Mixtures of Experts Unlock Parameter Scaling for Deep RL

Johan Obando-Ceron* [1 2 3] **Ghada Sokar*** [1] **Timon Willi*** [1 4] **Clare Lyle** [1] **Jesse Farebrother** [1 2 5]
Jakob Foerster [4] **Karolina Dziugaite** [1] **Doina Precup** [1 2 5] **Pablo Samuel Castro** [1 2 3]

Abstract

The recent rapid progress in (self) supervised learning models is in large part predicted by empirical scaling laws: a model's performance scales proportionally to its size. Analogous scaling laws remain elusive for reinforcement learning domains, however, where increasing the parameter count of a model often hurts its final performance. In this paper, we demonstrate that incorporating Mixture-of-Expert (MoE) modules, and in particular Soft MoEs (Puigcerver et al., 2023), into value-based networks results in more parameter-scalable models, evidenced by substantial performance increases across a variety of training regimes and model sizes. This work thus provides strong empirical evidence towards developing scaling laws for reinforcement learning. **We make our code publicly available.**

1. Introduction

Deep Reinforcement Learning (RL) – the combination of reinforcement learning algorithms with deep neural networks – has proven effective at producing agents that perform complex tasks at super-human levels (Mnih et al., 2015; Berner et al., 2019; Vinyals et al., 2019; Fawzi et al., 2022; Bellemare et al., 2020). While deep networks are critical to any successful application of RL in complex environments, their design and learning dynamics in RL remain a mystery. Indeed, recent work highlights some of the surprising phenomena that arise when using deep networks in RL, often going against the behaviours observed in supervised learning settings (Ostrovski et al., 2021; Kumar et al., 2021a; Lyle et al., 2022a; Graesser et al., 2022; Nikishin et al., 2022; Sokar et al., 2023; Ceron et al., 2023).

*Equal contribution [1]Google DeepMind [2]Mila - Québec AI Institute [3]Université de Montréal [4]University of Oxford [5]McGill University. Correspondence to: Johan Obando-Ceron <jobando0730@gmail.com>, Pablo Samuel Castro <psc@google.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

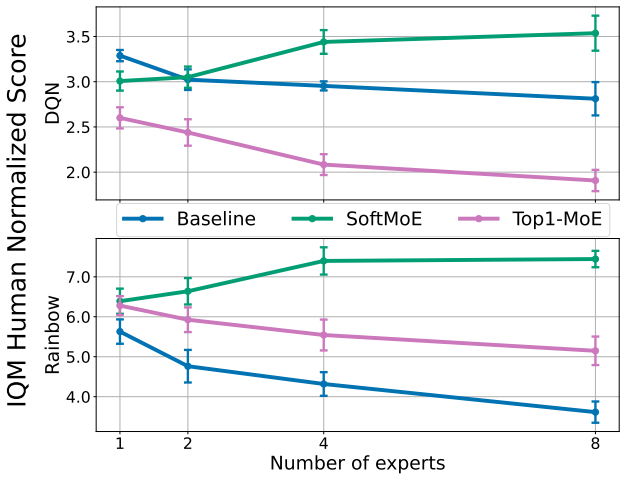

Figure 1. **The use of Mixture of Experts allows the performance of DQN (top) and Rainbow (bottom) to scale with an increased number of parameters**. While Soft MoE helps in both cases and improves with scale, Top1-MoE only helps in Rainbow, and does not improve with scale. The corresponding layer in the baseline is scaled by the number of experts to (approximately) match parameters. IQM scores computed over 200M environment steps over 20 games, with 5 independent runs each, and error bars showing 95% stratified bootstrap confidence intervals. The replay ratio is fixed to the standard 0.25.

The supervised learning community convincingly showed that larger networks result in improved performance, in particular for language models (Kaplan et al., 2020). In contrast, recent work demonstrates that scaling networks in RL is challenging and requires the use of sophisticated techniques to stabilize learning, such as supervised auxiliary losses, distillation, and pre-training (Farebrother et al., 2022; Taiga et al., 2022; Schwarzer et al., 2023). Furthermore, deep RL networks are *under-utilizing* their parameters, which may account for the observed difficulties in obtaining improved performance from scale (Kumar et al., 2021a; Lyle et al., 2022a; Sokar et al., 2023). Parameter count cannot be scaled efficiently if those parameters are not used effectively.

Architectural advances, such as transformers (Vaswani et al., 2017), adapters (Houlsby et al., 2019), and Mixtures of Experts (MoEs; Shazeer et al., 2017), have been central to the

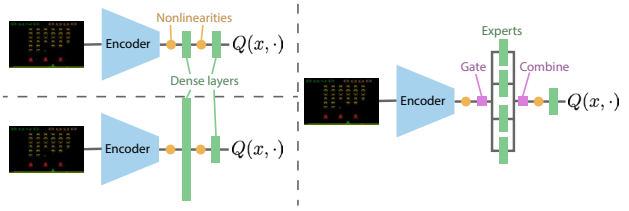

Figure 2. **Incorporating MoE modules into deep RL networks. Top left:** Baseline architecture; **bottom left:** Baseline with penultimate layer scaled up; **right:** Penultimate layer replaced with an MoE module.

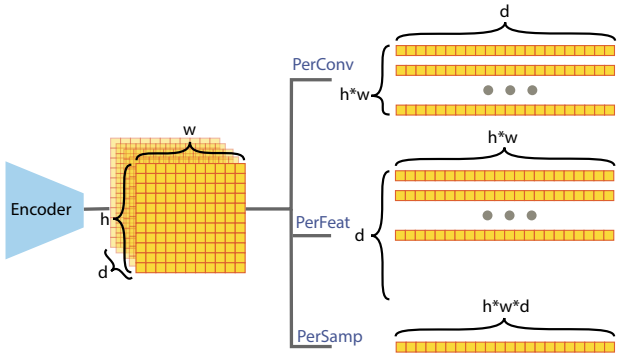

Figure 3. **Tokenization types considered:** PerConv (per convolution), PerFeat (per feature), and PerSamp (per sample).

scaling properties of supervised learning models, especially in natural language and computer vision problem settings. MoEs, in particular, are crucial to scaling networks to billions (and recently trillions) of parameters, because their modularity combines naturally with distributed computation approaches (Fedus et al., 2022). Additionally, MoEs induce *structured sparsity* in a network, and certain types of sparsity have been shown to improve network performance (Evci et al., 2020; Gale et al., 2019).

In this paper, we explore the effect of mixture of experts on the parameter scalability of value-based deep RL networks, i.e., does performance increase as we increase the number of parameters? We demonstrate that incorporating Soft MoEs (Puigcerver et al., 2023) strongly improves the performance of various deep RL agents, and performance improvements scale with the number of experts used. We complement our positive results with a series of analyses that help us understand the underlying causes for the results in Section 4. For example, we investigate different gating mechanisms, motivating the use of Soft MoE, as well as different tokenizations of inputs. Moreover, we analyse different properties of the experts hidden representations, such as dormant neurons (Sokar et al., 2023), which provide empirical evidence as to why Soft MoE improves performance over the baseline.

Finally, we present a series of promising results that pave the way for further research incorporating MoEs in deep RL networks in Section 5. For instance, in Section 5.1 we show preliminary results that Soft MoE outperforms the baseline on a set of Offline RL tasks; in Section 5.2, we evaluate Soft MoE's performance in low-data training regimes; and lastly, we show that exploring different architectural designs is a fruitful direction for future research in Section 5.3.

2. Preliminaries

2.1. Reinforcement Learning

Reinforcement learning methods are typically employed in sequential decision-making problems, with the goal of finding optimal behavior within a given environment (Sutton & Barto, 1998). These environments are typically formalized as Markov Decision Processes (MDPs), defined by the tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{X}$ represents the set of states, $\mathcal{A}$ the set of available actions, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \Delta(\mathcal{X})$[1] the transition function, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ the reward function, and $\gamma \in [0, 1)$ the discount factor.

An agent's behaviour, or policy, is expressed via a mapping $\pi : \mathcal{X} \to \Delta(\mathcal{A})$. Given a policy π, the *value* V^π of a state x is given by the expected sum of discounted rewards when starting from that state and following π from then on: $V^\pi(x) := \mathop{\mathbb{E}}_{\pi,\mathcal{P}} \left[\sum_{t=i}^{\infty} \gamma^t \mathcal{R}(x_t, a_t) \mid x_0 = x \right]$. The state-action function Q^π quantifies the value of first taking action a from state x, and then following π thereafter: $Q^\pi(x, a) := \mathcal{R}(x, a) + \gamma \mathop{\mathbb{E}}_{x' \sim \mathcal{P}(x,a)} V^\pi(x')$. Every MDP admits an optimal policy π^* in the sense that $V^{\pi^*} := V^* \geq V^\pi$ uniformly over $\mathcal{X}$ for all policies π.

When $\mathcal{X}$ is very large (or infinite), function approximators are used to express Q, e.g., DQN by Mnih et al. (2015) uses neural networks with parameters θ, denoted as $Q_\theta \approx Q$. The original architecture used in DQN, hereafter referred to as the CNN architecture, comprises of 3 convolutional layers followed by 2 dense layers, with ReLu nonlinearities (Fukushima, 1969) between each pair of layers. In this work, we mostly use the newer Impala architecture (Espeholt et al., 2018), but will provide a comparison with the original CNN architecture. DQN also used a *replay buffer*: a (finite) memory where an agent stores transitions received while interacting with the environment, and samples minibatches from to compute gradient updates. The *replay ratio*[2] is defined as the ratio of gradient updates to environment

[1] $\Delta(X)$ denotes a distribution over the set X.

[2] In the hyperparameters established in (Mnih et al., 2015), the policy is updated every 4 environment steps collected, resulting in a replay ratio of 0.25.

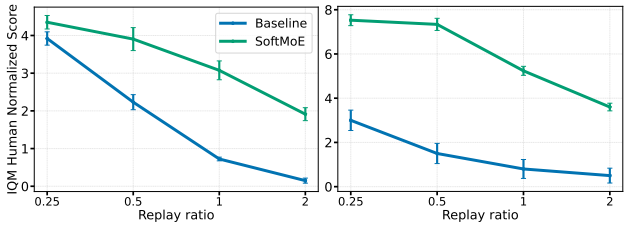

Figure 4. **Soft MoE yields performance gains even at high replay ratio values.** DQN (left) and Rainbow (right) with 8 experts. See Section 4 for training details.

interactions, and plays a role in our analyses below.

Rainbow (Hessel et al., 2018) extended the original DQN algorithm with multiple algorithmic components to improve learning stability, sample efficiency, and overall performance. Rainbow was shown to significantly outperform DQN and is an important baseline in deep RL research.

2.2. Mixtures of Experts

Mixtures of experts (MoEs) have become central to the architectures of most modern Large Language Models (LLMs). They consist of a set of n "expert" sub-networks activated by a gating network (typically learned and referred to as the *router*), which routes each incoming token to k experts (Shazeer et al., 2017). In most cases, k is smaller than the total number of experts (k=1 in our work), thereby inducing sparser activations (Fedus et al., 2022). These sparse activations enable faster inference and distributed computation, which has been the main appeal for LLM training. MoE modules typically replace the dense feed forward blocks in transformers (Vaswani et al., 2017). Their strong empirical results have rendered MoEs a very active area of research in the past few years (Shazeer et al., 2017; Lewis et al., 2021; Fedus et al., 2022; Zhou et al., 2022; Puigcerver et al., 2023; Lepikhin et al., 2020; Zoph et al., 2022; Gale et al., 2023).

Such hard assignments of tokens to experts introduce a number of challenges such as training instabilities, dropping of tokens, and difficulties in scaling the number of experts (Fedus et al., 2022; Puigcerver et al., 2023). To address some of these challenges, Puigcerver et al. (2023) introduced Soft MoE, which is a fully differentiable soft assignment of tokens-to-experts, replacing router-based hard token assignments.

Soft assignment is achieved by computing (learned) mixes of per-token weightings for each expert, and averaging their outputs. Following the notation of Puigcerver et al. (2023), let us define the input tokens as $\mathbf{X} \in \mathbb{R}^{m \times d}$, where m is the number of d-dimensional tokens. A Soft MoE layer applies a set of n experts on individual tokens, $\left\{f_i : \mathbb{R}^d \to \mathbb{R}^d\right\}_{1:n}$. Each expert has p input- and output-slots, represented respectively by a d-dimensional vector of parameters. We

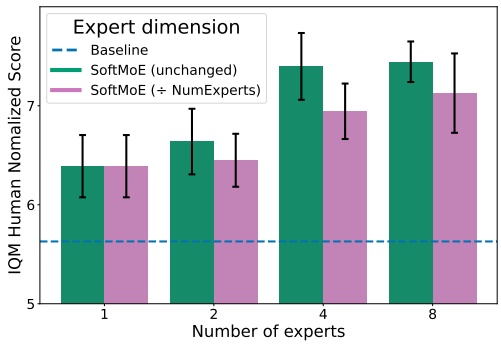

Figure 5. **Scaling down the dimensionality of Soft MoE experts has no significant impact on performance in Rainbow**. See Section 4 for training details.

denote these parameters by $\mathbf{\Phi} \in \mathbb{R}^{d \times (n \cdot p)}$.

The input-slots $\tilde{\mathbf{X}} \in \mathbb{R}^{(n \cdot p) \times d}$ correspond to a weighted average of all tokens: $\tilde{\mathbf{X}} = \mathbf{D}^\top \mathbf{X}$, where

$$\mathbf{D}_{ij} = \frac{\exp\left((\mathbf{X}\mathbf{\Phi})_{ij}\right)}{\sum_{i'=1}^{m} \exp\left((\mathbf{X}\mathbf{\Phi})_{i'j}\right)}.$$

D is typically referred to as the *dispatch weights*. We then denote the expert outputs as $\tilde{\mathbf{Y}}_i = f_{\lfloor i/p \rfloor}\left(\tilde{\mathbf{X}}_i\right)$. The output of the Soft MoE layer $\mathbf{Y}$ is the combination of $\tilde{\mathbf{Y}}$ with the *combine weights* $\mathbf{C}$ according to $\mathbf{Y} = \mathbf{C}\tilde{\mathbf{Y}}$, where

$$\mathbf{C}_{ij} = \frac{\exp\left((\mathbf{X}\mathbf{\Phi})_{ij}\right)}{\sum_{j'=1}^{n \cdot p} \exp\left((\mathbf{X}\mathbf{\Phi})_{ij'}\right)}.$$

Note how $\mathbf{D}$ and $\mathbf{C}$ are learned only through $\mathbf{\Phi}$, which we will use in our analysis. The results of Puigcerver et al. (2023) suggest that Soft MoE achieves a better trade-off between accuracy and computational cost compared to other MoE methods.

3. Mixture of Experts for Deep RL

Below we discuss some important design choices in our incorporation of MoE modules into DQN-based architectures.

Where to place the MoEs? Following the predominant usage of MoEs to replace dense feed-forward layers (Shazeer et al., 2017; Fedus et al., 2022; Gale et al., 2023), we replace the penultimate layer in our networks with an MoE module, where each expert has the same dimensionality as the original dense layer. Thus, we are effectively widening the penultimate layer's dimensionality by a factor equal to the number of experts. Figure 2 illustrates our deep RL MoE architecture. We discuss some alternatives in Section 5.3.

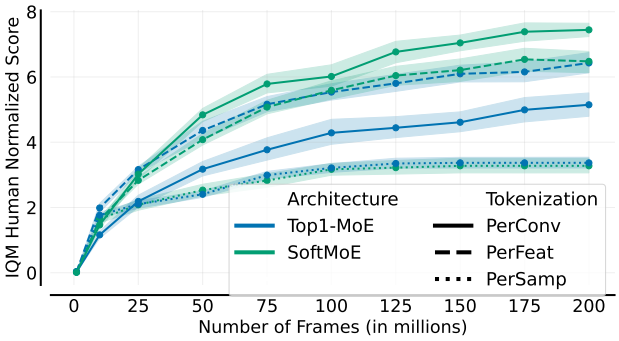

Figure 6. **Comparison of different tokenization methods on Rainbow with 8 experts.** Soft MoE with PerConv achieves the best results. Top1-MoE works best with PerFeat. PerSamp is worst over both architectures. See Section 4 for training details.

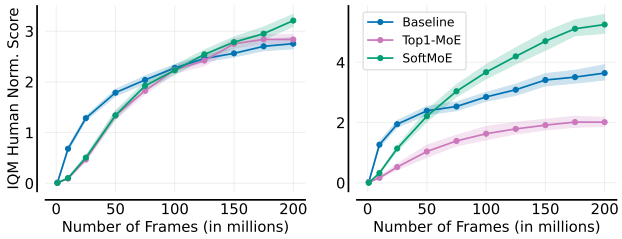

Figure 7. **In addition to the Impala encoder, MoEs also show performance gains for the standard CNN encoder on DQN (left) and Rainbow (right)**, with 8 experts. See Section 4 for training details.

What is a token? MoEs sparsely route inputs to a set of experts (Shazeer et al., 2017) and are mostly used in the context of transformer architectures where the inputs are *tokens* (Fedus et al., 2022). For the vast majority of supervised learning tasks where MoEs are used there is a well-defined notion of a token; except for a few works using Transformers (Chen et al., 2021), this is not the case for deep RL networks. Denoting by $C^{(h,w,d)} \in \mathbb{R}^3$ the output of the convolutional encoders, we define *tokens* as d-dimensional slices of this output; thus, we split C into $h \times w$ tokens of dimensionality d (PerConv in Figure 3). This approach to tokenization is taken from what is often used in vision tasks (see the Hybrid architecture of Dosovitskiy et al. (2021)). We did explore other tokenization approaches (discussed in Section 4.2), but found this one to be the best performing and most intuitive. Finally, a trainable linear projection is applied after each expert to maintain the token size d at the output.

What flavour of MoE to use? We explore the top-k gating architecture of Shazeer et al. (2017) following the simplified $k = 1$ strategy of Fedus et al. (2022), as well as the Soft MoE variant proposed by Puigcerver et al. (2023); for the rest of the paper we refer to the former as Top1-MoE and the latter as Soft MoE. We focus on these two, as Top1-MoE is the predominantly used approach, while Soft MoE is simpler and shows evidence of improving performance. Since Top1-MoEs activate one expert per token while Soft MoEs can activate multiple experts per token, Soft MoEs are arguably more directly comparable in terms of the number of parameters when widening the dense layers of the baseline.

4. Empirical evaluation

We focus our investigation on DQN (Mnih et al., 2015) and Rainbow (Hessel et al., 2018), two value-based agents that have formed the basis of a large swath of modern deep RL research. Recent work has demonstrated that using the ResNet architecture (Espeholt et al., 2018) instead of the original CNN architecture (Mnih et al., 2015) yields strong empirical improvements (Graesser et al., 2022; Schwarzer et al., 2023), so we conduct most of our experiments with this architecture. As in the original papers, we evaluate on 20 games from the Arcade Learning Environment (ALE), a collection of a diverse and challenging pixel-based environments (Bellemare et al., 2013b).

Our implementation, included with this submission, is built on the Dopamine library (Castro et al., 2018)[3], which adds stochasticity (via sticky actions) to the ALE (Machado et al., 2018). We use the recommendations from Agarwal et al. (2021) for statistically-robust performance evaluations, in particular focusing on interquartile mean (IQM). Every experiment was run for 200M environment steps, unless reported otherwise, with 5 independent seeds, and we report 95% stratified bootstrap confidence intervals. All experiments were run on NVIDIA Tesla P100 GPUs, and each took on average 4 days to complete.

4.1. Soft MoE Helps Parameter Scalability

To investigate the efficacy of Top1-MoE and Soft MoE on DQN and Rainbow, we replace the penultimate layer with the respective MoE module (see Figure 2) and vary the number of experts. Given that each expert is a copy of the original penultimate layer, we are effectively increasing the number of parameters of this layer by a factor equal to the number of experts. To compare more directly in terms of number of parameters, we evaluate simply widening the penultimate layer of the base architectures by a factor equal to the number of experts.

As Figure 1 demonstrates, Soft MoE provides clear performance gains, and these gains increase with the number of experts; for instance in Rainbow, increasing the number of experts from 1 to 8 results in a 20% performance improve-

[3]Dopamine code available at https://github.com/google/dopamine.

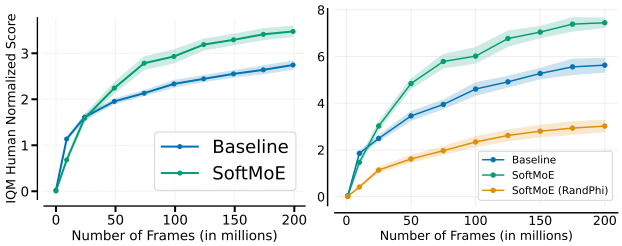

Figure 8. **(Left) Evaluating the performance on 60 Atari 2600 games; (Right) Evaluating the performance of Soft MoE with a random Φ matrix**. (Left) Even over 60 games, Soft MoE performs better than the baseline. (Right) Learning Φ is beneficial over a random phi, indicating that Soft MoE's perfomance gains are not only due to the distribution of tokens to experts. Both plots run with Rainbow using the Impala architecture and 8 experts. See Section 4 for training details.

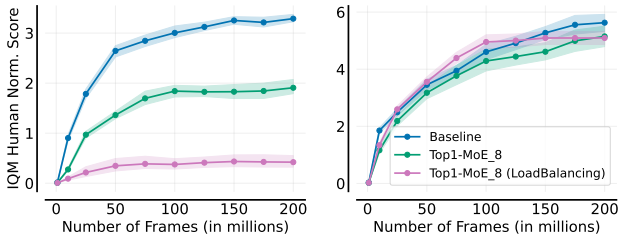

Figure 9. **Load-balancing losses from Ruiz et al. (2021) are unable to improve the performance of Top1-MoE** with 8 experts on neither DQN (left) nor Rainbow (right). See Section 4 for training details.

ment. In contrast, the performance of the base architectures *declines* as we widen its layer; for instance in Rainbow, as we increase the layer multiplier from 1 to 8 there is a performance decrease of around 40%. This is a finding consistent with prior work demonstrating the difficulty in scaling up deep RL networks (Farebrother et al., 2022; Taiga et al., 2022; Schwarzer et al., 2023). Top1-MoE seems to provide gains when incorporated into Rainbow, but fails to exhibit the parameter-scalability we observe with Soft MoE.

It is known that deep RL agents are unable to maintain performance when scaling the *replay ratio* without explicit interventions (D'Oro et al., 2023; Schwarzer et al., 2023). In Figure 4 we observe that the use of Soft MoEs maintains a strong advantage over the baseline even at high replay ratios, further confirming that they make RL networks more parameter efficient.

4.2. Impact of Design Choices

Number of experts Fedus et al. (2022) argued that the number of experts is the most efficient way to scale models in supervised learning settings. Our results in Figure 1 demonstrate that while Soft MoE does benefit from more experts, Top1-MoE does not.

Dimensionality of experts As Figure 1 confirms, scaling the corresponding layer in the base architectures does not match the performance obtained when using Soft MoEs (and in fact worsens it). We explored dividing the dimensionality of each expert by the number of experts, effectively bringing the number of parameters on-par with the original base architecture. Figure 5 demonstrates that Soft MoE maintains its performance, even with much smaller experts. This suggests the observed benefits come largely from the structured sparsity induced by Soft MoEs, and not necessarily the size of each expert.

Gating and combining Top1-MoEs and Soft MoEs use learned gating mechanisms, albeit different ones: the former uses a top-1 router, selecting an expert for each token, while the latter uses dispatch weights to assign weighted tokens to expert slots. The learned "combiner" component takes the output of the MoE modules and combines them to produce a single output (see Figure 2). If we use a single expert, the only difference between the base architectures and the MoE ones are then the extra learned parameters from the gating and combination components. Figure 1 suggests that most of the benefit of MoEs comes from the combination of the gating/combining components with multiple experts. Interestingly, Soft MoE with a single expert still provides performance gains for Rainbow, suggesting that the learned Φ matrix (see Section 2.2) has a beneficial role to play. Indeed, the right panel of Figure 8, where we replace the learned Φ with a random one, confirms this.

Tokenization As mentioned above, we focus most of our investigations on PerConv tokenization, but also explore two others: PerFeat is essentially a transpose of PerConv, producing d tokens of dimensionality $h \times w$; PerSamp uses the entire output of the encoder as a *single* token (see Figure 3). In Figure 6 we can observe that while PerConv works best with Soft MoE, Top1-MoE seems to benefit more from PerFeat.

Encoder Although we have mostly used the encoder from the Impala architecture (Espeholt et al., 2018), in Figure 7 we confirm that Soft MoE still provides benefits when used with the standard CNN architecture from Mnih et al. (2015).

Game selection To confirm that our findings are not limited to our choice of 20 games, we ran a study over all the 60 Atari 2600 games with 5 independent seeds, similar to previous works (Fedus et al., 2020; Ceron et al., 2023). In the left panel of Figure 8, we observe that Soft MoE results in improved performance over the full 60 games in the suite.

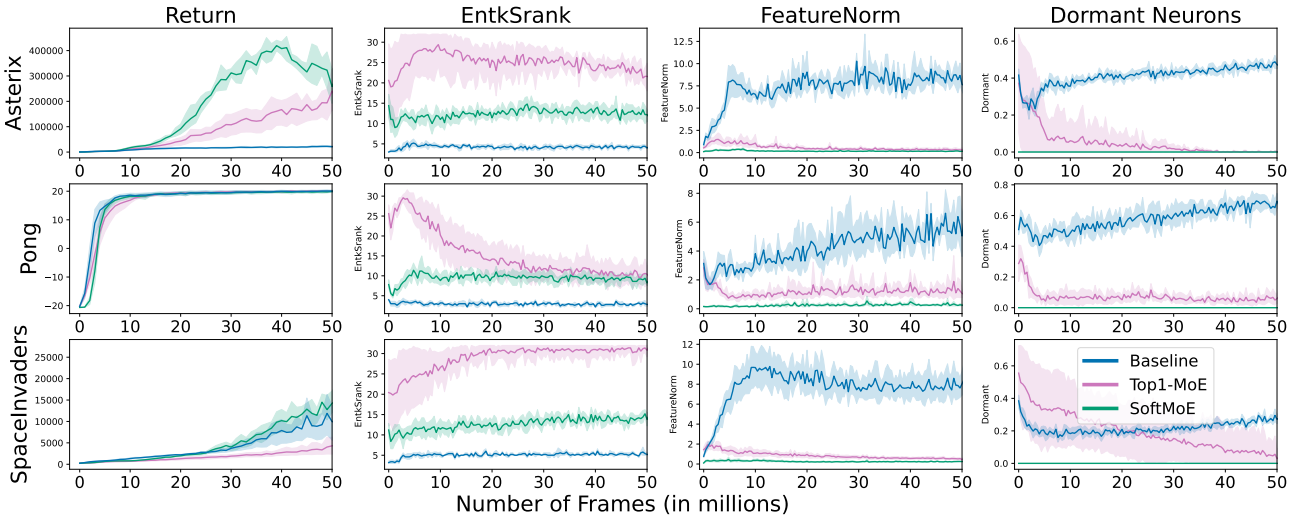

Figure 10. **Additional analyses:** both standard Top1-MoE and Soft MoE architectures exhibit similar properties of the hidden representation: both are more robust to dormant neurons and have higher effective rank of both the features and the gradients. We also see that the MoE architectures exhibit less feature norm growth than the baseline. See Section 4 for training details.

Number of active experts A crucial difference between the two flavors of MoEs considered here is in the activation of experts: Top1-MoE activates only *one* expert per forward pass (hard-gating), whereas Soft MoE activates all of them. Hard-gating is known to be a source of training difficulties, and many works have explored adding load-balancing losses (Shazeer et al., 2017; Ruiz et al., 2021; Fedus et al., 2022; Mustafa et al., 2022). To investigate whether Top1-MoE is underperforming due to improper load balancing, we added the load and importance losses proposed by Ruiz et al. (2021) (equation (7) in Appendix 2). Figure 9 suggests the addition of these load-balancing losses is insufficient to boost the performance of Top1-MoE. While it is possible other losses may result in better performance, these findings suggest that RL agents benefit from having a weighted combination of the tokens, as opposed to hard routing.

4.3. Additional Analysis

In the previous sections, we have shown that deep RL agents using MoE networks are better able to take advantage of network scaling, but it is not obvious a priori how they produce this effect. While a fine-grained analysis of the effects of MoE modules on network optimization dynamics lies outside the scope of this work, we zoom in on three properties known to correlate with training instability in deep RL agents: the rank of the features (Kumar et al., 2021a), interference between per-sample gradients (Lyle et al., 2022b), and dormant neurons (Sokar et al., 2023). We conduct a deeper investigation into the effect of MoE layers on learning dynamics by studying the Rainbow agent using the Impala architecture, and using 8 experts for the runs with the MoE modules. We track the norm of the features, the rank of the empirical neural tangent kernel (NTK) matrix (i.e. the matrix of dot products between per-transition gradients sampled from the replay buffer), and the number of dormant neurons – all using a batch size of 32 – and visualize these results in Figure 10.

We observe significant differences between the baseline architecture and the architectures that include MoE modules. The MoE architectures both exhibit higher numerical ranks of the empirical NTK matrices than the baseline network, and have negligible dormant neurons and feature norms. These findings suggest that the MoE modules have a stabilizing effect on optimization dynamics, though we refrain from claiming a direct causal link between improvements in these metrics and agent performance. For example, while the rank of the ENTK is higher for the MoE agents, the best-performing agent does not have the highest ENTK rank. The absence of pathological values of these statistics in the MoE networks does however, suggest that whatever the precise causal chain is, the MoE modules have a stabilizing effect on optimization.

5. Future directions

To provide an in-depth investigation into both the resulting performance and probable causes for the gains observed, we focused on evaluating DQN and Rainbow with Soft MoE and Top1-MoE on the standard ALE benchmark. The observed performance gains suggest that these ideas would also be beneficial in other training regimes. In this section

we provide strong empirical results in a number of different training regimes, which we hope will serve as indicators for promising future directions of research.

5.1. Offline RL

We begin by incorporating MoEs in offline RL, where agents are trained on a fixed dataset without environment interactions. Following prior work (Kumar et al., 2021b), we train the agents over 17 games and 5 seeds for 200 iterations, where 1 iteration corresponds to 62,500 gradient updates. We evaluated on datasets composed of 5%, 10%, 50% of the samples (drawn randomly) from the set of all environment interactions collected by a DQN agent trained for 200M steps (Agarwal et al., 2020). In Figure 11 we observe that combining Soft MoE with two modern offline RL algorithms (CQL (Kumar et al., 2020) and CQL+C51 (Kumar et al., 2022)) attains the best aggregate final performance. Top1-MoE provides performance improvements when used in conjunction with CQL+C51, but not with CQL. Similar improvements can be observed when using 10% of the samples (see Figure 17).

5.2. Agent Variants For Low-Data Regimes

DQN and Rainbow were both developed for a training regime where agents can take millions of steps in their environments. Kaiser et al. (2020) introduced the 100k[4] benchmark, which evaluates agents on a much smaller data regime (100k interactions) on 26 games. We evaluate the performance on two popular agents for this regime: DrQ(ϵ), an agent based on DQN (Yarats et al., 2021; Agarwal et al., 2021), and DER (Van Hasselt et al., 2019), an agent based on Rainbow. When trained for 100k environment interactions we saw no real difference between the different variants, suggesting that the benefits of MoEs, at least in our current setup, only arise when trained for a significant amount of interactions. However, when trained for 50M steps we do see gains in both agents, in particular with DER (Figure 12). The gains we observe in this setting are consistent with what we have observed so far, given that DrQ(ϵ) and DER are based on DQN and Rainbow, respectively.

5.3. Expert Variants

Our proposed MoE architecture replaces the feed-forward layer after the encoder with an MoE, where the experts consist of a single feed-forward layer (Figure 2). This is based on what is common practice when adding MoEs to transformer architectures, but is by no means the only way to utilize MoEs. Here we investigate three variants of using Soft MoE for DQN and Rainbow with the Impala architecture: **Big:** Each expert is a full network.The final linear layer is included in DQN (each expert has its own layer), but excluded from Rainbow (so the final linear layer is shared amongst experts).[5] **All:** A separate Soft MoE is applied at each layer of the network, excluding the last layer for Rainbow. **Regular:** the setting used in the rest of this paper. For **Big** and **All**, the routing is applied at the the input level, and we are using PerSamp tokenization (see Figure 3).

Puigcerver et al. (2023) propose using ℓ_2 normalization to each Soft MoE layer for large tokens, but mention that it makes little difference for smaller tokens. Since the tokens in our experiments above are small (relative to the large models typically using MoEs) we chose not to include this in the experiments run thus far. This may no longer be the case with **Big** and **All**, since we are using PerSamp tokenization on the full input; for this reason, we investigated the impact of ℓ_2 normalization when running these results.

Figure 13 summarizes our results, from which we can draw a number of conclusions. First, these experiments confirm our intuition that *not* including regularization in the setup used in the majority of this paper performs best. Second, consistent with Puigcerver et al. (2023), normalization seems to help with **Big** experts. This is particularly noticeable in DQN, where it even surpasses the performance of the Regular setup; the difference between the two is most likely due to the last layer being shared (as in Rainbow) or non-shared (as in DQN). Finally, using separate MoE modules for each layer (**All**) seems to not provide many gains, *especially* when coupled with normalization, where the agents are completely unable to learn.

In summary, our results suggest there may be promise in exploring alternative architectures for use with mixtures of experts.

6. Related Work

Mixture of Experts MoEs were first proposed by Jacobs et al. (1991) and have recently helped scaling language models up to trillions of parameters thanks to their modular nature, facilitating distributed training, and improved parameter efficiency at inference (Lepikhin et al., 2020; Fedus et al., 2022). MoEs are also widely studied in computer vision (Wang et al., 2020; Yang et al., 2019; Abbas & Andreopoulos, 2020; Pavlitskaya et al., 2020), where they enable scaling vision transformers to billions of parameters while reducing the inference computation cost by half (Riquelme et al., 2021), help with vision-based continual learning (Lee et al., 2019), and multi-task problems (Fan et al., 2022). MoEs also help performance in transfer- and

[4]Here, 100k refers to agent steps, or 400k environment frames, due to skipping frames in the standard training setup.

[5]This design choice was due to the use of C51 in Rainbow, which makes it non-trivial to maintain‘ the “token-preserving” property of MoEs.

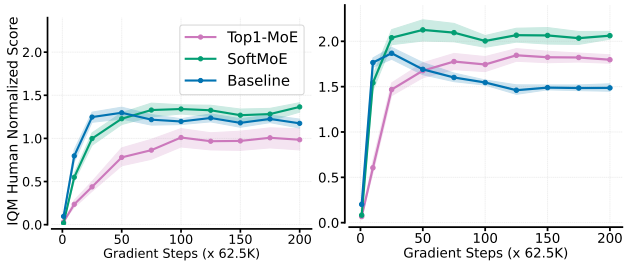

Figure 11. **Normalized performance across** 17 **Atari games for CQL (left) and CQL + C51 (right)**, with the ResNet (Espeholt et al., 2018) architecture and 8 experts trained on offline data. Soft MoE not only remains generally stable with more training, but also attains higher final performance. See Section 4 for training details.

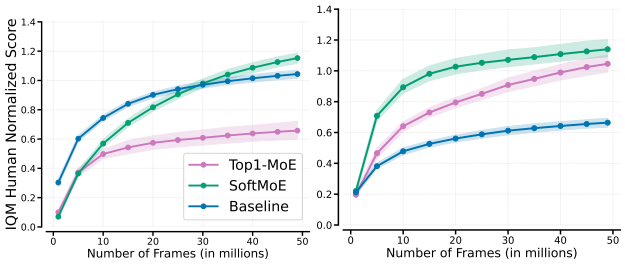

Figure 12. **Normalized performance across** 26 **Atari games for DrQ(ϵ) (left) and DER (right)**, with the ResNet architecture (Espeholt et al., 2018) and 8 experts (see Figure 16, for 4 experts). Soft MoE not only remains generally stable with more training, but also attains higher final performance. We report interquantile mean performance with error bars indicating 95% confidence intervals.

multi-task learning settings, e.g. by specializing experts to sub-problems (Puigcerver et al., 2020; Chen et al., 2023; Ye & Xu, 2023) or by addressing statistical performance issues of routers (Hazimeh et al., 2021). There have been few works exploring MoEs in RL for single (Ren et al., 2021; Akrour et al., 2022) and multi-task learning (Hendawy et al., 2024). However their definition and usage of "mixtures-of-experts" is somewhat different than ours, focusing more on orthogonal, probabilistic, and interpretable MoEs.

Parameter Scalability and Efficiency in Deep RL Lack of parameter scalability in deep RL can be partly explained by a lack of parameter efficiency. Parameter count cannot be scaled efficiently if those parameters are not used effectively. Recent work shows that networks in RL under-utilize their parameters. Sokar et al. (2023) demonstrates that networks suffer from an increasing number of inactive neurons throughout online training. Similar behavior is observed by Gulcehre et al. (2022) in offline RL. Arnob et al. (2021) shows that 95% of the network parameters can be pruned at initialization in offline RL without loss in performance. Numerous works demonstrate that periodic resetting of the network weights improves performance (Nikishin et al., 2022; Dohare et al., 2021; Sokar et al., 2023; D'Oro et al., 2022; Igl et al., 2020; Schwarzer et al., 2023).

Another line of research demonstrates that RL networks can be trained with a high sparsity level ($\sim$90%) without loss in performance (Tan et al., 2022; Sokar et al., 2022; Graesser et al., 2022; Ceron et al., 2024). These observations call for techniques to better utilize the network parameters in RL training, such as using MoEs, which we show decreases dormant neurons drastically over multiple tasks and architectures. To enable scaling networks in deep RL, prior works focus on algorithmic methods (Farebrother et al., 2022; Taiga et al., 2022; Schwarzer et al., 2023; Farebrother et al., 2024). In contrast, we focus on alternative *network topologies* to enable robustness towards scaling.

7. Discussion and Conclusion

As RL continues to be used for increasingly complex tasks, we will likely require larger networks. As recent research has shown (and which our results confirm), naïvely scaling up network parameters does not result in improved performance. Our work shows empirically that MoEs have a beneficial effect on the performance of value-based agents across a diverse set of training regimes.

Mixtures of Experts induce a form of *structured sparsity* in neural networks, prompting the question of whether the benefits we observe are simply a consequence of this sparsity rather than the MoE modules themselves. Our results suggest that it is likely a combination of both: Figure 1 demonstrates that in Rainbow, adding an MoE module with a *single* expert yields statistically significant performance improvements, while Figure 5 demonstrates that one can scale down expert dimensionality without sacrificing performance. The right panel of Figure 8 further confirms the necessity of the extra parameters in Soft MoE modules.

Recent findings in the literature demonstrate that while RL networks have a natural tendency towards neuron-level sparsity which can hurt performance (Sokar et al., 2023), they can benefit greatly from explicit parameter-level sparsity (Graesser et al., 2022). When taken in combination with our findings, they suggest that there is still much room for exploration, and understanding, of the role sparsity can play in training deep RL networks, especially for parameter scalability.

Even in the narrow setting we have focused on (replacing the penultimate layer of value-based agents with MoEs for off-policy learning in single-task settings), there are many open questions that can further increase the benefits of MoEs: different values of k for Top1-MoEs, different tokenization choices, using different learning rates (and perhaps optimizers) for routers, among others. Of course, expanding

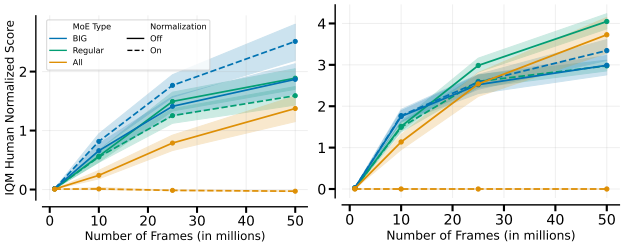

Figure 13. **Normalized performance over** 20 **games for expert variants** with DQN (left) and Rainbow (right), also investigating the use of the normalization of Puigcerver et al. (2023). We report interquantile mean performance with shaded areas indicating 95% confidence intervals.

beyond the ALE could provide more comprehensive results and insights, potentially at a fraction of the computational expense (Ceron & Castro, 2021).

The results presented in Section 5 suggest MoEs can play a more generally advantageous role in training deep RL agents. More broadly, our findings confirm the impact architectural design choices can have on the ultimate performance of RL agents. We hope our findings encourage more researchers to further investigate this – still relatively unexplored – research direction.

Acknowledgements

The authors would like to thank Gheorghe Comanici, Owen He, Alex Muzio, Adrien Ali Taïga, Rishabh Agarwal, Hugo Larochelle, Ayoub Echchahed, and the rest of the Google DeepMind Montreal team for valuable discussions during the preparation of this work; Gheorghe deserves a special mention for providing us valuable feedback on an early draft of the paper. We thank the anonymous reviewers for their valuable help in improving our manuscript. We would also like to thank the Python community (Van Rossum & Drake Jr, 1995; Oliphant, 2007) for developing tools that enabled this work, including NumPy (Harris et al., 2020), Matplotlib (Hunter, 2007), Jupyter (Kluyver et al., 2016), Pandas (McKinney, 2013) and JAX (Bradbury et al., 2018).

Impact statement

This paper presents work whose goal is to advance the field of Machine Learning, and reinforcement learning in particular. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abbas, A. and Andreopoulos, Y. Biased mixtures of experts: Enabling computer vision inference under data transfer limitations. *IEEE Transactions on Image Processing*, 29: 7656–7667, 2020.

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 104–114. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/agarwal20c.html.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Akrour, R., Tateo, D., and Peters, J. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(10): 6795–6806, oct 2022. ISSN 0162-8828. doi: 10.1109/TPAMI.2021.3103132. URL https://doi.org/10.1109/TPAMI.2021.3103132.

Arnob, S. Y., Ohib, R., Plis, S., and Precup, D. Single-shot pruning for offline reinforcement learning. *arXiv preprint arXiv:2112.15579*, 2021.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013a. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013b.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588:77 – 82, 2020.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., et al. Jax: composable transformations of python+ numpy programs. 2018.

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL http://arxiv.org/abs/1812.06110.

Ceron, J. S. O. and Castro, P. S. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, pp. 1373–1383. PMLR, 2021.

Ceron, J. S. O., Bellemare, M. G., and Castro, P. S. Small batch deep reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=wPqEvmwFEh.

Ceron, J. S. O., Courville, A., and Castro, P. S. In value-based deep reinforcement learning, a pruned network is a good network. In *Forty-first International Conference on Machine Learning*. PMLR, 2024. URL https://openreview.net/forum?id=seo9V9QRZp.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Chen, Z., Shen, Y., Ding, M., Chen, Z., Zhao, H., Learned-Miller, E. G., and Gan, C. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11828–11837, 2023.

Ding, X., Zhang, X., Han, J., and Ding, G. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11963–11975, 2022.

Dohare, S., Sutton, R. S., and Mahmood, A. R. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022. URL https://openreview.net/forum?id=4GBGwVIEYJ.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OpC-9aBBVJe.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2943–2952. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/evci20a.html.

Fan, Z., Sarkar, R., Jiang, Z., Chen, T., Zou, K., Cheng, Y., Hao, C., Wang, Z., et al. M^{3}vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems*, 35:28441–28457, 2022.

Farebrother, J., Greaves, J., Agarwal, R., Le Lan, C., Goroshin, R., Castro, P. S., and Bellemare, M. G. Proto-value networks: Scaling representation learning with auxiliary tasks. In *The Eleventh International Conference on Learning Representations*, 2022.

Farebrother, J., Orbay, J., Vuong, Q., Taïga, A. A., Chebotar, Y., Xiao, T., Irpan, A., Levine, S., Castro, P. S., Faust, A., Kumar, A., and Agarwal, R. Stop regressing: Training value functions via classification for scalable deep rl. In *Forty-first International Conference on Machine Learning*. PMLR, 2024.

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R Ruiz, F. J., Schrittwieser, J., Swirszcz, G., et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pp. 3061–3071. PMLR, 2020.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

Fukushima, K. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Trans. Syst. Sci. Cybern.*, 5(4):322–333, 1969. doi: 10.1109/TSSC.1969.300225. URL https://doi.org/10.1109/TSSC.1969.300225.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019. URL http://arxiv.org/abs/1902.09574.

Gale, T., Narayanan, D., Young, C., and Zaharia, M. MegaBlocks: Efficient Sparse Training with Mixture-of-Experts. *Proceedings of Machine Learning and Systems*, 5, 2023.

Graesser, L., Evci, U., Elsen, E., and Castro, P. S. The state of sparse training in deep reinforcement learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7766–7792. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/graesser22a.html.

Gulcehre, C., Srinivasan, S., Sygnowski, J., Ostrovski, G., Farajtabar, M., Hoffman, M., Pascanu, R., and Doucet, A. An empirical study of implicit regularization in deep offline rl. *arXiv preprint arXiv:2207.02099*, 2022.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., and Chi, E. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347, 2021.

Hendawy, A., Peters, J., and D'Eramo, C. Multi-task reinforcement learning with mixture of orthogonal experts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=aZH1dM3GOX.

Hessel, M., Modayil, J., Hasselt, H. V., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Jesson, A., Lu, C., Gupta, G., Filos, A., Foerster, J. N., and Gal, Y. Relu to the rescue: Improve your on-policy actor-critic with positive advantages. *arXiv preprint arXiv:2306.01460*, 2023.

Kaiser, L., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., and Jupyter Development Team. Jupyter Notebooks—a publishing format for reproducible computational workflows. In *IOS Press*, pp. 87–90. 2016. doi: 10.3233/978-1-61499-649-1-87.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

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

Kumar, A., Agarwal, R., Ma, T., Courville, A., Tucker, G., and Levine, S. Dr3: Value-based deep reinforcement learning requires explicit regularization. In *International Conference on Learning Representations*, 2021b.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. In *The Eleventh International Conference on Learning Representations*, 2022.

Lee, S., Ha, J., Zhang, D., and Kim, G. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2019.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2020.

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:232428341.

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

Lyle, C., Rowland, M., Dabney, W., Kwiatkowska, M., and Gal, Y. Learning dynamics and generalization in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 14560–14581. PMLR, 2022b.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the arcade learning environment: evaluation protocols and open problems for general agents. *J. Artif. Int. Res.*, 61 (1):523–562, jan 2018. ISSN 1076-9757.

McKinney, W. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 1 edition, February 2013. ISBN 9789351100065. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/1449319793.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, February 2015.

Mustafa, B., Riquelme, C., Puigcerver, J., Jenatton, R., and Houlsby, N. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems*, 35:9564–9576, 2022.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16828–16847. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/nikishin22a.html.

Oliphant, T. E. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007. doi: 10.1109/MCSE.2007.58.

Ostrovski, G., Castro, P. S., and Dabney, W. The difficulty of passive learning in deep reinforcement learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=nPHA8fGicZk.

Pavlitskaya, S., Hubschneider, C., Weber, M., Moritz, R., Huger, F., Schlicht, P., and Zollner, M. Using mixture of expert models to gain insights into semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 342–343, 2020.

Puigcerver, J., Ruiz, C. R., Mustafa, B., Renggli, C., Pinto, A. S., Gelly, S., Keysers, D., and Houlsby, N. Scalable transfer learning with expert models. In *International Conference on Learning Representations*, 2020.

Puigcerver, J., Riquelme, C., Mustafa, B., and Houlsby, N. From sparse to soft mixtures of experts, 2023.

Ren, J., Li, Y., Ding, Z., Pan, W., and Dong, H. Probabilistic mixture-of-experts for efficient deep reinforcement learning. *CoRR*, abs/2104.09122, 2021. URL https://arxiv.org/abs/2104.09122.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34: 8583–8595, 2021.

Ruiz, C. R., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Pinto, A. S., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=FrIDgjDOH1u.

Schwarzer, M., Obando Ceron, J. S., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger,

better, faster: Human-level Atari with human-level efficiency. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 30365–30380. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/schwarzer23a.html.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. 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.

Sokar, G., Mocanu, E., Mocanu, D. C., Pechenizkiy, M., and Stone, P. Dynamic sparse training for deep reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2022.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. The dormant neuron phenomenon in deep reinforcement learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32145–32168. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/sokar23a.html.

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

Taiga, A. A., Agarwal, R., Farebrother, J., Courville, A., and Bellemare, M. G. Investigating multi-task pretraining and generalization in reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Tan, Y., Hu, P., Pan, L., Huang, J., and Huang, L. Rlx2: Training a sparse deep reinforcement learning model from scratch. In *The Eleventh International Conference on Learning Representations*, 2022.

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

Van Rossum, G. and Drake Jr, F. L. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wang, X., Yu, F., Dunlap, L., Ma, Y.-A., Wang, R., Mirhoseini, A., Darrell, T., and Gonzalez, J. E. Deep mixture of experts via shallow embedding. In *Uncertainty in artificial intelligence*, pp. 552–562. PMLR, 2020.

Yang, B., Bender, G., Le, Q. V., and Ngiam, J. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in neural information processing systems*, 32, 2019.

Yarats, D., Kostrikov, I., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

Ye, H. and Xu, D. Taskexpert: Dynamically assembling multi-task representations with memorial mixture-of-experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21828–21837, 2023.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A. M., Le, Q. V., Laudon, J., et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. St-moe: Designing stable and transferable sparse expert models, 2022.

A. Experimental details

Unless otherwise specified, in all experiments below we report the interquantile mean after 40 million environment steps; error bars indicate 95% stratified bootstrap confidence intervals (Agarwal et al., 2021). Most of our experiments were run with 20 games from the ALE suite (Bellemare et al., 2013a), as suggested by Fedus et al. (2020). However, for the Atari $100k$ agents (subsection 5.2), we used the standard set of 26 games (Kaiser et al., 2020) to be consistent with the benchmark. Finally, we also ran some experiments with the full set of 60 games. The specific games are detailed below.

20 game subset: AirRaid, Asterix, Asteroids, Bowling, Breakout, DemonAttack, Freeway, Gravitar, Jamesbond, MontezumaRevenge, MsPacman, Pong, PrivateEye, Qbert, Seaquest, SpaceInvaders, Venture, WizardOfWor, YarsRevenge, Zaxxon.

26 game subset: 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.

60 game set: The 26 games above in addition to: AirRaid, Asteroids, Atlantis, BeamRider, Berzerk, Bowling, Carnival, Centipede, DoubleDunk, ElevatorAction, Enduro, FishingDerby, Gravitar, IceHockey, JourneyEscape, MontezumaRevenge, NameThisGame, Phoenix, Pitfall, Pooyan, Riverraid, Robotank, Skiing, Solaris, SpaceInvaders, StarGunner, Tennis, TimePilot, Tutankham, Venture, VideoPinball, WizardOfWor, YarsRevenge, Zaxxon.

B. Hyper-parameters list

Default hyper-parameter settings for DER (Van Hasselt et al., 2019) and DrQ(ϵ) (Kaiser et al., 2020; Agarwal et al., 2021) agents. Table 1 shows the default values for each hyper-parameter across all the Atari games.

Table 1. Default hyper-parameters setting for DER and DrQ(ϵ) agents.

	Atari	
Hyper-parameter	DER	DrQ(ϵ)
Adam's(ϵ)	0.00015	0.00015
Adam's learning rate	0.0001	0.0001
Batch Size	32	32
Conv. Activation Function	ReLU	ReLU
Convolutional Width	1	1
Dense Activation Function	ReLU	ReLU
Dense Width	512	512
Normalization	None	None
Discount Factor	0.99	0.99
Exploration ϵ	0.01	0.01
Exploration ϵ decay	2000	5000
Minimum Replay History	1600	1600
Number of Atoms	51	0
Number of Convolutional Layers	3	3
Number of Dense Layers	2	2
Replay Capacity	1000000	1000000
Reward Clipping	True	True
Update Horizon	10	10
Update Period	1	1
Weight Decay	0	0
Sticky Actions	False	False

Default hyper-parameter settings for DQN (Mnih et al., 2015) and Rainbow (Hessel et al., 2018) agents. Table 2 shows the default values for each hyper-parameter across all the Atari games.

Table 2. Default hyper-parameters setting for DQN and Rainbow agents.

	Atari	
Hyper-parameter	DQN	Rainbow
Adam's (ϵ)	1.5e-4	1.5e-4
Adam's learning rate	6.25e-5	6.25e-5
Batch Size	32	32
Conv. Activation Function	ReLU	ReLU
Convolutional Width	1	1
Dense Activation Function	ReLU	ReLU
Dense Width	512	512
Normalization	None	None
Discount Factor	0.99	0.99
Exploration ϵ	0.01	0.01
Exploration ϵ decay	250000	250000
Minimum Replay History	20000	20000
Number of Atoms	0	51
Number of Convolutional Layers	3	3
Number of Dense Layers	2	2
Replay Capacity	1000000	1000000
Reward Clipping	True	True
Update Horizon	1	3
Update Period	4	4
Weight Decay	0	0
Sticky Actions	True	True

Default hyper-parameter settings for CQL (Kumar et al., 2020) and CQL+C51 (Kumar et al., 2022) offline agents. Table 3 shows the default values for each hyper-parameter across all the Atari games.

Table 3. Default hyper-parameters setting for CQL and CQL+C51 agents.

	Atari	
Hyper-parameter	CQL	CQL+C51
Adam's(ϵ)	0.0003125	0.00015
Batch Size	32	32
Conv. Activation Function	ReLU	ReLU
Convolutional Width	1	1
Dense Activation Function	ReLU	ReLU
Normalization	None	None
Dense Width	512	512
Discount Factor	0.99	0.99
Learning Rate	0.00005	0.0000625
Number of Atoms	0	51
Number of Convolutional Layers	3	3
Number of Dense Layers	2	2
Fixed Replay Capacity	2,500,000 steps	2,500,000 steps
Reward Clipping	True	True
Update Horizon	1	3
Update Period	1	1
Weight Decay	0	0
Replay Scheme	Uniform	Uniform
Dueling	False	True
Double DQN	False	True
CQL coef	0.1	0.1

Default hyper-parameter settings for CNN architecture (Mnih et al., 2015) and Impala-based ResNet (Espeholt et al., 2018) Table 4 shows the default values for each hyper-parameter across all the Atari games.

Table 4. Default hyper-parameters for neural networks.

	Atari	
Hyper-parameter	CNN architecture (Mnih et al., 2015)	Impala-based ResNet (Espeholt et al., 2018)
Observation down-sampling	(84, 84)	(84, 84)
Frames stacked	4	4
Q-network (channels)	32, 64, 64	32, 64, 64
Q-network (filter size)	8 x 8, 4 x 4, 3 x 3	8 x 8, 4 x 4, 3 x 3
Q-network (stride)	4, 2, 1	4, 2, 1
Num blocks	-	2
Use max pooling	False	True
Skip connections	False	True
Hardware	Tesla P100 GPU	Tesla P100 GPU

C. Extra results

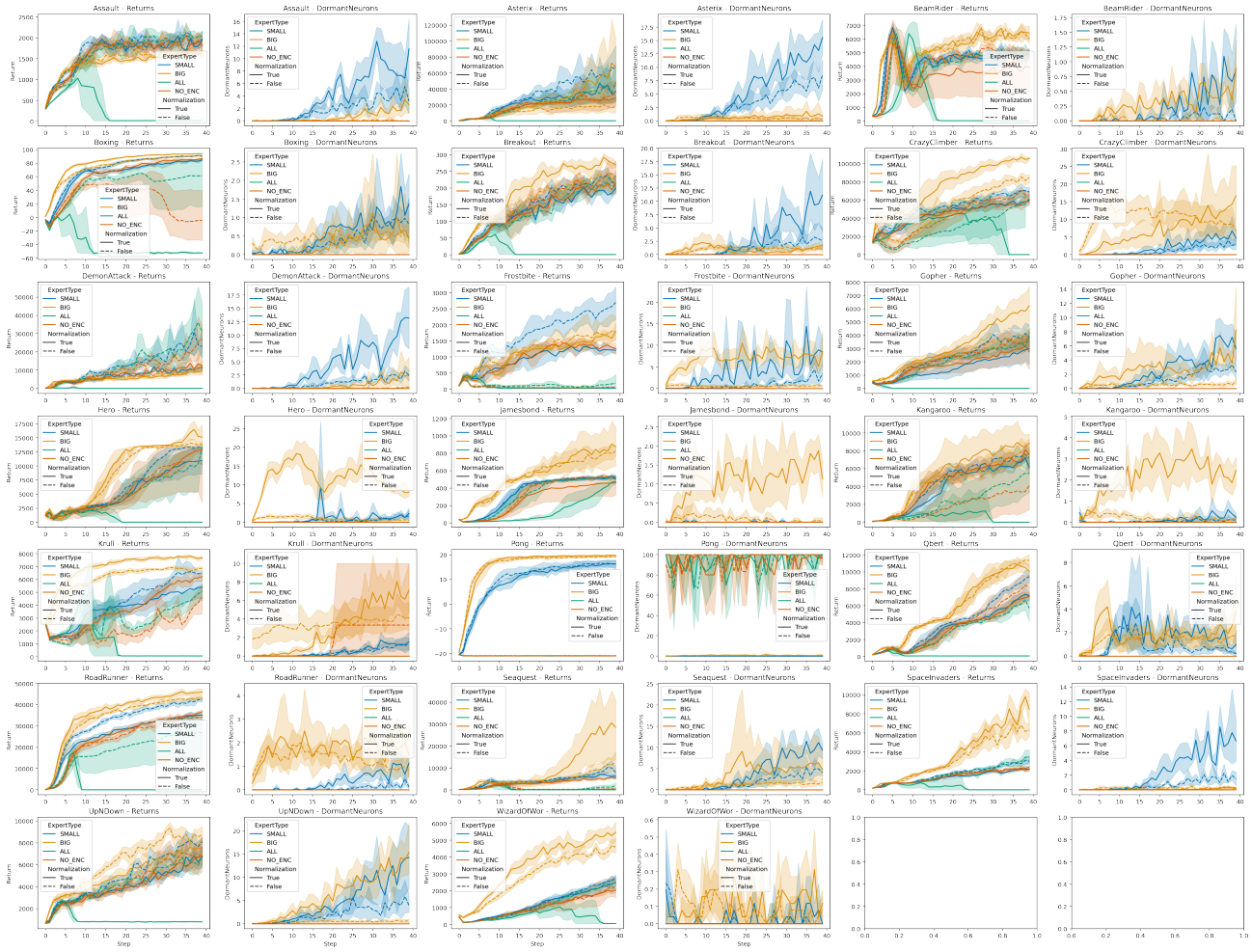

Figure 14. Results for architectural ablations as described in Section 5.3 on DQN. Additionally, we investigate the effect of the normalization that was proposed in the original Soft MoE paper.

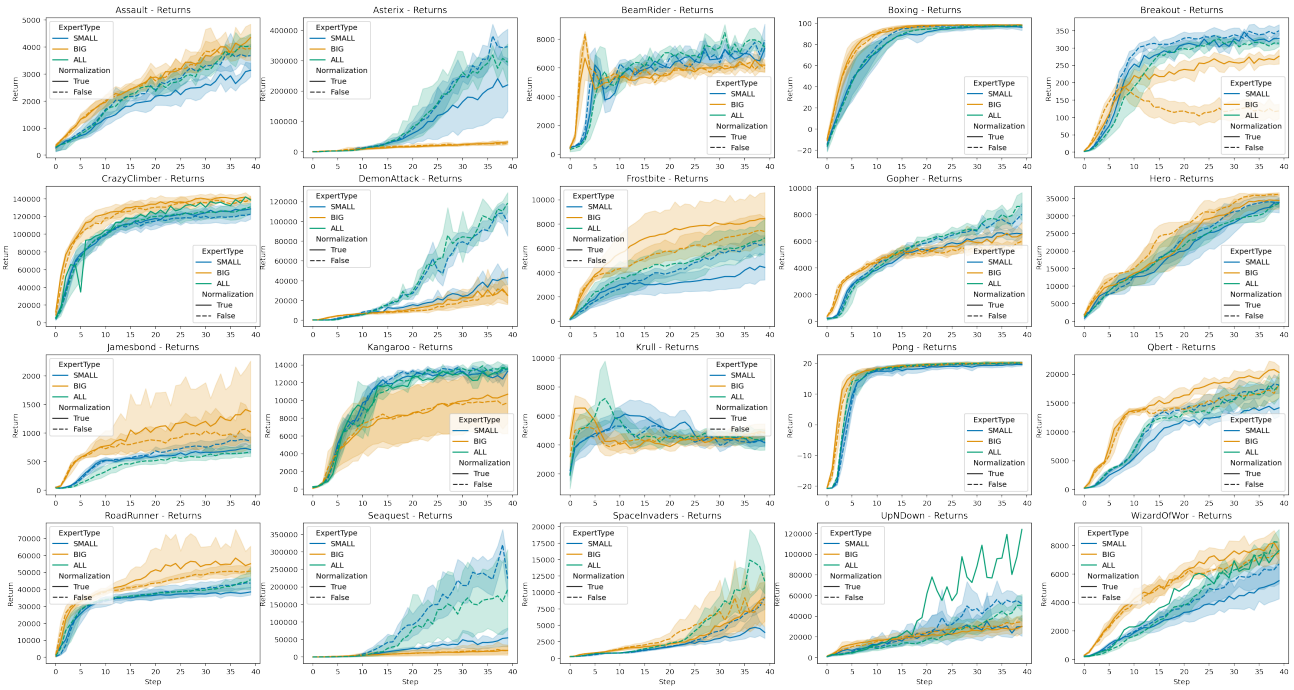

Figure 15. Results for architectural exploration as described in Section 5.3 on Rainbow. Additionally, we investigate the effect of the normalization that was proposed in the original Soft MoE paper.

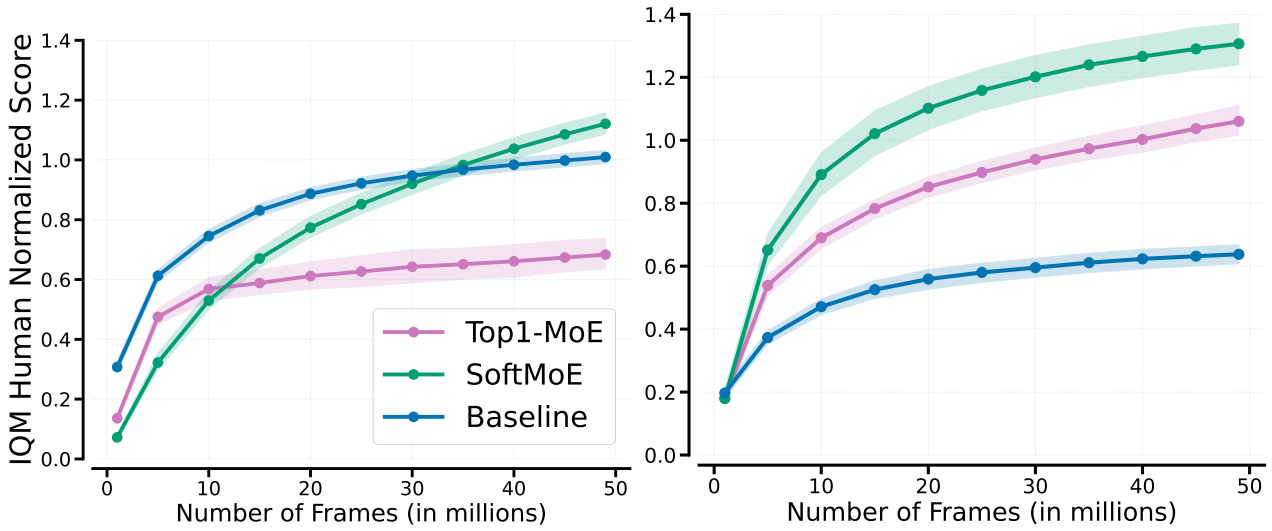

Figure 16. **Normalized performance across** 26 **Atari games for DrQ(ϵ) (left) and DER (right)**, with the ResNet architecture (Espeholt et al., 2018) and 4 experts. Soft MoE not only remains generally stable with more training, but also attains higher final performance. We report interquantile mean performance with error bars indicating 95% confidence intervals.

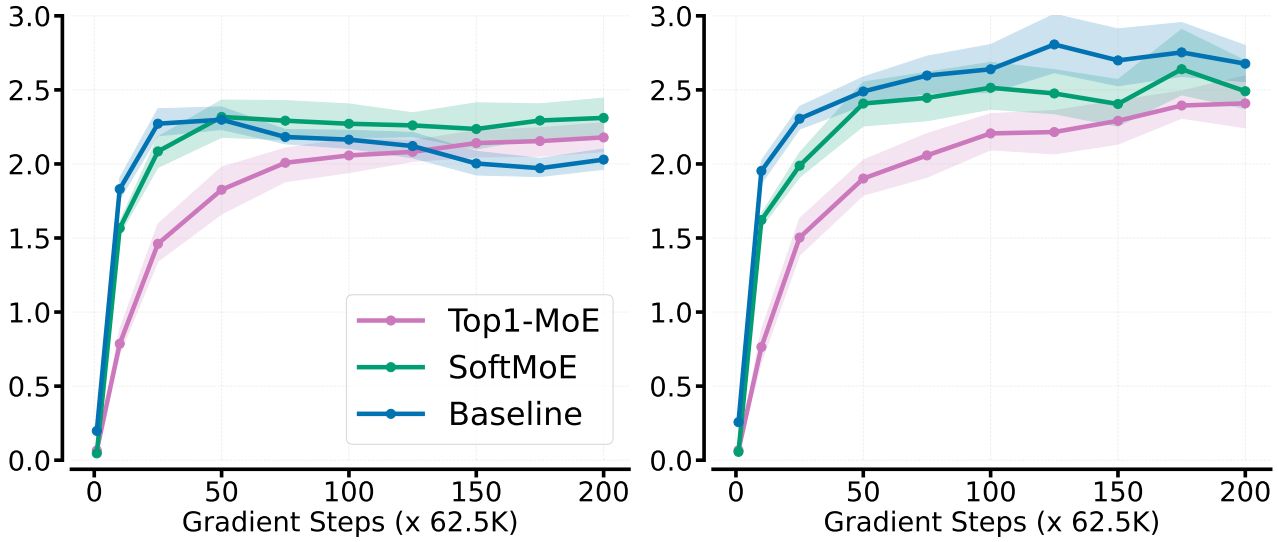

Figure 17. **Normalized performance across** 17 **Atari games for CQL+C51**. x-axis represents gradient steps; no new data is collected. **Left:** 10% and **Right:** 50% uniform replay. We report IQM with 95% stratified bootstrap CIs (Agarwal et al., 2021)

.

D. Varying Impala filter sizes

When dealing with small models, it's common to scale them up to enhance performance. This makes the scaling strategy crucial for balancing accuracy and efficiency. For Convolutional Neural Networks (CNNs), traditional scaling methods usually emphasize model depth, width, and input resolution (Ding et al., 2022), as well as the *filter*. The default filter size is 3x3 for the Impala CNN, and we ran experiments with and without SoftMoE using 4x4 and 6x6 filters to investigate the filter size scaling benefits. In both cases, SoftMoE outperforms the baseline.

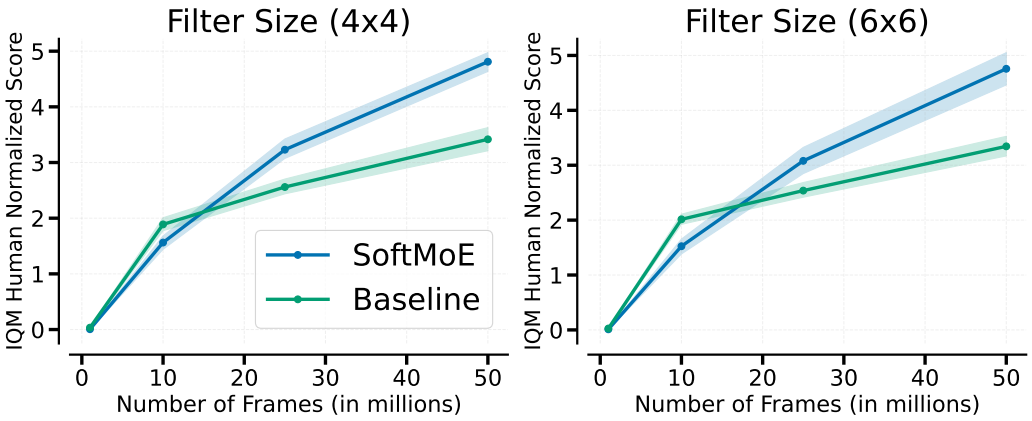

Figure 18. **Normalized performance across** 20 **Atari games with the ResNet architecture.** SoftMoE achieves the best results in both scenarios; default filter size (3x3) is increased to (4x4) and (6x6).

E. Measuring runtime

We plotted IQM performance against wall time, instead of the standard environment frames. SoftMoE and baseline have no noticeable difference in running time, whereas Top1-MoE is slightly faster than both.

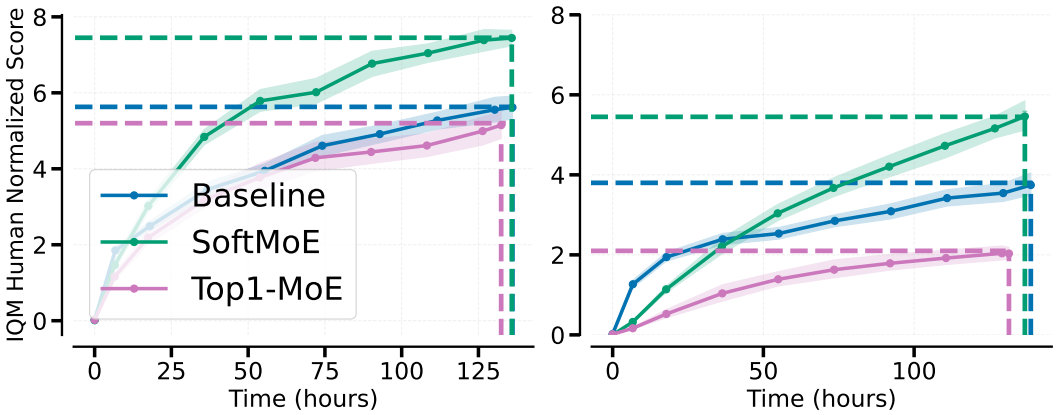

Figure 19. **Measuring wall-time versus IQM of human-normalized scores** in Rainbow over 20 games. **Left:** ImpalaCNN and **Right:** CNN network. Each experiment had 3 independent runs, and the confidence intervals show 95% confidence intervals.

F. Experiments with PPO

Based on reviewer suggestions, we have run some initial experiments with PPO and SAC on MuJoCo. We have not observed significant performance gains nor degradation with SoftMoE; with Top1-MoE we see a degradation in performance, similar to what we observed in our submission. We see a few possible reasons for the lack of improvement with SoftMoE:

1. For ALE experiments, all agents use Convolutional layers, whereas for the MuJoCo experiments (where we ran SAC and PPO) the networks only use dense layers. It is possible the induced sparsity provided by MoEs is most effective when combined with convolutional layers.
2. The suite of environments in MuJoCo are perhaps less complex than the set of experiments in the ALE, so performance with agents like SAC and PPO is somewhat saturated.

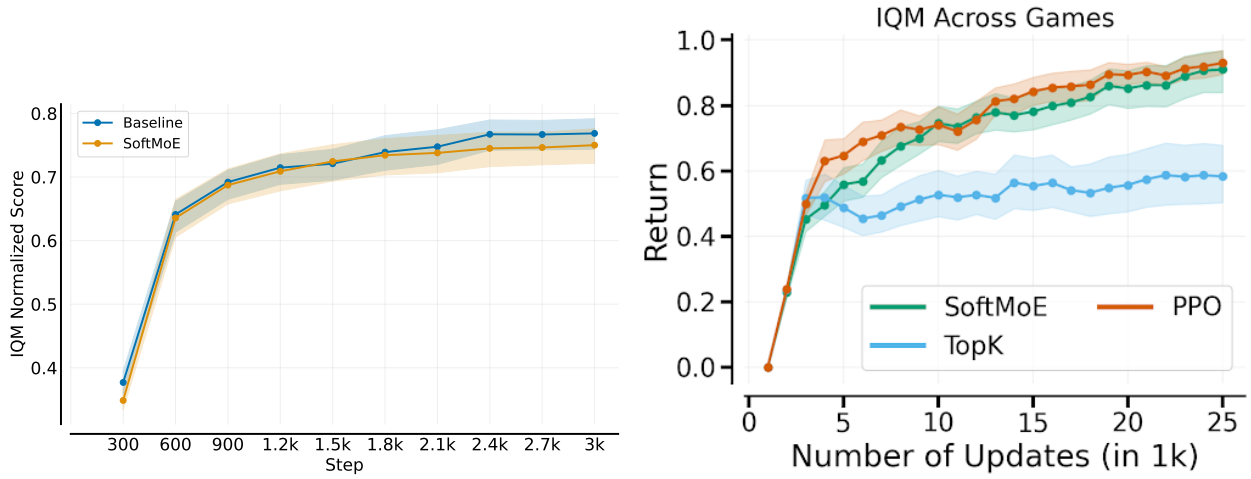

Figure 20. **Left:** Evaluating SAC with SoftMoE on 28 MuJoCo environments and **Right:** Evaluatin PPO on 9 MuJoCo-Brax environments. SoftMoEs seems to provide no gains nor degradation, whereas TopK seems to degrade performance (consistent with paper's findings). MuJoCo scores are normalized between 0 and 1000, with 5 seeds each; error bars indicate 95% stratified bootstrap confidence intervals. MuJoCo-Brax scores are normalized with respect to Jesson et al. (2023).