Colored Noise in PPO: Improved Exploration and Performance Through Correlated Action Sampling

Jakob Hollenstein[1,3] **Georg Martius**[3,4] **Justus Piater**[1,2]
[1]Department of Computer Science, Universität Innsbruck, Innsbruck, Austria
[2]Digital Science Center, Universität Innsbruck, Innsbruck, Austria
[3] Max Planck Institute for Intelligent Systems, Tübingen, Germany
[4] Department of Computer Science, Eberhard Karls University, Tübingen, Germany
`{jakob.hollenstein,justus.piater}@uibk.ac.at`
`georg.martius@uni-tuebingen.de`

Abstract

Proximal Policy Optimization (PPO), a popular on-policy deep reinforcement learning method, employs a stochastic policy for exploration. In this paper, we propose a colored noise-based stochastic policy variant of PPO. Previous research highlighted the importance of temporal correlation in action noise for effective exploration in *off-policy* reinforcement learning. Building on this, we investigate whether correlated noise can also enhance exploration in *on-policy* methods like PPO. We discovered that correlated noise for action selection improves learning performance and outperforms the currently popular uncorrelated white noise approach in on-policy methods. Unlike off-policy learning, where pink noise was found to be highly effective, we found that a colored noise, intermediate between white and pink, performed best for on-policy learning in PPO. We examined the impact of varying the amount of data collected for each update by modifying the number of parallel simulation environments for data collection and observed that with a larger number of parallel environments, more strongly correlated noise is beneficial. Due to the significant impact and ease of implementation, we recommend switching to correlated noise as the default noise source in PPO.

1 Introduction

Exploration plays a crucial role in deep reinforcement learning, particularly in continuous action space applications like robotics, where the number of states and actions is infinite. In the continuous action space setting, exploration is typically achieved by introducing variability around the mean action proposed by the policy. For instance, deterministic off-policy algorithms such as TD3 [6] and DDPG [11] use additive action noise, while stochastic off-policy algorithms such as SAC [7] and MPO [1] employ a Gaussian distributed policy to sample actions and induce variation. On-policy algorithms such as TRPO [18] and PPO [19] follow a similar approach by sampling variations from a Gaussian distribution.

While previous research [9, 4, 17] has emphasized the importance of temporal correlation in action noise for exploration in off-policy algorithms, on-policy algorithms such as PPO do not incorporate correlated action noise and instead sample uncorrelated variations from a Gaussian distributed policy.

PPO, as an on-policy deep reinforcement learning algorithm, offers several advantages over off-policy algorithms: by not relying on replay buffers, on-policy algorithms exhibit fewer instabilities due to distributional shift between the distribution of previously collected data and the current policy-induced distribution. Moreover, on-policy algorithms mitigate Q-function divergence issues since the policy evaluation can leverage data collected under the current policy. These benefits make on-policy methods particularly advantageous when environment samples are inexpensive to obtain. However, exploration in on-policy methods needs to be induced by changes to the environment (exploration reward) or by changes to the policy itself, as, by definition, on-policy methods learn from data induced by the current policy. Consequently, altering the exploration technique without changing the environment implies a change in the policy distribution.

Contributions In this paper, we introduce and empirically evaluate a modification to PPO incorporating temporally correlated colored noise into the stochastic policy's distribution. Utilizing the re-parameterization trick, we maintain a Gaussian distributed behavior while injecting correlated noise instead of uncorrelated noise. The correlated noise is parameterized by its color β.

Our experiments demonstrate that adopting colored noise enhances performance in the majority of tested environments. Surprisingly and in contrast to previous research on *off-policy* methods, where Eberhard et al. [4] found pink noise to perform best, we found the optimal colored noise for improved performance — for the *on-policy* method PPO — to lie *between* white and pink noise ($\beta = 0.5$, see Sec. 2). Furthermore, as the number of parallel data collection environments increases, we observe a trend towards more correlated noise. However, for the benchmarks considered in this work, our results indicate that utilizing about four parallel environments, resulting in 8192 samples per update step (see Sec. H for the relation of N_{steps} and N_{env}), together with noise in between white and pink noise, is the most efficient approach.

1.1 Related Work

Reinforcement learning is a promising technique for solving complex control problems. An important milestone for the development of deep reinforcement learning was the work of Mnih et al. [12] showing human level performance of reinforcement learning on the Atari Games benchmark. In contrast to the Atari Games benchmarks which has complicated observation spaces but discrete action spaces, reinforcement learning for robotic applications has to deal with continuous action spaces, for example in the benchmarks provided by Tassa et al. [20], where the task is to control simulated robots. Because the action selection process is more complicated in continuous spaces, policy search methods [23] are favored. This includes off-policy methods such as DDPG [11], TD3 [6], MPO [1] or SAC [7] which can learn from arbitrary data and on-policy methods such as TRPO [18] and PPO [19] which iteratively improve the policy using data collected by the current iteration of the policy. On-policy methods exhibit better convergence properties and behave more stably but require larger amounts of training data.

Exploration, the problem of discovering better action sequences, is of pivotal importance in RL and is tackled in many ways. Amin et al. [2], Ladosz et al. [10] and Yang et al. [24] provide recent surveys of the exploration research landscape in deep reinforcement learning. In this work, we turn to the simple yet effective method which forms the backbone of most RL algorithms: undirected noisy exploration.

In off-policy methods, exploration is often achieved by perturbing the action selection process, for example by perturbing the parameters of the policy [15] or by additively perturbing the action, i.e., by adding *action noise*. This action perturbation can be done, for example, using temporally uncorrelated Gaussian noise, or temporally correlated noise generated by an Ornstein-Uhlenbeck process [22]. Previous research has shown that temporal correlation of actions, i.e., Gaussian noise vs. Ornstein-Uhlenbeck noise can be beneficial or harmful for state-space coverage and policy learning, but that this depends on the task environment and its dynamics [8, 9]. Another method for temporally correlated perturbations was introduced by Raffin et al. [17] where the “action noise” is generated deterministically by a function only dependent on the state, but the function parameters are randomly changed after a number of steps.

While different environments respond differently to the temporal correlation of noise, Eberhard et al. [4] found *pink noise* to act as a kind of middle ground between uncorrelated white Gaussian noise, and correlated Brownian-motion-like noise, e.g., red noise and Ornstein-Uhlenbeck noise. This middle ground, while not always the perfect choice, was found to be a much better default choice.

Petrazzini and Antonelo [14] propose to improve exploration by changing the policy distribution of PPO from a Gaussian distribution to a Beta distribution. Similarly, we propose to change the policy distribution of PPO, but instead of a Beta distribution we propose to keep the Gaussian distribution and bias the sampling to be temporally correlated using a colored-noise process.

Algorithmic details and hyperparameter choices are known to be important for PPO and were empirically analyzed by Andrychowicz et al. [3] and Engstrom et al. [5]. Similarly, we performed a comprehensive empirical evaluation of the impact of different noise correlation settings and the setting of parallel data collection. From this evaluation, we recommend switching the default noise process to a correlated noise process with $\beta = 0.5$.

2 Method

2.1 Colored Noise

Colored noise is a class of noise that exhibits temporal correlation and is characterized by a change of $\frac{1}{f^\beta}$ in its power spectral density (PSD) components, where f denotes the frequency and β determines the "color" of the noise. The power spectral density (PSD) of a sequence consists of the squared magnitude of the frequency components of the sequence's Fourier transform. Given sequences of noise samples $\tau_\varepsilon^{(i)} = \{\varepsilon_1^{(i)}, \dots \varepsilon_T^{(i)}\}$, and the power spectral density of each sequence $|\mathcal{F}(\tau_\varepsilon^{(i)})|^2$, the *expected PSD* is calculated by averaging over each trajectory's PSD: $\mathbb{E}_i[|\mathcal{F}(\tau_\varepsilon^{(i)})|^2]$.

Uncorrelated noise is called *white noise* and has a constant, flat line, expected power spectral density ($\beta = 0$). The expected PSD of *pink noise* decreases with $1/f$ ($\beta = 1$) and for *red noise* the decrease happens at a rate of $1/f^2$ ($\beta = 2$).

The effect of sampling actions from colored noise processes of different colors can be exemplified by a velocity controlled robot that can take steps of various sizes in the x and y directions. The paths taken by the robot, when the steps are controlled by noise, depend on the correlation of the noise. The position (x_t, y_t) of the robot is the result of integrating over the noise sequence. Examples of such integrated sequences, *random walks*, for colored noises of different β coefficients are shown in Figure 1. Brownian motion, for example, exhibits red noise ($\beta = 2$) characteristics and is itself the result of integrating white noise ($\beta = 0$) over time.

Colored noise can be generated in the time domain by drawing white noise samples and applying an autoregressive filter or directly in the frequency domain by sampling the frequency components accordingly. In our work we use the latter approach, by building on an implementation by Patzelt [13] and adapted by Eberhard et al. [4] following an algorithm by Timmer and König [21]. We generate colored noise sequences of length τ: $\tau_\varepsilon = \{\varepsilon_1, \dots, \varepsilon_\tau\}$. Sampling action noise is then implemented as consuming the items from this sequence. Sec. C for further details.

2.2 PPO

PPO is a popular on-policy reinforcement learning algorithm that utilizes a policy gradient approach to optimize the policy. To avoid learning instabilities, PPO aims to prevent large policy updates. As an on-policy algorithm, PPO collects trajectories by interacting with the environment using the

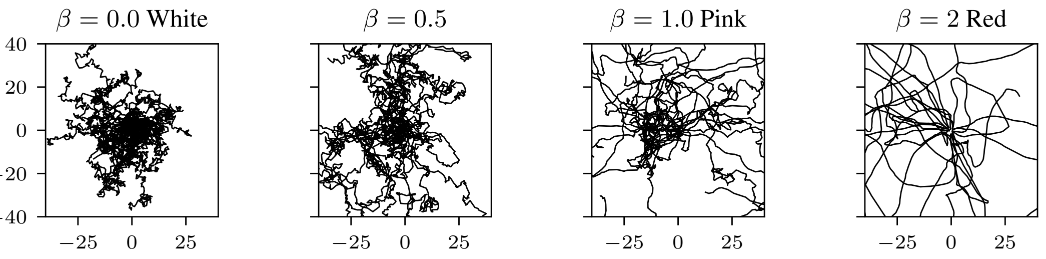


Figure 1: Two-dimensional random walks caused by colored noise of different β. Lower β values cause more energy in high frequency parts of the power spectral density, causing the random walk to change direction more frequently and thus causing more local and less global exploration. Higher values of β result in more energy in the lower frequencies of the power spectral density. This translates to random walks that change direction less frequently, and thus explore more globally.

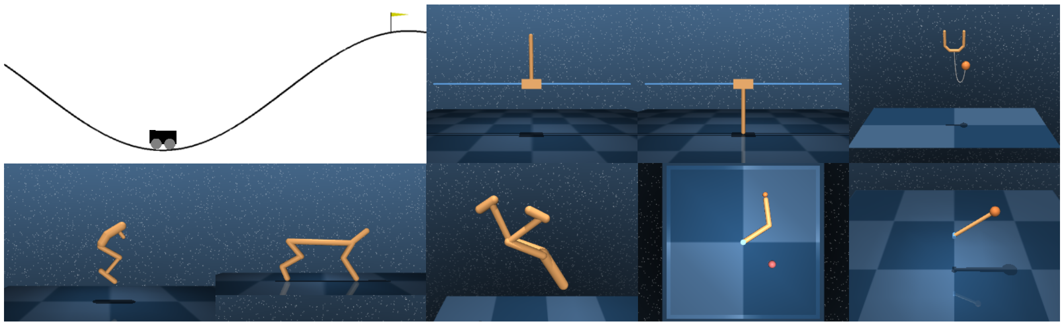


Figure 2: Benchmark environments: (top) Mountain Car, Cartpole Balance, Cartpole Swingup, Ball in Cup (Catch), (bottom) Hopper Hop, Cheetah Run, Walker Run, Reacher Hard, Pendulum Swingup,

current policy. Advantages are then computed for these data and the policy is updated. For the next policy improvement, new data is collected. The number of data points collected for each update is a hyperparameter.

The action selection process uses a policy network that outputs the mean (μ_t) and standard deviation (σ_t) of a Gaussian distribution. An action is then sampled from this distribution:

$$a_t \sim \mathcal{N}(\mu_t, \sigma_t). \tag{1}$$

Using the re-parameterization trick, this is re-written:

$$a_t = \mu_t + \varepsilon_t \cdot \sigma_t; \quad \varepsilon_t \sim \mathcal{N}(0, 1). \tag{2}$$

Instead of sampling ε_t from a white noise Gaussian process, the default in PPO, we propose to use a colored noise Gaussian process, similar to the approach of Eberhard et al. [4] for *off-policy* methods.

The generated noise samples $\varepsilon_t^{(i)}$ still show Gaussian distributions at each time step (Figure A.1b). Because we modify the ε in $\mu + \sigma \cdot \varepsilon$ and ε_t remains Gaussian, the data collection, viewed at each individual step, remains asymptotically on-policy. However, while the marginal stays Gaussian, the two-step correlation of ε_t vs. ε_{t+1} changes with β (Figure A.1a). In this paper, we empirically evaluate the impact of this correlation due to the use of colored noise.

3 Experiments

We perform an empirical evaluation of the impact of colored noise on the performance of PPO. To this end we perform training runs, repeated with 20 independent seeds, on the benchmarks Ball in Cup (Catch), Cartpole Balance, Cartpole Swingup, Cheetah Run, Hopper Hop, Mountain Car, Pendulum Swingup, Reacher Hard and Walker Run. We vary the noise color $\beta \in \{-1$ (blue), 0 (white), 0.2, 0.5, 0.75, 1 (pink), 1.25, 2 (red) $\}$ and test different numbers of parallel collection environments $N_{\text{envs}} \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. This results in a total of 11520 experiments. See Sec. E for further details.

3.1 Evaluation Details

We train each agent for a total of 2 048 000 time steps and evaluate every 10240 steps, resulting in 200 evaluation points. At each evaluation point, 50 evaluation episodes are performed. We average the mean returns collected at each evaluation point, forming an estimate that captures the area under the learning curve, and refer to this as the *performance*. Evaluation results are grouped by environment and standardized to zero mean and unit variance to control for the impact of the environments. We combine these standardized results by averaging across different environments and seeds.

3.1.1 Does Colored-Noise Affect the Performance of PPO?

Adding action-noise to the action selection process creates a divergence between the action distribution of the policy and the actual action distribution, i.e., it induces a difference between the distribution of the data the undisturbed policy would collect, compared to the policy disturbed by additive action

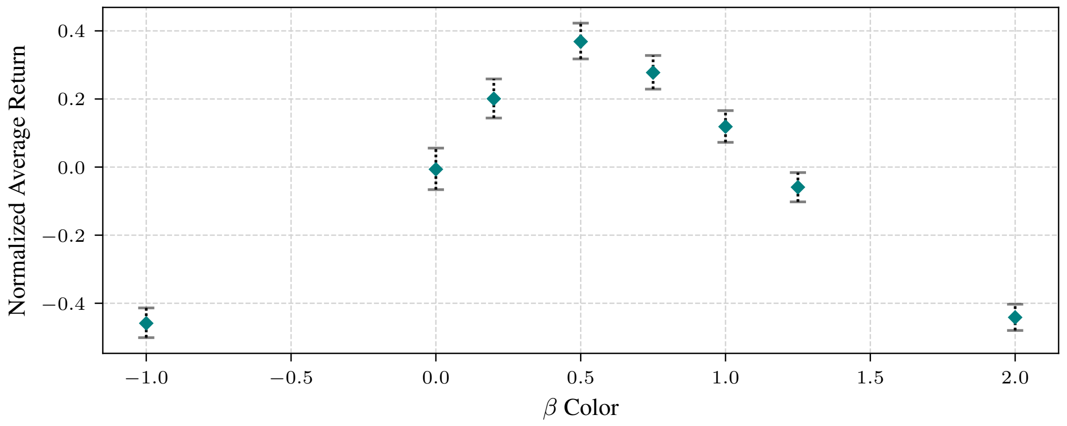


Figure 3: Performance averaged across Environments: Correlated noise $\beta = 0.5$ significantly outperforms the default white noise ($\beta = 0$, Sec. F) used by PPO. The bars indicate the 95% bootstrapped confidence intervals.

noise. Similarly, modifying the action selection process, by changing ε in (2), can potentially break the learning process, as on-policy algorithms operate under the assumption that the collected data match the distribution induced by the policy.

In this experiment, we compare how the learning performance (area under the learning curve, averaged across environments, Sec. 3.1) of PPO reacts to a modified action selection process: to the use of correlated noise ($\beta \neq 0$) as compared to the standard case of using uncorrelated noise ($\beta = 0$). Figure 3 shows the performance averaged across the different environments and across the different numbers of parallel data collection environments. Results for each environment are standardized to control for differences in reward scale. The mean performance, and 95% confidence intervals of the mean, are depicted. The confidence intervals are estimated using (bias-corrected and accelerated) bootstrapping. The results indicate a significant effect of the correlation of the noise. This is in line with previous findings highlighting an impact of action-noise correlation on learned performance [9, 4]. The results depicted in Figure 3 show an overall preference for colored noise at $\beta = 0.5$, which lies between white noise ($\beta = 0$) and pink noise ($\beta = 1$). This contrasts with the results in off-policy learning, where $\beta = 1$ was found to be superior across many environments [4]. On-policy methods assume matching state-visitation frequencies between the collected data and what the current policy would induce. More correlated noise induces larger state space coverage [9] and thus a larger deviation from the states the deterministic-mean-action policy would visit. Thus, a potential reason for this difference in noise color preference is that increasing the state-space coverage, by increasing the noise correlation, increases the distributional shift between the data and policy-induced marginal state-visitation frequencies.

In summary, we found colored noise to have a significant impact on learned policy performance averaged across environments, with $\beta = 0.5$ performing best.

3.1.2 Is $\beta = 0.5$ a Better Default for PPO?

The default stochastic policy for PPO relies on uncorrelated noise sampled from a Gaussian distribution. The results shown in Figure 3 indicate that across environments $\beta = 0.5$ performs best. Previous research found an environment specific response to action noise correlation for purely noisy policies [8], as well as learning performance in off-policy methods [9, 4]. In accordance with these earlier results, we found the best β^* to vary depending on the environment. Table 1 lists the best performing β^* for each environment. In addition, we perform Welch t-tests to compare the performance of β^* to each $\beta \in \{0.5, 0.0, 1.0\}$. This shows that $\beta = 0.5$ performs comparable to β^* in $5/9$ environments. In contrast, the current default $\beta = 0$ is significantly outperformed by β^* in six environments and only performs comparable in $3/9$ environments. The best default found for off-policy learning, pink noise ($\beta = 1.0$), is significantly outperformed in five out of nine cases and performs comparable in $4/9$ environments. *We therefore recommend switching the default from temporally uncorrelated noise $\beta = 0$ to temporally correlated noise $\beta = 0.5$.*

Environment	β^*	Noise Color	$\beta = 0.5$	$p_{0.5}$	$\beta = 0.0$	$p_{0.0}$	$\beta = 1.0$	$p_{1.0}$
Cartpole Balance	0.00	White	✗	0.04	✓	–	✗	0.01
Cheetah Run	0.00	White	✓	0.14	✓	–	✗	0.01
Reacher Hard	0.00	White	✗	0.01	✓	–	✗	0.01
Cartpole Swingup	0.50		✓	–	✗	0.03	✗	0.01
Ball in Cup (Catch)	0.75		✓	0.90	✗	0.01	✓	0.96
Hopper Hop	0.75		✓	0.68	✗	0.01	✓	0.09
Pendulum Swingup	0.75		✓	0.92	✗	0.04	✗	0.03
Walker Run	1.00	Pink	✗	0.01	✗	0.01	✓	–
Mountain Car	1.25		✗	0.01	✗	0.01	✓	0.45

Table 1: Optimal noise color per environment. ✓ vs. ✗ indicate whether each fixed $\beta \in \{0.5, 0.0, 1.0\}$ performs comparable to β^* (✓) or is significantly (Welch t-test) outperformed by β^* (✗). The p-values are listed in $p_{0.5}, p_{0.0}$ and $p_{1.0}$. $\beta = 0.5$ performs favorable (✓$5/9$) and improves over the current default of white noise $\beta = 0$ (✓$3/9$)

3.1.3 How Does the Number of Parallel Collection Environments Affect the Performance?

On-policy methods are less efficient with respect to environment interaction samples compared to off-policy methods. This makes on-policy methods particularly interesting when environment samples are cheap to collect and can be collected in parallel. We collected environment samples with different numbers of parallel environments $N_{\text{envs}} \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ and kept the number of total environment interactions (total time steps) constant across experiments. With each of the parallel environments, 2048 samples are collected as the dataset for each update cycle. Thus, depending on the number of environments, more (or less) samples are used for each update. Because the total number of samples is limited (Sec. 3.1) this implies fewer (or more) updates in total. Figure 4 indicates that a larger number of parallel environments negatively affects performance. This is in line with findings by Andrychowicz et al. [3], who also found that the most beneficial number of parallel environments is dependent on the environment type. In our study, we find that about four parallel environments are preferable. This translates to a preference for about $2048 \cdot 4 = 8192$ samples in each update if the episode lengths are small enough to fit the 2048 step limit. In summary, we found 8192 samples per update, collected by four environments in parallel, to perform most efficiently for the investigated class of tasks.

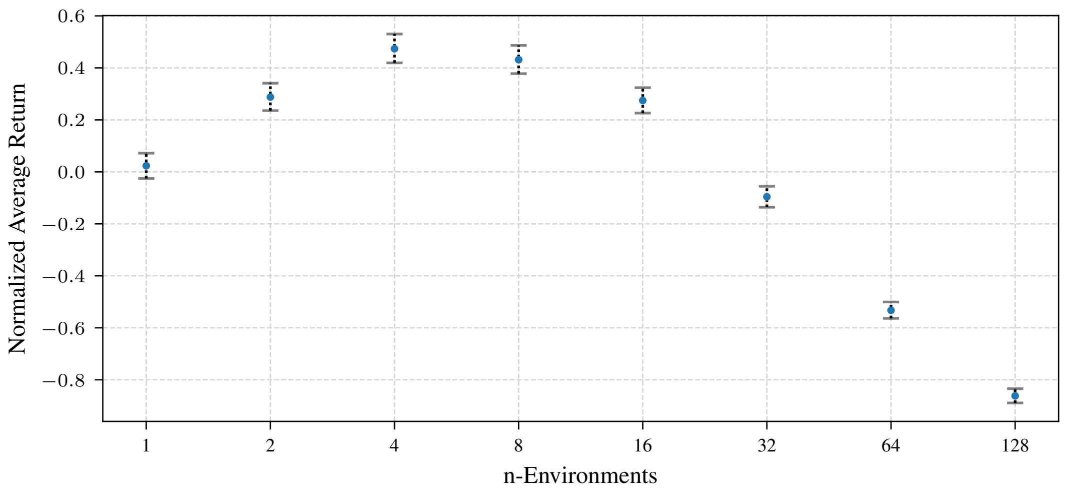
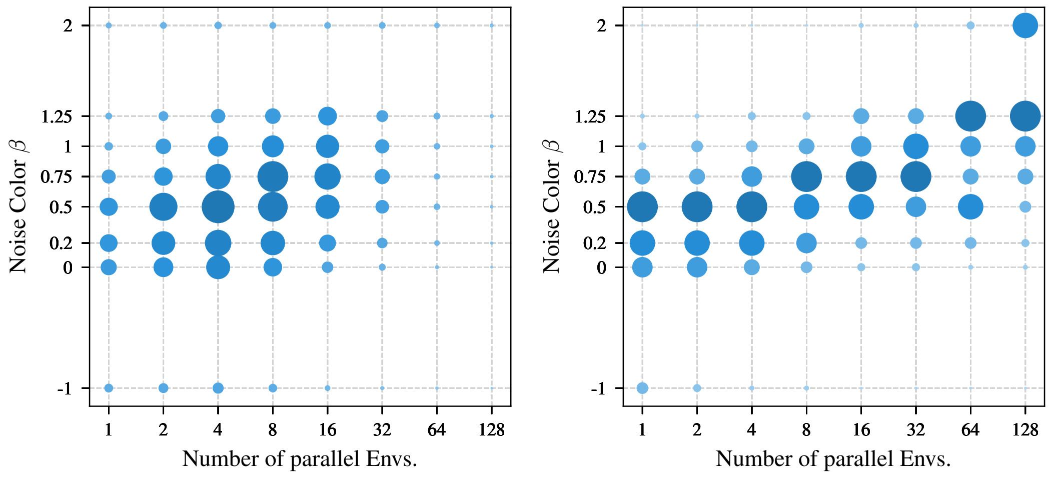


Figure 4: Performance averaged across environments and noise colors: the number of parallel data collection environments has a significant impact on the performance. Bootstrapped 95% confidence intervals for the mean are shown. With $N_{\text{env}} = 4$ achieving the highest performance, though not significantly outperforming $N_{\text{env}} = 8$



(a) Marker size represents averaged performance (scaled to the sixth power to highlight differences)

(b) Marker size represents rank of averaged performance within the same number of environments

Figure 5: Preferred noise color depends on number of environments: Average performance across environment, impact of noise color β combined with n-envs number of parallel environments: (a) A trend is visible: the averaged performance is larger for larger β when more collection environments are used, but the decline due to the number of environments outweighs this trend. (b) Ranks of average performance are indicated by circle size, ranks are calculated across noise-colors but within the same number of environments. The positive trend between number of environments and larger β is clearly visible.

3.1.4 How Does the Number of Parallel Collection Environments and Noise Color Interact?

The previous section indicated that the learning performance is impacted by the number of parallel collection environments and hence the size of the dataset used in each update cycle. We found a significant difference between the noise preference in off-policy methods ($\beta = 1.0$) and PPO as an on-policy method ($\beta = 0.5$). Since off-policy methods showed a preference for a larger β ($\beta = 1$) and off-policy methods employ a replay buffer, and thus have access to more samples in each update, this begs the question: does the number of samples used in each PPO update show any interaction with the preferred noise type β?

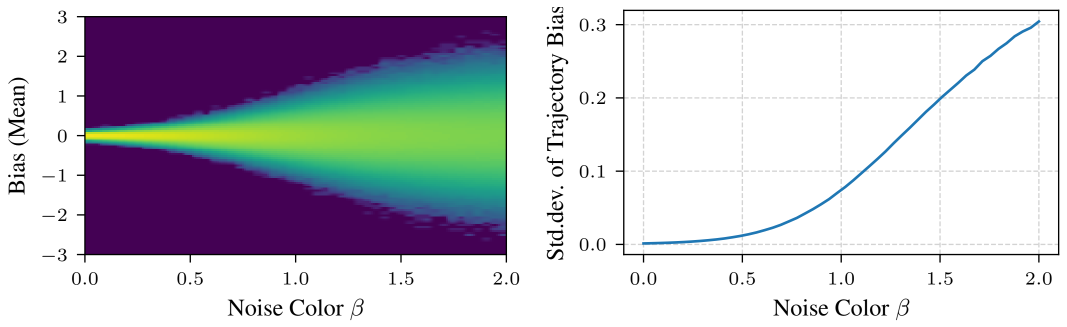
Figure 5a shows the average performance across environments depending on the number of collection environments (x) and noise color (y). A positive association between larger β (more correlated noise) and larger number of environments N_{env} becomes visible. This trend is more discernible when the performances for each noise color are compared separately for each number of parallel collection environments. Figure 5b shows the rank of the average performance, separately ranked for each color: a larger number of environments leads to favorable performance of more correlated noise, indicating that with larger samples (i.e., larger N_{env}), more exploration is beneficial. However, in combination with the finding from Sec. 3.1.3, we find the best performing configuration as $\beta = 0.5$, $N_{\text{envs}} = 4$.

3.1.5 Why Do Larger β Work Better for Larger N_{env}?

In the previous section we showed that, grouping for each N_{env} and comparing the different β within each group, there is a tendency for better performance in larger β values when N_{env} was increased.

Larger β values lead to more correlated perturbation sequences $\{\varepsilon_1, \dots \varepsilon_T\}$ and, since many environments feature integrative dynamics, larger state space coverage [4, 9]. While this is beneficial for acquiring new information, it also implies that the samples are more spread out and that the density around the mean actions proposed by the policy is smaller. Similarly, a systematic non-zero bias would prevent the policy from collecting on-policy data and would lead to learning instability.

We measured the bias, as the mean across each sequence of $\{\varepsilon_1, \dots, \varepsilon_T\}$, for different β and found that, while there is no systematic bias ($= 0$) for all β, the spread of the bias increases with β (Figure 6a). The resulting standard deviation of the bias as a function of β is shown in Figure 6b.



(a) Distribution of the bias in ε_t, calculated by the mean across sequences of perturbations $\{\varepsilon_0, \ldots, \varepsilon_T\}$, as a function of the noise color β.

(b) Standard deviation of the bias in $\{\varepsilon_0, \ldots, \varepsilon_T\}$, calculated as the standard deviations over the biases in (a) as a function of noise color β.

Figure 6: Increasing the noise color β, increases the correlation in the perturbations and the spread of the bias of each noise sequence.

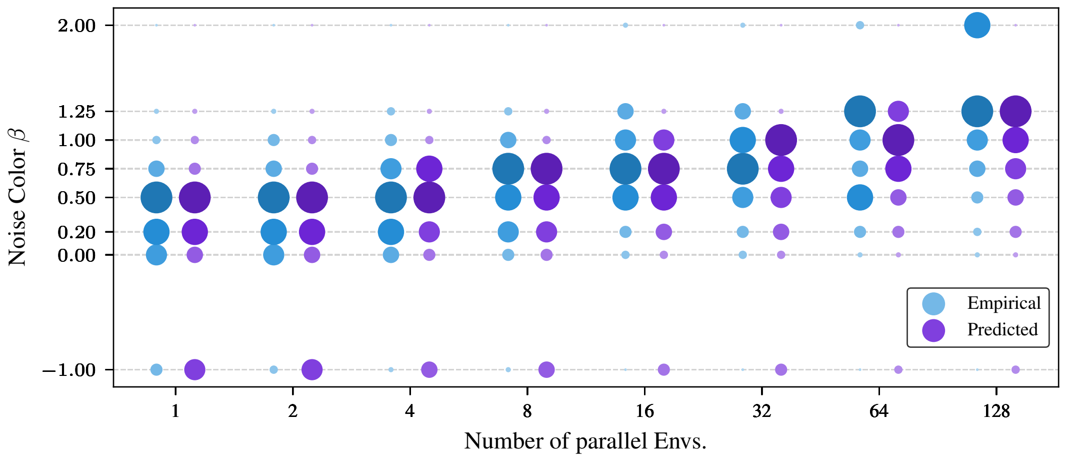


Figure 7: Performance rank, viewed separately for each number of parallel environments. Results from our empirical analysis are compared to a projected best noise color. Larger numbers of parallel environments decrease the variance of the bias, allowing for larger β.

When N_{env} is increased, more sequences of $\{\varepsilon_1, \ldots, \varepsilon_T\}$ are pooled. This results in a decrease of the variance of the bias of the collected sample $\sigma_N = \frac{\sigma}{\sqrt{N_{\text{env}}}}$. We can estimate a standard deviation $\hat{\sigma}(\beta, N_{\text{env}})$ of the bias for a given β and N_{env}, by combining $\sigma(\beta)$, the standard deviation of the bias as a function of β (Figure 6b), and the reduction due to the sample size $\frac{1}{\sqrt{N_{\text{env}}}}$.

We assume that the variance of the bias was optimal for the best performing β for each N_{env}. We estimate this optimal variance of the bias σ^* as the average of $\hat{\sigma}(n, \beta)$ for the best performing β (Table 2).

We calculate the difference between all combinations and the optimal combination: $E_{ij} = (\sigma^* - \hat{\sigma}(\beta_i, N_j))^2$ and rank these differences E_{ij} separately for each N_{env} (Figure 7). We find a trend closely matching the performance ranking: larger variance due to larger β is compensated by larger number of environments (and thus samples). The *theoretical* best β for each N_{env} (Figure 7) closely follows our *empirical* observation (Figure 5b).

This indicates, that the increased variance in the bias due to larger β is compensated by larger N_{env} and since generally more exploration (i.e, larger β) would be favorable to collect more diverse samples, the set of best trade-off configurations of (β, N_{env}) shows a positive correlation between β and N_{env}.

4 Conclusion

In this work, we performed a comprehensive empirical evaluation of temporally correlated colored noise in the on-policy method PPO. The temporal correlation of colored noise can be seen in the power spectral density (PSD) of the noise sequence: the PSD varies with $1/f^{\beta}$, where β determines the color of the noise. By employing the re-parameterization trick ε is sampled from a Gaussian distributed, yet temporally correlated, noise source. We evaluated on a set of standard benchmark tasks and found the average expected policy performance was significantly improved when switching from the default uncorrelated white noise source ($\beta = 0$) to a correlated noise ($\beta = 0.5$) between white and pink noise ($\beta = 1$). We evaluated different numbers of parallel collection environments and found that too large or too small numbers of parallel collection environments hinder learning performance (with $n = 4$ performing best in our evaluation). We found that with an increase in parallel collection environments, more correlated noise performs better. We recommend correlated colored noise ($\beta = 0.5$) for on-policy learning as a more efficient default than white noise.

Acknowledgments We want to thank Onno Eberhard and Samuele Tosatto for helpful feedback on earlier revisions of this text. Georg Martius is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a Posteriori Policy Optimisation. *arXiv:1806.06920 [cs, math, stat]*, June 2018. http://arxiv.org/abs/1806.06920.

[2] Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A Survey of Exploration Methods in Reinforcement Learning. *CoRR*, abs/2109.00157, September 2021. http://arxiv.org/abs/2109.00157.

[3] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study. In *International Conference on Learning Representations*, January 2021. https://openreview.net/forum?id=nIAxjsniDzg.

[4] Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. Pink Noise Is All You Need: Colored Noise Exploration in Deep Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*, February 2023. https://openreview.net/forum?id=hQ9V5QN27eS.

[5] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations*, 2020. https://openreview.net/forum?id=r1etN1rtPB.

[6] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, October 2018. http://arxiv.org/abs/1802.09477.

[7] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. *arXiv:1812.05905 [cs, stat]*, January 2019. http://arxiv.org/abs/1812.05905.

[8] Jakob Hollenstein, Matteo Saveriano, Auddy Sayantan, Erwan Renaudo, and Justus Piater. How does the type of exploration-noise affect returns and exploration on Reinforcement Learning benchmarks? In *Austrian Robotics Workshop 2021, Vienna, Austria*, pages 22–26, June 2021. https://iis.uibk.ac.at/public/papers/Hollenstein-2021-ARW.pdf.

[9] Jakob Hollenstein, Sayantan Auddy, Matteo Saveriano, Erwan Renaudo, and Justus Piater. Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance. *Transactions on Machine Learning Research*, November 2022. ISSN 2835-8856. https://openreview.net/forum?id=NljBlZ6hmG&referrer=%5BAuthor%20Console%5D(%2Fgroup%3Fid%3DTMLR%2FAuthors%23your-submissions).

[10] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Inf. Fusion*, 85:1–22, 2022. doi: 10.1016/j.inffus.2022.03.003. https://doi.org/10.1016/j.inffus.2022.03.003.

[11] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proc. 4th Int. Conf. Learning Representations, (ICLR)*, 2016. http://arxiv.org/abs/1509.02971.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236. http://www.nature.com/articles/nature14236.

[13] Felix Patzelt. Felixpatzelt/colorednoise. https://github.com/felixpatzelt/colorednoise, June 2019.

[14] Irving G. B. Petrazzini and Eric A. Antonelo. Proximal Policy Optimization with Continuous Bounded Action Space via the Beta Distribution. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, December 2021. doi: 10.1109/SSCI50451.2021.9660123.

[15] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. `https://openreview.net/forum?id=ByBAl2eAZ`.

[16] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. `http://jmlr.org/papers/v22/20-1364.html`.

[17] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 1634–1644. PMLR, 2021. `https://proceedings.mlr.press/v164/raffin22a.html`.

[18] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust Region Policy Optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1889–1897, Lille, France, 2015. JMLR.org. `http://dl.acm.org/citation.cfm?id=3045118.3045319`.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.

[20] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv:1801.00690 [cs]*, January 2018. `http://arxiv.org/abs/1801.00690`.

[21] J. Timmer and M. König. On generating power law noise. *Astron. Astrophys*, 300:707–710, 1995.

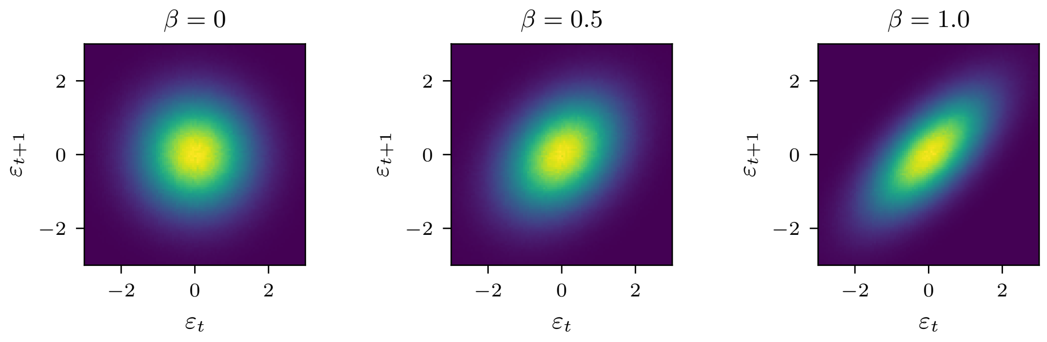
[22] George E. Uhlenbeck and Leonard S. Ornstein. On the theory of the Brownian motion. *Physical review*, 36(5):823, 1930.

[23] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. `http://link.springer.com/article/10.1007/BF00992696`.

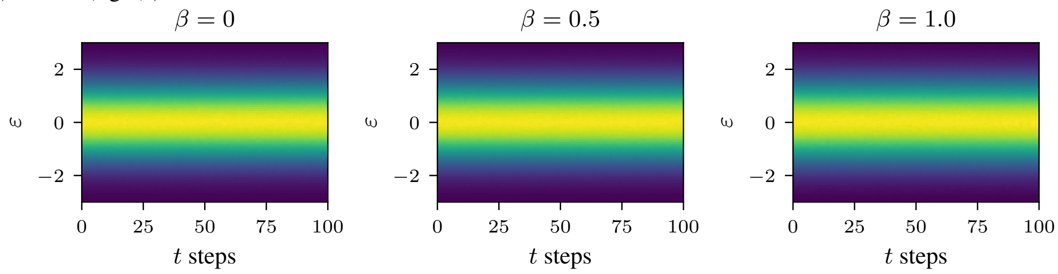
[24] Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Jianye Hao, Zhaopeng Meng, Peng Liu, and Zhen Wang. Exploration in Deep Reinforcement Learning: A Comprehensive Survey. *CoRR*, abs/2109.06668, July 2022. doi: 10.48550/arXiv.2109.06668. `http://arxiv.org/abs/2109.06668`.

A Colored-Noise Properties

Noise samples drawn from a colored-noise process are correlated between time steps when $\beta > 0$.



(a) Probability of ε_t vs. ε_{t+1}: subsequent noise samples are correlated when $\beta > 0$. (left) $\beta = 0$, (middle) $\beta = 0.5$ (right) $\beta = 1.0$



(b) Averaging over sequences of ε_t shows ε_t is Gaussian distributed at each step. The noise color does not influence marginal distribution properties.

Figure A.1: While colored noise shows increasing two-step correlation between ε_t and ε_{t+1} with increasing β, the marginal distribution of ε_t at each time step remains Gaussian.

B Best β for N_{env}

N_{env}	β	Average Performance
1	0.5	0.44
2	0.5	0.87
4	0.5	1.04
8	0.75	0.97
16	0.75	0.81
32	0.75	0.28
64	1.25	-0.36
128	1.25	-0.71

Table 2: Noise color β associated with the best average performance across environments. The listed performances were standardized to zero mean and unit variance for each environment.

C Colored noise generation

We build on the algorithm by Timmer and König [21] as implemented by Patzelt [13], Eberhard et al. [4]: we generate noise sequences $\tau_\varepsilon = \{\varepsilon_1 \dots \varepsilon_\tau\}$ of a pre-defined length τ. See Algorithm 1. In our experiments, we use sequences of length $\tau = 1000$, regenerating the whole sequence when all items are consumed.

Algorithm 1 Generating Colored Noise using Inverse Fourier Transform

```
 1: procedure GenerateColoredNoise($N, \beta$)
 2:     $L \leftarrow \lfloor N/2 \rfloor$
 3:     $f \leftarrow \{\frac{1}{N}, \frac{1}{N}, \dots, \frac{i}{N}, \dots, \frac{L}{N}\}$        ▷ Frequencies of components $0 \dots L$
 4:     $s \leftarrow \{\dots, f_i^{-\beta/2}, \dots\}$        ▷ Calculate scales
 5:     $w_L \leftarrow \begin{cases} s_L, & \text{if } L \text{ is odd} \\ s_L/2, & \text{otherwise} \end{cases}$
 6:     $w \leftarrow \{s_1, \dots s_{L-1}, w_L\}$
 7:     $\sigma \leftarrow \frac{2}{N} \cdot \sqrt{\sum w_i^2}$
 8:     $\alpha = \{\dots, \alpha_i, \dots\} : \alpha_i \sim \mathcal{N}(0, s_i)$        ▷ Real part
 9:     $\beta = \{\dots, \beta_i, \dots\} : \beta_i \sim \mathcal{N}(0, s_i)$        ▷ Imaginary part
10:     $\alpha_0 \sim \mathcal{N}(0, s_0 \cdot \sqrt{2})$
11:     $\beta_0 \leftarrow 0$
12:     $\alpha_L \sim \begin{cases} \mathcal{N}(0, s_0 \cdot \sqrt{2}), & \text{if } odd \\ \mathcal{N}(0, s_0), & \text{otherwise} \end{cases}$
13:     $\beta_L \sim \mathcal{N}(0, s_0) \cdot \begin{cases} 0, & \text{if } odd \\ 1, & \text{otherwise} \end{cases}$
14:     $\gamma \leftarrow \{\dots, \gamma_i, \dots\} : \gamma_i = \alpha_i + \boldsymbol{i}\beta_i$
15:     $\tau_\varepsilon = \mathcal{F}^{-1}[\gamma] \cdot 1/\sigma$
16:     return $\tau_\varepsilon$        ▷ Return noise sequence of length $N$
17: end procedure
```

D Benchmark Environments

Table 3: Environment name used throughout the paper and exact spec-id used with gym

Environment	Gym Spec.
Mountain Car	MountainCarContinuous-v0
Ball in Cup (Catch)	dm2gym.envs:Ball_in_cupCatch-v0
Cartpole Balance	dm2gym.envs:CartpoleBalance_sparse-v0
Cartpole Swingup	dm2gym.envs:CartpoleSwingup_sparse-v0
Cheetah Run	dm2gym.envs:CheetahRun-v0
Hopper Hop	dm2gym.envs:HopperHop-v0
Pendulum Swingup	dm2gym.envs:PendulumSwingup-v0
Reacher Hard	dm2gym.envs:ReacherHard-v0
Walker Run	dm2gym.envs:WalkerRun-v0

E Hyperparameters & Experiment Details

We use the PPO implementation by Raffin et al. [16].

Table 4: Hyperparameters used in our experiments and default values used by the PPO implementation [16] when different.

Param.	Value	Default
lr	0.00025	0.0003
n-steps	2048	
batch-size	128	64
n-epochs	10	
gamma	0.99	
gae-lambda	0.95	
clip-range	0.2	
normalize-advantage	True	
ent-coef	0	
vf-coef	0.5	
max-grad-norm	0.5	
use-sde	False	
sde-sample-freq	-1	
stats-window-size	100	

F Colored Noise $\beta = 0$ matches Vanilla PPO

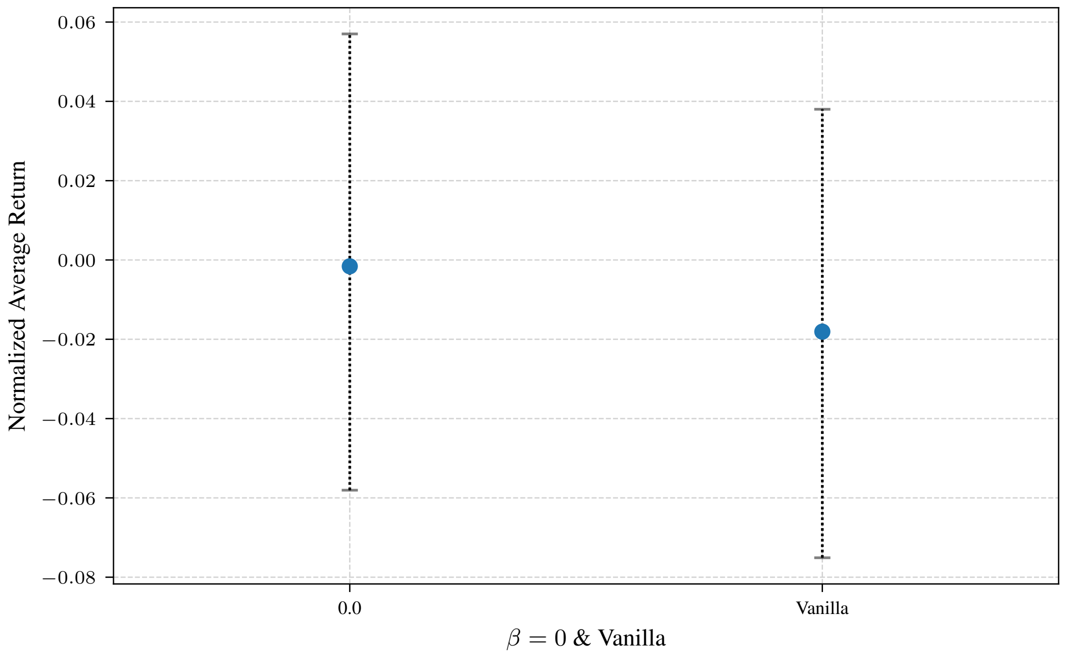


Figure F.1: Gaussian noise as used in vanilla PPO corresponds to white ($\beta = 0$) colored noise. Thus, the two algorithms are equivalent: Due to slightly different code paths minor deviations in performance estimates are expected, but the confidence intervals largely overlap and thus there is no significant difference between $\beta = 0$ and the vanilla PPO implementation.

G Final Returns & Performance

	Final Return		Performance	
	mean	max	mean	max
env				
Ball in Cup (Catch)	685.85	966.92	500.22	869.97
Cartpole Balance	742.18	1000.00	608.22	976.19
Cartpole Swingup	126.55	843.10	75.24	627.93
Cheetah Run	222.70	558.55	139.34	351.39
Hopper Hop	5.44	77.86	2.39	34.25
Mountain Car	66.92	98.33	61.88	93.53
Pendulum Swingup	338.21	885.04	205.43	678.91
Reacher Hard	266.71	935.94	139.72	613.31
Walker Run	105.10	473.11	69.66	194.53

Table 5: Return of the final policy (Final Return), *mean/max* across noise colors and number of parallel collection environments. Return averaged across training (Performance), *mean/max* across noise colors and number of parallel collection environments

H Update Size Matters: N_{env} vs. N_{steps}

In this work we found $N_{\text{env}} = 4$ to perform best. This has to be viewed in combination with $N_{\text{steps}} = 2048$, which is the default value for the used implementation. Note that the environments used in our experiments have an episode length $< N_{\text{steps}}$ and thus, the relevant quantity is the update size ($N_{\text{env}} \cdot N_{\text{steps}}$). This is confirmed by the experiment in Figure H.1. Which shows the main results (filtered by $\beta = 0.5$) and additional experimental results ($\beta = 0.5$) over various $N_{\text{env}} \in \{2, 4, 8\}$ and matching changes in $N_{\text{steps}} \in \{4096, 2048, 1024\}$ to keep the overall update size at 8192. The results indicate that the change of N_{env} vs. N_{steps} (when keeping the total update size constant), are not significant.

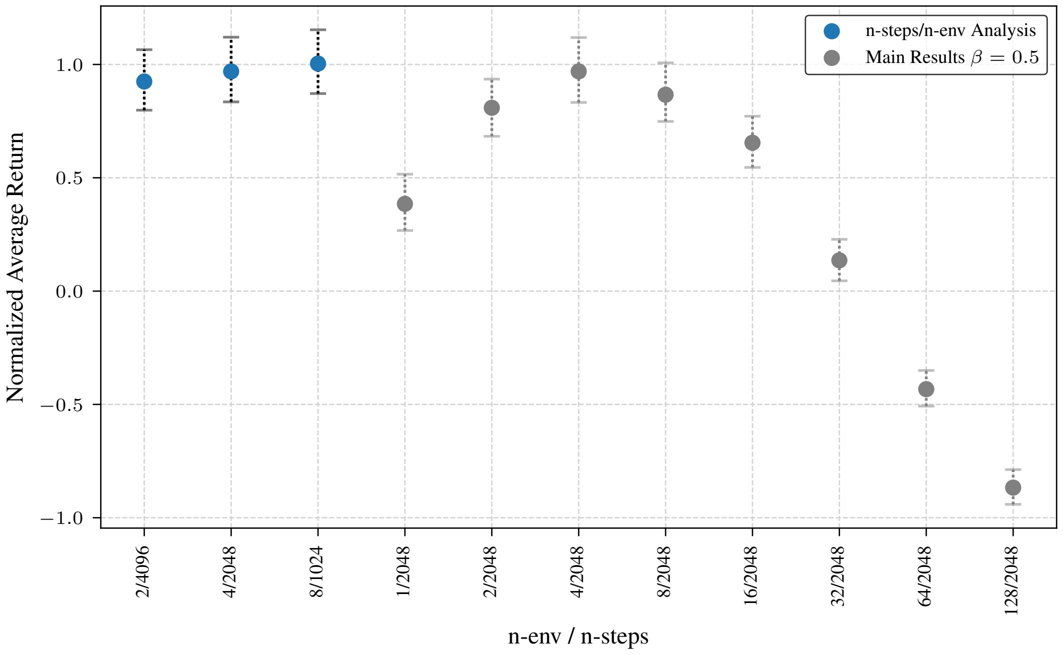


Figure H.1: Analysis of the impact of N_{env} vsN_{steps} for $\beta = 0.5$: When the total sample size is investigated ($N_{\text{env}} \cdot N_{\text{steps}}$), the influence of N_{env} and N_{steps} appears not significant (for $N_{\text{steps}} >$ episode length).

I N_{epochs} vs β

In this experiment we analyze the impact of varying $N_{\text{epochs}} \in \{5, 10, 20\}$ where 10 is the default used in the main experiments. Figure I.1 shows that the color preference for $\beta = 0.5$ does not appear to change significantly with variation in N_{epochs}. The experiments were performed with $N_{\text{env}} = 4$ on the same environments and seeds as before. Halving N_{epochs} reduces the performance, while doubling the number of epochs does not appear to have a significant impact on the color preference.

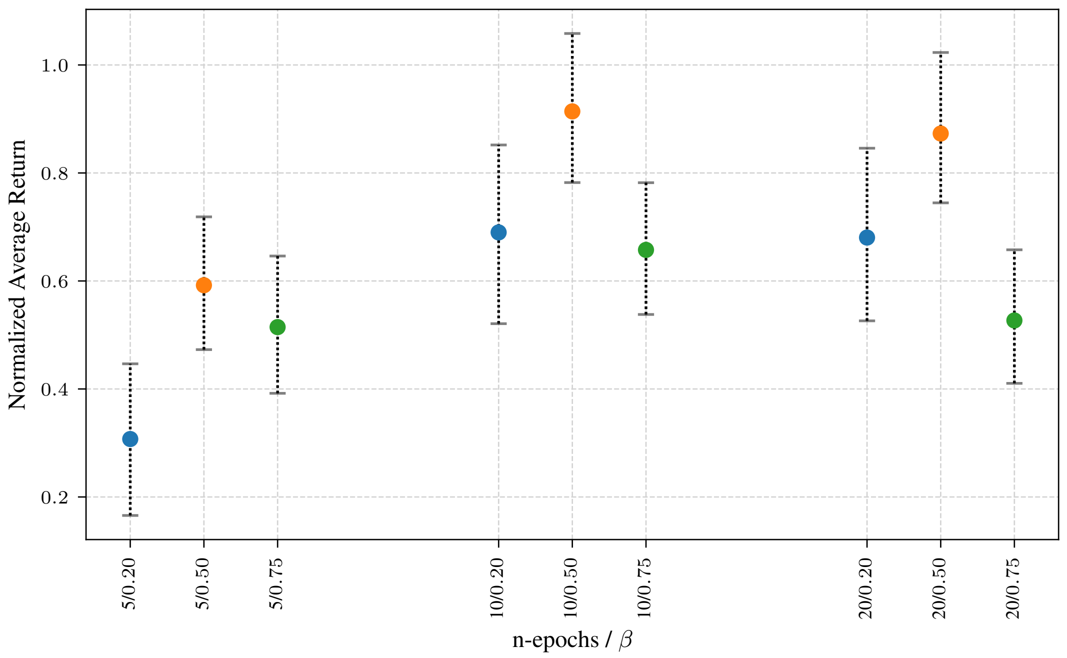


Figure I.1: Analysis of the impact of N_{epochs} vs. β on performance. $\beta \in \{0.2, 0.5, 0.75\}$, $N_{\text{env}} = 4$ and $N_{\text{epochs}} \in \{5, 10, 20\}$. The color preference does not appear to change significantly with variations in N_{epochs}.

J Final Returns per β

Table 6: Final Return per β and N_{envs}. $\beta = 0$ is equivalent to the vanilla PPO implementation. Mean across seeds is reported.

	β	-1.00	0.00	0.20	0.50	0.75	1.00	1.25	2.00	Vanilla
env	n-envs									
Ball in Cup (Catch)	1	701.4	833.4	843.5	811.4	721.8	308.4	356.1	290.8	811.4
	2	826.3	816.7	900.4	863.6	798.3	751.1	376.5	386.4	888.5
	4	887.7	936.8	929.1	922.8	829.2	861.8	584.3	393.3	926.6
	8	886.6	933.2	921.9	930.4	887.7	766.8	793.4	403.4	896.8
	16	799.0	881.0	932.2	938.6	906.4	913.8	922.3	385.8	884.8
	32	570.2	575.2	829.0	907.4	929.0	891.9	832.6	532.3	488.4
	64	451.5	146.5	349.4	721.5	688.3	753.8	855.9	664.3	140.0
	128	225.7	126.0	133.5	218.7	306.0	574.7	733.8	743.2	127.4
Cartpole Balance	1	820.6	981.2	1000.0	1000.0	810.1	691.5	171.1	51.0	973.4
	2	902.7	992.5	1000.0	1000.0	998.2	983.6	637.7	59.4	997.2
	4	892.3	1000.0	1000.0	1000.0	999.0	988.1	799.1	49.0	998.4
	8	762.2	996.7	1000.0	986.1	1000.0	999.4	743.8	89.0	1000.0
	16	221.3	1000.0	1000.0	990.8	1000.0	1000.0	723.8	35.4	1000.0
	32	153.9	990.7	1000.0	1000.0	1000.0	1000.0	901.3	96.7	1000.0
	64	115.0	967.3	1000.0	1000.0	1000.0	920.8	860.2	418.3	1000.0
	128	99.5	865.7	771.4	569.9	525.6	319.4	322.7	225.3	711.4
Cartpole Swingup	1	45.0	275.8	347.6	221.1	29.6	15.2	8.6	3.3	239.5
	2	55.6	404.4	534.2	337.8	146.9	25.1	26.3	10.2	268.8
	4	27.4	469.8	488.0	501.7	113.6	70.0	24.4	7.2	461.3
	8	27.5	407.5	393.8	365.3	118.8	29.0	19.7	13.9	430.5
	16	39.0	372.1	328.5	320.9	159.9	55.9	29.8	19.8	264.0
	32	11.9	119.1	235.0	264.7	149.5	71.9	38.6	10.3	191.4
	64	4.5	19.8	23.1	72.0	35.8	46.7	28.2	15.7	16.6
	128	0.7	6.1	7.0	5.4	9.0	14.0	5.8	14.0	4.1
Cheetah Run	1	220.8	279.5	242.9	184.6	145.6	95.3	99.9	124.6	240.0
	2	273.5	327.2	316.8	290.0	206.2	170.3	152.3	162.2	331.4
	4	311.6	357.2	357.3	311.6	326.8	299.0	239.6	217.2	337.7
	8	314.2	368.1	330.3	320.7	348.4	350.9	320.1	223.5	362.6
	16	289.9	334.3	328.6	354.5	338.4	323.7	297.5	200.5	358.8
	32	204.4	276.2	294.7	296.4	254.2	230.3	166.5	100.0	242.8
	64	147.8	118.5	178.6	182.2	166.7	93.1	70.3	39.9	131.7
	128	84.2	123.5	128.9	104.9	87.1	64.8	45.9	37.8	127.2
Hopper Hop	1	3.7	2.4	1.6	4.3	3.4	0.4	1.0	2.0	2.4
	2	3.7	1.2	2.5	3.0	4.0	5.0	2.3	2.5	1.7
	4	9.5	5.2	3.3	14.1	7.9	6.5	9.2	9.0	6.5
	8	4.9	5.9	12.1	16.2	28.7	10.3	9.9	5.1	7.8
	16	7.6	4.7	9.8	23.2	21.4	14.7	13.6	10.6	2.8
	32	4.0	3.5	3.1	5.1	5.9	7.5	5.5	3.4	0.4
	64	0.3	0.3	0.3	0.3	0.3	0.4	0.6	0.2	0.3
	128	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Mountain Car	1	-0.0	-0.0	28.0	93.9	93.9	93.9	93.8	93.8	-0.0
	2	-0.0	-0.0	32.4	93.9	93.9	93.8	93.9	93.9	-0.0
	4	-0.0	-0.0	23.4	93.9	93.8	93.8	93.8	93.8	-0.0
	8	-0.0	-0.0	84.6	93.9	93.8	93.8	93.8	93.7	-0.0
	16	-0.0	-0.0	79.9	93.9	93.8	93.8	93.8	93.6	4.7
	32	-0.0	-0.0	93.8	93.8	93.9	93.8	93.8	93.4	-0.0
	64	-0.0	-0.0	95.7	93.2	93.5	93.5	93.5	92.5	-0.0
	128	-0.0	-0.0	96.7	95.0	94.3	93.6	92.7	91.9	-0.0
Pendulum Swingup	1	501.9	714.5	648.0	521.1	328.5	153.2	106.3	58.5	697.2
	2	401.4	673.6	675.2	646.3	362.6	168.9	165.5	101.1	737.0
	4	420.3	677.5	766.9	694.6	625.0	403.1	270.7	153.1	733.1
	8	333.9	655.1	694.4	675.7	554.9	210.9	152.8	56.5	642.5
	16	149.9	370.7	432.6	640.7	646.9	379.6	207.3	29.7	381.5
	32	32.6	192.3	221.6	441.9	582.9	602.7	413.1	304.5	135.3

Continued on next page

env	β / n-envs	-1.00	0.00	0.20	0.50	0.75	1.00	1.25	2.00	Vanilla
	64	9.8	4.4	10.4	71.5	367.8	494.0	526.3	523.5	7.6
	128	2.0	1.2	1.1	2.2	15.1	69.5	126.2	199.0	1.1
Reacher Hard	1	10.9	22.6	30.5	19.0	14.6	17.5	13.4	12.1	66.4
	2	55.1	426.0	291.9	186.0	43.4	18.9	20.9	16.3	350.2
	4	133.8	663.8	513.8	332.5	188.5	45.5	37.2	19.0	559.5
	8	292.0	696.4	516.2	533.3	310.8	133.2	76.2	60.7	692.0
	16	322.5	776.5	718.1	654.4	499.5	422.7	308.4	169.7	742.9
	32	341.5	833.6	791.8	712.1	600.5	447.4	264.7	188.9	665.7
	64	207.2	551.8	597.6	546.7	283.2	132.1	105.3	150.8	499.4
	128	55.4	208.8	204.4	78.4	35.7	30.7	20.7	56.2	237.1
Walker Run	1	39.2	97.6	100.5	96.5	93.3	89.1	82.8	84.5	103.8
	2	48.7	126.4	122.4	141.8	127.0	130.3	102.8	82.2	122.2
	4	75.9	131.1	141.5	173.7	169.3	178.2	145.7	87.7	117.4
	8	84.8	95.6	116.4	167.9	222.2	261.4	191.8	89.4	111.2
	16	70.0	68.2	71.5	126.5	251.2	269.0	196.2	76.1	66.9
	32	59.4	55.0	60.8	117.8	188.2	181.6	140.1	67.2	57.5
	64	55.7	46.2	51.4	61.5	70.9	73.3	68.8	55.3	47.3
	128	41.6	39.5	41.4	48.7	46.8	42.7	43.1	43.2	39.3