REPRESENTATIONAL MANIFOLD HYPOTHESIS ENABLES SAMPLE-EFFICIENT ONLINE CONTRASTIVE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

General-purpose mechanisms from supervised deep learning, such as residual blocks, normalization, and depth, have recently been shown to transfer to online reinforcement learning (RL) and to improve its sample efficiency. One further mechanism, the manifold hypothesis, has been observed in RL but used only architecturally, through bottlenecks whose fixed rank must be chosen before training. We instead make the representation low-dimensional by learning. Building on neural regression collapse, we propose NRC2Reg, a regularizer that adds no parameters and pulls intermediate features toward the row space of the network's own readout, with a single coefficient and no rank to tune. On Contrastive RL, the features settle near task intrinsic dimension at effective rank ten of a 64-dimensional budget, a regime no swept hard bottleneck reaches, while NRC2Reg improves sample efficiency in all seven environment and depth settings and final performance in six, with the largest gains on maze tasks where the unregularized agent barely learns.

1 INTRODUCTION

Online Reinforcement Learning (RL) is now part of widely used systems, from the post-training of large language models (DeepSeek-AI et al., 2025) to high-dimensional robot control (Kim et al., 2026), yet its sample inefficiency still limits its use at scale. One principle that may help, but remains underdeveloped in online RL, is the manifold hypothesis: natural data, and the features that represent them, lie near a low-dimensional manifold (Bengio et al., 2013; Fefferman et al., 2016). Recent work has applied this hypothesis to online RL in two directions. In *parameter space*, it motivates confining the actor or critic to a low-dimensional subspace of its weights (Zhuang et al., 2026; Tenedini et al., 2026; Fraschini et al., 2026). In *representation space*, the states an agent visits under continuous control concentrate around a manifold whose dimension scales with the action dimension (Tiwari et al., 2025), which suggests that the features encoding them should be low-dimensional too. In this work we follow the second direction, since it acts directly on the representation the agent learns from.

The most direct way to make features low-dimensional is to constrain the architecture. Todorov & Sabatelli (2026) recently insert an orthogonal bottleneck that caps the rank of the features, and show that this costs no performance as long as the bottleneck is wider than the intrinsic rank of the value function. Yet a bottleneck is rigid. Its rank must be determined before training. The right rank value depends on the task. And it removes the same directions throughout training regardless of what the agent learns. Ideally, the features would become low-dimensional in an adaptive way, settling on the subspace that the task requires. This leads to our central question:

Can the features of an online RL agent be made low-dimensional by learning, without changing its architecture, and does this make the agent more sample-efficient?

To answer it, we draw on what supervised learning reveals about the terminal geometry of trained networks. In classification, Papyan et al. (2020) observed a phenomenon called Neural Collapse (NC): late in training, the last-layer features of each class concentrate around their class mean, the class means spread into a maximally separated configuration, and the classifier's weights align with

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

them, so that features and classifier share one simple, low-dimensional structure. Andriopoulos et al. (2024) discovered its counterpart in multivariate regression, Neural Regression Collapse (NRC). First, the last-layer features collapse onto a subspace whose dimension equals that of the regression target (NRC1). Second, and central to our study, this subspace coincides with the space spanned by the weights of the linear output projection (NRC2). Intuitively, a well-trained regression network stops carrying directions its readout ignores: every direction the features occupy is one the readout actually uses. NRC2 thus offers exactly the target a learned approach needs: a low-dimensional subspace that the network defines for itself.

We turn NRC2 from a property observed at convergence into a training objective, which we call NRC2Reg, and study it in scalable Contrastive RL (CRL; Eysenbach et al., 2022), a leading algorithm for goal-conditioned RL whose performance scales with network depth (Wang et al., 2025). CRL fits the regression picture naturally: its critic pulls the embedding of a state–action pair toward the embedding of the goal it later reaches. NRC2Reg adds a loss that pulls an intermediate feature of each network toward the subspace spanned by that network's own readout. This target has the properties a bottleneck lacks. Its dimension is the readout width, which CRL already fixes, so there is no rank to tune; within that space, the features are free to use as few directions as the task needs; and because the readout keeps learning, the target subspace adapts with it.

The answer to our question is positive. Where a bottleneck at best recovers the unregularized agent, NRC2Reg improves on it. Rather than discarding information to lower rank, it compresses the features through learning, keeping nearly all input information while raising their relevance to the goal. Across nine locomotion, navigation, and manipulation tasks, NRC2Reg improves sample efficiency and performance. On the challenging Humanoid mazes, it even makes a shallower network, which on its own lacks the capacity to solve the task, outperform a deeper unmodified network.

2 Preliminaries

2.1 Scalable Contrastive Reinforcement Learning

Goal-conditioned contrastive RL has recently proven effective for challenging online goal-conditioned tasks (Eysenbach et al., 2022; Liu et al., 2025; Bortkiewicz et al., 2025; Bastankhah et al., 2026), learning a goal-conditioned critic without explicit rewards. A state-action encoder $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^r$ and a goal encoder $\psi : \mathcal{G} \to \mathbb{R}^r$ define a similarity function $f_{\phi,\psi}(s, a, g) = -\|\phi(s, a) - \psi(g)\|_2$. Through Hindsight Experience Replay (Andrychowicz et al., 2017), each transition (s_i, a_i) in a batch of size M is paired with a goal $g_i = m(s_{t+\Delta})$ geometrically sampled from its future trajectory ($\Delta \geq 1$, with probability proportional to γ^{Δ}), where the goal map $m : \mathcal{S} \to \mathcal{G}$ extracts goal-relevant state features. The remaining batch goals serve as negatives, yielding the InfoNCE critic loss (van den Oord et al., 2018):

$$\mathcal{L}_{\text{critic}}(\phi, \psi) = -\frac{1}{M} \sum_{i=1}^{M} \log \frac{\exp f_{\phi,\psi}(s_i, a_i, g_i)}{\sum_{j=1}^{M} \exp f_{\phi,\psi}(s_i, a_i, g_j)}$$

A goal-conditioned policy $\pi_\theta(a \mid s, g)$ then optimizes

$$\max_{\theta} \; \mathbb{E}_{p(s)\, p_g(g)\, \pi_\theta(a'|s,g)} \big[f_{\phi,\psi}(s, a', g) - \alpha \log \pi_\theta(a' \mid s, g) \big],$$

with entropy regularization as in soft actor-critic (Haarnoja et al., 2018).

Recently, Wang et al. (2025) unlocked CRL scalability by parametrizing ϕ, ψ and π_θ as deep residual networks with layer normalization and Swish nonlinearities. In particular, let $u(h) = \mathrm{Swish}(\mathrm{LN}(Ah + c))$ denote a dense layer of width $d = 256$ followed by layer normalization and Swish, with its own parameters at each occurrence. A network maps its input x ((s, a) for ϕ, g for ψ, (s, g) for π_θ) to an output $z \in \mathbb{R}^n$ through an input layer, L residual blocks and a linear readout:

$$h_0 = u_{\text{in}}(x), \qquad h_\ell = h_{\ell-1} + F_\ell(h_{\ell-1}), \quad \ell = 1, \ldots, L, \qquad z = W h_L + b,$$

where each block $F_\ell = u_{\ell,4} \circ u_{\ell,3} \circ u_{\ell,2} \circ u_{\ell,1} : \mathbb{R}^d \to \mathbb{R}^d$ stacks four units, $W \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$. The output dimension is $n = r$ for the two encoders and $n = 2|\mathcal{A}|$ for the policy, whose readout stacks its mean and log-standard-deviation heads. The depth of a network is the number of dense layers in its blocks, that is, $4L$.

We call $h_\ell \in \mathbb{R}^d$ the *trunk feature* of ℓ-th block's post-skip output and write $H_\ell \in \mathbb{R}^{M\times d}$ for a batch of them. Unrolling Eq. (2.1) from any block X gives the *penultimate-layer feature*

$$h_L = h_X + \sum_{\ell=X+1}^{L} F_\ell(h_{\ell-1}),$$

so every trunk feature lives in the same d-dimensional coordinate system as the readout's input, and each block writes into it additively. This is what lets us compare an intermediate feature with the readout in Section 4.

2.2 Experimental Setup

We run all experiments on JaxGCRL (Bortkiewicz et al., 2025), a GPU-accelerated suite of state-based goal-reaching tasks on which CRL is the standard baseline achieving leading performance, and its scaling ability with network depth was also benchmarked (Wang et al., 2025). In this study, we evaluate nine environments, which evenly span control of a robotic arm, an ant agent, and a humanoid agent. Task types include *locomotion* (Humanoid), *navigation* (Ant and Humanoid mazes) and *manipulation* (arm tasks with a small action space). Within each control group, task difficulty also varies from easy to challenging. Detailed descriptions of the selected environments are included in Appendix.

All CRL experiments on JaxGCRL respect the training protocol and hyperparameters whenever possible, except for task-dependent *critical depth* by which the scaled networks can achieve significant performance. The learning policy is evaluated every epoch with 10^6 environment steps, on 128 episodes. We report *time-at-goal* as the main performance metric, which indicates the number of steps spent near the commanded goal. To ensure robust RL statistics, we define final performance as the interquartile mean (IQM; Agarwal et al., 2021) of time-at-goal over the last five epochs, pooled across all seeds. Sample efficiency is reported as the mean time-at-goal over all epochs, e.g., the normalized area under the learning curve (AUC). Every comparison is between two agents that differ only in the presence of low-dimensional feature intervention. We repeat each experiment with at least 3 seeds to account for variance.

3 Motivation: from an architectural bottleneck to a learned objective

Recently, Li & He (2026) give a sharp answer in flow-based image generation: when the network predicts a clean image lying on a low-dimensional manifold(Ansuini et al., 2019; Brown et al., 2023), the prediction task is naturally easier than predicting high-dimensional noise. They further show that embedding a raw 768-dim image patch through two linear layers with an intermediate bottleneck (anywhere from 32 to 512 dimensions) generally improves generation quality.

For online RL, parallel findings suggest similar structure. Tiwari et al. (2025, Theorem 1) show that for a continuous-control system under a wide two-layer actor-critic policy, the visited states concentrate around a manifold of dimension at most $2|\mathcal{A}| + 1$, independent of $|\mathcal{S}|$. Building on this, Todorov & Sabatelli (2026) apply an orthogonal bottleneck, a fixed orthonormal projection $B \in \mathbb{R}^{d\times k}$, $B^\top B = I_k$, to state embeddings, so subsequent layers receive only low-dimensional features. Under linear realizability, once k exceeds the intrinsic rank of the optimal value function in feature space, the projection preserves expressivity and leaves gradient dynamics unaffected; in practice, this

3.1 CRL with Orthogonal Bottleneck

We apply an orthogonal bottleneck $B \in \mathbb{R}^{d\times k}$ to the intermediate trunk features H_X of the CRL's X-th block, where B is drawn orthonormal at initialization and then frozen. Inserting it after block X replaces the trunk feature by its projection, $h_X \leftarrow BB^\top h_X$. We experiment on Humanoid at depth 32 ($L = 8$), where CRL learns reliably (Wang et al., 2025). Bottleneck is placed either on the penultimate feature ($X = 8$) or early in the stack ($X = 3$), with k from 4 to 64. Table 1 shows that the bottleneck never helps, wherever it sits. On the penultimate features, the result reveals a critical

threshold: at $k = 4$ the agent largely fails, at $k = 8$ it recovers most of the way, and from $k = 16$ on it plateaus, just below the unmodified network.

To understand why the bottleneck fails, we look at what it does to the representation it acts on. Figure 1 tracks the effective rank and the intrinsic dimension of the state-action encoder's features across layers, for the unmodified network and for bottlenecks under rank 8 and 64 after block $X = 3$. In the unmodified network, the rank of trunk features at block $X = 3$ expands early in training, to about 150 within the first few million steps, before settling near 70 (Figure 1a). The strict bottleneck $k = 8$ removes this expansion: the projected feature is bounded by 8 from the first epoch and only declines. At the end of training (Figure 1b), the unmodified rank grows monotonically across CRL blocks h_0 to h_8, while the bottlenecked networks drop to k at block 3 and restore the rank through the remaining blocks, eventually rejoining the unmodified network by the penultimate feature. Finally, the output of the state-action encoder $\phi(s, a)$ spans a linear space of dimension around 11.

Table 1: Applying a frozen orthogonal bottleneck of rank k after block X of CRL networks does not improve on the CRL baseline (first row) at any rank or place.

Block	Rank	IQM	AUC
–	–	639 ± 12	0.222
$X = 8$	$k = 4$	282 ± 28	0.075
$X = 8$	$k = 8$	596 ± 10	0.192
$X = 8$	$k = 16$	624 ± 14	0.208
$X = 8$	$k = 64$	625 ± 13	0.219
$X = 3$	$k = 8$	569 ± 11	0.217
$X = 3$	$k = 16$	494 ± 34	0.177
$X = 3$	$k = 64$	600 ± 8	0.222

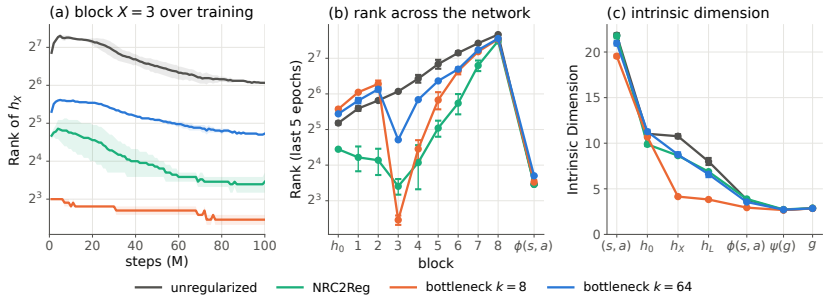

Figure 1: A frozen bottleneck cuts rank only where it sits: the state-action encoder with an orthogonal bottleneck after block $X = 3$ (humanoid, depth 32, three seeds, mean $\pm$ standard error). (a) Rank of the block-X feature over training (for the bottleneck runs, the projected feature $BB^\top h_X$). (b) Rank across the network at the end of training: the input layer's output h_0, the eight blocks, the output $\phi(s, a)$ and the goal encoder's penultimate feature (squares). (c) Intrinsic dimension (two-NN estimator) of the input, h_0, the block-X output h_X (before the projection), the penultimate feature h_L, the two outputs and the raw goal. Rank: number of singular directions holding 99% of the energy of a 4096-sample batch; end of training: mean of the last five epochs.

However, effective rank is a linear description of the feature geometry and says little about the finer, nonlinear structure of the feature manifold (Andriopoulos et al., 2026). We therefore also estimate the intrinsic dimension (ID) with the classic two-NN estimator (Facco et al., 2017), which was also adopted by Tiwari et al. (2025) for RL data. Here, the trend reveals what really happens on the feature manifold at the end of the training. Figure 1c shows that the unmodified CRL encoder compresses the intrinsic dimension via model depth from above 20 at the input to 8 at the penultimate features. In contrast, the $k = 8$ bottleneck pushes the penultimate feature to a much lower-dimensional manifold. This is over-compression, in which the features no longer carry the information needed to recover the target, and it is known to limit generalization in supervised regression (Zhang et al., 2024; Andriopoulos et al., 2026) and to hurt the online RL plasticity (Kumar et al., 2021; Lyle et al., 2022; 2023).

These observations lead to our hypothesis: a rigid bottleneck filters out information that is key to shaping the representation, which matters especially for an algorithm like CRL whose performance rests on subtle contrastive feature shaping (Bastankhah et al., 2026). To bring the representational manifold hypothesis into practice, the low-dimensional structure has to be learned by a loss rather than an architectural module. We next propose a learning-based approach, an objective motivated

by terminal feature structure in a supervised learning regime, that lets the network choose which subspace to occupy and then stay low-rank while preventing information loss.

4 METHOD

In supervised image classification, Papyan et al. (2020) observed that (1) during the terminal phase of training, penultimate features of each image class concentrate around their class mean; (2) the class means arrange themselves into a maximally separated symmetric configuration; and (3) the classifier's weight vectors align with them, and thus the representation and the readout settle into one shared low-dimensional structure. This *Neural Collapse* (NC) phenomenon, despite an optimization property, has since been used as a design principle, as fixing the classifier to this collapsed geometry at initialization removes the need to learn it and thus can lead to faster convergence (Yang et al., 2022). For regression, the counterpart has been derived recently. In this section, we abuse the math notation in order to introduce *Neural Regression Collapse* (NRC; Andriopoulos et al., 2024), which serves as the central motivation of our method.

In a standard regression problem, consider M training pairs (x_i, y_i) with targets $y_i \in \mathbb{R}^n$. A network $f(x; W, b, \theta) = W h_\theta(x) + b$ with feature extractor $h_\theta : \mathbb{R}^D \to \mathbb{R}^d$ and linear readout $W \in \mathbb{R}^{n \times d}$ is usually trained on a regularized squared loss. Denote the penultimate feature as $h_i = h(x_i)$. The normalized feature is $\hat{h}_i = (h_i - \bar{h})/\|h_i - \bar{h}\|_2$ with the feature mean $\bar{h} = M^{-1} \sum_i h_i$. Let $\mathrm{proj}(v \mid C)$ denote the orthogonal projection of v onto the column span of a matrix C. Andriopoulos et al. (2024) show that, at convergence, the features $\hat{h}_i$ concentrate in an n-dimensional subspace (NRC1) and that the same subspace aligns with the row space of the readout W (NRC2). Formally,

$$\mathrm{NRC2} = \frac{1}{M} \sum_{i=1}^{M} \left\| \hat{h}_i - \mathrm{proj}(\hat{h}_i \mid W^\top) \right\|_2^2, \tag{1}$$

which vanishes when every feature lies in $\mathrm{row}(W)$. NRC is derived based on a simplified theoretical formulation, Unconstrained Features Model (UFM). It treats the penultimate-layer features as free variables to be optimized jointly with the readout (Mixon et al., 2022), and has been verified in trained regression networks on control and imitation tasks (Andriopoulos et al., 2024; 2026). Subsequent work has extended the UFM analysis below the penultimate layer. For example, Fang et al. (2021) free the features of more shallow layers as well, and for regression, Rangamani & Unal (2026) empirically show that NRC1 and NRC2 also hold in intermediate layers, where each collapsed layer's features align with the row space of the output projection layer.

4.1 NRC2REG OBJECTIVE

CRL's state–action encoder fits the regression picture that NRC describes. The positive term of the InfoNCE loss (Eq. 2.1) minimizes $\|\phi(s_i, a_i) - \psi(g_i)\|_2$, regressing the encoder output onto the goal embedding $\psi(g_i)$, which is itself low-dimensional (intrinsic dimension about 3; Figure 1c). Because RL training is non-stationary, the target $\psi(g_i)$ co-evolves with the encoder; the network never reaches the terminal state that NRC describes. Nevertheless, the readout W is continually shaped by this regression signal, so its row space tracks the directions the task requires. We therefore turn NRC2 from a property observed at optimization convergence into a regularizer applied during training online CRL.

For a batch of trunk features $H_X = [h_1, \ldots, h_M]^\top \in \mathbb{R}^{M \times d}$ at block X, let $\bar{h} = M^{-1} \sum_{i=1}^{M} h_i$ be the batch mean and $\hat{h}_i = (h_i - \bar{h})/\|h_i - \bar{h}\|_2$ the centered, unit-norm feature, as in Eq. (1). Given the network's own readout $W \in \mathbb{R}^{n \times d}$, we define **NRC2Reg** as:

$$\mathcal{L}_{\mathrm{NRC2}}(H_X; W) = \frac{1}{M} \sum_{i=1}^{M} \left\| \hat{h}_i - \mathrm{proj}(\hat{h}_i \mid W^\top) \right\|_2^2 \;=\; \frac{1}{M} \sum_{i=1}^{M} \sin^2 \theta_i \;\in [0, 1], \tag{2}$$

where $\mathrm{proj}(\cdot \mid W^\top)$ denotes orthogonal projection onto $\mathrm{row}(W)$ and θ_i is the principal angle between $\hat{h}_i$ and that subspace. Centering and normalizing makes the loss invariant to the scale and mean of the features, so the network cannot reduce it by shrinking. The only way to decrease $\mathcal{L}_{\mathrm{NRC2}}$ is to rotate the features toward $\mathrm{row}(W)$ or the readout toward the features. The gradient reaches every block up to X and the readout itself.

We add the regularizer to each of CRL's three networks, each against its own readout:

$$\mathcal{L}_{\text{critic}} + \lambda_c \big[\mathcal{L}_{\text{NRC2}}(H_X^{\phi}; W_\phi) + \mathcal{L}_{\text{NRC2}}(H_X^{\psi}; W_\psi)\big], \qquad \mathcal{L}_{\text{actor}} + \lambda_p \, \mathcal{L}_{\text{NRC2}}(H_X^{\theta}; W_\theta), \tag{3}$$

where $\mathcal{L}_{\text{actor}}$ is the negative of the actor objective in Section 2, and $W_\theta \in \mathbb{R}^{2|\mathcal{A}| \times d}$ stacks the policy's mean and log-standard-deviation heads. We set $\lambda_c = \lambda_\pi = 1$ in all experiments, and select a single block X for all three networks. Both choices, and when to switch the policy term off, are examined in Section 5.2.

4.2 How learned compression differs from a bottleneck

The linear readout $z = Wh_L + b$ acts as a projection: it maps the penultimate feature h_L to an n-dimensional output by reading only the component of h_L that lies in $\mathrm{row}(W)$, discarding everything orthogonal to it. In Figure 1b this filtering is visible: the trunk feature expands to an effective rank of about 200 across the residual blocks, yet the encoder output spans only about 11 directions—all necessarily inside $\mathrm{row}(W)$. The task-relevant structure is therefore already a subspace of $\mathrm{row}(W)$, but the trunk arrives there only at the very last step, after carrying many uninformative directions through the entire network.

NRC2Reg (Eq. 2) asks the trunk to shed those directions earlier, by pulling an intermediate feature toward $\mathrm{row}(W)$ before the readout enforces it. Figure 1b shows that this influence is not confined to the intervened block: features at every block before X also become lower-rank. More broadly, the target $\mathrm{row}(W)$ has three properties that a fixed bottleneck lacks:

- The subspace dimension is the readout width n, which the base algorithm already sets ($n = r = 64$ for CRL's encoders), so there is no rank hyperparameter to sweep. A bottleneck introduces a rank k that must be tuned per task. Figure 1b confirms that under NRC2Reg, the rank of the trunk features aligns with that of the encoder output.
- Because n exceeds the intrinsic dimension of the target (e.g., the 3-D positions of humanoid locomotion), the regularizer leaves room for the features to occupy as many or as few of the n available directions as the task requires. A bottleneck of rank $k < n$ can push the features below the target's intrinsic dimension, and Section 3.1 showed that it does. Figure 1c confirms that NRC2Reg, unlike either bottleneck setting, does not compress the penultimate feature below the target's intrinsic dimension.
- The subspace $\mathrm{row}(W)$ is learned jointly with the features: as W updates, the target co-adapts, unlike a bottleneck drawn once at initialization and then frozen. Table 2 shows that this co-adaptation also conditions the readout: its condition number $\kappa(W_\phi)$ drops from $\sim 10^4$ under the unregularized network and both bottlenecks to $\sim 10^2$ under NRC2Reg, while the effective rank of W_ϕ remains at 11.

Rank and intrinsic dimension describe the geometry of the features but not what information they carry. To measure that, we estimate the Gaussian mutual information between pairs of representations. Table 2 reports three quantities: $I(x;\, h_X)$, the information the block-X feature retains about the raw input $x = [s, a]$; $I(h_X;\, \psi(g))$, its relevance to the goal embedding of the positive pair; and $I(\phi(s, a);\, \psi(g))$, the relevance of the final encoder output. From an information-bottleneck perspective (Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017), good compression removes variability the target does not explain while raising relevance to the target (Zhang et al., 2024).

Table 2: Readout geometry and information content of the state-action encoder's features (humanoid, depth 32, block X=3, two seeds, mean $\pm$ s.e. over the last five epochs).

	$\mathrm{rank}(W_\phi)$	$\kappa(W_\phi)$	$I(x;\, h_X)$	$I(h_X;\, \psi(g))$	$I(\phi(s,a);\, \psi(g))$
CRL Baseline	11.0 ± 0.0	6.5×10^4	36.6 ± 0.5	11.4 ± 0.8	30.1 ± 0.0
Bottleneck k=8	11.5 ± 0.5	7.4×10^4	4.9 ± 0.2	4.6 ± 0.2	30.3 ± 0.1
Bottleneck k=64	13.0 ± 0.0	8.9×10^4	21.4 ± 0.4	9.5 ± 0.4	33.3 ± 0.2
NRC2Reg	11.0 ± 0.0	1.4×10^2	35.8 ± 1.3	16.7 ± 3.0	39.6 ± 1.8

The bottleneck fails at this: at k=8 the projected feature retains only 4.9 of the unregularized 36.6 about the input, and its relevance to the goal drops from 11.4 to 4.6. A subspace chosen before training cannot select the directions that carry the target. The remaining blocks compensate—the

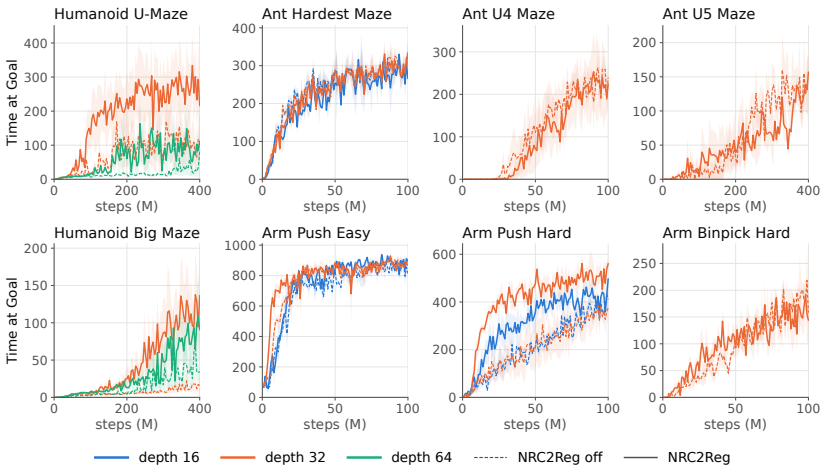

Figure 3: Learning curves of CRL (dashed) and NRC2Reg (solid) on selected nine tasks.

output mutual information $I(\phi; \psi)$ recovers to 30.3—but the mid-network damage limits the final policy (Figure 2). NRC2Reg compresses differently: $I(x; h_X)$ stays at 35.8, nearly unchanged, while $I(h_X; \psi(g))$ rises to 16.7, well above the unregularized 11.4. The learned subspace keeps the input information that matters and discards what does not, lifting the output relevance to 39.6 and translating to both faster learning and a higher final score.

5 NRC2Reg across the CRL benchmark

Section 4 showed on Humanoid that NRC2Reg compresses the state–action features without the information loss of a frozen bottleneck, and that the agent learns faster with it (Figure 2). We now ask whether this holds across the benchmark. Following the protocol of Section 2.2, the CRL baselines are left unmodified and inherit all hyperparameters and training budgets from Wang et al. (2025). Unless ablated later in this section, NRC2Reg is applied to all three CRL networks at block $X = 3$ with $\lambda_c = \lambda_p = 1$. Every task is run at least at its critical depth, which guarantees significant training progress; tasks whose critical depth is 16 are also run at depth 32, and the two challenging humanoid mazes at both depth 32 and depth 64.

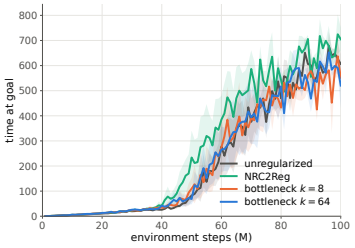

Figure 2: Comparison of NRC2Reg with the unregularized network and the frozen orthogonal bottleneck, on the Humanoid environment. NRC2Reg all at block $X = 3$.

5.1 Empirical Evaluation

Figure 3 shows the learning curves for the remaining eight environments. First, NRC2Reg does not harm learning progress on any environment. Second, on the hard tasks (Humanoid U-Maze, Humanoid Big Maze, and Arm Push Hard) where the depth of CRL matters, NRC2Reg significantly improves both final performance and sample efficiency at every depth tested. Moreover, on these tasks, a shallower network equipped with NRC2Reg surpasses a deeper unmodified CRL baseline. This not only demonstrates the sample efficiency gained from learning low-dimensional features in high-dimensional control, but also points to more efficient parameter usage as RL continues to scale (Lee et al., 2025b; Kim et al., 2026; Wang et al., 2025).

Zooming into the maze environments, we note that Wang et al. (2025) reported the humanoid-maze learning curves on the training layout, where goals are placed throughout the maze including near the fixed start, whereas the ant-maze curves were reported on the evaluation layout, which keeps only the goals beyond the wall or at the far end. We therefore additionally report humanoid-maze performance on the evaluation layout in Figure 4(a,b). Alongside time-at-goal, Figure 4(c,d) records the survival time, the number of steps the humanoid remains upright in a healthy posture (Li et al.,

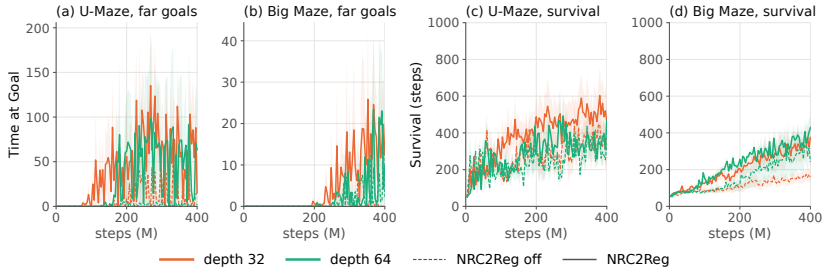

Figure 4: Humanoid U-Maze and Big Maze, with 400M environment steps.**Left**: time at goal on the far-goal layout, which keeps only the goals beyond the wall (U-Maze) or at the far end (Big Maze). **Right**: survival time, the number of steps before the humanoid falls and the episode terminates.

2026). We can observe that performance gain in both humanoid mazes therefore does not rest on the easy goals near the start in the training layout (Figure 3). With the same sample budget, NRC2Reg learns to navigate to the far-end goals and, as the increased survival time shows, also acquires a more stable locomotion behavior along the way. This reduced learning difficulty is likely attributable to easier optimization in the learned lower-dimensional feature space, which an architectural low-rank intervention can be less efficient at figuring out.

5.2 Ablation Study

NRC2Reg leaves three choices: the coefficient λ, which networks carry the loss, and at which block to apply. We study each on Humanoid at depth 32 in this section.

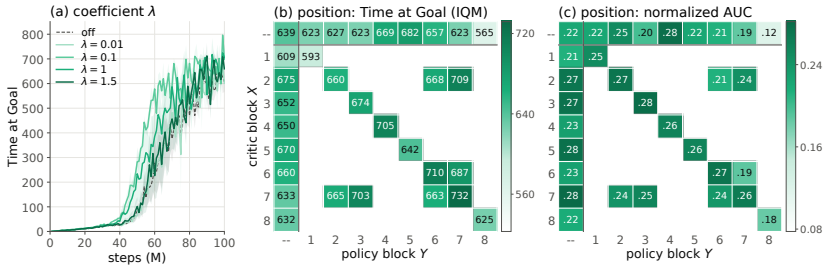

Figure 5: Coefficient and position of NRC2Reg applied to Humanoid with depth 32. **(a)** Learning curves for $\lambda = \lambda_c = \lambda_p$ with both terms at $X = 3$. **(b)** Critic term at block X, policy term at block Y; "–" means the term is off, so the first column is critic-only, the first row is policy-only, and the corner entry is unregularized. We report both IQM of time at goal and normalized AUC over the last five epochs.

Figure 5(a) sweeps $\lambda = \lambda_c = \lambda_p$ with both terms at $X = 3$. At $\lambda = 0.01$ the curve is indistinguishable from CRL's, and at $\lambda = 1.5$ most of the lead is lost again. Between 0.1 and 1 the curves leave CRL's at about 40M steps and reach a time at goal of 500 roughly 10–20M steps earlier. The coefficient thus acts mainly on the rate of learning, with a broad plateau of good values; our default $\lambda = 1$ was fixed before this sweep.

The first column and first row of Figure 5(b,c) apply the loss to the critic alone or the policy alone. Either gives a modest gain at some blocks (up to 675 and 682 against 639), and the two are hard to tell apart. Applying both terms at the same block, along the diagonal, often gives much better performance up to 732 at block 7. The two terms thus act in tandem rather than redundantly.

Final score and sample efficiency favor different positions. The late block 7 gives the highest final score (732) but only a middling AUC (0.25), whereas the early blocks 2 and 3 end lower (660 and 674) with the highest AUC (0.27 and 0.28). As a rule of thumb, regularizing early speeds up learning, while regularizing late raises the ceiling. Two positions are never good targets: at the first block the feature is barely shaped by the network, and at the last block it is the readout's own input,

so pulling it toward $\mathrm{row}(W)$ repeats what the readout already does. Based on these observations we apply NRC2Reg at block 3, which balances sample efficiency and final performance, and the experiments across the CRL benchmark confirm this choice.

6 Related work

Low-dimensional structure in trained networks. Learned features occupy far fewer dimensions than their ambient space: intrinsic dimension first expands and then contracts across layers (Ansuini et al., 2019), and at convergence penultimate features and classifier settle into one shared geometry, neural collapse (Papyan et al., 2020; Mixon et al., 2022). Its regression counterpart, NRC (Andriopoulos et al., 2024), collapses features onto the row space of the readout, a property that also holds in intermediate layers (Rangamani & Unal, 2026) and that can over-compress below the target's topology (Zhang et al., 2024; Andriopoulos et al., 2026). We turn NRC from an observation into an objective.

The manifold hypothesis in RL. Low-dimensional structure has been exploited in the parameter space of RL agents (Zhuang et al., 2026; Tenedini et al., 2026; Fraschini et al., 2026) and in feature space (Tiwari et al., 2025; Todorov & Sabatelli, 2026), on which we build. A separate line treats *loss* of feature rank as a pathology of non-stationary training, implicit under-parameterization (Kumar et al., 2021), capacity loss (Lyle et al., 2022), dormant neurons (Sokar et al., 2023), primacy bias (Nikishin et al., 2022), and adds regularizers that restore rank (Kumar et al., 2022; He et al., 2023). That work concerns uncontrolled collapse of the penultimate feature under bootstrapping; NRC2Reg induces a controlled contraction of trunk features toward $\mathrm{row}(W)$, which Section 4 shows preserves task information.

Representations and scaling in online RL. Better representations are increasingly the route to sample-efficient model-free RL, through learned embeddings (Fujimoto et al., 2025; Obando-Ceron et al., 2026a) or constraints on feature geometry (Obando-Ceron et al., 2026b; Pasand et al., 2026). Our testbed is CRL (Eysenbach et al., 2022; Bortkiewicz et al., 2025; Liu et al., 2025; Bastankhah et al., 2026), whose depth scaling (Wang et al., 2025) follows a broader effort to make large networks trainable in RL via architecture (Lee et al., 2025a;b), critic regularization (Nauman et al., 2024) and gradient stability (Castanyer et al., 2025; Gallici et al., 2025). NRC2Reg adds no parameters and composes with any of these.

7 Conclusion

In this paper, we asked whether the features of an online RL agent can be made low-dimensional by learning, without changing its architecture, and whether this improves sample efficiency. Our answer is yes. NRC2Reg turns the NRC2 property of neural regression collapse into a training objective, pulling intermediate trunk features toward the row space of the network's own readout. This subspace is dimensioned by the base algorithm, adapts as training proceeds, and leaves the features free to occupy only the directions the task requires. Where an architectural bottleneck of the same rank budget discards task-relevant information mid-network and at best recovers baseline performance, the learned compression preserves nearly all input information while more than doubling measured relevance to the goal embedding. Across nine goal-conditioned control tasks at multiple depths, this translates into faster learning everywhere and a higher final score almost everywhere. The effect is most striking on humanoid mazes, where the unregularized agent barely learns and a shallower regularized network surpasses a deeper one.

Beyond the empirical gains, the work suggests a shift in how structure from deep-learning theory can enter RL practice. Neural collapse and NRC have so far described what networks look like at convergence; our results indicate they can also serve as objectives during training. Doing so requires a care that purely architectural compression lacks: the low-dimensional target must itself be learned, and held above the intrinsic dimension of the task. Because NRC2Reg adds no parameters, no tuned rank, and a single broadly effective coefficient, it composes naturally with the architectural and optimization advances that have made deep RL scalable. We expect it to transfer beyond Contrastive RL to other actor-critic methods whose critics regress low-dimensional targets.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for providing feedback on the presentation of research methodology and evaluation protocols, checking the consistency of reported statistics, and reviewing the interpretation of results. We have not used generative AI tools for developing or implementing the proposed method, generating or processing experimental data, or conducting experiments, and formal theorem formulation and proof writing are not applicable to this work. Additionally, we used generative AI tools for drafting, rewriting, and polishing the manuscript, checking relevant literature and citations, and reviewing manuscript formatting. We have reviewed all AI-assisted work. The authors verified methodological descriptions and numerical claims against the underlying experiments and records, and checked cited claims against the original sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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

George Andriopoulos, Zixuan Dong, Li Guo, Zifan Zhao, and Keith W. Ross. The prevalence of neural collapse in neural multivariate regression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Wq6aY6fC2H.

George Andriopoulos, Zixuan Dong, Bimarsha Adhikari, and Keith Ross. Geometric analysis of neural regression collapse via intrinsic dimension, 2026. URL https://arxiv.org/abs/2510.01105.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Mahsa Bastankhah, Grace Liu, Dilip Arumugam, Thomas L. Griffiths, and Benjamin Eysenbach. Demystifying the mechanisms behind emergent exploration in goal-conditioned RL. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mwgYORsqtv.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

Michał Bortkiewicz, Władysław Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating goal-conditioned reinforcement learning algorithms and research. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4gaySj8kvX.

Bradley CA Brown, Anthony L. Caterini, Brendan Leigh Ross, Jesse C Cresswell, and Gabriel Loaiza-Ganem. Verifying the union of manifolds hypothesis for image data. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Rvee9CAX4fi.

Roger Creus Castanyer, Johan Obando-Ceron, Lu Li, Pierre-Luc Bacon, Glen Berseth, Aaron Courville, and Pablo Samuel Castro. Stable gradients for stable learning at scale in deep reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Vqj65VeDOu.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7, 2017. URL https://api.semanticscholar.org/CorpusID:3991422.

Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proceedings of the National Academy of Sciences*, 118(43):e2103091118, 2021.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

Andrea Fraschini, Davide Tenedini, Riccardo Zamboni, Mirco Mutti, and Marcello Restelli. Unsupervised behavioral compression: Learning low-dimensional policy manifolds through state-occupancy matching. *Reinforcement Learning Journal*, 7, 2026. URL https://arxiv.org/abs/2603.27044.

Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=R1hIXdST22.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7IzeL0kflu.

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*, pp. 1861–1870, 2018.

Qiang He, Huangyuan Su, Jieyu Zhang, and Xinwen Hou. Frustratingly easy regularization on representation can boost deep reinforcement learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20215–20225. IEEE, 2023.

Donghu Kim, Youngdo Lee, Minho Park, Kinam Kim, Takuma Seno, I. Made Aswin Nahrendra, Sehee Min, Daniel Palenicek, Florian Vogt, Danica Kragic, Jan Peters, Jaegul Choo, and Hojoon Lee. FlashSAC: Fast and Stable Off-Policy Reinforcement Learning for High-Dimensional Robot Control. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.099. URL https://www.roboticsproceedings.org/rss22/p099.html.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=O9bnihsFfXU.

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. DR3: Value-based deep reinforcement learning requires explicit regularization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=POvMvLi91f.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R. Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. SimBa: Simplicity bias for scaling up parameters in deep reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=jXLiDKsuDo.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=kfYxyvCYQ4.

Guopeng Li, Yiyang Duan, Yiru Jiao, and Chengcheng Xu. Arrive and survive: Scaling safe goal-conditioned policy learning from one-bit failure signals, 2026. URL https://arxiv.org/abs/2608.26571.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Grace Liu, Michael Tang, and Benjamin Eysenbach. A single goal is all you need: Skills and exploration emerge from contrastive RL without rewards, demonstrations, or subgoals. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xCkgX4Xfu0.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZkC8wKoLbQ7.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International conference on machine learning*, pp. 23190–23211. PMLR, 2023.

Dustin G Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. *Sampling Theory, Signal Processing, and Data Analysis*, 20(2):11, 2022.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=fu0xdh4aEJ.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International conference on machine learning*, pp. 16828–16847. PMLR, 2022.

Johan Obando-Ceron, Lu Li, Scott Fujimoto, Pierre-Luc Bacon, Aaron Courville, and Pablo Samuel Castro. Representation learning enables scalable multitask deep reinforcement learning, 2026a. URL https://arxiv.org/abs/2606.05555.

Johan Obando-Ceron, Walter Mayor, Samuel Lavoie, Scott Fujimoto, Aaron Courville, and Pablo Samuel Castro. Simplicial embeddings improve sample efficiency in actor–critic agents. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=mCpq1GCKxA.

Vardan Papyan, Xiao Y Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020.

Ali Saheb Pasand, Johan Obando-Ceron, Aaron Courville, Pouya Bashivan, and Pablo Samuel Castro. Stable deep reinforcement learning via isotropic gaussian representations. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gc7Gg18ejz.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Akshay Rangamani and Altay Unal. Deep neural regression collapse. In *Conference on Parsimony and Learning*, volume 328 of *Proceedings of Machine Learning Research*, pp. 565–581. PMLR, 23–26 Mar 2026. URL https://proceedings.mlr.press/v328/rangamani26a.html.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information, 2017. URL https://arxiv.org/abs/1703.00810.

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.

Davide Tenedini, Riccardo Zamboni, Mirco Mutti, and Marcello Restelli. From parameters to behaviors: Unsupervised compression of the policy space. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VqnBaeu43F.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop*, pp. 1–5. IEEE, 2015.

Saket Tiwari, Omer Gottesman, and George Konidaris. Geometry of neural reinforcement learning in continuous state and action spaces. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AP0ndQloqR.

Aleksandar Todorov and Matthia Sabatelli. Learning in low-dimensional subspaces: Orthogonal bottlenecks for reinforcement learning. *Reinforcement Learning Journal*, 7, 2026. URL https://arxiv.org/abs/2605.26012.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzcinski, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=s0JVsx3bx1.

Yibo Yang, Shixiang Chen, Xiangtai Li, Liang Xie, Zhouchen Lin, and Dacheng Tao. Inducing neural collapse in imbalanced learning: Do we really need a learnable classifier at the end of deep neural network? *Advances in neural information processing systems*, 35:37991–38002, 2022.

Shihao Zhang, Kenji Kawaguchi, and Angela Yao. Deep regression representation learning with topology. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=HbdeEGVfEN.

Yuan Zhuang, Yuexin Bian, Sihong He, Jie Feng, Qing Su, Songyang Han, Jonathan Petit, Shihao Ji, Yuanyuan Shi, and Fei Miao. Low-rank adaptation for critic learning in off-policy reinforcement learning, 2026. URL https://arxiv.org/abs/2604.18978.

A ENVIRONMENTS

All tasks come from JaxGCRL (Bortkiewicz et al., 2025) and follow the setup of Wang et al. (2025). An episode lasts at most 1000 steps, and the agent is at the goal when its goal-relevant state $m(s)$ is within a radius ϵ of the commanded goal g. Table 3 lists the tasks.

Humanoid. A humanoid with 17 actuators must bring its torso to a target placed in a random direction and at a random distance from its start. The episode ends early when the torso height leaves the range $[1, 2]$, that is, when the humanoid falls.

Ant mazes. A quadruped with 8 actuators navigates a maze to a goal location. U4 Maze and U5 Maze are U-shaped corridors of increasing length; the evaluation layout keeps only the goals beyond the turn. Hardest Maze is a large maze with many branches and is evaluated on the whole maze.

Humanoid mazes. The humanoid of the Humanoid task navigates U-Maze and Big Maze, and its episodes end under the same rule. The training layout places goals throughout the maze, including next to the start; the far-goal layout of Figure 4 keeps only the goals beyond the wall (U-Maze) or at the far end (Big Maze).

Arm tasks. A Franka Panda arm, controlled through four joint targets and the gripper opening, must move a cube to a goal position. In Push Easy and Push Hard the cube is pushed across a table to a small or a larger goal region; in Binpick Hard it is picked from one bin and placed at a random position in another.

Table 3: Tasks. $\dim \mathcal{S}$: state dimension; $|\mathcal{A}|$: action dimension; goal: the goal-relevant state $m(s)$; ϵ: goal radius; depth: $4L$, critical depth first and the second depth after the comma.

task	$\dim \mathcal{S}$	$\lvert\mathcal{A}\rvert$	goal	ϵ	evaluation goals	steps	depth
Humanoid	268	17	torso position (3-D)	0.5	random target	100M	32
Ant Hardest Maze	29	8	x-y position	0.5	whole maze	100M	16, 32
Ant U4 Maze	29	8	x-y position	0.5	beyond the turn	100M	32
Ant U5 Maze	29	8	x-y position	0.5	beyond the turn	400M	32
Arm Push Easy	17	5	cube position (3-D)	0.1	goal region	100M	16, 32
Arm Push Hard	17	5	cube position (3-D)	0.1	goal region	100M	16, 32
Arm Binpick Hard	17	5	cube position (3-D)	0.1	target bin	100M	32
Humanoid U-Maze	268	17	torso position (3-D)	0.5	whole maze*	400M	64, 32
Humanoid Big Maze	268	17	torso position (3-D)	0.5	whole maze*	400M	64, 32

*Also evaluated on the far-goal layout (Figure 4).

B METRICS AND STATISTICS

Performance is measured on 128 evaluation episodes after every epoch of 10^6 environment steps. Representation metrics are computed on the state-action encoder from $N = 4096$ samples of the replay buffer after every epoch, each sample a positive pair (s_i, a_i, g_i), and are reported as the mean over the last five epochs.

Time at goal. The performance measure of JaxGCRL (Bortkiewicz et al., 2025): the number of steps of an episode spent within ϵ of the goal,

$$\mathrm{TaG} = \sum_{t=1}^{T} \mathbb{1}\big[\|m(s_t) - g\|_2 < \epsilon\big] \in [0, 1000], \tag{4}$$

averaged over the evaluation episodes, where $T \leq 1000$ is the episode length.

Interquartile mean. Following Agarwal et al. (2021), final performance pools the time at goal of the last $K = 5$ epochs over n seeds, sorts the nK values $v_{(1)} \leq \cdots \leq v_{(nK)}$, and averages the

middle half,

$$\mathrm{IQM} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} v_{(j)}, \qquad \mathcal{J} = \big\{ \lfloor nK/4 \rfloor + 1, \ldots, nK - \lfloor nK/4 \rfloor \big\}. \tag{5}$$

The reported uncertainty is the standard error of the retained values, $\mathrm{sd}(v_{\mathcal{J}})/\sqrt{|\mathcal{J}|}$. It describes the spread over seeds and late epochs, not a seed-level confidence interval.

Normalized AUC. Sample efficiency is the area under the learning curve, normalized by the episode length, over the E epochs of training,

$$\mathrm{AUC} = \frac{1}{1000\,E} \sum_{e=1}^{E} \mathrm{TaG}_e \in [0, 1], \tag{6}$$

averaged over seeds.

Seed statistics. Learning curves and the representation metrics report the mean over seeds $\pm$ the standard error $\mathrm{sd}/\sqrt{n}$.

Survival time. The mean evaluation episode length T. On the humanoid tasks the episode ends when the humanoid falls (Li et al., 2026), so survival bounds the time at goal.

Effective rank. For a batch of features $H \in \mathbb{R}^{N \times d}$ with centered singular values $\sigma_1 \geq \sigma_2 \geq \ldots$, the effective rank is the number of directions that hold 99% of the energy,

$$\mathrm{rank}_{99}(H) = \min\Big\{ k : \sum_{i \leq k} \sigma_i^2 \geq 0.99 \sum_i \sigma_i^2 \Big\}, \tag{7}$$

a variant of the srank of Kumar et al. (2021) that counts squared singular values. The rank of the readout $W_\phi \in \mathbb{R}^{64 \times 256}$ uses the same definition on its singular values.

Condition number. The ratio of the largest to the smallest of the 64 singular values of the readout, $\kappa(W_\phi) = \sigma_{\max}(W_\phi)/\sigma_{\min}(W_\phi)$.

Intrinsic dimension. We use the two-NN estimator (Facco et al., 2017). For each sample, let $r_{i,1} \leq r_{i,2}$ be the distances to its two nearest neighbors and $\mu_i = r_{i,2}/r_{i,1}$. If the density is locally uniform on a D-dimensional manifold, μ follows a Pareto law with $-\log\big(1 - F(\mu)\big) = D \log \mu$. With $\mu_{(1)} \leq \cdots \leq \mu_{(N)}$ and $F_j = j/N$, the estimate is the least-squares slope through the origin,

$$\hat{D} = \frac{\sum_j x_j y_j}{\sum_j x_j^2}, \qquad x_j = \log \mu_{(j)}, \quad y_j = -\log(1 - F_j), \tag{8}$$

over the smallest 90% of the ratios (Ansuini et al., 2019). The input (s, a) is standardized per coordinate; all other tensors are used as they are.

Mutual information. For jointly Gaussian u and v with canonical correlations $\rho_1, \rho_2, \ldots$, the mutual information is

$$I(u; v) = \frac{1}{2} \log \frac{\det \Sigma_u \det \Sigma_v}{\det \Sigma_{uv}} = -\frac{1}{2} \sum_i \log(1 - \rho_i^2). \tag{9}$$

We estimate the ρ_i between the top 64 principal components of the two batches (Raghu et al., 2017) and subtract the same estimate with the rows of v permuted, which removes the finite-sample bias of independent batches. Values are in nats; $x = [s, a]$ is standardized per coordinate, and $\psi(g)$ is the goal embedding of the positive pair.

C HYPERPARAMETERS

CRL follows Wang et al. (2025) on every task (Table 4); NRC2Reg adds two coefficients and one block index.

Table 4: Hyperparameters. Unlisted values follow Wang et al. (2025).

CRL	
residual block	four units $u = \mathrm{Swish}(\mathrm{LN}(Ah + c))$, one skip connection
width d	256
encoder output dimension r	64
depth $4L$	per task (Table 3)
optimizer	Adam, learning rate 3×10^{-4} (actor, critic, temperature)
batch size M	512
discount γ	0.99
log-sum-exp regularization of InfoNCE	0.1
entropy temperature α	tuned automatically (Haarnoja et al., 2018), target entropy $-\lvert\mathcal{A}\rvert/2$
parallel environments	512
NRC2Reg	
coefficients λ_c, λ_p	1, 1
block X	3 at every depth
Orthogonal bottleneck	
projection B	orthonormal columns from the QR decomposition of a Gaussian matrix, frozen
networks	both critic encoders ϕ and ψ

D PER-TASK RESULTS

Table 5 reports the final performance and the AUC behind Figure 3 and Figure 2.

Table 5: Final performance and sample efficiency of CRL without (off) and with NRC2Reg (both terms at block 3, $\lambda_c = \lambda_p = 1$). Time at goal: IQM over the last five epochs pooled over seeds, $\pm$ standard error of the retained values; AUC: normalized area under the learning curve (Appendix B). Three seeds per entry, five for the humanoid mazes at depth 32, two for the regularized depth-16 runs of Ant Hardest Maze and Arm Push Easy. Bold: the higher value. †: the task's second depth. Ant U5 Maze: regularized seeds from separate runs evaluated every 4M steps.

task	depth	budget	time at goal		AUC	
			off	NRC2Reg	off	NRC2Reg
Humanoid	32	100M	639 ± 12	$\mathbf{674 \pm 20}$	0.222	**0.278**
Ant Hardest Maze	16	100M	$\mathbf{284 \pm 5}$	276 ± 5	**0.235**	0.214
Ant Hardest Maze†	32	100M	296 ± 7	$\mathbf{309 \pm 9}$	**0.235**	0.227
Ant U4 Maze	32	100M	$\mathbf{246 \pm 9}$	204 ± 24	**0.111**	0.094
Ant U5 Maze	32	400M	122 ± 12	$\mathbf{157 \pm 22}$	**0.059**	0.051
Arm Push Easy	16	100M	856 ± 12	$\mathbf{902 \pm 4}$	0.699	**0.746**
Arm Push Easy†	32	100M	$\mathbf{887 \pm 7}$	879 ± 7	0.760	**0.787**
Arm Push Hard	16	100M	384 ± 17	$\mathbf{435 \pm 12}$	0.213	**0.312**
Arm Push Hard†	32	100M	354 ± 9	$\mathbf{533 \pm 12}$	0.209	**0.409**
Arm Binpick Hard	32	100M	$\mathbf{184 \pm 7}$	152 ± 14	0.094	**0.100**
Humanoid U-Maze†	32	400M	61 ± 13	$\mathbf{276 \pm 17}$	0.070	**0.190**
Humanoid U-Maze	64	400M	25 ± 9	$\mathbf{74 \pm 18}$	0.016	**0.052**
Humanoid Big Maze†	32	400M	9 ± 1	$\mathbf{111 \pm 12}$	0.006	**0.044**
Humanoid Big Maze	64	400M	39 ± 7	$\mathbf{101 \pm 18}$	0.014	**0.030**

E READOUT DIMENSION

NRC2Reg pulls the trunk feature toward $\mathrm{row}(W_\phi)$, whose dimension is the encoder output dimension r. Varying r therefore changes how far the regularizer can compress (Figure 6). Without it, CRL degrades as r grows (660, 639 and 583 for $r = 32$, 64 and 128); with it the trend reverses (646, 674 and 724). The regularized block-3 feature settles at rank 4.7, 10.5 and 24.2, while the encoder output uses 11 to 12 directions at every r. At $r = 32$ the target subspace is too narrow: the feature is compressed to about the rank of the $k = 8$ bottleneck (5.5), and NRC2Reg no longer helps. From $r = 64$ on, the target leaves room for the output's rank and the gain grows with r. What helps is

compression toward the readout into a subspace at least as wide as the output's rank, not low rank as such.

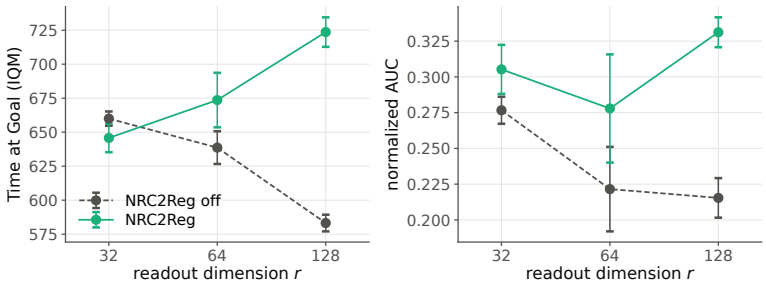

Figure 6: Readout dimension r: final IQM (left) and AUC (right). A wider readout hurts CRL and helps NRC2Reg. $r = 64$: three seeds per agent; $r = 32, 128$: runs of 150M steps read over 95M to 100M, three seeds with NRC2Reg, nine without.