A Pseudocode for PPO-EWMA

We provide pseudocode PPO as well as PPO-EWMA to make it clear what changes need to be made:

Algorithm 1 PPO

```
for iteration = $1, 2, \ldots$ do
    for actor = $1, 2, \ldots, N$ do
        Run policy $\pi_{\theta_{\text{old}}}$ in environment
          for $T$ timesteps
        Compute advantage estimates
          $\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_T$
    end for
    for epoch = $1, 2, \ldots, E$ do
        for each minibatch do

            Optimize objective $L$
              with respect to $\theta$ on minibatch

        end for
    end for
    $\theta_{\text{old}} \leftarrow \theta$
end for
```

Algorithm 2 PPO-EWMA

```
for iteration = $1, 2, \ldots$ do
    for actor = $1, 2, \ldots, N$ do
        Run policy $\pi_{\theta_{\text{behav}}}$ in environment
          for $T$ timesteps
        Compute advantage estimates
          $\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_T$
    end for
    for epoch = $1, 2, \ldots, E$ do
        for each minibatch do
            Compute $\pi_{\theta_{\text{prox}}}$ for minibatch
            Optimize objective $L_{\text{decoupled}}$
              with respect to $\theta$ on minibatch
            $\theta_{\text{prox}} \leftarrow \text{EWMA}_{\beta_{\text{prox}}}(\theta)$
        end for
    end for
    $\theta_{\text{behav}} \leftarrow \theta$
end for
```

The expression $\text{EWMA}_{\beta_{\text{prox}}}(\theta)$ is shorthand for

$$\frac{\theta_t + \beta_{\text{prox}}\theta_{t-1} + \beta_{\text{prox}}^2\theta_{t-2} + \cdots + \beta_{\text{prox}}^t\theta_0}{1 + \beta_{\text{prox}} + \beta_{\text{prox}}^2 + \cdots + \beta_{\text{prox}}^t},$$

where $\theta_0, \theta_1, \ldots \theta_t$ are the values of θ after each gradient step. In practice, we compute this incrementally by initializing $\theta_{\text{prox}} \leftarrow \theta$ and $w \leftarrow 1$, and treating the update $\theta_{\text{prox}} \leftarrow \text{EWMA}_{\beta_{\text{prox}}}(\theta)$ as shorthand for

$$\begin{aligned} w_{\text{new}} &\leftarrow 1 + \beta_{\text{prox}} w \\ \theta_{\text{prox}} &\leftarrow \frac{1}{w_{\text{new}}}\theta + \beta_{\text{prox}}\frac{w}{w_{\text{new}}}\theta_{\text{prox}} \\ w &\leftarrow w_{\text{new}}. \end{aligned}$$

For PPG-EWMA, we make the same changes to the policy phase, while leaving the auxiliary phase unchanged. However, **the EWMA should be reinitialized at the start of each policy phase**, since θ changes a lot during the auxiliary phase.

Code for both PPO-EWMA and PPG-EWMA may be found at `https://github.com/openai/ppo-ewma`.

B Hyperparameters

All experiments were on Procgen's hard difficulty, without frame stack, using the convolutional neural network from IMPALA [Espeholt et al., 2018]. Unless stated otherwise, experiments lasted for 100 million environment steps.

Table 1: Default hyperparameters shared between PPO and PPG.

Hyperparameter	Value
Workers	4
Parallel environments per worker	64
Timesteps per rollout (T)	256
Minibatches per epoch	8
Adam step size (α)	5×10^{-4}
Value function coefficient	0.5
Entropy coefficient	0.01
PPO clipping parameter (ϵ)	0.2
GAE discount rate (γ)	0.999
GAE bootstrapping parameter (λ)	0.95
Reward normalization?	Yes
Advantage normalization?	Yes

Table 2: Default PPO-specific hyperparameter.

Hyperparameter	Value
Epochs (E)	3

Table 3: Default PPG-specific hyperparameters.

Hyperparameter	Value
Policy iterations per phase (N_π)	32
Policy phase policy epochs (E_π)	1
Policy phase value function epochs (E_V)	1
Auxiliary phase epochs (E_{aux})	6
Auxiliary phase minibatches per epoch	$16N_\pi$
Auxiliary phase cloning coefficient (β_{clone})	1

For the purpose of the artificial staleness experiments, we clipped $\pi_{\theta_{\text{behav}}}$ to keep the ratio $\frac{\pi_\theta}{\pi_{\theta_{\text{behav}}}}$ below 100, for numerical stability.

For PPG-EWMA, we chose the default EWMA decay rate β_{prox} such that the center of mass of the EWMA, $\frac{1}{1-\beta_{\text{prox}}} - 1$, equaled the number of minibatches per policy phase iteration (8), so that the maximum age of the proximal policy is the same in PPG and PPG-EWMA. We tuned this on the first 8 of the 16 Procgen environments by also trying $\frac{1}{1-\beta_{\text{prox}}} - 1 = 2$ and $\frac{1}{1-\beta_{\text{prox}}} - 1 = 32$, but did not find these to perform better. We did not re-tune β_{prox} on the last 8 Procgen environments or on PPO-EWMA.

For the batch size-invariance experiments, we made the following changes to the above defaults:

- We reduced the number of parallel environments, first by reducing the number of workers from 4 to 1, and then by reducing the number of parallel environments per worker from 64 to 16 to 4 to 1.

Table 4: Default PPO-EWMA and PPG-EWMA specific hyperparameter.

Hyperparameter	Value
Proximal policy EWMA decay rate (β_{prox})	0.889

- For the policy phase, we adjusted the Adam step size (α), the proximal policy EWMA decay rate (β_{prox}), advantage normalization, and the number of policy iterations per phase (N_π) in the way described in Section 4.
- For the auxiliary phase, we initially tried adjusting the Adam step size in the same way as for the policy phase. This worked well in terms of batch size-invariance, but resulted in prohibitively large wall-clock times at small batch sizes, due to the large number of auxiliary epochs. We therefore simply kept the auxiliary phase minibatch size per worker constant, and only adjusted the Adam step size when reducing the number of workers, not when reducing the number of parallel environments per worker.

C Adam square root step size adjustment

In Section 3, we stated that SGD and Adam have different learning rate adjustment rules. To compensate for the batch size being divided by some constant c, one must divide the SGD learning rate by c, but divide the Adam step size by $\sqrt{c}$ [Hardin, 2017].

The reason for the difference is that Adam divides the gradient by a running estimate of the root mean square gradient. If the gradient vector at the current step is g_t, then this denominator is approximately

$$\sqrt{\mathbb{E}\left[g_t^2\right]} = \sqrt{\mathbb{E}\left[g_t\right]^2 + \operatorname{Var}\left[g_t\right]} = \mathbb{E}\left[g_t\right]\sqrt{1 + \frac{\operatorname{Var}\left[g_t\right]}{E\left[g_t\right]^2}} = \mathbb{E}\left[g_t\right]\sqrt{1 + \frac{\mathcal{B}}{n}},$$

where all operations including the variance operator are applied componentwise, n is the batch size, and $\mathcal{B}$ is a componentwise version of the *gradient noise scale* defined by McCandlish et al. [2018], a measure of the noise-to-signal ratio of the gradient that approximates the critical batch size. Hence if the batch size is small compared to the critical batch size, then $\mathcal{B} \gg n$ for most components, and so the Adam denominator is approximately proportional to $\frac{1}{\sqrt{n}}$.

It follows that if the batch size is divided by some constant c, then the Adam denominator is multiplied by approximately $\sqrt{c}$ (providing the batch size is small compared to the critical batch size). Hence Adam is effectively dividing the learning rate by $\sqrt{c}$ automatically, and so the step size α only needs to be adjusted by an additional $\sqrt{c}$ to effectively divide the learning rate by c overall.

This all ignores Adam's ϵ hyperparameter, which is usually negligible, but is sometimes used to interpolate between Adam and momentum SGD.

To verify the square root rule for Adam, we conducted an ablation of our batch size-invariance experiments, in which we made the exact same adjustments, except that we divided the Adam step size by c instead of by $\sqrt{c}$. Our results are shown in Figure 5. When compared with Figure 2, this clearly shows that the square root rule is superior in our setting. Full results on each of the individual environments can be found in Appendix F.2.

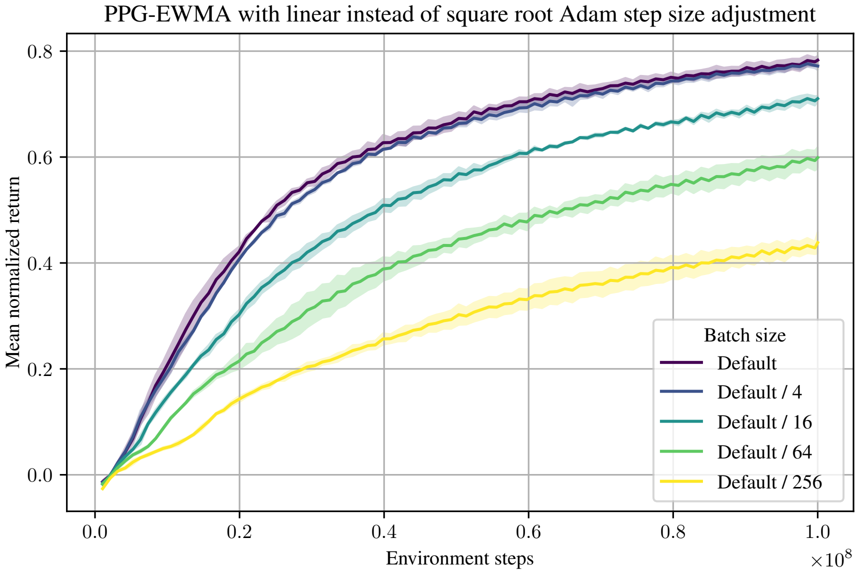

Figure 5: PPG-EWMA at different batch sizes, averaged over all 16 Procgen environments, with hyperparameters adjusted as in Figure 2, except with a linear rather than a square root adjustment to the Adam learning rate. Mean and standard deviation over 3 seeds shown.

Our results are in tension with those of Smith et al. [2017], who verified batch size-invariance for Adam using the linear rather than the square root rule. However, they achieved a lower degree of batch size-invariance with Adam than with SGD (see Figure 4 in that work), and moreover, our results show that the batch size needs to be reduced significantly before the difference between the two rules is noticeable. We believe that this accounts for their experimental results, and that the square root rule is superior in general (with the exception of when Adam's ϵ hyperparameter is high enough for it to behave like momentum SGD).

D Adam β_1 and β_2 adjustments

As discussed in Section 3, there is an additional adjustment one should make when using Adam, other than to the step size α. To compensate for the batch size being divided by some constant c, one should also raise the exponential decay rates β_1 and β_2 to the power of $1/c$ [Hardin, 2017].

We omitted this adjustment in most of our experiments, and were still able to achieve a high degree of batch size invariance. For all except one environment, the difference in normalized return between the largest and smallest batch sizes at the end of training was at most 0.11 (see Figure 3). For these environments, it would probably have required many additional experiments to detect any further improvement that adjusting β_1 and β_2 might provide. However, for the Heist environment, this difference was 0.55. We hypothesized that this might be explained by the fact that we did not adjust β_1 and β_2.

We therefore conducted a version of our batch size-invariance experiments in which we either adjusted only β_2 using the above rule, or adjusted both β_1 and β_2. Our results are shown in Figure 6. In both cases there was still a large difference in performance at the largest and smallest batch sizes.

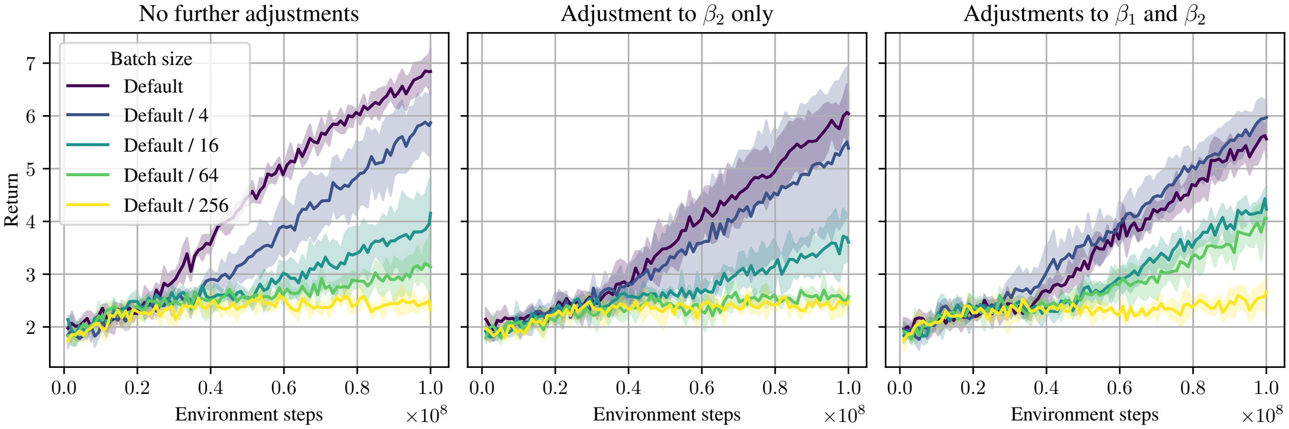

Figure 6: PPG-EWMA at different batch sizes on Heist, with hyperparameters adjusted as in Figure 2, together with further adjustments to Adam's β_1 and β_2 hyperparameters as indicated. Mean and standard deviation over 3 seeds shown.

E Hypothesis tests for ablations

As discussed in Section 3, we conducted hypothesis tests to check the statistical significance of the effects produced by the ablations to our batch size-invariance experiments.

Our primary metric for measuring batch size-invariance was the difference in final performance of the algorithm at different batch sizes. To get a complete picture of how important our ablations were at different batch sizes, we compared our default (largest) batch size with each of the other batch sizes, and tested the hypothesis that the difference was larger for the ablation. This resulted in 16 hypotheses, corresponding to the 4 ablations and the 4 non-default batch sizes. To test each hypothesis, we used a van Elteren test [van Elteren, 1960], a stratified version of the Mann–Whitney U-test, treating the different environments as strata. This gives a non-parametric Z-test of the null hypothesis that for each of the environments, the probability of the ablation outperforming the original experiment is the same as the probability of the original experiment outperforming the ablation [LaVange and Koch, 2006]. To reduce noise (and thereby increase statistical power), we measured average performance over the last 4 million timesteps (the length of a single PPG phase). We used a significance level of 0.1% and applied a Bonferroni correction to account for multiple comparisons.

Our results are shown in Table 5. For ablation (b), the difference is only significant at the smallest batch size. For all other ablations, the difference is significant at every batch size, execpt for the largest batch size for ablation (d).

Table 5: Effect sizes (and Z-scores, in parentheses) for each of our hypotheses. The effect size is a difference of differences in final normalized return, between the ablation and the original experiment and between the different batch sizes. In bold are the effect sizes found to be signifcant at the 0.1% level after applying a Bonferroni correction (i.e., with a one-tailed p-value below $^{0.001}/_{16}$, or equivalently, a Z-score above 3.84).

Batch sizes: Default vs …	(a) No Adam step size adjustment		(b) No adv. norm. adjustment		(c) No EWMA adjustment		(d) No EWMA, just PPG	
Default / 4	$\mathbf{0.046}$	(5.02)	-0.014	(-1.53)	$\mathbf{0.020}$	(4.58)	0.019	(3.27)
Default / 16	$\mathbf{0.347}$	(7.31)	-0.033	(-1.53)	$\mathbf{0.021}$	(4.47)	$\mathbf{0.016}$	(4.04)
Default / 64	$\mathbf{0.621}$	(6.98)	0.054	(-0.11)	$\mathbf{0.027}$	(5.02)	$\mathbf{0.042}$	(5.13)
Default / 256	$\mathbf{0.709}$	(6.87)	$\mathbf{0.224}$	(4.47)	$\mathbf{0.041}$	(4.80)	$\mathbf{0.030}$	(4.58)

F Results on individual environments

F.1 Artificial staleness

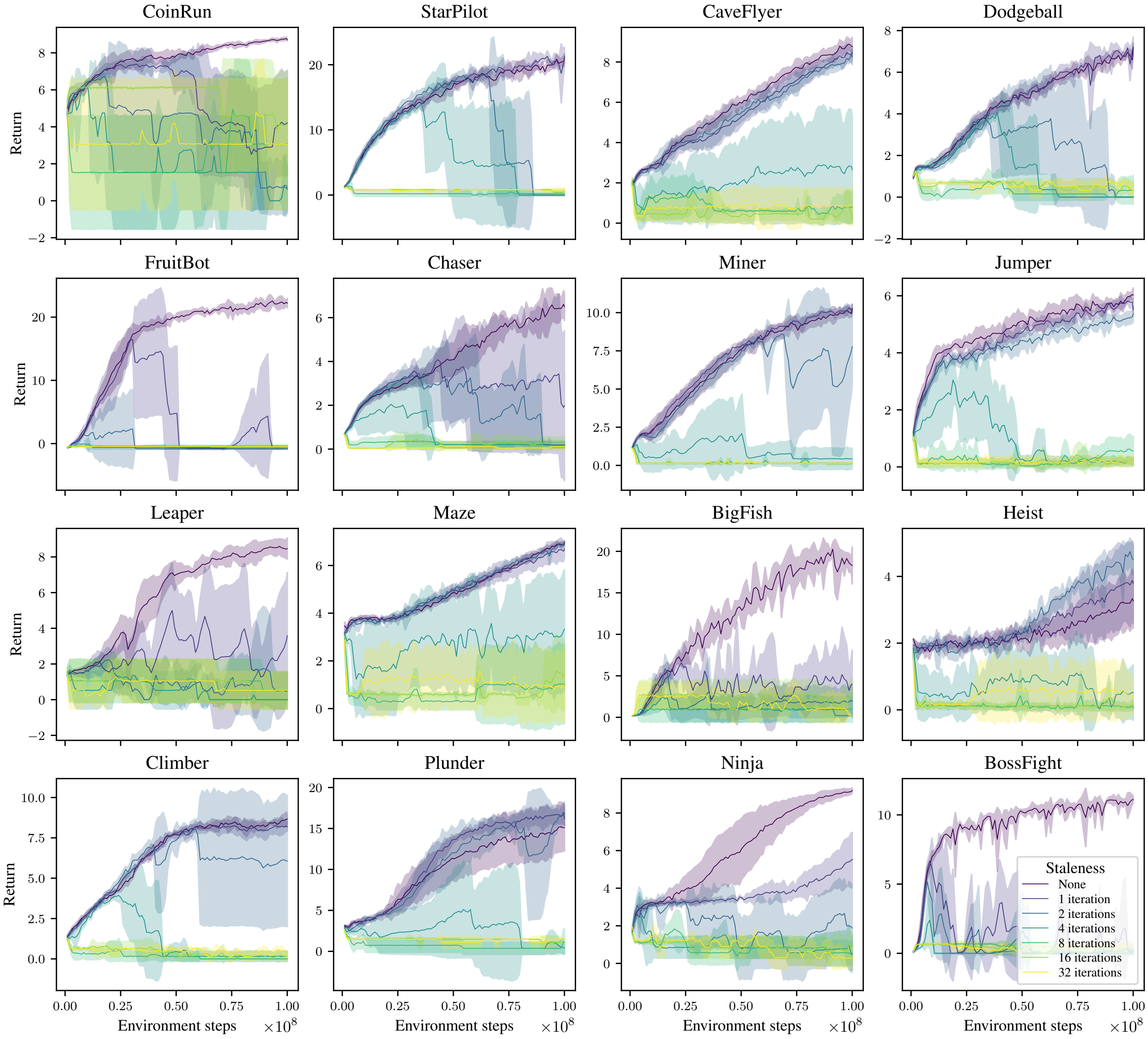

Figure 7: Results from Figure 1(a) (PPO with $\pi_{\theta_{\text{old}}} = \pi_{\theta_{\text{recent}}}$) split across the individual environments. Mean and standard deviation over 4 seeds shown.

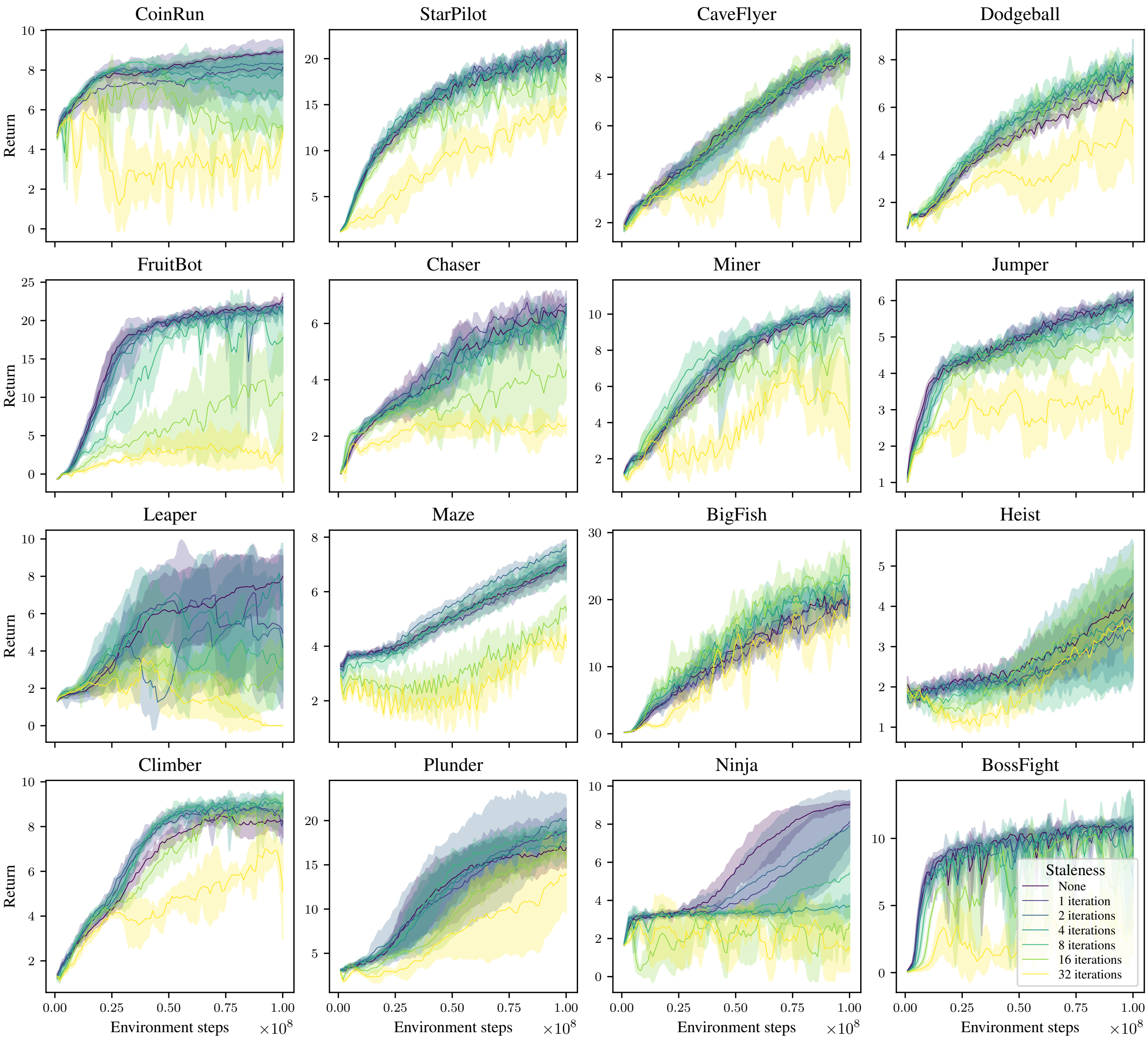

Figure 8: Results from Figure 1(b) (PPO with decoupled objective) split across the individual environments. Mean and standard deviation over 4 seeds shown.

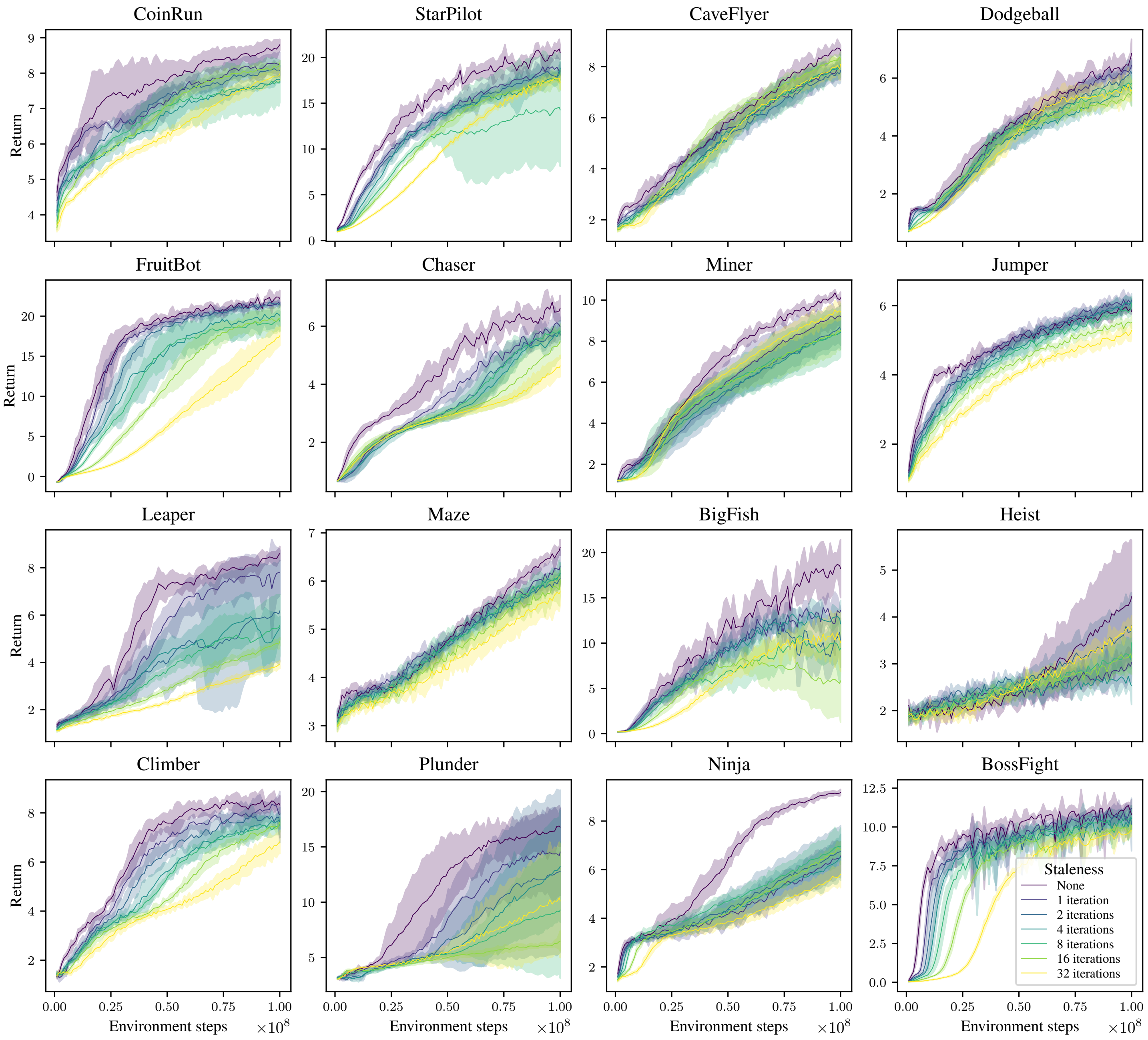

Figure 9: Results from Figure 1(c) (PPO with $\pi_{\theta_{\text{old}}} = \pi_{\theta_{\text{behav}}}$) split across the individual environments. Mean and standard deviation over 4 seeds shown.

F.2 Batch size-invariance

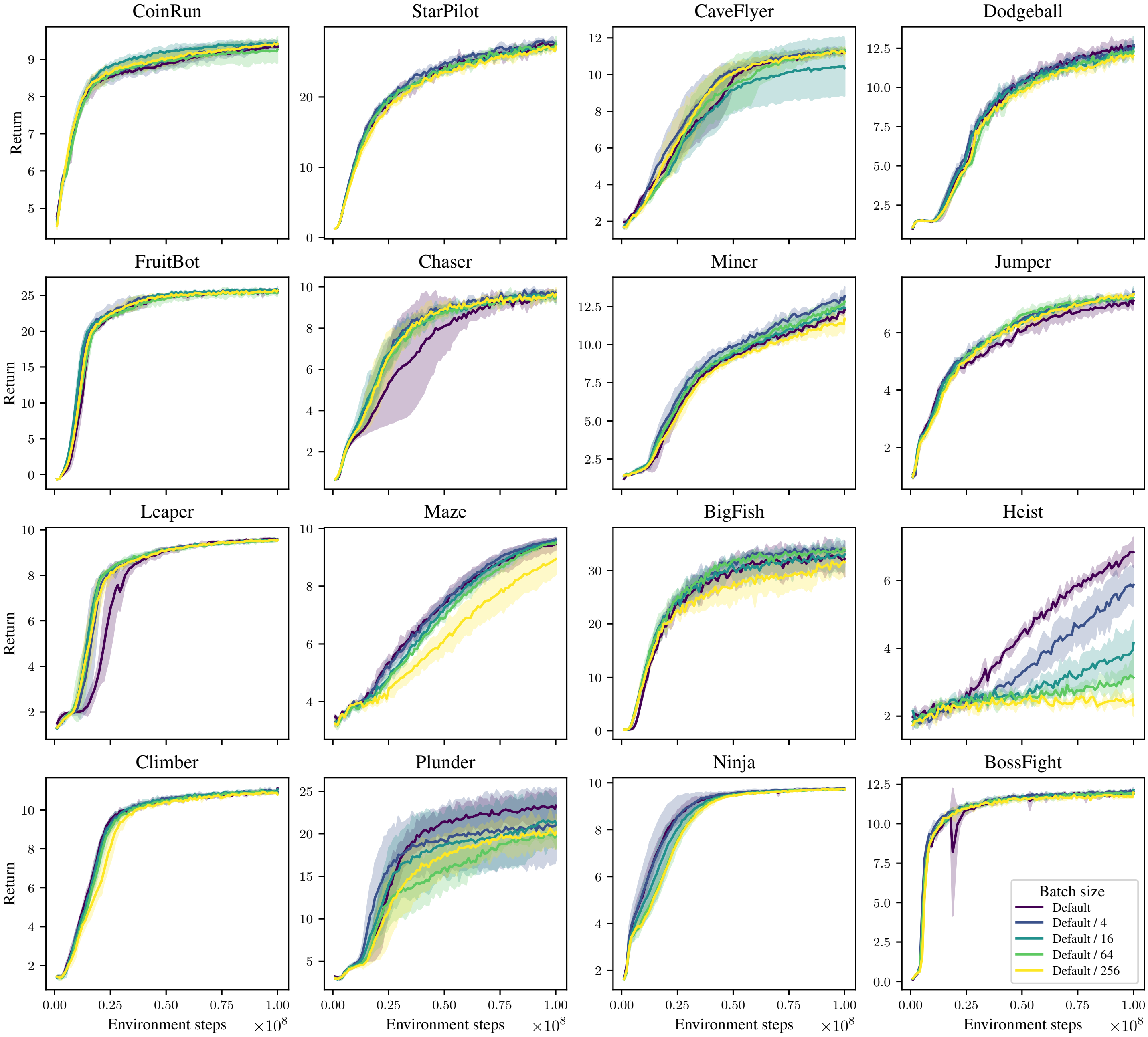

Figure 10: Results from Figure 2 (PPG-EWMA with all batch size-invariance adjustments) split across the individual environments. Mean and standard deviation over 3 seeds shown.

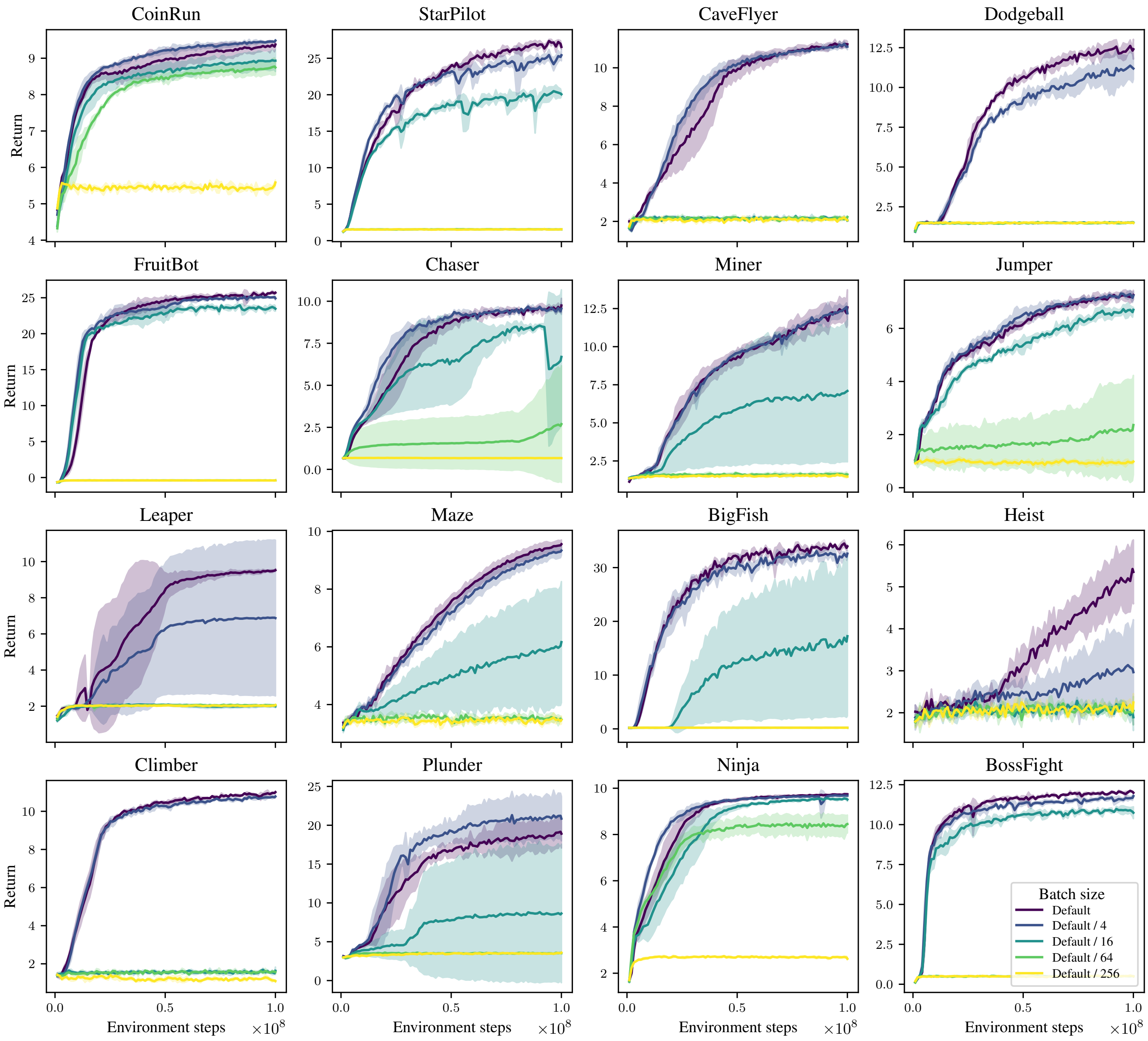

Figure 11: Results from Figure 2(a) (no Adam step size adjustment) split across the individual environments. Mean and standard deviation over 3 seeds shown.

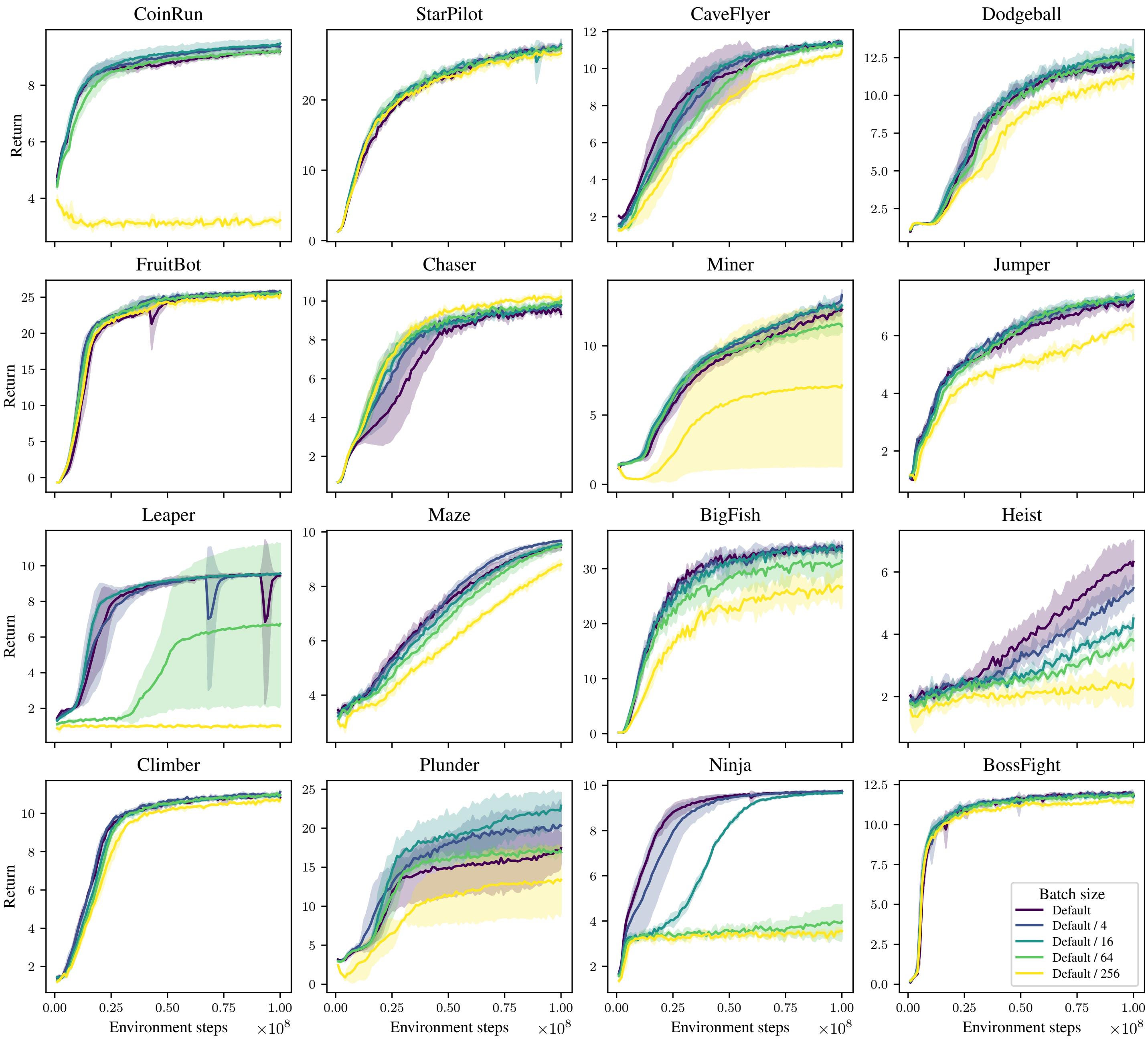

Figure 12: Results from Figure 2(b) (no advantage normalization adjustment) split across the individual environments. Mean and standard deviation over 3 seeds shown.

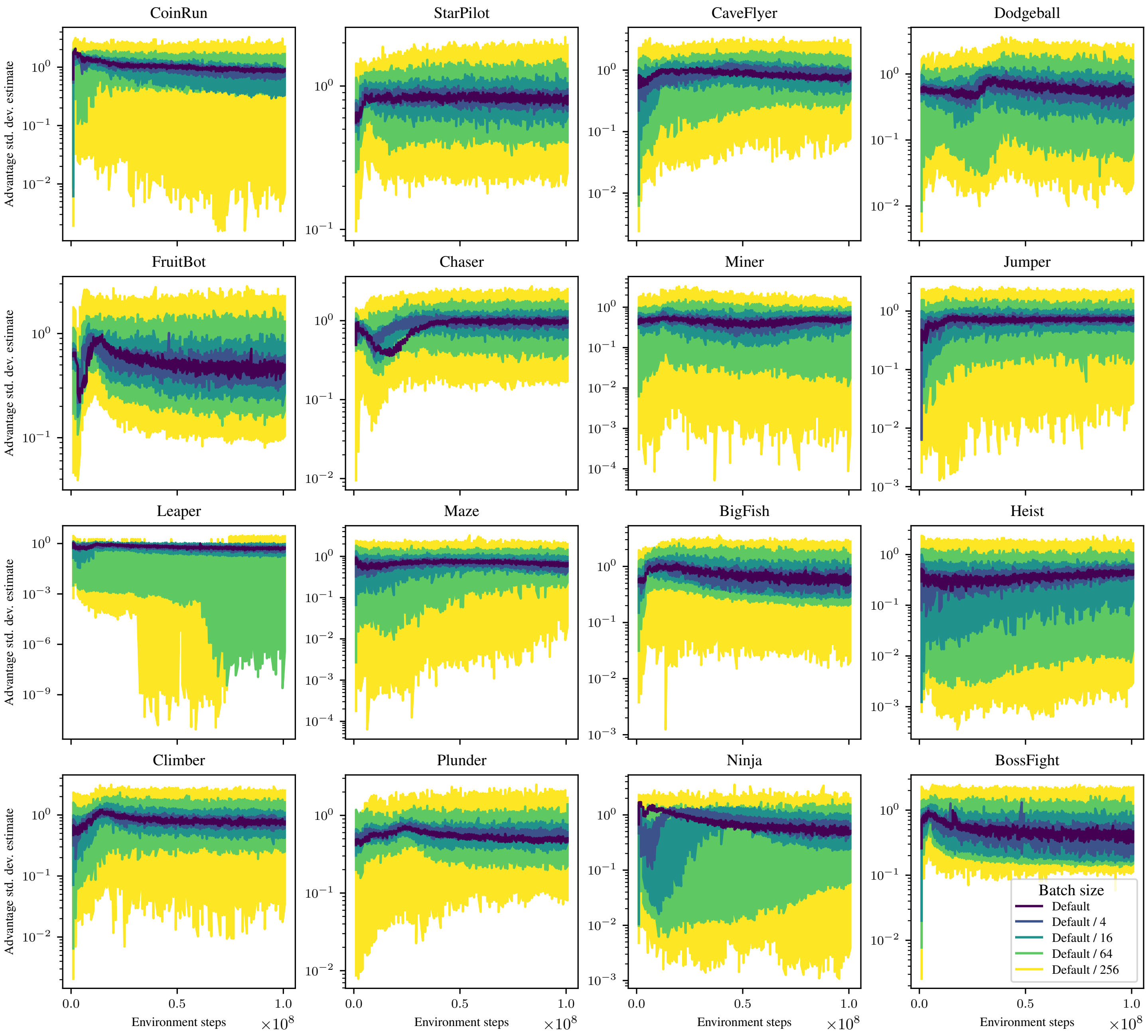

Figure 13: Advantage standard deviation estimates for the results from the previous figure. We plot estimates from the first seed only and perform no smoothing, since we are interested in the amount of oscillation. Note that performance degrades with no advantage normalization adjustment only once the estimates oscillate by factor of around 10 or more.

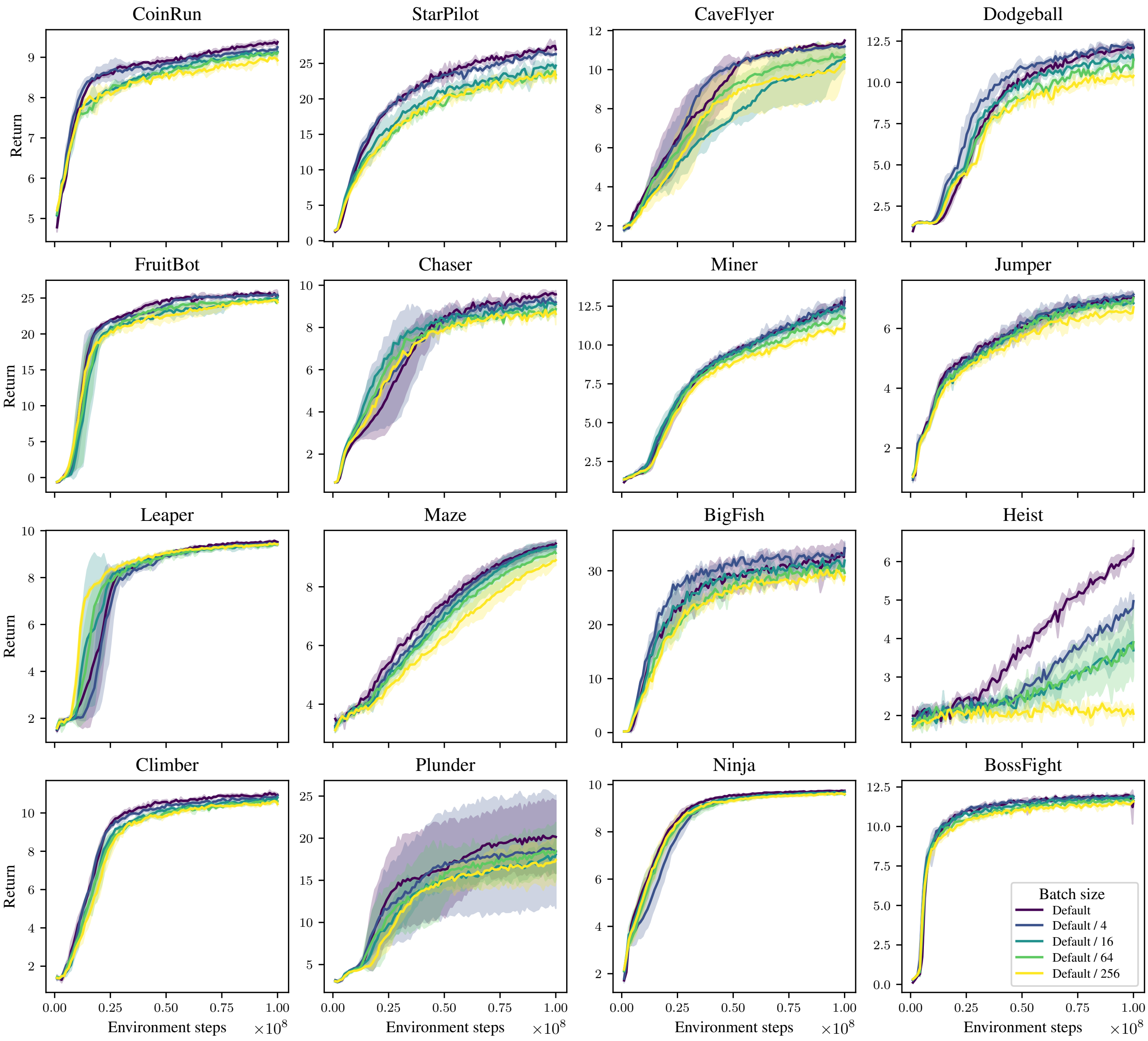

Figure 14: Results from Figure 2(c) (no EWMA adjustment) split across the individual environments. Mean and standard deviation over 3 seeds shown.

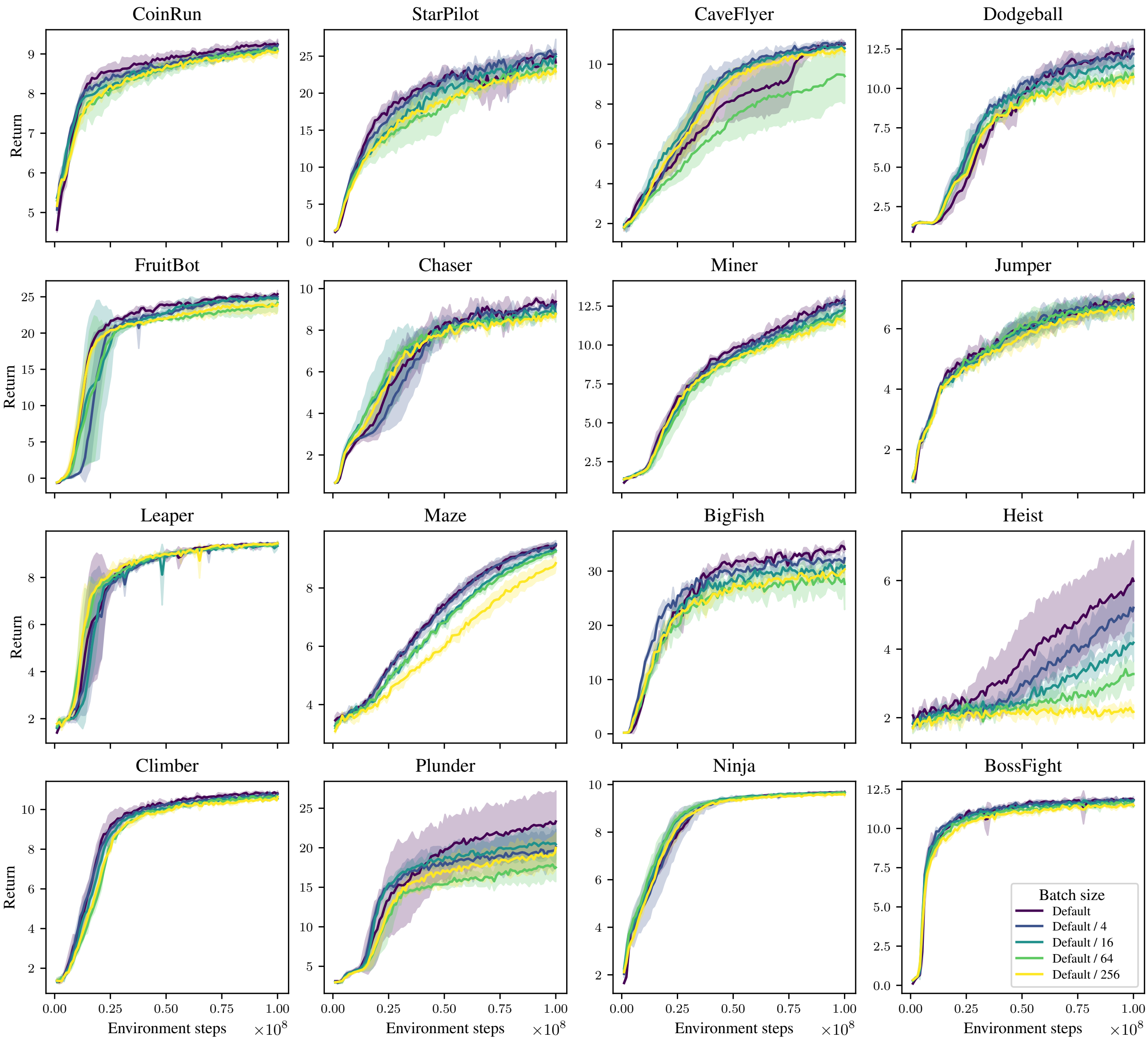

Figure 15: Results from Figure 2(d) (no EWMA at all, just PPG) split across the individual environments. Mean and standard deviation over 3 seeds shown.

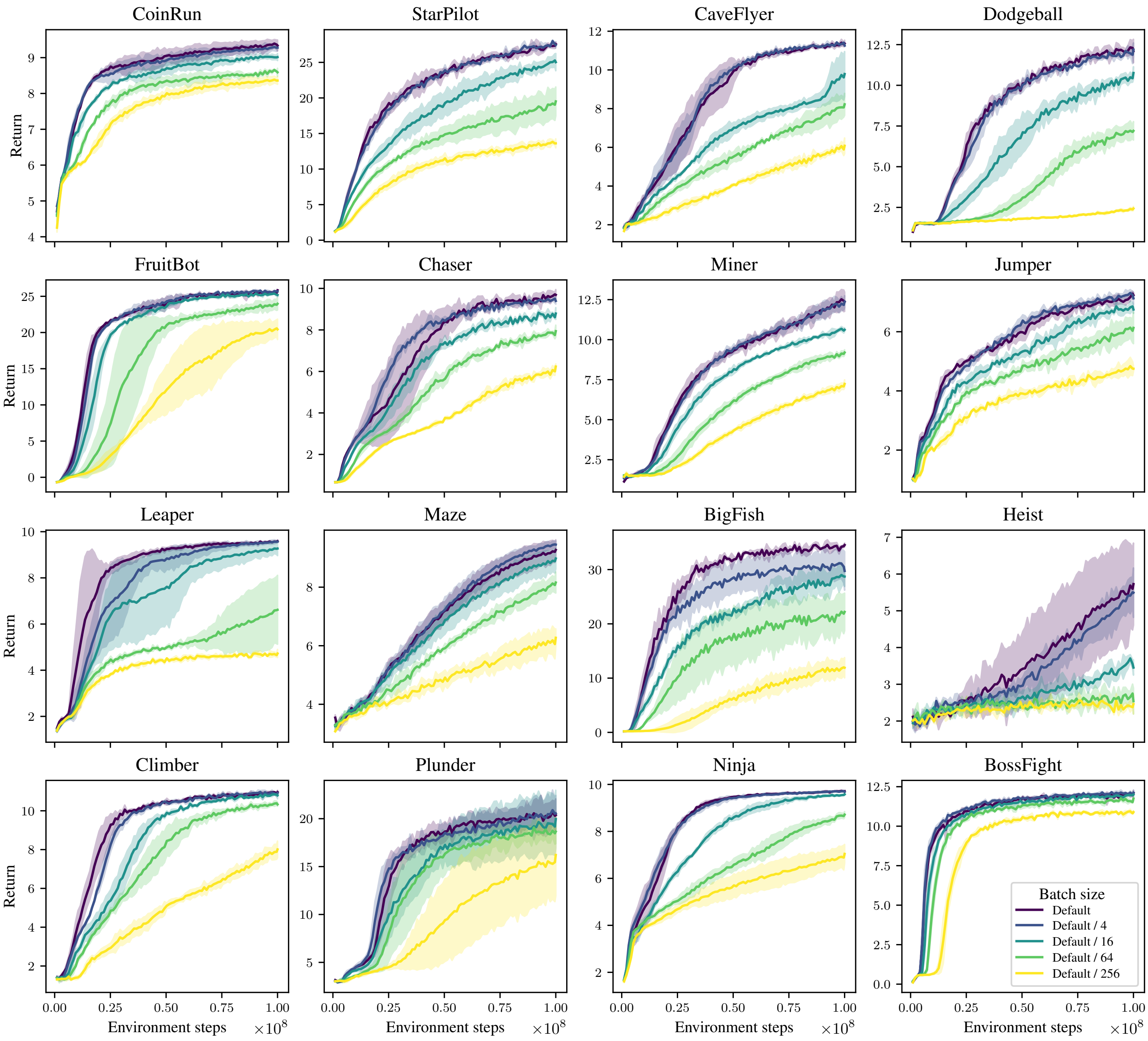

Figure 16: Results from Figure 5 (PPG-EWMA with linear instead of square root Adam step size adjustment) split across the individual environments. Mean and standard deviation over 3 seeds shown.

F.3 EWMA comparison

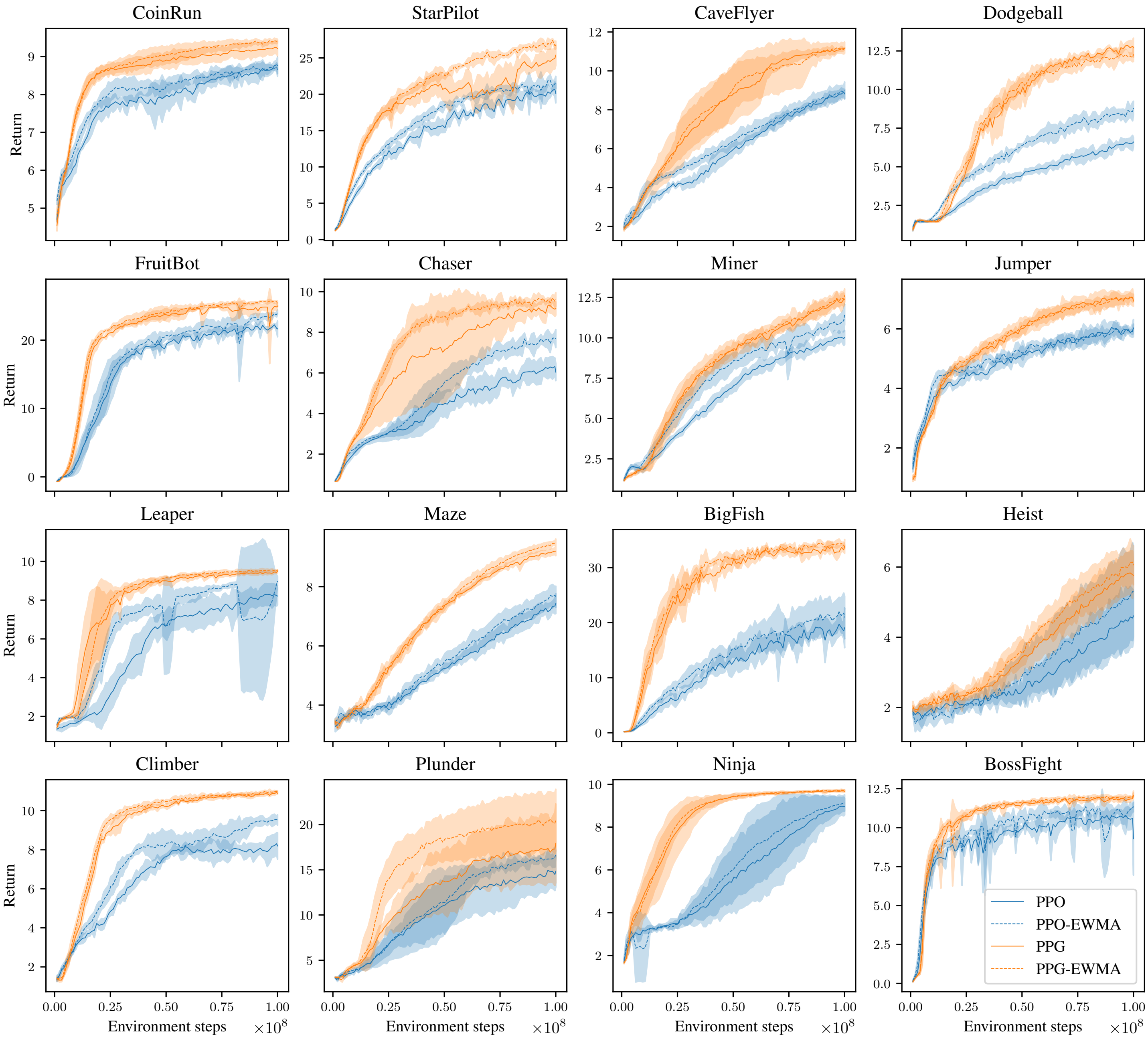

Figure 17: Results from Figure 4 (performance of all 4 algorithms) split across the individual environments. Mean and standard deviation over 4 seeds shown.

G Role of the proximal policy EWMA decay rate

In this section we discuss the role of the hyperparameter β_{prox} in PPO-EWMA and PPG-EWMA in more depth.

Recall that in PPO-EWMA and PPG-EWMA, the proximal policy network parameter vector θ_{prox} is an exponentially-weighted moving average (EWMA) of the policy network parameter vector θ, meaning that

$$\theta_{\text{prox}} = \frac{\theta_t + \beta_{\text{prox}}\theta_{t-1} + \beta_{\text{prox}}^2\theta_{t-2} + \cdots + \beta_{\text{prox}}^t\theta_0}{1 + \beta_{\text{prox}} + \beta_{\text{prox}}^2 + \cdots + \beta_{\text{prox}}^t},$$

where $\theta_0, \theta_1, \ldots \theta_t$ are the values of θ after each gradient step.

Rather than working with the decay rate β_{prox} of this EWMA directly, it is conceptually clearer to work with the center of mass of this EWMA,

$$\mathrm{COM}_{\text{prox}} := \lim_{t\to\infty} \frac{0 + \beta_{\text{prox}}1 + \beta_{\text{prox}}^2 2 + \cdots + \beta_{\text{prox}}^t t}{1 + \beta_{\text{prox}} + \beta_{\text{prox}}^2 + \cdots + \beta_{\text{prox}}^t} = \frac{1}{1-\beta_{\text{prox}}} - 1.$$

This is average age of a term in the EWMA in the limit as $t \to \infty$, and so if θ were to follow a straight line path for example, then $\theta - \theta_{\text{prox}}$ would be approximately proportional to $\mathrm{COM}_{\text{prox}}$.

Suppose then that we halve $\mathrm{COM}_{\text{prox}}$. What is the effect of this?

Consider the gradient of the KL divergence from the current policy to the proximal policy, as a function of the proximal policy parameter vector,

$$\text{Grad-KL}\left(\theta_{\text{prox}}\right) := \nabla_\theta \, \mathrm{KL}\left[\pi_{\theta_{\text{prox}}}\left(\cdot \mid s_t\right), \pi_\theta\left(\cdot \mid s_t\right)\right].$$

Since KL divergence is always greater than or equal to 0, with equality if and only if the input distributions are equal, $\text{Grad-KL}\left(\theta\right) = 0$, and hence, to first-order, $\text{Grad-KL}\left(\theta_{\text{prox}}\right)$ is a linear function of $\theta - \theta_{\text{prox}}$. Therefore halving $\mathrm{COM}_{\text{prox}}$ should have a similar effect to halving the KL penalty coefficient β. In other words, we should be able to compensate for halving $\mathrm{COM}_{\text{prox}}$ by doubling the KL penalty coefficient.

Intuitively, the KL penalty acts like a rubber band pulling the policy towards the proximal policy. Halving $\mathrm{COM}_{\text{prox}}$ is analogous to attaching the rubber band to a point half as far away, while doubling the KL penalty coefficient is analogous to doubling the thickness of the rubber band. Doing both simultaneously results in the same overall force.

We tested this hypothesis using a hyperparameter grid search over the EWMA center of mass and the KL penalty coefficient, for PPG-EWMA on StarPilot. We used our smallest batch size, along with our corresponding batch size-invariance adjustments, to allow the greatest scope for reducing C_{prox} without making the EWMA degenerate into averaging over a single data point.

Our results are shown in Figure 18. The diagonal banding clearly demonstrates the expected effect. However, the effect only holds locally: as $\mathrm{COM}_{\text{prox}}$ is continually halved and the KL penalty coefficient is continually doubled, performance gradually degrades.

We believe that this is because reducing $\mathrm{COM}_{\text{prox}}$ has a second-order effect, which is to increase the variance of $\theta - \theta_{\text{prox}}$. This is both because the EWMA is averaging over a smaller effective sample size, and because $\theta_t - \theta_{t-k}$ has a lower signal-to-noise ratio as k decreases. Therefore if $\mathrm{COM}_{\text{prox}}$ has been halved too many times, we should expect to no longer be able to fully compensate for this by continuing to double the KL penalty coefficient.

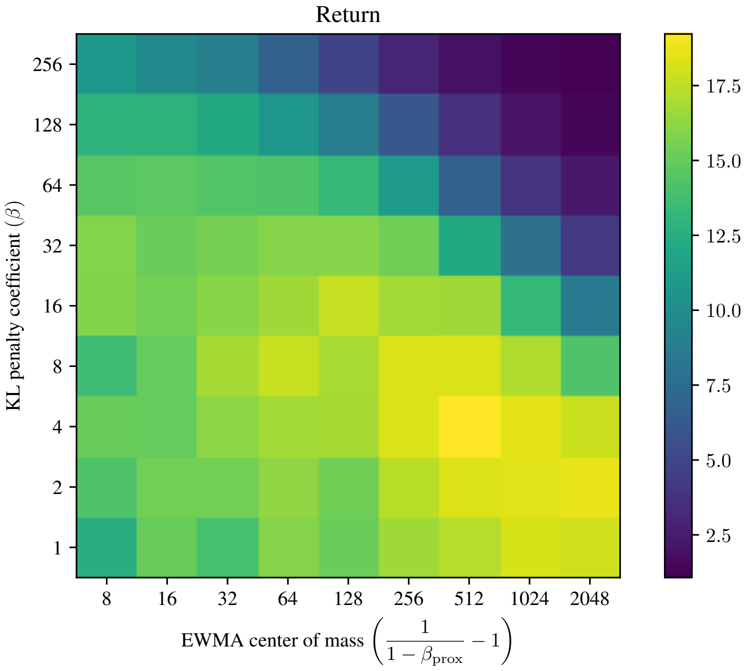

Figure 18: Performance on PPG-EWMA on StarPilot after 20 million environment timesteps, using a single parallel copy of the environment along with our batch size-invariance adjustments. The default hyperparameter settings correspond to square in the bottom right corner. Mean over 2 seeds shown.