BEYOND FIXED LEARNING RATES: STOCHASTIC ARMIJO LINE SEARCH FOR ON-POLICY POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Most reinforcement learning algorithms remain far from general purpose: they must be re-tuned for each new problem. The step size is a particularly challenging hyperparameter. Good step sizes depend on many interacting algorithmic components such as function approximators, objectives, and optimizers. Even the environment and experimental procedure can influence which step sizes work. Although the community has developed good default step sizes for many benchmark suites, the inherent dependence of the step size on the environment and experimental process often necessitates re-tuning on new problems or experiment setups. The optimization community has long relied on line search as a robust method of step size selection, and stochastic line searches are gaining popularity in supervised learning. Yet, a systematic exploration of line search strategies in reinforcement learning is lacking. In this paper, we present such a systematic empirical study, developing a line search method for on-policy policy optimization with PPO, which we call ***A**rgmin **A**rmijo **A**dam* (Adam3). We benchmark Adam3 across multiple popular benchmark suites and provide ablations to show failures in basic line search approaches and the utility of the additions in Adam3. Overall, we find Adam3 is easy to implement, does not increase wall-clock time and removes the step size as a hyperparameter, incurring little cost when the defaults work but performing well when they fail.

1 INTRODUCTION

Hyperparameter selection remains one of the central challenges in training reinforcement learning (RL) algorithms. Tiny changes to hyperparameter values can induce large fluctuations in performance. The stochastic nature of reinforcement learning only exacerbates this challenge. Good hyperparameter values are difficult to ascertain a priori, and different problems can require different values. Although we appear to have default hyperparameter configurations that work well on popular benchmark suites (Achiam, 2018; Raffin et al., 2021; Castro et al., 2018; Liang et al., 2018; Wu et al., 2021), they do not always work well in new tasks, as our results show. More innocuously, good performance does not generalize to different interaction budgets (Mesbahi et al., 2025).

The step size hyperparameter is particularly challenging. A good setting depends on the optimization landscape, itself a function of the environment, objective, and function approximator. Further, compared to the common practice of holding the step size fixed, adapting it over time can often be better. Modern optimizers like RMSProp or Adam (Kingma & Ba, 2015; Hinton, 2012) improve step size sensitivity compared to traditional stochastic gradient descent, and much of deep RL's empirical success is owed to these optimization algorithms. Nevertheless, modern optimizers can still fail with poorly chosen or constant step sizes (Jordan et al., 2024; Degris et al., 2024).

Despite these challenges, modern optimizers with fixed but tuned step sizes remain the default choice in RL. Perhaps the most common approach to tuning is via informal or systematic grid search, where many hyperparameter combinations are tested. Grid searches can often be quite effective, but they suffer from three major drawbacks. They require knowing the experiment length ahead of time. Trying many different step sizes requires a simulator. Finally, grid search identifies a single constant step-size value that cannot be adapted in the face of learning progress. Bayesian and evolutionary

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

approaches (e.g., AutoML) can more efficiently find good fixed hyperparameter configurations, but are not typically run online incurring the same limitations as grid search.

Meta-gradient methods adapt hyperparameters online by differentiating the learner's update with respect to hyperparameters and descending a meta-objective, removing the need to know experiment length and reducing reliance on sweeps. For step sizes, this idea dates back to IDBD (Sutton, 1992), SMD (Schraudolph, 1999), Autostep (Mahmood et al., 2012), AdaGain (Jacobsen et al., 2019), hypergradient descent (Baydin et al., 2018), and TD extensions (Kearney et al., 2018; Young et al., 2019; Thill, 2015), but none has been shown to work reliably in deep RL. A separate line differentiates through deep RL updates (Xu et al., 2018) to learn discount factors, λ-returns, targets, objectives, intrinsic rewards, and optimizers (Xu et al., 2020; Oh et al., 2020; Kirsch et al., 2020; Wang & Ni, 2020; Zheng et al., 2018; 2020; Zahavy et al., 2020; Bonnet et al., 2022; Lan et al., 2024). These methods remain uncommon, likely because of their complexity, compute and data costs, and their own meta-hyperparameters, such as the meta step size.

Backtracking line search is a simple alternative, widely used in the optimization literature (Nocedal & Wright, 2006; Zhang & Hager, 2004; Armijo, 1966). They evaluate a direction d for a step size that sufficiently reduces the objective's value. The Armijo Condition is one such *sufficient decrease* condition, selecting step sizes η that satisfy

$$f(x + \eta d) \leq f(x) + c\,\eta\,\langle \nabla f(x), d\rangle \quad \text{where} \quad c \in (0, 1) \tag{1}$$

for the objective function f. Line searches often require only $\langle \nabla f(x), d\rangle < 0$, providing a general mechanism for selecting selecting step sizes (Vaswani et al., 2020; Fan et al., 2023a;b; Shea & Schmidt, 2024). Although originally designed for the deterministic setting, they have been extended to the stochastic supervised setting and tabular RL (Lu et al., 2024), demonstrating impressive empirical performance and enjoying robust theoretical guarantees (Vaswani et al., 2019; Galli et al., 2023; Shea & Schmidt, 2024; Vaswani et al., 2020; Fox et al., 2024b; Vaswani & Babanezhad, 2025; Fan et al., 2023a;b; Jiang & Stich, 2023). Outside a minor role in TRPO (Schulman et al., 2015), stochastic line search methods have not been successfully combined with deep RL to the best of our knowledge.

In this paper, we present the first systematic empirical study of stochastic line search methods for proximal policy optimization (PPO). Naively applying line search to PPO performs poorly, as our ablations show, but our new algorithm, ***A**rgmin **A**rmijo **A**dam* (Adam3), works well across a range of problems. Four changes make it work: searching along momentum-corrected directions, adjusting PPO's standard batch sizes, using a minimizing-loss Armijo step size selection criterion, and starting the actor line search from the chosen critic step size. We benchmark Adam3 on environments from six suites: Brax (Freeman et al., 2021), Jumanji (Bonnet et al., 2024), MinAtar (Young & Tian, 2019), Navix-MiniGrid (Pignatelli et al., 2025), Bsuite (Osband et al., 2020), and Craftax (Matthews et al., 2024). Our results show that (1) Adam3 matches PPO's default step sizes on the widely used benchmarks for which those defaults were tuned; (2) Adam3 outperforms the defaults on several tasks in the newer Jumanji and Navix-Minigrid suites, showing that defaults do not always transfer; and (3) default step sizes degrade significantly in longer experiments, whereas Adam3 adapts automatically and maintains performance. The case for line search is therefore simple: it is easy to implement, removes the step size as a hyperparameter, costs little where the defaults work, and wins where they fail. To provide further insight, we investigated whether sharpness issues play a role; unlike supervised learning (Roulet et al., 2024; 2023), we found little evidence that progressive sharpening hinders learning.

2 Background and Problem Formulation

The agent-environment interaction is formalized as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, d_0, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ provides a (possibly stochastic) scalar signal following each transition. The dynamics function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ maps state-action pairs to distributions over subsequent states, where $\Delta_{\mathcal{X}}$ denotes the set of distributions over $\mathcal{X}$. The initial state is sampled from the starting distribution $d_0 \in \Delta_{\mathcal{S}}$, and $\gamma \in [0, 1]$ is the discount factor, which can take on the value of 1 only in finite-horizon settings.

The agent-environment interaction proceeds in discrete time steps $t = 0, 1, \ldots$. The agent starts in a state $S_0 \sim d_0$. At each step, it observes the current environment state S_t and selects an action $A_t \sim$

$\pi(\cdot \mid S_t)$ according to its policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$. The environment transitions to a new state $S_{t+1} \sim \mathcal{P}(S_t, A_t)$ and provides the agent with a reward $R_{t+1} = \mathcal{R}(S_t, A_t, S_{t+1})$. The agent's objective is to maximize the return G_t in expectation, where G_t is defined as the cumulative discounted future reward $G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. The task of finding a policy to maximize the expected return $\mathbb{E}_{\pi,\mathcal{P},d_0}[G_0]$ is referred to as the *control* problem.

To learn a policy which maximizes return, algorithms often rely on value functions which quantify the utility of states or actions in terms of reward. The state-value function $v_\pi : \mathcal{S} \to \mathbb{R}$ quantifies the expected return when starting in some state and following policy π thereafter, $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$. The action-value function $q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the expected return when taking an action in a given state and following π thereafter, $q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$. The advantage function $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$ quantifies the average benefit of taking action a in state s under π. In environments with high-dimensional or continuous state or action spaces, these functions are typically estimated using function approximation. In this case, we learn functions $v_{\boldsymbol{\theta}}$ and $q_{\boldsymbol{w}}$ such that $v_{\boldsymbol{\theta}}(s) \approx v_\pi(s)$ and $q_{\boldsymbol{w}}(s,a) \approx q_\pi(s,a)$, where $\boldsymbol{\theta}$ and $\boldsymbol{w}$ are learnable parameters. The task of accurately estimating these values for a fixed policy is known as the *prediction* problem. Monte Carlo methods for prediction provide unbiased targets for learning these value functions using full trajectory rollouts, but can suffer from high target variance. Temporal Difference (TD) methods can reduce variance by bootstrapping targets from existing value estimates but may introduce some bias.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) learns a parameterized policy and value function. We denote the policy by $\pi_{\boldsymbol{\phi}}(a \mid s)$, with parameters $\boldsymbol{\phi}$, and the state-value function by $v_{\boldsymbol{\theta}}(s)$, with parameters $\boldsymbol{\theta}$. To update $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, PPO collects trajectories using a fixed behavior policy $\pi_{\boldsymbol{\phi}_{\text{old}}}$. Advantage and value targets are constructed using a fixed critic $v_{\boldsymbol{\theta}_{\text{old}}}$. Generalized advantage estimation (GAE) (Schulman et al., 2016) constructs the advantage estimate

$$\widehat{A}_t \doteq \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \Delta_{t+l} \qquad \Delta_t \doteq R_{t+1} + \gamma v_{\boldsymbol{\theta}_{\text{old}}}(S_{t+1}) - v_{\boldsymbol{\theta}_{\text{old}}}(S_t),$$

where the bootstrap term is omitted when S_{t+1} is terminal, T denotes the last time step of the trajectory, and $\lambda \in [0,1]$. PPO then minimizes[1] the clipped surrogate actor objective

$$f^{\text{actor}}(\boldsymbol{\phi}) \doteq -\mathbb{E}_t\left[\min\left(r_t(\boldsymbol{\phi})\widehat{A}_t, \text{clip}\left(r_t(\boldsymbol{\phi}), 1-\epsilon, 1+\epsilon\right)\widehat{A}_t\right)\right] \quad r_t(\boldsymbol{\phi}) \doteq \frac{\pi_{\boldsymbol{\phi}}(A_t \mid S_t)}{\pi_{\boldsymbol{\phi}_{\text{old}}}(A_t \mid S_t)},$$

where $\epsilon > 0$ controls the clipping range. PPO optimizes f^{actor} by separating the trajectory rollouts into multiple minibatches and descending on those minibatches for multiple epochs. The critic is similarly trained by regressing toward the fixed target $\widehat{V}_t$:

$$f^{\text{critic}}(\boldsymbol{\theta}) \doteq \frac{1}{2}\mathbb{E}_t\left[\left(v_{\boldsymbol{\theta}}(S_t) - \widehat{V}_t\right)^2\right] \qquad \widehat{V}_t \doteq \widehat{A}_t + v_{\boldsymbol{\theta}_{\text{old}}}(S_t).$$

To optimize these objective functions, PPO requires multiple hyperparameters to be set, including step sizes η^{actor} and η^{critic} for both actor and critic respectively.[2]

Armijo Line Search A prominent class of methods for automatic step size selection is the line search (Nocedal & Wright, 2006; Armijo, 1966; Vaswani et al., 2019; Vaswani & Babanezhad, 2025; Galli et al., 2023; Fox et al., 2024a;b; Shea & Schmidt, 2024). These methods dynamically search for a good step size η_k along the direction u_k. In particular, the stochastic Armijo line search selects an η_k that satisfies the *stochastic Armijo condition* (Armijo, 1966; Vaswani et al., 2019):

$$f_k(x_k + \eta_k u_k) \leq f_k(x_k) + c\eta_k \langle \nabla f_k(x_k), u_k \rangle, \tag{2}$$

where f_k is a stochastic objective function, x_k is the optimization parameter, $c \in (0,1)$ is a hyperparameter and $\langle \nabla f_k(x_k), u_k \rangle < 0$. Line searches are flexible and can be layered atop any optimizer that produces an update vector u_k that satisfies this property.

[1] While it is standard to consider maximizing the actor objective, we consider minimizing it to unify notation between actor and critic later.

[2] We may drop the qualifying superscript from the step size, writing η, when the appropriate step size is clear from context.

Backtracking is often used to recover an η_k that satisfies Equation 2. In the context of an Armijo line search, backtracking evaluates a sequence of step sizes and chooses

$$\max_{i=0,1,2,\ldots,B-1} \eta^{\text{init}}\delta^i \quad \text{such that Equation 2 is satisfied} \tag{3}$$

where $\eta^{\text{init}} > 0$ is some initial step size. The backtracking procedure typically falls back to $\eta^{\text{init}}\delta^{B-1}$ if no candidate is found. Historically, the sequential cost of B backtracking evaluations led to heuristics aimed at optimizing the initial guess η^{init}. Notably, Vaswani et al. (2019) propose to initialize each iteration k of the line search from $\eta_{k-1}\delta^{b/n}$ for finite-sum supervised learning objectives, where $\delta > 1$, n is the dataset size, and b is the batch size. Galli et al. (2023) suggest using Polyak step sizes for initialization.

3 ADAM3: LINE SEARCH FOR PPO WITH ADAM

In this section, we introduce ***A**rgmin **A**rmijo **A**dam* (Adam3), our line search algorithm that is compatible with Adam (Kingma & Ba, 2015). At a high-level, there are three key components to this algorithm. 1) It uses a modified Armijo condition that searches a different direction than the Adam direction, which may not be a descent direction. 2) It replaces the standard backtracking search with a criterion that picks the step size with minimum loss, using a parallelized search to avoid increasing wall-clock time. 3) It exponentially updates the step size to be more robust to stochasticity.

Adam3 is summarized in Algorithm 1. Algorithm 3 combines it with the PPO training loop. On update k, Adam3 forms the Adam update direction u_k as well as its momentum-free analogue, $\tilde{u}_k$ using Algorithm 2:

$$u_k = -\frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \varepsilon_{\text{adam}}} \qquad m_k = \beta_m m_{k-1} + (1-\beta_m) g_k \qquad v_k = \beta_v v_{k-1} + (1-\beta_v) g_k^2 \tag{4}$$

$$\tilde{u}_k = -\frac{g_k}{\sqrt{\hat{v}_k} + \varepsilon_{\text{adam}}} \qquad \hat{m}_k = \frac{m_k}{1-\beta_m^{k+1}} \qquad \hat{v}_k = \frac{v_k}{1-\beta_v^{k+1}} \tag{5}$$

where f_k is some stochastic objective, $g_k = \nabla f_k(x_k)$, and x_k are some function approximator parameters. Notice that $\langle g_k, \tilde{u}_k \rangle < 0$, making it an appropriate line search direction, and that $\tilde{u}_k$ should be somewhat aligned with u_k.

Adam3 selects step sizes η_k using a line search along $\tilde{u}_k$ while updating parameters along u_k. Since line searches are parallelizable on modern hardware accelerators, we replace the backtracking procedure with a more careful search. Adam3 considers the same set of geometrically decaying step sizes as a backtracking search $H_{\eta^{\text{init}}} = \left\{\eta = \eta^{\text{init}}\delta^i \mid i = 0, 1, \ldots, B-1\right\}$, but chooses the step size that obtains the minimum loss across Armijo-feasible step sizes:

$$\eta_k = \text{argmin}_{\eta \in H_\eta^{\text{init}}} f_k(x_k + \eta\tilde{u}_k) \quad \text{s.t.} \quad f_k(x_k + \eta\tilde{u}_k) \leq f_k(x_k) + \eta\, c\, \langle \nabla f_k(x_k), \tilde{u}_k \rangle \tag{6}$$

where f_k is the stochastic objective at step k. In this way, Adam3 is an approximately-exact line search – it discretizes the objective along $\tilde{u}_k$ and exactly optimizes this discretization. Adam3 therefore uses an *argmin* criterion to select amongst step sizes that satisfy the Armijo condition, unlike backtracking which selects the largest step size that satisfies the Armijo condition. Finally, to remain robust to stochasticity, Adam3 smooths this step size with a bias-corrected exponential moving average (EMA) of recent step sizes to obtain $\widehat{\eta}_k$. The parameters are updated as $x_{k+1} \leftarrow x_k + \widehat{\eta}_k u_k$.

To combine Adam3 with PPO, we update both actor and critic using Adam3. At iteration k, the critic begins its line search from $\eta_k^{\text{init,critic}} = \eta^{\text{init,critic}} = 1$ and chooses the step size that satisfies Equation 6 from the set $H_{\eta^{\text{init,critic}}}$. Similarly, the actor selects its step size from $H_{\eta_k^{\text{init,actor}}}$ to satisfy Equation 6. The actor's line search starts from $\eta_k^{\text{init,actor}}$, which we set to the step size used in the last critic parameter update. We found that the actor line search was susceptible to choosing excessively large step sizes which would ruin learning, and this strategy helps to prevent that.

Vaswani et al. (2020) studied a similar algorithm, namely using a line search on a preconditioned gradient direction while updating on a preconditioned momentum direction. Their line search was layered atop AMSGrad Reddi et al. (2018), for which they provided convergence guarantees in convex settings. We layer atop the related Adam optimizer using an explicit moving average of step

Algorithm 1 Argmin Armijo Adam Step (ADAM3)

Require: Objective f, current parameters x, previous raw moments m_-, v_-, previous EMA step size $\bar{\eta}_-$, iteration index k, line search starting step η^{init}, $c, \delta \in (0,1)$, $B > 0$, β_η

$g, A, \hat{m}, m, v \leftarrow \text{ADAMPROBE}(f, x, m_-, v_-, k)$
$u = -A\hat{m}, \quad \tilde{u} = -Ag, \quad s = \langle \tilde{u}, g \rangle$
Let $H = \{\eta = \eta^{\text{init}}\delta^i \mid i = 0, 1, \ldots, B-1\}$
$\eta^* = \text{argmin}_{\eta \in H} f(x + \eta\tilde{u})$ s.t. $f(x + \eta\tilde{u}) \leq f(x) + \eta\, c\, s$ or $\eta^{\text{init}}\delta^{B-1}$ if none exist
$\bar{\eta} = \beta_\eta \bar{\eta}_- + (1 - \beta_\eta)\eta^*$
$x' = x + \frac{\bar{\eta}}{1-\beta_\eta^{k+1}}\, u$
return $x', m, v, \bar{\eta}$

Algorithm 2 Adam Preconditioner and Momentum (ADAMPROBE)

Require: Objective f, current parameters x, previous raw moments m_-, v_-, iteration index k

$g = \nabla f(x)$
$m = \beta_m m_- + (1 - \beta_m) g$
$v = \beta_v v_- + (1 - \beta_v) g^2$
$A = \text{diag}\left(\sqrt{v/(1 - \beta_v^{k+1})} + \varepsilon_{\text{adam}}\right)^{-1}$
$\hat{m} = \frac{m}{1-\beta_m^{k+1}}$
return $g, A, \hat{m}, m, v$

sizes to remain robust to stochasticity and a loss-minimizing step size acceptance criterion. While the analysis of Vaswani et al. (2020) was mainly theoretical, we focus on an empirical characterization of this approach in deep reinforcement learning.

Algorithm 3 PPO with Adam3

Require: PPO critic loss f^{critic}, PPO actor surrogate f^{actor}, initial parameters $\boldsymbol{\theta}_0, \boldsymbol{\phi}_0$, initial critic step $\eta^{\text{init}} > 0$, number of epochs E, number of minibatches M

Initialize: $t \leftarrow 0$; $\bar{\eta}_{-1}^{\text{critic}}, \bar{\eta}_{-1}^{\text{actor}} \leftarrow 0, 0$; all moments $\leftarrow 0$
for $k = 0, 1, \ldots$ **do**
  Collect a batch $\mathcal{D}_k$ of on-policy rollout data using $\boldsymbol{\phi}_t$
  Compute advantage estimates and value targets on $\mathcal{D}_k$; set $\boldsymbol{\phi}_{\text{old}} \leftarrow \boldsymbol{\phi}_t$
  for $e = 1, \ldots, E$ **do**
    Shuffle $\mathcal{D}_k$ and partition it into minibatches $\mathcal{B}_1, \ldots, \mathcal{B}_M$
    for $j = 1, \ldots, M$ **do**
      $\boldsymbol{\theta}_{t+1}, m_t^{\text{critic}}, v_t^{\text{critic}}, \bar{\eta}_t^{\text{critic}} \leftarrow \text{ADAM3}\left(f_{\mathcal{B}_j}^{\text{critic}}, \boldsymbol{\theta}_t, m_{t-1}^{\text{critic}}, v_{t-1}^{\text{critic}}, \bar{\eta}_{t-1}^{\text{critic}}, t, \eta^{\text{init}}\right)$
      $\boldsymbol{\phi}_{t+1}, m_t^{\text{actor}}, v_t^{\text{actor}}, \bar{\eta}_t^{\text{actor}} \leftarrow \text{ADAM3}\left(f_{\mathcal{B}_j}^{\text{actor}}, \boldsymbol{\phi}_t, m_{t-1}^{\text{actor}}, v_{t-1}^{\text{actor}}, \bar{\eta}_{t-1}^{\text{actor}}, t, \frac{\bar{\eta}_t^{\text{critic}}}{1-\beta_\eta^{t+1}}\right)$
      $t \leftarrow t + 1$
    end for
  end for
end for
return $\boldsymbol{\theta}_t, \boldsymbol{\phi}_t$

4 EMPIRICALLY CHARACTERIZING ADAM3 WITH PPO

In this section, we empirically characterize PPO with Adam3 across a number of popular benchmark suites. We demonstrate that Adam3 matches PPO with its defaults on widely used benchmarks for which those defaults were tuned but outperforms the defaults on several environments from the Jumanji and Navix-Minigrid suites. Further Adam3 remains useful while defaults degrade in longer experiments. We show that since line searches are parallelize on modern hardware accelerators, our Adam3 implementation does not increase runtime significantly on a seed-by-seed basis. We present an ablation study of Adam3, motivating each of its algorithmic components. Finally, we

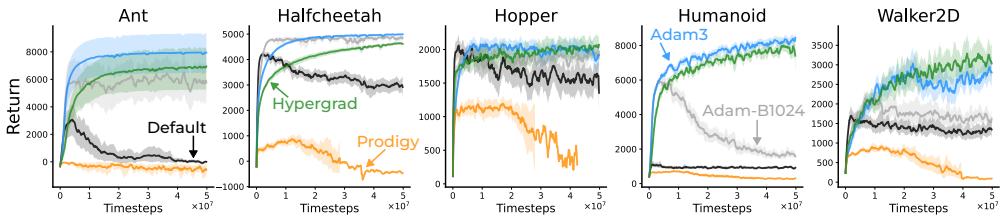


Figure 2: Learning curves for PPO on Brax for 50 million timesteps, 20 seeds.

demonstrate a lack of evidence suggesting progressive sharpening hinders learning in PPO control, in stark contrast with supervised learning (Roulet et al., 2023; 2024). Unless otherwise noted, we report mean performance across random seeds. Error bars denote 95% bootstrap confidence intervals with stratified sampling when aggregated across environments.

Baselines and Hyperparameters Our empirical study compares Adam3 to using both default (fixed) step sizes of $\eta = 3e-4$ (Raffin et al., 2021; Huang et al., 2022) and adaptive step size baselines using PPO with Adam. For all methods, we use default hyperparameters for PPO from Stable-Baselines3 (Raffin et al., 2021) unless otherwise noted. We used rollouts of 2048 timesteps and 10 epochs as typical. For the adaptive step size baselines, we used both Hypergradient Descent (Baydin et al., 2018) and Prodigy (Mishchenko & Defazio, 2024). Hypergradient is a popular meta-gradient approach that adjusts a global step size via gradient descent by backpropagating through one step of the underlying optimizer. We set the meta step size to $1e{-}7$, which was used when training neural networks with Adam in supervised learning (Baydin et al., 2018). Prodigy is a parameter-free optimizer that extends D-Adaptation (Defazio & Mishchenko, 2023) by dynamically estimating the distance to the global optimum in the convex setting, $D = \|x_0 - x^*\|$, and scaling the step size proportionally.

Consistent with findings in the supervised learning literature, we found that line searches systematically select larger step sizes with smaller batch sizes (Roulet et al., 2023). To mitigate this, we used a minibatch size of 1024 per epoch for Adam3. To control for this difference in batch size, we ran PPO with both of these batch sizes: 64 and 1024. We refer to this variant of PPO as *Adam-B1024*, and we refer to the variant that uses the default step size and batch size of 64 simply as *Default*. For all other methods, we used the standard 64 samples per minibatch since we found it to perform better. We kept all other hyperparameters at their defaults.

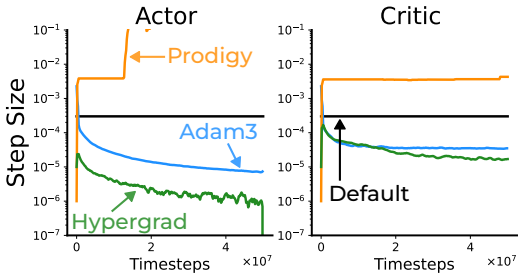


Figure 1: Step sizes chosen on HalfCheetah.

There are a few other hyperparameters to note for Adam3. We set $c = 0.7$, $\delta = 0.7$, $\eta^{\text{init,critic}} = 1.0$, $B = 30$ backtracking steps, and $\beta_\eta = 0.99$. We set δ, B, $\eta^{\text{init,critic}}$, and β_η heuristically without tuning. The value of $c = 0.7$ was chosen by a small sweep (see Section 4.3) as well as theory (Vaswani et al., 2020). We study hyperparameter sensitivity in Appendix C.3; we found Adam3 was not sensitive to its hyperparameters. Our implementation uses `jax`, which allows us to use `vmap` across all 30 backtracking steps. Details on other hyperparameters are given in Appendix B.

4.1 Adam3 Discovers Good Step Sizes Across Popular Benchmark Suites

In this section, we empirically demonstrate that Adam3 can attain good performance and discover good step sizes across environments from popular benchmark suites. The suites include Brax (50 million steps, 20 seeds); four environments from the Jumanji suite (30 seeds), namely Game2048-v1, Graphcolouring-v1, Knapsack-v1, and Snake-v1; MinAtar (20 seeds; Young & Tian (2019)); Navix-MiniGrid (50 seeds; Pignatelli et al. (2025)); Bsuite (100 seeds; Osband et al. (2020)); and Craftax (30 seeds; Matthews et al. (2024)). We will first focus on experiments with Brax, then discuss the combined experiments on the remaining environment suites.

Adam3 Chooses Good Step Sizes Across Brax Environments Figure 2 shows the learning curves on Brax. Across 50 million timesteps, Adam3 does well, outperforming Default across

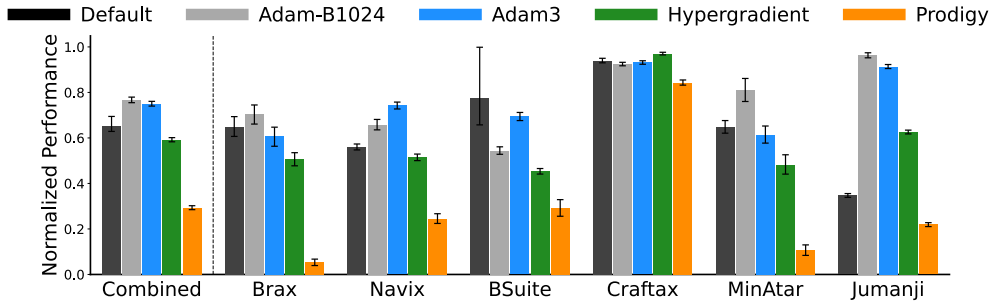


Figure 3: Mean normalized return across benchmark suites.

all problems. Adam-B1024 performs noticeably better than Default and often matches the performance of Adam3. Notably, however, Adam-B1024 fails in Humanoid and its performance is poor in Walker2D, whereas Adam3 performs well. Prodigy does quite poorly throughout, while Hypergradient does well.

Figure 1 shows the step sizes chosen over time by Adam3, Hypergradient, and Prodigy on HalfCheetah. Step sizes for the remaining environments are shown in Appendix C. Across both actor and critic, Prodigy picks the largest step sizes, Hypergradient the smallest, and Adam3 between the two. For the actor step size, both Adam3 and Hypergradient select step sizes that mimic a scheduled decay, but discovered automatically. For the critic, Adam3 and Hypergradient pick similar step sizes.

Adam3 Performs Well Across Standard Benchmark Problems We next compare these methods in standard benchmarks. We do not expect large wins from Adam3 compared to the defaults in this standard setting, unlike the previous experiments, as here PPO's defaults are tuned for these standard benchmarks. Rather, we want comparable and consistent performance from Adam3. For Brax in these experiments, we switched back to the standard 3 million timesteps rather than the 50 million timesteps used in Figure 2.

Figure 3 shows the average normalized return attained by each algorithm on the remaining benchmark suites. We normalized performance using Winsorized Min-Max normalization (5th and 95th percentiles) per seed per environment. More details on the normalization procedure are given in Appendix B. Learning curves are shown in Appendix C.

Adam3 performs consistently well across these environment suites. It slightly outperforms Default and Adam-B1024 on Navix and otherwise typically matches Default and Adam-B1024, often doing nearly as well as the best of the two. In MinAtar, Adam-B1024 is better than Adam3. In BSuite, Default is a bit better and Adam-B1024 notably worse than Adam3. Finally in Jumanji, we again see the opposite: Default performs quite poorly, and Adam-B1024 and Adam3 perform similarly. No one setting with defaults (small or big minibatch) consistently match Adam3. Across all suites, Adam3 outperforms Prodigy and often outperforms Hypergradient.

4.2 Ablation Study on Adam3

In this section, we empirically motivate different algorithmic choices in Adam3. Four major algorithmic choices contribute to the Adam3 method and its application to PPO. The first and most important algorithmic choice is the line search along $\tilde{u}_k$ rather than along the update direction u_k itself. The second is the larger batch size used in the PPO update. The third is starting the actor line search from the critic step size rather than from 1. The final choice is selecting the loss-minimizing step size, as in Equation 6, rather than utilizing backtracking (Equation 3). We ablated each of these choices

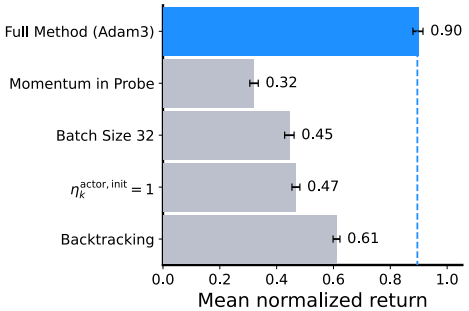


Figure 4: Mean normalized performance across ablations on Brax (3M) and Jumanji, aggregated across environments. The **blue** bar denotes Adam3; others denote ablations.

across Brax (3M steps, 20 runs) and Jumanji (30 runs) and report the aggregate normalized performance of each ablation in Figure 4. Figure 7 in Appendix C.1 shows the ablation in each individual environment.

Momentum-Free Search Directions are Essential We compared the momentum-free direction used in Adam3 for the line search versus using the momentum direction u_k, denoted *Momentum in Probe* in Figure 4. Recall that u_k may not be a descent direction on f^{actor}. In such cases, we consider step sizes η that make any progress on the surrogate objective (setting $c = 0$ in Equation 6), again choosing the step size that makes the most progress, falling back to the minimum if none exists. Figure 4 shows that this kind of line search is catastrophic, even though step sizes are selected to make progress on the objective. We found that very often $\langle \nabla f^{\text{actor}}(\phi), u_k \rangle \approx 0$, and that the line search chose excessively large step sizes under these conditions.

Small Batch Sizes can Ruin Learning We compared using batches of 1024 samples versus the standard 64, denoted *Batch Size 64* in Figure 4. We see larger batch sizes are essential for effective learning. We found that line searches tended to pick systematically larger step sizes with smaller batch sizes, and hypothesize that this explains this failure.

Line Searches should start from a Reasonable Initial Actor Step Size We also compared starting the actor line search from $\eta^{\text{init,actor}} = 1$ instead of from the critic step size. This variant, denoted $\eta^{\text{init,actor}} = 1$ Figure 4, notably decreased normalized return across the benchmark suites. Interestingly, this approach is actually quite good in some environments, but catastrophically bad in others (see Figure 7 in Appendix C.1). Starting the actor line search from the critic step size serves as a compromise across these problems that overall improves learning.

Argmin Step size Selection Edges out Backtracking We ablated our choice of using the argmin selection criterion, outlined in Equation 6, that selects the step size that minimizes the loss amongst the step sizes that satisfy the Armijo condition. This contrasts with the standard approach of backtracking, which selects the largest step size that satisfies the Armijo condition, as in Equation 3. Figure 4 shows that backtracking, denoted *Backtracking*, results in lower normalized return across the considered benchmark environments. In general, across all these ablations, performance was significantly reduced when deviating from the choices in Adam3.

4.3 Runtime and Sensitivity

In this section, we report runtime throughput for Adam3 compared to the baselines as well as sensitivity to Adam3's hyperparameters.

We report normalized environment steps per second relative to Default (higher is better) in Table 1 averaged across Brax environments. Results are reported as mean $\pm$ standard deviation over 10 trials per environment. To isolate hardware noise, all five methods were run back-to-back in rotated order on the same NVIDIA L40S GPU with one CPU core per trial, and each method's throughput is normalized by the Default run from that same trial (each trial runs PPO for 100,000 steps with a fixed seed for each method). With Adam3, we evaluate the Armijo condition for all candidate step sizes in parallel using the GPU. Adam3 therefore incurs negligible runtime overhead as compared to the default, fixed step size method.

We empirically characterized the sensitivity of Adam3 to its hyperparameters, namely c, the contraction factor δ, and the number of backtracking steps B on Brax (3 million timesteps) and Jumanji using the same experiment setup as outlined in Sec-

Table 1: **Training throughput**

Method	**Throughput**
Default	**1.000**
Adam-B1024	**1.012** $\pm$ **0.006**
Adam3	**1.011** $\pm$ **0.007**
Hypergradient	**0.991** $\pm$ **0.003**
Prodigy	**0.997** $\pm$ **0.004**

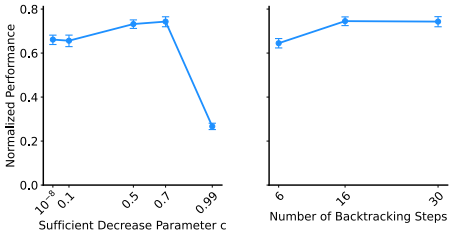


Figure 5: Sensitivity to (left) c and (right) (δ, B) with normalized performance aggregated over Brax (3M steps) and Jumanji.

tion 4.1. We evaluated $c \in \{10^{-8}, 0.1, 0.5, 0.7, 0.99\}$. We coupled the sensitivity study of δ and B, as they control the effective step size range considered. We varied these hyperparameters so that the effective range was similar to that used in the preceding experiments, varying $(\delta, B) \in \{(0.1, 6), (0.5, 16), (0.7, 30)\}$. Figure 5 shows the results. We found little sensitivity to both c and (δ, B). The latter indicates that a fairly coarse line search can be sufficient to attain good performance across these problem suites.

4.4 Little Evidence for Progressive Sharpening Issues in Control

A marked failure case of line search methods in supervised learning is their tendency to induce catastrophic progressive sharpening (Roulet et al., 2023; 2024). We did not find evidence of such behaviour in our experiments. Following Cohen et al. (2024), we plot the maximum preconditioned objective Hessian eigenvalue in Figure 6 for both the actor and critic across the Brax environments. We focus on the first 3 million timesteps for computational feasibility. Jumanji is shown in Figure 9 in Appendix C.2. In contrast to the supervised learning case, we found little evidence of progressive sharpening for Adam3 (Roulet et al., 2023; 2024). Conversely, in some environments, Adam3 exhibits lower sharpness than PPO with the default (fixed) step size. This may perhaps indicate that some inherent structure of the control problem or PPO learning dynamics itself mitigates the risk of a predominant line search failure case in supervised learning.

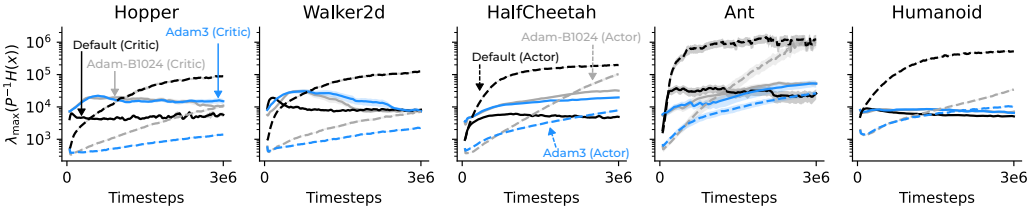


Figure 6: Preconditioned maximum objective Hessian eigenvalues across the Brax environments.

5 Discussion and Conclusion

In this paper, we explored step size selection using line search in PPO (Schulman et al., 2017). We developed Adam3, a line search algorithm based on a modified Armijo condition (Armijo, 1966) that enables PPO to discover good step sizes automatically. Across multiple popular benchmark suites, Adam3 discovered good step sizes, outperforming other adaptive baselines. Further, we found that default step sizes could perform quite poorly, particularly failing in instances where the agent learned for much longer than standard, whereas Adam3 performed consistently across settings. We then provided insight into each of the algorithmic decisions we made in Adam3 through an ablation study. Finally, we demonstrated a marked drift from line searches in supervised learning. We found little evidence that line searches with PPO control produced progressive sharpening to the point of harming learning. Conversely, we found that line searches often found parameters in flatter regions than the default fixed step sizes, in direct contrast to what has been observed in the supervised learning literature (Roulet et al., 2023; 2024). Overall, we find that Adam3 is a promising optimizer for PPO, improving consistency of performance across settings with minimal overhead.

We suggest a few avenues for future development of adaptive step sizes for reinforcement learning. First, step sizes and entropy share an intimate connection in on-policy policy optimization algorithms (Jordan et al., 2024). Future research should focus on developing a better understanding of this interaction, particularly with adaptive step size methods. Second, line searches are parallelizable. The entire search can be executed in parallel on modern accelerator hardware, providing valuable statistics about the geometry of our objective function. In this work, we used this additional information to for a better step size acceptance criterion (argmin) than backtracking, but future work could further understand how this additional objective information can be exploited to choose even better step sizes. Finally, over-optimizing surrogate policy objectives can harm performance, and it is unclear how well the PPO policy surrogate tracks return (Ilyas et al., 2020; Bouftini & Benzaouia, 2026). This relationship complicates the use of line search methods in algorithms like PPO, since the line search actively seeks out step sizes to optimize the surrogate. Future work should develop a deeper understanding of the conditions under which surrogate objective progress remains meaningful for return optimization.

AI USE STATEMENT

We have not used generative AI tools in generating: data, proofs or other theoretical results, algorithmic ideas, or experiments to run. Generative AI was used for developing plotting scripts. We used generative AI to help generate code, but not to run our experiments. We used generative AI to help find related work and basic writing support (grammar checking and rewording suggestions). Generative AI wrote no parts of the paper directly.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on foundational aspects of reinforcement learning and the main contributions are algorithmic and empirical on simulated worlds.

REPRODUCIBILITY STATEMENT

Adam3 is fully specified in Section 3 and Algorithms 1–3. Section 4 describes the baselines, benchmark suites, number of seeds, and all Adam3 hyperparameters. Appendix B lists the remaining PPO implementation details and hyperparameters and describes the normalization procedure used for aggregate results. Both the main text and Appendix A describe the hardware and protocol for our runtime comparison. The main text and Appendix C.3 report sensitivity to Adam3's hyperparameters. All results are averaged over at least 20 seeds and reported with 95

ACKNOWLEDGMENTS

Acknowledgments will be included in the camera-ready manuscript upon acceptance.

REFERENCES

Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. In *Proceedings of the International Conference on Learning Representations*, 2021.

Larry Armijo. Minimization of Functions having Lipschitz Continuous First Partial Derivatives. *Pacific Journal of Mathematics*, 1966.

Atılım Güneş Baydin, Robert Cornish, David Martínez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *Proceedings of the International Conference on Learning Representations*, 2018.

Clément Bonnet, Laurence Midgley, and Alexandre Laterre. Debiasing Meta-Gradient Reinforcement Learning by Learning the Outer Value Function. In *Workshop on Meta-Learning*, 2022.

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.

Rayane Bouftini and Mohammed Benzaouia. Proximal policy optimization suffices for on-policy reinforcement learning. In *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, 2026.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A Research Framework for Deep Reinforcement Learning, 2018.

Jeremy M. Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive Gradient Methods at the Edge of Stability, 2024.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. In *International Conference on Machine Learning*, 2023.

Thomas Degris, Khurram Javed, Arsalan Sharifnassab, Yuxin Liu, and Richard Sutton. Step-Size Optimization for Continual Learning. *arXiv preprint arXiv:2401.17401*, 2024.

Chen Fan, Gaspard Choné-Ducasse, Mark Schmidt, and Christos Thrampoulidis. BiSLS/SPS: Autotune Step Sizes for Stable Bi-level Optimization. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2023a.

Chen Fan, Sharan Vaswani, Christos Thrampoulidis, and Mark Schmidt. MSL: An adaptive momentem-based stochastic line-search framework. *OPT2023 Workshop on Optimization for Machine Learning*, 2023b.

Curtis Fox, Leonardo Galli, Mark Schmidt, and Holger Rauhut. Nonmonotone Line Searches Operate at the Edge of Stability. In *Workshop on Optimization for Machine Learning*, 2024a.

Curtis Fox, Aaron Mishkin, Sharan Vaswani, and Mark Schmidt. Glocal Smoothness: Line Search and Adaptive Step Sizes can Help in Theory too! *Transaction on Machine Learning*, 2024b.

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.

Leonardo Galli, Holger Rauhut, and Mark Schmidt. Don't be so Monotone: Relaxing Stochastic Line Search in Over-Parameterized Models. *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2023.

Geoffrey Hinton. Lecture 6.5-RMSProp: Divide the Gradient by a Running Average of its Recent Magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 Implementation Details of Proximal Policy Optimization, 2022. ICLR Blog Track.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *Proceedings of the International Conference on Learning Representations*, 2020.

Andrew Jacobsen, Matthew Schlegel, Cameron Linke, Thomas Degris, Adam White, and Martha White. Meta-Descent for Online, Continual Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3943–3950, 2019.

Xiaowen Jiang and Sebastian U Stich. Adaptive SGD with Polyak stepsize and Line-search: Robust Convergence and Variance Reduction. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2023.

Scott M. Jordan, Samuel Neumann, James E. Kostas, Adam White, and Philip S. Thomas. The Cliff of Overcommitment with Policy Gradient Step Sizes. *Reinforcement Learning Journal*, 2024.

Alex Kearney, Vivek Veeriah, Jaden B. Travnik, Richard S. Sutton, and Patrick M. Pilarski. TIDBD: Adapting Temporal-difference Step-sizes Through Stochastic Meta-descent, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Improving Generalization in Meta Reinforcement Learning using Learned Objectives. In *International Conference on Learning Representations*, 2020.

Qingfeng Lan, A. Rupam Mahmood, Shuicheng Yan, and Zhongwen Xu. Learning to Optimize for Reinforcement Learning. *Reinforcement Learning Journal*, 2024.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Michael Lu, Matin Aghaei, Anant Raj, and Sharan Vaswani. Towards principled, practical policy gradient for bandits and tabular mdps. *arXiv preprint arXiv:2405.13136*, 2024.

Ashique Rupam Mahmood, Richard S. Sutton, Thomas Degris, and Patrick M. Pilarski. Tuning-Free Step-Size Adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

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 *International Conference on Machine Learning*, 2024.

Golnaz Mesbahi, Parham Panahi, Olya Mastikhina, Steven Tang, Martha White, and Adam White. Position: Lifetime Tuning is Incompatible with Continual Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*, 2025.

Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. In *Proceedings of the International Conference on Machine Learning*, 2024.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2nd edition, 2006. ISBN 978-0-387-30303-1.

Junhyuk Oh, Matteo Hessel, Wojciech M. Czarnecki, Zhongwen Xu, Hado P. van Hasselt, Satinder Singh, and David Silver. Discovering Reinforcement Learning Algorithms. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2020.

Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2020.

Eduardo Pignatelli, Jarek Liesen, Robert Lange, Chris Lu, Pablo Samuel Castro, and Laura Toni. Navix: Scaling minigrid environments with jax. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Proceedings of the Internation Conference on Neural Information Processing Systems*, 2025.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

Sashank Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. In *Proceedings of the International Conference on Learning Representations*, 2018.

Vincent Roulet, Atish Agarwala, and Fabian Pedregosa. On the Interplay Between Stepsize Tuning and Progressive Sharpening. *OPT2023 Workshop on Optimization for Machine Learning*, 2023.

Vincent Roulet, Atish Agarwala, Jean-Bastien Grill, Grzegorz Swirszcz, Mathieu Blondel, and Fabian Pedregosa. Stepping on the Edge: Curvature Aware Learning Rate Tuners. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2024.

N.N. Schraudolph. Local Gain Adaptation in Stochastic Gradient Descent. In *International Conference on Artificial Neural Networks*, 1999.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *Proceedings of the International Conference on Machine Learning*, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017.

Betty Shea and Mark Schmidt. Why Line Search when you can Plane Search? SO-Friendly Neural Networks allow Per-Iteration Optimization of Learning and Momentum Rates for Every Layer, 2024.

Richard S. Sutton. Adapting Bias by Gradient Descent: an Incremental Version of Delta-Bar-Delta. In *National Conference on Artificial Intelligence*, 1992.

Markus Thill. Temporal Difference Learning Methods with Automatic Step-Size Adaptation for Strategic Board Games: Connect-4 and Dots-and-Boxes, 2015. Cologne University of Applied Sciences, Master's Thesis.

Sharan Vaswani and Reza Babanezhad. Armijo Line-search Can Make (Stochastic) Gradient Descent Provably Faster. In *International Conference of Machine Learning*, 2025.

Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates. In *Advances in Neural Information Processing Systems*, 2019.

Sharan Vaswani, Issam Laradji, Frederik Kunstner, Si Yi Meng, Mark Schmidt, and Simon Lacoste-Julien. Adaptive Gradient Methods Converge Faster with Over-Parameterization (But You Should do a Line-Search). *Annual Workshop on Optimization for Machine Learning*, 2020.

Yufei Wang and Tianwei Ni. Meta-SAC: Auto-Tune the Entropy Temperature of Soft Actor-Critic via Metagradient. *ICML Workshop on Automated Machine Learning*, 2020.

Zhanghao Wu, Eric Liang, Michael Luo, Sven Mika, Joseph E. Gonzalez, and Ion Stoica. RLlib flow: Distributed reinforcement learning is a dataflow problem. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2021.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 31, 2018.

Zhongwen Xu, Hado P van Hasselt, Matteo Hessel, Junhyuk Oh, Satinder Singh, and David Silver. Meta-Gradient Reinforcement Learning with an Objective Discovered Online. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2020.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Kenny Young, Baoxiang Wang, and Matthew E. Taylor. Metatrace Actor-Critic: Online Step-Size Tuning by Meta-gradient Descent for Reinforcement Learning Control. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado P. van Hasselt, David Silver, and Satinder Singh. A Self-Tuning Actor-Critic Algorithm. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2020.

Hongchao Zhang and William W. Hager. A Nonmonotone Line Search Technique and Its Application to Unconstrained Optimization. *SIAM Journal on Optimization*, 2004.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On Learning Intrinsic Rewards for Policy Gradient Methods. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*, 2018.

Zeyu Zheng, Junhyuk Oh, Matteo Hessel, Zhongwen Xu, Manuel Kroiss, Hado Van Hasselt, David Silver, and Satinder Singh. What Can Learned Intrinsic Rewards Capture? In *Proceedings of the International Conference on Machine Learning*, 2020.

A Runtime Comparison Across Methods

Table 2 shows the steps per second for all evaluated methods remained within 2% of the default baseline. When using Adam3, we evaluate the Armijo condition for each candidate step size in parallel using the GPU. Adam3 therefore incurs negligible runtime overhead as compared to default fixed step size methods.

Table 2: **Training throughput** (normalized environment steps per second relative to Default, higher is better) on Brax environments. Results are reported as mean ± s.d. over 10 trials per environment. The **Mean** column averages across all five environments and uses the s.d. across the environment means. To control for hardware noise, all five methods were run back-to-back in rotated order on the same NVIDIA L40S GPU with one CPU core per trial, and each method's throughput is normalized by the Default run from that same trial (each trial runs PPO for 100,000 steps with a fixed seed for each method).

Method	**Mean**	Hopper	Walker2d	Ant	Humanoid	HalfCheetah
Default	**1.000**	1.000	1.000	1.000	1.000	1.000
Adam-B1024	**1.012 ± 0.006**	1.017 ± 0.001	1.018 ± 0.001	1.006 ± 0.001	1.009 ± 0.001	1.008 ± 0.001
Adam3	**1.011 ± 0.007**	1.021 ± 0.001	1.016 ± 0.001	1.004 ± 0.001	1.006 ± 0.001	1.007 ± 0.001
Hypergradient	**0.991 ± 0.003**	0.986 ± 0.001	0.991 ± 0.001	0.992 ± 0.001	0.995 ± 0.001	0.992 ± 0.001
Prodigy	**0.997 ± 0.004**	0.990 ± 0.001	0.998 ± 0.001	0.999 ± 0.001	0.997 ± 0.001	1.000 ± 0.001

B Further Implementation Details and Hyperparameters

In this section, we provide further details on implementation and hyperparameters.

Hyperparameters and Implementation Details Our PPO implementation and hyperparameters follow the best practices described by Huang et al. (2022). We did not parallelize across environments, instead gathering data from a single environment. Rollout trajectories consisted of 2048 transitions, updating over 10 epochs across these transitions with minibatches of size 64 or 1024, as further detailed in the main paper. We set $\gamma = 0.99$ and did not bootstrap across episode truncations. We set the generalized advantage estimate coefficient $\lambda = 0.95$ and did not use entropy regularization. We set Adam's epsilon parameter as $\varepsilon_{\text{adam}} = 10^{-5}$, leaving Adam's other hyperparameters at their defaults. Following the recommendation of Andrychowicz et al. (2021), we did not use learning rate annealing for PPO when using fixed step sizes. We set PPO's clipping coefficient to $\epsilon = 0.2$ following standard practice. We normalized rewards and observations using the method in standard PPO implementations (Huang et al., 2022). We used a separate neural network for actor and critic, each consisting of two hidden layers of hyperbolic tangent activations and 64 hidden units. Weights were initialized using orthogonal initialization. Bias units were initialized to 0. All other hyperparameters were set to the defaults from Stable-Baselines3 (Raffin et al., 2021) unless otherwise noted.

Normalization Procedure In a number of cases, we report aggregate performance of algorithms across environments. In order to make such an aggregate meaningful, we normalized performance within each environment before aggregating. To determine a method's aggregate performance across a suite, we first normalize the mean episodic returns of all runs within each environment using Winsorized Min-Max Normalization (5^{th} and 95^{th} percentiles). For each method, we average these normalized scores across its runs per environment, and then average across all environments to yield the overall mean normalized performance. We use the 5^{th} and 95^{th} percentiles to prevent outliers from distorting the min-max normalization; this does not alter the set of runs used in our analysis. We report 95% confidence intervals for this metric using a stratified bootstrap with 10,000 resamples.

Environments Our empirical study revolved around a few environments from popular benchmark suites. On Brax (Freeman et al., 2021), we used the Hopper, Walker2D, Ant, HalfCheetah, and Humanoid environments. On Jumanji (Bonnet et al., 2024), we used the Snake-v1, Knapsack-v1, Game2048-v1, and Graphcoloring-v1 environments. From MinAtar (Young & Tian, 2019), we used

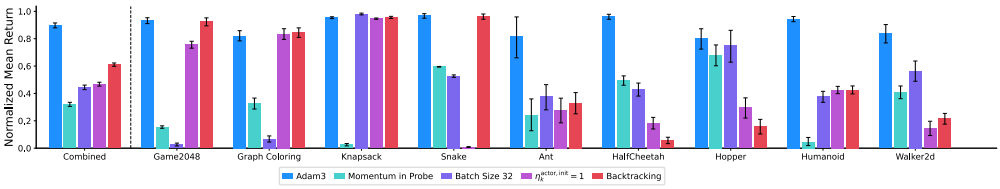


Figure 7: Mean normalized performance with 95% bootstrap confidence intervals across ablations on Brax and Jumanji.

the Freeway, Breakout, Asterix, Seaquest, and Space Invaders environments. On Navix-MiniGrid (Pignatelli et al., 2025), we used the DistShift1, DoorKey-8x, Dynamic-Obstacles-8x8, Empty-8x8, Empty-Random-8x8, FourRooms, KeyCorridorS3R1, LavaGapS7, and SimpleCrossingS9N1 environments. On Bsuite (Osband et al., 2020), we used the Cartpole/0, Cartpole Swingup/0, Catch/0, Deep Sea/0, Discounting Chain/1, Mountain Car/0, Umbrella Distract/22, and Umbrella Length/9 environments. On Craftax (Matthews et al., 2024), we used the Classic-Symbolic environment.

C Additional Experimental Results

In this section, we present additional experiments and results.

C.1 Ablations

Figure 7 provides a finer-grained view of the results of our ablation study in Section 4.2, on a per-environment basis. We tested four ablations: (1) using a momentum-uncorrected direction in the line search procedure (**teal**); (2) using a small batch size of 64, rather than 1024 (**purple**); (3) starting the actor line search from $\eta_k^{\text{init,actor}} = \eta^{\text{init,actor}} = 1$ (**pink**); and (4) selecting step sizes using backtracking, rather than our loss-minimizing procedure in Equation 6 (**red**). Further experimental details are provided in Section 4.2.

Overall, we found that using a momentum-uncorrected direction in the line search was a poor strategy across both Jumanji and Brax suites. Using the smaller batch size of 32 was often detrimental to learning. Starting the actor line search from a step size of $\eta^{\text{init,actor}} = 1$ sometimes performed well, but could catastrophically fail, as in the case of Snake. Using a backtracking procedure for step size selection overall performed worse than a loss-minimizing approach.

C.2 Sharpness

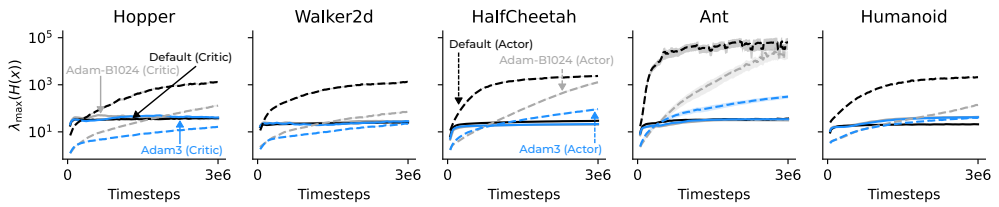


Figure 8: Raw maximum loss hessian eigenvalues across the Brax environments. We found little evidence of sharpness issues in the control setting.

As outlined in Section 4.2, we did not find evidence of line searches inducing catastrophic progressive sharpening, as has been reported in the supervised learning literature (Roulet et al., 2023; 2024). Figure 8 shows the maximum eigenvalues for the objective (non-preconditioned) Hessian for both actor and critic on Brax. Figure 9 shows the maximum eigenvalues for both preconditioned and non-preconditioned objective Hessians for both actor and critic on Jumanji. In all cases, we did not find evidence of catastrophic progressive sharpening for the Adam3 line search method, in stark contrast to what has been reported in supervised learning (Roulet et al., 2023; 2024). In fact, we often see

that the Adam3 line search method attains lower sharpness than the fixed step size baseline. This perhaps provides evidence that a major failure case of line searches in supervised learning may not be applicable to reinforcement learning control.

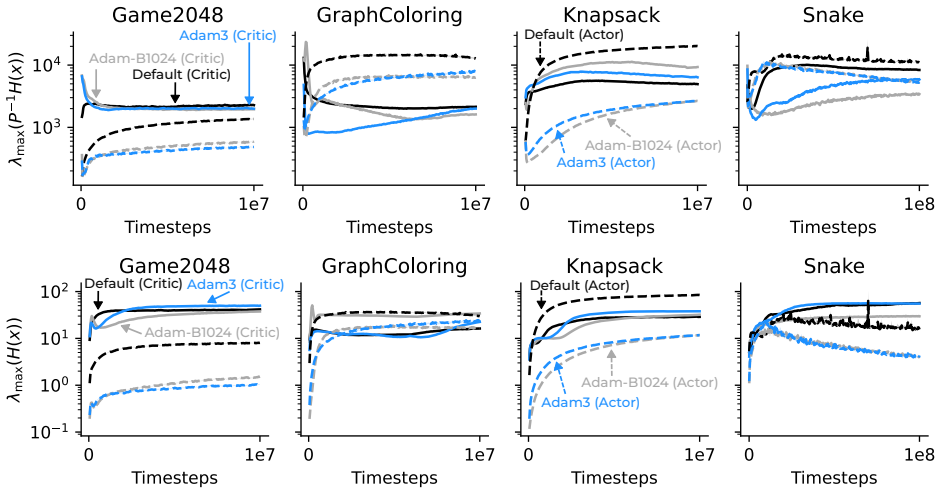


Figure 9: **(top)** Preconditioned and **(bottom)** raw maximum loss hessian eigenvalues across the Jumanji environments. We found little evidence of sharpness issues in the control setting.

C.3 Sensitivity Study

In this section, we empirically characterize the sensitivity of Adam3 to its hyperparameters, namely c, the contraction factor δ, and the number of backtracking steps B. We focus on characterizing hyperparameter sensitivity on Brax and Jumanji using the same experiment setup as outlined in Section 4.1. We evaluated the Armijo constant $c \in \{10^{-8}, 0.1, 0.5, 0.7, 0.99\}$. We coupled the sensitivity study of δ and B, as they control the effective step size range that the line search considers. We varied these hyperparameters so that the effective step size range was similar to that used in the main paper, varying $(\delta, B) \in \{(0.1, 6), (0.5, 16), (0.7, 30)\}$.

Figure 11 shows the sensitivity of Adam3 to the Armijo constant c across both Jumanji and Brax (3M and 50M timesteps). We found relative insensitivity to c across these environments. Figure 10 show the sensitivity of Adam3 to the number of backtracking steps B, coupled with the contraction factor δ. The figure also demonstrates relative insensitivity to these hyperparameters and indicates that a very coarse line search is sufficient to attain reasonable performance across these environment suites.

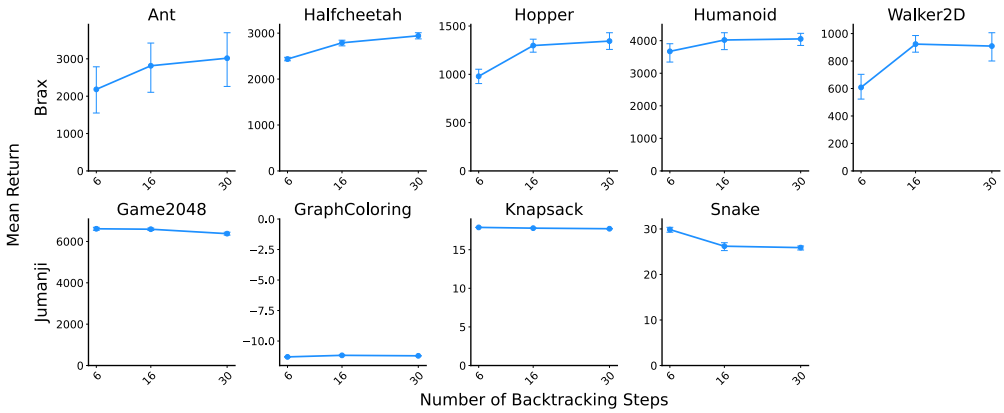


Figure 10: Sensitivity to the number of backtracking steps B with adjusted contraction factor δ to maintain equal range of examined step size on **(top)** Brax (3M steps) and **(bottom)** Jumanji.

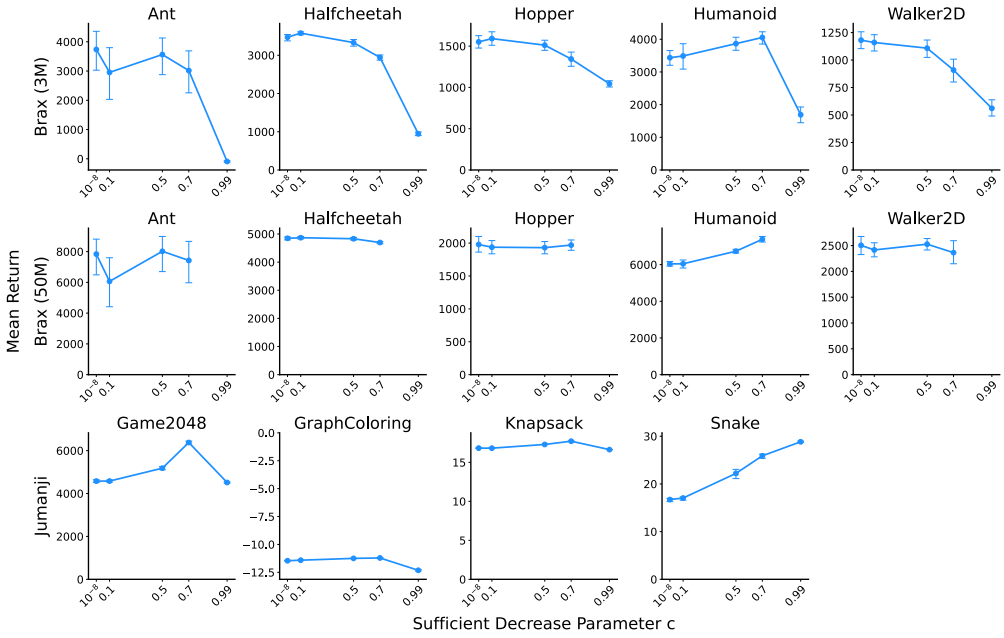


Figure 11: Sensitivity to the Armijo sufficient decrease parameter c on **(top)** Brax (3M steps), **(middle)** Brax (50M steps), and **(bottom)** Jumanji.

C.4 Additional Learning Curves

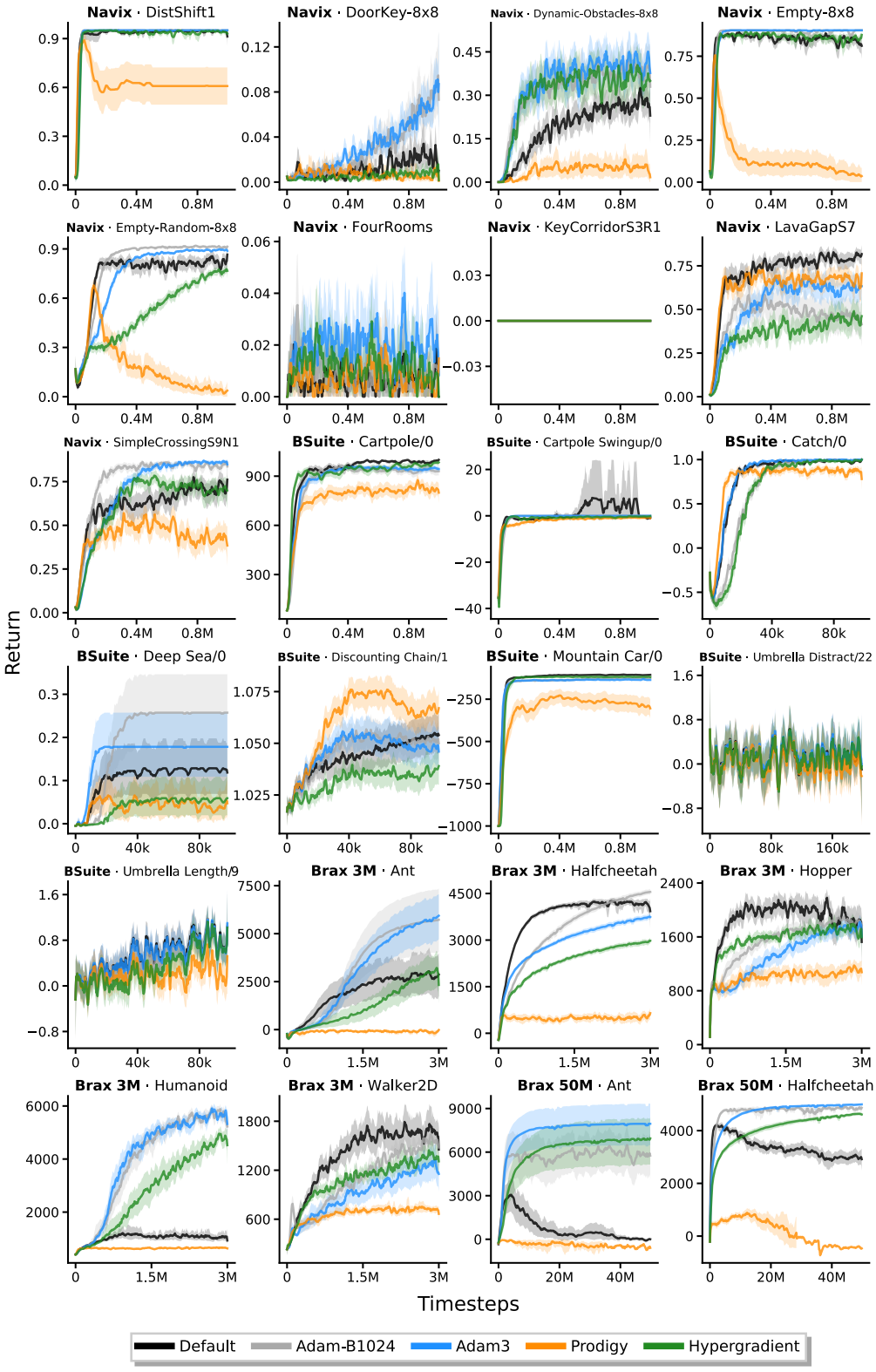


Figure 12: Learning curves for all environments in all examined suites.

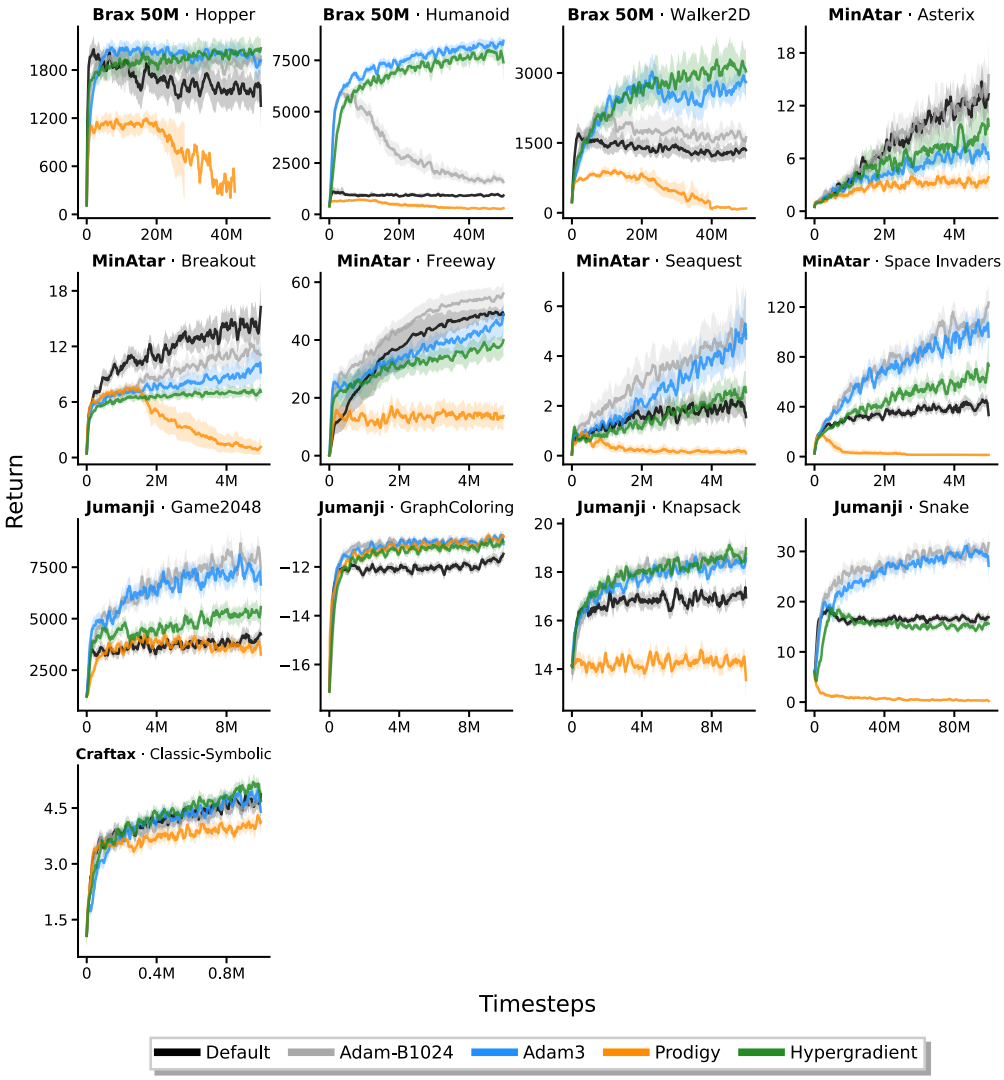


Figure 13: Learning curves for all environments in all examined suites (cont.)

C.5 Step Sizes

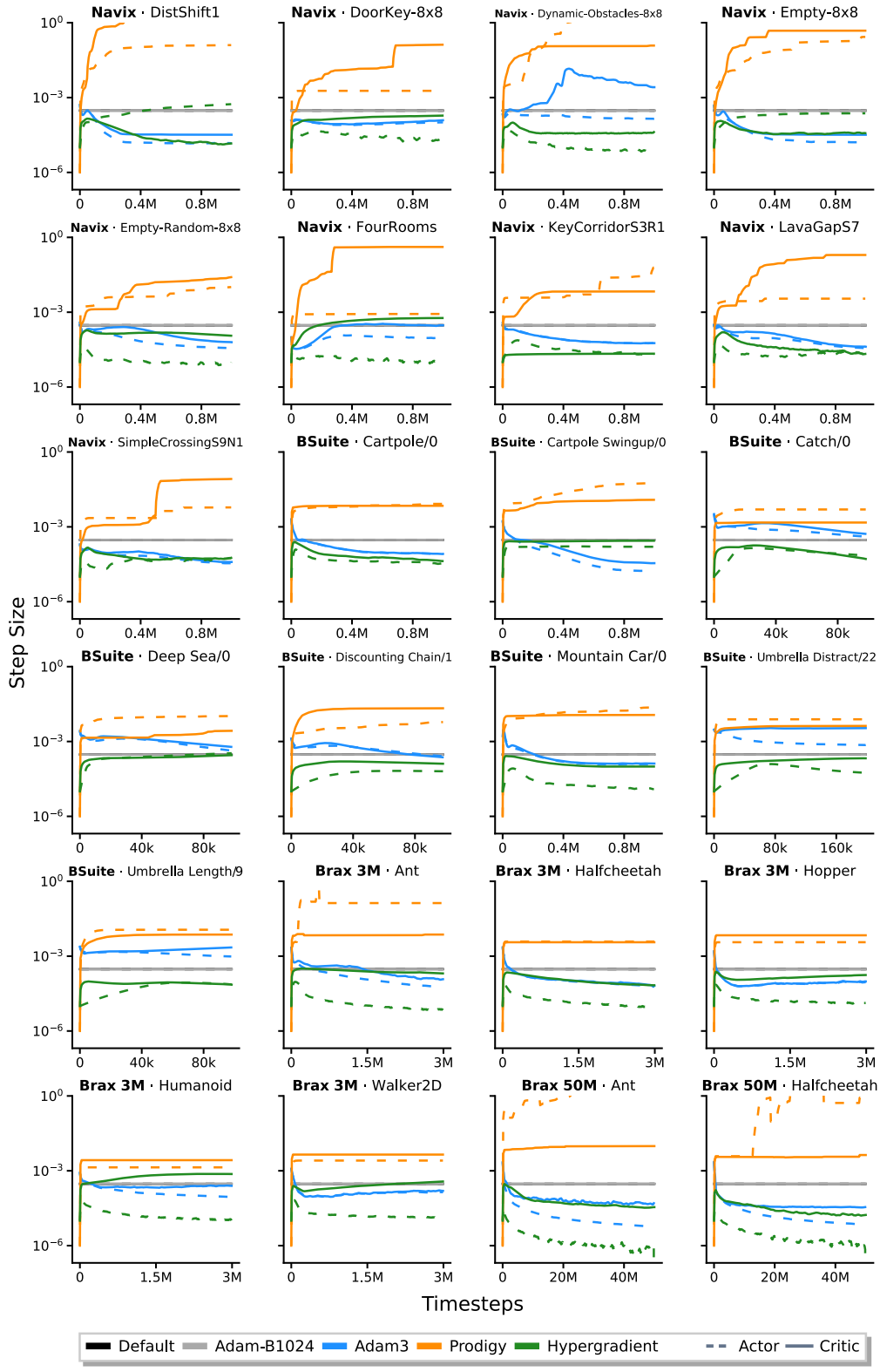


Figure 14: Actor and critic step size for all environments in all examined suites

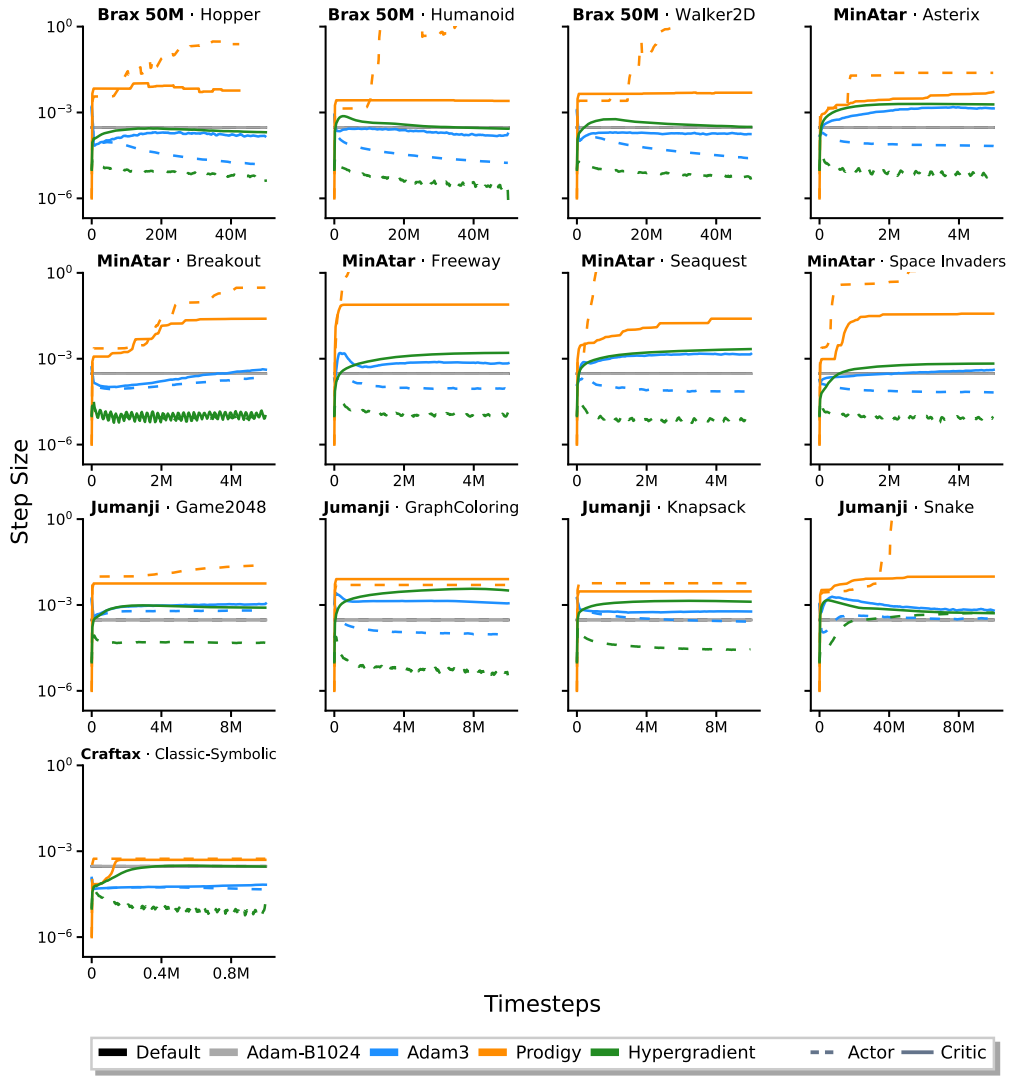


Figure 15: Actor and critic step size for all environments in all examined suites (cont.)