Preventing Learning Stagnation in PPO by Scaling to 1 Million Parallel Environments

Michael Beukman, Khimya Khetarpal, Zeyu Zheng, Will Dabney, Jakob Foerster, Michael Dennis, Clare Lyle

Keywords: Learning Stagnation, PPO, Optimization, Parallelisation, Open Ended Learning

Summary

An agent's performance stagnating at a suboptimal level is a common problem in deep on-policy RL. Focusing on PPO, we show that plateaus in certain regimes arise not because of known exploration, capacity, or optimisation challenges, but because sample-based estimates of the loss eventually become poor proxies for the true objective over the course of training. Looking deeper, PPO alternates between sampling rollouts from several parallel environments *online* using the current policy (which we call the "outer loop") and performing repeated mini-batch SGD steps against this *offline* dataset (the "inner loop"). In our work, we abstract away the inner loop, and conceptually model the outer loop as standard stochastic optimisation. The step size is then controlled by the regularisation strength towards the previous policy and the gradient noise by the number of samples collected between policy update steps. This framing predicts that, much like in SGD, if the outer step size is too large relative to the noise, updates become uninformative and lead to the policy thrashing around a local optimum instead of converging. Recasting PPO in this light makes it clear that there are two ways to address this particular type of learning stagnation: either reduce the step size or increase the number of samples collected between updates. We validate the predictions of our model and conclude that increasing the number of parallel environments is a simple way to avoid these plateaus by simultaneously altering both these factors. Applying our analysis and scaling PPO to more than 1M parallel environments enables monotonic performance improvement up to one trillion transitions and leads to vastly superior performance compared to prior baselines in a complex open-ended domain.

Contribution(s)

1. We revisit the perspective of PPO as an inner and outer loop process and show that some cases of suboptimal plateaus are caused by an overly large *outer step size*.
   Context: Schulman et al. (2017) present this inner and outer loop structure, but do not link plateaus with large outer step sizes, or use this conceptual model to set hyperparameters.
2. We experimentally identify several factors governing the outer step size and decouple these from the hyperparameters that primarily affect the inner optimisation process. We further demonstrate that increasing the number of parallel environments in PPO provides a reliable mechanism to modulate the outer step size, leading to higher asymptotic performance.
   Context: Rather than viewing parallelisation solely as a means of controlling hardware utilisation (Freeman et al., 2021; Hilton et al., 2022), we show it can directly impact agents' plateauing behaviour when paired with appropriate hyperparameter settings.
3. We propose a simple recipe for how to co-scale other hyperparameters when changing the number of parallel environments, and demonstrate that this makes PPO highly effective in difficult robotics domains. We additionally show that increasing parallelisation is one way to avoid learning stagnation in the challenging and open-ended domain of `Kinetix`.
   Context: Singla et al. (2024) show that co-scaling hyperparameters is nontrivial. Increasing parallelisation using our recipe significantly outperforms Matthews et al. (2025) by continuing to improve long after the baseline stagnation point of five billion timesteps.

Preventing Learning Stagnation in PPO by Scaling to 1 Million Parallel Environments

Michael Beukman[1,*], **Khimya Khetarpal**[2], **Zeyu Zheng**[2],
Will Dabney[2], **Jakob Foerster**[1], **Michael Dennis**[2], **Clare Lyle**[2]
{mbeukman,jakob}@robots.ox.ac.uk
{khimya,zeyuzheng,wdabney,dennismi,clarelyle}@google.com

[1]**FLAIR, University of Oxford**
[2]**Google DeepMind**

[*] Work done while at Google DeepMind.

Abstract

An agent's performance stagnating at a suboptimal level is a common problem in deep on-policy RL. Focusing on PPO, we show that plateaus in certain regimes arise not because of known exploration, capacity, or optimisation challenges, but because sample-based estimates of the loss eventually become poor proxies for the true objective over the course of training. Looking deeper, PPO alternates between sampling rollouts from several parallel environments *online* using the current policy (which we call the "outer loop") and performing repeated minibatch SGD steps against this *offline* dataset (the "inner loop"). In our work, we abstract away the inner loop, and conceptually model the outer loop as standard stochastic optimisation. The step size is then controlled by the regularisation strength towards the previous policy and the gradient noise by the number of samples collected between policy update steps. This framing predicts that, much like in SGD, if the outer step size is too large relative to the noise, updates become uninformative and lead to the policy thrashing around a local optimum instead of converging. Recasting PPO in this light makes it clear that there are two ways to address this particular type of learning stagnation: either reduce the step size or increase the number of samples collected between updates. We validate the predictions of our model and conclude that increasing the number of parallel environments is a simple way to avoid these plateaus by simultaneously altering both these factors. Applying our analysis and scaling PPO to more than 1M parallel environments enables monotonic performance improvement up to one trillion transitions and leads to vastly superior performance compared to prior baselines in a complex open-ended domain.

1 Introduction

A common failure mode in RL is the tendency for an agent's performance to plateau well below the theoretical optimal return in an environment (Nikishin et al., 2022; Lyle et al., 2022; Nauman et al., 2024). This is becoming an increasingly visible problem as highly-parallelised and complex RL environments have gained popularity (Freeman et al., 2021; Makoviychuk et al., 2021; Lange, 2022; Nikulin et al., 2023; Rutherford et al., 2023; Matthews et al., 2024; Zakka et al., 2025), meaning that it is becoming feasible to run agents for billions or trillions of timesteps with only modest hardware requirements (Matthews et al., 2025). However, if our algorithms cannot improve beyond a subpar plateau even in the limit of additional experience, there is little use for these trillions of timesteps.

Prior work has explored different reasons behind why an algorithm might plateau. One explanation is plasticity loss or the primacy bias, where the network accumulates pathologies during training that

hinder the optimisation process (Nikishin et al., 2022; Lyle et al., 2022; 2025). Other works highlight insufficient exploration (Thrun, 1992; Bellemare et al., 2016; Küttler et al., 2020; Taiga et al., 2021), e.g., due to the agent collapsing to a near-deterministic policy too early. While this may be a problem in certain settings, empirical results suggest that plateaus can still occur in dense reward tasks which do not pose hard exploration challenges, highlighting the importance of implementation details and hyperparameters (Henderson et al., 2018; Engstrom et al., 2020; Andrychowicz et al., 2020).

We take a different perspective, one inspired by PPO's roots in proximal gradient methods and the empirical similarities between plateaus in RL and stochastic optimisation. In particular, we abstract away the inner neural network optimisation process and focus only on the outer loop, conceptually modelling it as standard stochastic optimisation. In this model, the *step size* represents how much the policy changes between update iterations, whereas the *update noise* represents how well minimising the loss on a sampled batch of trajectories corresponds to maximising the true objective. It now becomes clear that PPO is vulnerable to plateaus when the outer step size is too large relative to the update noise level; much like in SGD, this prevents convergence by causing the policy to thrash around a local optimum. Crucially, such a mechanism suggests that these plateaus are caused not by the policy changing slowly, but by it changing too much between update iterations. There are therefore two primary levers to address these plateaus: we can either reduce the step size through increased regularisation towards the behaviour policy or decrease the noise by collecting more data per update.

Based on this perspective, we show that one simple way to influence PPO's plateauing behaviour—by modulating both of these factors—is to change the number of parallel environments. However, how best to adjust the other hyperparameters when doing so remains unclear, since more parallel rollouts require either larger minibatches or more optimisation steps, both of which may in turn require adjusting, for instance, the learning rate or regularisation strength (Hilton et al., 2022; Singla et al., 2024). We demonstrate that a simple and reliable strategy is to keep the *inner* optimisation process the same: in other words, fix the minibatch size and learning rate, and only increase the number of optimisation steps. In a difficult robotics domain, we find that this recipe makes PPO more amenable to massive parallelisation compared to when changing the inner optimisation hyperparameters. Finally, we significantly exceed the prior performance ceiling in the challenging 2D physics-based open-ended environment, `Kinetix` (Matthews et al., 2025). While standard configurations plateau after less than ten billion interactions, scaling PPO to over one million parallel environments allows for sustained monotonic improvement far beyond this point, up to one *trillion* timesteps.

We structure this paper as follows. First, Section 3 empirically justifies our conceptual model of PPO as stochastic optimisation, and shows that both settings share the same mechanisms (e.g., thrashing around a local optimum) and remedies (e.g., reducing the step size) associated with plateaus. We further validate this analogy by showing that changing the outer step size during training is sufficient to either induce a plateau or recover from one. Next, Section 4 examines the effect of various hyperparameters on the outer step size and update noise and isolates (a) the regularisation strength towards the previous policy, (b) the number of transitions collected per iteration, and (c) the number of optimisation epochs performed on each batch of data as key factors. We then investigate how the optimal step size changes as a function of the training budget, and find that it becomes smaller as the total interaction budget increases. This section ends by arguing that increasing the number of parallel environments is a simple and robust way to lower both the update noise and step size. Section 5 determines how to co-scale the other hyperparameters when increasing parallelisation, and demonstrates the benefits of our recipe in a challenging robotics domain. Finally, we showcase significant performance benefits by using our analysis to circumvent learning stagnation in `Kinetix`.

2 Background

2.1 Proximal Policy Optimisation

Proximal Policy Optimisation (Schulman et al., 2017, PPO) is a particularly prevalent, on-policy RL algorithm, with training comprising two distinct phases, which we call the outer and inner loops,

respectively.[1] In the outer loop, the current policy (also known as the behaviour policy) collects data by rolling out N_{envs} parallel environments for K steps each, resulting in a dataset with $N_{\text{envs}} \cdot K$ transitions. The inner loop consists of N_{epochs} passes over this dataset, each pass performing $N_{\text{minibatches}}$ minibatch-SGD gradient steps, usually with the Adam optimiser (Kingma & Ba, 2014).

The agent consists of a policy (also known as the actor) $\pi_\theta(a|s)$, defining a probability distribution over actions for each state, and a critic $V_\theta(s)$, which approximates the sum of discounted returns starting from a particular state s and following π_θ. Typically, both of these consist of deep neural networks parametrised by θ, representing either a shared architecture or distinct weights for the actor and critic. The following loss function is maximised with respect to θ:

$$L_t(\theta) = \mathbb{E}\left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{VF}(\theta) + c_2 \mathcal{H}[\pi_\theta](s_t)\right] \tag{1}$$

$$L_t^{\text{CLIP}}(\theta) = \min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right) \tag{2}$$

$$L_t^{VF}(\theta) = (V_\theta(s_t) - V_t^{\text{target}})^2, \tag{3}$$

where $\mathcal{H}$ is the entropy of the policy, $\hat{A}_t$ is the advantage calculated using GAE (Schulman et al., 2016) and $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{behaviour}}}(a_t|s_t)}$ is the probability ratio between the current and behaviour policy. V_t^{target} is defined as $V_\theta(s_t) + \hat{A}_t$, computed once before the first optimisation step. Here, the purpose of the clipping term is to prevent the agent's policy moving too rapidly in any single iteration, and has the effect of zeroing out gradients from transitions where the current policy and the behaviour policy's action probabilities differ by more than ϵ (Schulman et al., 2017).

2.2 PPO-EWMA

In PPO, the behaviour policy serves two different purposes (Schulman et al., 2017; Hilton et al., 2022). The first is to calculate the **importance sampling** ratio, in order to correct for the fact that the data-collecting policy is not the same as the current policy being learned (since we do multiple minibatches and epochs per policy update step). The second is to act as a **regulariser**, to ensure that the current policy does not drift too far away from a reasonable reference. A key insight from Hilton et al. (2022) is that we can use two *different* policies to serve these distinct purposes: the behaviour policy collects a fixed amount of data, and the proximal policy (i.e., the reference policy we regularise towards) is set to the policy from a fixed number of minibatch update steps ago (where older proximal policies lead to stronger regularisation). This allows us to control the relative strength of regularisation without altering the data collection process. Since storing all intermediate policies is expensive in terms of memory, the approximation the authors suggest is an exponentially-weighted moving average (EWMA) of the current policy's weights.

Hilton et al. (2022) further argue that the number of parallel environments in standard PPO implicitly influences the regularisation, since it changes the age of the behaviour policy, which in standard PPO is the same as the policy that we regularise towards. PPO-EWMA decouples these factors, allowing practitioners to set regularisation independent of parallelisation, and has been used in several recent works to improve training stability, often in asynchronous settings (Hilton et al., 2023; Zheng et al., 2025; Fu et al., 2025). This is done by modifying the PPO loss function as follows:

$$L_{t,\text{decoupled}}^{\text{CLIP}}(\theta) = \mathbb{E}\left[\frac{\pi_{\theta_{\text{prox}}}(a_t|s_t)}{\pi_{\theta_{\text{behaviour}}}(a_t|s_t)} \min\left(r_t^{\text{prox}}(\theta)\hat{A}_t, \text{clip}(r_t^{\text{prox}}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{4}$$

where $r_t^{\text{prox}}(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{prox}}}(a_t|s_t)}$ and θ_{prox} is an EWMA of θ, updated after every minibatch as $\theta_{\text{prox}} \leftarrow \beta_{\text{prox}}\theta_{\text{prox}} + (1-\beta_{\text{prox}})\theta$. The first ratio now is for importance sampling whereas $r_t^{\text{prox}}(\theta)$

[1] These correspond to the data collection and model update steps of standard implementations, see Huang et al. (2022a;b).

controls regularisation. Importantly, if $\epsilon = \infty$ and no clipping happens, the product of the ratios $\frac{\pi_{\theta_{\text{prox}}}(a_t|s_t)}{\pi_{\theta_{\text{behaviour}}}(a_t|s_t)} \cdot \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{prox}}}(a_t|s_t)}$ recovers the standard importance sampling ratio $r_t(\theta)$ (Hilton et al., 2022).

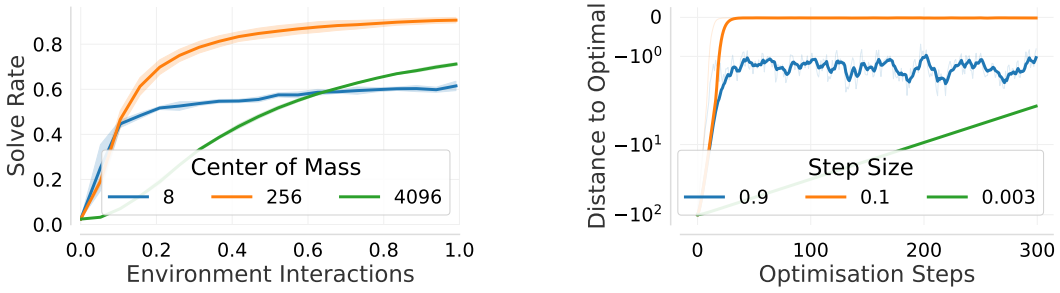
For the EWMA, the center of mass (COM) is defined as $\frac{1}{1-\beta_{\text{prox}}} - 1$, measured in minibatch update steps. This quantity controls the "age" of the reference policy, and thereby the regularisation strength. As an example, $\beta_{\text{prox}} = \frac{1024}{1025} \approx 0.99902$ corresponds to a center of mass of 1024, meaning that the average age of each term in the EWMA calculation is 1024 minibatches. In other words, the regularisation here is comparable to normal PPO when using 256 minibatches and 8 epochs: because standard PPO refreshes the behaviour policy every 2048 minibatches (8×256), the average age of its regularisation target is likewise 1024 minibatches.

For some experiments in this paper, we use PPO-EWMA as an analysis tool which provides a more interpretable and granular way to control the regularisation strength compared to directly altering the clipping threshold. We also find that low COMs lead to more stable training than high ϵ's, allowing us to more easily study the effects of weak regularisation. However, Appendix B shows that we can largely counteract changes to the COM by appropriately adjusting ϵ, and vice versa, confirming that these hyperparameters affect the same underlying mechanism. Furthermore, our final results in Section 5 use standard PPO, showing that our insights transfer to the more commonly-used variant.

3 PPO as a Stochastic Optimisation Process

In this section we empirically justify our conceptual model of PPO's outer loop as stochastic optimisation. For these experiments, we use a state-based robotic locomotion task comprising 512 procedurally-generated morphologies built using the `Jax2D` physics engine (Matthews et al., 2025) and a simple noisy convex optimisation problem: minimizing $\mathbf{x}^T\mathbf{x}$, with $\mathbf{x} \in \mathbb{R}^{50}$. We perform standard gradient descent, but add Gaussian noise with standard deviation $\frac{3}{\sqrt{50}}$ to the gradients. To more clearly demonstrate the similarities to RL, we plot the negative of the euclidean distance to the optimal solution $\mathbf{x}^* = \mathbf{0}$. For all figures, we plot the mean and shade the 95% CI over 5 seeds unless otherwise noted. See Appendix A for more details and hyperparameters.

We start with the observation that initially inspired our conceptual model: scaling the outer-loop step size (regularisation strength in this case) in PPO has a similar effect on the resulting learning curves as scaling the learning rate in SGD. When it is too high, e.g., the blue lines in Figure 1, performance plateaus at a suboptimal level, and when it is too low (the green lines), the optimisation process fails to converge within the allocated budget.



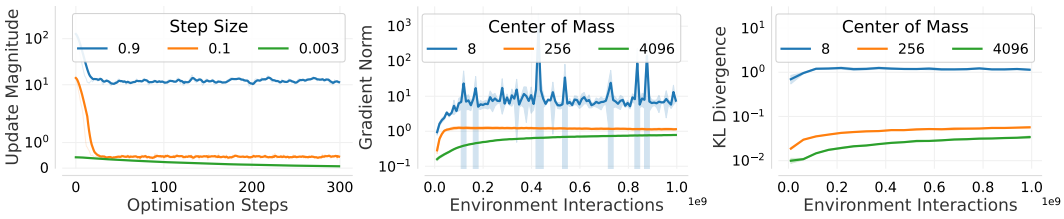
(a) PPO

(b) Noisy Convex Optimisation

Figure 1: Comparing the behaviour in (a) PPO and (b) convex optimisation with stochastic gradients. In (a) having too large of an outer step size (in particular, having a center of mass of the proximal policy being too low) leads to a suboptimal plateau, with the same behaviour occurring in (b). Solve rate corresponds to the policy's average success rate over all 512 morphologies.

3.1 Learning Dynamics Under Excessive Step Size

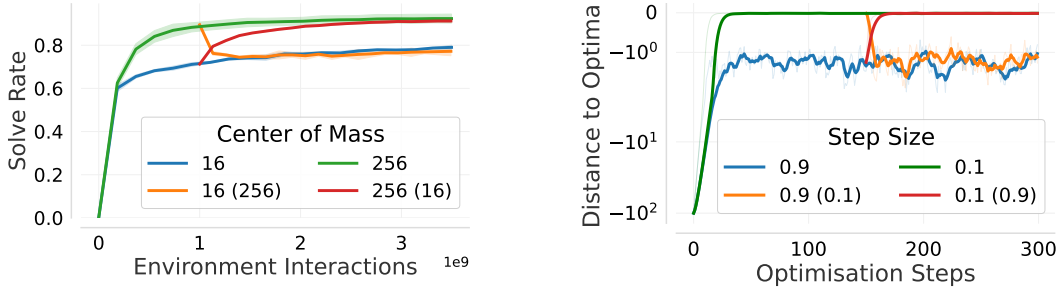
While the similarity in the learning curves is striking, it does not provide insight into the mechanisms by which the outer loop step size influences learning progress. As illustrated in Figure 2a, large learning rates in gradient descent induce updates that bounce around the local minimum, experiencing large gradient norms but no decrease in the loss. Figures 2b and 2c show an analogous effect in PPO agents: large outer-loop step sizes due to weak regularisation result in performance stagnating despite large policy updates and gradient norms.[2] This suggests that the performance plateau is caused by thrashing around a local optimum rather than converging to a suboptimal stationary point.



(a) Update size (SGD) (b) Gradient Norm (c) KL (Behavior Policy)

Figure 2: (a) In stochastic optimisation, the update magnitude is consistently large when the step size is too large, despite a stagnating loss. (b,c) Showing that PPO shares similar dynamics.

We next confirm that these plateaus are a direct consequence of the outer step size rather than the policy network being unable to learn or the generated data being insufficient to learn from. Figure 3a shows that increasing the proximal policy's COM (thereby reducing the outer step size) after the agent has plateaued allows it to immediately resume learning, ultimately recovering the same asymptotic performance as the higher COM. Moreover, if we reduce the COM, performance drops to the suboptimal plateau associated with the larger step size, exactly matching the behaviour of increasing the learning rate in noisy stochastic optimisation, shown in Figure 3b.



(a) PPO-EWMA (b) SGD

Figure 3: (a) Loading checkpoints and retraining with a different COM recovers the performance of the most recent regularisation strength. The legend indicates the center of mass, and the number in brackets indicates the starting COM. (b) The same phenomenon occurs in stochastic optimisation.

3.2 Decoupling the Inner and Outer Loops

Having established the importance of the outer step size in influencing an agent's plateauing behaviour, we close off this section by comparing how different properties of the inner and outer loop differ, and which hyperparameters influence each process in Table 1.

Furthermore, to demonstrate that changing the inner loop cannot always compensate for a poor outer loop step size, we tune the learning rate (and sweep over annealing vs not annealing it to zero over

[2] While the raw gradient norm is high, we perform standard gradient clipping whenever the norm is above 0.5.

Table 1: The differences between properties of the inner loop (parameter-space updates) and the outer loop (policy-space updates) in PPO. $J(\pi)$ is defined as the expected discounted return of π.

Property	Inner loop	Outer loop
Learning rate	Adam learning rate η	Regularisation (COM, ϵ) & epochs
Noise	Minibatch vs rollout batch	Rollout batch vs true gradient
Curvature	Neural network hessian $\|H_\theta\|$	Policy landscape $\|\nabla_\pi J(\pi)\|$

the course of training) separately for each center of mass and show the results in Figure 4. Here we can see that if we have an outer loop step size that is too large (with COM = 8), then regardless of the learning rate, performance is still significantly worse than when we tune the COM appropriately. Further, we see that the same learning rate is roughly optimal for all COMs, showing that these two hyperparameters affect different learning mechanisms and, importantly, are not interchangeable.

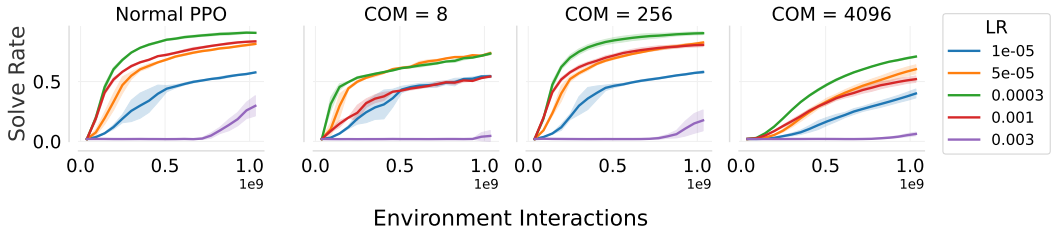
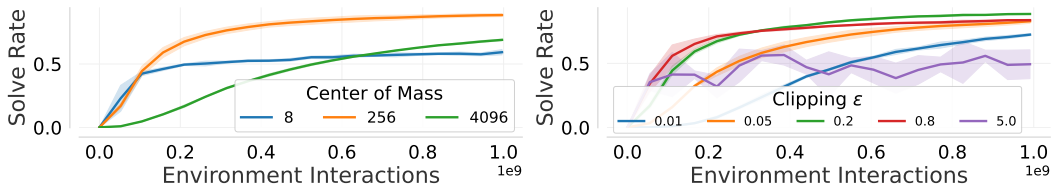


Figure 4: Tuning the learning rate cannot counteract a poor outer step size. Here we sweep over whether or not to anneal LR for each run, and show the best result per learning rate.

4 Understanding PPO's Outer Loop

Having established PPO's similarities to stochastic optimisation, we next focus on understanding which hyperparameters modulate the outer step size and noise level. We first show in Section 4.1 that *regularisation* towards previous policies directly influences the outer step size; next, in Section 4.2, we demonstrate that the number of optimisation epochs we perform also controls the agent's plateauing behaviour; in Section 4.3, we show that, similarly to SGD, the update noise matters too, and larger batch sizes admit higher step sizes without plateauing, whereas smaller batch sizes are very susceptible to overly large steps. Finally, Section 4.4 suggests some rules of thumb for how to set the step size appropriately, and how this depends on the available computational budget.

4.1 Regularisation



(a) Changing COM

(b) Changing ϵ

Figure 5: Weak regularisation, corresponding to either (a) too low of a COM or (b) too large of a clipping ϵ can lead to premature plateaus.

We consider two ways to control regularisation: either altering the center of mass of the proximal policy in PPO-EWMA (controlling how old the policy is we regularise towards) or changing PPO's

clipping ϵ parameter. As shown in Figure 5, weak regularisation (either via a high ϵ or low COM) leads to premature plateaus, whereas overly strong regularisation (low ϵ or high COM) leads to slow learning relative to the available environment sample budget.

4.2 Optimisation Epochs

In Figure 6, we show that changing the number of inner optimisation epochs we perform on each batch of data also has a direct effect on an agent's plateauing behaviour. This is again dependent on the strength of the regularisation, where weak regularisation plateaus more easily when increasing the number of epochs, and strong regularisation learns slowly with a small number of epochs. One

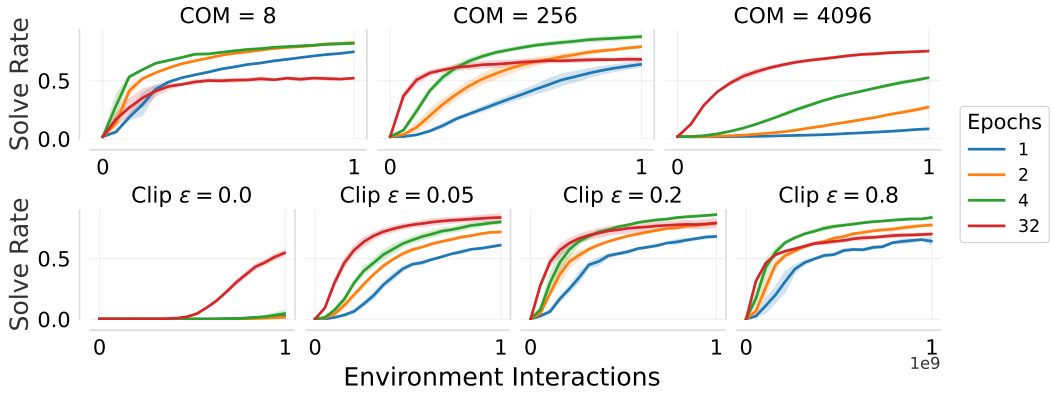


Figure 6: How changing number of epochs influences (top) PPO-EWMA and (bottom) normal PPO. Stronger regularisation can partially alleviate plateaus due to too many epochs, thereby reaching higher asymptotic performance. See Appendix A for full hyperparameters and experimental details.

notable result in bottom left panel is that even with a clipping term of $\epsilon = 0$, a large number of epochs can be beneficial. This in particular is likely due to the Adam momentum term, and the fact that the PPO update can overshoot the ϵ threshold (Ilyas et al., 2018; Wang et al., 2020). Specifically, clipping only zeros out the gradients once the ratio *already* exceeds the threshold, so if the first update step is large (either because of a large (inner) learning rate, or a large momentum term as is the case here), the ratio after this initial update can be far outside the $1 \pm \epsilon$ range. See Appendix B for more details.

4.3 Rollout Batch Size

Following from the analogy to stochastic optimisation, if the core problem is that we take steps that are too large on noisy targets, then another solution would be to increase the signal-to-noise ratio via larger batches (Smith et al., 2018; McCandlish et al., 2018). To investigate this, we compare the performance of agents when varying the number of parallel environments, and keeping the same number of minibatches—meaning we change only the minibatch size, keeping everything else constant. Further, to isolate the impact of update *quality*, we compare agents based on the number of policy update steps; therefore, agents with larger batches do see more data, and the comparison is not fair in terms of environment transitions. Nevertheless, it allows us to analyse how much the quantity of data per update step changes the effect of regularisation.

Figure 7 shows that larger batch sizes are significantly more robust to weaker regularisation than smaller ones. For instance, if we have a small batch size (e.g., 4096), weak regularisation (e.g. COM of 8 or $\epsilon = 0.6$) performs significantly worse compared to when it is paired with a larger batch size.[3] This suggests that the higher signal-to-noise ratio achieved through larger batches admits

[3]While these minibatches may seem large, they are consistent with recent work on hardware-accelerated environments, which achieve speedups via parallelisation (Makoviychuk et al., 2021; Nikulin et al., 2023)

larger outer step sizes without plateauing, again echoing known results from stochastic gradient descent (Krizhevsky, 2014; Goyal et al., 2017; Smith et al., 2018; McCandlish et al., 2018).

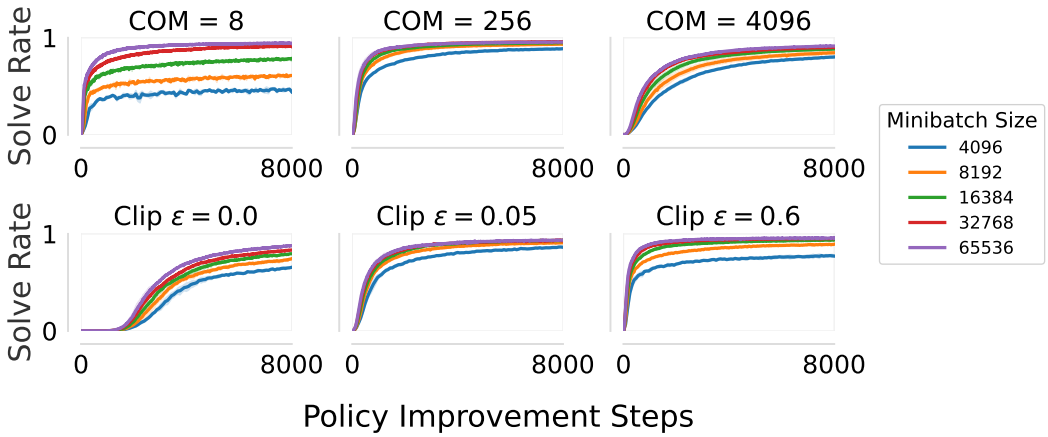


Figure 7: Showing the effect of larger minibatches when changing the (top) COM of θ_{prox} in PPO-EWMA; and (bottom) clipping ϵ term in standard PPO. Here the x-axis is the number of policy update steps. Larger batches are less susceptible to plateauing when paired with weak regularisation.

4.4 Choosing an Appropriate Step Size

The analysis from this section suggests that the important factors influencing whether or not an agent plateaus at a suboptimal performance ceiling are (a) the number of transitions we use per policy update step, and (b) the size of the deviation from the reference policy. We next turn to the question of how to set the corresponding hyperparameters to reasonable values. To do so, we unify (a) and (b) into the Data to Divergence Ratio (DDR): the number of data points per unit KL divergence from the behaviour policy. Figure 8 shows that performance suffers at both ends of the DDR spectrum; however, the mechanisms underlying this behaviour are distinct for each extreme. Low DDR values lead to early plateaus (and therefore do not improve much when given additional training time), whereas high DDR values cause learning to progress slowly, meaning that these agents fail to reach their performance ceiling within the fixed sample budget. However, since slower learning is acceptable under larger interaction budgets, our results suggest that *as we increase the training budget, we should increase the DDR accordingly to avoid a premature plateau.*

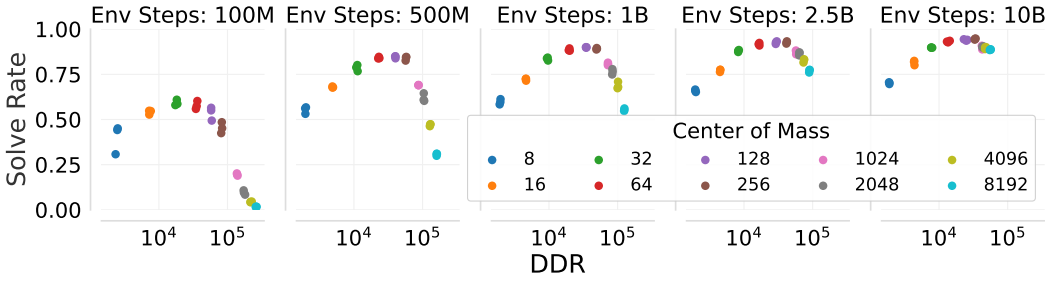


Figure 8: DDR (averaged over training) vs. the maximum solve rate achieved in that run for various compute budgets. Each dot is a single training run and different dots of the same colour are different random seeds. The dashed red line indicates the approximate performance ceiling.

Having established the importance of increasing the DDR as we train for more samples, we propose that one simple way to do so is to increase the number of parallel environments. This directly increases the amount of data per policy improvement step (thereby reducing the update noise) and

indirectly lowers the outer step size due to the behaviour policy age, measured in environment samples, increasing (Hilton et al., 2022). However, it remains unclear how we should adjust the other hyperparameters when increasing parallelisation, and we address this question in the next section.

5 A Reliable Recipe for Scaling Parallelisation in PPO

Our results thus far suggest that we need to scale down the outer step size as our computational budget increases in order to avoid premature stagnation. In addition, increasing the number of parallel environments is a desirable way to do so, since it reduces both the step size and the update noise, while allowing more samples to be processed within the same amount of wall-clock time. However, when we increase the number of parallel environments in PPO, we have more data per policy update step, and this necessitates adjusting some of the other hyperparameters. There are three primary ways to partition this data:

1. Have more minibatches of the same size.
2. Have larger minibatches with the same learning rate.
3. Have larger minibatches, and scale the learning rate according to the square-root rule for Adam (Krizhevsky, 2014; Malladi et al., 2022; Granziol et al., 2022; Hilton et al., 2022).

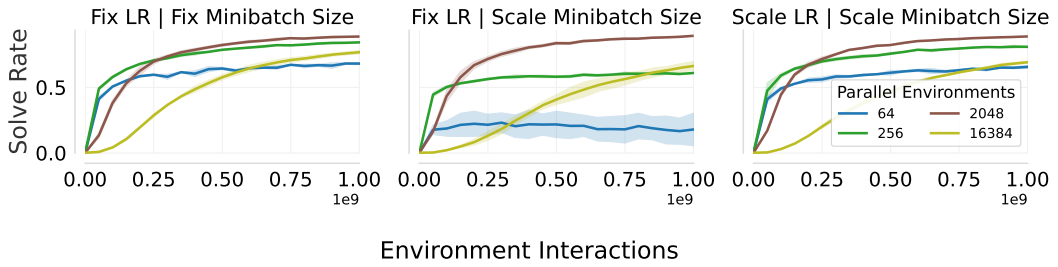


Figure 9: Comparing different approaches when varying N_{envs}. Keeping the inner optimisation process unchanged performs best, whereas performance of small minibatches suffers when scaling the minibatch size without adjusting the learning rate. See Appendix C for more granular plots.

Figure 9 shows that having more minibatches while keeping everything else fixed works reliably. This recipe preserves the dynamics of the inner optimisation process—with an unchanged learning rate, minibatch size, etc., and merely changes the number of optimisation steps we do. This strategy is further justified by Section 3, where we show that the optimal inner and outer step sizes are largely independent. However, larger minibatches (with a scaled learning rate) tend to result in better hardware utilization, and thus faster training.[4] While larger minibatches can work well in certain environments, Figure 10 shows that they sometimes lead to training instability and lower plateaus (Do et al., 2024; Su et al., 2025). In summary, we recommend a stability-first procedure: increase the number of minibatches while keeping the learning rate and minibatch size fixed. Only increase minibatch size (adjusting the learning rate appropriately) if hardware utilization is a bottleneck.

5.1 Robotics Results

To demonstrate the practical utility of our scaling recipe, we consider a set of difficult robotics tasks from Isaacgym (Makoviychuk et al., 2021) used by Singla et al. (2024). The default minibatch size for several tasks in Isaacgym is 16384.[5] However, when increasing parallelisation, Singla et al. (2024) fix the learning rate and increase the minibatch size to $4\times$ the number of parallel

[4] This is typically beneficial when compute bound (e.g. by using larger models), whereas in our sample-bound locomotion task, the wall-clock gains are marginal (see Figure 13 in Appendix C).

[5] https://github.com/isaac-sim/IsaacGymEnvs/blob/main/isaacgymenvs/cfg/train/AllegroHandLSTM_BigPPO.yaml#L88

environments—resulting in a minibatch size of 98304 for 24576 environments. Following our recommendations, we make a single adjustment: we revert the minibatch size back to the default 16384 for both PPO and SAPG, the new method introduced by Singla et al. (2024). Figure 10 shows that this one change significantly outperforms the default setting for both methods, makes PPO more amenable to additional parallelisation, and reduces the performance gap between it and SAPG.

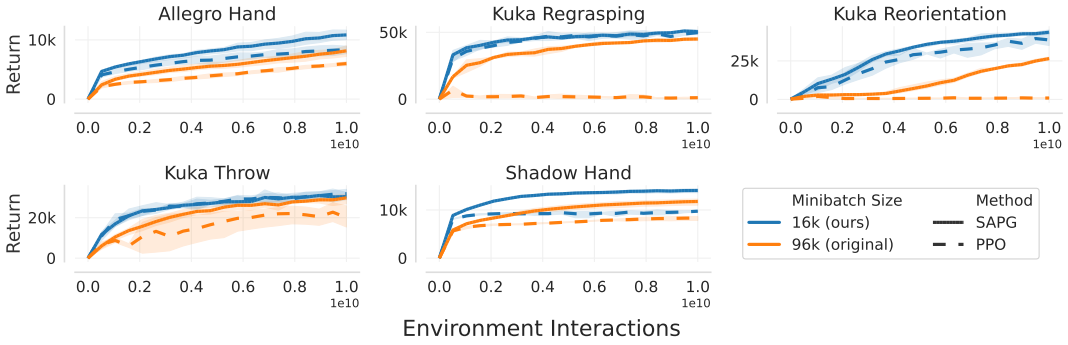


Figure 10: We take the code from Singla et al. (2024), and make **one** change—setting the minibatch size to 16k (which is the default in Isaacgym) instead of using 96k like Singla et al. (2024) do. When using our recommendations, vanilla PPO performs much better across the board, and the gap between it and SAPG is reduced. Furthermore, SAPG also benefits from the same change.

6 Batch Size Scaling Enables Open-Ended Learning

Finally, we show that, by using our analysis, we can overcome learning stagnation in the challenging open-ended domain of `Kinetix` (Matthews et al., 2025). In this setting, agents train on a procedurally-generated distribution of tasks with the objective of achieving robust generalisation over the entire space of tasks. There are three different training distributions (**`s`**`mall`, **`m`**`edium` or **`l`**`arge`), corresponding to the maximum number of entities there are in the scene; each of these is treated as a separate experiment. The best performing approach on `Kinetix` is SFL (Rutherford et al., 2024)—an autocurriculum method that samples training tasks that have high learnability (i.e., those where the agent has about a 50% chance of success, meaning they are neither too easy nor too hard). Like much of the field of Unsupervised Environment Design, SFL uses PPO as the underlying learning algorithm (Dennis et al., 2020; Jiang et al., 2021; Parker-Holder et al., 2022).

As a way to measure how well the agent is performing on the full distribution of tasks, we calculate performance on a fixed set of environments randomly sampled from the training distribution. In Figure 11 we show that the default configuration used by Matthews et al. (2025) plateaus early, and its performance even starts degrading when given more samples. This means that any additional compute is effectively wasted. However, by simply increasing the number of parallel environments (using our scaling recipe),[6] we are able to sustain performance improvement for much longer, ultimately reaching significantly higher performance. This effect is more pronounced in the more difficult and wider `large` distribution of tasks, whereas 65k environments seems sufficient for the `small` setting. Importantly, we find that we can reliably scale to over 1M parallel environments (512× more than Matthews et al. (2025) used) across 128 GPUs, and this level of parallelisation is imperative in order to be able to collect more than a trillion environment transitions within a reasonable wall-clock time. As predicted by the shifting optima in Figure 8, such large training budgets require the small, high-quality updates provided by scaling up to avoid premature plateaus.

We note that as we use additional GPUs, we process more environments in the filtering stage (since this is effectively free in terms of wall-clock time); however, Appendix E shows that this alone is insufficient to prevent stagnation unless paired with increased training parallelisation.

[6]For wall-clock reasons, at the cost of some learning efficiency, we use a hybrid approach, where we have 32× the number of minibatches, each minibatch is 16× the size, and the learning rate is 4× larger. See Appendix D for more details.

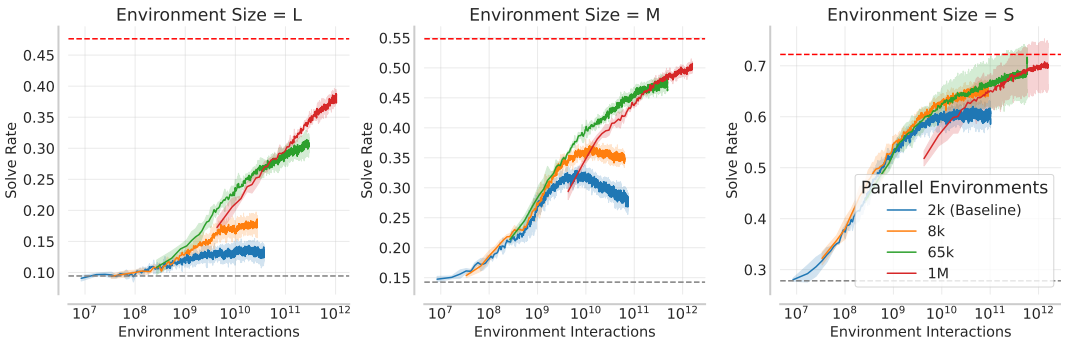


Figure 11: SFL on `Kinetix`, showing that increasing the number of parallel environments maintains performance improvement for much longer. The dashed red line is an approximation of optimal performance, since not all sampled environments are solvable, while the grey line indicates a random policy's performance. We plot mean and 95% CI over 3 seeds. The curves are truncated at different x-values since using fewer parallel environments takes a much longer wall-clock time to generate a particular number of transitions. We run the baseline from Matthews et al. (2025) for longer to clearly show the performance degradation.

7 Related Work

There has been a long line of work that demonstrates scaling RL tends to be much more difficult and less straightforward than supervised learning (Ota et al., 2021; Bjorck et al., 2021; Schwarzer et al., 2023; Ceron et al., 2024; Lee et al., 2025a;b; Rybkin et al., 2025; Wang et al., 2025). While model scaling is a common topic of study (Lee et al., 2025a;b), less focus has been put on data scaling, i.e., what happens when we give agents a massive amount of *online* experience. Part of this has been due to the difficulty in running these experiments; however, the recent wave of GPU-accelerated RL environments has made it feasible to study these questions with only modest hardware requirements (Freeman et al., 2021; Nikulin et al., 2024; Bonnet et al., 2024; Matthews et al., 2024; 2025). On this topic, Bharthulwar et al. (2025) demonstrate that one benefit of parallelisation is an increase in data diversity, but that this is not always the case when all parallel environments are very similar. They propose to stagger their resets, so that each worker simulates temporally unrelated chunks of experience. This provides an added explanation for why increasing the parallelisation is so effective in `Kinetix`—since nearly every parallel environment is simulating a unique environment, the diversity of experience is directly influenced by the parallelisation. Relatedly, McLean et al. (2025) show that increasing the number of tasks an agent trains on can reduce plasticity loss, but they consider only the 10 and 50 tasks from MetaWorld (Yu et al., 2020), significantly less than the millions of tasks we train on. Finally, Mayor et al. (2025) investigate the tradeoffs between scaling parallelisation and rollout length in PPO, and find that for a fixed data budget, more parallelisation tends to be preferred. By contrast, we find that *more* data per rollout batch has benefits.

Our work shows that some of the instabilities of PPO are similar to pathologies from classic optimisation theory (Robbins & Monro, 1951; Luenberger et al., 1984; Bertsekas, 1997), and therefore many remedies from that field are related (Polyak, 1964; Nocedal & Wright, 2006). However, we are not the first to view PPO's outer step as analogous to an optimisation problem. Tan et al. (2024) directly view the difference in parameters across successive PPO updates as a "gradient", and either use momentum or apply the update with an outer "learning rate" that is not equal to 1. By contrast, we use our conceptual model to understand PPO's behaviour better, thereby finding better ways to train agents that do not stagnate without altering the underlying, tried and tested, algorithm.

This paper is also related to two-timescale analyses of reinforcement learning, a field that includes learning nonlinear representations and linear value functions at different timescales (Chung et al., 2018), and the decoupled optimisation of the actor and critic (Wu et al., 2020; Zeng et al., 2024; Zeng & Doan, 2024). However, instead of analysing theoretical implications or developing new

algorithms, we investigate the empirical similarities between PPO and stochastic optimisation, and how this provides practical guidance for preventing premature plateaus.

Another related field is that of open-endedness, where the goal is to obtain an algorithm that we can run forever, and that will continually result in novel artifacts (Stanley, 2019; Dennis et al., 2020; Team et al., 2021; Parker-Holder et al., 2022; Team et al., 2023). However, one prerequisite for such an algorithm is agents that do not stagnate (Hughes et al., 2024)—which we have shown can happen in several different domains, and that increasing parallelisation is one way to alleviate this issue.

Finally, our work sheds some light on the folk knowledge that higher parallelisation levels tend to result in faster wall-clock training times, at the cost of worse sample efficiency (Freeman et al., 2021; Makoviychuk et al., 2021). As Hilton et al. (2022) originally showed, and we confirmed, this is largely due to the implicitly stronger regularisation at higher numbers of parallel environments. One remedy for this is to use PPO-EWMA, with a fixed COM, but a large number of parallel environments—leading to a consistent level of regularisation while achieving fast wall-clock times.

8 Limitations & Future work

As with any conceptual model, there are simplifications and situations where the model may not apply. While we have shown that our model makes accurate predictions in certain settings, future work should investigate where this model breaks down, and where additional care is required. In particular, our analysis focuses solely on dense reward tasks, which can be viewed similarly to smooth optimisation problems, where small changes in the input lead to small changes in the output. Future work is required to more deeply understand sparse-reward tasks and environments that rely on strong exploration. Additionally, scaling parallelisation can be cost-prohibitive or impossible in many settings. However, one can perfectly simulate a larger number of parallel environments by accumulating experience into a buffer before training on it, although this requires additional storage, and may be too slow for practical purposes. This is why we performed our final experiments on Kinetix, as its hardware-accelerated implementation allows us to collect trillions of transitions within a reasonable wall clock time. In addition, prior baselines in Kinetix were far from optimal, so there is a lot of headroom to show that our techniques can be helpful, which is not the case in many other benchmarks. In Appendix G we have preliminary results in Isaacgym, showing that increasing parallelisation leads to higher asymptotic performance; however, future work should thoroughly investigate how the insights from our work transfer to other domains. In this vein, Khatri et al. (2026) investigate scaling LLM RL, also finding that larger batch sizes lead to higher asymptotic performance, suggesting that our results may have wide applicability. Finally, using a fixed batch size throughout training may not be optimal, and investigating adaptive batch size techniques is another promising avenue for future work; see Park (2026) for an interesting piece of work in this direction.

9 Conclusion

While there are many causes of plateaus in RL, in this work we demonstrate that under-regularisation is one reason behind learning stagnation in deep on-policy algorithms. By comparing PPO to stochastic optimisation, we find that many of the pathologies in the latter setting can occur in the former setting if the outer step size is too large. However, this is easy to remedy, by either increasing the regularisation (e.g., decreasing ϵ or increasing the proximal policy's COM in PPO-EWMA) or directly increasing the parallelisation factor. Based on our insights, we recommend a simple approach to scaling PPO—keep the minibatch size fixed as the number of parallel environments is changed—which allows it to remain competitive with more complex methods, and reliably scale to more than 1M parallel environments. Finally, we demonstrate that, in a difficult and open-ended setting where current approaches are far from optimal, simply increasing the parallelisation leads to performance increasing monotonically across orders of magnitudes more experience. Ultimately, we hope our work is one step in the direction of designing RL algorithms that can predictably scale with additional compute, and continue to benefit from additional experience indefinitely.

Acknowledgements

Thank you to Charlie Cowen-Breen and Vincent Roulet for helpful discussions throughout the course of this project. Part of the compute for this work was provided by the Isambard-AI National AI Research Resource, under the project "FLAIR 2025 Moonshot Projects". MB is funded by the Rhodes Trust. JF is partially funded by the UKRI grant EP/Y028481/1 (originally selected for funding by the ERC), the JPMC Research Award and the Amazon Research Award.

References

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334, 1997.

Sid Bharthulwar, Stone Tao, and Hao Su. Staggered environment resets improve massively parallel on-policy reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=hesM5BWtOJ.

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

Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence I. Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries P. Smit, Nathan Grinsztajn, Raphael Boige, Cemlyn N. Waters, Mohamed A. Mimouni, Ulrich A. Mbou Sob, Ruan de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Jumanji: a diverse suite of scalable reinforcement learning environments in jax, 2024. URL https://arxiv.org/abs/2306.09884.

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

Wesley Chung, Somjit Nath, Ajin Joseph, and Martha White. Two-timescale networks for nonlinear value function approximation. In *International conference on learning representations*, 2018.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Khoi Do, Minh-Duong Nguyen, Nguyen Tien Hoa, Long Tran-Thanh, Nguyen H Tran, and Quoc-Viet Pham. Revisiting lars for large batch training generalization of neural networks. *IEEE Transactions on Artificial Intelligence*, 6(5):1321–1333, 2024.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *arXiv preprint arXiv:2005.12729*, 2020.

C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation, 2021. URL http://github.com/google/brax.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Diego Granziol, Stefan Zohren, and Stephen Roberts. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *Journal of Machine Learning Research*, 23(173):1–65, 2022.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098, 2022.

Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning. *arXiv preprint arXiv:2301.13442*, 2023.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022a. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022b. URL http://jmlr.org/papers/v23/21-1342.html.

Edward Hughes, Michael D Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktäschel. Position: Open-endedness is essential for artificial superhuman intelligence. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=Bc4vZ2CX7E.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Are deep policy gradient algorithms truly policy gradient algorithms. *arXiv preprint arXiv:1811.02553*, 2018.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FMjeC9Msws.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL http://github.com/RobertTLange/gymnax.

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

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

David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

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

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the causes of plasticity loss in neural networks. 274:750–783, 29 Jul–01 Aug 2025. URL https://proceedings.mlr.press/v274/lyle25a.html.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/28dd2c7955ce926456240b2ff0100bde-Abstract-round2.html.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35: 7697–7711, 2022.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *ICML*, 2024.

Michael Matthews, Michael Beukman, Chris Lu, and Jakob Foerster. Kinetix: Investigating the training of general agents through open-ended physics-based control tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.23208.

Walter Mayor, Johan Obando-Ceron, Aaron Courville, and Pablo Samuel Castro. The impact of on-policy parallelized data collection on deep reinforcement learning networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine*

Learning, volume 267 of *Proceedings of Machine Learning Research*, pp. 43331–43352. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/mayor25a.html.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Reginald McLean, Evangelos Chatzaroulas, J K Terry, Isaac Woungang, Nariman Farsad, and Pablo Samuel Castro. Multi-task reinforcement learning enables parameter scaling. In *Reinforcement Learning Conference*, 2025. URL https://openreview.net/forum?id=eBWwBIFV7T.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 37:113038–113071, 2024.

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

Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Viacheslav Sinii, Artem Agarkov, and Sergey Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX. In *Intrinsically-Motivated and Open-Ended Learning Workshop, NeurIPS2023*, 2023. URL https://openreview.net/forum?id=xALDC4aHGz.

Alexander Nikulin, Ilya Zisman, Alexey Zemtsov, Viacheslav Sinii, Vladislav Kurenkov, and Sergey Kolesnikov. Xland-100b: A large-scale multi-task dataset for in-context reinforcement learning. *CoRR*, abs/2406.08973, 2024. DOI: 10.48550/ARXIV.2406.08973. URL https://doi.org/10.48550/arXiv.2406.08973.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

Kei Ota, Devesh K Jha, and Asako Kanezaki. Training larger networks for deep reinforcement learning. *arXiv preprint arXiv:2102.07920*, 2021.

Jongchan Park. Scalable reinforcement learning via adaptive batch scaling. In *ICML*, 2026.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the International Conference on Machine Learning*, pp. 17473–17498. PMLR, 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, et al. Jaxmarl: Multi-agent rl environments in jax. *arXiv preprint arXiv:2311.10090*, 2023.

Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. *Advances in Neural Information Processing Systems*, 37:16071–16101, 2024.

Oleh Rybkin, Michal Nauman, Preston Fu, Charlie Victor Snell, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Value-based deep RL scales predictably. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=FLPFPYJeVU.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Jayesh Singla, Ananye Agarwal, and Deepak Pathak. Sapg: split and aggregate policy gradients. *arXiv preprint arXiv:2407.20230*, 2024.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1Yy1BxCZ.

Kenneth O Stanley. Why open-endedness matters. *Artificial life*, 25(3):232–235, 2019.

Huangyuan Su, Mujin Kwun, Stephanie Gil, Sham Kakade, and Nikhil Anand. Characterization and mitigation of training instabilities in microscaling formats. *arXiv preprint arXiv:2506.20752*, 2025.

Adrien Ali Taiga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. On bonus-based exploration methods in the arcade learning environment. *arXiv preprint arXiv:2109.11052*, 2021.

Charlie B Tan, Edan Toledo, Benjamin Ellis, Jakob N Foerster, and Ferenc Huszár. Beyond the boundaries of proximal policy optimization. *arXiv preprint arXiv:2411.00666*, 2024.

Adaptive Agent Team, Jakob Bauer, Kate Baumli, Satinder Baveja, Feryal M. P. Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, Karol Gregor, Edward Hughes, Sheleem Kashem, Maria Loks-Thompson, Hannah Openshaw, Jack Parker-Holder, Shreya Pathak, Nicolas Perez Nieves, Nemanja Rakicevic, Tim Rocktäschel, Yannick Schroecker, Jakub Sygnowski, Karl Tuyls, Sarah York, Alexander Zacherl, and Lei Zhang. Human-timescale adaptation in an open-ended task space. *CoRR*, abs/2301.07608, 2023. DOI: 10.48550/arXiv.2301.07608. URL https://doi.org/10.48550/arXiv.2301.07608.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michaël Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents. *CoRR*, abs/2107.12808, 2021. URL https://arxiv.org/abs/2107.12808.

Sebastian B Thrun. *Efficient exploration in reinforcement learning*. Carnegie Mellon University, 1992.

Edan Toledo. Stoix: Distributed Single-Agent Reinforcement Learning End-to-End in JAX, April 2024. URL https://github.com/EdanToledo/Stoix.

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

Yuhui Wang, Hao He, and Xiaoyang Tan. Truly proximal policy optimization. In Ryan P. Adams and Vibhav Gogate (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 113–122. PMLR, 22–25 Jul 2020. URL https://proceedings.mlr.press/v115/wang20b.html.

Yue Frank Wu, Weitong Zhang, Pan Xu, and Quanquan Gu. A finite-time analysis of two time-scale actor-critic methods. *Advances in Neural Information Processing Systems*, 33:17617–17628, 2020.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carlo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer., 2025. URL https://github.com/google-deepmind/mujoco_playground.

Sihan Zeng and Thinh Doan. Fast two-time-scale stochastic gradient method with applications in reinforcement learning. In *The Thirty Seventh Annual Conference on Learning Theory*, pp. 5166–5212. PMLR, 2024.

Sihan Zeng, Thinh T Doan, and Justin Romberg. A two-time-scale stochastic optimization framework with applications in control and reinforcement learning. *SIAM Journal on Optimization*, 34 (1):946–976, 2024.

Chujie Zheng, Kai Dang, Bowen Yu, Mingze Li, Huiqiang Jiang, Junrong Lin, Yuqiong Liu, Hao Lin, Chencan Wu, Feng Hu, et al. Stabilizing reinforcement learning with llms: Formulation and practices. *arXiv preprint arXiv:2512.01374*, 2025.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Experimental Details

For all of our investigative experiments, we use the `Jax2D` physics engine (Matthews et al., 2025), where we construct a set of 512 procedurally-generated 2D robotic locomotion tasks. The goal in each task is to move the morphology to the goal position, marked by a blue circle, and we measure performance by calculating average success rate over these tasks, in line with Matthews et al. (2025). Furthermore, we use Stoix's implementation of PPO (Toledo, 2024) for these experiments. All experiments used NVIDIA A100 40GB GPUs.

A.1 Hyperparameters

For all experiments, we run five independent seeds, and plot the mean solve rate or return, with the 95% CI shaded. For the Kinetix SFL experiments, we run three seeds due to computational constraints.

Locomotion Tasks For all of the locomotion experiments, the hyperparameters stayed mostly the same, with the exception of the hyperparameters we sweep over for a particular plot. The default settings are given in Table 2. We use a 3-layer MLP with width 256.

SFL The PPO hyperparameters are given in Table 2, and the SFL environment filtering parameters are shown in Table 3. We use the same exact values as Matthews et al. (2025) for the 2048 parallel environment run, but adjust these hyperparameters as we use additional hardware. In particular, since filtering is trivially parallelizable, we increase the number of levels we search through as we increase the number of GPUs without a noticeable wall-clock-time penalty.

SAPG We use the same code and hyperparameters as Singla et al. (2024) do (see Table 4), with the exception of the minibatch size. We run for 10B timesteps.

Stochastic Optimisation Example The stochastic optimisation problem we consider is to minimise $\mathbf{x}^T\mathbf{x}$, with $\mathbf{x} \in \mathbb{R}^{50}$. We perform standard gradient descent, but add noise with standard deviation $\frac{3}{\sqrt{50}}$ to the gradients. To more clearly demonstrate the similarities to RL, we plot the negative of the euclidean distance to the optimal solution $\mathbf{x}^* = \mathbf{0}$.

Table 2: Hyperparameters for the investigative and large-scale open-ended learning experiments.

Parameter	Jax2D Locomotion	SFL
γ	0.995	0.995
λ_{GAE}	0.9	0.9
PPO number of steps	256	256
PPO epochs	8	8
PPO ϵ	0.2	0.2
PPO max gradient norm	0.5	0.5
PPO value clipping	yes	yes
Value loss coefficient	0.5	0.5
Entropy coefficient	0.01	0.01
PPO # parallel environments	2048	—
Adam learning rate	0.0003	5e-5
PPO minibatches per epoch	32	—

Table 3: Configuration for the different SFL (Rutherford et al., 2024) runs. L is the rollout length used to compute the learnability score per environment, ρ is the fraction of high-learnability environments used (the rest being filled with random levels), N is how many levels we search through and K is how many levels we save. Finally, T is the number of PPO update steps between buffer updates, where each iteration consists of $256 \cdot N_{\text{envs}}$ transitions. We set T such that we have the same number of environment transitions between buffer update steps for the 8k/65k/1M settings.

Parameter	**2048**	**8192**	**65536**	**1M**
Rollout Length L	512	512	512	512
Sample Ratio ρ	0.5	0.5	0.5	0.5
Filtering Batch Size N	12288	256k	256k	4M
Update Period T	128	256	32	2
Buffer Size K	1024	8192	8192	8192

Table 4: Training Hyperparameters for AllegroKuka, Shadow Hand, and Allegro Hand. The values here were taken directly from Singla et al. (2024) and our results were obtained using the code here.

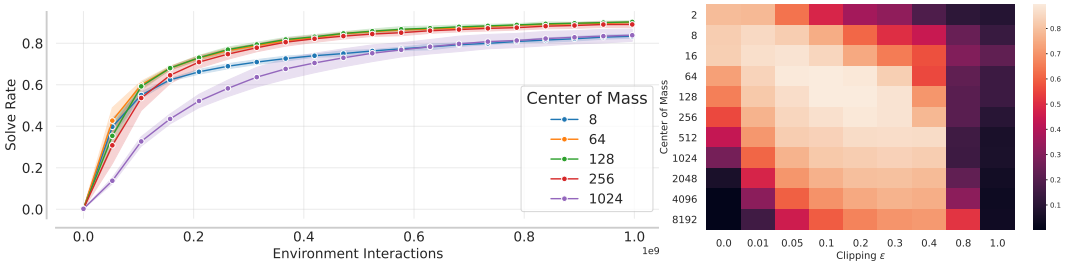
Hyperparameter	**AllegroKuka**	**Shadow Hand**	**Allegro Hand**
Discount factor, γ	0.99	0.99	0.99
τ	0.95	0.95	0.95
Learning rate	1e-4	5e-4	5e-4
KL threshold for LR update	0.016	0.016	0.016
Grad norm	1.0	1.0	1.0
Entropy coefficient	0	0	0
Clipping factor ϵ	0.1	0.1	0.2
Critic coefficient λ'	4.0	4.0	4.0
Horizon length	16	8	8
LSTM Sequence length	16	—	—
Bounds loss coefficient	0.0001	0.0001	0.0001
Mini epochs	2	5	5

B Center of Mass vs ϵ

In Figure 12, we compare the effect of changing the COM vs changing the clipping ϵ. One takeaway from this plot is that we can mostly counteract changes in one of these quantities by appropriately altering the other, suggesting that both of these settings act on the same mechanism.

However, one difference is in the susceptibility of the agent to ϵ-overshooting. As mentioned in Section 4.2, ϵ does not actually constrain the ratio to be within the $1 \pm \epsilon$ range; instead, it stops updates once the ratio already exceeds the bounds. Therefore, if the Adam learning rate is high enough, there is little difference between all ϵ values lower than some threshold, which is roughly how much one gradient step can change the probability ratio of a particular action. However, when increasing the COM of PPO-EWMA, this overshooting is less of a problem, since we are measuring the ratio with respect to the (potentially quite old) proximal policy, meaning that the gradients are only non-zero when the proximal policy has caught up enough with the current behaviour policy, i.e., when the ratio is within the $1 \pm \epsilon$ range.

Finally, we find that when studying very weak regularisation, a low COM in PPO-EWMA tends to be more stable than a high ϵ (e.g., compare the first row of Figure 12b with the last column). One potential reason for this is that with a large ϵ, single, perhaps unreliable transitions can lead to large gradients which can drown out the actual signal. These large gradients can also potentially cause destructive weight updates that completely change the policy's behaviour.



(a) (b)

Figure 12: Comparing the effect of COM vs ϵ. (a) Showing the performance of the *best* ϵ for various COMs, showing that we can find an ϵ to (mostly) counteract the effect of changing the COM in PPO-EWMA. (b) A heatmap of final performance for a 2D grid search over the PPO-EWMA COM and ϵ. Overall, most reasonable values of the COM have a corresponding ϵ that performs well; however, extreme values of ϵ are too unstable to learn.

C Additional Scaling Results

Figure 13 shows the number of environment steps per second we can process in the locomotion task, when taking into account all learning and environment stepping. We see that there is little difference between the various scaling approaches, and this translates to little difference in overall wall-clock time. Figure 14 provides a condensed version of Figure 9, whereas Figure 15 contains more granular data.

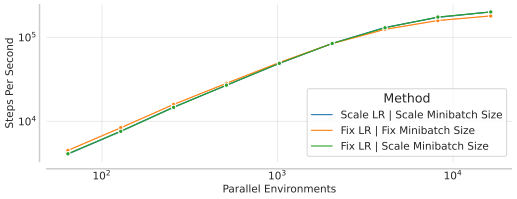


Figure 13: Plotting the steps per second for the training, which includes the environment step and the neural network optimisation. This corresponds to the same results as in Figure 9.

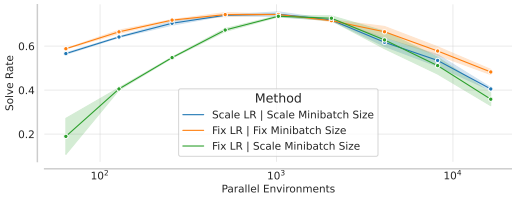


Figure 14: Comparing different approaches when changing the number of parallel environments.

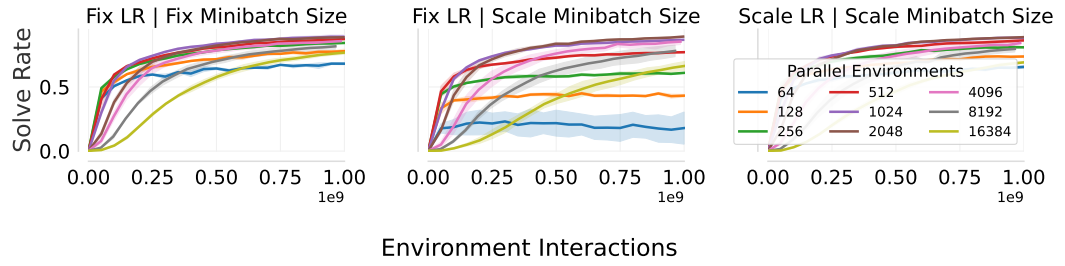


Figure 15: A version of Figure 9 with more options for the number of parallel environments.

D SFL Scaling

For the primary SFL results, we scaled to 1048576 parallel environments, which is $512\times$ more than the default used by Matthews et al. (2025). According to our recipe, this means we must have $512\times$ the number of minibatches, i.e., 16384 instead of the default 32. However, we parallelise

across 128 GPUs, meaning each GPU has 8192 parallel environments, and due to how the baseline code is written (which we wanted to keep unchanged), we cannot have more minibatches than we have parallel environments. In other words, we must have 8192 or fewer minibatches. For our main results, we use 1024 minibatches (32× more than the default), meaning each minibatch is 16× larger than the default; we therefore scale the learning rate by $4 = \sqrt{16}$ to account for this. This setting provides a balance between performance and wall-clock time, and the difference in performance between the 1024 minibatch setting and the 8192 minibatch setting reduces as we train for longer, as evidenced by Figure 16. Furthermore, the wall-clock time difference between 8192 minibatches and 1024 minibatches is substantial, since the former case does not come close to saturating the GPUs (see Figure 17a). Taken together, in Figure 17b, we see that the 1024 minibatch setting provides the best performance as a function of wallclock-time, and is not materially different to the final performance of the 8192 minibatch setting, even if the latter is slightly more sample efficient.

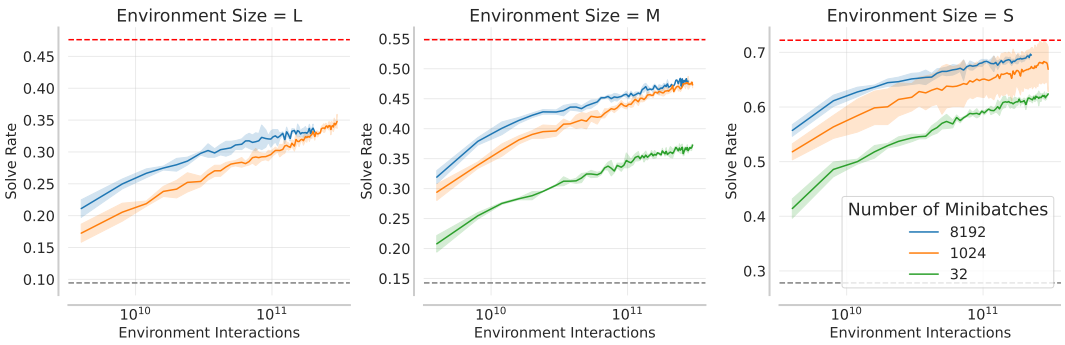
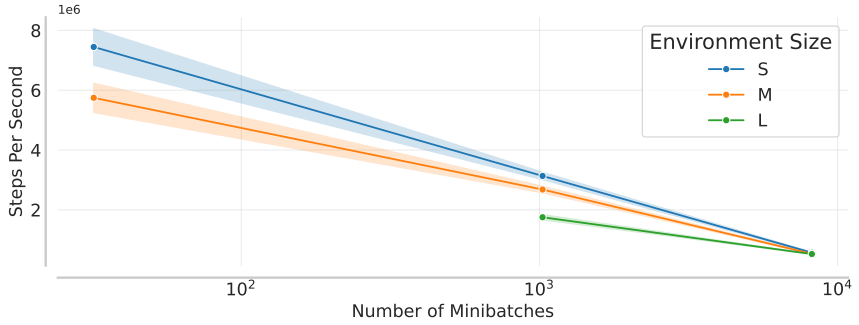
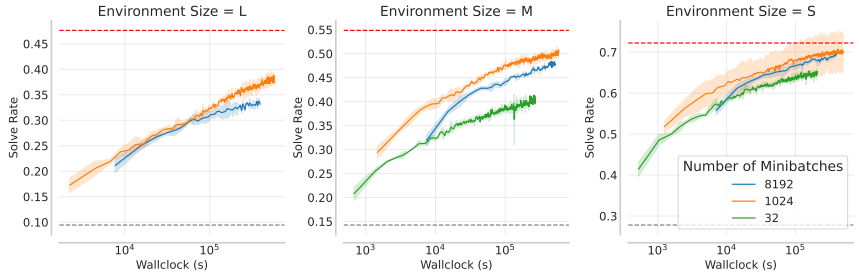


Figure 16: Comparing performance when using 1M parallel environments, but a different number of minibatches. Matthews et al. (2025) use 32 minibatches for the default setting of 2048 parallel environments, and using this for 1M parallel environments performs significantly worse, in line with the results from Figure 9. The `large` environment size runs out of memory when using only 32 minibatches.



(a) Steps per second vs. minibatches.



(b) Performance as a function of wall-clock time.

Figure 17: Comparing runtime when using different minibatch sizes and 1M parallel environments. Note that `L` runs out of memory with 32 minibatches.

E SFL Ablations

In this section we perform some ablations on the SFL results, to demonstrate which factors influence performance. For all cases, the ablations and "baseline" use 8192 environments, so that we could train for more timesteps than if we had used 2048.

E.1 Learning Rate

Figure 18 shows the effect of changing the learning rate. While reducing the learning rate by a factor of $5\times$ to $1e-5$ can avoid the early plateauing, it requires a prohibitively long wall-clock time to obtain a large amount of samples, whereas increasing parallelisation allows us to obtain the result in significantly less time, albeit at the cost of additional hardware.

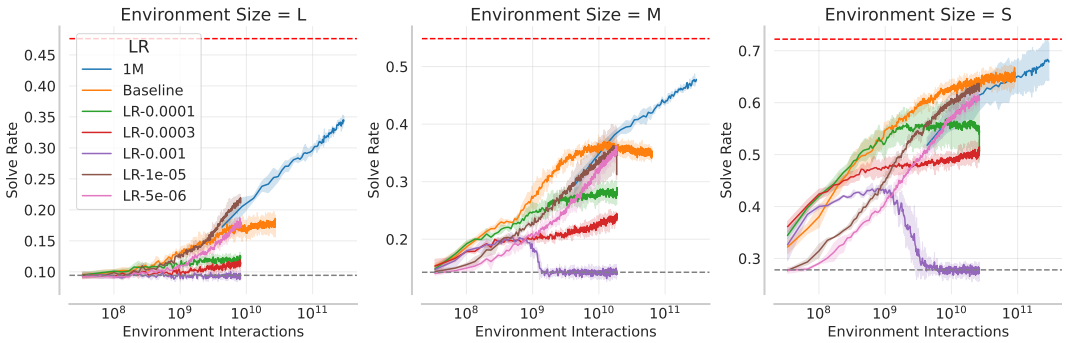


Figure 18: SFL results when keeping the number of environments fixed at 8192, but changing the learning rate. The baseline ($5e-5$) and 1M parallel environment run are shown for reference. Reducing the learning rate can avoid plateaus, but training is too slow to be able to process enough environment samples within a reasonable time.

E.2 PPO-EWMA

Next, in Figure 19, we show that using PPO-EWMA with a large center of mass can also improve performance over the baseline, and plateaus later, supporting the argument that the smaller outer step size due to increased parallelisation is beneficial.

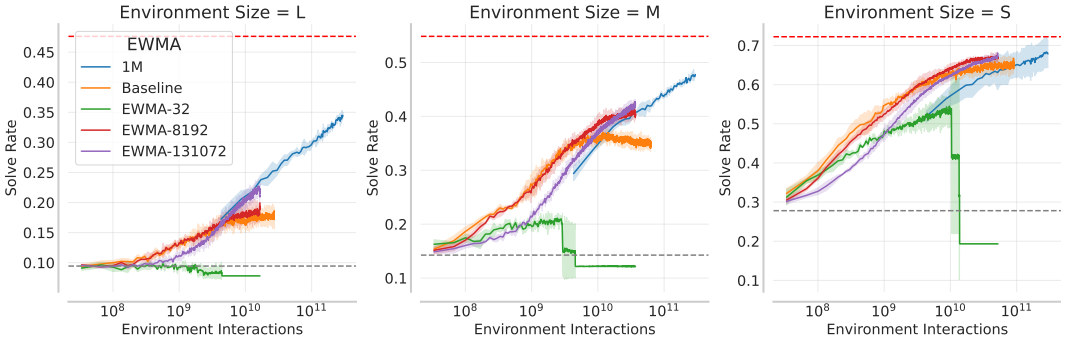


Figure 19: SFL results when keeping the number of environments fixed at 8192, but using PPO-EWMA. The baseline and 1M parallel environment run, both using normal PPO, are shown for reference. Increasing the center of mass, and thereby the regularisation, can alleviate plateaus.

E.3 Additional Filtering

Finally, the 1M parallel environment run performed additional filtering of environments to select the subset we actually train on. In particular, each GPU processed the same number of levels, meaning that the total number of pre-filtering levels we considered is larger than the baseline. We further increased the size of the high-learnability level buffer we sample from, since we have orders of magnitude more parallel environments. In Figure 20, we show that these changes alone are insufficient to improve performance, unless they are also coupled with increasing the parallelisation.

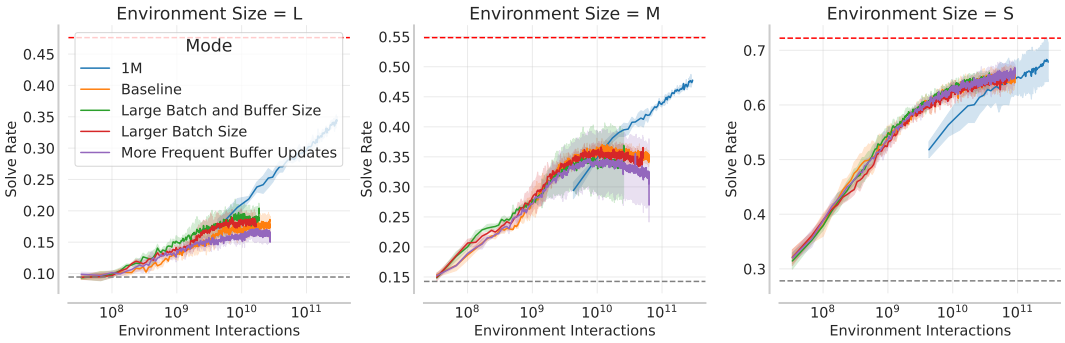


Figure 20: SFL results using 8192 environments, but either sampling more levels to filter through, or doing this and having a larger buffer of stored, high-learnability levels. This shows that while the 1M parallel environment run samples more levels, and has a larger buffer, these changes are insufficient to prevent the plateau that the 8192 environments agent succumbs to.

F SFL Hand Designed Results

Figure 21 contains the same agents as in Figure 11, but measured on the hand-designed evaluation set of levels (Matthews et al., 2025). The same overall trend is visible, in that more parallel environments lead to higher asymptotic performance.

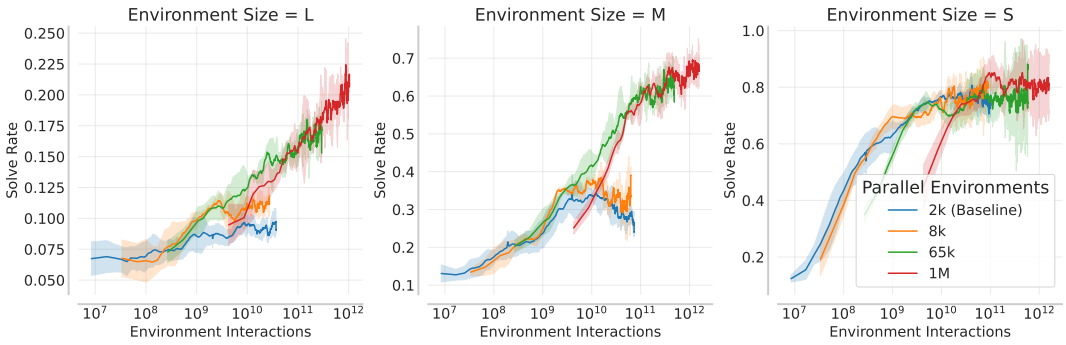


Figure 21: SFL Results on hand-designed environments. While the performance is much more noisy, the same rough trend holds, where additional parallelisation improves performance.

G Isaacgym Parallelisation Results

Figure 22 shows that increasing parallelisation leads to higher asymptotic performance in Isaacgym.

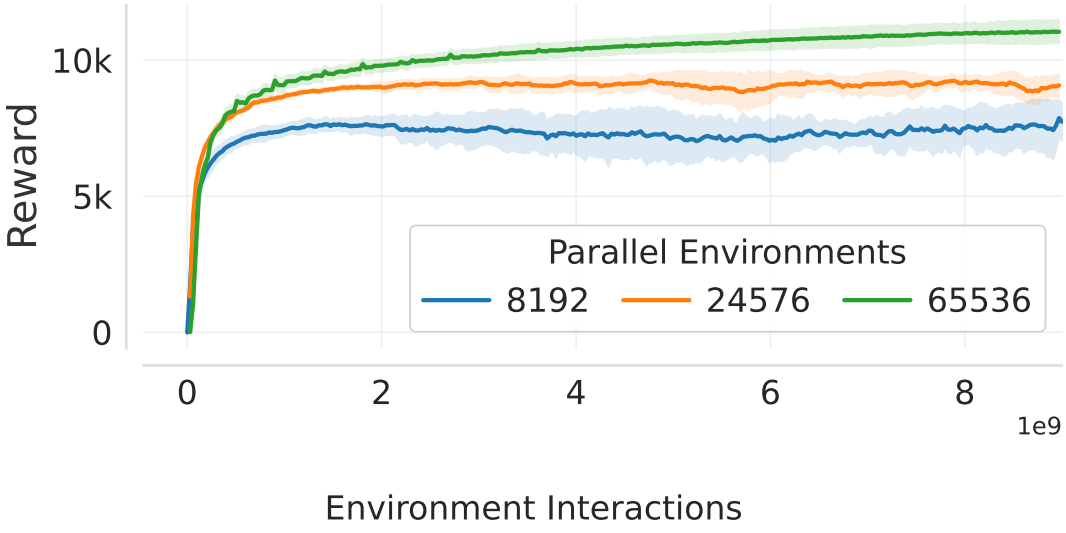


Figure 22: Scaling the number of parallel environments in the Isaacgym task Shadow Hand.