Quantized Representations Prevent Dimensional Collapse in Self-predictive RL

Aidan Scannell
Aalto University
aidan.scannell@aalto.fi

Kalle Kujanpää
Aalto University
kalle.kujanpaa@aalto.fi

Yi Zhao
Aalto University
yi.zhao@aalto.fi

Mohammadreza Nakhaei
Aalto University
mohammadreza.nakhaei@aalto.fi

Arno Solin
Aalto University
arno.solin@aalto.fi

Joni Pajarinen
Aalto University
joni.pajarinen@aalto.fi

Abstract

Learning representations for reinforcement learning (RL) has shown much promise for continuous control. We propose an efficient representation learning method using only a self-supervised latent-state consistency loss. Our approach employs an encoder and a dynamics model to map observations to latent states and predict future latent states, respectively. We achieve high performance and prevent dimensional collapse by quantizing the latent representation such that the rank of the representation is empirically preserved. Our method, named iQRL: **i**mplicitly **Q**uantized **R**einforcement **L**earning, is straightforward, compatible with any model-free RL algorithm, and demonstrates excellent performance by outperforming other recently proposed representation learning methods in continuous control benchmarks from DeepMind Control Suite.

1 Introduction

Reinforcement learning (RL, *e.g.*, [2]) has shown much promise for solving complex continuous control tasks. However, applying RL in real-world environments is challenging as it typically requires millions of data points which can be unpractical—*i.e.* RL is sample inefficient. On the other hand, representation learning has become a widely adopted solution for improving sample efficiency in deep learning. The core idea is to learn features which capture the underlying structure and patterns of the data. In the context of RL, such features can be learned independently from the downstream task. Whilst representation learning has had successes in RL, these have mainly been restricted to image-based observations (*e.g.*, CURL [3], DrQ [4], DrQ-v2 [5], and TACO [6]).

The investigation of representation learning for state-based RL is much less common. This is likely because learning a compact representation of an already compact state vector seems unnecessary. However, recent work suggests that the difficulty of a task is due to the complexity of the underlying transition dynamics, as opposed to the size of the observation space [7, 8]. As such, investigating representation learning for state-based RL is a promising research direction.

Recently, TCRL [8] and SPR [9] have obtained state-of-the-art performance on continuous control benchmarks by learning representations with self-supervised losses. Self-supervised learning (SSL) approaches (which do not reconstruct observations) learn good features without labels by minimizing

See Scannell et al. [1] for a longer 9 page version of this paper.

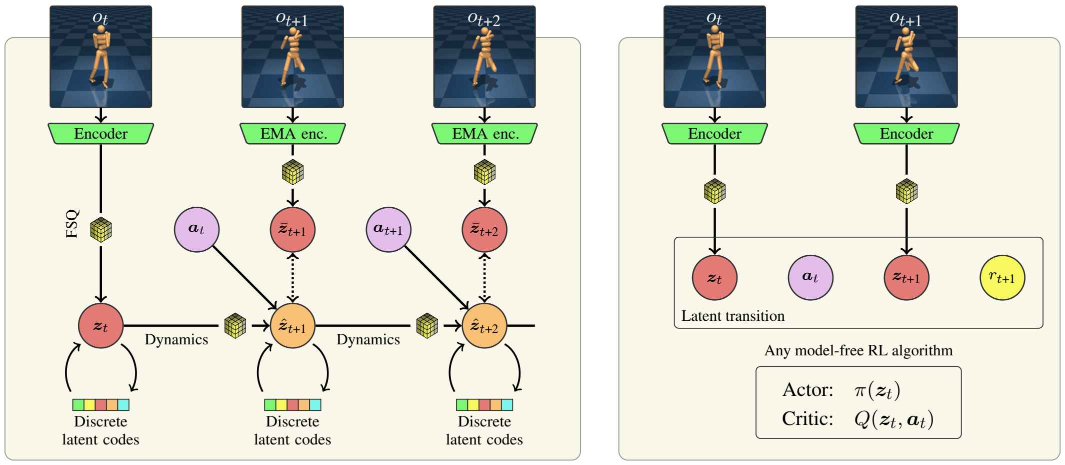

Figure 1: **Overview** iQRL is a stand-alone representation learning technique that is compatible with any model-free RL algorithm (we use TD3 [16]). Importantly, iQRL quantizes the latent representation with Finite Scalar Quantization (FSQ,), using only a self-supervised latent-state consistency loss, *i.e.* no decoder (see Eq. (5)). Making the latent representation discrete with an implicit codebook (🟩🟨🟥🟧🟦🟪) contributes to the very high sample efficiency of iQRL and empirically prevents dimensional collapse.

distances between two embedding vectors [10]. However, there is a trivial solution where the encoder outputs a constant for all inputs, known as representation collapse [11].

Definition 1.1 (Complete representation collapse)**.** Given an encoder $e_\theta : \mathcal{O} \rightarrow \mathcal{Z}$ which maps observations $\boldsymbol{o} \in \mathcal{O}$ to latent states $\boldsymbol{z} \in \mathcal{Z}$, the representation is said to be completely collapsed when the latent representation is constant for all observations, *i.e.*, $e_\theta(\boldsymbol{o}) = c, \forall \boldsymbol{o} \in \mathcal{O}$.

In the context of SSL, Jing et al. [11] investigated another type of representation collapse known as dimensional collapse.

Definition 1.2 (Dimensional collapse)**.** Let $e_\theta : \mathcal{O} \rightarrow \mathcal{Z}$ be an encoder which maps observations $\boldsymbol{o} \in \mathcal{O}$ to latent states $\boldsymbol{z}_t \in \mathcal{Z} = \mathbb{R}^d$, of dimension d. Given a data set of N latent states $\mathcal{D}_z = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$, the representation is said to be dimensionally collapsed when the latent states span a lower dimensional space than the original latent space $\mathcal{Z}$, *i.e.* $\dim(\mathrm{Span}(\mathcal{D}_z)) < d$.

Whilst complete representation collapse is a clear issue when learning representations for RL, it is not immediately obvious if dimensional collapse is an issue because the goal of representation learning is often considered to be learning a lower-dimensional representation. Our experiments show that whilst dimensional collapse is not always an issue, in some more complex environments, it can prevent agents from learning to solve a task (see Fig. 3). It is worth noting that previous approaches utilize auxiliary loss terms to help prevent representation collapse, *e.g.*, minimizing the reward prediction error in the latent space [12, 8, 13–15].

In this paper, we propose a simple representation learning technique which learns a task-agnostic representation using only a self-supervised loss. Importantly, our method empirically prevents dimensional collapse as it preserves the rank of the representation. We accomplish this by quantizing our latent representation with Finite Scalar Quantization [17], without using any reconstruction loss. As a result, our latent space is bounded and associated with an *implicit* codebook, whose size we can control. Our method can be combined with any model-free RL method (we use TD3, [16]). See Fig. 1 for an overview of our representation learning method. Importantly, our method *(i)* alleviates dimensional collapse, *(ii)* demonstrates excellent sample efficiency outperforming TCRL and TD7 on a wide range of different continuous control tasks, *(iii)* is simple to implement, and *(iv)* learns a task-agnostic representation that could be helpful in downstream tasks.

2 Method

In this section, we detail our method, named *implicitly Quantized Reinforcement Learning* (iQRL). iQRL is conceptually simple, it *(i)* learns a representation of the observation space and then, *(ii)* performs model-free RL (*e.g.*, TD3) on this representation. See Fig. 1 and Algorithm 1.

We consider Markov Decision Processes (MDPs, [18]) $\mathcal{M} = (\mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, with discount factor $\gamma \in [0, 1)$, where an agent receives an observation $\boldsymbol{o}_t \in \mathcal{O}$ at time step t, performs an action $\boldsymbol{a}_t \in \mathcal{A}$, and then obtains the next observation $\boldsymbol{o}_{t+1} \sim \mathcal{P}(\cdot \mid \boldsymbol{o}_t, \boldsymbol{a}_t)$ and a reward $r_t = \mathcal{R}(\boldsymbol{o}_t, \boldsymbol{a}_t)$.

Method components iQRL has four main components which we wish to learn:

$$\begin{aligned}
&\text{Encoder:} && \boldsymbol{z}_t = f(e_\theta(\boldsymbol{o}_t)) && (1)\\
&\text{Dynamics:} && \hat{\boldsymbol{z}}_{t+1} = f(\boldsymbol{z}_t + d_\phi(\boldsymbol{z}_t, \boldsymbol{a}_t)) && (2)\\
&\text{Value:} && \boldsymbol{q}_t = \mathbf{q}_\psi(\boldsymbol{z}_t, \boldsymbol{a}_t) && (3)\\
&\text{Policy:} && \boldsymbol{a}_t \sim \pi_\eta(\boldsymbol{z}_t) && (4)
\end{aligned}$$

The encoder e_θ and latent-space dynamics model d_ϕ are responsible for representation learning. $f(\cdot)$ denotes our quantization scheme, which implicitly quantizes our latent representation (more details to follow). The encoder $f \circ e_\theta(\cdot)$ maps observations $\boldsymbol{o}_t$ to latent states $\boldsymbol{z}_t$ and is responsible for learning a representation which can aid RL. The latent-space dynamics model $d_\phi(\cdot)$ predicts the next latent states $\hat{\boldsymbol{z}}_{t+1}$ given a latent state $\boldsymbol{z}_t$ and an action $\boldsymbol{a}_t$. Once we have the representation learned by our encoder, we map all observations to the latent space and perform model-free RL in this latent space. Throughout this paper, we use Twin Delayed Deep Deterministic Policy Gradient (TD3, [16]) as the base algorithm. It consists of two action-value functions $\{q_{\psi_1}, q_{\psi_2}\}$, known as critics, and a deterministic actor π_η. However, we follow Yarats et al. [5], Zhao et al. [8] and augment the loss with n-step returns. The only difference to TD3 is that we map observations $\boldsymbol{o}_t$ through the encoder $\boldsymbol{z}_t = f(e_\theta(\boldsymbol{o}_t))$ and learn the actor/critic in the quantized latent space.

Representation learning Our representation learning uses the latent-state consistency loss,

$$\mathcal{L}_{\text{rep}}(\theta, \phi; \tau) = \sum_{h=0}^{H-1} \gamma_{\text{rep}}^h \left(\frac{f(\hat{\boldsymbol{z}}_{t+h} + d_\phi(\hat{\boldsymbol{z}}_{t+h}, \boldsymbol{a}_{t+h}))}{\|f(\hat{\boldsymbol{z}}_{t+h} + d_\phi(\hat{\boldsymbol{z}}_{t+h}, \boldsymbol{a}_{t+h}))\|_2} \right)^\top \left(\frac{f(e_{\bar{\theta}}(\boldsymbol{o}_{t+h+1}))}{\|f(e_{\bar{\theta}}(\boldsymbol{o}_{t+h+1}))\|_2} \right), \quad (5)$$

which minimizes the cosine similarity between the next state predicted by the dynamics model $\hat{\boldsymbol{z}}_{t+1} = f(\hat{\boldsymbol{z}}_t + d_\phi(\hat{\boldsymbol{z}}_t, \boldsymbol{a}_t))$ and the next state predicted by the momentum encoder $\bar{\boldsymbol{z}}_{t+1} = f(e_{\bar{\theta}}(\boldsymbol{o}_{t+1}))$. The latent states are obtained with multi-step predictions in the latent space $\hat{\boldsymbol{z}}_{t+1} = f(\hat{\boldsymbol{z}}_t + d_\phi(\hat{\boldsymbol{z}}_t, \boldsymbol{a}_t))$. The initial mapping to the latent space $\hat{\boldsymbol{z}}_0 = f(e_\theta(\boldsymbol{o}_0))$ uses the online encoder which is being trained jointly with the dynamics model $d_\phi(\hat{\boldsymbol{z}}_t, \boldsymbol{a}_t)$. The target $e_{\bar{\theta}}(\boldsymbol{o}_{t+1})$ is calculated with the momentum encoder which uses an exponential moving average (EMA) of the encoder's weights $\bar{\theta} \leftarrow (1 - \tau)\bar{\theta} + \tau\theta$. The target network update rate is denoted τ. Note that we do not use reward or value prediction for learning our representation and as a result, our representation is task-agnostic.

Quantization Motivated by preventing dimensional collapse we quantize our latent space following the approach from Finite Scalar Quantization (FSQ, [17]). Their important observation is that carefully bounding each dimension gives rise to an *implicit* codebook $\mathcal{C}$ of a chosen size $|\mathcal{C}|$. Having requested a d-dimensional latent space, iQRL configures the encoder to output c channels per dimension such that the representation from the encoder $\boldsymbol{x} = e_\theta(\boldsymbol{o}) \in \mathbb{R}^{d \times c}$ and the dynamics model $\hat{\boldsymbol{x}} = \boldsymbol{z} + d_\phi(\boldsymbol{z}, \boldsymbol{a}) \in \mathbb{R}^{d \times c}$ are in $\mathbb{R}^{d \times c}$. To quantize $\boldsymbol{x}$ (and $\hat{\boldsymbol{x}}$) into a finite set of codewords, we first apply a bounding function $f(\cdot)$ and then we round to integers. Let us consider a single dimension j of the encoder's output $\boldsymbol{v} = [\boldsymbol{x}]_{j,:} \in \mathbb{R}^c$ which consists of c-channels, and demonstrate how it is quantized. We follow FSQ and choose $f(\cdot)$ such that each entry in $\tilde{\boldsymbol{v}} = \text{round}(f(\boldsymbol{v}))$ takes one of L_i unique values, $f : \boldsymbol{v} \to \lfloor L_i/2 \rfloor \tanh(\boldsymbol{v})$, where L_i is a hyperparameter for channel i, specified as FSQ levels $\mathcal{L} = \{L_1, \ldots, L_c\}$. This gives an entry in our codebook $\tilde{\boldsymbol{v}} \in \mathcal{C}$, where the *implied* codebook is given by the product of these per-channel codebook sets. The vectors in $\mathcal{C}$ can be enumerated giving a bijection from any $\tilde{\boldsymbol{v}}$ to an integer in $\{1, 2, \ldots, L^c\}$. As an example, in some of our experiments, we used $d = 512$ latent dimensions each with $c = 2$ channels consisting of 8 levels, *i.e.* we used FSQ levels $\mathcal{L} = \{L_1 = 8, L_2 = 8\}$. This corresponds to a codebook of size $|\mathcal{C}| = \prod_{i=1}^{c} L_i = 8 \times 8 = 64 = 2^6$ for each dimension.

Note that this quantization requires a round operation. As such, to propagate gradients through the round operation we use straight-through gradient estimation (STE). This is easily accomplished in deep learning libraries using stop gradient sg as $\text{round\_ste}(x) : x \to x + \text{sg}(\text{round}(x) - x)$. FSQ has the following hyperparameters: we must specify the number of channels c and the number of levels per channel $\mathcal{L} = \{L_1, \ldots, L_c\}$. Table 1 shows the recommended number of channels and number of levels per channel to obtain codebooks of different sizes [17]. In practice, we found codebooks of size $|\mathcal{C}| = 2^6$ sufficient for all environments in the DeepMind Control suite. However, for more complex environments we hypothesize that larger codebooks will be required.

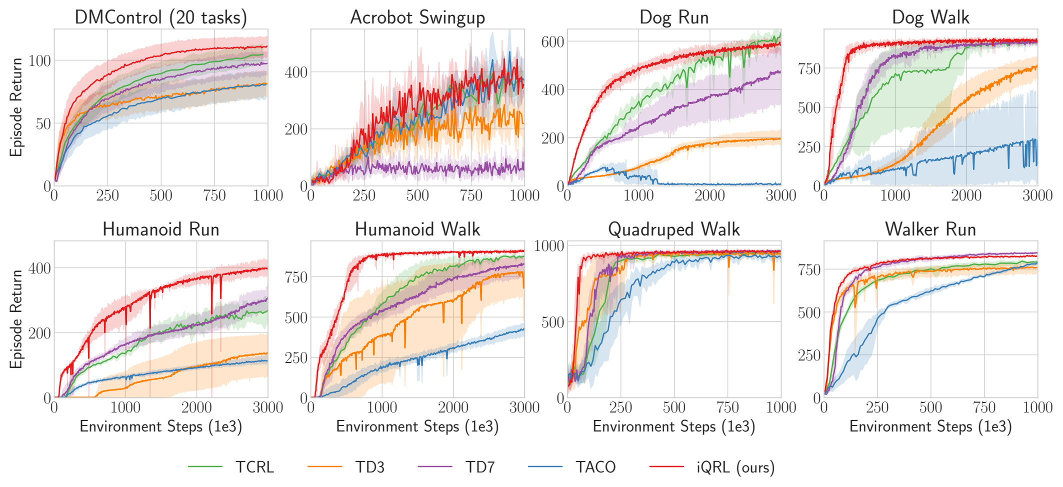

Figure 2: **DeepMind Control Suite results.** IQRL (red) is significantly more sample efficient than other model-free baselines TCRL (green), TD7 (purple), TACO (blue) and TD3 (orange). IQRL performs particularly well in the high-dimensional locomotion tasks and outperforms TCRL, which is the most similar baseline. Results are for 20 DMC tasks with UTD=1. We plot the mean (solid line) and the 95% confidence intervals (shaded) across 5 random seeds, where each seed averages over 10 evaluation episodes. See Fig. 4 for results in other DMC tasks.

3 Experiments

We evaluate IQRL in a variety of tasks from the DeepMind Control (DMC) Suite [19]. We compare to TD3 [16], and the representation learning-based RL methods TCRL [8], TD7 [7], and TACO [6].

IQRL is simple, fast, and performant In Fig. 2, we evaluate sample efficiency by plotting the average performance of the algorithms across 20 DMC tasks as a function of environment steps. On average, IQRL outperforms the baselines and shows significant advantages in many environments. Furthermore, TD3 is noncompetitive with IQRL, highlighting the importance of representation learning in state-based RL. For complete results on all 20 tasks, see Fig. 4.

IQRL does not suffer from rank collapse We examine the behaviour of adding quantization to our MLP encoder during training. Following Ni et al. [20], we estimate the rank of the linear operator associated with the MLP encoder by calculating the matrix rank[1] of the latent states for a batch of inputs. We ensure full rank at the start of training by orthogonally initializing the MLP encoders. Fig. 3 shows the orthogonality-preserving effect of our quantization scheme as the matrix rank stays close to the maximum. Without quantization, a dimensional collapse occurs, which can have significant harmful effects as the representational power of the latent state diminishes [11]. In three of the four environments, removing quantization has a deteriorating impact on sample efficiency, and in Dog Run, the algorithm completely fails to solve the task without quantization.

For more details about the baselines, the DMC tasks, and further experiments, see Appendices C, D and E, respectively.

4 Conclusion

We have presented IQRL, a technique for learning representations using only a self-supervised temporal consistency loss, which demonstrates strong performance in continuous control tasks, including the complex DMC Humanoid and Dog tasks. Our quantization of the latent space empirically preserves the representation's matrix rank, indicating that it alleviates dimensional collapse. Our experiments further demonstrate that IQRL is extremely sample efficient whilst being fast to train, which we believe is a strong selling point. Importantly, our method is *(i)* straightforward, *(ii)* compatible with any model-free RL algorithm, and *(iii)* learns a task-agnostic representation.

[1]Rank of an $m \times n$ matrix $\boldsymbol{A}$ is the dimension of the image of the mapping $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$, with $g(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$

References

[1] 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.

[2] R.S. Sutton and A.G. Barto. *Reinforcement Learning, Second Edition: An Introduction*. Adaptive Computation and Machine Learning Series. MIT Press, 2018.

[3] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5639–5650. PMLR, November 2020.

[4] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. In *International Conference on Learning Representations*, October 2020.

[5] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning. In *International Conference on Learning Representations*, October 2021.

[6] 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. In *Advances in Neural Information Processing Systems*, volume 36, pages 48203–48225, December 2023.

[7] Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang (Shane) Gu, Doina Precup, and David Meger. For SALE: State-Action Representation Learning for Deep Reinforcement Learning. *Advances in Neural Information Processing Systems*, 36:61573–61624, December 2023.

[8] Yi Zhao, Wenshuai Zhao, Rinu Boney, Juho Kannala, and Joni Pajarinen. Simplified Temporal Consistency Reinforcement Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 42227–42246. PMLR, July 2023.

[9] 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*, October 2020.

[10] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised State Representation Learning in Atari. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[11] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding Dimensional Collapse in Contrastive Self-supervised Learning. In *International Conference on Learning Representations*, October 2021.

[12] 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*, October 2020.

[13] Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal Difference Learning for Model Predictive Control. In *Proceedings of the 39th International Conference on Machine Learning*, pages 8387–8406. PMLR, June 2022.

[14] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2170–2179. PMLR, May 2019.

[15] 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*, volume 35, pages 20189–20204, December 2022.

[16] Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1587–1596. PMLR, July 2018.

[17] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite Scalar Quantization: VQ-VAE Made Simple, September 2023.

[18] Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[19] 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.

[20] 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*, October 2023.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[23] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *The Twelfth International Conference on Learning Representations*, October 2023.

[24] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016.

[25] Zixin Wen and Yuanzhi Li. The Mechanism of Prediction Head in Non-contrastive Self-supervised Learning. *Advances in Neural Information Processing Systems*, 35:24794–24809, December 2022.

Appendices

A Further Method Details

In this section, we provide further details of our method.

FSQ levels Table 1 shows how to select the FSQ levels $\mathcal{L}$ hyperparameter in order to approximate codebooks of different sizes.

Table 1: FSQ levels $\mathcal{L}$ to approximate different codebook sizes $|\mathcal{C}|$

TARGET SIZE $\vert\mathcal{C}\vert$	2^4	2^6	2^8	2^9	2^{10}
PROPOSED $\mathcal{L}$	$\{5,3\}$	$\{8,8\}$	$\{8,6,5\}$	$\{8,8,8\}$	$\{8,5,5,5\}$

Model-free reinforcement learning We learn the policy (actor) and action-value function (critic) using TD3 [16]. However, we follow Yarats et al. [5], Zhao et al. [8] and augment the loss with n-step returns. The only difference to TD3 is that instead of using the original observations $\boldsymbol{o}_t$, we map them through the online encoder $\boldsymbol{z}_t = f(e_\theta(\boldsymbol{o}_t))$ and learn the actor/critic in the quantized latent space $\boldsymbol{z}_t$. The critic is then updated by minimizing the following objective:

$$\mathcal{L}_q(\psi;\tau) = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{k=1}^{2}(q_{\psi_k}(f(e_\theta(\boldsymbol{o}_t)),\boldsymbol{a}_t) - y)^2\right], \quad \forall k \in 1,2 \tag{6}$$

$$y = \sum_{n=0}^{N-1} r_{t+n} + \gamma^n \min_{k\in\{1,2\}} q_{\bar{\psi}_k}(e_\theta(\boldsymbol{o}_{t+n+1}),\boldsymbol{a}_{t+n+1}), \quad \text{with } \boldsymbol{a}_{t+n} = \pi_{\bar{\eta}}(\boldsymbol{z}_{t+n}) + \epsilon_{t+n}$$

where we use policy smoothing by adding clipped Gaussian noise $\epsilon_{t+n} \sim \text{clip}\left(\mathcal{N}(0,\sigma^2), -c, c\right)$ to the action $\boldsymbol{a}_{t+n} = \pi_{\bar{\eta}}(\boldsymbol{z}_{t+n}) + \epsilon_{t+n}$. Note that we use the online encoder to get the latent states in both the prediction and the target. We then use the target action-value functions $\mathbf{q}_{\bar{\psi}}$ and the target policy $\pi_{\bar{\eta}}$ to calculate the TD target. Following TD3, we learn the actor's parameters by minimizing

$$\mathcal{L}_\pi(\eta;\tau) = -\mathbb{E}_{\boldsymbol{o}_t\sim\mathcal{D}}\left[\min_{k\in\{1,2\}} q_{\psi_k}(\underbrace{f(e_\theta(\boldsymbol{o}_t))}_{\boldsymbol{z}_t}, \pi_\eta(f(e_\theta(\boldsymbol{o}_t))))\right]. \tag{7}$$

That is, we maximize the Q-value using the clipped double Q-learning trick to combat overestimation in Q-learning. Note that we do not use the momentum encoder in the actor/critic objectives. In our experiments, using the momentum encoder resulted in worse performance. Whilst our method shares similarities with TCRL [8], it is important to note that our transition model does not predict the reward. Instead, iQRL leverages quantization to help alleviate dimensional collapse, and, as a result, learns a task-agnostic representation.

Algorithm 1 iQRL

```
Input: Encoder $e_\theta$, dynamics $d_\phi$, critics $\{q_{\psi_1}, q_{\psi_2}\}$, policy $\pi_\eta$, learning rate $\alpha$, target network update rate $\tau$
for $i$ to $N_{\text{episodes}}$ do
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{o}_t, \boldsymbol{a}_t, \boldsymbol{o}_{t+1}, r_{t+1}\}_{t=0}^{T}$        ▷ Collect data in environment
    for $i = 1$ to $T$ do
        $[\theta,\phi] \leftarrow [\theta,\phi] + \alpha\nabla\left(\mathcal{L}_{\text{rep}}(\theta,\phi;\mathcal{D})\right)$        ▷ Update representation, Eq. (5)
        $\psi \leftarrow \psi + \alpha\nabla\left(\mathcal{L}_q(\psi;\mathcal{D})\right)$        ▷ Update critic, Eq. (6)
        if $i$ % 2 == 0 then
            $\eta \leftarrow \eta + \alpha\nabla\left(\mathcal{L}_\pi(\eta;\mathcal{D})\right)$        ▷ Update actor less frequently than critic, Eq. (7)
        end if
        $[\bar{\theta},\bar{\psi},\bar{\eta}] \leftarrow (1-\tau)[\bar{\theta},\bar{\psi},\bar{\eta}] + \tau[\theta,\psi,\eta]$        ▷ Update target networks
    end for
end for
```

B Implementation Details

Architecture We implemented iQRL with PyTorch [21] and used the AdamW optimizer [22] for training the models. All components (encoder, dynamics, actor and critic) are implemented as MLPs. Following Hansen et al. [23] we let all intermediate layers be linear layers followed by LayerNorm [24]. Using LayerNorm is what led to our base TD3 implementation performing so well. We use Mish activation functions throughout. Below we summarize the iQRL architecture for our base model.

```
iQRL(
  (fsq): FSQ(
    (project_in): Identity()
    (project_out): Identity()
  )
  (encoder): ModuleDict(
    (state): Sequential(
      (0): NormedLinear(in_features=O, out_features=256, act=Mish)
      (1): Linear(in_features=256, out_features=512)
    )
  )
  (encoder_tar): ModuleDict(
    (state): Sequential(
      (0): NormedLinear(in_features=O, out_features=256, act=Mish)
      (1): Linear(in_features=256, out_features=512)
    )
  )
  (dynamics): Sequential(
    (0): NormedLinear(in_features=512+A, out_features=512, act=Mish)
    (1): NormedLinear(in_features=512, out_features=512, act=Mish)
    (2): Linear(in_features=512, out_features=512)
  )
  (pi): Actor(
    (_pi): Sequential(
      (0): NormedLinear(in_features=512, out_features=512, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, act=Mish)
      (2): Linear(in_features=512, out_features=A)
    )
  )
  (pi_tar): Actor(
    (_pi): Sequential(
      (0): NormedLinear(in_features=512, out_features=512, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, act=Mish)
      (2): Linear(in_features=512, out_features=A)
    )
  )
  (critic): Critic(
    (_q1): Sequential(
      (0): NormedLinear(in_features=512+A, out_features=512, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, act=Mish)
      (2): Linear(in_features=512, out_features=1)
    )
    (_q2): Sequential(
      (0): NormedLinear(in_features=512+A, out_features=512, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, act=Mish)
      (2): Linear(in_features=512, out_features=1)
    )
  )
  (critic_tar): Critic(
    (_q1): Sequential(
      (0): NormedLinear(in_features=512+A, out_features=512, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, act=Mish)
      (2): Linear(in_features=512, out_features=1, bias=True)
    )
    (_q2): Sequential(
      (0): NormedLinear(in_features=512+A, out_features=512, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, act=Mish)
      (2): Linear(in_features=512, out_features=1)
    )
  )
)
```

where O is the dimensionality of the observation space and A is the dimensionality of the action spaces.

Hyperparameters Table 2 lists all of the hyperparameters for training IQRL which were used for the main experiments and the ablations.

Table 2: **IQRL hyperparameters.** We kept most hyperparameters fixed across all tasks.

HYPERPARAMETER	VALUE	DESCRIPTION
TRAINING		
ACTION REPEAT	2	
MAX EPISODE LENGTH	500	ACTION REPEAT MAKES THIS 1000
EVAL. EVERY EPISODES	10	
NUM. EVAL EPISODES	10	
RANDOM EPISODES	10	NUM. RANDOM EPISODES AT START
TD3		
ACTOR UPDATE FREQ.	2	UPDATE ACTOR LESS THAN CRITIC
BATCH SIZE	256	
BUFFER SIZE	10^6	
DISCOUNT FACTOR γ	0.99	
EXPLORATION NOISE	$\text{Linear}(1.0, 0.1, 50)$ (EASY)	
	$\text{Linear}(1.0, 0.1, 150)$ (MEDIUM)	
	$\text{Linear}(1.0, 0.1, 500)$ (HARD)	
LEARNING RATE	3×10^{-4}	
MLP DIMS	$[512, 512]$	FOR ACTOR/CRITIC/DYNAMICS
MOMENTUM COEF. (τ)	0.005	
NOISE CLIP	0.3	
N-STEP TD	1 OR 3	
POLICY NOISE	0.2	
UPDATE-TO-DATA (UTD) RATIO	1	
ENCODER		
DISCOUNT FACTOR γ_{REP}	0.9	
ENCODER LEARNING RATE	10^{-4}	
ENCODER MLP DIMS	$[256]$	
ENCODER MOMENTUM COEF. (τ)	0.005	
FSQ LEVELS	$[8, 8]$	
HORIZON (H)	5	FOR REPRESENTATION LEARNING
LATENT DIMENSION (d)	512	
	1024 (HUMANOID/DOG)	

Statistical significance We used five seeds for the main figures, at least three seeds for all ablations, and plotted the 95 % confidence intervals as the shaded area, which corresponds to approximately two standard errors of the mean.

Hardware We used Nvidia A100s and AMD Instinct MI250X GPUs to run our experiments. All our experiments have been run on a single GPU with a single-digit number of CPU workers.

Open-source code For full details of the implementation, model architectures, and training, please check the code, which is available in the submitted supplementary material and will be made public upon acceptance to guarantee seamless reproducibility.

–appendices continue on next page–

C Baselines

In this section, we provide further details of the baselines we compare against. In particular, we provide details of how we modified the original codebases and tuned the hyperparameters in an effort to offer a fair comparison.

- **Temporal Consistency Reinforcement Learning (TCRL, [8])** is a reinforcement learning algorithm consisting of four components, an encoder and transition, policy and value functions, similarly to iQRL. TCRL uses a temporal consistency loss similar to model-based reinforcement learning to learn a representation used for model-free policy and value function training. The most crucial difference between TCRL and iQRL is that we replace the reward prediction head in the transition function with the FSQ-based normalization scheme. We used the official TCRL implementation on GitHub to run the TCRL experiments in our paper. For the DeepMind Control Suite (DMC) tasks, we used the tuned hyperparameters from the original paper. We used the official PyTorch implementation[2].
- **Temporal Action-driven Contrastive Learning (TACO, [6])** is a temporal contrastive learning framework that learns a latent representation of states and actions with a contrastive loss that optimizes the mutual information between the representations of current states and the following action sequences, and those of the corresponding future states. TACO was primarily designed for vision-based tasks, whereas our benchmarks are state-based. We adapted TACO to the state-based setting by increasing the learning rate and update-to-data ratios to match those of iQRL. We also replaced their CNN-based encoder with the MLP-based encoder of iQRL. Then, we performed a grid search over feature dimensions of 50 and 128, hidden dimensions of 256, 512, and 1024, and frame stacking and no frame stacking. We found the combination of a feature dimension of 50 and a hidden dimension of 1024 without frame stacking to perform the best.
- **Twin Delayed DDPG (TD3, [16])** is a model-free RL algorithm for continuous control, extending deep deterministic policy gradient (DDPG) to deal with value overestimation bias. Compared to DDPG, this algorithm uses two critics and takes the minimum over the two for training, adds clipped noise to the actions selected for bootstrapping (policy smoothing), and updates the actor less frequently compared to the critics. iQRL is based on TD3 and we simply replace the observations with their corresponding latent representation by mapping them through our encoder. This baseline uses our TD3 implementation which obtains very strong results. Comparing to this baseline allows us to investigate the impact of representation learning on sample-efficiency.
- **TD7 [7]** is a model-free reinforcement learning algorithm for continuous control that builds on TD3. TD7 builds on a representation learning method, state-action learned embeddings (SALE). The embeddings are learned using a temporal consistency term in the latent state. Other improvements that TD7 has over TD3 are prioritized experience replay and checkpointing. TD7 was initially evaluated on MuJoCo. To adapt it for DeepMind Control Suite, we added action repeats, essential for good performance on DMC. Then, we compared the original hyperparameters of TD7 to those of iQRL and found iQRL to perform the best, so we used those for the final evaluation. In particular, the exploration noise decay of iQRL was crucial for high performance in the DMC environments, and without it, TD7 struggled. Note that both TD7 and iQRL use TD3 as the underlying algorithm, allowing us to reliably compare the impact of SALE and our FSQ-based representations. We used the official PyTorch implementation of TD7[3].

–appendices continue on next page–

[2]`https://github.com/zhaoyi11/tcrl`

[3]`https://github.com/sfujim/TD7`

D Tasks

We evaluate our method in 20 tasks from the DeepMind Control suite [19]. Table 3 provides details of the environments we used, including the dimensionality of the observation and action spaces.

Table 3: **DMControl.** We consider a total of 20 continuous control tasks from the DeepMind Control suite.

Task	Observation dim	Action dim	Sparse?
Acrobot Swingup	6	1	N
Cheetah Run	17	6	N
Cup Catch	8	2	Y
Dog Run	223	38	N
Dog Trot	223	38	N
Dog Stand	223	38	N
Dog Walk	223	38	N
Fish Swim	24	5	N
Hopper Hop	15	4	N
Hopper Stand	15	4	N
Humanoid Run	67	24	N
Humanoid Stand	67	24	N
Humanoid Walk	67	24	N
Quadruped Run	78	12	N
Quadruped Walk	78	12	N
Reacher Easy	6	2	Y
Reacher Hard	6	2	Y
Walker Run	24	6	N
Walker Stand	24	6	N
Walker Walk	24	6	N

–appendices continue on next page–

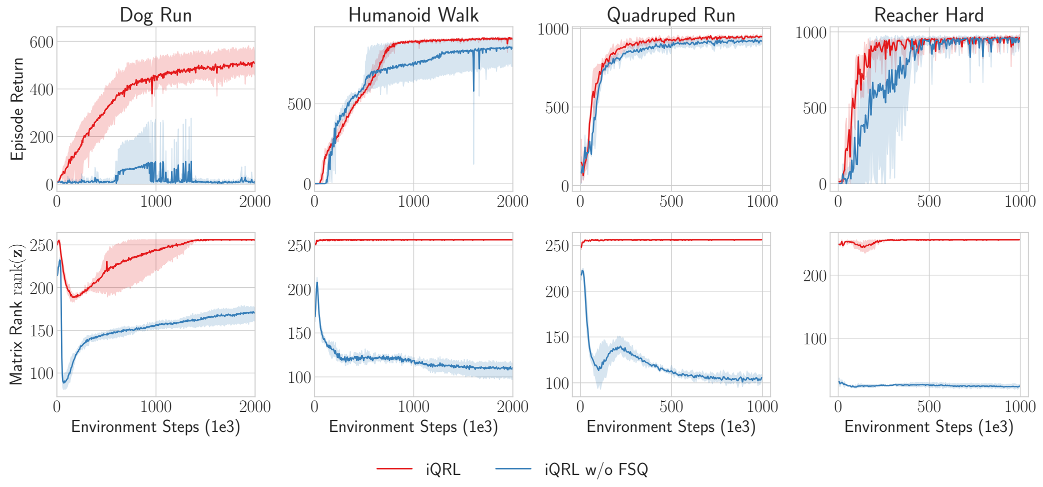

Figure 3: **Ablation of quantization.** We show how our quantization scheme prevents dimensional collapse. In all tasks, our FSQ scheme prevents dimensional collapse (red) as the rank of the representation remains high. In contrast, when our quantization is not used (blue) the representation undergoes dimensional collapse, indicated by the rank reducing. In the Dog Run task, this results in the agent not learning to solve the task.

E Experiments

In this section, we provide further insights from our experiments.

High-dimensional control Many DMC tasks are high-dimensional, for instance, the observation space of the Dog tasks is $\mathcal{O} \in \mathbb{R}^{223}$ and the action space is $\mathcal{A} \in \mathbb{R}^{38}$. Figs. 2 and 4 show that IQRL excels in the high dimensional Dog and Humanoid environments when compared to the baselines. We hypothesize that our discretized representations are particularly beneficial for simplifying learning transition dynamics in high-dimensional spaces, making IQRL highly sample efficient.

Further DMC Results Fig. 4 compares IQRL to the baselines in the 20 DMC tasks. IQRL's representation learning significantly improves sample efficiency when compared to TD3. Note that IQRL uses the same TD3 implementation with the same hyperparameters, so the only difference is our representation learning. IQRL also outperforms TCRL in terms of sample efficiency, even without the reward prediction head. Our experiments indicate that this improvement is due to our quantization and the inclusion of LayerNorm in our encoder. We compare IQRL to TACO (which uses a contrastive loss) and observe that IQRL outperforms TACO in most environments. TACO seems to particularly struggle in the Dog tasks. Finally, IQRL outperforms TD7, a state-of-the-art representation learning method for state-based RL.

–appendices continue on next page–

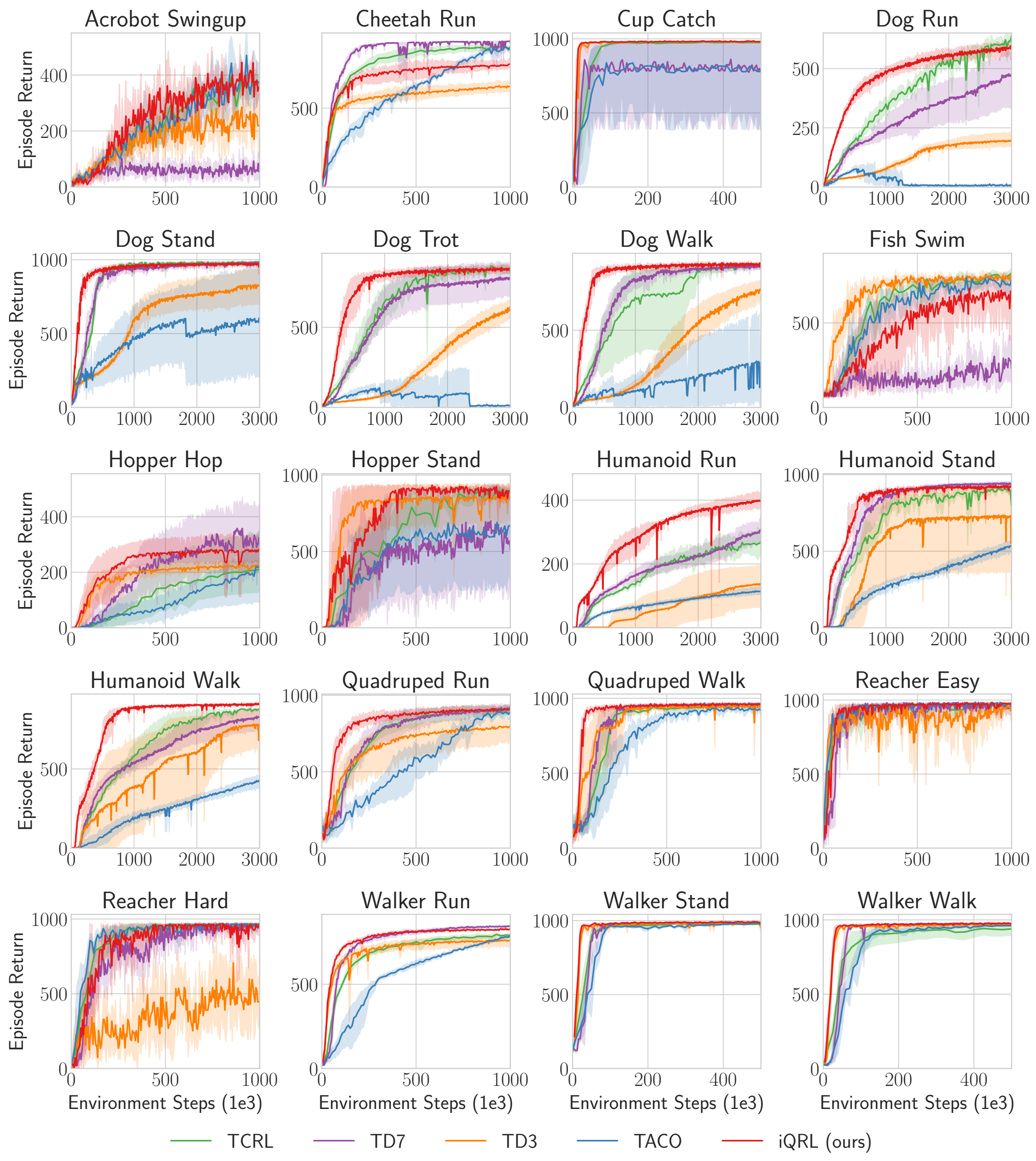

Figure 4: **DeepMind Control results.** IQRL performs well across a variety of DMC tasks. We plot the mean (solid line) and the 95% confidence intervals (shaded) across 5 random seeds, where each seed averages over 10 evaluation episodes.

–appendices continue on next page–

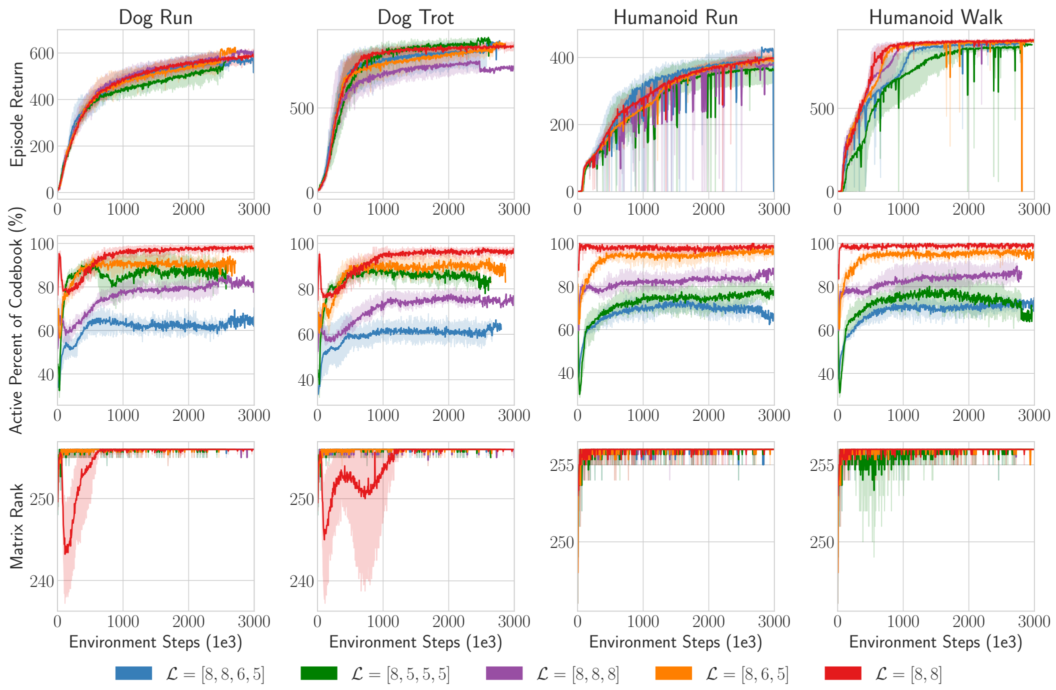

Figure 5: **Codebook size ablation.** We compare how the codebook size affects the performance of IQRL (top), the percentage of the codebook that is active during training (middle), and how the different codebook sizes affect the encoder's ability to preserve the rank of the representation (bottom). In general, smaller codebooks become fully active faster than larger codebooks, and the rank of the representation is maintained for all codebook sizes. We plot the mean and the 95% confidence intervals (shaded) across 3 random seeds for all environments.

Codebook size $|\mathcal{C}|$ We evaluate how the size of the codebook $|\mathcal{C}|$ influences training. We indirectly configure different codebook sizes via the FSQ levels $\mathcal{L} = \{L_1, \dots, L_c\}$ hyperparameter. This is because the codebook size is given by $|\mathcal{C}| = \prod_{i=1}^{c} L_i$. The top row of Fig. 5 compares the training curves for different codebook sizes. The algorithm's performance is not particularly sensitive to the codebook size. A codebook that is too large can result in slower learning. The best codebook size varies between environments. The most difficult environment, Humanoid Run, benefits from the largest codebook.

Given that a codebook has a particular size, we can gain insights into how quickly IQRL's encoder starts to activate all of the codebook. The connection between the codebook size and the activeness of the codebook is intuitive: the middle row of Fig. 5 shows that the smaller the codebook, the larger the active proportion.

In the bottom row of Fig. 5, we evaluate how different codebook sizes affect the encoder's ability to preserve the rank of the representation. We see that the rank of the representation is maintained no matter the codebook size.

–appendices continue on next page–

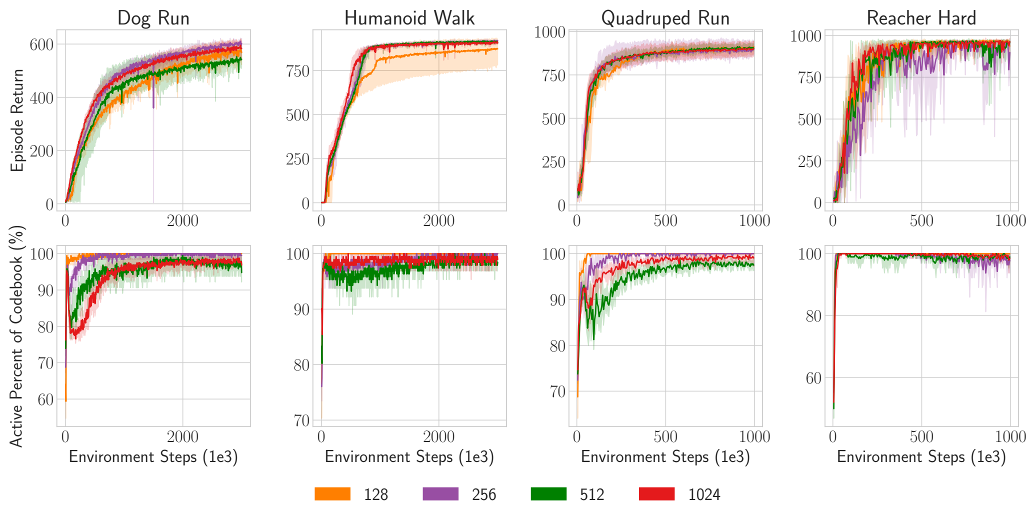

Figure 6: **Latent dimension d ablation.** We compare how the latent dimension d affects the performance of iQRL (top) and the percentage of the codebook that is active during training (bottom). In general, our algorithm is robust to the latent dimension of the representation, although in more difficult environments, such as Humanoid Walk, a d too small can harm the agent's performance.

Latent dimension d Next, we investigate how the latent dimension d affects the behavior and performance of iQRL in four different environments. The latent dimension d corresponds to the dimension of the representation corresponding to each FSQ level before and after quantization is applied. In the top row of Fig. 6, we see that the performance of our algorithm is robust to the latent dimension d, although a latent dimension too small can result in inferior performance, especially in the more difficult environments. The bottom row of Fig. 6 demonstrates that iQRL learns to use the complete codebook irrespective of the latent dimension. However, a larger d can also correspond to the codebook becoming fully active slightly slower.

–appendices continue on next page–

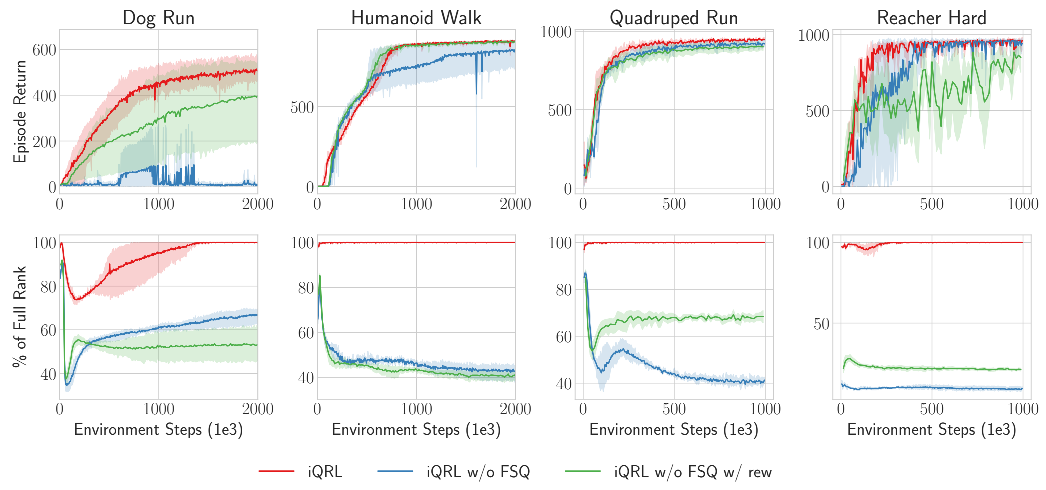

Figure 7: **Adding a reward head is not enough to prevent loss of rank.** We show how removing quantization leads to dimensional collapse measured in terms of the rank of the representation and how adding a reward prediction head to IQRL without quantization is insufficient to counteract this and maintain full rank.

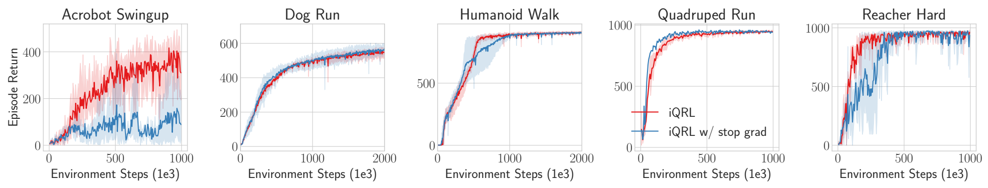

Figure 8: **Replacing EMA encoder with stop gradient.** We show that removing IQRL's EMA encoder and replacing it with only stop gradient hurts performance in DMC tasks. This is particularly apparent in the Acrobot Swingup task.

Reward head for representation learning In Fig. 7, we show how incorporating a reward prediction objective into IQRL is insufficient for maintaining the rank of the representation when our FSQ-based quantization scheme is not applied. There is a clear loss of rank in all four environments without the quantization; that is, the agent suffers from dimensional collapse. This results in performance inferior to IQRL in two environments, Dog Run and Reacher Hard.

Stop gradient [20] proved that using stop gradients should suffice for preventing representation collapse. However, their experiments suggested that using an EMA encoder improves performance over simply using stop gradients. In Fig. 8, we show how replacing IQRL's EMA encoder with a stop gradient operation decreases performance. Further, using stop gradient in the Acrobot Swingup tasks results in the agent struggling to solve the task.

–appendices continue on next page–

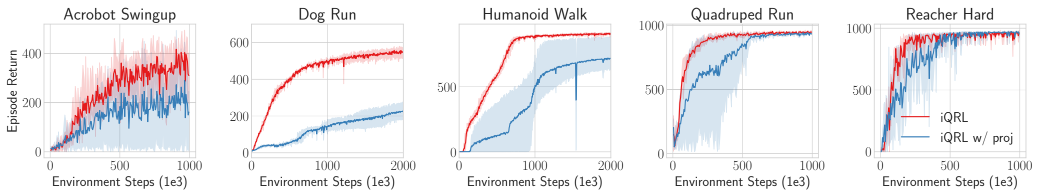

Figure 9: **Adding a projection head decreases sample efficiency.** We show that adding a projection head to IQRL, similar to what is done in SPR [9], decreases IQRL's sample efficiency. We plot the mean and the 95% confidence intervals (shaded) across 3 random seeds for all environments.

Projection head Wen and Li [25] and Schwarzer et al. [9] investigated the role of a learnable projection head in non-contrastive self-supervised learning and found that it helps RL algorithms learn more diversified and therefore, superior representations. Whilst IQRL shares similarities with SPR [9], in particular, a temporal consistency loss using cosine similarity, it differs in that it does not use a learnable projection head and quantizes the representation instead. In Fig. 9, we show the impact of adding a projection head to IQRL. It shows that the projection head decreases the sample efficiency of IQRL. Whilst projection heads are effective for learning representations from images, our results suggest that they have a significant negative impact on sample efficiency when learning representations of state-based observations, reaffirming that state-based RL has a different set of challenges to image-based RL and techniques designed to combat dimensional collapse are not always transferable between the settings.