Mind the GAP! The Challenges of Scale in Pixel-based Deep Reinforcement Learning

Ghada Sokar
Google DeepMind
gsokar@google.com

Pablo Samuel Castro
Google DeepMind
psc@google.com

Abstract

Scaling deep reinforcement learning in pixel-based environments presents a significant challenge, often resulting in diminished performance. While recent works have proposed algorithmic and architectural approaches to address this, the underlying cause of the performance drop remains unclear. In this paper, we identify the connection between the output of the encoder (a stack of convolutional layers) and the ensuing dense layers as the main underlying factor limiting scaling capabilities; we denote this connection as the **bottleneck**, and we demonstrate that previous approaches implicitly target this bottleneck. As a result of our analyses, we present Global Average Pooling (GAP) as a simple yet effective way of targeting the bottleneck, thereby avoiding the complexity of earlier approaches.

1 Introduction

Reinforcement Learning (RL) is widely considered one of the most effective approaches for complex sequential decision-making problems [Mnih et al., 2015, Vinyals et al., 2019, Bellemare et al., 2020, Degrave et al., 2022, Wurman et al., 2022], in particular when combined with deep neural networks (typically referred to as deep RL). In contrast to the so-called "scaling laws" observed in supervised learning (where larger networks typically result in improved performance) [Kaplan et al., 2020], it is difficult to scale RL networks without sacrificing performance. There has been a recent line of work aimed at developing techniques for effectively scaling value-based networks, such as via the use of mixtures-of-experts [Obando Ceron* et al., 2024], network pruning [Obando Ceron et al., 2024], tokenization [Sokar et al., 2025], and regularization [Nauman et al., 2024]. Most of these techniques tend to focus on structural modifications to standard deep RL networks by leveraging sparse-network training techniques from the supervised learning literature.

Obando Ceron* et al. [2024] first demonstrated that naïvely scaling the penultimate (dense) layer in an RL network results in *decreased* performance, and proposed the use of soft mixtures-of-experts [SoftMoEs; Puigcerver et al., 2024] to enable improved performance from this form of scaling. Sokar et al. [2025] argued that the gains from SoftMoEs were mostly due to the use of tokenization. Relatedly, Obando Ceron et al. [2024] demonstrated that naïvely scaling the convolutional layers hurts performance, and showed that incremental parameter pruning yields gains that grow with the size of the original, unpruned, network. While effective, all these methods are non-trivial to implement and can result in increased computational costs.

One unifying aspect of the aforementioned works is that they tend to be most effective on networks that process pixel inputs, such as when training on the Arcade Learning Environment (ALE) [Bellemare et al., 2013]. These networks are typically divided into an *encoder* ϕ consisting of a series of convolutional layers, followed by a series of dense layers ψ; thus, for an input x, the network output is given by $\psi(\phi(x))$. Obando Ceron* et al. [2024] and Sokar et al. [2025] scaled the first layer of ψ while Obando Ceron et al. [2024] scaled all layers in ϕ. It is worth highlighting that $\psi \circ \phi$ is, in practice, a set of weights connecting the output of $\phi(x)$ with the input layer of ψ.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

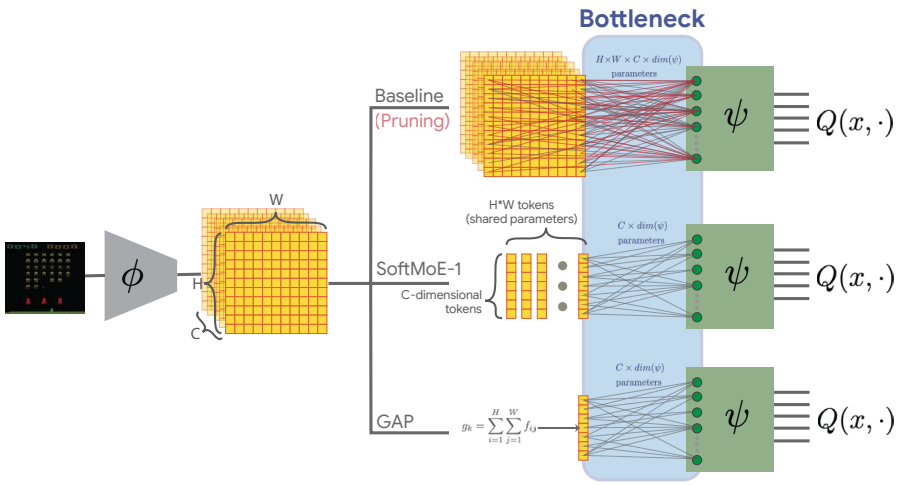

Figure 1: Illustration of the bottleneck in pixel-based networks. Standard dense networks (**Baseline**) connect all ϕ outputs with ψ, resulting in $H \times W \times C \times dim(\psi)$ parameters (scaled down when using **pruning**, shown in red). **SoftMoE-1** converts ϕ's outputs into $H \times W$ tokens of dimension C; the sharing of learned parameters across tokens results in a bottleneck with $C \times dim(\psi)$ parameters. **GAP** performs average pooling across $H \times W$ spatial dimensions, resulting in C feature maps and $C \times dim(\psi)$ parameters in the bottleneck.

In this work, we argue that the underlying cause behind the effectiveness of the aforementioned methods is that they result in a **bottleneck** between $\phi(x)$ and ψ (see Figure 1). As we will argue, performance gains are mostly due to a well-structured bottleneck, rather than the recently proposed architectural modifications to ψ. This insight suggests that simpler architectural interventions may be just as effective, which we demonstrate by using Global Average Pooling (GAP) as an effective mechanism for enabling improved performance from scaling.

Our contributions can be summarized as follows: (i) We investigate the challenges leading to the performance degradation of scaled RL networks; (ii) we study the underlying reasons behind the success of existing architectural approaches in scaling; and (iii) we present pooling as a faster, simpler method yielding superior performance. We begin in Section 2 by providing the necessary background on reinforcement learning, detailing the architecture employed in pixel-based deep RL, and outlining architectural modifications proposed by recent methods for addressing scaling challenges. Section 3.1 then describes our experimental setup. Our analyses investigating the difficulties of scaling RL networks and how existing methods attempt to address them are presented in Sections 3.2 and 3.3, respectively. Section 4 is dedicated to our results and analysis on RL networks with GAP. Finally, Section 5 covers the related work, followed by a conclusion and discussion in Section 6.

2 Preliminaries

2.1 Reinforcement learning

Reinforcement learning involves an agent moving through a series of states $x \in \mathcal{X}$ by selecting an action $a \in \mathcal{A}$ at discrete timesteps. After selecting action a_t from state x_t, the agent will receive a reward $r_t(x_t, a_t)$, and its goal is to maximize the discounted sum of cumulative rewards $\sum_{t=0}^{\infty} \gamma^t r_t$, where $\gamma \in [0, 1)$ by finding an optimal *policy* $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ which quantifies the agent's behavior at each state. Value-based methods [Sutton and Barto, 1998] maintain estimates of the value of selecting action a from state x and following π afterwards: $Q(x, a) := \mathbb{E}\left[\sum_{t=0}^{\infty} [\gamma^t r_t(x_t, a_t) | x_0 = x, a_0 = a, a_t \sim \pi(x_t)]\right]$, where π is induced from Q, for instance with the use of softmax: $\pi(x)(a) := \frac{e^{Q(x,a)}}{\sum_{a' \in \mathcal{A}} e^{Q(x,a')}}$.

Mnih et al. [2015] demonstrated deep neural networks can be very effective at approximating Q-values, even for complex domains such as Atari games [Bellemare et al., 2013]; their network has served as the backbone for most deep RL networks. Later, Espeholt et al. [2018] proposed

using ResNet based architecture which demonstrates significant performance improvements over the original convolutional neural network (CNN) architecture. For pixel-based environments, this family of networks consists of a set of convolutional layers [Fukushima, 1980], which we will collectively refer to as ϕ (and often referred to as the *encoder* or *representation*), followed by a set of dense layers, which we will collectively refer to as ψ. Thus given an input x, the network approximates the Q-values as $\tilde{Q}(x, \cdot) = \psi(\phi(x))$.

The output of ϕ is a 3-dimensional tensor $H \times W \times C$, where H is height, W is width, and C is the number of feature maps (channels); this output is typically flattened before being fed to ψ. Thus, the number of parameters for the functional composition $\psi \circ \phi$ is equal to $H \times W \times C \times dim(\psi)$, where $dim(\psi)$ is the dimensionality of the first dense layer in ψ. We refer to connection between ϕ and ψ as the **bottleneck**, and it will be the focus of most of our work. See Figure 1 for an illustration.

2.2 Network scaling

As we will demonstrate below, this bottleneck has a direct impact on learning efficiency. The standard approach is to flatten the output of ϕ into a single vector before feeding it to ψ, which we refer to as an 'unstructured' representation. This unstructured representation risks diluting the spatial structure captured by ϕ, making learning more difficult. This is exacerbated when scaling the width of ψ, as demonstrated in prior works [Obando Ceron* et al., 2024, Sokar et al., 2025]. We hypothesize that the effectiveness of prior architectural modifications for scaling RL networks lies in their ability to induce some form of structure at this bottleneck. The rest of this section details these specific modifications.

2.2.1 SoftMoE

To address scaling limitations, Obando Ceron* et al. [2024] proposed a notable architectural change: replacing the standard dense layer in ψ with a mixture-of-experts (MoEs). This required first restructuring the 3-dimensional output of the encoder (ϕ) into a set of tokens to be fed into the mixture. Their investigation into tokenization strategies concluded that forming $H \times W$ tokens, each with C dimensions (the channel depth), yielded the best performance. This restructuring effectively changes the connection, reducing the bottleneck's parameter count to $C \times dim(\psi)$ (See Figure 1). This architectural change has proven highly effective for various agents and at different scales.

Building on this, Sokar et al. [2025] isolated the source of these performance gains. They demonstrated that the tokenization step itself—not the MoE architecture—was the most critical component. Their evidence showed that a single expert model (SoftMoE-1), which retains the tokenization but omits the expert routing, achieves performance nearly identical to that of the full multi-expert model across different scales.

2.2.2 Sparse networks

Sparse methods offer an alternative approach to managing network complexity. Sokar et al. [2022] demonstrated that using sparse neural networks in place of dense ones can increase learning speed and lead to improved performance. Following this, Graesser et al. [2022] studied various ways to induce this sparsity, such as by pruning network weights during training or by using networks that are sparse from scratch. By imposing sparsity, these techniques directly structure the bottleneck, reducing the density of its connections. With a given sparsity level s, the effective number of parameters in the bottleneck is reduced to $s \times H \times W \times C \times dim(\psi)$. These sparse approaches have shown promise for the scaling of larger architectures; indeed, Obando Ceron et al. [2024] recently demonstrated that pruning is an effective approach that facilitates the scaling of networks.

Gradual pruning This strategy begins with a standard, fully-dense network. As training progresses, connections (parameters) with low magnitudes—which are considered less salient to the network's function—are progressively removed. This pruning process often follows a polynomial schedule [Zhu and Gupta, 2017]. Once the pruning schedule concludes, the network achieves its target sparsity level, and this fixed sparse architecture is maintained for the remainder of training.

Sparse from scratch In contrast to gradual pruning, this approach defines a sparse network at initialization, and this specific sparsity level is maintained throughout training. The sparse

topology can be *static*, meaning the set of active connections is fixed for the entire training duration. Alternatively, the topology can be *dynamic*. For example, the *RigL* method [Evci et al., 2020] dynamically optimizes the connections by periodically pruning a portion of the weights and growing new ones elsewhere, effectively "rewiring" the network while maintaining the overall sparsity.

3 Analyses

Main hypothesis

A low-density and well-structured **bottleneck** enables scaling deep RL networks.

We conduct a series of analyses, both quantitative and qualitative, to provide evidence for our main hypothesis. We investigate impacts on performance, plasticity, and properties of the learned features. Given the effectiveness of SoftMoEs [Obando Ceron* et al., 2024], Pruning [Obando Ceron et al., 2024], and Tokenization [Sokar et al., 2025] for scaling value-based networks, our analyses in this section will focus on these.

3.1 Experimental setup

Architectures We employ the Impala architecture [Espeholt et al., 2018] for its superior performance, and while our analysis centers on it, we also confirm our findings' broad applicability using the standard CNN architecture [Mnih et al., 2015]. Our main experiments and analyses present results across various scaling factors for the width of ψ (specifically, $\times 1$, $\times 2$, $\times 4$, and $\times 8$). When a scaling factor is not explicitly mentioned, our analysis defaults to the $\times 4$ scale. This is because, as shown in prior work [Obando Ceron* et al., 2024, Sokar et al., 2025, Obando Ceron et al., 2024], the $\times 4$ scale provides a substantial performance improvement, while further scaling tends to yield only marginal gains. Although layer width is our primary focus, we also include some preliminary investigations into scaling network depth in the appendix.

Agents and environments Our primary experimental setup involves the Rainbow agent [Hessel et al., 2018] evaluated on the Arcade Learning Environment (ALE) suite [Bellemare et al., 2013]. For direct comparison with recent work, we use the same 20-game subset from Obando Ceron* et al. [2024] and Sokar et al. [2025]. However, we present our main results on the full ALE suite of 60 Atari games. We ran each experiment for a total of 200 million environment steps, with results averaged over 5 independent seeds, except for the experiments with an increased replay ratio of 2, where we report the results at 50M steps. To further test the generality of our approach on discrete tasks, we also evaluate Rainbow on the Procgen benchmark [Cobbe et al., 2019]. Additionally, we assess performance on the data-efficient 100k benchmark [Łukasz Kaiser et al., 2020], for which we use DER, a version of Rainbow specifically tuned for that data-constrained setting. Finally, to demonstrate our findings extend beyond discrete action spaces, we also evaluate the Soft Actor-Critic (SAC) agent [Haarnoja et al., 2018] on continuous control tasks from the DeepMind Control (DMC) suite [Tassa et al., 2018].

Code and compute resources For all our experiments, we use the Dopamine library[1] with Jax implementations [Castro et al., 2018]. For SoftMoE [Obando Ceron* et al., 2024], we use the official implementation integrated in Dopamine. For the sparse methods [Obando Ceron et al., 2024, Graesser et al., 2022], we use the same JaxPruner library[2] [Lee et al., 2024] used by [Obando Ceron et al., 2024]. All libraries have Apache-2.0 license. All experiments were run on NVIDIA Tesla P100 GPUs. The duration of each experiment ranged from 4 to 13 days, depending on the specific scale and algorithm. We present the exact run time for each case in Section 4.

Implementation details For all algorithms, we use the default hyperparameters in the Dopamine library. For sparse-training algorithms, we follow [Graesser et al., 2022] and use 90% sparsity. Additional analysis covering other sparsity levels is included in the appendix. For gradual pruning, we start pruning at 8M environment steps and stop at 160M (80% into training), following the schedules

[1]Dopamine: https://github.com/google/dopamine
[2]JaxPruner: https://github.com/google-research/jaxpruner

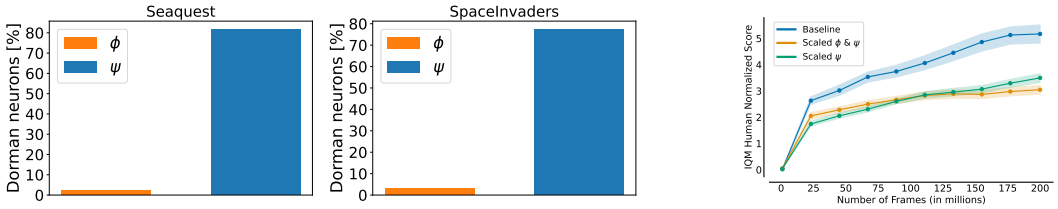

Figure 2: **(Left)** Distribution of dormant neurons across ϕ and ψ in scaled baseline across different games at the end of training. The fully connected layer exhibits the highest percentage of dormancy. **(Right)** The performance degradation associated with scaling the entire network architecture is comparable to that observed when only the bottleneck is scaled. The performance is aggregated over 20 games.

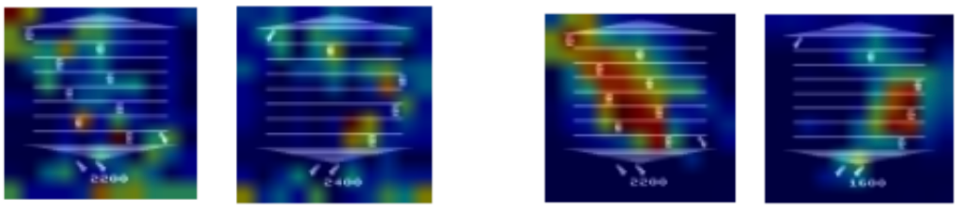

Figure 3: **GAP helps improve attention to relevant areas of input.** Visualizing influential regions for network decisions using Grad-CAM [Selvaraju et al., 2017]. **(Left)** The scaled baseline fails to attend to the important regions, focusing on irrelevant background details. **(Right)** GAP attends to the important regions in the input.

recommended by Graesser et al. [2022] and Obando Ceron et al. [2024]. For RigL [Evci et al., 2020], we use a drop fraction of 20% and a connection update interval of 5000. For SoftMoE, we use the single-expert variant (SoftMoE-1). This choice was twofold: first, it has been shown to yield performance comparable to the standard multi-expert SoftMoE [Sokar et al., 2025], and second, it ensures a more direct architectural comparison with the other methods considered in this study. We report interquartile mean (IQM) with 95% stratified bootstrap confidence intervals as recommended in [Agarwal et al., 2021]. Full experimental details are provided in the appendix.

3.2 Why scaling deep RL networks hurts performance

As has been previously demonstrated, naïvely scaling networks deteriorates performance [Obando Ceron* et al., 2024, Obando Ceron et al., 2024, Nauman et al., 2024]. In this section we conduct a series of experiments to diagnose the underlying causes for this difficulty.

We start by analyzing the training dynamics of a network where the width of all layers in both ϕ and ψ are uniformly scaled by a factor of 4. We examine neuron activity by measuring the fraction of dormant neurons, a metric that serves as a key indicator of network plasticity. Following Sokar et al. [2023], a neuron is considered "dormant" if its average activation falls below a certain threshold. As shown in the left plot of Figure 2, we find that ψ exhibits a large fraction of dormant neurons, while ϕ has low dormancy rates. This suggests that **scaling mostly affects the plasticity of the bottleneck**.

In the right plot of Figure 2, we compare the performance of the baseline network against the same network with scaled ϕ and ψ. As consistently observed in previous studies, the scaled network exhibits a degradation in performance. To investigate the contribution of ψ in this performance decrease, we only scale the bottleneck (via the first layer of ψ) to match the parameter count of the fully scaled network. As can be observed, the two scaled models has comparable performance, suggesting that **the bottleneck drives most of the performance degradation when scaling**.

To interpret the quality of the learned features of the scaled network, we generate saliency maps for the areas that have the greatest impact on the scaled network's output using Grad-CAM [Selvaraju et al., 2017]. Figure 3 (left) reveals that naïvely scaled networks fail to focus on important regions and focus on irrelevant background areas. This suggests that **scaling the bottleneck impairs a network's ability to process and learn effective combinations of the representation** ϕ.

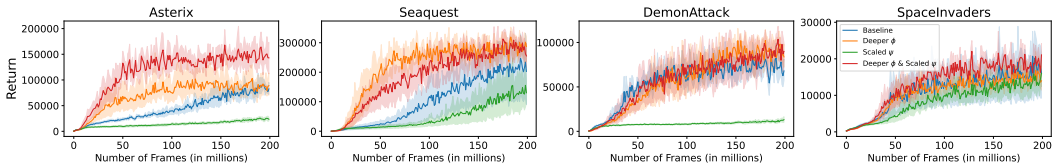

Figure 4: Scaling ψ hinders learning effective combinations of encoder's features, leading to significant performance drop. However, performance dramatically improves when the scaled ψ is fed with higher-level, more abstract features obtain by increasing the depth of ϕ.

To further investigate potential challenges in feature learning within ψ, we study the network behavior when providing more abstract, higher-level features to ψ. Specifically, we increased the depth of ϕ by adding four additional ResNet blocks, while using the scaled bottleneck. This modification makes ψ receive more structured, high-level features from the encoder. We present the performance throughout training in Figure 4. The fact that we observe a dramatic increase in performance when compared to the network that only scaled ψ suggests that **structured representations helps feature learning in scaled networks.**

3.3 Existing techniques are mainly targetting the bottleneck

As previously mentioned, there have been a number of recent proposals to enable scaling deep RL networks by using sophisticated architectural modifications. We hypothesize that a core reason for their effectiveness is that they are implicitly targetting the bottleneck. The strong performance of SoftMoE-1 and tokenized baselines presented by Sokar et al. [2025] support this claim, as they are primarily re-structuring the encoder output.

To validate our hypothesis on sparse methods, we evaluated the impact of sparse training techniques when limited to the bottleneck, while keeping all other layers dense. We performed this analysis on gradual pruning [Obando Ceron et al., 2024], dynamic sparsity (RigL) and static sparsity [Graesser et al., 2022, Sokar et al., 2022]. As shown in the top plot of Figure 5, restricting sparsification to the scaled bottleneck results in improved performance across all sparse training techniques. This suggests that **applying sparsity only to the bottleneck is sufficient to enable scaling RL networks**.

In addition to structuring the encoder output, existing techniques effectively reduce the density of the bottleneck by either structuring the output as tokens (as in SoftMoE) which results in a lower dimensional input to ψ, or explicitly masking the majority of input weights (as in sparse methods). The bottom plot of Figure 5 confirms this reduction in parameters, and illustrates a positive correlation between scale and performance, which is notably absent in the baseline. This suggests that **low-density and structured bottlenecks facilitate scaling deep RL networks**.

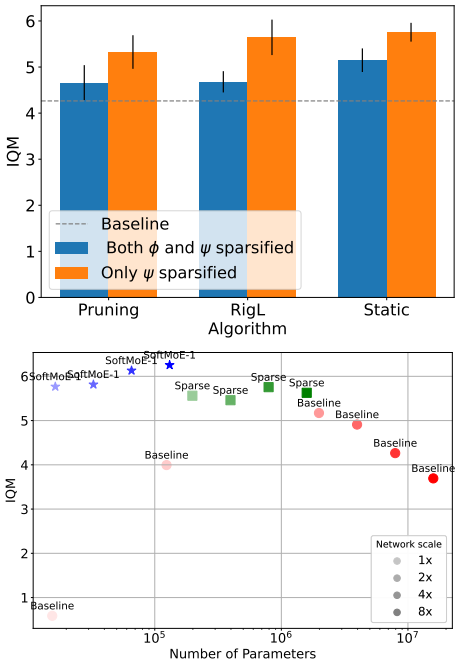

Figure 5: **(Top)** Across different sparse algorithms, sparsification of only ψ yields better performance than sparsifying ϕ and ψ. **(Bottom)** The relation between performance and the effective number of parameters in ψ for different approaches. Architectural methods have lower effective density than the baseline which correlates with the observed performance improvements.

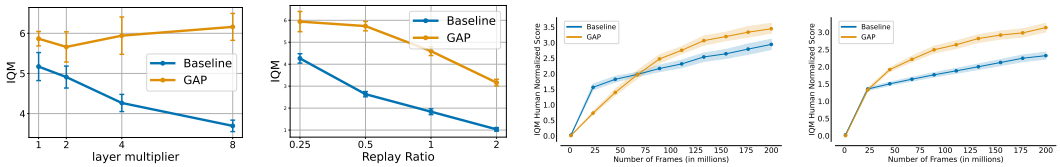

Figure 6: The impact of GAP across wide range of setting. From left to right: performance across different network scales, sample-efficient training with various high replay ratio values (default $= 0.25$), performance of the CNN architecture used by [Mnih et al., 2015], and performance on the full 60 games of Atari 2600. In all cases, GAP significantly improves performance.

4 Mind the GAP!

Having identified low-density and well-structured bottlenecks as the main factor enabling scaling networks, we demonstrate a *simple* alternative to the more sophisticated techniques recently explored: Global Average Pooling (GAP) [Lin et al., 2013]. We demonstrate its efficacy across wide range of settings and aspects, including: (1) *effectiveness* in network scaling under different scales, architectures, and in sample-efficient regime with high replay ratios (Figure 6); (2) *computational efficiency* (Figure 7); (3) improved training *stability* and feature learning (Figure 8); (4) unlocking *width and depth* scaling (Figure 9); and (5) *generalized gains* across various domains (Figure 11).

Simple The output feature maps of ϕ, denoted as $F \in \mathbb{R}^{H \times W \times C}$, are processed by average pooling. For each feature map (F^c), GAP computes the average over its spatial dimensions, resulting in the output ($\mathbf{g} \in \mathbb{R}^C$), which is then fed to the fully connected layers ψ (see Figure 1 for an illustration):

$$g^c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} F_{ij}^c. \tag{1}$$

Effective We evaluate the impact of this architectural change on various settings. *Scale*: We assess the performance of GAP across different network scales. The left plot of Figure 6 demonstrates that this simple architectural change unlocks scaling in RL networks, leading to significant performance improvements across different scales. *Sample-efficient regime*: increasing the number of gradient updates per environment interaction (replay ratio) is favorable for sample-efficiency. Yet, higher replay ratios often hurt performance [Nikishin et al., 2022]. We study the performance of the scaled network across varying the replay ratio values (default $= 0.25$). We find that even in this challenging setting, GAP has very strong performance compared to the baseline of the same network size, yielding more sample-efficient agents, as shown in the center left plot of Figure 6. *Varying architecture*: we evaluate the effect of GAP on a different architecture for ϕ: the original CNN architecture used in [Mnih et al., 2015]. The right center plot in Figure 6 shows that GAP can also provide performance gain to this architecture. *Full suite:* we assess the generalization of our findings beyond the 20 games used for most results and evaluate on the full set of 60 games of Atari. The rightmost plot of Figure 6 confirms that GAP consistently improves performance on the full suite.

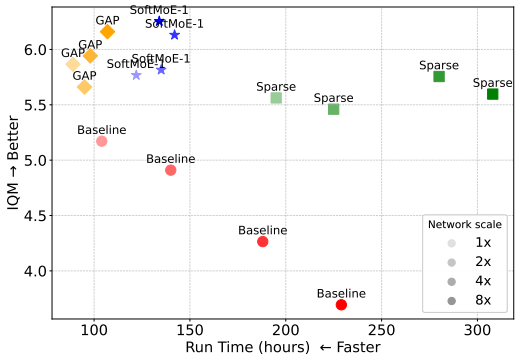

Figure 7: The computational cost versus performance across varying network scales for different algorithms. GAP offers the *highest* speed while obtaining substantial performance improvements.

Efficient In Figure 7 we report the total number of GPU hours per game for every scale, to compare the computational cost of the different methods. While scaling the baseline increases computational costs and degrades the performance, applying GAP to the encoder's output reduces

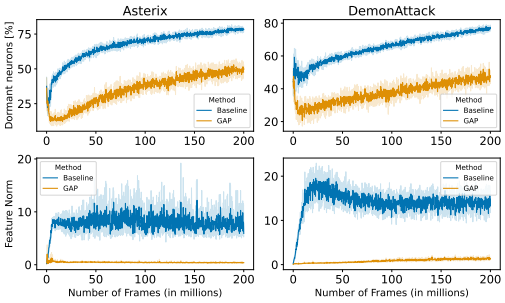

Figure 8: Scaled networks with GAP exhibit less dormant neurons than the baseline and have lower feature norm.

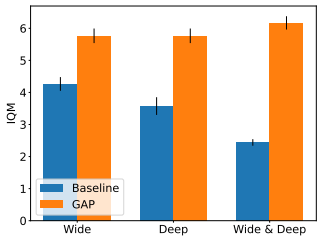

Figure 9: Impact of scaling network depth and width. Increasing depth hurts performance for the baseline networks, and scaling both width and depth makes it even worse. GAP, however, unlocks scaling even for increased network depth, leading to significant performance gains.

the effective density of ψ (and hence computational cost), while yielding performance gains. Despite having a similar effective density to SoftMoE, GAP's inherent simplicity makes it more efficient by avoiding the extra computations associated with SoftMoE's token construction and post-MoE projection. The high runtime observed in sparse methods, despite their low effective density, stems from the fact that sparsity is only simulated with parameter masking [Hoefler et al., 2021, Mocanu et al., 2018, Evci et al., 2020].

Stable training dynamics Figure 3 (right) displays the saliency maps when training with GAP, where the network seems to be attending to the most important and relevant areas of the input. We further examine the dormant neurons [Sokar et al., 2023] and the norm of the features in Figure 8. We observe that the network exhibits fewer dormant neurons than the baseline and lower feature norms, suggesting improved plasticity and training stability.

Unlocking width-depth scaling We demonstrate the effectiveness of GAP in scaling network width which is the focus of previous works [Obando Ceron* et al., 2024, Obando Ceron et al., 2024, Sokar et al., 2025]. Beyond this, we also explore how increasing ψ's depth, or both width and depth, impacts the performance of RL networks. Specifically, we add extra fully connected layer in ψ and evaluate the performance with both unscaled and scaled width of ψ. More analysis with varying depth are included in the appendix. We present the results in Figure 9. We find that the baseline's performance drops with increased depth, worsening significantly when scaling both dimensions. In contrast, interestingly GAP maintains strong performance across all scaling dimensions, confirming the impact of the representation learned by the bottleneck in the overall performance of the network.

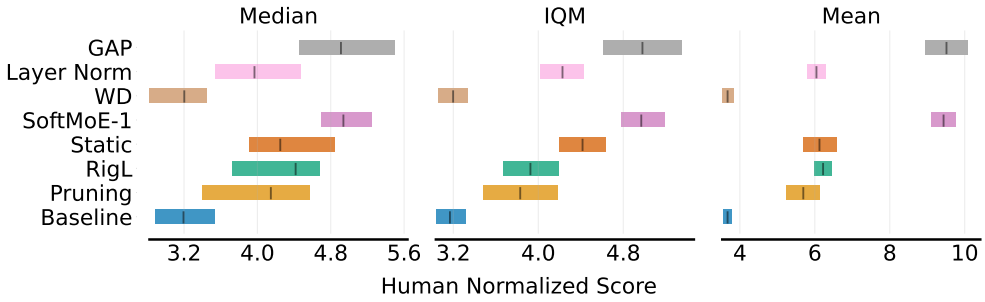

Figure 10: Comparison against architectural and algorithmic techniques for scaling RL networks. We report Median, IQM, and Mean scores [Agarwal et al., 2021] at 100M environment steps. GAP presents a notably simpler alternative to the baseline approaches, while achieving the best performance.

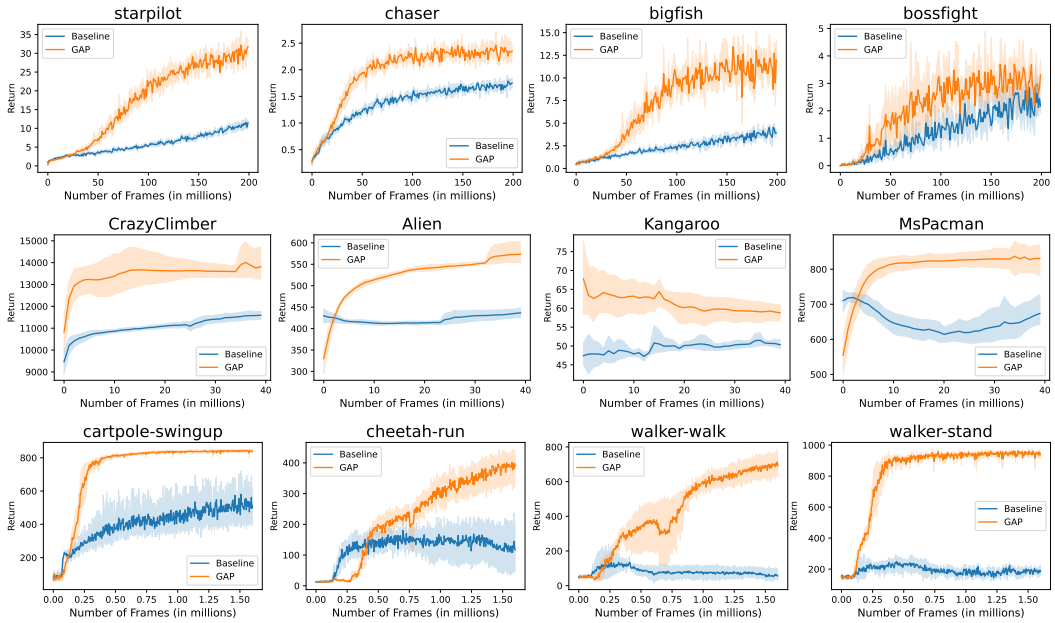

Figure 11: Performance for Rainbow on Procgen [Cobbe et al., 2019] **(top)**, DER on Atari100K [Łukasz Kaiser et al., 2020] **(middle)**, and SAC on DMC [Tassa et al., 2018] **(bottom)**. GAP leads to performance gains for scaled networks in diverse domains.

Comparison against other methods We compare the performance of GAP against various architectural techniques including pruning [Obando Ceron et al., 2024], static and dynamic sparsity (RigL) [Graesser et al., 2022], and SoftMoE-1 [Sokar et al., 2025]. Moreover, we include a comparison against algorithmic methods including weight decay [Sokar et al., 2023, Obando Ceron et al., 2024], and layer normalization [Nauman et al., 2024]. Figure 10 shows that GAP outperforms all baseline methods and achieves performance comparable to SoftMoE, while being more efficient and simpler. Although GAP is a more aggressive compression technique than granular approaches like per-conv tokens in SoftMoEs [Obando Ceron* et al., 2024], which could lead to a loss of spatial structure, it still preserves spatial information within each feature map more effectively than a flattening operation.

Generalization to other domains To further assess the broad applicability of our findings, we extended our evaluation to a diverse set of domains and agents. We assessed the Rainbow agent on Procgen [Cobbe et al., 2019], the DER agent [Van Hasselt et al., 2019] in the data-efficient Atari100K setting [Łukasz Kaiser et al., 2020], and the SAC agent [Haarnoja et al., 2018] on the continuous DeepMind Control (DMC) suite [Tassa et al., 2018]. For SAC, we follow the CNN architecture presented in [Yarats et al., 2021b] and scale the embedding layer of the actor and critic by 8. Figure 11 illustrates that these experiments align with our main results, confirming that GAP provides a consistent and significant performance benefit for scaled networks across these distinct benchmarks.

5 Related Work

Several works have shown that scaling RL network causes substantial performance degradation due to training instabilities exhibited by the network [Hessel et al., 2018, Bjorck et al., 2021, Obando Ceron* et al., 2024]. Although the precise causes of these issues remain unclear, several approaches aim to mitigate them. We categorize these methods to architectural methods that alter the standard network architecture and algorithmic methods.

Scaling through architectural changes Recent works have investigated architectural modifications to improve the scaling of RL networks. Obando Ceron* et al. [2024] integrated a Mixture-of-Experts (MoE) after the encoder in single-task RL networks. Their results across multiple domains

highlight the effectiveness of this approach for scaling RL, with SoftMoE [Puigcerver et al., 2024] outperforming traditional MoE Shazeer et al. [2017]. This MoE approach was later extended by Willi* et al. [2024], who demonstrated its applicability in the multi-task setting. Relatedly, Sokar et al. [2025] provided insights into why such methods might succeed, showing that replacing the standard flattened representation from the encoder with a tokenized representation that preserves spatial structure significantly improves the performance of scaled networks. Another line of research studies replacing dense parameters by sparse ones in both online [Graesser et al., 2022, Tan et al., 2023, Sokar et al., 2022] and offline RL [Arnob et al., 2021]. This approach has demonstrated its effectiveness in increasing the learning speed and performance of RL agents. Network sparsity is achieved either by starting with a dense network and progressively pruning weights, or by initializing with a sparse network and maintaining a consistent sparsity level throughout training. In the latter case, the sparse structure can be kept static or optimized dynamically during training using methods like SET [Mocanu et al., 2018] and RigL [Evci et al., 2020]. Recently, Obando Ceron et al. [2024] showed that gradual magnitude pruning helps in scaling RL networks, leading to improved performance. Concurrent works have independently proposed similar approaches to GAP [Trumpp et al., 2025, Kooi et al., 2025], which provides extra evidence for the efficacy of this method.

Scaling through algorithmic changes A primary goal in this line of research is maintaining training stability as networks grow. Bjorck et al. [2021] show that using spectral normalization [Miyato et al., 2018] helps to improve training stability and enable using large neural networks for actor-critic methods. Farebrother et al. [2023a] explore the usage of auxiliary tasks to learn scaled representations. Farebrother et al. [2024] demonstrated that training value networks using classification with categorical cross-entropy, as opposed to regression, leads to better performance in scaled networks. Other regularization techniques have been shown to be critical. Schwarzer* et al. [2023] propose several tricks to enable scaling including weight decay, network reset [Nikishin et al., 2022], increased discount factor, among others. Similarily, Nauman et al. [2024] employ a combination of layer normalization [Ba et al., 2016], weight decay, and weight reset.

6 Conclusion

Recent proposals for enabling scaling in deep reinforcement learning have relied on sophisticated architectural interventions, such as the use of mixtures-of-experts and sparse training techniques. We demonstrated that these methods are indirectly targeting the **bottleneck** connecting the encoder ϕ and dense layers ψ, in a standard deep RL network. A consequence of our analyses is that directly targeting this bottleneck, for instance with global average pooling, can achieve the same (or higher) performance gains.

Our work highlights the importance of better understanding the training dynamics of neural networks in the context of reinforcement learning. The fact that a simple technique like global average pooling can outperform existing literature suggests that architecture design is ripe for exploration. There is also a likely connection of our findings with works exploring representation learning for RL [Castro et al., 2021, Kemertas and Aumentado-Armstrong, 2021, Zhang et al., 2021, Farebrother et al., 2023b], given that these generally target the output of ϕ; indeed, most of these methods aim to *structure* the outputs of ϕ so as to improve the generalizability and efficiency of the networks. It would be valuable to investigate whether approaches like GAP are complementary with (and ideally help enhance) more sophisticated representation learning approaches.

Limitations Although our broad set of results suggest our findings are quite general, our investigation has focused on pixel-based environments where there is a clear bottleneck, or separation between ϕ (the convolutional layers) and ψ (the fully connected layers). It is not clear whether our findings extend to non-pixel based environments or architectures where there isn't a clear bottleneck, but would be an interesting line of future work.

Acknowledgements

The authors would like to thank Gheorghe Comanici, Joao Madeira Araujo, Karolina Dziugaite, Doina Precup, and the rest of the Google DeepMind team, as well as Roger Creus Castanyer and Johan Obando-Ceron, for valuable feedback on this work.

References

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

Samin Yeasar Arnob, Riyasat Ohib, Sergey Plis, and Doina Precup. Single-shot pruning for offline reinforcement learning. *arXiv preprint arXiv:2112.15579*, 2021.

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

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

Marc G. Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C. Machado, Subhodeep Moitra, Sameera S. Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Nils Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning with spectral normalization. *Advances in neural information processing systems*, 34:8242–8255, 2021.

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

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. MICo: Improved representations via sampling-based state similarity for markov decision processes. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=wFp6kmQELgu`.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International conference on machine learning*, pages 2943–2952. PMLR, 2020.

Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks. In *The Eleventh International Conference on Learning Representations*, 2023a. URL `https://openreview.net/forum?id=oGDKSt9JrZi`.

Jesse Farebrother, Joshua Greaves, Rishabh Agarwal, Charline Le Lan, Ross Goroshin, Pablo Samuel Castro, and Marc G. Bellemare. Proto-value networks: Scaling representation learning with auxiliary tasks, 2023b. URL `https://arxiv.org/abs/2304.12567`.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taiga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. In *International Conference on Machine Learning*, pages 13049–13071. PMLR, 2024.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The state of sparse training in deep reinforcement learning. In *International Conference on Machine Learning*, pages 7766–7792. PMLR, 2022.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

Matteo Hessel, Joseph Modayil, H. V. Hasselt, T. Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, M. G. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. *CoRR*, abs/2110.14096, 2021. URL `https://arxiv.org/abs/2110.14096`.

Jacob E. Kooi, Zhao Yang, and Vincent François-Lavet. Hadamax encoding: Elevating performance in model-free atari, 2025.

Joo Hyung Lee, Wonpyo Park, Nicole Elyse Mitchell, Jonathan Pilault, Johan Samir Obando Ceron, Han-Byul Kim, Namhoon Lee, Elias Frantar, Yun Long, Amir Yazdanbakhsh, et al. Jaxpruner: A concise library for sparsity research. In *Conference on Parsimony and Learning*, pages 515–528. PMLR, 2024.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. URL `http://arxiv.org/abs/1312.4400`.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

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

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

Johan Samir Obando Ceron, Aaron Courville, and Pablo Samuel Castro. In value-based deep reinforcement learning, a pruned network is a good network. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 38495–38519. PMLR, 21–27 Jul 2024.

Johan Samir Obando Ceron*, Ghada Sokar*, Timon Willi*, Clare Lyle, Jesse Farebrother, Jakob Nicolaus Foerster, Gintare Karolina Dziugaite, Doina Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep RL. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=X9VMhfFxwn.

Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *The Twelfth International Conference on Learning Representations*, 2024.

Max Schwarzer*, Johan Samir Obando Ceron*, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 30365–30380. PMLR, 23–29 Jul 2023.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Ghada Sokar, Elena Mocanu, Decebal Constantin Mocanu, Mykola Pechenizkiy, and Peter Stone. Dynamic sparse training for deep reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2022.

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

Ghada Sokar, Johan Samir Obando Ceron, Aaron Courville, Hugo Larochelle, and Pablo Samuel Castro. Don't flatten, tokenize! unlocking the key to softmoe's efficacy in deep RL. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8oCrlOaYcc.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Yiqin Tan, Pihe Hu, Ling Pan, Jiatai Huang, and Longbo Huang. Rlx2: Training a sparse deep reinforcement learning model from scratch. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Raphael Trumpp, Ansgar Schäfftlein, Mirco Theile, and Marco Caccamo. Impoola: The power of average pooling for image-based deep reinforcement learning. In *Reinforcement Learning Conference*, 2025.

Hado P Van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

Timon Willi*, Johan Samir Obando Ceron*, Jakob Nicolaus Foerster, Gintare Karolina Dziugaite, and Pablo Samuel Castro. Mixture of experts in a mixture of RL settings. In *Reinforcement Learning Conference*, 2024. URL https://openreview.net/forum?id=5FFO6RlOEm.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*, 2021a.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 10674–10681, 2021b.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.

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 paper aims at understanding the challenges in network scaling in RL and underlying reasons behind the success of existing algorithms. The abstract and introduction include this.

   Guidelines:

   - The answer NA 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 NA 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 discussed the limitations in Section 6.

   Guidelines:

   - The answer NA 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: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA 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: We provide all experimental details in Section 3.1 and the appendix.

   Guidelines:

   - The answer NA 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: The paper only uses already available open-source code, and provides sufficient instructions to faithfully reproduce.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all experimental details in Section 3.1.

   Guidelines:

   - The answer NA 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 run each experiment using 5 seeds. Error bars represent 95% stratified bootstrap confidence intervals.

   Guidelines:

   - The answer NA 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: We provide the compute resources in Section 3.1.

   Guidelines:

   - The answer NA 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: The work conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA 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: We discuss the broader impacts in the appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA 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: We include the license of used libraries.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The paper does not include LLMs.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Broader impacts

This paper studies the challenges in scaling networks in pixel-based deep reinforcement learning. We present a simple, effective alternative to current sophisticated approaches by identifying the effective network representation. The approach has a positive impact by requiring no hyperparameters, simplifying implementation and showing robust performance across various scenarios. While this research aims to advance RL agent capabilities without direct negative impact, we urge careful consideration of potential implications when building upon this work.

B Experimental Details

Table 1: Hyper-parameters for Rainbow and DER agents.

	Atari		
	Hyper-parameter	Rainbow	DER
Training	Adam's (ϵ)	1.5e-4	0.00015
	Adam's learning rate	6.25e-5	0.0001
	Batch Size	32	32
	Weight Decay	0	0
Architecture	Activation Function	ReLU	ReLU
	Fully connected layer Width	512	512
Algorithm	Replay Capacity	1000000	1000000
	Minimum Replay History	20000	1600
	Number of Atoms	51	51
	Reward Clipping	True	True
	Update Horizon	3	10
	Update Period	4	1
	Discount Factor	0.99	0.99
	Exploration ϵ	0.01	0.01
	Sticky Actions	True	False

Hyperparameter details We use the default hyperparameters for all the studied algorithms. We present the values of these parameters in Table 1. For the dormant neuron analysis, we use a dormancy threshold of 0.001. For the feature learning analysis (Figure 4), we increase the depth of the encoder by adding two ResNet blocks.

Atari Games [Bellemare et al., 2013] We use the set of 20 games used in Sokar et al. [2025], Obando Ceron* et al. [2024] for direct comparison. The set has the following games: Asterix, SpaceInvaders, Breakout, Pong, Qbert, DemonAttack, Seaquest, WizardOfWor, RoadRunner, BeamRider, Frostbite, CrazyClimber, Assault, Krull, Boxing, Jamesbond, Kangaroo, UpNDown, Gopher, and Hero. This set is used in most of our analysis, nevertheless we provide our main results on the full suite of 60 games.

Atari100K Games [Łukasz Kaiser et al., 2020] We test on the 26 games of this benchmark. It includes the following games: Alien, Amidar, Assault, Asterix, BankHeist, BattleZone, Boxing, Breakout, ChopperCommand, CrazyClimber, DemonAttack, Freeway, Frostbite, Gopher, Hero, Jamesbond, Kangaroo, Krull, KungFuMaster, MsPacman, Pong, PrivateEye, Qbert, RoadRunner, Seaquest, UpNDown.

C Extra Experiments

Sparse methods address the bottleneck Extending our analysis in section 3, we validate our hypothesis on various sparsity levels. Consistent with our main results, sparsification of only ψ yields better performance than sparsifying ϕ and ψ across all sparsity levels as shown in Figure 12.

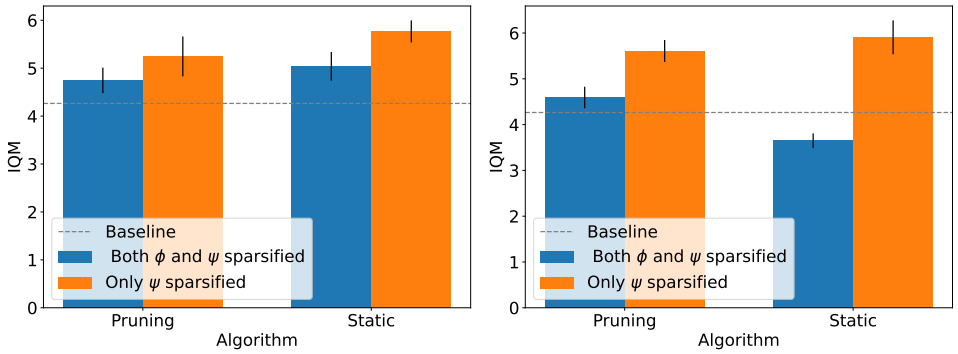

Figure 12: Sparsification of ψ yields better performance than sparsifying ϕ and ψ for 80% sparsity **(left)** and 95% sparsity **(right)**.

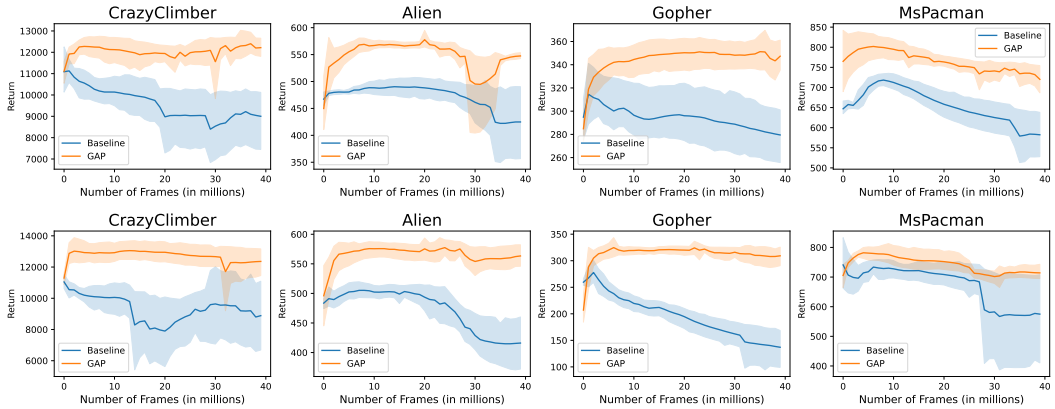

Figure 13: Performance for DrQ **(top)** and DrQ(ϵ) **(bottom)** [Yarats et al., 2021a, Agarwal et al., 2021] on Atari100K.

More agents We evaluate two more algorithms in the sample-efficient regime: DrQ and DrQ(ϵ) [Yarats et al., 2021a, Agarwal et al., 2021] on Atari100K. Consistent with our main results, GAP improves performance of scaled networks as shown in Figure 13.

Comparison against max pooling We compare the performance of global average pooling with global max pooling for Rainbow on the set of 20 games used in our main results. We find the GAP outperforms max pooling as demonstrated in Figure 14. We hypothesize this is due to GAP's ability to retain more comprehensive information through its averaging operation.

Encoder width scaling We perform additional experiment analyzing the effect of scaling the width of layers in ϕ by a factor of 4, while maintaining ψ unscaled. Figure 15 shows that GAP significantly improves performance.

Deeper networks While our main focus in this work is scaling the width of the network, we also performed some preliminary analysis on scaling the depth of the fully connected layers (ψ), exploring architectures with 1, 2, and 3 additional layers. As shown in Figure 16, increasing the depth degrades the performance of the agent. Increasingly, GAP improves performance over the corresponding baseline network of the same size across varying depth.

Performance throughout training Figure 17 and Figure 18 present the performance per game throughout training for Atari and Atari100K benchmarks, respectively.

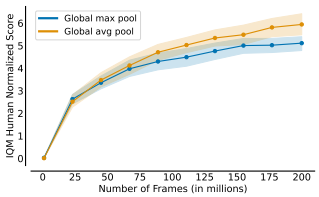

Figure 14: Comparison between GAP and global max pooling for Rainbow.

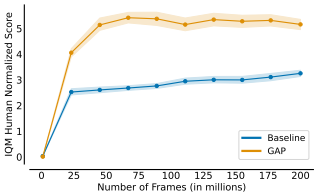

Figure 15: Effect of increasing the width of ϕ. GAP improves performance over the scaled baseline.

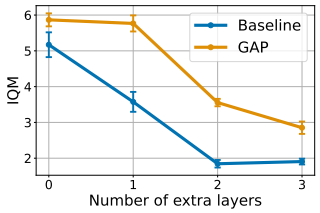

Figure 16: Effect of increasing the depth of ψ. GAP achieves better performance than the corresponding baseline with the same network size.

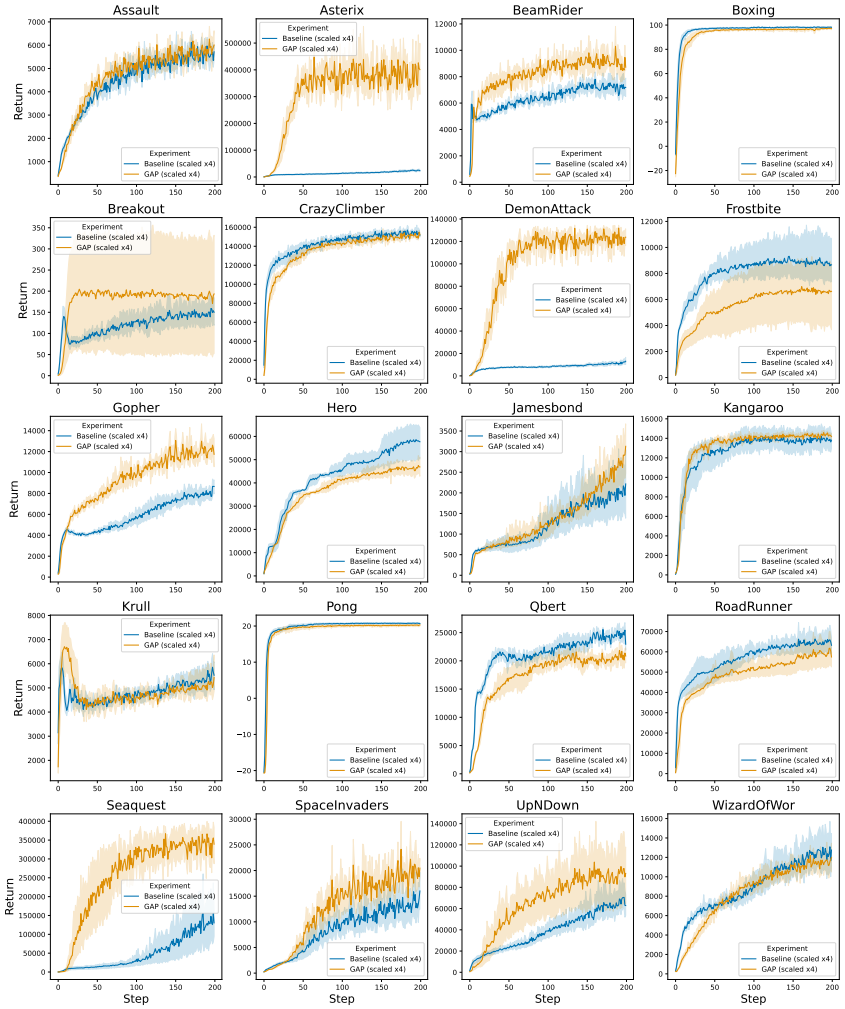

Figure 17: Performance during training on the 20 games of the Atrai benchmark.

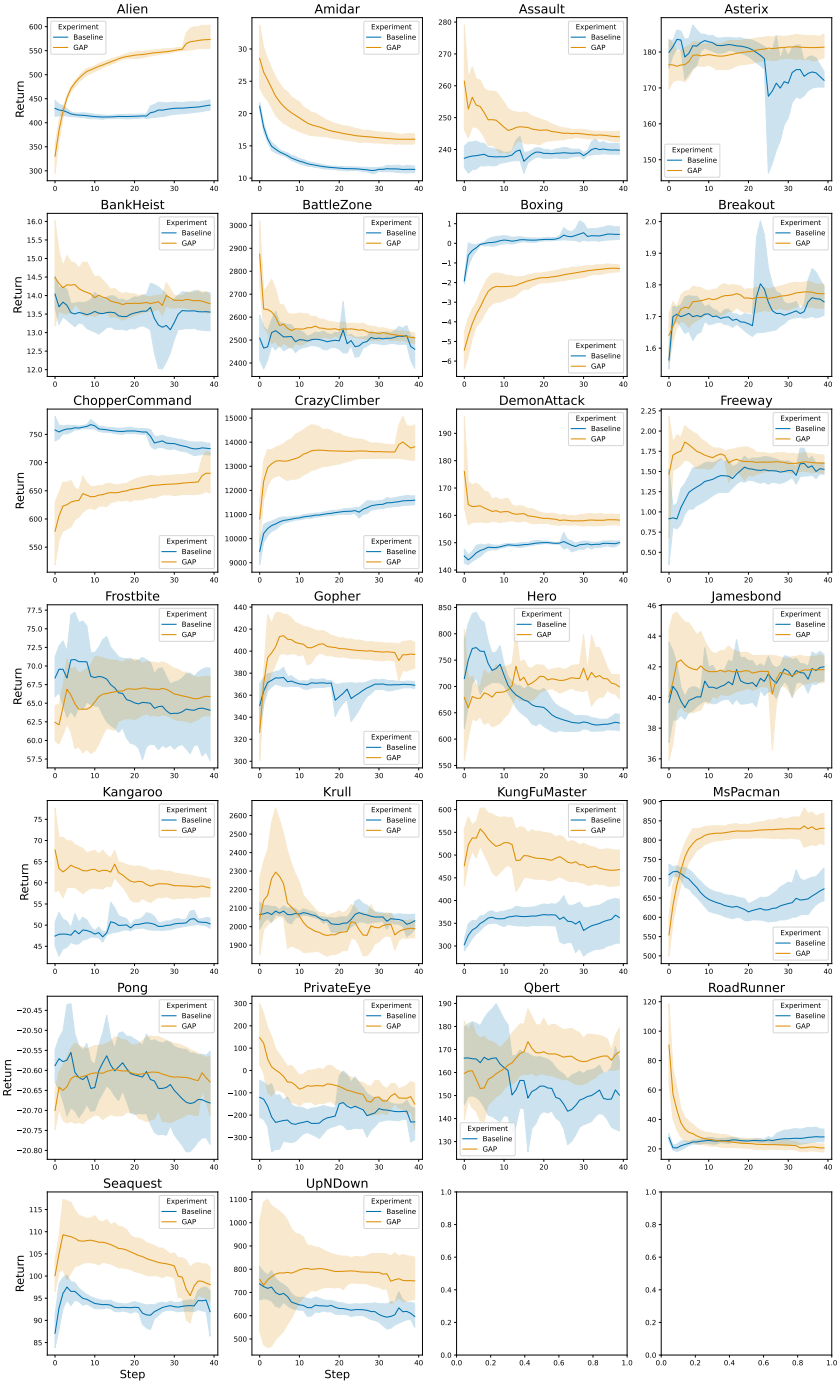

Figure 18: Performance during training on the 26 games of the Atrai100k benchmark.