-0.00, 0.00]	0.00 [-0.00, 0.00]	-0.00 [-0.01, 0.00]
IQM	-	-0.00 [-0.00, 0.00]	0.00 [-0.00, 0.00]	-0.01 [-0.02, 0.00]

Task	MR.Q	No MR	1-step return	No unroll
acrobot-swingup	287 [253, 317]	362 [305, 421]	91 [76, 112]	126 [77, 159]
ball_in_cup-catch	977 [975, 980]	898 [746, 977]	980 [979, 982]	976 [973, 980]
cartpole-balance	999 [999, 999]	998 [998, 999]	998 [998, 1000]	998 [999, 999]
cartpole-balance_sparse	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]	1000 [1000, 1000]
cartpole-swingup	868 [861, 875]	871 [864, 878]	858 [835, 875]	872 [863, 879]
cartpole-swingup_sparse	797 [777, 818]	459 [139, 780]	712 [685, 759]	529 [0, 816]
cheetah-run	775 [752, 805]	782 [765, 805]	675 [674, 677]	753 [679, 845]
dog-run	60 [44, 80]	22 [21, 24]	30 [21, 43]	45 [36, 56]
dog-stand	216 [201, 233]	137 [127, 148]	160 [143, 191]	209 [195, 217]
dog-trot	65 [54, 79]	32 [29, 36]	29 [25, 32]	47 [46, 49]
dog-walk	77 [70, 83]	42 [36, 50]	55 [33, 67]	67 [63, 76]
finger-spin	965 [938, 982]	887 [757, 965]	984 [978, 989]	738 [588, 960]
finger-turn_easy	953 [925, 974]	694 [539, 805]	942 [874, 979]	869 [766, 962]
finger-turn_hard	932 [905, 957]	622 [436, 825]	908 [872, 974]	902 [780, 973]
fish-swim	79 [67, 93]	72 [60, 93]	64 [58, 72]	67 [55, 90]
hopper-hop	270 [229, 317]	192 [166, 216]	248 [231, 280]	242 [219, 270]
hopper-stand	852 [705, 932]	918 [897, 935]	877 [820, 915]	925 [907, 940]
humanoid-run	1 [1, 2]	1 [1, 2]	1 [1, 2]	1 [1, 1]
humanoid-stand	7 [7, 8]	7 [7, 8]	7 [6, 9]	7 [5, 9]
humanoid-walk	2 [2, 3]	2 [2, 3]	2 [2, 3]	2 [2, 3]
pendulum-swingup	829 [815, 842]	819 [787, 844]	828 [811, 839]	665 [382, 811]
quadruped-run	498 [474, 523]	478 [432, 515]	456 [424, 476]	398 [326, 465]
quadruped-walk	833 [796, 868]	701 [663, 731]	666 [627, 720]	730 [663, 769]
reacher-easy	979 [978, 982]	978 [976, 981]	972 [948, 985]	978 [977, 980]
reacher-hard	965 [945, 977]	545 [214, 858]	978 [972, 982]	893 [776, 967]
walker-run	615 [571, 655]	568 [538, 599]	672 [639, 730]	656 [619, 696]
walker-stand	980 [977, 984]	974 [962, 986]	986 [982, 991]	983 [981, 987]
walker-walk	970 [968, 973]	955 [948, 964]	971 [967, 978]	971 [971, 972]
Mean	-	-0.07 [-0.09, -0.03]	-0.03 [-0.03, -0.02]	-0.04 [-0.06, -0.01]
Median	-	-0.02 [-0.03, -0.01]	-0.01 [-0.01, -0.00]	-0.01 [-0.01, -0.00]
IQM	-	-0.03 [-0.03, -0.01]	-0.01 [-0.02, -0.01]	-0.02 [-0.02, -0.01]

D.4 ATARI

Task	MR.Q	Linear value function	Dynamics target	No target encoder
Alien	1884 [1466, 2317]	596 [561, 631]	1176 [1138, 1215]	2040 [1585, 2495]
Amidar	318 [233, 391]	48 [38, 61]	214 [182, 247]	249 [232, 268]
Assault	1162 [1063, 1274]	366 [359, 374]	911 [906, 917]	1057 [880, 1234]
Asterix	2413 [1925, 2760]	810 [585, 1035]	1940 [1865, 2015]	2407 [1860, 2955]
Asteroids	674 [561, 778]	609 [595, 623]	776 [716, 837]	765 [591, 939]
Atlantis	609450 [372300, 778910]	17683 [13080, 20330]	529745 [145750, 913740]	76930 [76090, 77770]
BankHeist	295 [203, 425]	93 [76, 111]	646 [326, 966]	40 [38, 43]
BattleZone	16120 [9540, 24220]	11300 [11100, 11500]	8250 [2100, 14400]	4600 [2500, 6700]
BeamRider	1903 [1782, 2022]	584 [528, 642]	1201 [1015, 1387]	1468 [1298, 1639]
Berzerk	418 [399, 441]	315 [216, 415]	381 [359, 403]	359 [275, 444]
Bowling	66 [63, 70]	31 [31, 33]	81 [81, 82]	40 [33, 48]
Boxing	93 [91, 96]	39 [37, 42]	90 [88, 93]	92 [89, 96]
Breakout	37 [26, 50]	3 [2, 4]	9 [8, 11]	1 [1, 2]
Centipede	12542 [10782, 14550]	6709 [6207, 7213]	7853 [7680, 8026]	5167 [3661, 6674]
ChopperCommand	3880 [2850, 4814]	890 [780, 1000]	3055 [2870, 3240]	1385 [1060, 1710]
CrazyClimber	104544 [100800, 109334]	23016 [21610, 25010]	92455 [84000, 100910]	40240 [29150, 51330]
Defender	26584 [21155, 32013]	7825 [5640, 10010]	17592 [9290, 25895]	11627 [5745, 17510]
DemonAttack	4336 [3284, 5390]	1608 [1365, 1852]	281 [242, 322]	278 [204, 352]
DoubleDunk	-14 [-20, -8]	-18 [-24, -13]	-15 [-22, -10]	-20 [-21, -19]
Enduro	1367 [1258, 1476]	343 [319, 368]	622 [621, 623]	690 [667, 713]
FishingDerby	-31 [-37, -26]	-90 [-96, -86]	-64 [-66, -62]	-81 [-87, -75]
Freeway	31 [32, 32]	20 [19, 23]	32 [33, 33]	11 [0, 22]
Frostbite	4156 [1809, 6712]	198 [198, 198]	268 [267, 269]	824 [258, 1390]
Gopher	11752 [7257, 17674]	598 [524, 672]	853 [832, 874]	4371 [3794, 4948]
Gravitar	295 [226, 356]	190 [130, 250]	352 [250, 455]	60 [40, 80]
Hero	8057 [5442, 11106]	615 [112, 1118]	7560 [7560, 7560]	2200 [1377, 3024]
IceHockey	-1 [-3, -1]	-15 [-16, -15]	-6 [-10, -4]	-8 [-10, -6]
Jamesbond	564 [519, 611]	46 [40, 50]	412 [310, 515]	102 [80, 125]
Kangaroo	5388 [3012, 7180]	555 [520, 590]	6830 [600, 13060]	685 [590, 780]
Krull	9033 [8405, 9663]	6078 [5777, 6379]	7460 [6961, 7959]	9088 [8624, 9552]
KungFuMaster	27832 [21968, 32936]	10400 [9320, 11480]	17020 [7520, 26520]	12130 [11280, 12980]
MontezumaRevenge	100 [0, 260]	0 [0, 0]	0 [0, 0]	0 [0, 0]
MsPacman	4347 [3736, 4741]	826 [757, 896]	2950 [2490, 3410]	3171 [1873, 4469]
NameThisGame	8077 [7400, 8907]	2339 [2258, 2436]	6660 [6568, 6752]	7015 [6396, 7634]
Phoenix	4876 [4628, 5085]	570 [286, 854]	3996 [3843, 4150]	4260 [3944, 4577]
Pitfall	0 [0, 0]	-25 [-50, 0]	0 [0, 0]	-66 [-122, -12]
Pong	15 [12, 18]	-20 [-21, -20]	14 [14, 16]	-10 [-19, -2]
PrivateEye	100 [100, 100]	45 [0, 90]	100 [100, 100]	90 [80, 100]
Qbert	4600 [4390, 4808]	256 [228, 285]	493 [488, 500]	747 [615, 880]
Riverraid	5267 [3820, 6714]	1997 [1829, 2165]	6860 [6391, 7330]	6342 [5450, 7234]
RoadRunner	38842 [32362, 44588]	3245 [2410, 4080]	21835 [20960, 22710]	35120 [33050, 37190]
Robotank	11 [8, 13]	7 [3, 11]	7 [4, 10]	6 [3, 9]
Seaquest	1977 [1658, 2285]	305 [214, 400]	895 [834, 956]	1227 [378, 2076]
Skiing	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]
Solaris	1196 [814, 1578]	480 [0, 960]	710 [378, 1042]	1219 [1198, 1240]
SpaceInvaders	504 [461, 538]	242 [230, 255]	303 [263, 344]	296 [269, 323]
StarGunner	1020 [1000, 1040]	1060 [880, 1240]	960 [920, 1000]	970 [960, 980]
Surround	-6 [-8, -4]	-9 [-9, -9]	-7 [-8, -7]	-7 [-9, -6]
Tennis	-2 [-6, 0]	-24 [-24, -24]	-5 [-10, 0]	0 [0, 0]
TimePilot	3564 [2586, 4212]	2525 [1430, 3620]	2525 [2040, 3010]	1710 [1020, 2400]
Tutankham	133 [126, 145]	90 [86, 95]	164 [150, 179]	78 [2, 155]
UpNDown	15889 [7216, 29047]	3180 [2321, 4040]	3045 [2667, 3424]	4523 [4422, 4625]
Venture	0 [0, 0]	65 [0, 130]	0 [0, 0]	0 [0, 0]
VideoPinball	18537 [12410, 25939]	12994 [10570, 15419]	9170 [7880, 10460]	19734 [14479, 24989]
WizardOfWor	1970 [1786, 2154]	635 [480, 790]	1260 [670, 1850]	1440 [930, 1950]
YarsRevenge	28638 [25422, 31525]	6404 [6391, 6417]	23613 [16984, 30244]	21163 [21047, 21280]
Zaxxon	2684 [1234, 4144]	0 [0, 0]	690 [0, 1380]	0 [0, 0]
Mean	-	-1.56 [-1.62, -1.50]	-0.59 [-1.02, -0.16]	-1.07 [-1.10, -1.04]
Median	-	-0.43 [-0.44, -0.36]	-0.15 [-0.15, -0.11]	-0.20 [-0.28, -0.20]
IQM	-	-0.57 [-0.57, -0.54]	-0.21 [-0.22, -0.14]	-0.28 [-0.31, -0.28]

Task	MR.Q	Revert
Alien	1884 [1466, 2317]	66 [32, 100]
Amidar	318 [233, 391]	39 [19, 60]
Assault	1162 [1063, 1274]	366 [359, 374]
Asterix	2413 [1925, 2760]	492 [425, 560]
Asteroids	674 [561, 778]	249 [239, 259]
Atlantis	609450 [372300, 778910]	7310 [3140, 11480]
BankHeist	295 [203, 425]	14 [0, 28]
BattleZone	16120 [9540, 24220]	3550 [3300, 3800]
BeamRider	1903 [1782, 2022]	546 [510, 582]
Berzerk	418 [399, 441]	315 [275, 355]
Bowling	66 [63, 70]	41 [27, 55]
Boxing	93 [91, 96]	29 [21, 38]
Breakout	37 [26, 50]	2 [1, 3]
Centipede	12542 [10782, 14550]	2877 [2821, 2933]
ChopperCommand	3880 [2850, 4814]	530 [420, 640]
CrazyClimber	104544 [100800, 109334]	1700 [0, 3400]
Defender	26584 [21155, 32013]	1917 [1780, 2055]
DemonAttack	4336 [3284, 5390]	149 [147, 151]
DoubleDunk	-14 [-20, -8]	-23 [-24, -23]
Enduro	1367 [1258, 1476]	3 [0, 6]
FishingDerby	-31 [-37, -26]	-86 [-91, -82]
Freeway	31 [32, 32]	0 [0, 0]
Frostbite	4156 [1809, 6712]	170 [151, 190]
Gopher	11752 [7257, 17674]	385 [280, 490]
Gravitar	295 [226, 356]	82 [80, 85]
Hero	8057 [5442, 11106]	0 [0, 0]
IceHockey	-1 [-3, -1]	-14 [-17, -11]
Jamesbond	564 [519, 611]	60 [45, 75]
Kangaroo	5388 [3012, 7180]	80 [0, 160]
Krull	9033 [8405, 9663]	10 [0, 20]
KungFuMaster	27832 [21968, 32936]	1815 [130, 3500]
MontezumaRevenge	100 [0, 260]	0 [0, 0]
MsPacman	4347 [3736, 4741]	283 [182, 385]
NameThisGame	8077 [7400, 8907]	2675 [2609, 2742]
Phoenix	4876 [4628, 5085]	728 [517, 939]
Pitfall	0 [0, 0]	-1012 [-2000, -24]
Pong	15 [12, 18]	-20 [-21, -21]
PrivateEye	100 [100, 100]	50 [20, 80]
Qbert	4600 [4390, 4808]	137 [125, 150]
Riverraid	5267 [3820, 6714]	1660 [1131, 2190]
RoadRunner	38842 [32362, 44588]	0 [0, 0]
Robotank	11 [8, 13]	2 [3, 3]
Seaquest	1977 [1658, 2285]	134 [20, 248]
Skiing	-30000 [-30000, -30000]	-30000 [-30000, -30000]
Solaris	1196 [814, 1578]	878 [8, 1748]
SpaceInvaders	504 [461, 538]	207 [198, 216]
StarGunner	1020 [1000, 1040]	930 [710, 1150]
Surround	-6 [-8, -4]	-9 [-9, -9]
Tennis	-2 [-6, 0]	-12 [-24, -2]
TimePilot	3564 [2586, 4212]	2195 [2040, 2350]
Tutankham	133 [126, 145]	14 [0, 30]
UpNDown	15889 [7216, 29047]	1692 [1526, 1859]
Venture	0 [0, 0]	0 [0, 0]
VideoPinball	18537 [12410, 25939]	6961 [6299, 7623]
WizardOfWor	1970 [1786, 2154]	575 [550, 600]
YarsRevenge	28638 [25422, 31525]	148 [0, 297]
Zaxxon	2684 [1234, 4144]	0 [0, 0]
Mean	-	-1.90 [-1.91, -1.88]
Median	-	-0.64 [-0.64, -0.60]
IQM	-	-0.66 [-0.69, -0.65]

Task	MR.Q	Non-linear model	MSE reward loss	No reward scaling
Alien	1884 [1466, 2317]	2167 [1426, 3169]	734 [617, 856]	2074 [1402, 2821]
Amidar	318 [233, 391]	466 [364, 570]	157 [140, 177]	402 [345, 454]
Assault	1162 [1063, 1274]	1033 [998, 1068]	923 [873, 987]	1235 [1140, 1330]
Asterix	2413 [1925, 2760]	1987 [1560, 2414]	2503 [2050, 3040]	2476 [2051, 2901]
Asteroids	674 [561, 778]	563 [424, 740]	765 [624, 952]	614 [520, 711]
Atlantis	609450 [372300, 778910]	444370 [241216, 647524]	87410 [31610, 153510]	608658 [262742, 946168]
BankHeist	295 [203, 425]	1006 [961, 1042]	245 [195, 309]	490 [176, 806]
BattleZone	16120 [9540, 24220]	23820 [20300, 26200]	4566 [3900, 5200]	13660 [8040, 18780]
BeamRider	1903 [1782, 2022]	1904 [1777, 2046]	1489 [1446, 1543]	1989 [1947, 2031]
Berzerk	418 [399, 441]	527 [498, 579]	334 [295, 398]	427 [361, 524]
Bowling	66 [63, 70]	69 [58, 82]	30 [27, 35]	78 [66, 88]
Boxing	93 [91, 96]	95 [91, 98]	93 [89, 98]	91 [87, 96]
Breakout	37 [26, 50]	17 [17, 18]	11 [7, 18]	22 [20, 26]
Centipede	12542 [10782, 14550]	10053 [6514, 13594]	11624 [8061, 16184]	15952 [12806, 19014]
ChopperCommand	3880 [2850, 4814]	2918 [2006, 3830]	2806 [1610, 3570]	2796 [2228, 3378]
CrazyClimber	104544 [100800, 109334]	103950 [98066, 110282]	107220 [104990, 109150]	105014 [94550, 112412]
Defender	26584 [21155, 32013]	24283 [22936, 25425]	15231 [7875, 21485]	30912 [27871, 33695]
DemonAttack	4336 [3284, 5390]	2467 [1370, 3548]	311 [288, 331]	4893 [4282, 5345]
DoubleDunk	-14 [-20, -8]	-10 [-14, -8]	-10 [-14, -9]	-11 [-14, -8]
Enduro	1367 [1258, 1476]	1117 [1059, 1173]	800 [694, 899]	1450 [1311, 1593]
FishingDerby	-31 [-37, -26]	-33 [-36, -30]	-71 [-75, -66]	-22 [-25, -20]
Freeway	31 [32, 32]	31 [31, 32]	30 [30, 31]	25 [13, 32]
Frostbite	4156 [1809, 6712]	2693 [834, 4491]	3954 [266, 7285]	3247 [1532, 4771]
Gopher	11752 [7257, 17674]	7216 [3645, 11049]	2484 [886, 3514]	5802 [1467, 13322]
Gravitar	295 [226, 356]	309 [156, 419]	140 [50, 310]	256 [215, 305]
Hero	8057 [5442, 11106]	7635 [7577, 7693]	933 [0, 2799]	8775 [7575, 11125]
IceHockey	-1 [-3, -1]	-1 [-2, -1]	-8 [-9, -7]	-3 [-5, -2]
Jamesbond	564 [519, 611]	495 [462, 538]	355 [280, 455]	543 [460, 610]
Kangaroo	5388 [3012, 7180]	5732 [3148, 7954]	2793 [540, 7060]	6148 [3412, 8600]
Krull	9033 [8405, 9663]	8396 [8178, 8593]	8886 [7771, 9458]	8878 [7898, 9491]
KungFuMaster	27832 [21968, 32936]	24644 [19158, 30310]	19536 [12710, 31840]	24292 [18954, 29568]
MontezumaRevenge	100 [0, 260]	240 [80, 400]	0 [0, 0]	0 [0, 0]
MsPacman	4347 [3736, 4741]	3721 [3169, 4499]	1457 [1382, 1549]	4086 [3847, 4413]
NameThisGame	8077 [7400, 8907]	6162 [5941, 6450]	6091 [5656, 6475]	8323 [7308, 9338]
Phoenix	4876 [4628, 5085]	4611 [4300, 4800]	3638 [3317, 3959]	4940 [4541, 5255]
Pitfall	0 [0, 0]	-3 [-9, 0]	-22 [-68, 0]	0 [0, 0]
Pong	15 [12, 18]	16 [15, 19]	13 [8, 17]	15 [14, 18]
PrivateEye	100 [100, 100]	100 [100, 100]	33 [0, 100]	40 [0, 80]
Qbert	4600 [4390, 4808]	4295 [4006, 4586]	861 [785, 968]	2848 [1410, 4287]
Riverraid	5267 [3820, 6714]	6679 [5053, 7770]	3418 [482, 6556]	6669 [5104, 7779]
RoadRunner	38842 [32362, 44588]	26678 [19418, 32016]	24583 [15870, 35950]	37306 [32906, 40582]
Robotank	11 [8, 13]	10 [8, 13]	8 [2, 13]	15 [13, 17]
Seaquest	1977 [1658, 2285]	2344 [1541, 3450]	1676 [368, 2336]	1998 [1141, 2626]
Skiing	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]
Solaris	1196 [814, 1578]	1280 [498, 2062]	870 [370, 1736]	1175 [660, 1770]
SpaceInvaders	504 [461, 538]	536 [405, 667]	253 [235, 278]	458 [431, 493]
StarGunner	1020 [1000, 1040]	1014 [994, 1048]	976 [960, 1000]	1074 [964, 1266]
Surround	-6 [-8, -4]	-5 [-7, -5]	-8 [-9, -8]	-5 [-7, -5]
Tennis	-2 [-6, 0]	0 [-1, 0]	-2 [-5, 0]	0 [0, 0]
TimePilot	3564 [2586, 4212]	3822 [3282, 4418]	2376 [2070, 2880]	3118 [2620, 3618]
Tutankham	133 [126, 145]	138 [127, 151]	37 [0, 112]	144 [87, 184]
UpNDown	15889 [7216, 29047]	34574 [9812, 71080]	4568 [4174, 4972]	13859 [6662, 21316]
Venture	0 [0, 0]	74 [0, 210]	0 [0, 0]	0 [0, 0]
VideoPinball	18537 [12410, 25939]	14689 [10497, 17911]	11244 [9717, 12780]	18345 [15176, 21920]
WizardOfWor	1970 [1786, 2154]	1852 [1650, 2062]	1190 [1000, 1430]	1906 [1070, 2984]
YarsRevenge	28638 [25422, 31525]	29495 [25242, 32299]	14267 [10884, 16770]	27358 [21924, 31727]
Zaxxon	2684 [1234, 4144]	3144 [1128, 5118]	0 [0, 0]	1336 [0, 3300]
Mean	-	-0.28 [-0.53, -0.03]	-1.00 [-1.07, -0.93]	-0.03 [-0.46, 0.36]
Median	-	-0.00 [-0.02, -0.00]	-0.17 [-0.24, -0.14]	0.00 [-0.01, 0.01]
IQM	-	-0.02 [-0.05, -0.00]	-0.20 [-0.24, -0.18]	-0.01 [-0.02, 0.03]

Task	MR.Q	MSE terminal loss	No min	No LAP
Alien	1884 [1466, 2317]	2428 [1859, 2976]	2375 [1921, 3052]	2932 [2168, 3690]
Amidar	318 [233, 391]	411 [365, 458]	567 [465, 686]	415 [380, 477]
Assault	1162 [1063, 1274]	1182 [1140, 1225]	1154 [967, 1254]	1233 [1165, 1275]
Asterix	2413 [1925, 2760]	2390 [2136, 2614]	2881 [2600, 3045]	2735 [2090, 3165]
Asteroids	674 [561, 778]	629 [564, 696]	769 [713, 878]	571 [458, 770]
Atlantis	609450 [372300, 778910]	669246 [468396, 842242]	386213 [77480, 941750]	488080 [170330, 941990]
BankHeist	295 [203, 425]	403 [217, 594]	304 [205, 436]	169 [144, 203]
BattleZone	16120 [9540, 24220]	19120 [14830, 22560]	21300 [20100, 22200]	19400 [17800, 21100]
BeamRider	1903 [1782, 2022]	1888 [1838, 1935]	1989 [1948, 2016]	2007 [1911, 2089]
Berzerk	418 [399, 441]	495 [449, 544]	601 [398, 708]	510 [435, 578]
Bowling	66 [63, 70]	69 [66, 73]	38 [28, 50]	61 [55, 69]
Boxing	93 [91, 96]	96 [95, 97]	95 [92, 97]	97 [98, 98]
Breakout	37 [26, 50]	22 [21, 24]	27 [24, 30]	45 [17, 93]
Centipede	12542 [10782, 14550]	15317 [13231, 17456]	11288 [8365, 16843]	13510 [12038, 14695]
ChopperCommand	3880 [2850, 4814]	3792 [3234, 4422]	3283 [2330, 4770]	3776 [2710, 5180]
CrazyClimber	104544 [100800, 109334]	102474 [97044, 108223]	110046 [107390, 113610]	90546 [75700, 99520]
Defender	26584 [21155, 32013]	25138 [20950, 28778]	32336 [27280, 38695]	30935 [19430, 44430]
DemonAttack	4336 [3284, 5390]	5667 [5073, 6198]	2468 [712, 3513]	6314 [6032, 6873]
DoubleDunk	-14 [-20, -8]	-12 [-17, -9]	-9 [-13, -7]	-9 [-11, -9]
Enduro	1367 [1258, 1476]	1191 [914, 1397]	967 [0, 1461]	1386 [1249, 1592]
FishingDerby	-31 [-37, -26]	-33 [-40, -26]	-17 [-28, -13]	-27 [-36, -14]
Freeway	31 [32, 32]	32 [32, 32]	32 [31, 34]	32 [32, 32]
Frostbite	4156 [1809, 6712]	1492 [262, 2887]	2401 [267, 4457]	2842 [1798, 3550]
Gopher	11752 [7257, 17674]	15602 [6322, 27469]	11774 [7552, 18634]	9414 [7522, 12320]
Gravitar	295 [226, 356]	121 [56, 191]	393 [185, 570]	351 [325, 395]
Hero	8057 [5442, 11106]	6978 [5971, 7603]	7594 [7525, 7670]	9523 [7242, 13556]
IceHockey	-1 [-3, -1]	-1 [-2, 0]	-2 [-3, -1]	-1 [-3, 0]
Jamesbond	564 [519, 611]	557 [486, 637]	546 [470, 650]	436 [270, 655]
Kangaroo	5388 [3012, 7180]	7127 [4428, 9596]	8033 [5540, 9390]	9380 [7080, 13780]
Krull	9033 [8405, 9663]	8723 [8307, 9149]	8430 [6464, 9987]	8099 [6716, 9360]
KungFuMaster	27832 [21968, 32936]	25514 [24145, 26801]	25553 [23960, 27040]	27920 [25740, 29670]
MontezumaRevenge	100 [0, 260]	20 [0, 50]	190 [100, 370]	33 [0, 100]
MsPacman	4347 [3736, 4741]	4075 [3424, 4685]	3860 [3394, 4470]	5294 [4943, 5905]
NameThisGame	8077 [7400, 8907]	8102 [7682, 8540]	7992 [7194, 8458]	7922 [6678, 10006]
Phoenix	4876 [4628, 5085]	5045 [4849, 5252]	4780 [4605, 5008]	4883 [4713, 5131]
Pitfall	0 [0, 0]	0 [-1, 0]	0 [0, 0]	0 [0, 0]
Pong	15 [12, 18]	14 [13, 16]	18 [19, 19]	14 [12, 16]
PrivateEye	100 [100, 100]	90 [70, 100]	125 [100, 177]	100 [100, 100]
Qbert	4600 [4390, 4808]	3661 [2872, 4283]	6567 [4638, 7722]	4476 [4200, 4798]
Riverraid	5267 [3820, 6714]	5471 [4112, 6785]	7464 [7213, 7688]	5467 [3674, 7901]
RoadRunner	38842 [32362, 44588]	38960 [34429, 43161]	38703 [35530, 45050]	39616 [34460, 46920]
Robotank	11 [8, 13]	11 [10, 12]	15 [10, 19]	14 [11, 16]
Seaquest	1977 [1658, 2285]	2355 [1940, 2861]	2005 [1596, 2348]	2007 [1762, 2186]
Skiing	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]
Solaris	1196 [814, 1578]	1056 [715, 1518]	772 [216, 1606]	1197 [912, 1726]
SpaceInvaders	504 [461, 538]	453 [412, 495]	515 [488, 532]	408 [387, 444]
StarGunner	1020 [1000, 1040]	1007 [990, 1030]	11993 [1240, 22010]	986 [960, 1040]
Surround	-6 [-8, -4]	-6 [-7, -6]	-5 [-6, -4]	-6 [-7, -5]
Tennis	-2 [-6, 0]	0 [0, 0]	6 [-4, 20]	-1 [-2, -1]
TimePilot	3564 [2586, 4212]	3735 [3230, 4152]	4433 [4190, 4830]	3263 [2810, 3860]
Tutankham	133 [126, 145]	131 [116, 147]	156 [154, 160]	196 [165, 240]
UpNDown	15889 [7216, 29047]	24052 [8033, 48260]	37177 [17986, 54020]	27202 [5318, 61104]
Venture	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]
VideoPinball	18537 [12410, 25939]	19949 [16946, 23123]	22721 [18131, 28516]	22686 [18506, 29417]
WizardOfWor	1970 [1786, 2154]	1905 [1757, 2085]	1790 [1560, 2010]	2153 [1720, 2520]
YarsRevenge	28638 [25422, 31525]	24064 [22031, 26005]	26211 [21118, 33567]	22918 [20813, 26077]
Zaxxon	2684 [1234, 4144]	2262 [1188, 3524]	6746 [4940, 7760]	2806 [500, 4290]
Mean	-	0.14 [-0.09, 0.33]	-0.08 [-0.31, 0.37]	-0.01 [-0.34, 0.56]
Median	-	0.00 [-0.00, 0.01]	0.01 [0.00, 0.03]	0.01 [-0.01, 0.02]
IQM	-	0.00 [-0.01, 0.01]	0.03 [0.02, 0.06]	0.02 [-0.01, 0.05]

Task	MR.Q	No MR	1-step return	No unroll
Alien	1884 [1466, 2317]	2886 [2458, 3315]	1342 [1226, 1454]	3056 [2300, 3614]
Amidar	318 [233, 391]	312 [204, 421]	240 [213, 287]	385 [335, 471]
Assault	1162 [1063, 1274]	1105 [1075, 1142]	1045 [904, 1134]	1079 [1003, 1140]
Asterix	2413 [1925, 2760]	2298 [1920, 2731]	2708 [2110, 3710]	2388 [2190, 2740]
Asteroids	674 [561, 778]	485 [387, 584]	723 [689, 783]	581 [406, 766]
Atlantis	609450 [372300, 778910]	12834 [9038, 17186]	60986 [25200, 87460]	57426 [40850, 88730]
BankHeist	295 [203, 425]	404 [73, 726]	123 [77, 160]	783 [231, 1080]
BattleZone	16120 [9540, 24220]	22000 [12100, 29080]	4700 [3600, 5800]	23733 [19600, 26100]
BeamRider	1903 [1782, 2022]	1849 [1726, 2004]	1438 [1213, 1604]	1389 [1274, 1535]
Berzerk	418 [399, 441]	437 [333, 537]	361 [290, 400]	443 [399, 489]
Bowling	66 [63, 70]	84 [73, 95]	44 [30, 67]	72 [60, 88]
Boxing	93 [91, 96]	92 [89, 95]	93 [92, 95]	95 [94, 96]
Breakout	37 [26, 50]	15 [14, 18]	14 [7, 23]	27 [17, 44]
Centipede	12542 [10782, 14550]	10517 [9157, 11949]	8927 [7098, 10732]	11984 [11005, 13691]
ChopperCommand	3880 [2850, 4814]	3394 [3140, 3764]	2520 [1980, 2900]	2866 [1670, 3990]
CrazyClimber	104544 [100800, 109334]	83734 [71466, 93070]	66980 [65140, 70490]	104076 [82650, 114820]
Defender	26584 [21155, 32013]	14469 [9247, 18763]	9658 [3245, 17870]	34718 [29120, 44180]
DemonAttack	4336 [3284, 5390]	746 [527, 994]	1861 [1675, 1990]	2840 [2064, 3618]
DoubleDunk	-14 [-20, -8]	-14 [-18, -11]	-7 [-10, -6]	-11 [-14, -10]
Enduro	1367 [1258, 1476]	897 [784, 1075]	1064 [1055, 1069]	1075 [1018, 1131]
FishingDerby	-31 [-37, -26]	-21 [-30, -8]	-82 [-95, -64]	-54 [-63, -45]
Freeway	31 [32, 32]	26 [13, 33]	32 [32, 33]	32 [32, 33]
Frostbite	4156 [1809, 6712]	3098 [1645, 4358]	1231 [254, 3182]	4182 [3473, 5451]
Gopher	11752 [7257, 17674]	1833 [1552, 2114]	5597 [4242, 6512]	4746 [2378, 6822]
Gravitar	295 [226, 356]	89 [0, 267]	166 [70, 310]	270 [145, 415]
Hero	8057 [5442, 11106]	7584 [7548, 7644]	7594 [7544, 7676]	6879 [5384, 7694]
IceHockey	-1 [-3, -1]	-5 [-8, -4]	-6 [-8, -5]	-2 [-4, -1]
Jamesbond	564 [519, 611]	433 [406, 462]	518 [495, 545]	546 [460, 625]
Kangaroo	5388 [3012, 7180]	7508 [5608, 9500]	6940 [1520, 10460]	9166 [8520, 9500]
Krull	9033 [8405, 9663]	8403 [7433, 9087]	7386 [6611, 7881]	7785 [7176, 8271]
KungFuMaster	27832 [21968, 32936]	19066 [15450, 22028]	15300 [14760, 16350]	29020 [27150, 30700]
MontezumaRevenge	100 [0, 260]	0 [0, 0]	0 [0, 0]	0 [0, 0]
MsPacman	4347 [3736, 4741]	3297 [2679, 3908]	2282 [1956, 2592]	2839 [2612, 2954]
NameThisGame	8077 [7400, 8907]	3638 [3207, 3999]	5590 [5026, 5941]	6529 [6069, 6983]
Phoenix	4876 [4628, 5085]	4101 [3752, 4503]	3825 [3435, 4306]	4941 [4613, 5193]
Pitfall	0 [0, 0]	-27 [-67, -5]	0 [0, 0]	0 [0, 0]
Pong	15 [12, 18]	12 [8, 16]	14 [9, 20]	12 [8, 16]
PrivateEye	100 [100, 100]	3068 [24, 9080]	100 [100, 100]	100 [100, 100]
Qbert	4600 [4390, 4808]	4491 [3362, 5870]	2517 [2032, 3105]	4220 [3918, 4508]
Riverraid	5267 [3820, 6714]	7479 [6818, 8239]	6733 [5928, 7759]	5856 [3607, 7855]
RoadRunner	38842 [32362, 44588]	29182 [23250, 35114]	36145 [33590, 38700]	37636 [33610, 40160]
Robotank	11 [8, 13]	11 [6, 16]	6 [6, 7]	12 [11, 17]
Seaquest	1977 [1658, 2285]	1556 [1248, 1866]	2166 [2154, 2178]	2194 [2112, 2284]
Skiing	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]	-30000 [-30000, -30000]
Solaris	1196 [814, 1578]	552 [319, 881]	1107 [662, 1552]	1088 [638, 1386]
SpaceInvaders	504 [461, 538]	550 [476, 632]	389 [382, 396]	551 [502, 596]
StarGunner	1020 [1000, 1040]	1272 [1020, 1730]	1290 [1000, 1580]	1423 [990, 2220]
Surround	-6 [-8, -4]	-8 [-9, -7]	-5 [-6, -5]	-5 [-10, 1]
Tennis	-2 [-6, 0]	0 [-2, 0]	0 [-1, 0]	0 [-2, 0]
TimePilot	3564 [2586, 4212]	2440 [1540, 3178]	2535 [2030, 3040]	2236 [1170, 3810]
Tutankham	133 [126, 145]	112 [65, 148]	151 [120, 182]	148 [127, 181]
UpNDown	15889 [7216, 29047]	25451 [17297, 34036]	4477 [2993, 5962]	31342 [3170, 84771]
Venture	0 [0, 0]	0 [0, 0]	0 [0, 0]	0 [0, 0]
VideoPinball	18537 [12410, 25939]	8524 [3899, 12519]	13506 [7955, 19058]	27525 [22242, 35815]
WizardOfWor	1970 [1786, 2154]	2058 [1632, 2640]	1545 [1490, 1600]	1573 [1430, 1700]
YarsRevenge	28638 [25422, 31525]	30666 [28659, 33225]	18513 [16940, 20088]	19082 [11442, 24565]
Zaxxon	2684 [1234, 4144]	5758 [4712, 6962]	330 [0, 660]	0 [0, 0]
Mean	-	-0.99 [-1.09, -0.90]	-0.91 [-1.02, -0.80]	-0.54 [-0.61, -0.48]
Median	-	-0.02 [-0.08, -0.01]	-0.16 [-0.19, -0.10]	-0.00 [-0.02, 0.00]
IQM	-	-0.07 [-0.13, -0.04]	-0.16 [-0.18, -0.15]	-0.02 [-0.03, -0.01]