Hadamard Representation: Scaffolding Performance Across Model-free RL

Jacob E. Kooi
jacobkooi92@gmail.com

Zhao Yang *
yangzhaooba@gmail.com

Mark Hoogendoorn
m.hoogendoorn@vu.nl

Vincent François-Lavet
vincent.francoislavet@vu.nl

Vrije Universiteit Amsterdam

Abstract

Deep reinforcement learning agents progressively lose representational capacity during training: neurons become dormant, removing active capacity from the network, and effective rank collapses, leaving surviving neurons redundant. Existing remedies such as periodic resets, and special neural network architectures, are largely algorithm- or domain-specific. We propose a simple architectural fix, the Hadamard Representation (HR), which replaces a standard hidden layer with the element-wise product of two independently parameterized layers. HR operates through two complementary mechanisms. First, it reduces the probability of a neuron becoming dormant, which is particularly valuable for continuously differentiable activations such as $\tanh$: unlike dormant ReLU neurons, which are effectively pruned, saturated $\tanh$ neurons silently corrupt downstream layers by turning their outgoing weights into fixed biases. Second, independently of dormancy, the multiplicative structure captures richer feature interactions and increases effective rank without widening the layer. We evaluate HR across five algorithms and three domains: DQN, PPO, and PQN on pixel-based discrete-action Atari, SimbaV2 on state-based continuous control, and MR.Q on visual continuous control. HR consistently improves performance over the strong baselines without any hyperparameter tuning, and gains persist against parameter-matched wider variants, ruling out parameter count as an alternative explanation.

1 Introduction

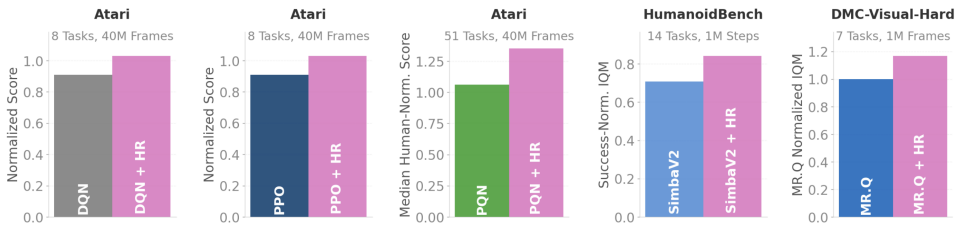


Figure 1: Improvements with Hadamard Representation (HR) over the baselines across five RL algorithms and three domains.

*Corresponding Author

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

Deep reinforcement learning agents progressively lose representational capacity as training unfolds: neurons become dormant [47], effective rank collapses [30], and learned features become impoverished [9]. Existing remedies such as auxiliary losses [42], periodic resets [47], and normalization [26] address these symptoms through algorithm- or domain-specific adjustments. In this work, we instead revisit the problem from an architectural angle, starting with the behavior of dormant neurons under different activation functions.

The dying ReLU phenomenon in RL is well-documented [47, 29, 15], but $\tanh$ has received far less attention despite producing representations with fewer dormant neurons and higher effective rank [15]. If the dormant count were the full story, $\tanh$ should outperform ReLU in RL. Empirically, it does not. This suggests that not all dormant neurons are equal: a dormant ReLU outputs zero and is effectively pruned, whereas a saturated $\tanh$ neuron collapses to ± 1 and converts its outgoing weights into a fixed, input-independent bias that silently distorts downstream layers.

Motivated by this observation, we study a simple architectural change that reduces the probability of a $\tanh$ neuron becoming dormant in the first place: the **Hadamard Representation (HR)**, which replaces a standard hidden layer with the element-wise product of two independently parameterized activations,

$$\mathbf{z}(\mathbf{x}) = f(\mathbf{A}_1\mathbf{x} + \mathbf{b}_1) \odot f(\mathbf{A}_2\mathbf{x} + \mathbf{b}_2), \tag{1}$$

where $\mathbf{x}$ is the input to the layer, $\mathbf{z}$ its output, f the activation function, and $(\mathbf{A}_i, \mathbf{b}_i)$ two independent sets of weights and biases. Under HR, product saturation requires both branches to saturate simultaneously, which reduces the per-neuron collapse probability for $\tanh$. Beyond saturation resistance, HR offers a second, complementary effect: the product of two independently parameterized activations captures second-order interactions between two views of the input, creating a richer function class at the same layer width. Similar multiplicative structures appear in gating mechanisms such as GLUs [7] and highway networks [50], but in HR both branches are full activation outputs rather than one being a scalar gate, so the structure remains symmetric. Even when dormant neurons are already rare, this multiplicative expressiveness can provide gains.

We evaluate HR across five algorithms and three domains, see Figure 1. In pixel-based discrete-action Atari [3] domain, HR delivers over 10% gains for DQN [32] and PPO [41] across 8 games and shows consistent scaling on 51 games with PQN [12]. In continuous control, HR improves state-of-the-art methods like SimbaV2 [27] on state-based domain HumanoidBench [45] and MR.Q [11] on visual-based domain DMC-Visual [53] **without** any hyperparameter tuning.

Our contributions are as follows:

- We identify a failure mode unique to smooth activations in RL: saturated $\tanh$ neurons silently convert their outgoing weights into hidden biases, a strictly more harmful failure mode than ReLU pruning.
- We study the Hadamard Representation (HR), a drop-in multiplicative layer that reduces the per-neuron collapse probability for $\tanh$, while amplifying it for ReLU. Independently of its effect on dormancy, HR also increases the effective rank of hidden representations, which benefits architectures where dormant neurons are already rare.
- We show consistent gains across three diverse domains: pixel-based Atari with discrete actions (DQN, PPO, PQN), state-based continuous control (SimbaV2 on HumanoidBench), and visual continuous control (MR.Q on DMC-Visual). PQN, SimbaV2, and MR.Q are among the strongest methods in their respective domains, and HR improves them further without any hyperparameter tuning.

2 Related Work

We organize related work along two axes: (i) representational capacity loss in RL and its proposed mitigation, and (ii) architectural innovations in model-free deep RL.

Capacity loss in RL. A growing body of work has documented how deep RL networks progressively lose representational capacity during training. Prior work has investigated the role of sparse representations in continuous control [28] and introduced effective rank as a measure of network expressiveness in RL [24]. Subsequent analysis showed that $\tanh$ activations preserve high effective rank and resist rank collapse [15], and capacity loss has been characterized as a general phenomenon

in which effective network capacity decays as training progresses [30]. In pixel-based robotic control, normalization combined with action penalization has been shown to reduce variance and capacity loss [5].

Several interventions have been proposed to mitigate these pathologies. In the sample-efficient regime, periodic network resets counteract the primacy bias [36], and the dying ReLU phenomenon has been characterized and mitigated through targeted resets in DQN [47]. For longer training horizons, plasticity injection [35], rational activations [33, 8], and continual backpropagation [9] have been proposed to preserve plasticity. A complementary line of work shows that a substantial fraction of network capacity is unnecessary for effective training [2, 14, 49, 52, 37], offering a plausible explanation for why ReLU networks achieve strong performance despite a significant fraction of dormant neurons. Our approach is complementary to these lines of work: rather than proposing an algorithmic fix that targets symptoms of capacity loss, we introduce a minimal architectural change that directly reduces the probability of neuron dormancy.

Architecture in model-free RL. An early empirical study of activation functions and architectures in RL is given by Henderson et al. [19]. Subsequent work has explored architectural design across the three main model-free settings. **Pixel-based Atari.** Impala [10] introduced deep ResNet [18] blocks to enable high data efficiency under distributional training, and BBF [44] further widened the Impala architecture to achieve strong performance on Atari-100k [62]. A separate line of work has investigated pooling: Hadamax combines max-pooling with Hadamard representations [22], while Impoola [55] and related approaches [48] use global average pooling, both tailored for Atari encoders. **State-based continuous control.** Purpose-built architectures have demonstrated strong performance through carefully chosen normalization: LayerNorm [34, 26], RSNorm [27], and BatchNorm [4, 39, 40]. **Visual continuous control.** Architectural innovations are comparatively scarce, and most gains have instead come from data augmentation [23, 61], improved representation-learning objectives [42, 11], or better exploration [57]. HR is orthogonal to these directions and can be combined with them: it is a drop-in layer-level modification that does not alter the encoder topology, normalization scheme, or training objective.

3 Preliminaries

Reinforcement learning. We consider an agent acting within its environment as a Markov Decision Process (MDP) defined as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ [51, 56, 60]. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \to \mathbb{P}(\mathcal{S})$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The agent's goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted return $V^\pi(s) = \mathbb{E}_\tau[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s]$, where τ is a trajectory following π.

However, in many practical settings, including the pixel-based tasks investigated in this work, the agent lacks direct access to the underlying environment state s_t. These scenarios are more accurately modeled as a Partially Observable Markov Decision Process (POMDP) [21], defined by the augmented tuple $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma)$. In this framework, Ω is the observation space and $O : \mathcal{S} \times \mathcal{A} \to \mathbb{P}(\Omega)$ is the observation function, characterizing the probability of perceiving $o_t \in \Omega$ given the current state and preceding action. In Atari or visual continuous control that we are used in this work, it is standard practice to stack multiple consecutive frames to form a state proxy, effectively approximating the Markov property by incorporating the temporal context necessary to resolve partial observability.

Dormant neurons. A neuron α_i in a hidden layer of dimension w is considered dormant if it produces an approximately constant output across observations, indicating a loss of plasticity. Specifically, a neuron is dormant if $\alpha_i \approx \Omega$ for all $o_t \in B$, where o_t is an observation in the minibatch B and Ω is the saturation limit of the activation function: $\Omega = 0$ for `ReLU` and $\Omega \in \{-1, +1\}$ for $\tanh$. For `ReLU`, this reduces to a direct check of whether activations are identically zero across the batch. For $\tanh$, an approximate equality is necessary since the function reaches ± 1 only asymptotically. We operationalize this via kernel density estimation (KDE) [46] on per-neuron activation distributions, flagging a neuron as dormant when the peak of its estimated density exceeds a threshold. Full details on bandwidth selection and KDE computation are provided in Appendix C.1.

Effective rank. Following Kumar et al. [24], the effective rank of a feature matrix Φ at threshold δ is defined as:

$$\text{srank}_\delta(\Phi) = \min \left\{ k : \frac{\sum_{i=1}^{k} \sigma_i(\Phi)}{\sum_{i=1}^{d} \sigma_i(\Phi)} \geq 1 - \delta \right\}, \tag{2}$$

where $\{\sigma_i(\Phi)\}$ are the singular values of Φ sorted in descending order, i.e., $\sigma_1 \geq \cdots \geq \sigma_d \geq 0$. Intuitively, srank_δ represents the minimum number of orthogonal singular-value directions required to capture $(1 - \delta)$ of the total spectral mass. This corresponds to the number of "effective" unique components that form the basis for linearly approximating the learned representation; a higher value indicates that a greater number of dimensions carry distinct, useful signals. Full details regarding the computation of effective rank can be found in Section C.2.

4 From Dormant Neurons to Hadamard Representations

We analyze DQN on 8 representative Atari games to understand what limits representation quality in deep RL, and use the resulting diagnosis to motivate our proposed architecture. Figure 2 tracks performance, dormant neuron fraction, and effective rank of the representation $\mathbf{z}_t$ (here for DQN, we apply HR on the last hidden layer of the network) during training under five configurations: the original DQN architecture (`ReLU`), DQN with $\tanh$, DQN with $\tanh$ + `LayerNorm`, and two variants with our proposed Hadamard Representation (HR), $\tanh$ + HR and `ReLU`+ HR.

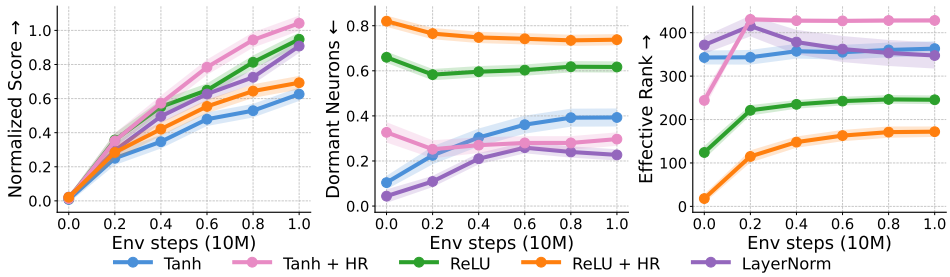
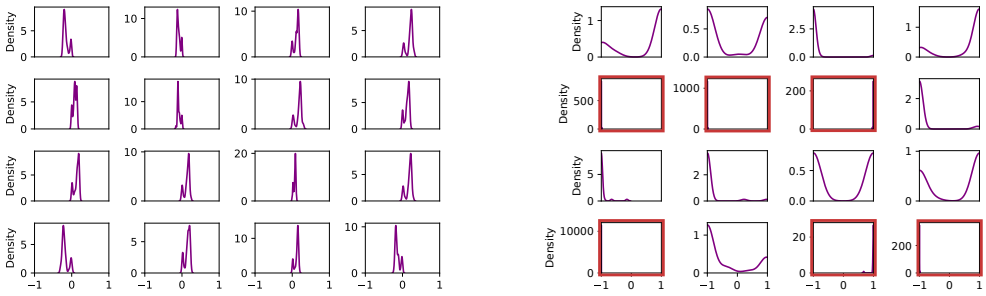


Figure 2: Training DQN on 8 Atari games for 10M iterations (40M frames) under five configurations. **(Left)** Aggregated normalized score. **(Middle)** Fraction of dormant neurons. **(Right)** Effective rank [24]. $\tanh$ activations exhibit strong dormancy comparable to `ReLU`, while HR with $\tanh$ reduces dormancy and increases effective rank simultaneously.



(a) $\tanh$ at 10^6 steps (b) $\tanh$ at 10^7 steps

Figure 3: Kernel density estimates (KDE) of 16 neurons from the compressed representation $\mathbf{z}_t \in \mathbb{R}^{512}$ during DQN training [31] on Breakout with $\tanh$ activations. Red outlines mark dormant neurons whose density concentrates sharply at ± 1, indicating a near-total loss of variance across observations.

4.1 Revisiting Dormant Neurons in RL

Observation 1: dormant `ReLU` costs capacity. The dormant `ReLU` phenomenon is well-studied in RL [47, 29, 15]. Figure 2 confirms it: around 60% of `ReLU` neurons become dormant during training,

with correspondingly low effective rank. A natural goal follows: reduce dormant neurons to recover capacity.

Observation 2: $\tanh$ satisfies this goal and still does not win. $\tanh$ has fewer dormant neurons than ReLU ($\sim$40% vs 60%) and substantially higher effective rank, which is also observed by prior work [15]. If the dormant count were the full story, $\tanh$ should outperform ReLU. In our experiments it performs even worse. Something is missing.

Observation 3: dormant $\tanh$ neurons are not silent. The count understates the damage when the activation saturates to a nonzero value. For ReLU ($\Omega = 0$), a dormant neuron contributes zero and is effectively pruned. For $\tanh$ ($\Omega \in \{-1, +1\}$), it continues to contribute in a harmful way: turning weights into biases.

Theorem 4.1. *When any subset of neurons in a hidden layer $\mathbf{z}^j$ collapses to a nonzero constant Ω across all observations, the pre-activation of the subsequent layer $\mathbf{h}^{j+1}$ effectively gains a fixed, input-independent bias B_*^{j+1}. The magnitude of this bias is determined by the outgoing weights $\mathbf{A}^{j+1}$ associated with the dormant neurons.*

Proof. Let $\mathcal{D}$ be the subset of dormant neurons and $\mathcal{C}$ be the subset of active neurons in layer j. The pre-activation of the subsequent layer is defined as $\mathbf{h}^{j+1} = \mathbf{A}^{j+1}\mathbf{z}^j + \mathbf{b}^{j+1}$. Partitioning the input based on these sets yields:

$$\mathbf{h}^{j+1} = \sum_{i \in \mathcal{C}} z_i^j \mathbf{A}_{:,i}^{j+1} + \underbrace{\sum_{i \in \mathcal{D}} \Omega \mathbf{A}_{:,i}^{j+1}}_{\text{Effective Bias } B_{\text{eff}}^{j+1}} + \mathbf{b}^{j+1}, \tag{3}$$

where $\mathbf{A}_{:,i}^{j+1}$ denotes the i-th column of the weight matrix. For ReLU, $\Omega = 0$, meaning the second term vanishes and the layer is effectively pruned. For $\tanh$, $\Omega \in \{-1, +1\}$, so the dormant neurons contribute a constant vector $B_*^{j+1} = \sum_{i \in \mathcal{D}} \Omega \mathbf{A}_{:,i}^{j+1}$ that shifts the original bias $\mathbf{b}^{j+1}$. This shift persists as a fixed, input-independent distortion regardless of the input $\mathbf{x}$. □

Figure 4 illustrates this on Seaquest: If a neuron becomes dormant in the layer right before the output, it retains the same value for any input observation, but a multiplication of the nonzero saturation value with its outgoing weights implements a substantial 'hidden' bias on the Q-values. The Q-value contribution from dormant $\tanh$ neurons grows over training until it is comparable to that of live neurons, injecting a large state-independent offset into Q-value estimates. Figure 3 shows the corresponding activations progressively saturating toward ± 1. For smooth activations, residual dormant neurons are actively harmful rather than merely inert, so an effective architectural intervention should prevent saturation in the first place.

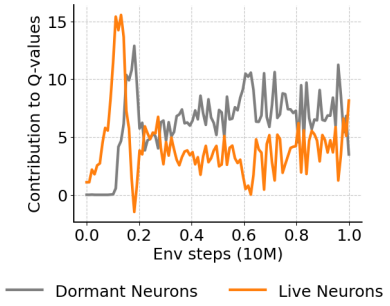


Figure 4: Average Q-value contribution of live and dormant neurons in the final hidden layer during 10M steps of DQN training with $\tanh$ activations on Seaquest. A dormant neuron retains a constant saturation value for any input.

4.2 Hadamard Representation

A standard hidden layer is $\mathbf{z}(\mathbf{x}) = f(\mathbf{A}_1\mathbf{x} + \mathbf{b}_1)$. HR augments this with a parallel, independently parameterized branch and defines the hidden layer as their element-wise product,

$$\mathbf{z}(\mathbf{x}) = f(\mathbf{A}_1\mathbf{x} + \mathbf{b}_1) \odot f(\mathbf{A}_2\mathbf{x} + \mathbf{b}_2), \tag{4}$$

which can be viewed as a highway layer with a closed carry gate [50] or as a symmetric variant of the Gated Linear Unit [7] in which both branches share the same activation. Figure 5 shows the architecture. HR improves representations through two complementary mechanisms.

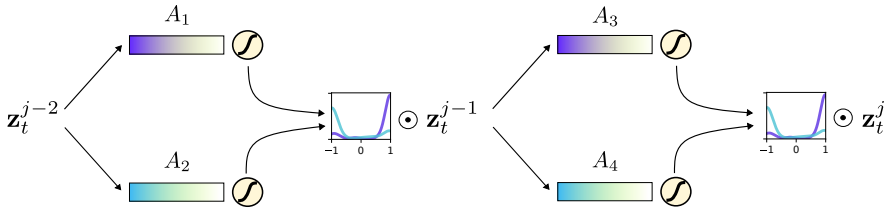


Figure 5: The Hadamard representation. Horizontal bars represent weight vectors and $\mathbf{z}_t$ represents a hidden layer. Two parallel, independently parameterized activation layers are formed between consecutive hidden layers, and their element-wise product defines the propagated hidden layer.

Mechanism 1: saturation resistance. Our key hypothesis is that HR prevents saturation, thereby alleviating vanishing gradients and the emergence of dormant neurons. To support this, we examine the derivative of a product of two functions $g(x) \cdot h(x)$, which is defined as $g'(x)h(x) + g(x)h'(x)$. For a product of two $\tanh$ functions, $\mathbf{z}'(\mathbf{x})$ is:

$$\mathbf{z}'(\mathbf{x}) = \mathbf{A}_1 \operatorname{sech}^2(\mathbf{A}_1\mathbf{x} + \mathbf{b}_1) \odot \tanh(\mathbf{A}_2\mathbf{x} + \mathbf{b}_2) + \mathbf{A}_2 \operatorname{sech}^2(\mathbf{A}_2\mathbf{x} + \mathbf{b}_2) \odot \tanh(\mathbf{A}_1\mathbf{x} + \mathbf{b}_1),$$

where sech^2 is the derivative of $\tanh$. Because $\tanh$ saturates to nonzero values ($\Omega \in \{-1, +1\}$), the product remains active unless both branches saturate simultaneously. Even if one branch saturates, the other preserves a non-trivial gradient path through its $\tanh$ factor, providing a mechanism to avoid vanishing gradients.

More formally, let p denote the saturation probability of a single baseline neuron. Assuming branch independence and symmetric saturation probabilities, the probability of collapse for $\tanh$ reduces to p^2, since saturation of the element-wise product requires both independent branches to fail. Conversely, for `ReLU` where $\Omega = 0$, the product collapses if either branch saturates to zero; thus, the probability of collapse increases to $2p - p^2$ (the probability that one of the two neurons does not saturate is $1 - p$, and the probability that both neurons do not saturate is $(1 - p)^2$, thus probability that at least one neuron saturates is thus equal to $1 - (1 - p)^2 = 2p - p^2$). Table 1 shows that these theoretical shifts track empirical measurements across 8 Atari games (-23% for $\tanh$, $+18\%$ for `ReLU`) closely. Furthermore, while `LayerNorm` reduces dormant $\tanh$ neurons more aggressively than HR, Figure 2 demonstrates that HR produces a corresponding gain in effective rank, compared with the original $\tanh$.

Table 1: Dormant neuron fractions under HR: theoretical predictions and empirical measurements averaged over 8 Atari games. HR reduces dormancy for $\tanh$ but amplifies it for ReLU, consistent with the predicted probabilities.

Activation	**Without HR**	**With HR**	**Δ (predicted)**	**Δ (measured)**
$\tanh$	p (0.39)	p^2 (0.30)	$-(p - p^2)$	-23%
ReLU	p (0.62)	$2p - p^2$ (0.73)	$+(p - p^2)$	$+18\%$

Mechanism 2: multiplicative expressiveness. Independently of dormancy, each output neuron z_i under HR computes a second-order interaction between two independently learned projections: $z_i(\mathbf{x}) = f(\mathbf{a}_{1,i}^\top \mathbf{x} + b_{1,i}) \cdot f(\mathbf{a}_{2,i}^\top \mathbf{x} + b_{2,i})$, where $\mathbf{a}_{k,i}^\top$ denotes the i-th row of the weight matrix $\mathbf{A}_k$. This architecture effectively expands the function class at a fixed layer width. Unlike the gating mechanisms in highway networks and GLUs, both branches in HR contribute activation mass equally rather than one branch acting as a scalar gate for the other. Section 5.2 demonstrates that this multiplicative structure produces effective rank gains even in regimes where dormant neurons are already rare, establishing it as an independent source of representational benefit.

When does each mechanism dominate? The two mechanisms are simultaneously active; their relative importance depends on the dormancy rate of the baseline. When dormant neuron rates are high (e.g., DQN and PPO on Atari), mechanism 1 is decisive and the activation must be $\tanh$: HR with `ReLU` amplifies dormancy and should be avoided. When dormant neuron rates are low (e.g.,

SimbaV2 and MR.Q, where LayerNorm is built into the architecture), mechanism 1 offers little to gain, mechanism 2 becomes the primary driver, and the original activation of the architecture can be preserved.

5 Experiments

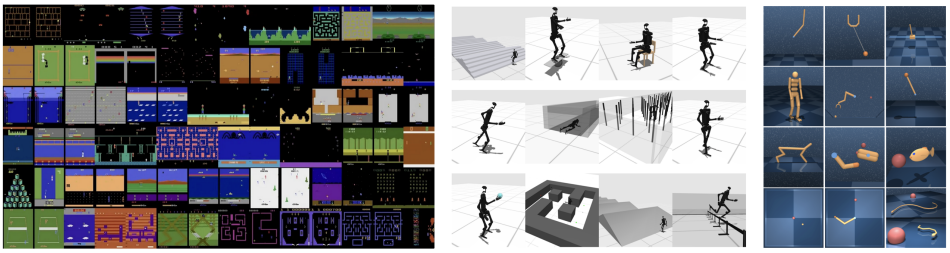


Figure 6: Three domains in our evaluation: pixel-based discrete control (Atari), state-based continuous control (HumanoidBench), and visual continuous control (DMC-Visual).

For all experiments, HR is applied **without** modifying or tuning any hyperparameters from the original baseline codebases. We evaluate HR across three domains selected to test the two mechanisms identified in Section 4: the high-dormancy regime (Atari), where saturation resistance should dominate, and the low-dormancy regime (state-based and visual continuous control), where multiplicative expressiveness serves as the primary driver. As detailed in the following sections, we confirm performance gains in the high-dormancy regime at scale (Section 5.1), isolate the expressiveness mechanism in the state-of-the-art continuous control methods (Section 5.2), and rule out parameter count as a confounding factor and isolate the multiplicative interaction via controlled ablations (Section 5.3).

Algorithms and benchmarks. We evaluate HR across five algorithms spanning these regimes. For Atari, we utilize DQN [32] for qualitative representational analysis, PPO [41] as a policy-gradient confirmation, and PQN [13] for a large-scale evaluation across 51 games. For continuous control, we pair HR with current state-of-the-art methods: SimbaV2 [27] on HumanoidBench and MR.Q [11] on DMC-Visual. All results are reported across 3–5 seeds per task. Aggregate scores are computed by first averaging over seeds per task and then calculating the Interquartile Mean (IQM) or median across the suite; normalization is applied per task prior to aggregation where noted. Detailed per-task comparisons are available in Section D.

5.1 Atari: High-Dormancy Regime

We confirm the DQN performance gain from Figure 2 with two additional algorithms: PPO on 8 Atari tasks, and PQN on 51 Atari tasks at scale, each trained for 10M steps (40M frames).

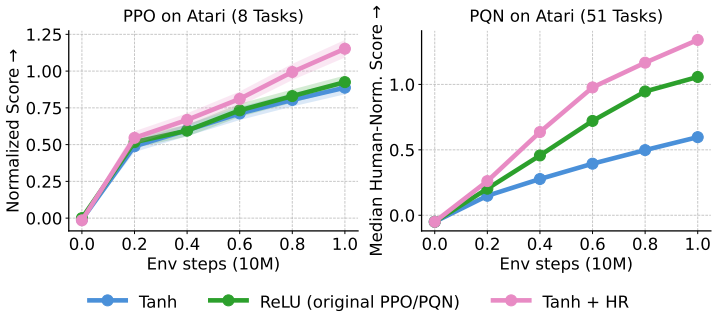


Figure 7: **Left:** Normalized performance on PPO across 8 Atari games. **Right:** Median human-normalized scores on PQN across 51 Atari games (5 seeds, 40M frames). HR applied to tanh surpass the standard tanh and ReLU baseline. No hyperparameters are changed for HR.

PPO. In PPO, the final hidden layer $\mathbf{z}_t$ (where HR is applied on) feeds both the actor and the critic, and therefore receives gradients from both policy and value losses. After 40M frames, tanh + HR attains the best performance (Figure 7, left), while tanh and ReLU perform comparably. This mirrors the DQN

results in Figure 2 and confirms that HR's gain generalizes across value-based and policy-gradient algorithms.

PQN at scale. PQN is a vectorized variant of DQN that delivers both faster training and improved convergence [12]. We apply HR in all hidden layers and evaluate across 51 non-exploration-driven Atari games with 5 seeds per game. Figure 7 (right) shows that HR increases the median human-normalized score substantially, with $\tanh$ + HR improving over standard $\tanh$ by more than 100%. `ReLU` outperforms $\tanh$ here, as it does in DQN, which is consistent with the broader observation that continuously differentiable activations are generally disfavored relative to `ReLU` in RL [54]; HR closes this gap while preserving the high effective rank that $\tanh$ naturally supports. We also show that HR is more effective than other methods that are proposed to prevent the loss plasticity in RL (i.e. CReLU [1] and ReDo [47], see Section C.4).

5.2 Continuous Control: Low-Dormancy Regime

Beyond Atari, we evaluate HR on two continuous control settings: 14 tasks from HumanoidBench (low-dimensional state-based) and 7 tasks from DMC-Visual-Hard (pixel-based); results on the full 28-task DMC-Visual suite are reported in Appendix D.4. In each setting we augment the current state-of-the-art method with HR: SimbaV2 (HR is applied on on the first hidden layer of the SimbaV2 block) on HumanoidBench and MR.Q (HR is applied on the encoder) on DMC-Visual. Both baselines already exhibit negligible dormant neuron rates, which isolates mechanism 2 (multiplicative expressiveness) as the only one available for HR to leverage.

Figure 8 shows that despite near-zero dormant neuron rates in both baselines, HR increases the effective rank and yields corresponding performance improvements. This confirms that HR's gains do not depend solely on the presence of dormant neurons, and that its multiplicative expressiveness is an independent source of benefit. Combined with the Atari results, these findings support the two-mechanism account: HR's primary contribution shifts from saturation resistance (Atari, $\tanh$-specific) to multiplicative expressiveness (continuous control, activation-agnostic) as the dormancy rate of the baseline decreases. Results with an additional high-throughput off-policy baseline, FlashSAC, on state-based DMC tasks are reported in Appendix E.1.

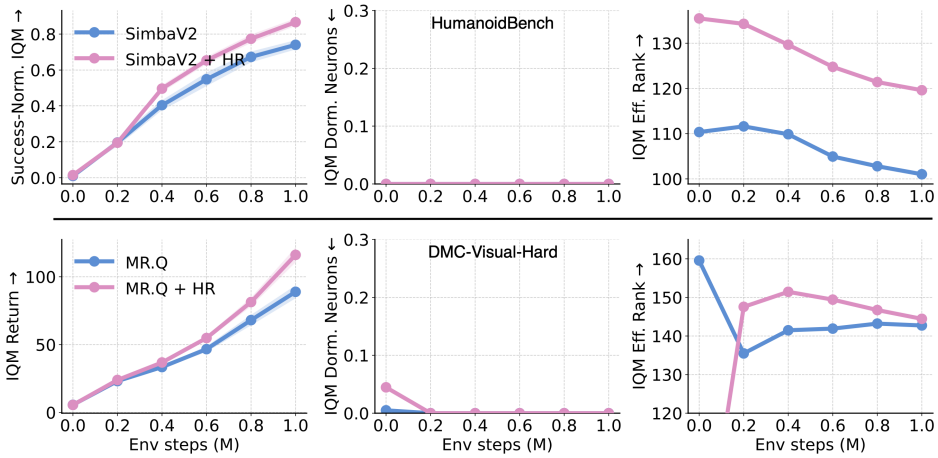


Figure 8: IQM Aggregated performance, dormant neuron fraction, and effective rank on 14 HumanoidBench tasks (top) and 7 DMC-Visual-Hard tasks (bottom). Dormant neurons are already negligible without HR in both settings, yet HR consistently improves effective rank and downstream performance, isolating multiplicative expressiveness as an independent source of gain.

Extended training on Visual Humanoid tasks. The three DMC-Visual-Humanoid tasks are particularly slow to converge, which makes the standard training budget insufficient to clearly differentiate methods. To obtain a stronger learning signal on these tasks, we extend training to $5\times$ and compare against two strong baselines: DrQ-v2 [61] and the recent state-of-the-art XQC [40] for DMC-Visual-Humanoid tasks. As shown in Figure 9, MR.Q + HR outperforms the MR.Q baseline

and performs on par with XQC, while DrQ-v2 fails to learn within the 5M-step budget. DrQ-v2 and XQC results are taken from the XQC paper.

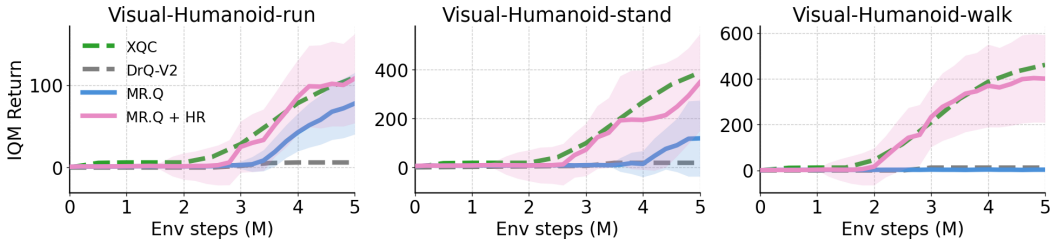


Figure 9: Evaluation on DMC-Visual-Humanoid tasks. MR.Q + HR matches XQC and substantially outperforms MR.Q and DrQ-v2. Results for DrQ-v2 and XQC are taken from [40].

5.3 Ablations

We conduct ablation experiments for HR on 8 Atari tasks (DQN), 14 HumanoidBench tasks (SimbaV2), and 7 DMC-Visual-Hard tasks (MR.Q); the mixture-of-experts and additive controls are run on DQN. A sweep over the layer at which HR is applied is reported in Appendix C.5.

Does HR just add parameters? HR maintains two parallel branches and therefore increases the parameter count of the hidden layer. To check whether the gains come from this added capacity rather than from the multiplicative structure, we compare HR to a parameter-matched wider baseline for each algorithm: we double the DQN hidden layer to match the $2\times$ weight count of $\tanh$ + HR, and analogously widen SimbaV2 and MR.Q to match their respective HR variants. Table 2 shows that naive widening does not consistently improve performance and, in the DQN case, substantially degrades it, which is also shown by recent work [37, 38]. HR, in contrast, yields gains across all three algorithms. As a stronger capacity-matched control, we additionally train DQN with a parameter-matched mixture-of-experts (MoE) layer with 4 experts [38]. MoE improves substantially on the widened variant but still falls short of HR (0.88 vs. 1.03); since widening, MoE, and HR all add comparable parameters, parameter count alone does not explain HR's benefits.

Table 2: HR versus parameter-matched alternatives across DQN, SimbaV2, and MR.Q (baseline-normalized score). HR outperforms the baseline, a parameter-matched wider layer, a parameter-matched mixture-of-experts layer (4 experts), and a parameter-matched two-branch additive layer. MoE and the additive control are evaluated on DQN (8 Atari games).

Variant	**DQN**	**SimbaV2**	**MR.Q**
Baseline	0.91	0.71	1.00
+ HR	1.03 (+13%)	0.84 (+18%)	1.17 (+17%)
+ Widen	0.49 (−46%)	0.76 (+7%)	0.97 (−3%)
+ MoE (4 experts)	0.88 (−3%)	—	—
+ Additive	0.53 (−42%)	—	—
+ 2HR	1.01 (+12%)	—	—

Is it the multiplication? HR differs from a standard hidden layer in two ways: it learns two parallel branches, and it combines them multiplicatively. To separate these, we train a parameter-matched two-branch *additive* layer, $\mathbf{z}(\mathbf{x}) = f(\mathbf{A}_1\mathbf{x} + \mathbf{b}_1) + f(\mathbf{A}_2\mathbf{x} + \mathbf{b}_2)$, with the same branch dimensions and output width as HR, on DQN. The additive variant falls behind from early training (0.30 vs. 0.67 at 5M steps) and finishes far below the product (0.53 vs. 1.03 at 10M steps, Table 2). Two parallel branches per se therefore provide no benefit; the multiplicative interaction accounts for the gain.

Does stacking HR help? We investigate whether stacking HR modules yields further gains by applying two HR blocks in DQN (2HR). Table 2 shows that 2HR performs comparably to a single HR: both improve over the baseline by a similar margin, with no additional benefit from a second HR block.

6 Conclusion and Discussion

We analyzed representational failure modes in deep RL, demonstrating that saturated $\tanh$ neurons do not merely stop learning but actively corrupt downstream layers by converting outgoing weights into fixed, silent biases. To address this, we proposed the Hadamard Representation (HR), a lightweight architectural modification that replaces standard hidden layers with the element-wise product of two independent activation branches. HR improves representational quality through two distinct mechanisms: saturation resistance, which quadratically reduces the probability of $\tanh$ collapse, and multiplicative expressiveness, which increases effective rank independently of dormancy. Evaluated across five algorithms and three diverse domains, HR consistently enhances performance, rank, and dormant rates without hyperparameter tuning, outperforming capacity-matched wider baselines.

Limitations and future work. As we maintained baseline hyperparameters throughout this work, dedicated tuning for HR remains a clear path for further gains. While we evaluated HR on a representative set of algorithms, future research could explore its integration into model-based frameworks [16, 17] or its interaction with specialized auxiliary losses [43]. More broadly, our findings indicate that the potential of non-piecewise-linear activations in RL is significantly underestimated. Extending this study to supervised and continual learning settings [1, 8] offers a promising direction for future architectural innovation; a preliminary analysis in permuted MNIST and continual RL is provided in Appendix E.2. We hope HR inspires future research on network architectural design in RL.

Acknowledgements

We would like to thank SURF (www.surf.nl) for the support in using the National Supercomputer Snellius. This work used the Dutch national e-infrastructure with the support of the SURF Cooperative.

References

[1] Z. Abbas, R. Zhao, J. Modayil, A. White, and M. C. Machado. Loss of plasticity in continual deep reinforcement learning. In S. Chandar, R. Pascanu, H. Sedghi, and D. Precup, editors, *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pages 620–636. PMLR, 22–25 Aug 2023.

[2] S. Y. Arnob, R. Ohib, S. M. Plis, and D. Precup. Single-shot pruning for offline reinforcement learning. *CoRR*, abs/2112.15579, 2021.

[3] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47:253–279, 2013.

[4] A. Bhatt, D. Palenicek, B. Belousov, M. Argus, A. Amiranashvili, T. Brox, and J. Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. *arXiv preprint arXiv:1902.05605*, 2019.

[5] J. Bjorck, C. P. Gomes, and K. Q. Weinberger. Is high variance unavoidable in RL? a case study in continuous control. In *International Conference on Learning Representations*, 2022.

[6] W. Chung, L. Cherif, D. Meger, and D. Precup. Parseval regularization for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 37:127937–127967, 2024.

[7] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR, 06–11 Aug 2017.

[8] Q. Delfosse, P. Schramowski, M. Mundt, A. Molina, and K. Kersting. Adaptive rational activations to boost deep reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.

[10] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[11] S. Fujimoto, P. D'Oro, A. Zhang, Y. Tian, and M. Rabbat. Towards general-purpose model-free reinforcement learning. *arXiv preprint arXiv:2501.16142*, 2025.

[12] M. Gallici, M. Fellows, B. Ellis, B. Pou, I. Masmitja, J. N. Foerster, and M. Martin. Simplifying deep temporal difference learning, 2024.

[13] M. Gallici, M. Fellows, B. Ellis, B. Pou, I. Masmitja, J. N. Foerster, and M. Martin. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

[14] L. Graesser, U. Evci, E. Elsen, and P. S. Castro. The state of sparse training in deep reinforcement learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7766–7792. PMLR, 17–23 Jul 2022.

[15] C. Gulcehre, S. Srinivasan, J. Sygnowski, G. Ostrovski, M. Farajtabar, M. Hoffman, R. Pascanu, and A. Doucet. An empirical study of implicit regularization in deep offline RL. *Transactions on Machine Learning Research*, 2022.

[16] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations, ICLR*, 2021.

[17] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016.

[19] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[20] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

[21] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[22] J. E. Kooi, Z. Yang, and V. François-Lavet. Hadamax encoding: Elevating performance in model-free atari. *arXiv preprint arXiv:2505.15345*, 2025.

[23] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[24] A. Kumar, R. Agarwal, D. Ghosh, and S. Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

[25] S. Kumar, H. Marklund, and B. Van Roy. Maintaining plasticity in continual learning via regenerative regularization. *arXiv preprint arXiv:2308.11958*, 2023.

[26] H. Lee, D. Hwang, D. Kim, H. Kim, J. J. Tai, K. Subramanian, P. R. Wurman, J. Choo, P. Stone, and T. Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. *arXiv preprint arXiv:2410.09754*, 2024.

[27] H. Lee, Y. Lee, T. Seno, D. Kim, P. Stone, and J. Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025.

[28] V. Liu, R. Kumaraswamy, L. Le, and M. White. The utility of sparse representations for control in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4384–4391, 2019.

[29] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis. Dying relu and initialization: Theory and numerical examples. *CoRR*, abs/1903.06733, 2019.

[30] C. Lyle, M. Rowland, and W. Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations*, 2022.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2 2015.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[33] A. Molina, P. Schramowski, and K. Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks. In *International Conference on Learning Representations*, 2019.

[34] M. Nauman, M. Ostaszewski, K. Jankowski, P. Miłoś, and M. Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

[35] E. Nikishin, J. Oh, G. Ostrovski, C. Lyle, R. Pascanu, W. Dabney, and A. Barreto. Deep reinforcement learning with plasticity injection. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 37142–37159. Curran Associates, Inc., 2023.

[36] E. Nikishin, M. Schwarzer, P. D'Oro, P.-L. Bacon, and A. Courville. The primacy bias in deep reinforcement learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16828–16847. PMLR, 17–23 Jul 2022.

[37] J. Obando-Ceron, A. Courville, and P. S. Castro. In deep reinforcement learning, a pruned network is a good network. *arXiv preprint arXiv:2402.12479*, 2024.

[38] J. S. Obando Ceron, G. Sokar, T. Willi, C. Lyle, J. Farebrother, J. N. Foerster, G. K. Dziugaite, D. Precup, and P. S. Castro. Mixtures of experts unlock parameter scaling for deep RL. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 38520–38540. PMLR, 21–27 Jul 2024.

[39] D. Palenicek, F. Vogt, and J. Peters. Scaling crossq with weight normalization. *arXiv preprint arXiv:2506.03758*, 2025.

[40] D. Palenicek, F. Vogt, J. Watson, I. Posner, and J. Peters. Xqc: Well-conditioned optimization accelerates deep reinforcement learning. *arXiv preprint arXiv:2509.25174*, 2025.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[42] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[43] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-Efficient Reinforcement Learning with Self-Predictive Representations. In *International Conference on Learning Representations, ICLR*, 2021.

[44] M. Schwarzer, J. S. O. Ceron, A. Courville, M. G. Bellemare, R. Agarwal, and P. S. Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR, 2023.

[45] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation, 2024.

[46] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[47] G. Sokar, R. Agarwal, P. S. Castro, and U. Evci. The dormant neuron phenomenon in deep reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[48] G. Sokar and P. S. Castro. Mind the gap! the challenges of scale in pixel-based deep reinforcement learning. *arXiv preprint arXiv:2505.17749*, 2025.

[49] G. Sokar, E. Mocanu, D. C. Mocanu, M. Pechenizkiy, and P. Stone. Dynamic sparse training for deep reinforcement learning. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3437–3443. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[50] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[51] R. S. Sutton, A. G. Barto, and A. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[52] Y. Tan, P. Hu, L. Pan, J. Huang, and L. Huang. Rlx2: Training a sparse deep reinforcement learning model from scratch. *CoRR*, abs/2205.15043, 2023.

[53] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

[54] D. Teney, A. M. Nicolicioiu, V. Hartmann, and E. Abbasnejad. Neural redshift: Random networks are not random functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4786–4796, June 2024.

[55] R. Trumpp, A. Schäfftlein, M. Theile, and M. Caccamo. Impoola: The power of average pooling for image-based deep reinforcement learning. *arXiv preprint arXiv:2503.05546*, 2025.

[56] F.-L. Vincent, H. Peter, I. Riashat, G. B. Marc, and P. Joelle. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[57] G. Xu, R. Zheng, Y. Liang, X. Wang, Z. Yuan, T. Ji, Y. Luo, X. Liu, J. Yuan, P. Hua, et al. Drm: Mastering visual reinforcement learning through dormant ratio minimization. *arXiv preprint arXiv:2310.19668*, 2023.

[58] L. Yang, Z. Li, M. van Leeuwen, and S. Salehkaleybar. Learning subgroups with maximum treatment effects without causal heuristics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 27565–27573, 2026.

[59] L. Yang and M. van Leeuwen. Conditional density estimation with histogram trees. *Advances in Neural Information Processing Systems*, 37:117315–117339, 2024.

[60] Z. Yang, Y. Jiang, T.-C. Chen, L. Yang, A. Wong, C. Gao, J. E. Kooi, Z. Li, J. Shi, K. Qiu, et al. Modularized reinforcement learning on llms: From mdp creation to exploration and learning. *arXiv preprint arXiv:2606.21943*, 2026.

[61] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[62] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao. Mastering atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021.

Appendix

Appendix Contents

A Impact Statement **15**

B Implementation Details **15**
B.1 Hyperparameters 15
B.2 Hadamard Implementation 15

C Experimental Details **18**
C.1 KDE Implementation Details 18
C.2 Effective Rank Implementation 18
C.3 Increasing Representation Parameters 19
C.4 Compare HR with CReLU and ReDo 20
C.5 Where to Apply HR: Layer Placement Sweep 20
C.6 Training Details 21

D Evaluation Details and Individual Task Performance **21**
D.1 DQN & PPO on 8 Atari 21
D.2 PQN on 51 Atari 21
D.3 SimbaV2 on 14 HumanoidBench Tasks 25
D.4 MR.Q on 28 DMC-Visual Tasks 27

E Additional Experiments **28**
E.1 HR on FlashSAC 28
E.2 Preliminary Continual-Learning Analysis 29

A Impact Statement

This research demonstrates that architectural advancements, such as Hadamard Representation, yield substantial performance gains in RL. By enhancing the efficiency and accessibility of AI, these innovations catalyze broader adoption and facilitate the exploration of learning systems across a wide range of real-world applications.

B Implementation Details

B.1 Hyperparameters

For DQN and PPO, we evaluate on 8 Atari environments using 5 random seeds. For the mean scores, we take the mean over the eight environments. Our baseline-normalized score is calculated with respect to the original implementation using a ReLU activation. All hyperparameters used in our experiments for DQN, PPO, and PQN follow `cleanrl` [20] and the original PQN release [12]; they are reproduced in Tables 3, 4, and 5. For continuous control baselines, we use the official codebase for Simbav2 [2] and MR.Q [3] and their default configurations without touching anything. For XQC and DrQ-v2 results in Figure 9, we obtain them from the published XQC [40] paper.

Table 3: DQN Hyperparameters

Hyperparameter	**Value**	**Description**
Learning Rate	1×10^{-4}	Learning rate for the optimizer
Discount Factor (γ)	0.99	Discount for future rewards
Replay Memory Size	1,000,000	Size of the experience replay buffer
Mini-batch Size	32	Number of samples per batch update
Target Network Update Frequency	1000	Update frequency for the target network
Initial Exploration	1.0	Initial exploration rate in ϵ-greedy
Final Exploration	0.1	Final exploration rate in ϵ-greedy
Final Exploration Frame	1,000,000	Frame number to reach final exploration
Exploration Decay Frame	1,000,000	Frames over which exploration rate decays
Action Repeat (Frame Skip)	4	Number of frames skipped per action
Reward Clipping	[-1, 1]	Range to which rewards are clipped
Input Dimension	84 x 84	Dimensions of the input image
Latent Dimension	512	Dimension of the latent representation
Input Frames	4	Number of frames used as input
Training Start Frame	80,000	Frame number to start training
Loss Function	Mean Squared Error	Loss function used for updates
Optimizer	Adam	Optimization algorithm used
Optimizer ϵ	10^{-5}	Adam Epsilon

B.2 Hadamard Implementation

The Hadamard Representation (HR) is a general architectural modification that is agnostic to the specific type of layer used. It is constructed by creating two parallel branches—each processing the same input—and computing the element-wise product of their respective non-linear activations. While the integration points vary by baseline (visualized in Figure 10), the fundamental requirement is simply an additional set of weights to form the second branch.

While any layer type (e.g., convolutional or linear) can be used, we illustrate the implementation using a standard linear layer as a representative example:

```
# hidden: input from the preceding layer (e.g., encoder output)
# Initialize two parallel layers (example using Linear)
self.branch1 = nn.Linear(input_dim, output_dim)
```

[2] SimbaV2: `https://github.com/DAVIAN-Robotics/SimbaV2`

[3] MR.Q: `https://github.com/facebookresearch/MRQ`

Table 4: PPO Hyperparameters

Hyperparameter	Value	Description
Learning Rate	2.5×10^{-4}	Learning rate for the optimizer
Discount Factor (γ)	0.99	Discount factor for future rewards
Number of Steps	128	Number of steps per environment before update
Anneal LR	True	Whether to anneal the learning rate
GAE Lambda	0.95	Lambda parameter for GAE
Number of Minibatches	4	Number of minibatches to split the data
Update Epochs	4	Number of epochs per update
Normalize Advantage	True	Whether to normalize advantage estimates
Clipping Coefficient	0.1	Clipping parameter for PPO
Clip Value Loss	True	Whether to clip value loss
Entropy Coefficient	0.01	Coefficient for entropy bonus
Value Function Coefficient	0.5	Coefficient for value function loss
Maximum Gradient Norm	0.5	Maximum norm for gradient clipping
Target KL	None	Target KL divergence between updates
Latent Dimension	512	Dimension of the latent representation
Optimizer	Adam	Optimization algorithm used
Optimizer ϵ	10^{-5}	Adam Epsilon

Table 5: PQN Hyperparameters [12]

Hyperparameter	Value	Description
Total Timesteps	10,000,000	Total timesteps for training
Timesteps for Decay	10,000,000	Timesteps for decay functions (epsilon and lr)
Number of Environments	128	Number of parallel environments
Steps per Environment	32	Steps per environment in each update
Number of Epochs	2	Number of epochs per update
Number of Minibatches	32	Number of minibatches per epoch
Starting Epsilon	1.0	Starting epsilon for exploration
Final Epsilon	0.001	Final epsilon for exploration
Epsilon Decay Ratio	0.1	Decay ratio for epsilon
Epsilon for Test Policy	0.0	Epsilon for greedy test policy
Learning Rate	0.00025	Learning rate
Learning Rate Linear Decay	True	Use linear decay for learning rate
Max Gradient Norm	10.0	Max gradient norm for clipping
Discount Factor (γ)	0.99	Discount factor for reward
Lambda (λ)	0.65	Lambda for generalized advantage estimation
Episodic Life	True	Terminate episode when life is lost
Reward Clipping	True	Clip rewards to range [-1, 1]
Frame Skip	4	Number of frames to skip
Max No-Ops on Reset	30	Max number of no-ops on reset
Test During Training	True	Run evaluation during training
Number of Test Envs	8	Number of environments used for testing

```
self.branch2 = nn.Linear(input_dim, output_dim)

# Compute parallel activations using a non-linearity (e.g., Tanh)
out1 = torch.tanh(self.branch1(hidden))
out2 = torch.tanh(self.branch2(hidden))

# Hadamard product (element-wise multiplication)
hr_output = out1 * out2
```

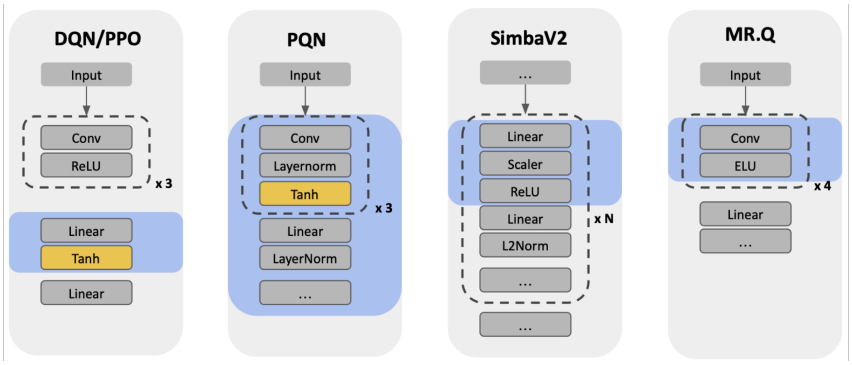


Figure 10: Architectural diagrams highlighting (in blue) the locations where the HR structure is integrated within the different baseline methods.

C Experimental Details

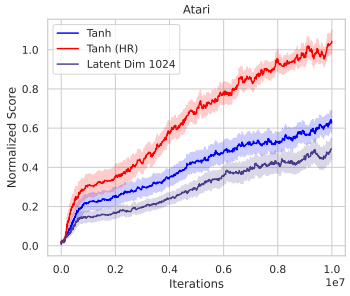
C.1 KDE Implementation Details

To stabilize KDE computation and avoid singularity issues, a small noise ϵ following a normal distribution is added to each neuron's activations: $\alpha_i' = \alpha_i + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $\sigma^2 = 1 \times 10^{-5}$. The bandwidth, crucial for accuracy, is computed using Scott's rule, adjusted by the standard deviation of the jittered activations: $bw = n^{-1/5} \cdot \text{std}(\alpha_i')$, where n is the number of samples in α_i. The density is estimated using a Gaussian kernel, $f(x) = \frac{1}{n \cdot bw} \sum_{j=1}^{n} K\left(\frac{x - \alpha_{ij}'}{bw}\right)$. A neuron is classified as dormant when $\max(f(x)) \geq \omega$. After analyzing individual KDEs, $\omega = 20$ provides a strong approximation of actual dormant neurons. Other nonparametric density estimators, such as histogram- or tree-based approaches [59], could be used in place of KDE; we found the Gaussian kernel with Scott's rule sufficient for identifying dormant neurons.

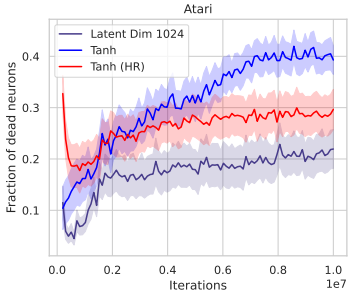
C.2 Effective Rank Implementation

```
def compute_rank_from_features(feature_matrix, rank_delta=0.01):
    sing_values = np.linalg.svd(feature_matrix, compute_uv=False)
    cumsum = np.cumsum(sing_values)
    nuclear_norm = np.sum(sing_values)
    approximate_rank_threshold = 1.0 - rank_delta
    threshold_crossed = (cumsum >= approximate_rank_threshold * nuclear_norm)
    effective_rank = sing_values.shape[0] - np.sum(threshold_crossed) + 1
    return effective_rank
```

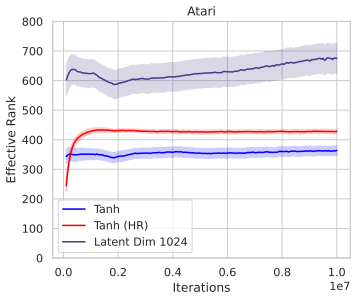
C.3 Increasing Representation Parameters



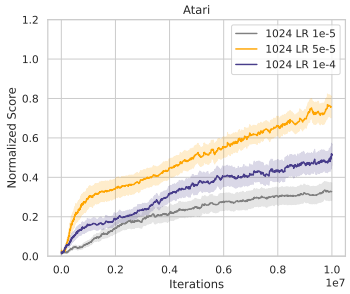
(a) Performance



(b) Dormant neurons



(c) Effective rank



(d) Learning rates, $z_t \in \mathbb{R}^{1024}$

Figure 11: Comparison of standard tanh, tanh with a higher representation dimension $z_t \in \mathbb{R}^{512 \to 1024}$, and a tanh Hadamard representation, on performance (a), fraction of dormant neurons (b), effective rank of z_t (c), and learning rates of the higher-dimensional latent (d). Increasing the representation dimension naturally raises the effective rank, but a larger representation dimension is not always preferable: it often requires different hyperparameters and can reduce performance [37, 38]. The learning-rate ablation in (d) shows that wider layers can prefer lower learning rates, but HR still significantly outperforms any of the 1024-dimensional learning-rate ablations, indicating that HR's improvement is not explained by parameter count.

C.4 Compare HR with CReLU and ReDo

We compare HR against two baselines that are proposed to prevent the loss of plasticity in RL. CReLU [1] focuses on architectural robustness by preserving both positive and negative signal components to ensure continuous gradient flow. ReDo [47], on the other hand, employs an active recycling strategy that identifies and resets dormant neurons based on an activation threshold τ.

As illustrated in Figure 12, HR provides a substantial performance improvement over both the recent PQN baseline (labeled as ReLU) and these specialized plasticity-preservation methods.

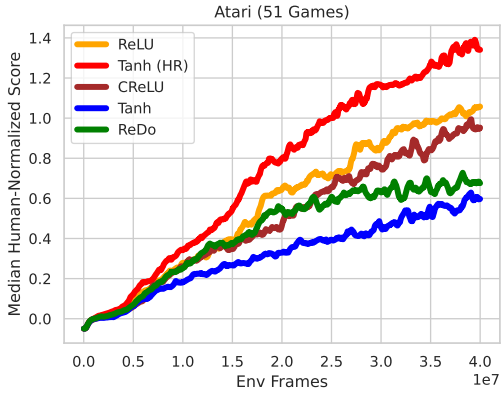


Figure 12: Median Human-Normalized scores on PQN in 51 Atari games for 5 seeds over 40M frames. Labels represent encoder activations. Employing a HR provides a significant performance improvement over the recent PQN baseline (displayed as ReLU), the CReLU [1] and ReDo (τ=0.025) [47].

C.5 Where to Apply HR: Layer Placement Sweep

Throughout the main paper, HR is applied to a single layer per architecture (Figure 10). To examine how this choice affects performance and saturation, we sweep the placement of HR over every layer of the DQN network on Amidar: the three convolutional layers (`ReLU`), the fully connected $\tanh$ representation layer, all convolutional layers jointly, and all layers jointly. All runs use 5M steps and 3 seeds, and we report the mean score over the final 1M steps. The last column reports the saturated fraction of the fully connected $\tanh$ layer at the end of training, using an explicit saturation criterion (mean $|a| \geq 0.95$ and std ≤ 0.05 across the batch).

Table 6: Placement sweep of HR on DQN (Amidar, 5M steps, 3 seeds, mean $\pm$ SE over the final 1M steps). "FC saturated" is the fraction of saturated neurons in the fully connected $\tanh$ layer at the end of training.

Placement	**Score**	**Δ vs. baseline**	**FC saturated**
HR @ fc (main paper)	266.9 ± 9.9	$+97.5$	3.2%
$\tanh$ (baseline)	169.3 ± 12.1	—	17.8%
HR @ all layers	233.0 ± 1.3	$+63.7$	2.5%
HR @ conv2	185.2 ± 5.0	$+15.8$	32.4%
HR @ conv1	179.3 ± 2.1	$+10.0$	30.1%
HR @ all conv	168.7 ± 2.3	-0.7	1.6%
HR @ conv3	165.5 ± 5.1	-3.8	13.7%

Three observations follow from Table 6. First, DQN has a single hidden fully connected layer, so the "fc" placement used in the main paper already covers all fully connected layers. Second, applying HR everywhere ($+63.7$) underperforms the fc-only placement ($+97.5$), and stacking two HR blocks (2HR, Table 2) adds nothing over a single one; one HR layer sufficed in all our settings. Third, the benefit is concentrated where the pathology lives: the $\tanh$ fully connected layer saturates strongly (17.8%) and is markedly rank-deficient (normalized effective rank 0.470), whereas the `ReLU`

convolutional layers cannot saturate at ± 1, have at most 1.6% dead units, and are close to full rank (normalized effective rank 0.875 to 0.953). HR placed in the convolutional layers therefore has little to fix, and can even worsen saturation of the downstream fully connected layer (30 to 32% for conv1/conv2). Identifying where an intervention has the largest effect is closely related to subgroup discovery for heterogeneous treatment effects [58]; our sweep is a simple exhaustive version of this over layer placement. This suggests a practical heuristic that is consistent with every placement in our experiments: apply HR once, at a bounded, rank-deficient representation layer.

C.6 Training Details

We train all experiments on a cluster of A100 GPUs. Adding HR slows down the training, roughly 1.2x.

D Evaluation Details and Individual Task Performance

D.1 DQN & PPO on 8 Atari

For DQN and PPO the Hadamard representation was applied to the final hidden layer of the Nature CNN. Performance is normalized with respect to the ReLU baseline on which the experiments were built [20]. The minimum and maximum scores of the ReLU baseline are taken for each environment, and the normalized score is

$$\text{Normalized Score} = \frac{\text{Score} - \text{Min Score}}{\text{Max Score} - \text{Min Score}}, \tag{5}$$

where *Min Score* and *Max Score* are the lowest and highest scores recorded by the ReLU baseline (the lowest is typically the random-policy score). To average, we sum the normalized scores for every run and take the mean. The Human-Normalized Score [31] is computed analogously using human and random performance:

$$\text{Human-Normalized Score} = \frac{\text{Score} - \text{Random Score}}{\text{Human Score} - \text{Random Score}}. \tag{6}$$

Computing performance under this normalization gives Fig. 13. Because we evaluate on a subset of Atari for DQN and PPO, the VideoPinball environment dominates the Human-Normalized Score; for fairer comparison we therefore use baseline-normalized scores in the main paper.

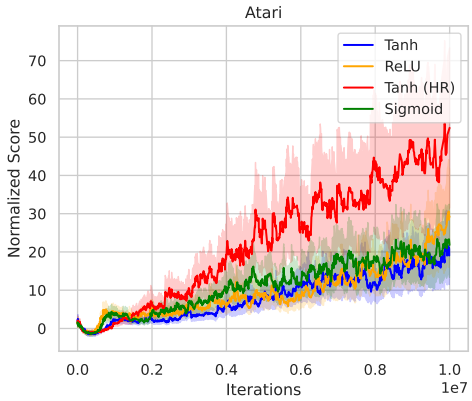


Figure 13: Human-Normalized performance (in multiples) with the standard deviation over the means in the Atari domain for 10M iterations (40M frames).

D.2 PQN on 51 Atari

We use Median Human-Normalized Score for PQN. For each game, compute the average score x_i across multiple independent seeds. Then compute the normalized score Z_i as:

$$Z_i = \frac{x_i - r_i}{h_i - r_i} \tag{7}$$

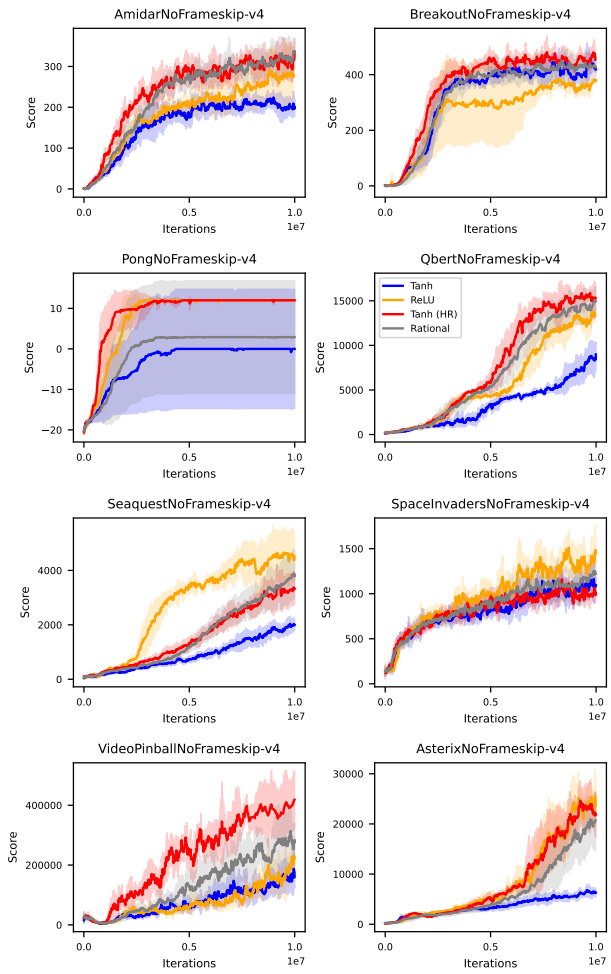


Figure 14: DQN per-environment performance. Lines are means over 5 seeds, shaded regions show standard deviation.

where x_i is the raw score, and r_i and h_i are the random and human scores for game i (we take these values from [22]), respectively. After computing the normalized scores for all 51 games * seeds, they are sorted and the median value is computed.

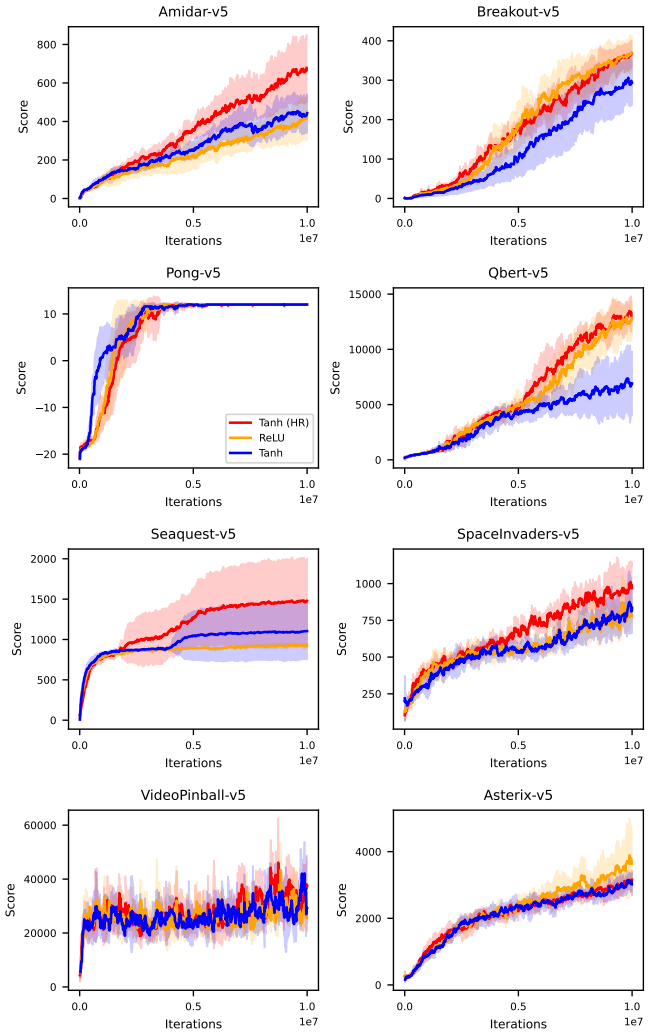


Figure 15: PPO per-environment performance. Lines are means over 5 seeds, shaded regions show standard deviation.

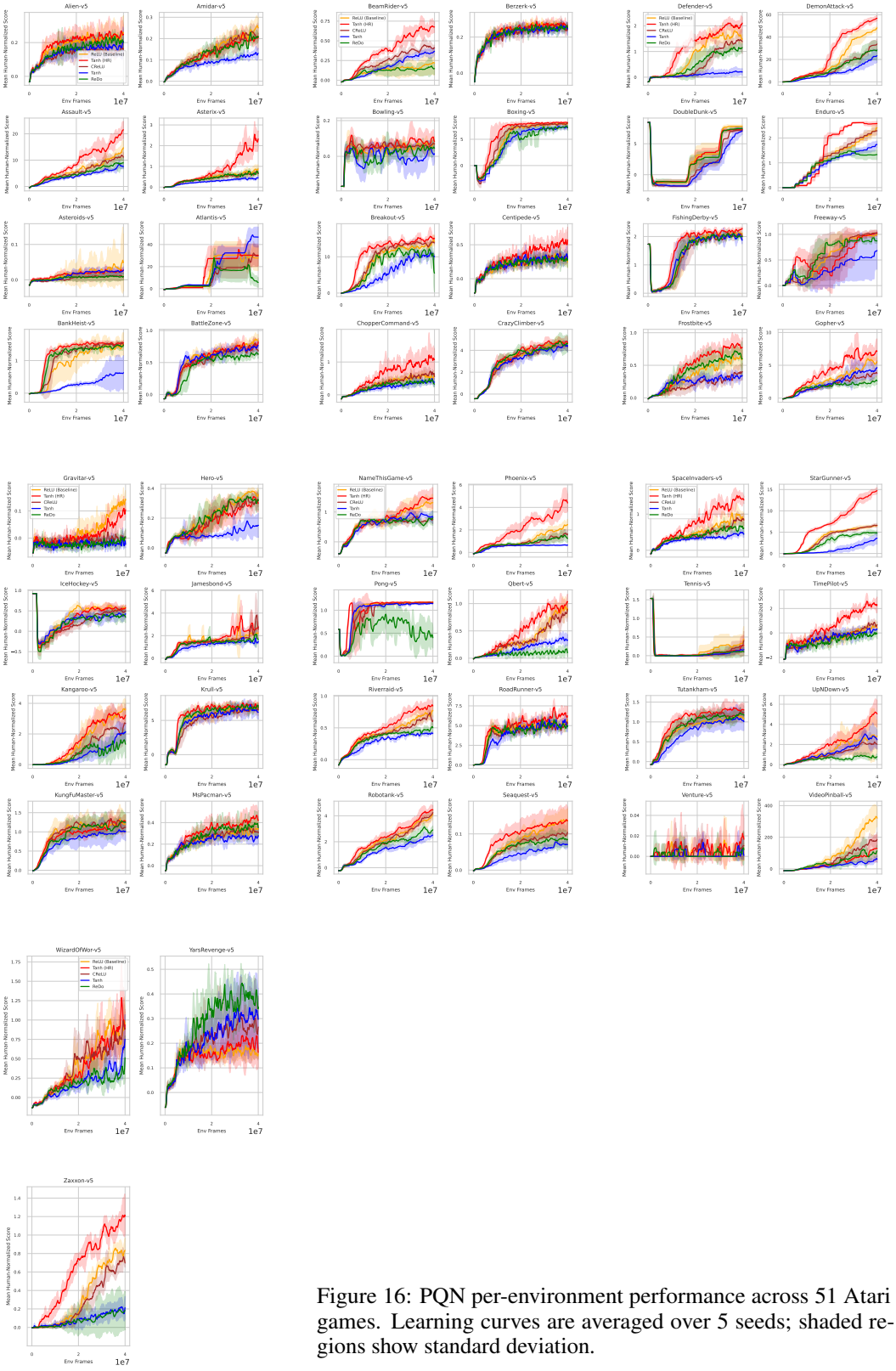


Figure 16: PQN per-environment performance across 51 Atari games. Learning curves are averaged over 5 seeds; shaded regions show standard deviation.

D.3 SimbaV2 on 14 HumanoidBench Tasks

For each HumanoidBench task t, we report a success-normalized score that linearly rescales the raw episodic return so that a random policy receives 0 and the task's success threshold receives 1:

$$\tilde{r}_t = \frac{r_t - r_t^{\text{random}}}{r_t^{\text{target}} - r_t^{\text{random}}} \tag{8}$$

where r_t is the agent's mean evaluation return on task t over the final 5% of training, r_t^{random} is the average return of a uniform random policy, and r_t^{target} is the per-task success threshold defined by HumanoidBench [45].

An episode is considered successful by the benchmark whenever its return exceeds r_t^{target}. Thus, $\tilde{r}_t = 1$ corresponds to consistently solving the task, while $\tilde{r}_t > 1$ indicates super-target performance. The per-task values of r_t^{random} and r_t^{target} are listed in Table 7.

To summarize across the 14 HumanoidBench tasks, we report the interquartile mean (IQM) of $\{\tilde{r}_t\}$, i.e., the mean after discarding the bottom and top 25% of (task, seed) scores. Per-task curves are plotted as the mean over seeds with shaded standard error.

Table 7: Per-task random and success-target scores for HumanoidBench used in Eq. (8).

Task	Random	Target
`h1-balance-simple`	9.39	800
`h1-balance-hard`	9.04	800
`h1-crawl`	272.66	700
`h1-hurdle`	2.21	700
`h1-maze`	106.44	1200
`h1-pole`	20.09	700
`h1-reach`	260.30	12000
`h1-run`	2.02	700
`h1-sit-simple`	9.39	750
`h1-sit-hard`	2.45	750
`h1-slide`	3.19	700
`h1-stair`	3.11	700
`h1-stand`	10.55	800
`h1-walk`	2.38	700

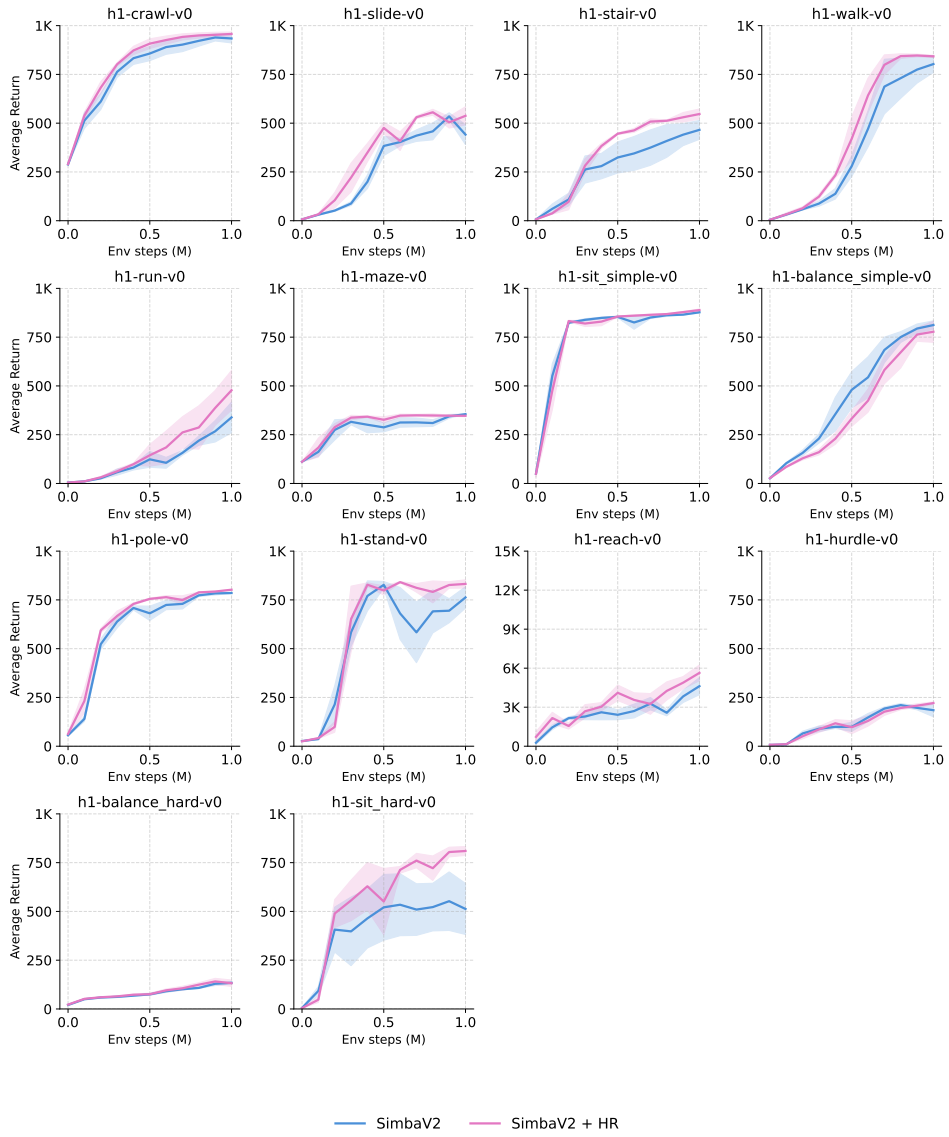


Figure 17: Performance on 14 HumanoidBench tasks. Learning curves are averaged over 3 seeds; shaded regions show standard error.

D.4 MR.Q on 28 DMC-Visual Tasks

DMC-Visual includes 28 tasks in total, 7 of them is categorized as **Hard**, i.e. 4 dog tasks and 3 humanoid tasks (see the last 2 rows in Figure 18).

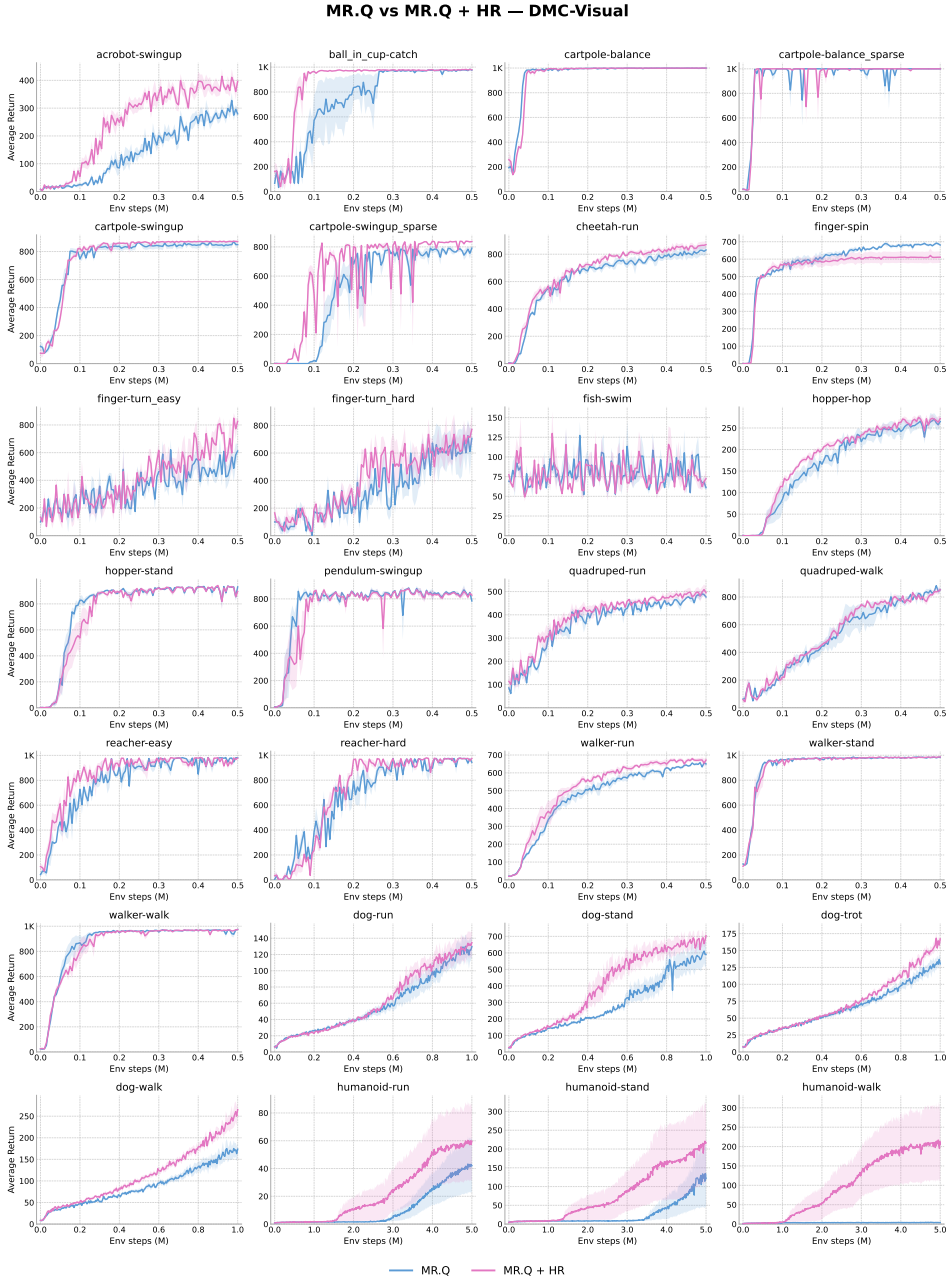


Figure 18: Performance on 28 DMC-Visual tasks. Learning curves are averaged over 3–5 seeds; shaded regions show standard error.

E Additional Experiments

E.1 HR on FlashSAC

To test HR on a high-throughput off-policy pipeline for high-dimensional state-based control, we integrate it into FlashSAC [4] using the official codebase with unchanged hyperparameters. Only the encoder block changes: the single inverted-bottleneck stream is replaced by two half-width branches whose outputs are multiplied element-wise. This integration is parameter-*reducing*: the HR encoder has 99,584 parameters versus 132,352 for the baseline ($\sim 0.75\times$), so any gain cannot come from added capacity.

DMC-Dog tasks. We evaluate on the four state-based DMC dog tasks for 1M steps with 5 seeds (`[0, 1000, 2000, 3000, 4000]`). We report mean $\pm$ standard error at 1M steps under two protocols: a windowed average for robustness (average over 900k–1M steps) and a single-checkpoint value, as used in the official FlashSAC benchmark. Both are shown in Table 8 as "window / point".

Table 8: FlashSAC and FlashSAC + HR on four state-based DMC dog tasks at 1M steps (5 seeds, mean $\pm$ SE). Each cell reports windowed average / single-checkpoint value.

Task	FlashSAC	FlashSAC + HR
dog-stand	987.0 ± 2.8 / 988.4 ± 2.4	988.3 ± 0.8 / 989.2 ± 1.7
dog-walk	954.2 ± 5.1 / 954.7 ± 5.4	945.0 ± 4.9 / 947.8 ± 5.0
dog-run	666.7 ± 35.6 / 681.6 ± 39.8	$\mathbf{740.8 \pm 27.8}$ / $\mathbf{753.9 \pm 27.9}$
dog-trot	858.5 ± 41.0 / 877.5 ± 32.6	$\mathbf{892.7 \pm 34.7}$ / $\mathbf{906.2 \pm 22.7}$

On the two easier tasks (dog-stand, dog-walk), both methods saturate. On the two harder tasks (dog-run, dog-trot), HR improves final return ($+11\%$ and $+4\%$ under the windowed protocol), with lower seed variance on dog-run; HR also learns faster, reaching higher returns at 500k steps on both tasks. Our reproduced FlashSAC baseline matches the officially reported single-checkpoint numbers within a few points at 1M steps on all four tasks.

DMC Humanoid-run. We additionally report state-based humanoid-run with 5 seeds under the same protocol (Table 9). With the default parameter-reducing integration (each branch at half the baseline stream's width, $0.76\times$ total parameters), HR underperforms the baseline. Matching each branch to the full baseline stream width (HR-large, $1.48\times$ total parameters) recovers baseline performance at 1M steps (365.1 vs. 354.9). Performance on humanoid-run is thus sensitive to network capacity, and whether HR can be further improved on this task with appropriate scaling remains open; we leave a systematic investigation to future work.

Table 9: FlashSAC variants on state-based DMC humanoid-run at 1M steps (5 seeds, mean $\pm$ SE; windowed average / single-checkpoint value).

Agent	Params	Branch width	Return @1M
FlashSAC	2.48M ($1.00\times$)	—	342.0 ± 56.2 / 354.9 ± 61.4
FlashSAC + HR	1.89M ($0.76\times$)	$h/2$ per branch	202.3 ± 52.2 / 214.4 ± 55.9
FlashSAC + HR-large	3.67M ($1.48\times$)	h per branch	353.4 ± 54.8 / 365.1 ± 61.6

Dormant neurons. We also measured the dormant-neuron fraction for FlashSAC and FlashSAC + HR across the four dog tasks and humanoid-run. In both methods the dormant fraction remains low throughout training and is essentially 0% for the critic. For the actor, the average dormant fraction is approximately 2.1% for FlashSAC + HR versus 2.7% for FlashSAC, i.e., marginally lower with HR. This is consistent with the low-dormancy regime of Section 5.2, where multiplicative expressiveness rather than saturation resistance is the operative mechanism.

[4] FlashSAC: `https://github.com/Holiday-Robot/FlashSAC`

E.2 Preliminary Continual-Learning Analysis

HR is proposed and evaluated for standard (single-task) RL. Because dormancy and rank collapse are also central to loss of plasticity in continual learning, we report a preliminary analysis in two continual settings. In both, HR uses no tuning, as throughout the paper.

Permuted MNIST. We train an MLP (784-128-128-10, $\tanh$ activations) on a sequence of 100 permuted-MNIST tasks, one epoch per task, with Adam (learning rate 10^{-3}, batch size 256), persistent optimizer state, no resets or replay, and an identical permutation sequence across methods (5 seeds). We compare plain $\tanh$, $\tanh$ + HR, and L2-init [25], a regularizer designed for this setting, whose coefficient λ was selected via a sweep over $\{10^{-2}, 10^{-3}, 10^{-4}\}$ on a 20-task run (best: 10^{-3}). We report accuracy on the current task, i.e., plasticity rather than retention.

Table 10: Permuted MNIST, 100 tasks, 5 seeds (mean $\pm$ SE). Task-0 accuracy, mean accuracy over the last 5 tasks, and the change between them.

Method	**Task-0 acc. (%)**	**Last-5 acc. (%)**	**Δ (pp)**
$\tanh$	93.07 ± 0.25	89.60 ± 0.16	-3.46
$\tanh$ + HR	94.90 ± 0.47	92.02 ± 0.19	-2.88
L2-init	92.68 ± 0.36	93.58 ± 0.16	$+0.90$

HR is above the $\tanh$ baseline at every task index ($+2.4$pp at task 100) and declines more slowly (-2.88 vs. -3.46pp), but does not eliminate the decline; L2-init, purpose-built for this setting, stays flat and finishes highest.

Continual RL. We evaluate HR on the continual RL benchmark of Chung et al. [6], using their released code, protocol, and metric. The environment changes at fixed intervals within a single lifetime (goal location in Grid-World, physical parameters in CartPole and LunarLander, and the manipulation task in Meta-World), and a single agent trains through the sequence without resets. Following their Figure 4, we report $\Pr(> \tau)$, the fraction of runs whose normalized return exceeds τ.

Table 11: Continual RL on the benchmark of Chung et al. [6]. Entries are the fraction of runs whose normalized return exceeds the threshold.

Environment	**Metric**	**Base**	**Parseval** [6]	**HR (this work)**
Grid-World	$\Pr(> 0.9)$	0.32	**0.78**	0.54
CartPole	$\Pr(> 0.9)$	0.27	**0.41**	0.39
LunarLander	$\Pr(R > 200)$	0.21	0.29	**0.32**
Meta-World	$\Pr(> 0.9)$	0.43	**0.59**	0.53

HR improves over the base agent on all four environments, but underperforms Parseval regularization on three of four (Grid-World, CartPole, Meta-World), leading only on LunarLander. Parseval, proposed for this setting, remains the strongest method here.

Summary. The two continual settings are consistent: HR improves over the baseline but does not match the method purpose-built for each setting (Parseval in continual RL, L2-init on permuted MNIST), whereas in the standard RL setting HR is stronger than the plasticity baselines we compared against (Section C.4). HR lowers the probability of saturation but does not eliminate it, so settings with severe, prolonged plasticity loss may require combining HR with dedicated continual-learning methods. A thorough continual-learning study is left for future work.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are directly supported by the results and discussions presented in Section 5 and Section 6.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our approach, including hyperparameters tuning and boarder baselines, in Section 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: All theoretical assumptions and proofs are explicitly stated Section 4.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: All details required to reproduce our main findings, including methodology and evaluation metrics, are detailed in Section 5 and Section D.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We have provided our source code and data access instructions in the supplementary material.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Complete experimental details, including hyperparameters and training regimes, are provided in Section D and Section B.1.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report IQM/median and standard deviation/error for all main experiments based on 3–5 independent runs.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Information regarding hardware and total training time is provided in Section C.6.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research adheres to the NeurIPS Code of Ethics as outlined in the conference guidelines.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: A discussion on the potential societal impacts of our work is included in Section A.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Our research does not involve models or datasets categorized as high-risk for misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All codebase and data used are cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

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

13. **New assets**

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

    Answer: [N/A]

    Justification: No new datasets or software assets were introduced in this work.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This study did not involve crowdsourcing or human subject research.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work did not involve human subjects and therefore did not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were not used as a core component of the research methodology.

    Guidelines:

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