A TRAINING DETAILS

Experiments on Control Suite and Matrix Game tasks were conducted on a single NVIDIA V100 GPU with 4 CPU cores (state-based) or 20 CPU cores (pixel-based). Experiments in MetaWorld and Isaac Gym were conducted on a single NVIDIA 2080Ti with 4 CPU cores. Benchmark performance of DecQN is reported in terms of the mean and one standard deviation around the mean on 10 seeds. We implemented DecQN within the Acme framework (Hoffman et al., 2020) in TensorFlow. For the Mini Cheetah (Katz et al., 2019) task we implement a version of DecQN in PyTorch.

Hyperparameters We provide hyperparameter values of DecQN used for benchmarking in Table 2. A constant set of hyperparameters is used throughout all experiments, with modifications to the network architecture for vision-based tasks. For the matrix games, we further set the n-step parameter to 1 to account for the underlying timescale and direct impact that actions have on both the state transition and reward. Results for DQN were obtained with the same parameters without decoupling. The baseline algorithms used the default parameter settings provided in the official implementations by the original authors, within the Acme library in the case of D4PG and DMPO.

Table 2: DecQN hyperparameters for state- and pixel-based control.

Parameter	Value [State]	Value [Pixel]
Optimizer	Adam	Adam
Learning rate	1×10^{-4}	1×10^{-4}
Replay size	1×10^{6}	1×10^{6}
n-step returns	3	3
Action repeat	1	1
Discount γ	0.99	0.99
Batch size	256	256
Hidden size	500	1024
Bottleneck size	—	100
Gradient clipping	40	40
Target update period	100	100
Imp. sampling exponent	0.2	0.2
Priority exponent	0.6	0.6
Exploration ϵ	0.1	0.1

Architecture For the state-based experiments, we leverage a fully-connected architecture that includes a single residual block followed by a layer norm. For the vision-based experiments, we leverage the convolutional encoder with bottleneck layer from Yarats et al. (2021) followed by two fully-connected layers. In both cases, a fully-connected output layer predicts the decoupled state-action values so that the torso up to the final layer remains shared among the decoupled critics.

Action discretization The continuous action space is discretized along each action dimension into evenly spaced discrete actions including the boundary values. Assuming symmetric action bounds, for axis i this yields bang-bang control in the case of 2 bins with $\mathcal{A}_{n_{\text{bin}=2},i} = \{-a_{\text{bound},i}, +a_{\text{bound},i}\}$ and bang-off-bang control in the case of 3 bins with $\mathcal{A}_{n_{\text{bin}=3},i} = \{-a_{\text{bound},i}, 0, +a_{\text{bound},i}\}$, and so on.

Decoupling The decoupling based on value decomposition (Sunehag et al., 2017) only requires small modifications of the original DQN structure (see also Sharma et al. (2017)). The output layer size is adapted to predict all state-action utilities for action dimensions n_a and discrete bins n_b to yield output size $[..., n_a n_b]$, where $[...]$ indicates the batch dimensions. This is reshaped into $[..., n_a, n_b]$, where the combined state-action value is computed either by indexing with the observed bin at the current state or maximizing over bins at the next state, followed by a mean over action dimensions to recover state-action values.

Licenses The Acme library is distributed under the Apache-2.0 license and available on GitHub. The D4PG and DMPO agents are part of the Acme library, while both DecQN and DQN for continuous control were implemented within the Acme framework. Both Dreamer-v2 and DrQ-v2 are

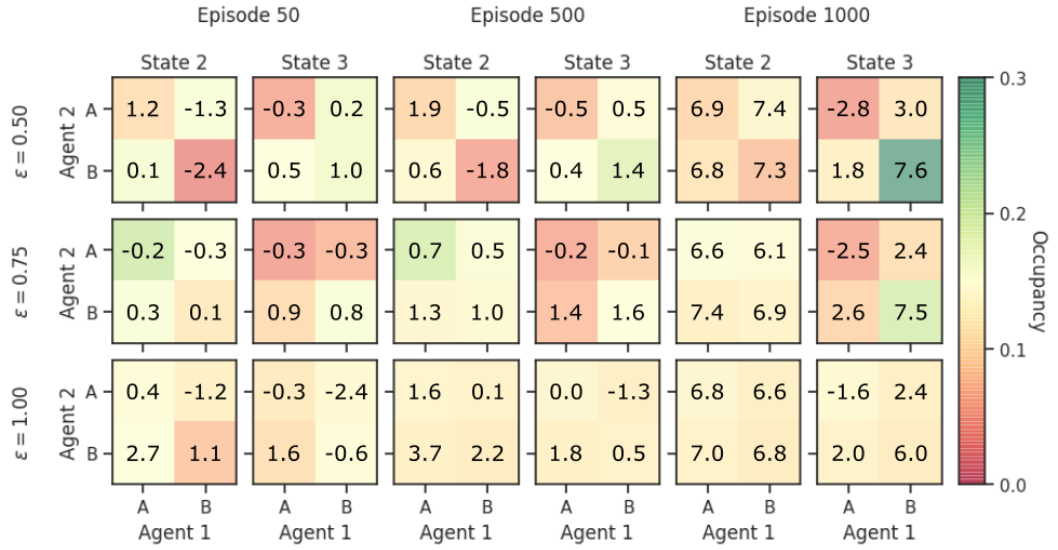

Figure 9: State distribution and action-values for DecQN on the two-step game from Section 5.1. We consider 3 training stages pre-convergence across 5 seeds. Color indicates cumulative state-action occurrence, numbers represent predicted mean action values. While the decoupled agent struggles to represent action values under a uniform distribution ($\epsilon = 1.0$, bottom), it accurately represents values under the current policy when directly influencing the state distribution ($\epsilon = 0.5$, top).

distributed under the MIT license and are available with their benchmarking results on GitHub here and here, respectively.

B State-Action Distribution: Two-Step Game

Expanding on the discussion of the two-step matrix game in Section 5.1 we provide state-action distribution data for the DecQN agent in Figure 9. We evaluate three stages of early training before convergence for 5 seeds. Color of state-action pairings indicates cumulative frequency within replay memory, while numbers represent mean predicted state-action values. We consider different degrees of randomness in the policy by varying the ϵ parameter. It can be noted that the agent fails to accurately represent the optimal state-action value in state 3 when learning based on a uniform state-action distribution (bottom row). However, enabling the agent to directly influence the underlying distribution through its policy results in accurate learning of state-action values around this policy (top and middle rows). While the decoupled agent may not be able to accurately reflect the full state-action value function, it can be sufficient to do so over a subspace relevant for solving the task.

C Environments

A brief overview of the environments evaluated throughout this paper is provide in Table 3, grouped based on their distributors. In particular, we highlight the dimensionality of the state space $\mathcal{S}$ and action space $\mathcal{A}$ as well as the total number of timesteps used for training on each environment. For the Mini Cheetah task, we note that the number of steps - marked with an $*$ - denotes the approximate number of steps per environment, where we use 1024 parallel environments within Isaac Gym.

D Additional baselines

We provide additional baselines on a selection of Control Suite environments in Figure 10. The baselines consist of a continuous-control algorithm with Categorical head based on MPO from Seyde et al. (2021), the continuous-control SAC agent from Yarats & Kostrikov (2020), as well as the critic-only methods QT-Opt and AQL-Seq based on data from Van de Wiele et al. (2020). DecQN is

Table 3: Description of benchmark environments used throughout the paper.

Suite	Task	$dim(\mathcal{S})$	$dim(\mathcal{A})$	Steps [State]	Steps [Pixel]
Control Suite	Ball in Cup Catch	8	2	—	1×10^6
	Cartpole Swingup	4	1	1×10^6	1×10^6
	Cartpole Swingup Sparse	4	1	2×10^6	1×10^6
	Cheetah Run	18	6	3×10^6	1×10^6
	Dog Run	158	38	5×10^6	—
	Dog Trot	158	38	5×10^6	—
	Dog Walk	158	38	5×10^6	—
	Finger Spin	6	2	2×10^6	1×10^6
	Finger Turn Hard	6	2	2×10^6	1×10^6
	Humanoid Run	54	21	30×10^6	—
	Humanoid Stand	54	21	10×10^6	—
	Humanoid Walk	54	21	20×10^6	30×10^6
	Quadruped Run	56	12	5×10^6	10×10^6
	Reacher Hard	4	2	2×10^6	1×10^6
	Walker Run	18	6	3×10^6	—
	Walker Walk	18	6	1×10^6	1×10^6
Isaac Gym	Mini Cheetah Tracking	48	12	$3 \times 10^{6*}$	—
Matrix Games	Two Step	3	4	6×10^4	—
	Penalty	4	9	2×10^5	—
	Climbing	4	9	2×10^5	—
Meta World	Assembly	39	4	2×10^6	—
	Door Open	39	4	2×10^6	—
	Drawer Open	39	4	2×10^6	—
	Hammer	39	4	2×10^6	—
	Pick Place	39	4	2×10^6	—

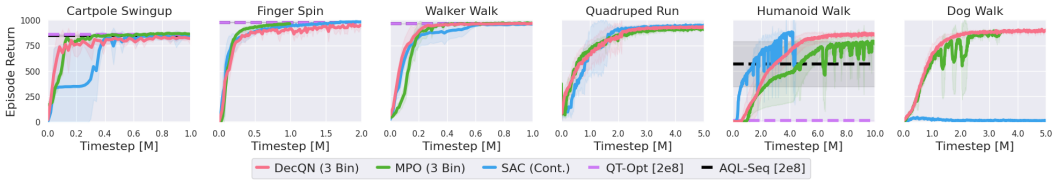

Figure 10: Comparison of DecQN to MPO with Categorical policy head, continuous SAC, as well as critic-only QT-Opt and AQL-Seq. DecQN displays state-of-the-art performance without relying on actor-critic methods or continuous control, while remaining sample-efficient in high-dimensional action spaces by avoiding sampling-based methods.

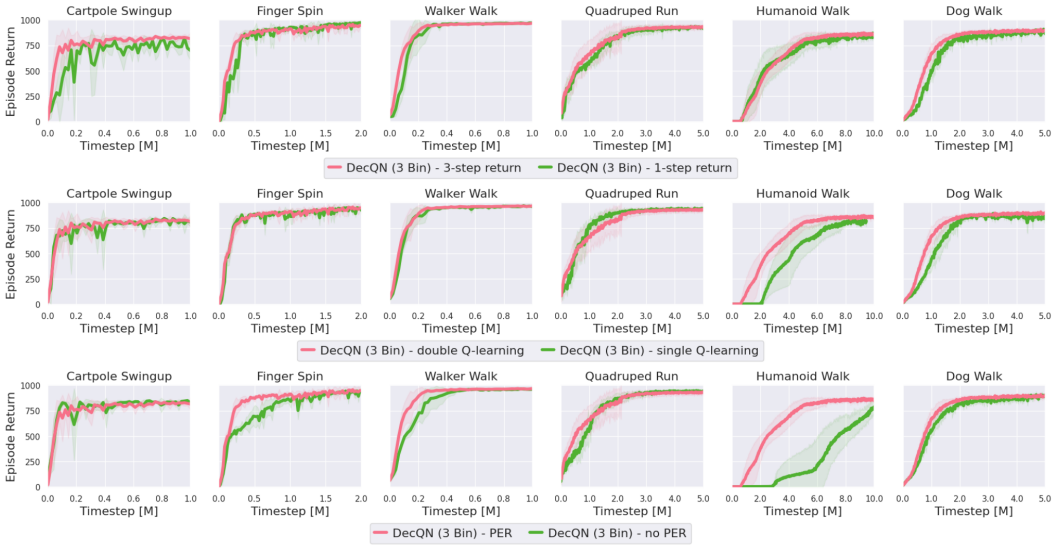

Figure 11: Ablations of the DecQN agent on multi-step returns, double Q-learning, and prioritized experience replay (top to bottom). The agent is robust to changes in individual components while profiting from PER to select useful interactions on complex multi-phase tasks such as the Humanoid.

competitive with discretized MPO, further underlining that actor-critic methods are not required to obtain strong benchmark performance. The additional SAC baseline shows improved performance on Humanoid at the cost of being unable to learn on the Dog task, mirroring results of Hansen et al. (2022). Generally, we believe that most current state-of-the-art continuous control algorithms are capable of achieving comparable benchmark performance. However, these results do not appear to be conditional on using continuous control or actor-critic methods, and can be achieved with much simpler Q-learning over discretized bang-bang action spaces with constant ϵ-greedy exploration. We further provide converged performance of QT-Opt and AQL-Seq. While these methods remain competitive on low-dimensional tasks, their performance quickly deteriorates for high-dimensional action spaces due to their reliance on sampling this space.

E Ablations on Rainbow components

We provide ablations on three components of the underlying Rainbow agent, namely multi-step returns, double Q-learning, and prioritized experience replay in Figure 11 (rows, respectively). Learning is generally robust to removal of individual components in light of cumulative reward at convergence, improving learning speed primarily on the more complex tasks. In particular, double Q-learning and PER provide a boost on the Humanoid task to enable state-of-the-art performance.

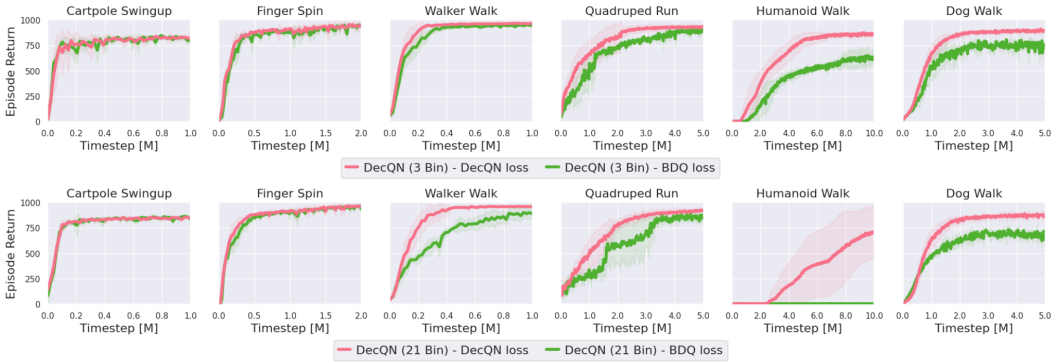

Figure 12: Comparison of DecQN with the original and BDQ loss for 3 (top) and 21 bins (bottom). Aggregating per-dimension utilities transforms independent into joint learners, yielding significant improvements in final performance and learning stability particularly for high-dimensional tasks.

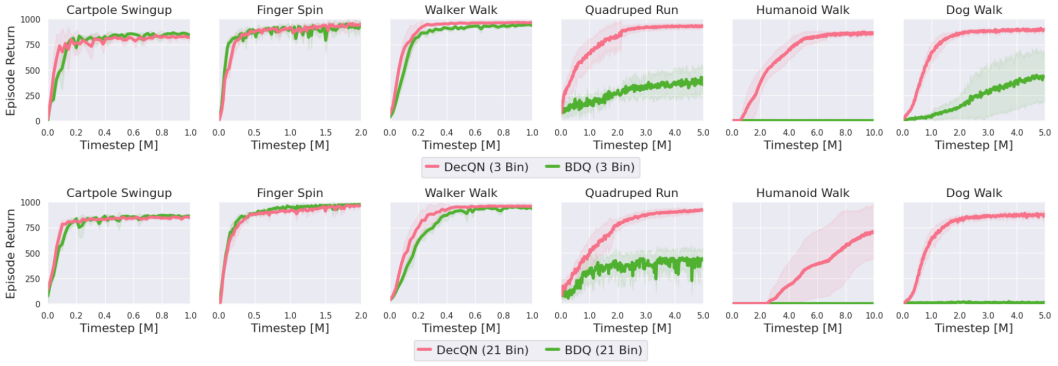

Figure 13: Comparison of DecQN and BDQ for 3 (top) and 21 bins (bottom). DecQN provides significant advantage on the more complex domains, underlining the importance of design choices such as strong architectural centralization and value decomposition via single-action utility functions.

F Discussion of BDQ and HGQN

In the following, we provide a more detailed discussion of the relation to Branching Dueling Q-Networks (BDQ) Tavakoli et al. (2018) as well as Hypergraph Q-Networks (HGQN) Tavakoli et al. (2021) together with associated ablations. Generally, our motivation is to highlight that minimal changes to the original DQN agent enable state-of-the-art performance on continuous control benchmarks solely based on bang-bang Q-learning with constant exploration.

The BDQ agent from Tavakoli et al. (2018) considers independent learning of per-dimension state action values, employing a dueling architecture with separate branches for each action dimension and exploration based on a Gaussian with scheduled noise in combination with fine-grained discretizations. DecQN forces a higher degree of centralization by predicting per-dimension state-action utilities without intermediate branching or dueling heads for joint learning within a value decomposition, while using constant ϵ-greedy exploration with a focus on only coarse bang-bang control. The most important difference is independent learning in comparison to joint learning based on value decomposition. We provide an ablation of our approach that does not aggregate per-action dimension values and thereby mimics BDQ's independent learning in Figure 12. Particularly for high-dimensional tasks learning a value decomposition can significantly improve performance, an effect that is amplified when selecting more fine-grained discretizations. We furthermore evaluated the original BDQ agent after modifying the default parameters (i.e. batch size, target update frequency, learning frequency, gradient clipping, exploration strategy, multi-step returns). The results provided in Figure 13 were obtained based on the best configuration we found by increasing the batch size and treating each episode as infinite-horizon. While we believe that the results in

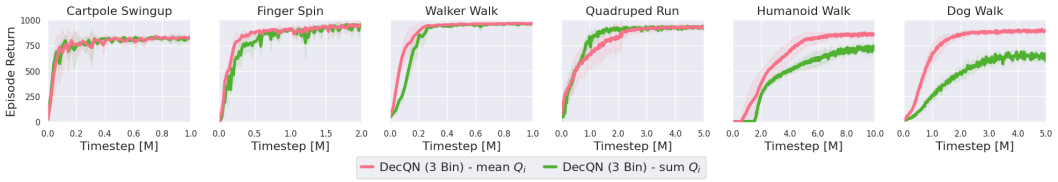

Figure 14: Comparison of DecQN with original and HGQN aggregation. Aggregating via the mean induces a more favorable scaling of the reward in the TD-error, which we found to enable graceful scaling to complex task with high-dimensional action spaces to achieve state-of-the-art performance.

Figure 12 offer a more informative comparison, we observe a similar trend regarding scaling to high-dimensional tasks in both cases.

The hypergraph Q-networks (HGQN) framework from Tavakoli et al. (2021) considers value decomposition across subsets of action dimensions and introduces higher-order hyperedges between action groupings. DecQN can therefore be interpreted as an instance of the conceptual HGQN (r=1) formulation that leverages single-action decomposition without higher-order edges as had previously been investigated with the Atari-based FARAQL agent (Sharma et al., 2017). Our primary focus is on simplicity and showing how far we can push basic concepts that constitute capable alternatives to more sophisticated recent algorithms. There are further several differences in per-dimension utility aggregation, architectural choices regarding the use of branching, loss function, exploration scheduling, as well as in the usage of PER and double Q-learning for continuous control. We provide a brief ablation on using the mean compared to sum aggregation when computing Bellman targets in Figure 14. While this appears as a subtle difference, we found that leveraging the mean yields graceful scaling to complex tasks with high-dimensional action spaces without any parameter adjustments. This enables state-of-the-art performance across a wide range of environments and input-output modalities with a single set of hyper-parameters.

Generally, our motivation is not to advocate for a novel algorithm that should be the go-to method for solving continuous control problems. Our core objective is to highlight that current state-of-the-art continuous control benchmark performance is at the level of decoupled Q-learning over bang-bang parameterized action spaces with constant exploration. This requires only minor if well-directed modification to the original DQN algorithm and yields strong performance for both feature- and pixel-based observation spaces as well as acceleration-, velocity-, and position-based action spaces. Our investigation provides additional motivation for existing work while establishing closer connections to classical MARL coordination challenges, as well as extensive experimental evaluation in comparison to current state-of-the-art algorithms.

G Rainbow DQN agent

We leverage several modifications of vanilla DQN that accelerate learning and improve stability based on the Rainbow implementation provided by Acme (Hessel et al., 2018; Hoffman et al., 2020):

Target Network Bootstrapping directly from the learned value function can lead to instabilities. Instead, evaluating actions based on a target network $Q_{\theta^-}(s_t, a_t)$ improves learning (Mnih et al., 2015). The target network's weights θ^- are updated periodically to match the online weights θ.

Double Q-learning Direct maximization based on the value target can yield overestimation error. Decoupling action selection from action evaluation improves stability (Van Hasselt et al., 2016). Action selection then queries the online network, while action evaluation queries the target network. Our implementation further leverages two sets of critics, where we bootstrap based on their average during learning and take their maximum during action selection.

Prioritized Experience Replay Uniform sampling from replay memory limits learning efficiency. Biasing sampling towards more informative transitions can accelerate learning (Schaul et al., 2015). The observed temporal difference error can serve as a proxy for expected future information content.

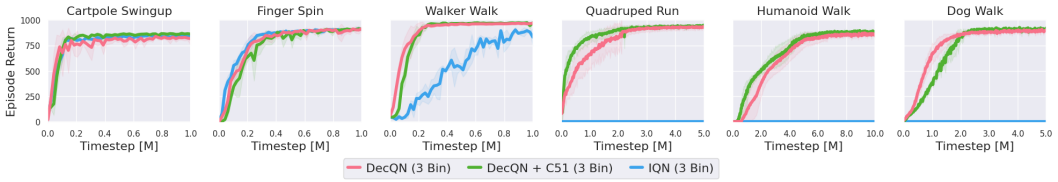

Figure 15: Comparison of DecQN, DecQN with a distributional C51 critic, and the distributional IQN agent. The distributional version of DecQN generally yields slightly improved performance at convergence, while decoupling improves performance over the non-decoupled IQN baseline.

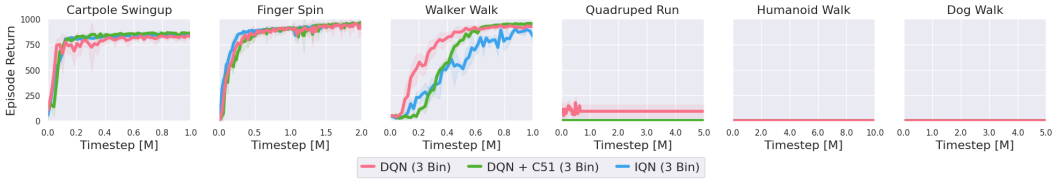

Figure 16: Comparison of DQN, DQN with a distributional C51 critic, and the distributional IQN agent. Without decoupling, we do not observe benefits of distributional critics in these domains. Slower convergence on Walker Walk with distributional representations could indicate that the associated increased parameter count translates to a more difficult optimization problem.

Multi-step Returns Instead of directly bootstrapping from the value function at the next state, evaluating the reward explicitly over several transitions as part of a multi-step return, such that $G_{t:t+n} = R_t + \cdots + \gamma^{n-1} R_{t+n-1} + \gamma^n V_{t+n}(S_{t+n})$ can improve learning (Sutton & Barto, 2018).

H Control-affine dynamics and linear rewards

Under deterministic environment dynamics and policy we can simplify the Bellman equation as

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma V^\pi(s_{t+1}). \tag{5}$$

Consider a reward structure that is linear in the action dimensions with $r(s_t, a_t) = \sum_{j=0}^{M} r^j(s_t, a_t^j)$. Under given transition tuples and fixed target values, the TD(0) objective can then be formulated as

$$Q^\pi(s_t, a_t) = \sum_{j=0}^{M} \left(r^j(s_t, a_t^j) + \frac{\gamma}{M} V^\pi(s_{t+1}) \right), \tag{6}$$

which can be solved exactly based on a linear value decomposition $Q(s_t, a_t) = \sum_{j=0}^{M} Q_j(s_t, a_t^j)$. Particularly for robotics applications, many common reward structures depend (approximately) linearly on the next state observation while the system dynamics are control-affine such that $s_{t+1} = f(s_t) + g(s_t)a_t$, with $f(s_t)$ and $g(s_t)$ only depending on the current state s_t. While these are strong assumptions, it may provide intuition for why the problem structures considered here may be amenable to decoupled local optimization and for the observed highly competitive performance.

I Distributional critics

We briefly investigate replacing our deterministic critic with a distributional C51 head (Bellemare et al., 2017) The decomposition now proceeds at the probability level via logits $\boldsymbol{l} = \sum_{j=1}^{M} \boldsymbol{l}_j / M$ during the C51 distribution matching. We do not make any parameter adjustments and compare performance with and without a distributional critic for both DecQN and DQN in Figures 15 and 16, respectively, and provide the IQN agent as an extension of the QR-DQN agent for reference (Dabney et al., 2018a;b). Our empirical evaluation suggests that a distributional critic can slightly increase performance at convergence and sample-efficiency. Both DecQN and DecQN + C51 yield similar performance on average with some environment specific variations. Without decoupling, DQN

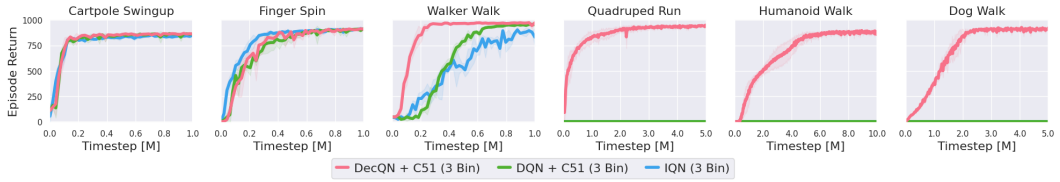

Figure 17: Comparison of the distributional versions of DecQN, DQN, and IQN. Decoupling of the value representation in conjunction with bang-bang action representations plays a key role in scaling these approaches to high-dimensional continuous control tasks.

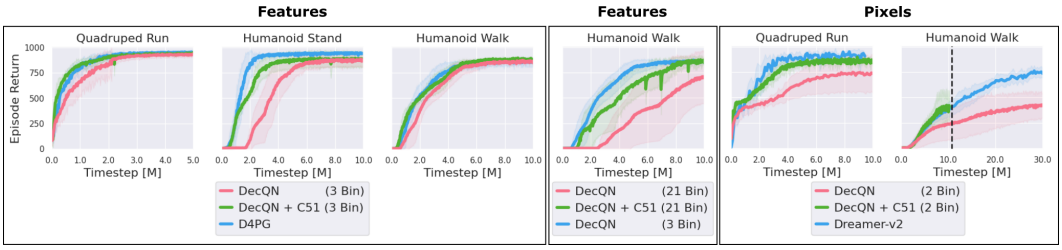

Figure 18: DecQN + C51 on tasks where regular DecQN did not perform at least as good as the best baseline. Adding a distributional critic can further boost performance, underlining the strength and versatility of this very simple approach (vertical line = current training status due to time constraints).

yields slightly faster convergence than DQN + C51 on Walker Walk which could indicate that the increased parameter count of distributional critics translates to a more challenging optimization problem (see also the immediate memory error of non-decoupled distributional agents on Quadruped Run in Figure 16.) Both DecQN and DQN with or without distributional critic improve performance over the IQN baseline. We further provide performance of only the distributional agents in Figure 17 for ease of comparison. It is likely that given sufficient tuning the performance of our distributional variations could be increased even further, while we note that the deterministic DecQN agent already matches performance of the distributional D4PG and DMPO actor-critic agents. A more extensive investigation into decoupled distributional Q-learning provides promising avenues for future work.

We note that without any parameter adjustments or tuning, replacing the deterministic critic with the distributional C51 critic in DecQN significantly boosts performance on the few environments where DecQN did not perform at least as good as the best baseline. This applies to large bin sizes, feature inputs and pixel inputs as exemplified in Figure 18 (vertical line in Humanoid Walk from pixels denotes current training status, subject to time constraints). These results further underline the versatility and strength of this very simple approach.

J STOCHASTIC ENVIRONMENTS

We extend our study to stochastic versions of a selection of Control Suite tasks. The results in Section 5.1 indicate that coordination among decoupled actors becomes more difficult if the action selection of other actors is less stationary from the perspective of each individual actor. To increase stochasticity, we consider both observation and reward noise represented by additive Gaussian white noise with standard deviation $\sigma_{\text{noise}} = 0.1$. Figures 19 and 20 provide results for DecQN, D4PG, and the distributional DecQN + C51 under observation and reward noise, respectively. While we observe similar performance of DecQN and D4PG on most tasks, performance of DecQN is visibly reduced on Humanoid Walk. Humanoid Walk combines several aspects that can hinder exploration, including implicit staged reward (first get up, then walk) and action penalties, which likely exacerbate coordination challenges when combined with stochasticity. We therefore believe that special care should be taken when applying regular DecQN in environments that have the potential to amplify the coordination challenges observed in the simple matrix game domains of Section 5.1. Adding a distributional critic to DecQN allows for better modelling of stochasticity and visibly improves

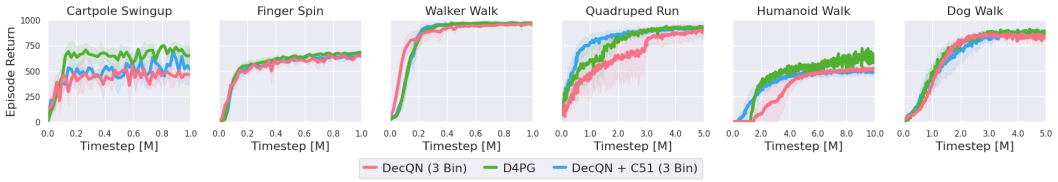

Figure 19: Stochastic observation tasks adding Gaussian white noise to the observations ($\sigma_{\text{noise}} = 0.1$). DecQN appears less robust to observation noise than D4PG, mirroring the findings regarding coordination challenges under high stochasticity in matrix games of Section 5.1. Adding a distributional critic improves performance and yields faster convergence than D4PG on Quadruped Run.

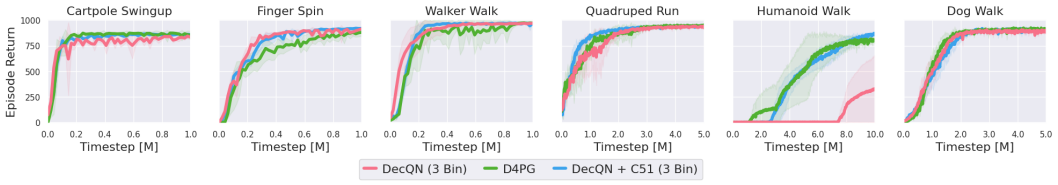

Figure 20: Stochastic reward tasks adding Gaussian white noise to the rewards ($\sigma_{\text{noise}} = 0.1$). DecQN is less robust to reward noise than the distributional D4PG, mirroring the findings regarding coordination challenges under high stochasticity in matrix games of Section 5.1. With a distributional critic, DecQN can directly account for stochastic returns and matches or outperforms D4PG.

performance in stochastic environments, where DecQN + C51 even improves on the distributional D4PG agent in some environments.