SCALEMOE: MIXTURE-OF-EXPERTS FOR SCALABLE CONTINUOUS CONTROL IN ACTOR–CRITIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling model size has been a key driver of progress in supervised learning, but remains a challenge in deep reinforcement learning (RL), where naively increasing the parameters of actor-critic networks often leads to instability and performance degradation. While recent architectures like SimBa and BRC have shown that careful inductive biases can enable positive scaling in continuous control, they remain monolithic, activating all parameters for every input. In this work, we introduce **ScaleMoE**, an architecture that integrates Mixture-of-Experts (MoE) modules into both the actor and critic of state-of-the-art continuous control algorithms. This approach effectively turns parameter growth into consistent performance gains. We propose two integration strategies: (i) *output-level gating*, where a learned gating network selects the top-K expert actors and critics per state and merges their outputs (policy means, variances, and Q-values) via gating weights; and (ii) *feature-level gating*, where experts produce penultimate features that are combined by top-K gating and passed through a shared output layer for both policy and value predictions. We implement ScaleMoE on a single-task actor–critic baseline (SimBa) and a multi-task baseline (BRC), two representative monolithic scaling RL methods. Experiments on the DeepMind Control Suite, MetaWorld and HumanoidBench demonstrate improved returns as the number of experts increases. In multi-task settings, ScaleMoE with smaller experts matches or outperforms a larger monolithic network with substantially less model parameters. Our findings indicate that MoE offers an effective and compute-efficient scaling axis for deep RL in continuous control, narrowing the gap with supervised learning.

1 INTRODUCTION

Deep reinforcement learning (RL) has yet to realize the scaling gains that have propelled advances in computer vision and NLP. Increasing model size in RL often *hurts* performance rather than improving it, as naive width or parameter scaling of actor–critic networks can induce overfitting and instability, obscuring clear scaling laws. For example, (Andrychowicz et al., 2021; Bjorck et al., 2021) report that standard agents plateau or degrade as parameters increase. To counteract this, researchers have introduced architectures that embed inductive biases or regularization. SimBa (Lee et al., 2024) shows that architectural modifications, e.g., observation normalization and residual connections, can yield *positive* scaling on continuous control benchmarks. In the multi-task regime, BRC (Nauman et al., 2025) leverages a high-capacity distributional critic and regularization to achieve state-of-the-art results across many tasks. While these studies demonstrate that scaling is possible with careful design, their *monolithic* nature, i.e., activating all parameters for every input, imposes a fundamental limit, causing performance to plateau as model size increases beyond a certain point.

In contrast, Mixture-of-Experts (MoE) enables conditional computation: a gating function activates only a subset of expert networks per input, decoupling total capacity from per-sample compute (Ye & Xu, 2023). MoE has been highly effective for scaling in NLP and vision by expanding parameters without proportional increases in computation (Zhou et al., 2022; Riquelme et al., 2021). Applying MoE to RL is therefore a natural step toward scalable performance. Recent studies demonstrate promising results in discrete-action settings: Obando-Ceron et al. incorporate soft MoE layers into value-based agents and observe improved scaling on Atari, while Willi et al. show that experts can

specialize in multi-task RL to boost sample efficiency. In continuous control, several works explore MoE from different perspectives, including interpretable expert gating (Akrour et al., 2021), task-oriented perturbations for visual RL (Huang et al., 2025), and decision transformer scaling (Kong et al., 2025). However, these approaches primarily focus on policy specialization without joint actor-critic scaling. The most relevant prior work (Hendawy et al., 2023) employs orthogonal constraints to enhance actor and critic expert diversity but does not investigate scaling effects. Crucially, existing MoE-in-RL efforts have not fully discovered the potentials of scalable continuous control with joint MoE scaling and optimization of actor and critic networks.

We introduce **ScaleMoE**, a MoE architecture for *continuous-action* deep RL. ScaleMoE replaces a single actor and critic with multiple expert actors and critics, and employs a learned gating mechanism that dynamically selects and combines expert outputs. We present two integration variants suited to actor–critic learning: (i) *output-level* gating, which fuses Gaussian policy parameters (means and variances) and aggregates Q-values across the top-K experts; and (ii) *feature-level* gating, which mixes penultimate-layer features with a shared output head. We instantiate ScaleMoE on a single-task Simba (Lee et al., 2024) agent and on a multi-task BRC (Nauman et al., 2025) agent, keeping the underlying learning algorithms unchanged, highlighting the method's generality. Across DeepMind Control Suite hard tasks (dog and humanoid domains) and the HumanoidBench of high-dimensional continuous control, ScaleMoE delivers **significant scaling of return** with increasing model size than strong monolithic baselines. For example, scaling to eight actor and critic experts yields an average **60%** improvement over the single-expert baseline on *humanoid-run*, whereas simply scaling a single network with an equivalent parameter budget provides limited gains or even degrades performance. We also study computational trade-offs: while increasing the number of experts raises training time and memory, MoE with smaller experts can outperform a single large network at far lower resource cost. Notably, an eight-expert or sixteen-expert BRC (each critic expert width 512) surpasses a single-network BRC with critic width 4096 in overall return while using roughly one-eighth/one-fourth of the parameters. We further analyze plasticity and specialization: ScaleMoE exhibits a substantially lower dormant-neuron ratio (Sokar et al., 2023), indicating more effective capacity utilization. Meanwhile, the gating policy induces clear expert specialization by task context in the multi-task setting. These results suggest that conditional computation via MoE offers a practical path to unlocking parameter scaling in RL continuous control.

Our contributions are threefold:

- **ScaleMoE for continuous control.** To the best of our knowledge, ScaleMoE is the first to investigate the scaling phenomenon in continuous control through the dual application of Mixture-of-Experts modules to both the actor and the critic, enabling a systematic study of how model capacity influences performance.
- **Actor–critic-tailored integrations with broad applicability.** We develop output-level gating and feature-level gating, and instantiate them as drop-in modules for SimBa (single-task) and BRC (multi-task) without modifying their learning algorithms.
- **Validated scaling, plasticity, and efficiency.** On DMC hard tasks, MetaWorld and HumanoidBench, ScaleMoE exhibits superior *scaling of return*, lower dormant-neuron ratios with clear expert specialization, and favorable compute/memory trade-offs.

2 Related Work

2.1 Scaling Up Deep RL Architectures

Architectural innovations have enabled more stable training of wider networks in deep RL. Residual connections and normalization layers mitigate gradient pathologies in value functions, e.g., spectral normalization in SpectralNet (Bjorck et al., 2021) and batch renormalization in CrossQ (Bhatt et al., 2024)—while simplicity-biased designs such as SimBa (Lee et al., 2024) and SimBaV2 (Lee et al., 2025) reduce overfitting in overparameterized policies. BRC (Nauman et al., 2024) pointed out that strong regularization paired with optimistic exploration could lead to effective scaling of the critic networks. Further, Nauman et al. proposed to apply BRO architecture into multi-task RL with the design of categorical loss of value networks. In contrast, systematic *depth* scaling remains under-explored for reward-guided training. Notable exceptions demonstrate feasibility in niche settings: DT-VINs (Wang et al., 2024) train planning networks with up to 5,000 layers via adaptive highway

losses, and self-supervised RL (Wang et al., 2025) reports emergent locomotion with 1,024-layer networks. However, these monolithic architectures often struggle to further improve performance as model size increases beyond a certain point.

2.2 Mixture-of-Experts (MoE)

Mixture-of-Experts (MoE) scales parameters through *expert routing*, activating only a subset of experts per input to decouple model capacity from per-sample compute (Ye & Xu, 2023). For discrete control, MoE has outperformed dense layers on Atari (Obando-Ceron et al., 2024) and shows expert specialization and improved sample efficiency in multi-task settings (Willi et al., 2024). In continous control, Akrour et al. proposed a MoE framework for continuous actions using interpretable experts with probabilistic gating. Huang et al. introduce task-oriented perturbations for visual RL with MoE, focusing on representation learning in pixel-based domains. Kong et al. scale decision transformers via MoE for massive multi-task learning. While these demonstrate the versatility of MoE architectures, they prioritize only policy specialization without joint actor-critic scaling. Most relevant to our method is (Hendawy et al., 2023), which focus on employ orthogonal constraints to enhance expert diversity, but scaling effect is not investigated in this work. Our work distinguishes itself by introducing MoE to *both* actor and critic networks to achieve *scalable* continuous control, with top-K gating mechanisms compatible with Gaussian policies and value aggregation.

Ensemble methods offer an alternative to single-model scaling by aggregating diverse components. Q-ensembles address exploration (Osband et al., 2016) and estimation biases (Lan et al., 2020). To improve sample efficiency, REDQ (Chen et al., 2021), DroQ (Hiraoka et al., 2022) and AQE (Wu et al., 2022) inject uncertainty with limited additional compute. Policy ensembles also show promise: ACE (Zhang & Yao, 2019) combines actor outputs via tree search, POLTER (Schubert et al., 2023) distills historical policies for unsupervised RL, and SAPG (Singla et al., 2024) partitions rollouts across parallel policies and reweights data via importance sampling. Unlike these ensembles that activate all components, MoE provides conditional computation benefits by activating only sparse subsets, achieving ensemble-like diversity at lower computational cost.

3 ScaleMoE: Scalable Mixture-of-Experts into Actor-Critic

Our goal is to incorporate a Mixture-of-Experts module into an actor-critic agent, so that the agent's capacity (number of parameters) can be increased via multiple expert networks, while a gating mechanism decides which experts to use for each state. We first describe the generic MoE actor-critic architecture, then detail two variants of the gating mechanism, and finally outline specific architectural choices for single-task and multi-task implementations.

3.1 MoE Actor-Critic Architecture

Multiple Actors and Critics: Instead of a single policy network (actor) and single value network (critic), ScaleMoE maintains a set of N actor networks $\{\pi_i\}_{i=1}^N$ and N critic networks $\{Q_i\}_{i=1}^N$ (or value functions V_i, depending on the algorithm). Each actor π_i outputs the actions of a stochastic policy or a deterministic policy. For stochastic policy, the output is typically the mean $\mu_i(s)$ and standard deviation $\sigma_i(s)$ of a Gaussian distribution for action a given state s. In the following part, we will describe our method based on stochastic policy. Each critic Q_i outputs an estimate of the return (e.g. $Q_i(s, a)$). All experts share the same network architecture design as the baseline, but potentially with **reduced per-expert width** so that total parameters remain reasonable. For example, if the baseline critic has width 1024, using $N = 4$ experts of width 512 each yields roughly comparable total parameters. In single-task ScaleMoE, we keep each expert's architecture identical to the baseline's, whereas in multi-task ScaleMoE we often shrink individual expert size to allow a larger number of experts without blowing up computation (see Section 4).

Gating Network: A learnable gating network $G(s)$ takes the state or some representation of the state as input and produces a set of N gating scores $\{g_1, \ldots, g_N\}$ for the experts. We implement G as a small feedforward network (two layers) that outputs N values, followed by a softmax to obtain

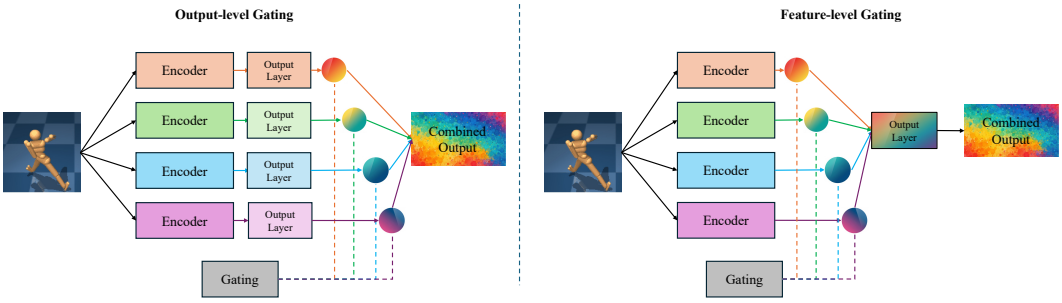


Figure 1: The illustration of two different level of gatiing in ScaleMoE. The architecture works both for actor and critic. The encoder part used in our experiments are Simba and BRC encoder.

gating weights:

$$w_i(s) \;=\; \frac{\exp(g_i(s))}{\sum_{j=1}^{N} \exp(g_j(s))}, \quad i = 1, \ldots, N. \tag{1}$$

Here $w_i(s) \in (0, 1)$ and $\sum_i w_i(s) = 1$. To encourage specialization and efficiency, we use a **top-K gating** strategy: the gating network selects the K highest-weight experts for the current state, and sets the other weights to zero. The chosen weights are then renormalized to sum to 1, denoted $\tilde{w}_i$ for selected experts. This means at most K experts are active per state, reducing computation if $K \ll N$. Given the gating output, the actor-critic combine the expert networks' outputs. We consider two approaches, illustrated in Figure 1.

3.2 Output-Level Gating (Late Fusion)

In this approach, each expert actor π_i independently outputs a distribution $\mathcal{N}(\mu_i(s), \Sigma_i(s))$, where Σ_i is typically diagonal with elements $\sigma_{i,1}^2, \ldots, \sigma_{i,d}^2$ for d-dimensional action. Similarly each expert critic produces a scalar $Q_i(s, a)$. The gating weights then fuse these outputs into a single policy and single value:

Mixture of policies: We form the final policy as a Gaussian with mean and variance given by a weighted combination of the expert means and variances. Specifically, if $\mathcal{K}(s)$ is the set of top-K experts for state s, we compute

$$\mu_{\text{final}}(s) \;=\; \sum_{i \in \mathcal{K}(s)} \tilde{w}_i(s)\, \mu_i(s), \quad \sigma^2_{\text{final}}(s) \;=\; \sum_{i \in \mathcal{K}(s)} \tilde{w}_i(s)\, \sigma_i^2(s), \tag{2}$$

for each component of the action vector. This yields a single Gaussian policy that approximately represents the mixture of expert policies. Intuitively, the gating network "blends" the mean actions of the selected expert policies, e.g., for $K = 1$, this simply picks the single best expert's policy.

While we aggregate means and variances independently, we note that the resulting mixture is not a true Gaussian mixture due to ignored covariances. However, in practice, the diagonal covariance assumption is common in policy parameterization, and our experiments show the approach remains highly effective.

Mixture of Q-values: For the critic, we similarly take the weighted sum of the selected experts' estimates:

$$Q_{\text{final}}(s, a) \;=\; \sum_{i \in \mathcal{K}(s)} \tilde{w}_i(s)\, Q_i(s, a). \tag{3}$$

The final Q is a convex combination of the expert critics' outputs. Since each Q_i is trained to estimate returns, the mixture Q_{final} remains a valid estimate (and is used in the actor-critic loss as usual).

Training. All expert networks and the gating network are trained jointly. We apply standard RL losses $\mathcal{L}_{\text{RL}}$ (e.g., actor and critic losses) using the final fused Q_{final} and the final policy. Note that

gradients are only backpropagated through the top-K experts, while non-selected experts receive no gradient update for the current sample. This encourages specialization while avoiding over-training of rarely used experts.

To ensure that all experts are trained more evenly, we also add two *auxiliary regularizers*. Let $p_i^{(b)}$ denote the (post-softmax) gating probability of expert $i \in \{1, \dots, N\}$ for sample $b \in \{1, \dots, B\}$ in a minibatch, and define the batch-averaged usage $u_i = \frac{1}{B} \sum_{b=1}^{B} p_i^{(b)}$. We use a *load-balancing* loss (negative entropy of expert usage) to encourage uniform utilization across experts,

$$\mathcal{L}_{\text{lb}} = \sum_{i=1}^{N} u_i \log\big(u_i + 10^{-10}\big), \tag{4}$$

and an *importance* loss to discourage overly peaked gating and reduce single-expert dominance,

$$\mathcal{L}_{\text{imp}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{b=1}^{B} \big(p_i^{(b)}\big)^2. \tag{5}$$

The total objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RL}} + \lambda_{\text{lb}} \mathcal{L}_{\text{lb}} + \lambda_{\text{imp}} \mathcal{L}_{\text{imp}}, \tag{6}$$

where λ_{lb} and λ_{imp} control the strength of the regularizers. Together they ensure all experts receive sufficient gradient coverage, akin to balanced assignment in MoE literature (Zoph et al., 2022).

3.3 Feature-Level Gating (Early Fusion)

Our second integration method moves the gating operation to an earlier point in the network. Instead of merging final outputs, we merge the experts' *internal representations* before the output layer. Concretely, we let each expert produce a **penultimate-layer feature** $h_i(s)$ (e.g. the activations of the last hidden layer in the actor/critic network). The gating network $G(s)$ again selects top-K experts and provides weights $\tilde{w}_i(s)$. We then compute a gated feature as the weighted sum of expert features:

$$h_{\text{gated}}(s) = \sum_{i \in \mathcal{K}(s)} \tilde{w}_i(s)\, h_i(s). \tag{7}$$

This aggregated feature is then passed through a **shared output layer** that is used to produce the policy or value outputs. In practice, this means we have one additional fully-connected layer or small neural network that takes $h_{\text{gated}}(s)$ and outputs *both* $\{\mu_{\text{final}}(s), \sigma_{\text{final}}(s)\}$ for the actor and $Q_{\text{final}}(s, a)$ for the critic.

Training: Training is similar, except now the gating influences the experts' contributions at the feature level. The shared output layer's gradients flow back into the selected experts' features h_i and into the gating network. This method ties the actor and critic experts more closely as they share the gated representation. In addition, this approach uses fewer total parameters than output-level gating, because the final layer is not duplicated across N experts.

3.4 Single-Task vs Multi-Task Implementation Design

ScaleMoE is a algorithm-agnostic general module and we apply it to two different base algorithms SimBa and BRC.

Single-Task (SimBa + MoE): In this setting, we integrate MoE into a DDPG-style off-policy agent enhanced with the SimBa architecture. The baseline SimBa network has certain normalization and residual blocks, which we preserve in each expert. All hyperparameters (learning rates, layer sizes, etc.) are kept the same as in the original DDPG+SimBa setup, except for the addition of multiple networks and update-to-data ratio. We typically increase the number of actors and critics N (e.g. from 1 to 4 or 8), while keeping each expert's size the same as the original network. This increases total parameters roughly linearly with N. For gating, we found a small network (two-layer MLP with 64 units) was sufficient to act as the gating network $G(s)$. The gating network in single-task case takes the state observation as input. It outputs N gating logits as described earlier.

Multi-Task (ScaleMoE BRC): The multi-task setting uses the BRC agent as the backbone. BRC is a SAC-style actor-critic method that can be trained on many tasks simultaneously by using a large categorical value function conditioned on a task embedding. In our implementation, we extend BRC by introducing multiple critic experts and actor experts. We found it beneficial to **reduce the per-expert network width** compared to the original BRC architecture, in order to accommodate more experts without excessive increase in total parameters. For example, the original BRC critic has a layer width 4096 neurons; in ScaleMoE-BRC we use 8 critic experts each of width 512, which yields much fewer total neurons. For the actor network, we keep the width 256 unchanged. Despite each expert being smaller, the combined model has high capacity due to the number of experts. The gating network in multi-task takes as input both the state and the task identifier or task embedding. In practice, we concatenate the task embedding provided by BRC with the state features as input to the gating network. The rest of the training procedure follows BRC's training protocol.

Algorithmic Pseudocode: *Due to space limitation, we summarize at a high level:* At each training step, for a given state (and task, if multi-task), the gating network computes $w_i(s)$ and selects top-K experts. The actor outputs (either each expert's policy or the shared output policy) produce the final action distribution, from which an action is sampled for environment interaction (for on-policy steps) or used in critic loss (for off-policy update). The critic outputs from experts are combined to get Q_{final} which is used in the loss (e.g. temporal difference error). Gradients update the selected top-K experts and gating network. Detailed algorithm is given in the Appendix B.

4 Experiments

4.1 Benchmarks and Tasks

We use the DeepMind Control Suite (DMC) Hard benchmark tasks (Tassa et al., 2018), including DMC DOGS (quadruped) and DMC HUMANOIDS (humanoid). For ScaleMoE with Simba, we train each task individually, while for BRC, we treat the entire DMC HUMANOIDS and DMC DOGS as a multi-task learning problem. Additionally, we test multi-task learning using Humanoid-Bench (Sferrazza et al., 2024), which includes 20 tasks for a simulated humanoid robot (Unitree H1), all trained concurrently with a task indicator. We also evaluate on MetaWorld (McLean et al., 2025a), a benchmark with 50 tasks for a robotic arm, treating them as a single multi-task problem with concurrent training.

4.2 Baselines

Baselines: For single-task experiments on DMC, our baseline is SimBa (DDPG) without MoE. Note that all the reported results of Simba is under the same update-to-data ratio with ScaleMoE. For multi-task DMC and HumanoidBench, the baseline is BRC, a strong multi-task agent that achieved state-of-the-art results on these benchmarks. We compare against BRC configured with its recommended network size (critic width 4096) and small sizes (critic width 512) to see if ScaleMoE is more parameter-efficient than simply using a bigger MLP. All the reported mean results and standard error of the proposed method and baselines are averaged across eight seeds.

4.3 Results: Scaling Performance with Number of Experts

Single-Task (Seven independent DMC Hard): Figure 2 illustrates the scaling performance on the seven DMC hard tasks. With the baseline (SimBa) architecture, increasing the network size to critic hidden size of 512 had been shown to improve performance on these tasks (Lee et al., 2024), but naive scaling beyond that led to plateaus or degradation as indicated using the gray dashed lines. In contrast, ScaleMoE shows a clear upward trend as we add more experts. For example, going from 1 expert (original Simba) to 4 experts (each same size as baseline) on ScaleMoE-Feature improves the average IQM by 32% on *dog-run*. On *humanoid-run*, using 8 experts yields gains up to 60%. Note that in each ScaleMoE method presented in Figure 2 uses a half active set out of all experts, which is sufficient to achieve overall scaling in performance. This suggests that the gating network is effectively routing each state to a good candidates of experts. We also note that both gating strategies (output-level and feature-level) perform similarly on single tasks, with a slight advantage for feature-level gating in final performance.

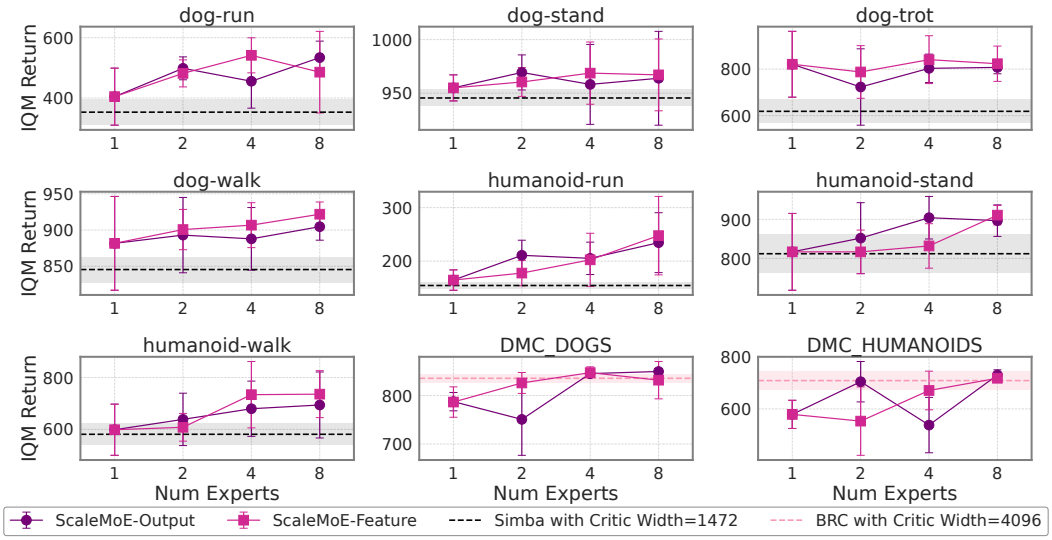
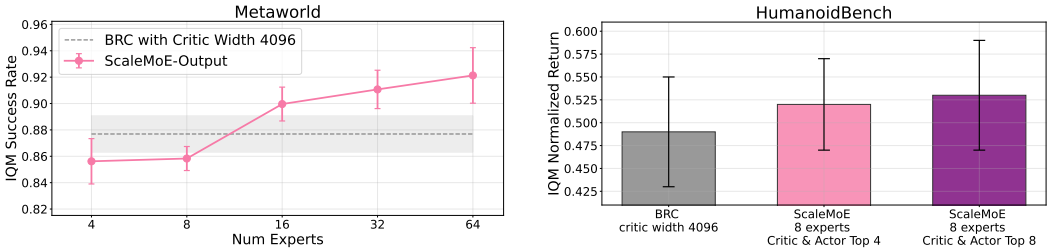


Figure 2: Scaling performance as the number of experts increases. Solid lines indicate results obtained by expanding from an identically sized monolithic architecture (critic hidden size of 512) to different number of experts. The black dashed line denotes the Simba with critic hidden size of 1472, which matches the model size of ScaleMoE with 8 experts of critic hidden size of 512. The gray dashed line denotes the original BRC configuration, i.e., a critic hidden size of 4096.

Multi-Task (DOGS 3 tasks and HUMANOIDS 4 tasks): The last two plots in Figure 2 also presents the average success rates across the three tasks in DMC DOGS and the four tasks in DMC HUMANOIDS, comparing our method with the baseline (BRC). The baseline BRC, employing a very large critic network (width 4096), achieves IQM of 835 on DOGS and 709 on HUMANOIDS, as indicated by the pink dashed lines. In contrast, our ScaleMoE-BRC model with 8 experts (each of width 512) attains final average IQM scores of 849 and 730, respectively. This results showed that ScaleMoE exceed the performance of the monolithic network. We also provide the model size comparison in Appendix F. ScaleMoE with 8 experts (roughly 32M parameters) uses only 12.5% of the parameters of the default monolithic BRC (roughly 256M), achieving a significant reduction in model size. This result is notable: instead of scaling a single network to 4096 units, employing *multiple smaller experts with a gating mechanism not only reduces memory usage but also leads to better performance*.



(a) Scaling N on Metaworld 50 envs. (b) Scaling K on HumanoidBench 20 envs.

Figure 3: Scaling performance on multi-task environments.

Multi-Task (Metaworld (50 tasks) and HumanoidBench (20 tasks). We conduct additional experiments on the MetaWorld benchmark, which involves 50 tasks. We activate only half of the total experts to scale the total expert number N. As shown in Figure 3, ScaleMoE with more than 16 experts (IQM $\geq$ 0.9) significantly outperforms BRC-4096 (IQM = 0.88), demonstrating clear scaling benefits. We also validate the effectiveness of scaling k on HumanoidBench, where ScaleMoE with 8 experts and full activation (Top-8/Top-8) achieves an IQM normalized return of 0.53, slightly surpassing BRC (critic width 4096) at 0.49 (+8% relative improvement) with fewer parameters. See Appendix K for training curves.

4.4 Analysis: Expert Utilization and Network Plasticity

Expert Utilization. Figure 4 illustrates the **expert activation frequencies** observed during the evaluation of ScaleMoE (with 8 experts, Critic Top 4, Actor Top 4) on representative tasks from HumanoidBench. The activation patterns are task-specific and reflect clear functional specialization: in locomotion tasks (walk, run), experts 1 to 3 (and occasionally expert 4) are predominantly selected, with expert 2 being the most frequently activated. For manipulation-oriented tasks (insert, reach), experts 0, 1, 2, and 6 are emphasized. Balance and sit tasks exhibit distinct routing behaviors: balance task primarily engages experts 1, 3, 4, 5, while sit task relies mainly on experts 0, 3, 4 and 7. Across tasks, the activation mass is typically distributed among 3 to 4 experts, each accounting for approximately 20–25% of the selections. This suggests that the gating mechanism effectively blends complementary skills rather than collapsing to a single dominant expert. Such task-adaptive co-activation underscores a key advantage of the MoE architecture: the gating network decomposes the multi-task problem into distinct subskills (e.g., locomotion, manipulation, and balance) and dynamically recombines them through top-K mixing.

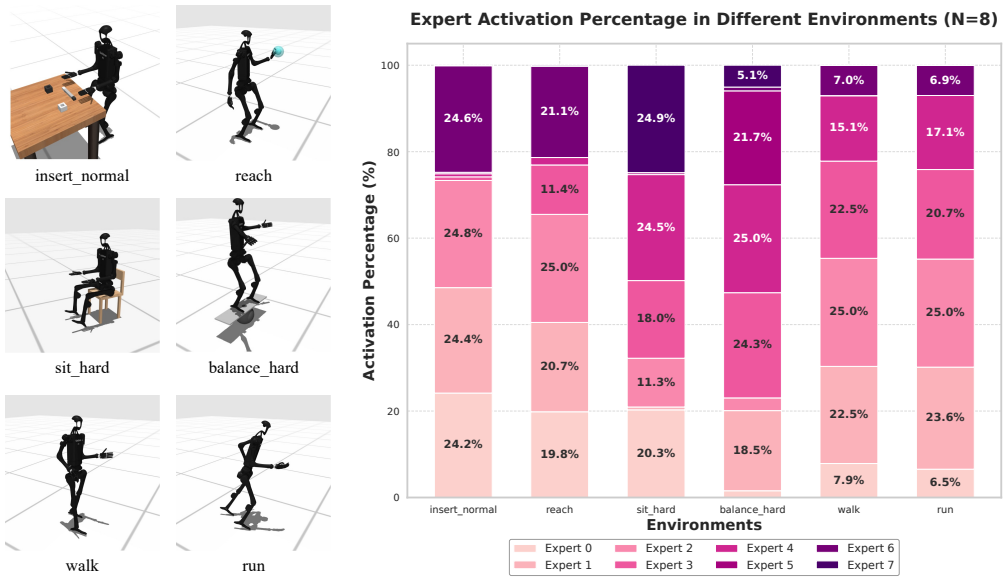


Figure 4: Expert activation heatmap on six representative tasks of HumanoidBench. Each cell shows the percentage of gating selections for an expert on a task. Percentage larger than 5% is marked.

Network Plasticity. We then computed the dormant neuron ratio (DMR) for ScaleMoE (DDPG-Simba) with different activated number of critic expert during training in Figure 5. For the baseline single network Simba, we found a high dormant ratio, indicating a lot of capacity was unused, consistent with findings that networks in RL often don't utilize all units. In contrast, ScaleMoE with 8 experts had a significantly lower dormant ratio. Even with a conservative activation of $K = 2$ experts, the dormant ratio dropped to approximately 6% at step 5e5. When we increased the number of activated experts to 4 and 8, the dormant ratio further decreased, effectively engaging a majority of the available parameters in processing each input. A critical finding is that this increased activation does not lead to the loss of plasticity or stability typically associated with oversaturating a monolithic network. Combined results in Figure 4, we posit that the gating network essentially learns to decompose the input space, fostering **functional specialization** among experts. Each expert thereby specializes in distinct regions or modes of the data distribution. This structure enables coherent parameter updates and learns a more powerful internal representation, explaining the robust performance gains across diverse domains. The lower dormant ratio thus signifies not merely more active parameters, but the emergence of an efficiently organized computational system. Some theoretical intuition are presented in Appendix M.

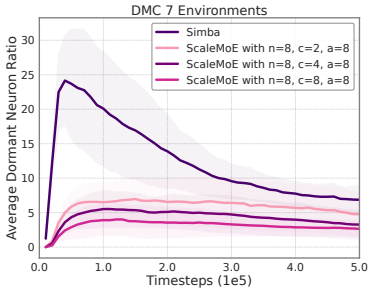


Figure 5: DMR during training.

5 Ablation Study

Gating Position (Output vs Feature-Level): We compare the two integration strategies on the DMC (Figure 2). On single-task DMC, feature-level (early-fusion) gating yields overall higher returns than output-level (late-fusion) gating. We conjecture that the shared output head in early fusion acts as a useful inductive bias: through a shared representation, it reduces variance in actor's Gaussian parameterization and critic's value estimation. In multi-task experiments, however, the two strategies perform comparably. A hypothesis is that the expressivity of late fusion are largely offset by the parameter efficiency of early fusion. Thus, we favor feature-level gating for single task, while for multi-task settings we prefer the more easy-to-implement output-level gating variant.

Number of Active Experts (K): Figure 6 summarizes a sweep over $K \in \{1, 2, 4, 8\}$ with $N = 8$ experts of ScaleMoE-Output on both signle-task and multi-task DMC. The overall best setting is fully expert activation. A lower-compute choice Top-2 and Top-4 on both actor and critic already attains satisfying performance. In contrast, when $K = 1$, many experts receive little to no gradient, leading to under-training and weak performance. These patterns suggest two prctical insights. First, avoid $K = 1$, i.e., using at least two active experts per side prevents expert under-training and stabilizes learning. Second, scale actor and critic K in tandem could bring more scaling benefits.

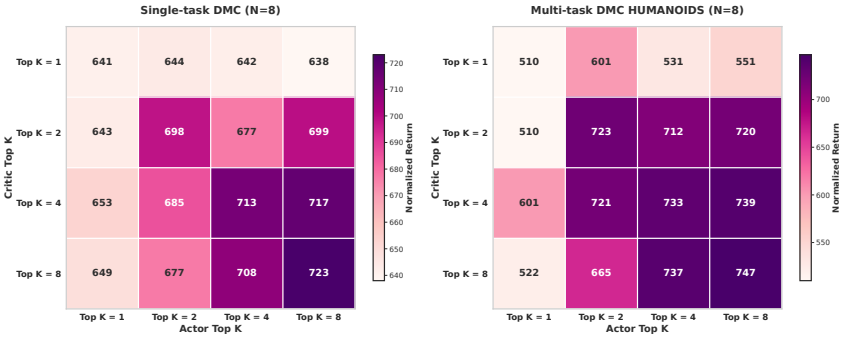


Figure 6: Normalized returns across different Top-K configurations on single tasks and multi tasks.

Scaling Number of Experts (N): We also ablated N (total experts) while fixing $K = 2$ on DMC using ScaleMoE (Simba). Increasing the number of experts from $N = 2 \to 4 \to 8$ yields mean final scores of 675, 697, and 698, respectively. Thus, gains are monotonic but saturate around $N \approx 4$ under a fixed training budget. We hypothesize that with small K, additional experts are under-utilized: experience is partitioned more finely while only two experts receive gradients per state. In practice, when scaling N, one should also consider increasing K, extending training steps, or shrinking per-expert width to maintain sufficient data and gradient coverage per expert.

Practical Guidance

(1) Use feature-level gating for single-task and output-level for multi-task settings.
(2) Given N, scale $K \geq 2$ in tandem for both actor and critic for higher score.
(3) When scaling N, also adjust K, training steps, or network size to maintain expert utilization.

6 Conclusion

We introduced ScaleMoE, a Mixture-of-Experts framework that brings conditional computation to continuous-control actor–critic RL. Increases model capacity in ScaleMoE enables scaling where monolithic networks plateau. The architecture is algorithm-agnostic: we demonstrated drop-in integrations with both single-task and multi-task baselines while preserving their original training procedures. Across diverse continuous-control benchmarks, ScaleMoE delivered higher returns, revealing expert specialization and improved network plasticity. These findings establish MoE as a promising, compute-efficient pathway toward larger and more capable reinforcement-learning agents. We hope that ScaleMoE inspires further research on bridging the gap between the scale of models in supervised learning and those in reinforcement learning, ultimately enabling agents that learn **better, faster, and across many domains**.

7 Ethics Statement

This work focuses on improving sample efficiency and computational scalability of model-based reinforcement-learning algorithms in simulated continuous-control domains. All experiments were conducted in silico on publicly available benchmarks (DeepMind Control Suite, HumanoidBench); no human or animal data were collected, and no sensitive information was involved. The open-source release will be under an MIT licence to facilitate transparent, reproducible research.

8 Reproducibility statement

For full reproducibility, we provide (i) a step-by-step description of the ScaleMoE algorithm in the main paper, (ii) hyper-parameters, network architectures, and compute budgets in the Appendix, and (iii) an open-source implementation that will be released on GitHub upon publication. All experimental environments (DeepMind Control Suite, HumanoidBench) are freely accessible.

References

Riad Akrour, Davide Tateo, and Jan Peters. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE TPAMI*, 44(10):6795–6806, 2021.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. In *ICLR*, 2021.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *ICLR*, 2024.

Nils Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning with spectral normalization. *NeurIPS*, 34:8242–8255, 2021.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W Ross. Randomized ensembled double q-learning: Learning fast without a model. In *ICLR*, 2021.

Ahmed Hendawy, Jan Peters, and Carlo D'Eramo. Multi-task reinforcement learning with mixture of orthogonal experts. In *ICLR*, 2023.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *ICLR*, 2022.

Suning Huang, Zheyu Aqa Zhang, Tianhai Liang, Yihan Xu, Zhehao Kou, Chenhao Lu, Guowei Xu, Zhengrong Xue, and Huazhe Xu. Mentor: Mixture-of-experts network with task-oriented perturbation for visual reinforcement learning. In *ICML*, 2025.

Yilun Kong, Guozheng Ma, Qi Zhao, Haoyu Wang, Li Shen, Xueqian Wang, and Dacheng Tao. Mastering massive multi-task reinforcement learning via mixture-of-expert decision transformer. In *ICML*, 2025.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *ICLR*, 2020.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter R Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. Simba: Simplicity bias for scaling up parameters in deep reinforcement learning. *arXiv preprint arXiv:2410.09754*, 2024.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025.

Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, KR Zentner, Ryan Julian, JK Terry, Isaac Woungang, et al. Meta-world+: An improved, standardized, rl benchmark. *arXiv preprint arXiv:2505.11289*, 2025a.

Reginald McLean, Evangelos Chatzaroulas, Jordan Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Multi-task reinforcement learning enables parameter scaling. *arXiv preprint arXiv:2503.05126*, 2025b.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. In *NeurIPS*, 2024.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. *arXiv preprint arXiv:2505.23150*, 2025.

Johan Obando-Ceron, Ghada Sokar, Timon Willi, Clare Lyle, Jesse Farebrother, Jakob Foerster, Karolina Dziugaite, Doina Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep rl. In *ICML*, pp. 38520–38540, 2024.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *NeurIPS*, 29, 2016.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *NeurIPS*, 34:8583–8595, 2021.

Frederik Schubert, Carolin Benjamins, Sebastian Döhler, Bodo Rosenhahn, and Marius Lindauer. Polter: Policy trajectory ensemble regularization for unsupervised reinforcement learning. *TMLR*, 2023.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation, 2024. URL https://arxiv.org/abs/2403.10506.

Jayesh Singla, Ananye Agarwal, and Deepak Pathak. Sapg: Split and aggregate policy gradients. In *ICML*, pp. 45759–45772. PMLR, 2024.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *ICML*, pp. 32145–32168. PMLR, 2023.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Kevin Wang, Ishaan Javali, MichaŁ Bortkiewicz, Benjamin Eysenbach, et al. 1000 layer networks for self-supervised rl: Scaling depth can enable new goal-reaching capabilities. *arXiv preprint arXiv:2503.14858*, 2025.

Yuhui Wang, Qingyuan Wu, Weida Li, Dylan R Ashley, Francesco Faccio, Chao Huang, and Jürgen Schmidhuber. Scaling value iteration networks to 5000 layers for extreme long-term planning. *arXiv preprint arXiv:2406.08404*, 2024.

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 *RLC*, 2024.

Yanqiu Wu, Xinyue Chen, Che Wang, Yiming Zhang, and Keith Ross. Aggressive q-learning with ensembles: achieving both high sample efficiency and high asymptotic performance. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Hanrong Ye and Dan Xu. Taskexpert: Dynamically assembling multi-task representations with memorial mixture-of-experts. In *ICCV*, pp. 21828–21837, 2023.

Dong Yin, Ramchandran Kannan, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. In *ICML*, pp. 7085–7094, 2019.

Shangtong Zhang and Hengshuai Yao. Ace: An actor ensemble algorithm for continuous control with tree search. In *AAAI*, volume 33, pp. 5789–5796, 2019.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *NeurIPS*, 35:7103–7114, 2022.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A THE USE OF LARGE LANGUAGE MODELS (LLMS)

We utilized LLMs to assist in polishing the language of our research paper and optimizing the clarity and impact of its graphical elements. In detail, we used LLMs to refine grammar, syntax, and academic style, ensuring clarity, conciseness, and a professional tone throughout the paper. We also employed AI tools to improve the design, layout, and visual impact of charts and illustrations, making data presentation more intuitive and engaging.

B ALGORITHM

Algorithm 1 ScaleMoE: Mixture-of-Experts Actor-Critic Training

Require: Number of experts N, top-K gating, replay buffer $\mathcal{D}$,
learning rates $\alpha_{\text{actor}}, \alpha_{\text{critic}}, \alpha_{\text{gate}}$

Initialize actor experts $\{\pi_i\}_{i=1}^{N}$, critic experts $\{Q_i\}_{i=1}^{N}$, gating network G
for each training iteration **do**
  Sample batch $\{(s_j, a_j, r_j, s'_j, \tau_j)\}_{j=1}^{B}$ from $\mathcal{D}$ # τ_j only in multi-task
  for each sample $j = 1 \dots B$ **do**
    Build gating input: $x_j = s_j$ (single-task) or $x_j = [s_j; \tau_j]$ (multi-task)
    $g_j \leftarrow G(x_j)$; $w_j \leftarrow \text{softmax}(g_j)$
    $\mathcal{K}_j \leftarrow \text{top-}K(w_j)$; renormalize $\tilde{w}_{j,i}$ for $i \in \mathcal{K}_j$
    if Output-Level Gating (Late Fusion) **then**
      for $i \in \mathcal{K}_j$ **do**
        $(\mu_{j,i}, \sigma_{j,i}) \leftarrow \pi_i(x_j)$; $Q_{j,i} \leftarrow Q_i(x_j, a_j)$
      end for
      $\mu_j \leftarrow \sum_{i \in \mathcal{K}_j} \tilde{w}_{j,i} \mu_{j,i}$; $\sigma_j \leftarrow \sum_{i \in \mathcal{K}_j} \tilde{w}_{j,i} \sigma_{j,i}$; $Q_j \leftarrow \sum_{i \in \mathcal{K}_j} \tilde{w}_{j,i} Q_{j,i}$
    else if Feature-Level Gating (Early Fusion) **then**
      for $i \in \mathcal{K}_j$ **do**
        $h_{j,i} \leftarrow \text{encoder}_i(x_j)$
      end for
      $h_j \leftarrow \sum_{i \in \mathcal{K}_j} \tilde{w}_{j,i} h_{j,i}$
      $(\mu_j, \sigma_j) \leftarrow \text{shared\_actor\_head}(h_j)$; $Q_j \leftarrow \text{shared\_critic\_head}(h_j, a_j)$
    end if
  end for
  Compute RL loss $\mathcal{L}_{\text{RL}}$ with Q_j, μ_j, σ_j
  $u_i \leftarrow \frac{1}{B} \sum_{j=1}^{B} w_{j,i}$ for $i = 1 \dots N$
  $\mathcal{L}_{\text{lb}} \leftarrow \sum_{i=1}^{N} u_i \log(u_i + \epsilon)$; $\mathcal{L}_{\text{imp}} \leftarrow \frac{1}{NB} \sum_{j=1}^{B} \sum_{i=1}^{N} w_{j,i}^2$
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{RL}} + \lambda_{\text{lb}} \mathcal{L}_{\text{lb}} + \lambda_{\text{imp}} \mathcal{L}_{\text{imp}}$
  Update all experts and G via $\nabla \mathcal{L}_{\text{total}}$
  Update target networks (if used)
end for

C HYPERPARAMETERS

Backbone algorithms: SimBa and BRC. Our work is built upon the JAX codes provided in official Simba (Lee et al., 2024) and BRC (Nauman et al., 2025) code repository. All experiments are run on NVIDIA GeForce RTX 4090 and H200 GPUs with INTEL(R) XEON(R) PLATINUM 8563C. For single task, we choose DDPG-Simba as our algorithm backbone. SimBa (Lee et al., 2024) is a single-task, off-policy actor–critic agent built on SAC that emphasizes a simplicity-biased architecture to stabilize training at larger widths. Concretely, SimBa employs observation normalization, residual MLP blocks, and lightweight normalization in the actor and critic to mitigate overfitting and gradient pathologies, enabling positive performance scaling on continuous-control benchmarks without altering the underlying SAC objective. Except for the MoE design and the updates per step, the other hyperparameters are kept the same with the original implementation in DDPG-Simba. The hyperparameters are shown in Table 1.

Table 1: ScaleMoE (DDPG-Simba) Hyperparameter Settings

	Hyperparameter	Value
Architecture (DDPG Simba)	Critic block type	SimBa Residual
	Critic num blocks	2
	Critic hidden dim	512
	Critic learning rate	$1e{-4}$
	Target critic momentum	$5e{-3}$
	Actor block type	SimBa Residual
	Actor num blocks	1
	Actor hidden dim	128
	Actor learning rate	$1e{-4}$
	Exploration noise	$\mathcal{N}(0,\, 0.1^2)$
	Batch size	256
	Optimizer	AdamW
	Optimizer momentum	(0.9, 0.999)
	Weight decay	$1e{-2}$
	Discount	Heuristic
	Updates per step	4
	Action repeat	2
	Clipped Double Q	False
MoE	Number of experts	1, 2, 4, 8
	Number of activated critic experts	1, 2, 4, 8
	Number of activated actor experts	1, 2, 4, 8
	weight of load-balancing loss (λ_{lb})	0.01
	weight of importance loss (λ_{imp})	0.01

For mutliple tasks, we choose SAC-BRC as our algorithm backbone BRC (Nauman et al., 2025) targets the multi-task regime with a high-capacity, regularized categorical value function (distributional critic) conditioned on task information, paired with a compatible actor. Its design increases critic expressivity while using regularization to reduce interference across tasks, yielding state-of-the-art results over large task suites. Except for the MoE design and the Critic hidden dimension, the other hyperparameters are kept the same with the original implementation in SAC-BRC. The hyperparameters are shown in Table 2.

Table 2: ScaleMoE (SAC-BRC) Hyperparameter Settings

	Hyperparameter	Value
Architecture (SAC BRC)	Critic block type	BroNet
	Critic depth	2
	Critic hidden dim	512
	Critic learning rate	$3e{-4}$
	Target critic momentum	$5e{-3}$
	Clipped Double Q	True
	Actor block type	BroNet
	Actor depth	1
	Actor hidden dim	256
	Actor learning rate	$3e{-4}$
	Temperature learning rate	$3e{-4}$
	V_{min}	-10
	V_{max}	10
	num atoms	101
	Buffer size per task	1e6
	Polyak τ	5e-3
	Target update frequency	1
	Batch size	1024
	Optimizer	AdamW
	Weight decay	$1e{-4}$
	Target entropy	$\lvert\mathcal{A}\rvert/2$
	Action repeat	1
	Discount	0.99
	Updates per step	2
MoE	Number of experts	1, 2, 4, 8
	Number of activated critic experts	1, 2, 4, 8
	Number of activated actor experts	1, 2, 4, 8
	weight of load-balancing loss (λ_{lb})	0.01
	weight of importance loss (λ_{imp})	0.01

D LIMITATIONS

While ScaleMoE shows promising results, there are several limitations to note. **First**, the improved performance comes at the cost of added complexity. Tuning an MoE architecture involves extra hyperparameters (number of experts N, gating network shape, K value, etc.), which could be daunting. **Second**, our experiments focused on continuous control tasks with state observations of modest dimensionality (mostly low-dimensional proprioceptive states, possibly some egocentric target encodings). We did not evaluate ScaleMoE on tasks with high-dimensional image observations; integrating MoE with deep convolutional encoders (e.g., for pixel-based RL) might pose additional challenges or require a different gating design. **Finally**, from a computational standpoint, while we demonstrated efficiency gains relative to a single huge network, training an MoE agent still requires more hardware resources than a standard small network. In settings where compute is very limited or real-time inference is required on embedded systems, an MoE might be impractical. Future work may explore distilling the MoE policy back into a smaller network for deployment.

E SCORE NORMALIZATION

We follow the method used in BRC (Nauman et al., 2025) to normalize the returns in the range of 0 to 1 in HumanoidBench using:

$$\text{Normalized Returns} = \frac{\text{Returns} - \text{Random Returns}}{\text{Success Returns} - \text{Random Returns}} \tag{8}$$

Detailed random returns and success returna are given in Table 3.

Table 3: Random and success scores for HumanoidBench tasks.

Task	Random Score	Success Score
h1hand-balance_hard-v0	10.032	800.0
h1hand-balance_simple-v0	10.170	800.0
h1hand-bookshelf_hard-v0	14.848	2000.0
h1hand-bookshelf_simple-v0	16.777	2000.0
h1hand-crawl-v0	278.868	800.0
h1hand-hurdle-v0	2.371	700.0
h1hand-insert_normal-v0	1.673	350.0
h1hand-insert_small-v0	1.653	350.0
h1hand-maze-v0	106.233	1200.0
h1hand-pole-v0	19.721	700.0
h1hand-reach-v0	-50.024	1200.0
h1hand-run-v0	1.927	700.0
h1hand-sit_hard-v0	2.477	750.0
h1hand-sit_simple-v0	10.768	750.0
h1hand-slide-v0	3.142	700.0
h1hand-spoon-v0	4.661	650.0
h1hand-stair-v0	3.161	700.0
h1hand-stand-v0	11.973	800.0
h1hand-walk-v0	2.505	700.0
h1hand-window-v0	2.713	650.0

F MODEL SIZE COMPARISON

Table 4: ScaleMoE (Simba) model sizes.

Method	Total Model Size (Critic + Actor)	Number of critic blocks	Critic Width	Expert Number
Simba (Standard)	$\approx$ 16.65MB	2	512	1
ScaleMoE (with 2 Simba Standard expert)	$\approx$ 33.24MB	2	512	2
ScaleMoE (with 4 Simba Standard expert)	$\approx$ 66.44MB	2	512	4
ScaleMoE (with 8 Simba Standard expert)	$\approx$ 132.83MB	2	512	8
Simba (Larger)	$\approx$ 133.12MB	2	1472	1

Table 5: ScaleMoE (BRC) model sizes.

Method	Total Model Size (Critic + Actor)	Number of Critic Blocks	Critic Width	Expert Number
BRC (Small)	≈4.8M	2	512	1
ScaleMoE (with 2 BRC Small expert)	≈9.6M	2	512	2
ScaleMoE (with 4 BRC Small expert)	≈19.2M	2	512	4
ScaleMoE (with 8 BRC Small expert)	≈38.3M	2	512	8
ScaleMoE (with 16 BRC Small expert)	≈76.6M	2	512	16
ScaleMoE (with 32 BRC Small expert)	≈153.1M	2	512	32
ScaleMoE (with 64 BRC Small expert)	≈306.3M	2	512	64
BRC (Standard)	≈260M	2	4096	1

G Resource Consumption

Table 6: Resource Consumption of ScaleMoE on DMC tasks.

Metric	DDPG-Simba (DMC)				SAC-BRC (DMC HUMANOIDS)			
	Number of Experts				Number of Experts			
	1	2	4	8	1	2	4	8
GPU Memory (MB)	750	1000	1500	2500	1100	1500	2000	3000
Training Time (Hours)	3.2	5.3	7.2	11.4	9.3	10.5	11.8	13.7

Table 6 shows the resource consumption of the ScaleMoE model on the single-task and multi-task DMC benchmark as the number of experts increases. Both GPU memory usage and training time grow when scaling from 1 to 8 experts. However, it should be noted that this consumption could be effectively converted to performance improvement as reported before.

H Results on MLP-based Network

Existing works show scaling simple feed-forward network could already bring benefits (McLean et al., 2025b). To further validate the effectiveness of ScaleMoE, we modify the BRC's network architecture by changing the BRONet backbone to a simple MLP and perform experiments. As shown in Figure 7, when scaling the MLP critic from a width of 256 to 1024 and 4096, we observed performance improvements beyond width 1024. However, further scaling width to 4096 leads to diminishing returns improvement.

We then applied our ScaleMoE approach to this simplified model. We used a 256-width critic network as the base architecture for each expert, and then scaled the number of experts from 8 to 16 to 32. Importantly, even when scaling up to 32 experts, the total number of parameters in ScaleMoE remained significantly smaller than that of a single monolithic critic network with a width of 4096. Despite having fewer parameters, the performance of ScaleMoE improved significantly, further justifying the use of MoE-based scaling instead of relying solely on a larger single network.

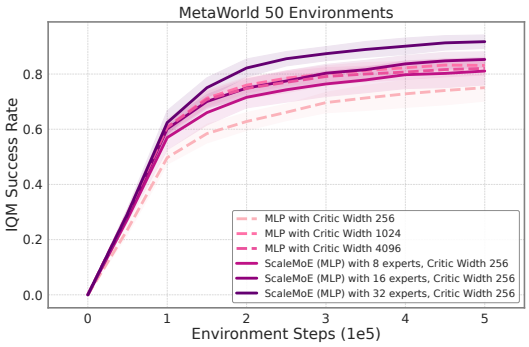


Figure 7: ScaleMoE with MLP-based networks.

I ADDITIONAL ABLATION STUDIES

I.1 UPDATING EXPERTS WITH THE SAME NUMBER OF GRADIENT FLOWS

Under our design with the load-balancing and importance loss, we ensure that experts are activated in a fairly balanced manner, which mitigates the risk of under-training. However, when using very small values of (K) (i.e., selecting only a few experts per state), each expert is updated fewer times, leading to a situation where some experts may not be effectively trained. This is particularly problematic in the case of sparse activation.

To address this, we maintain a fixed total number of updates for all experts by increasing the update-to-data ratio when k is small, ensuring that each expert is updated equally. This prevents experts from becoming under-trained. Our experiments demonstrate that with each expert receiving the same updates, ultimately the agent with small k could also lead to similar performance with that with large k.

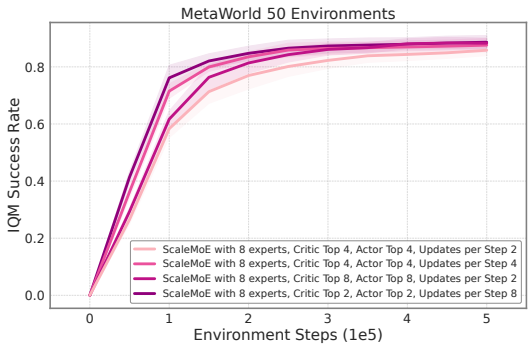


Figure 8: Ablation on experts updating times

It should be noted that even in the fully activated setting (Top-8/Top-8), the gating mechanism still plays a key role. The gating network does not simply activate all experts with equal weight, but instead, it assigns different weights to each expert based on the gating network's output. This means that, while all experts are being used, they are not used equally. Some experts may receive a larger weight in certain contexts, and others may be down-weighted accordingly.

I.2 ACTOR EXPRESSIVITY ABLATION

From an intuitive perspective, in Figure 6, the improvement of increasing activated actor comes from two obvious factors (1) more thorough training: activating more experts ensures each expert gets more updates, preventing under-training. (2) the expressibility of the combined policy: more experts lead to a more expressive policy that can capture multi-modal behaviors, improving performance.

To prove the improved expressivity of the gated actor, we keep the activated critic expert number to 4 unchanged, and change the number of actor experts with full activation to ensure each expert is trained with same gradient steps. We show the results on Metaworld with 50 tasks in the Figure 9. It shows that as the actor expert number increases, the overall performance also increases. which is consistent with our theoretical analysis in Appendix M (E). To further explore the underlying reasons, we investigate whether increasing actor could bring other benefits. We investigate the Q-target variance in the critic. We found that as more experts are activated, the critic's target Q-values become more stable and reliable, leading to better convergence.

I.3 AUXILIARY LOSS ABLATION

We compare the performance of ScaleMoE with and without the load-balancing and importance regularization loss. As expected, we found that with the auxiliary loss, the model performs better overall, especially in terms of stability and convergence in Figure 10.

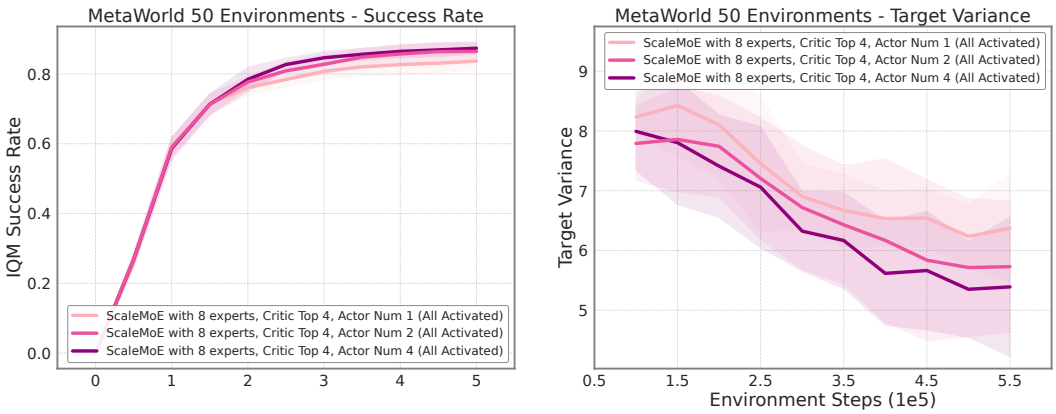


Figure 9: Ablation on actor expressivity.

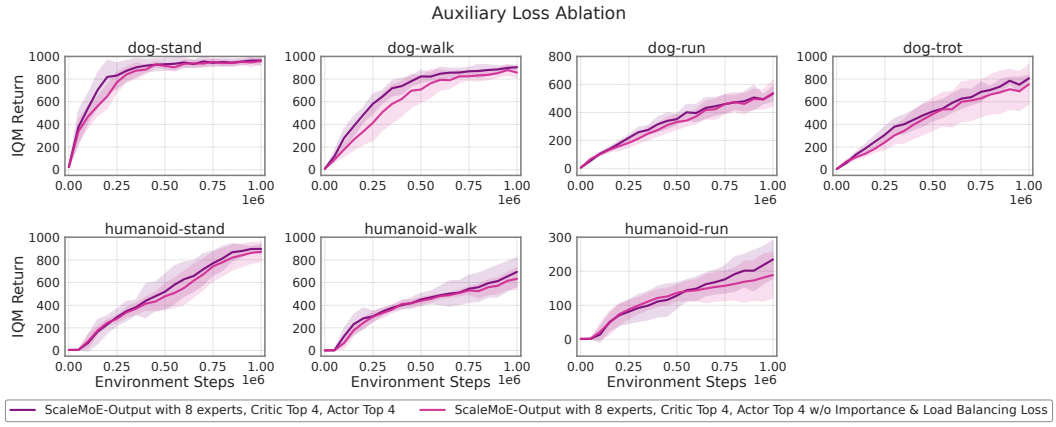


Figure 10: Ablation on load-balancing and importance loss.

To further validate the effectiveness of the load-balancing loss, we present an additional figure (Figure 11) that shows the expert activation frequency during training. This plot illustrates how often each expert is activated across all tasks. When the load-balancing loss is applied, we observe that expert activation is much more uniform across all experts. In contrast, without the load-balancing loss, some experts are hardly activated at all, which can lead to under-utilization. As a result, the network's overall representational capacity is significantly compromised, as certain experts fail to specialize and learn meaningful features.

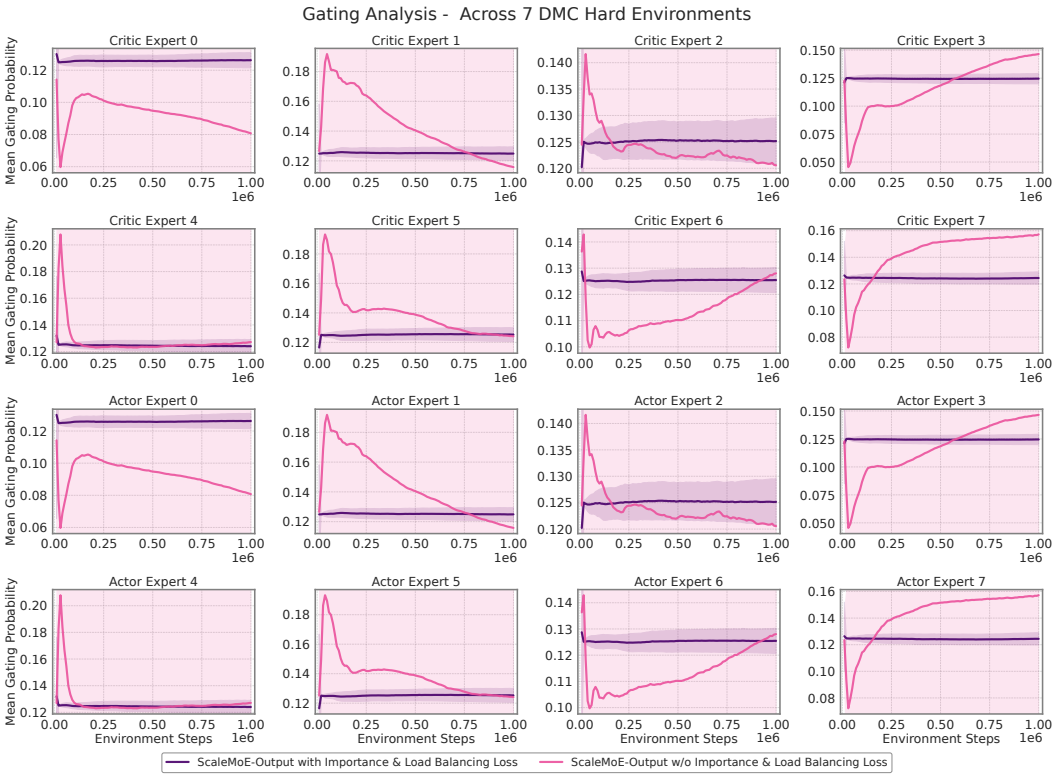


Figure 11: Expert activation during training on DMC Hard envs.

In Figure 12, we also provide a comparison of expert activation frequencies of a single run in one task, highlighting how the activation pattern changes with and without the load-balancing loss. With the load-balancing loss, experts are evenly activated, ensuring that each expert contributes to the learning process. Without the loss, certain experts receive significantly fewer updates, leading to poorer task-specific performance and less effective task specialization.

J In-depth Analysis

To further understand the underlying reasons for ScaleMoE's gains, we add three new quantifiable metrics to explain the benefits of ScaleMoE.

Gradient Interference Between Experts. The interference metric quantifies the degree to which the gradients from different experts conflict with each other during training. Specifically, it is measured by the cosine similarity between the gradients of the experts, denoted as g_i for expert i and g_j for expert j, as:

$$\cos(g_i, g_j) = \frac{\langle g_i, g_j \rangle}{|g_i||g_j|} \tag{9}$$

where $\langle g_i, g_j \rangle$ is the dot product of the gradients and $|g_i|$ is the L2 norm of the gradient vector. A low interference value corresponds to low conflict between the gradients of experts, meaning that they

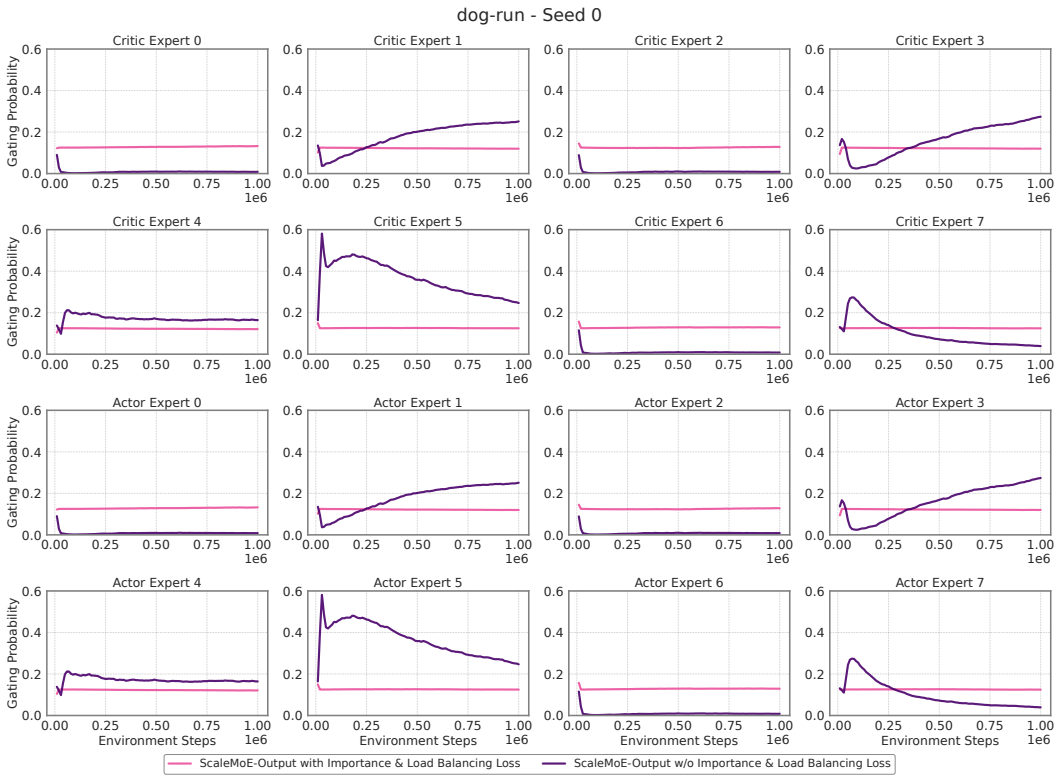


Figure 12: Expert activation during training on dog-run env.

are learning complementary representations. As more experts are activated (increasing (K)), we observe a decrease in interference, bringing the metric closer to 0, indicating that experts are working in harmony rather than conflicting. This reduction in interference improves gradient flow, stabilizing the learning process and allowing the model to efficiently leverage the expertise of different specialists without overlap. As shown in Figure 13, increasing (K) leads to more balanced gradient updates, which enhances learning efficiency, aligning with the theoretical analysis in Appendix M (D).

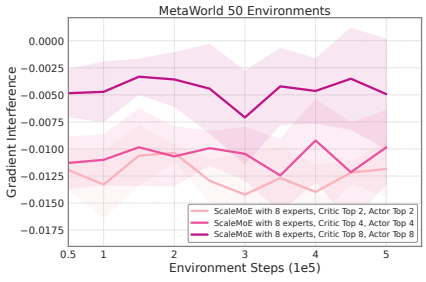


Figure 13: Gradient interference of ScaleMoE (BRC) on Metaworld 50 envs.

Expert Activation Entropy. We also assess the entropy of expert activation across different tasks or states. The activation entropy H is computed as:

$$H = -\sum_{i=1}^{N} p_i \log(p_i), \tag{10}$$

where p_i is the probability of expert i being activated, typically normalized by the gating weights. A high entropy value indicates that the gating mechanism distributes the activation more evenly across experts, ensuring that no single expert dominates the learning process, which would lead to

under-utilization of the model's capacity. Figure 14 demonstrates that as the number of activated experts (K) increases, the entropy of expert activation increases, reflecting a more uniform expert usage. This balanced activation improves the model's ability to generalize and prevents any expert from being neglected, which is essential for multi-task learning and task specialization.

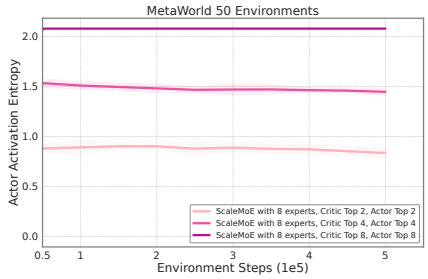


Figure 14: Actor expert activation entropy of ScaleMoE (BRC) on Metaworld 50 envs.

Critic Target Variance. The variance of the critic's target values is an important measure of the stability of the value function. We define the variance of the critic's target y_K as:

$$\mathrm{Var}(y_K) = \mathbb{E}[(y_K - \mathbb{E}[y_K])^2], \tag{11}$$

where y_K is the target value predicted by the critic, which depends on the output of the top-K experts. As the number of active experts (K) increases, the variance of the critic's target decreases. This reduction in variance indicates that the critic is more stable and provides more reliable target estimates, which helps to improve convergence and learning efficiency. Lower variance in the target signal is beneficial for both actor and critic updates, leading to faster and more stable training. Empirical evidence supporting this can be seen in Figure 15, where we show that variance decreases as (K) increases, corroborating our theoretical findings in Appendix M (B).

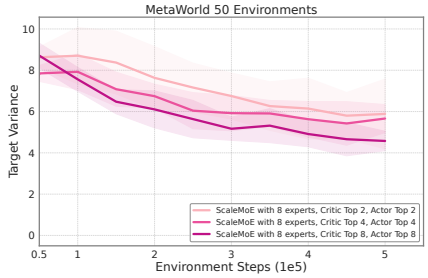
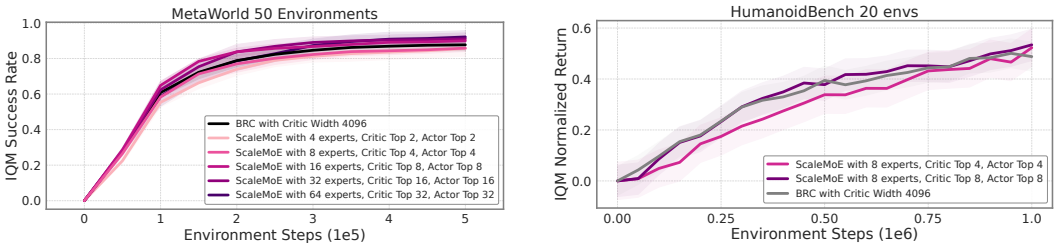


Figure 15: Critic target variance of ScaleMoE (BRC) on Metaworld 50 envs.

K Training Curves



(a) Training curves on Metaworld 50 envs.

(b) Training curves on HumanoidBench 20 envs.

Figure 16: Scaling performance on multi-task environments.

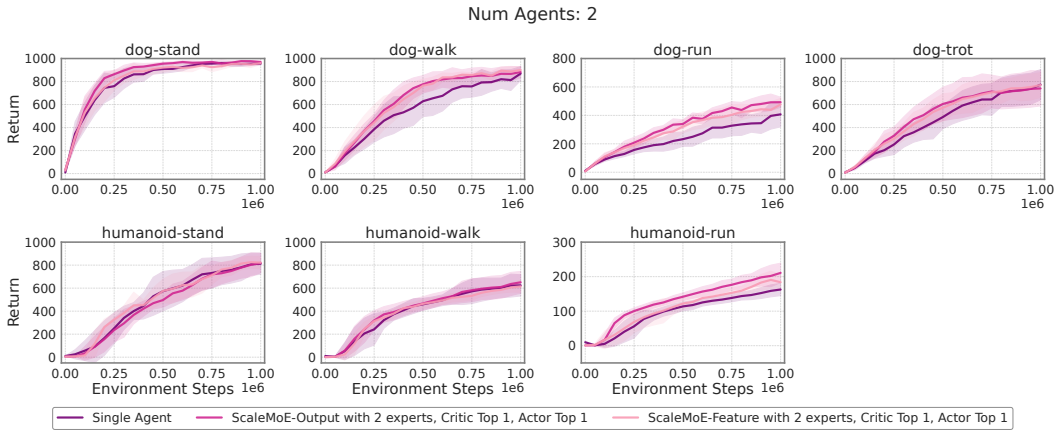


Figure 17: Single-task training curves of ScaleMoE (Simba) on each DMC env with two experts.

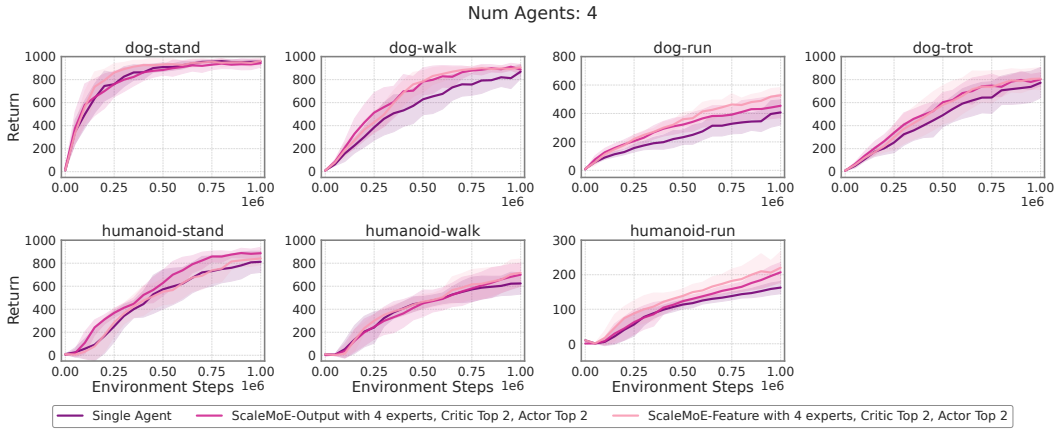


Figure 18: Single-task training of ScaleMoE (Simba) on each DMC env with four experts.

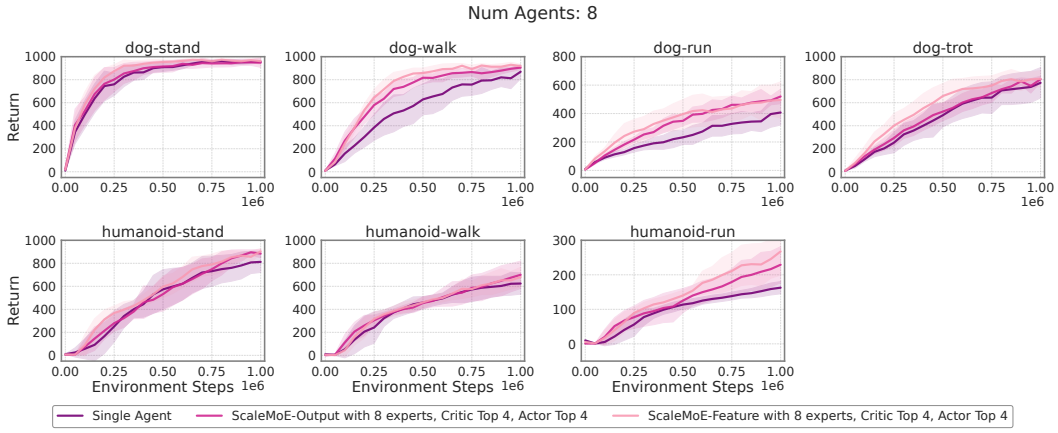


Figure 19: Single-task training curves of ScaleMoE (Simba) on each DMC env with eight experts.

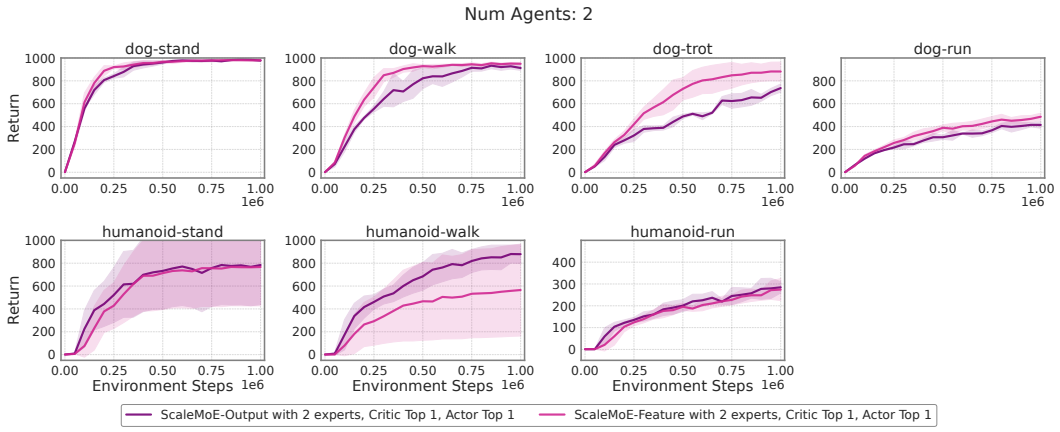


Figure 20: Multi-task training curves (normalized scores) of ScaleMoE (BRC) on each DMC env with two experts.

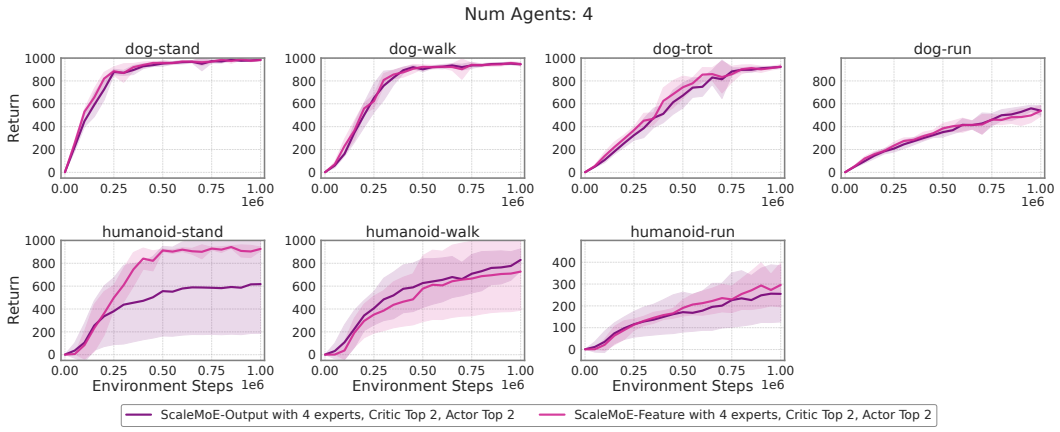


Figure 21: Multi-task training curves (normalized scores) of ScaleMoE (BRC) on each DMC env with four experts.

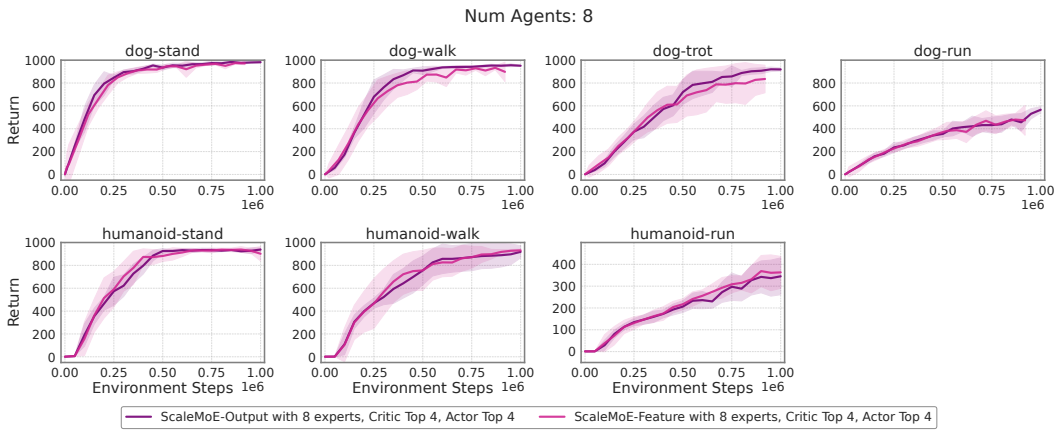


Figure 22: Multi-task training curves (normalized scores) of ScaleMoE (BRC) on each DMC env with eight experts.

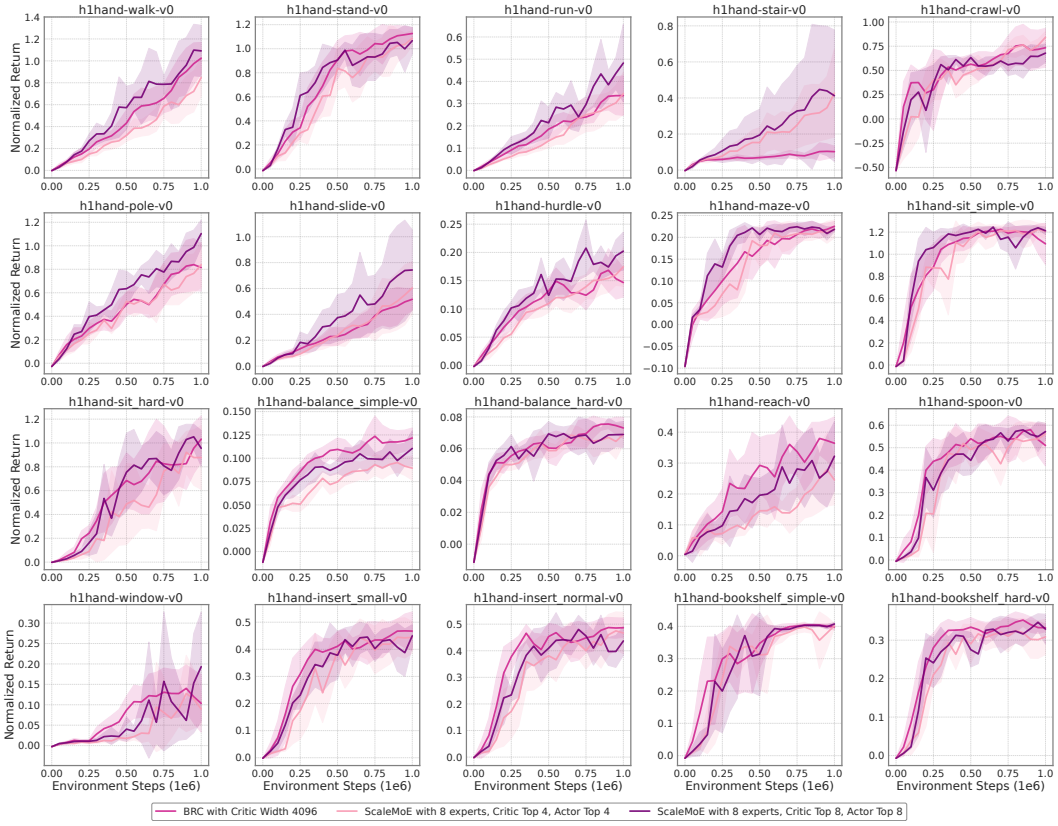


Figure 23: Multi-task training curves (normalized scores) of ScaleMoE (BRC) on each Humanoid-Bench env with eight experts.

L Broader Impact

Our proposed ScaleMoE method contributes to closing the gap between the scaling capabilities of deep RL and those of supervised learning domains. By enabling RL agents to effectively utilize larger models, we open the door to solving more complex control problems and multi-task scenarios that were previously out of reach due to capacity limitations. **Positive impacts** could include more capable robotic controllers that can handle diverse tasks (thanks to expert specialization) and improved sample efficiency through conditional reuse of knowledge. For example, a household robot could employ a ScaleMoE policy to skillfully handle a wide array of chores by activating different expert networks for cooking vs cleaning tasks, rather than requiring separate models for each. Additionally, our analysis of neuron utilization might inform better neural architecture design beyond RL, highlighting the importance of activating as much of the network as possible (which could lead to generally more efficient deep learning models).

On the **downside**, larger models and MoE architectures do raise concerns about computational cost and energy usage. Training the ScaleMoE agents on complex benchmarks consumed considerable GPU hour, although it can be more efficient than naive scaling. Developers should weigh the improvements in performance against the environmental and financial costs of training such models.

Overall, we believe the benefits of more scalable and generalizable RL agents outweigh the potential downsides, and we advocate for continued research in efficient scaling techniques like MoE to push the frontiers of what RL can achieve.

M Theoretical Intuition

We sketch several complementary arguments, i.e., approximation, variance, and optimization, that explain the gains we observe with ScaleMoE.

(A) Specialization reduces approximation error. Let $f^\star(s,a)$ denote the target function (a policy map or a Q-function). A soft-gated MoE induces a partition of unity $\{w_i(s)\}_{i=1}^N$ with $\sum_i w_i(s) = 1$, yielding

$$f_{\text{MoE}}(s,a) \;=\; \sum_{i=1}^{N} w_i(s)\, f_i(s,a).$$

If the state–action space decomposes into regions $\{\mathcal{R}_i\}$ on which $f^\star$ is simpler (e.g., lower curvature, lower intrinsic dimension), then fitting a smaller-capacity expert f_i on $\mathcal{R}_i$ can reduce the local estimation error compared to a single monolithic model of the same total size. Formally, a local Rademacher complexity bound (Yin et al., 2019) gives a generalization gap of order

$$\mathbb{E}\big[\ell(f_{\text{MoE}}) - \ell(f^\star)\big] \;\lesssim\; \sum_{i=1}^{N} \underbrace{\mathfrak{R}_n(\mathcal{F}_i;\mathcal{R}_i)}_{\text{local complexity}} + \text{noise},$$

where $\mathfrak{R}_n(\mathcal{F}_i;\mathcal{R}_i)$ is smaller than the global complexity $\mathfrak{R}_n(\mathcal{F}_{\text{mono}})$ when $f^\star$ is piecewise-structured. Intuitively, the gate learns to route states to experts with the right inductive bias, yielding lower bias without inflating variance.

(B) Variance reduction from top-K mixtures. For critic evaluation with squared loss, suppose each expert's TD target has zero-mean estimation noise ε_i with $\mathrm{Var}[\varepsilon_i] = \sigma_i^2$ and limited correlation. The fused critic

$$Q_{\text{final}}(s,a) \;=\; \sum_{i\in\mathcal{K}(s)} \tilde{w}_i(s)\, Q_i(s,a)$$

has conditional variance

$$\mathrm{Var}[Q_{\text{final}} \mid s,a] \;=\; \sum_{i\in\mathcal{K}(s)} \tilde{w}_i^2(s)\,\sigma_i^2 \;+\; 2\sum_{i<j} \tilde{w}_i(s)\tilde{w}_j(s)\,\mathrm{Cov}(\varepsilon_i,\varepsilon_j) \;\le\; \Big(\max_i \sigma_i^2\Big) \sum_{i\in\mathcal{K}(s)} \tilde{w}_i^2(s).$$

With equal weights and weak covariance, $\sum_i \tilde{w}_i^2 \approx 1/K$, giving a $\approx 1/K$ variance reduction relative to a single head. This ensemble-like effect improves target accuracy and stabilizes value learning. A similar argument applies to policy gradients: mixing multiple expert policies reduces gradient variance when experts are not perfectly correlated.

(C) Contraction and stability under convex fusion. For policy evaluation, the Bellman operator T^π is linear and a γ-contraction in $\|\cdot\|_\infty$ (Sutton & Barto, 2018). For any state-dependent convex weights $w_i(s) \geq 0$, $\sum_i w_i(s) = 1$,

$$T^\pi\left(\sum_i w_i Q_i\right) = \sum_i w_i\, T^\pi(Q_i), \qquad \|T^\pi Q_{\text{final}} - T^\pi Q^\star\|_\infty \leq \gamma\, \|Q_{\text{final}} - Q^\star\|_\infty,$$

so convex fusion preserves the contraction property and does not undermine the fixed point. Thus, variance reductions in (B) translate directly into tighter error bounds under iterative evaluation.

(D) Reduced interference and better optimization. Let $g_t(\theta)$ be the gradient at step t on parameters θ. In a monolithic model trained on heterogeneous data, expected gradient interference $\mathbb{E}[\langle g_t^{(u)}, g_t^{(v)}\rangle]$ between disparate regimes $u \neq v$ can be negative, slowing learning. Routing with MoE sparsifies updates so that each expert's parameters receive gradients primarily from the subset of states they serve:

$$\theta_i \leftarrow \theta_i - \eta\, \mathbb{E}_{(s,a)}\left[w_i(s)\, \nabla_{\theta_i} \ell_i(s,a)\right],$$

which lowers cross-regime interference and accelerates specialization. Empirically, this manifests as fewer dormant units and better plasticity; theoretically, it reduces the effective condition number of the local objectives, easing optimization.

(E) Expressivity for continuous actions. A single Gaussian policy is unimodal; many continuous-control tasks are multi-modal (e.g., equivalent symmetric actions, contact-rich choices). Even when the final policy is kept Gaussian for simplicity, routing over experts induces a piecewise-smooth mapping

$$\mu_{\text{final}}(s) = \sum_{i \in \mathcal{K}(s)} \tilde{w}_i(s)\, \mu_i(s), \qquad \Sigma_{\text{final}}(s) = \sum_{i \in \mathcal{K}(s)} \tilde{w}_i(s)\, \Sigma_i(s),$$

which can approximate mixtures or mode switches via state-dependent convex combinations, substantially more expressive than a single global head.

Takeaway

ScaleMoE improves performance by (i) matching local structure with specialized, lower-complexity experts (lower bias), (ii) averaging over multiple routed estimates (lower variance), (iii) preserving Bellman contraction under convex fusion (stability), and (iv) mitigating gradient interference through sparse, state-dependent updates (better optimization). Load-balancing and importance regularizers further prevent expert collapse, ensuring that increased capacity is effectively trained rather than left dormant.

N Median, IQM and Mean Results

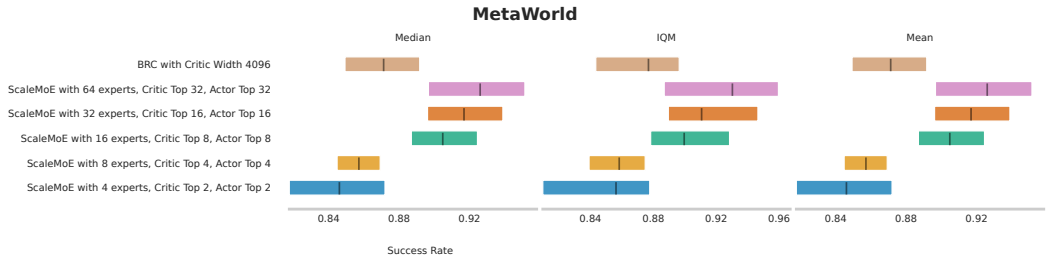


Figure 24: Median, IQM and Mean Comparison on Metaworld.

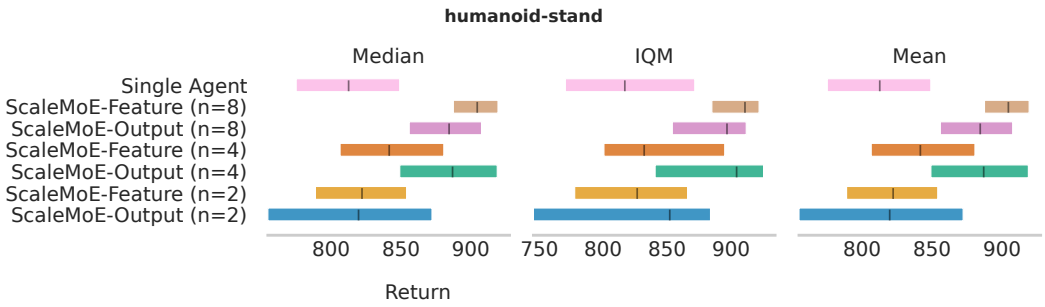


Figure 25: Median, IQM and Mean Comparison on humanoid-stand.

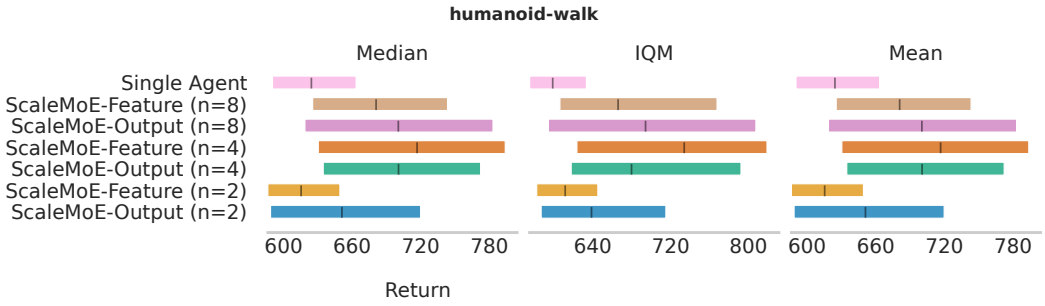


Figure 26: Median, IQM and Mean Comparison on humanoid-walk.

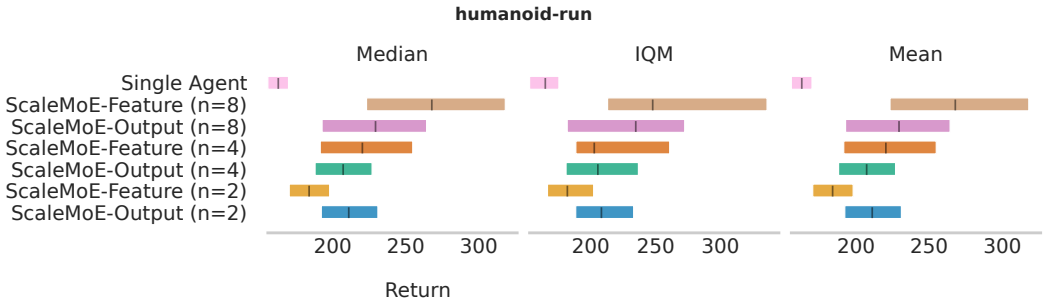


Figure 27: Median, IQM and Mean Comparison on humanoid-run.

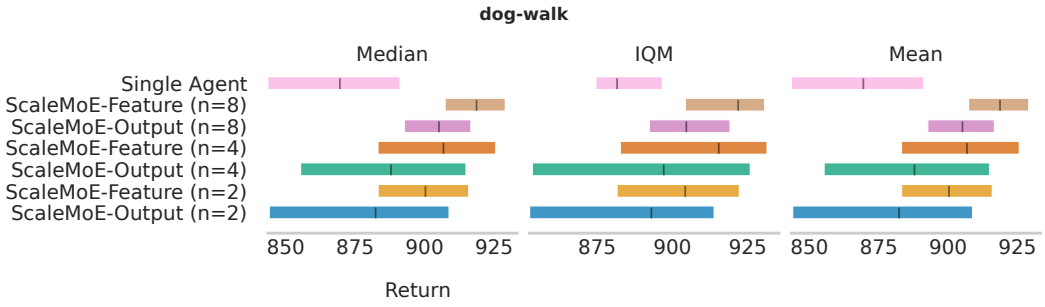


Figure 28: Median, IQM and Mean Comparison on dog-walk.

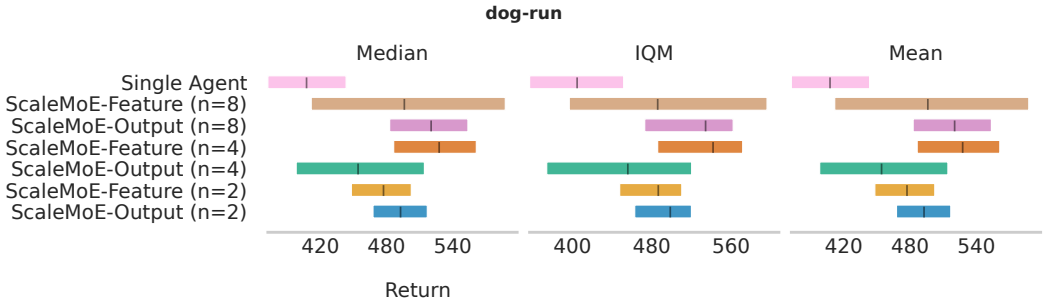


Figure 29: Median, IQM and Mean Comparison on dog-stand.

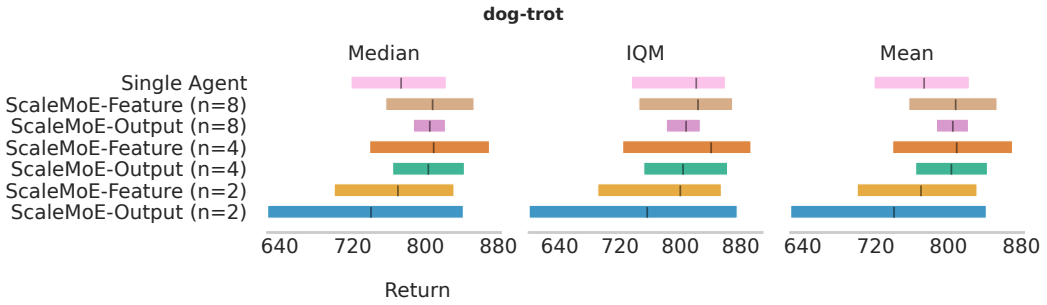


Figure 30: Median, IQM and Mean Comparison on dog-trot.

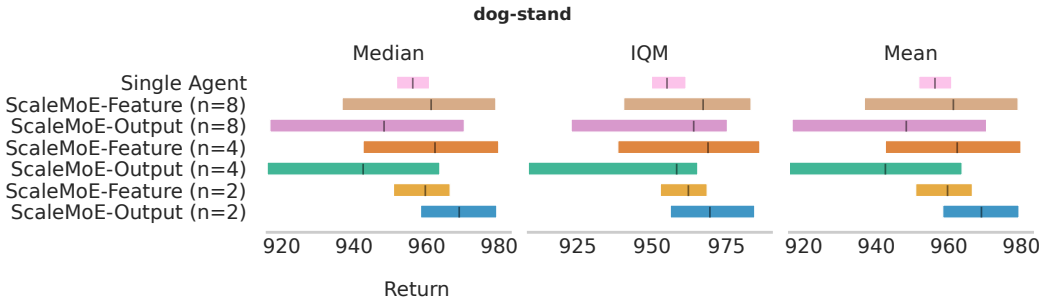


Figure 31: Median, IQM and Mean Comparison on dog-stand.