Batch Size-invariance for Policy Optimization,

J. Hilton et al, 2022

옥찬호

utilForever@gmail.com

Introduction

- The stability and reliability of policy gradient-based methods is typically improved by controlling the size of policy updates, using
  - "Trust Region" (TRPO) [Schulman et al, 2015]
  - "Surrogate Objective" (PPO) [Schulman et al, 2017]
- We cannot trust updates that take us too far from the policy used to collect experience (**Behavior Policy**)
  - The behavior policy is irrelevant to the justification.
  - What matter is that we control **how fast the policy is updated**, or put another way, that we approximate the natural policy gradient.

Introduction

- Our key insight is that the "old" policy serves two independent purposes.
  1. **For off-policy correction**, via importance sampling,
     for which the old policy must be **the behavior policy**.
  2. To control the size of policy updates, for which the old policy can be any recent policy.
     (**Proximal Policy**)
- It does not matter whether the proximal policy is also the behavior policy;
  It only matters **how old the proximal policy is**.

Introduction

- Our insight allows us to make PPO batch size-invariant, meaning that when the batch size is changed, we can preserve behavior, as a function of the number of examples processed, by changing other hyperparameters. (**Exponentially-Weighted Moving Average; EWMA**)
  - It's of practical benefit when we wish to increase batch size to reduce gradient variance, but computational resources such as GPU memory do not allow this.

Decoupled Policy Objectives

- KL penalized objective in PPO [Schulman et al, 2017]

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t - \beta KL\left[\pi_{\theta_{\text{old}}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)\right]\right]$$

  - where $\hat{A}_t$ is an estimator of the advantage at timestep t.
  - where $\hat{\mathbb{E}}_t[\dots]$ indicates the empirical average over a finite batch of timesteps t.
  - The first use of $\pi_{\theta_{\text{old}}}$ is as part of importance sampling ratio.
    In order for the policy gradient estimate to be unbiased, this policy needs to be the one that was used for sampling, so we call this the **Behavior Policy** $\boldsymbol{\pi_{\theta_{\text{behav}}}}$.
  - The second use of $\pi_{\theta_{\text{old}}}$ is as a recent target to pull the current policy towards,
    so we call this the **Proximal Policy** $\boldsymbol{\pi_{\theta_{\text{prox}}}}$.

Decoupled Policy Objectives

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- Our key insight is that <u>**the proximal policy need not equal the behavior policy**</u>. It matters **how old** the proximal policy is, but it does not matter whether or not the proximal policy was used for sampling.
- We therefore define **the decoupled KL penalized objective**

$$L^{KLPEN}_{\mathrm{decoupled}}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{behav}}}(a_t|s_t)}\hat{A}_t - \beta KL\left[\pi_{\theta_{\mathrm{prox}}}(\cdot\,|s_t), \pi_\theta(\cdot\,|s_t)\right]\right]$$

  - where $\pi_{\theta_{\mathrm{behav}}}$ is the policy used for sampling.
  - where $\pi_{\theta_{\mathrm{prox}}}$ is a recent policy yet to be specified.

Decoupled Policy Objectives

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- The clipped PPO objective [Schulman et al, 2017]

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\,\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

  - where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$

- However, we can rewrite this objective as

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\frac{1}{\pi_{\theta_{\mathrm{old}}}}\min(\pi_\theta\,\hat{A}_t, \mathrm{clip}(\pi_\theta, (1-\epsilon)\pi_{\theta_{\mathrm{old}}}, (1+\epsilon)\pi_{\theta_{\mathrm{old}}})\hat{A}_t)\right]$$

- Now the first use of $\pi_{\theta_{\mathrm{old}}}$ is as part of **an importance sampling ratio**, for which we must use the behavior policy, and the second and third uses are in **applying the implicit KL penalty**, for which we can use the proximal policy.

Decoupled Policy Objectives

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- Therefore, define the **decoupled clipped objective**

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

  - where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{prox}}}(a_t|s_t)}$

Batch size-invariance

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- When the batch size is changed, the original behavior can be approximately recovered by adjusting other hyperparameters to compensate.
- We consider behavior as a function of the total number of examples processed, so another way to put this is that doubling the batch size halves the number of steps needed.
- We treat batch size-invariance as a descriptive property that can hold to some degree, rather than as a binary property.
  In practice, the original behavior can never be recovered perfectly, and the extent to which it can be recovered depends on both how much and the direction in which the batch size is changed.

Batch size-invariance

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- Batch size-invariance for SGD
  - Consider running SGD on a loss function $L(\theta; x)$ of a parameter vector θ and a data point x. Two steps with batch size n and learning rate α corresponds to the update rule

$$\theta_{t+2} = \theta_t - \frac{\alpha}{n} \sum_{x \in B_t} \nabla_\theta L(\theta_t; x) - \frac{\alpha}{n} \sum_{x \in B_{t+1}} \nabla_\theta L(\theta_{t+1}; x)$$

  where B_t and B_{t+1} are the next two batches of size n. On the other hand, a single step with batch size $2n$ and learning rate 2α corresponds to the update rule

$$\theta_{t+2} = \theta_t - \frac{2\alpha}{2n} \sum_{x \in B_t \cup B_{t+1}} \nabla_\theta L(\theta_t; x)$$

  - These update rules are very similar, the only difference being whether the gradient for B_{t+1} is evaluated at θ_t or θ_{t+1}. If the batch size is small compared to the critical batch size, then the difference between θ_t and θ_{t+1} is mostly noise, and moreover this noise is small compared to the total noise accumulated by θ_t over previous updates.

Batch size-invariance

- Batch size-invariance for Adam
  - To compensate for the batch size being divided by some constant c, one must make the following adjustments [Hardin, 2017]:
    - Divide the step size α by $\sqrt{c}$.
    - (Raise the exponential decay rates β_1 and β_2 to the power of $^1/_c$.
  - The first adjustment should be contrasted with the linear learning rate adjustment for vanilla SGD.
  - The second adjustment is much less important in practice, since Adam is fairly robust to the β_1 and β_2 hyperparameters (hence it has been parenthesized). Note also that β_1 also affects the relationship between the current policy and the proximal policy in policy optimization.

Batch size-invariance

- Batch size-invariance for Policy Optimization
  - In policy optimization algorithms like PPO, there are two different batch sizes: the number of environment steps in each gradient step, which we call the **optimization batch size**, and the number of environment steps in each alternation between sampling and optimization, which we call the **iteration batch size**. When we say that such an algorithm is batch size-invariant, we mean that changes to both batch sizes **by the same factor simultaneously** can be compensated for. The motivation for this definition is that this is the effect of changing the degree of data-parallelism.
  - If the optimization algorithm (such as SGD or Adam) used by PPO is batch size-invariant, then by definition this makes PPO optimization batch size-invariant. In the next section, we show how to make PPO iteration batch size-invariant, and therefore batch size-invariant outright.

PPO/PPG-EWMA

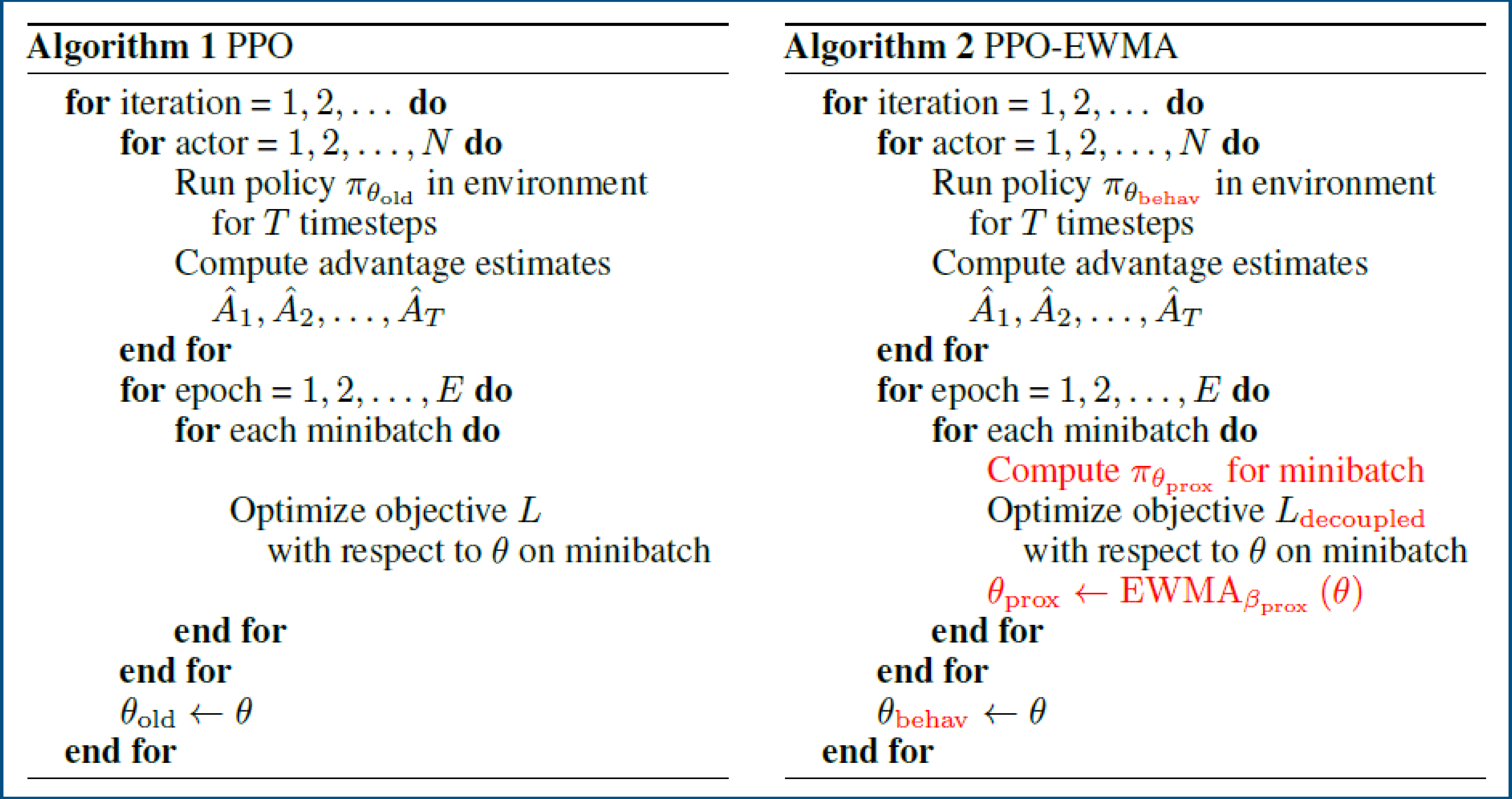
Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- A simple modification that can be made to any PPO-based algorithm
  - Maintain an exponentially-weighted moving average (EWMA) of the policy network, updating it after every policy gradient step using some decay rate β_{prox}.
  - Use this as the network for the proximal policy in one of the decoupled policy objectives.
- The motivation of an EWMA
  - We would like to be able to use a policy from some fixed number of steps ago as the proximal policy, but this requires storing a copy of the network from every intermediate step, which is prohibitive if this number of steps is large.
  - Using an EWMA allows us to approximate this policy using more reasonable memory requirements. Although this approximation is not exact, averaging in parameter space may actually improve the proximal policy, and the age of the proximal policy can still be controlled by adjusting β_{prox}.

PPO/PPG-EWMA

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- Pseudocode for PPO and PPO-EWMA



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
```

PPO/PPG-EWMA

- Note that the main effect of changing the iteration batch size in PPO is to change the age of the behavior and proximal policies (which are coupled).
  - The age of the behavior policy affects how on-policy the data is,
    but this does not matter much, as long as it is not too large.
  - However, the age of the proximal policy affects the strength of the KL penalty
    (or the implicit KL penalty in the case of the clipped objective),
    which influences how fast the policy can change.
  - We therefore need to maintain the age of the proximal policy
    as the iteration batch size is changed, which is what our modification enables.

PPO/PPG-EWMA

- More specifically, to achieve batch size-invariance for PPO- and PPG-EWMA, we make the following adjustments to compensate for the optimization and iteration batch sizes being divided by some constant c:
  - Adjust the optimization hyperparameters as described in the previous section, i.e., divide the vanilla SGD learning rate by c or the Adam step size by $\sqrt{c}$. (We use Adam.)
  - Multiply $\frac{1}{1-\beta_{\text{prox}}} - 1$ by c. (This expression is the center of mass of the proximal policy EWMA, measured in gradient steps.) This adjustment is what keeps the age of the proximal policy constant, measured in environment steps.

PPO/PPG-EWMA

- More specifically, to achieve batch size-invariance for PPO- and PPG-EWMA, we make the following adjustments to compensate for the optimization and iteration batch sizes being divided by some constant c:
  - If using advantage normalization, multiply the number of iterations used to estimate the advantage mean variance by c. (In practice, we use EWMAs to estimate the mean and variance, and multiply their effective sample sizes, measured in iterations, by c.) This keeps the overall sample sizes of these estimates constant, preventing their standard errors becoming too large.
  - For PPG, multiply the number of policy iterations per phase N_π by c. (We use PPG.)

PPO/PPG-EWMA

- We require that the optimization batch size is sufficiently small. We also require that the number of policy epochs (denoted E in PPO or E_π in PPG) is 1. This is because when the iteration batch size is very small, using multiple policy epochs essentially amounts to training on the same data multiple times in a row, which is redundant (modulo changing the learning rate).
- Our batch size-invariance experiments therefore use PPG-EWMA, where $E_\pi = 1$ is the default.

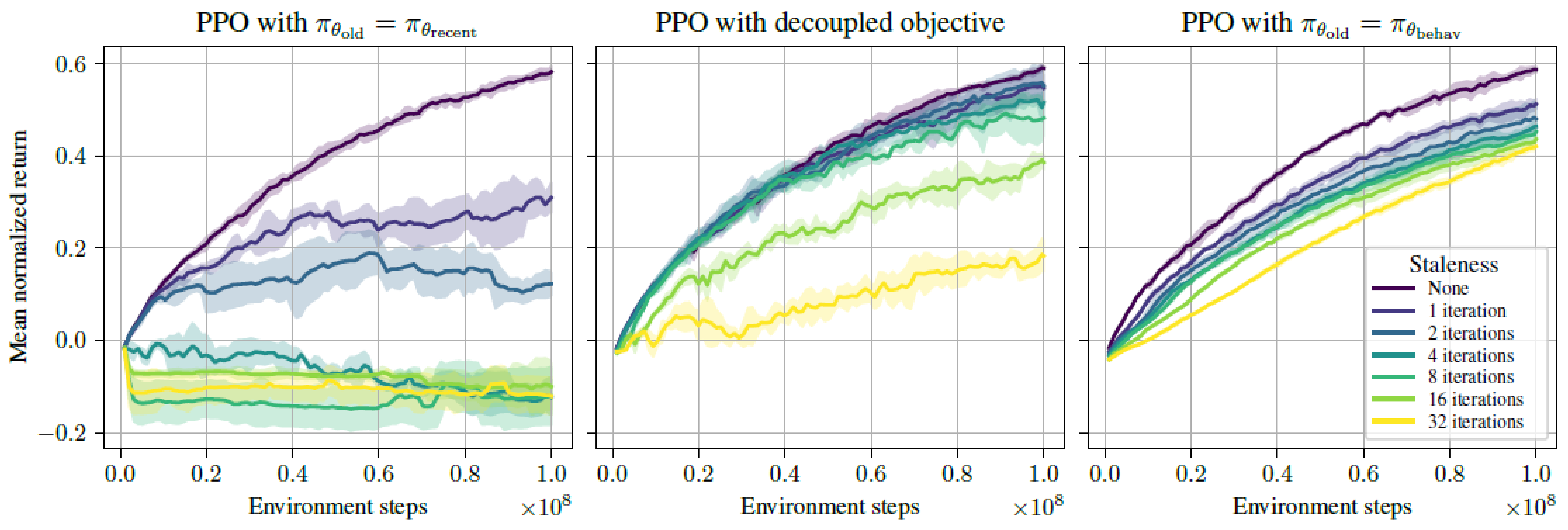
PPO/PPG-EWMA

- Note that PPG has a third batch size: the number of environment steps in each alternation between phases, which we call the **phase batch size**. The effect of our adjustment to N_π is to simply hold the phase batch size constant, thereby preserving the dynamics of the policy and auxiliary phases.

Experiments

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- Artificial Staleness



(a) Using the recent policy for importance sampling introduces bias that makes training highly unstable for even small amounts of staleness.

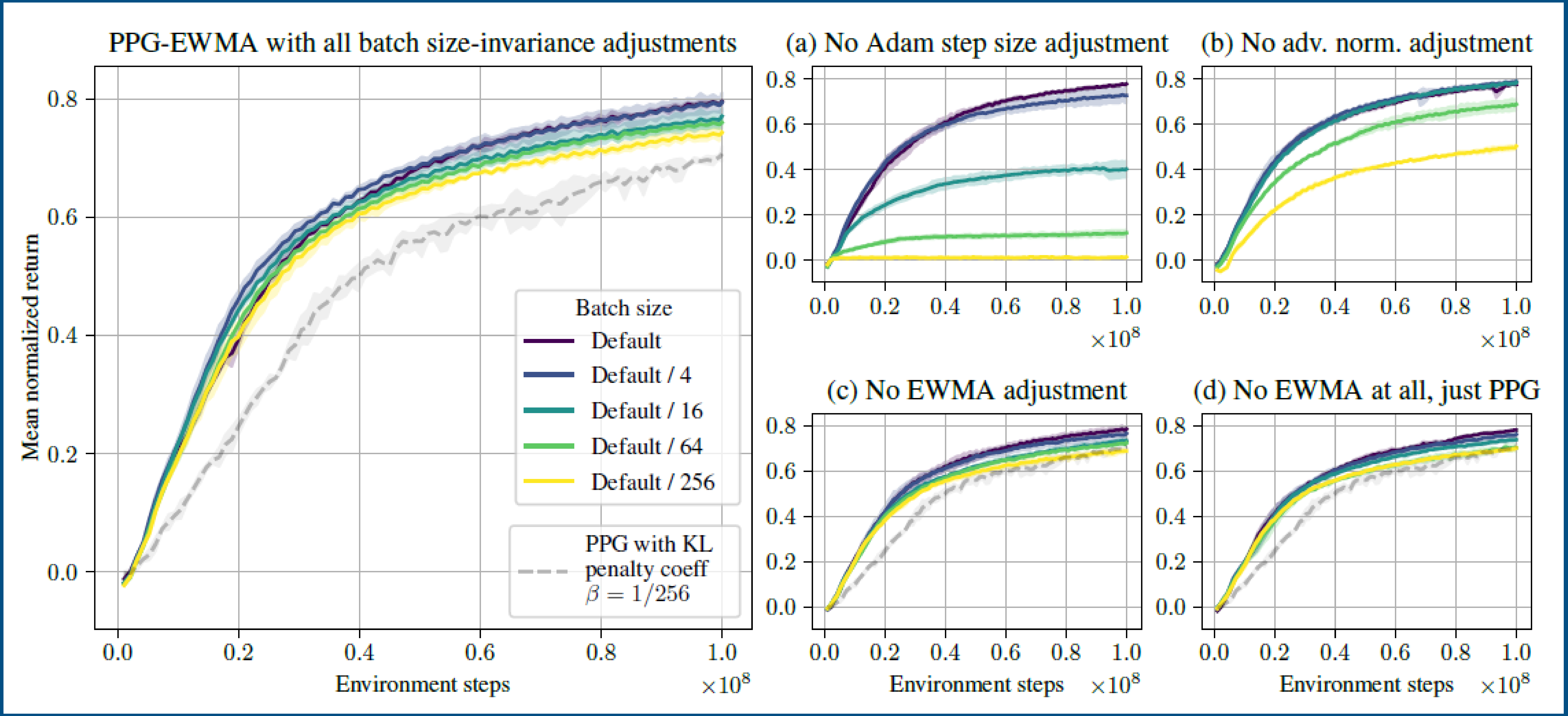
(b) The decoupled objective allows the correct importance sampling ratio to be used while maintaining the age of the proximal policy, preventing performance from degrading much until the data is very stale.

(c) Using the behavior policy to control the size of policy updates holds back learning unnecessarily for small amounts of staleness, but the additional stability is helpful for very stale data.

Experiments

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

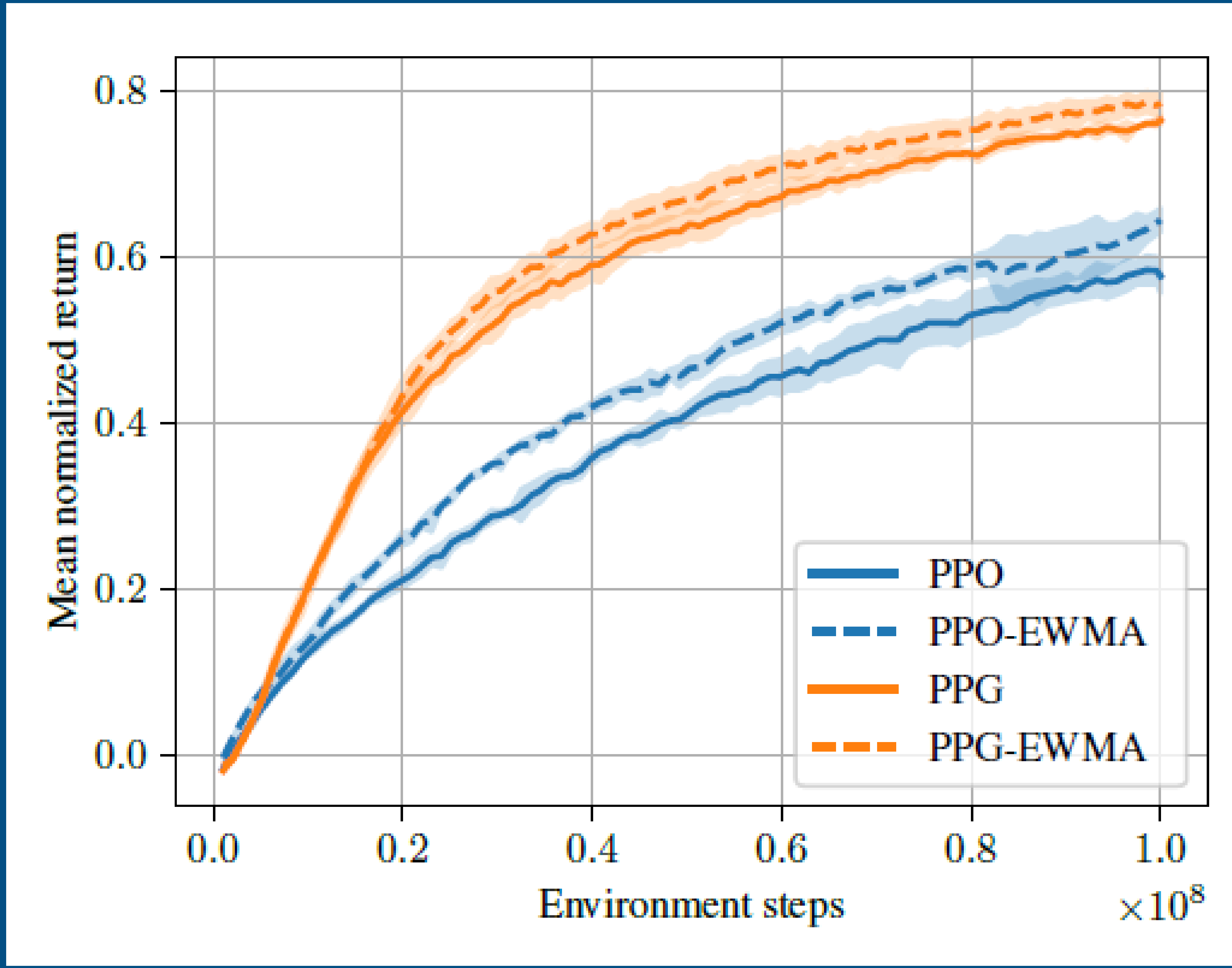
- Batch size-invariance

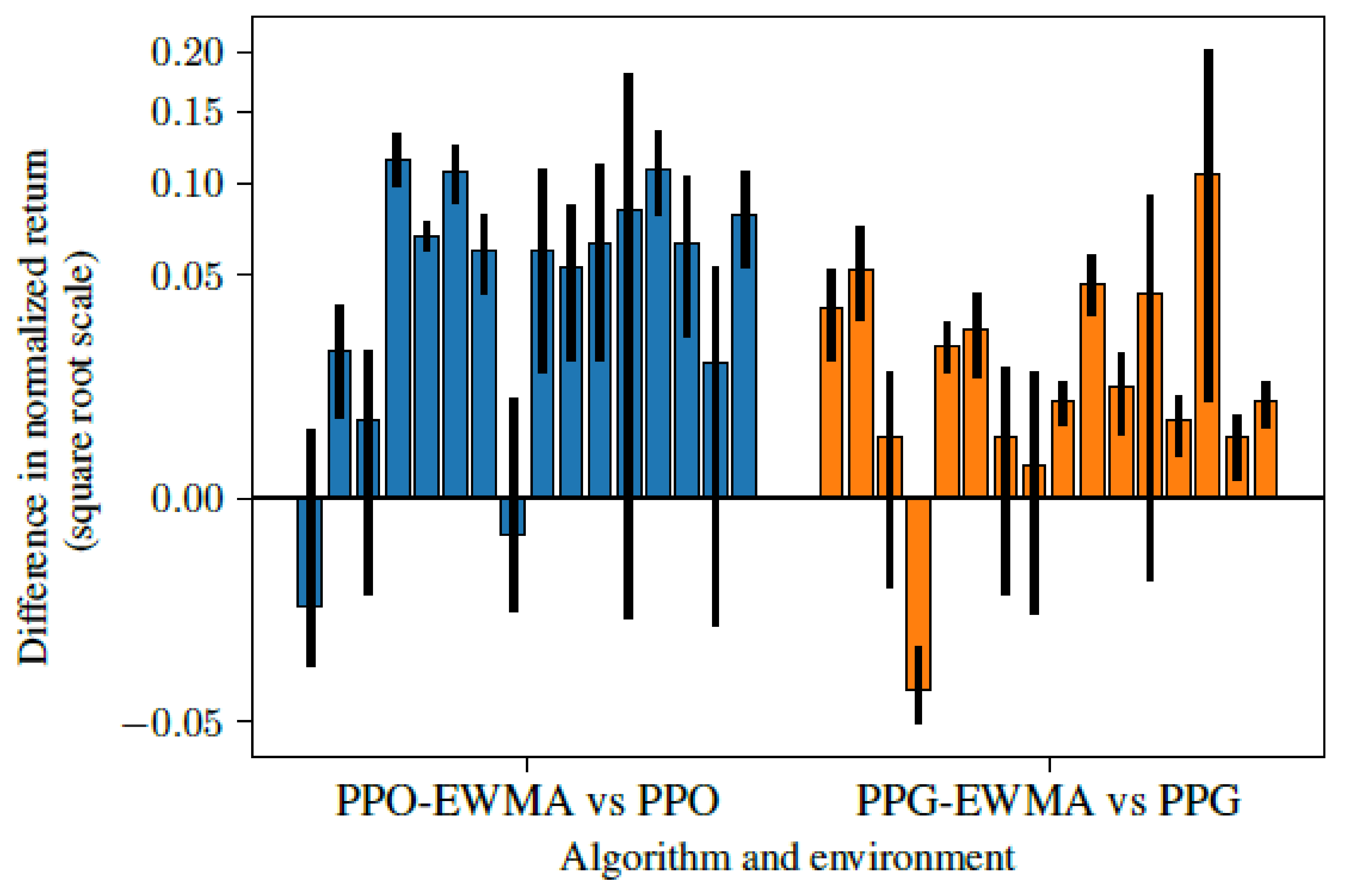


Experiments

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- EWMA comparison





Conclusion

- Policy optimization algorithms such as PPO typically control the size of policy updates using a recent policy we call the proximal policy. We have shown that this policy can be safely decoupled from the behavior policy, which is used to collect experience.
- We introduced PPO-EWMA and PPG-EWMA, variants of PPO and PPG in which the proximal policy is an exponentially-weighted moving average of the current policy. These variants allow stale data to be used more efficiently, and are slightly more sample efficient outright.

Conclusion

Batch Size-invariance for Policy Optimization
J. Hilton et al, 2022

- Finally, we showed how to make these algorithms batch size-invariant, meaning that when the batch size is changed, we can preserve behavior, as a function of the number of examples processed, by changing other hyperparameters (as long as the batch size is not too large).

Thank you!