PPO in the Bayesian World

Omar Tanner, Krisztina Sinkovics, RuiKang OuYang

Abstract

This study investigates optimising Proximal Policy Optimisation (PPO) for the Cartpole problem, a fundamental model mirroring a variety of complex real-world challenges, using Bayesian Optimisation (BO) with Gaussian Process (GP) surrogates. Sensitivity analysis highlighted the learning rate and entropy coefficient as critical hyperparameters. Our exploration of multifidelity approaches and actor/critic fidelities offered nuanced insights into computational efficiency and learning dynamics. Notably, fewer initial samples led to computational savings, with random sampling sometimes outperforming structured methods. The Matern52 kernel achieved an optimal balance with the Cartpole's dynamics, and the Expected Improvement (EI) acquisition function effectively balanced exploration and exploitation. The study also found a significant alignment between the periodic and combined kernels with the physical dynamics of the Cartpole problem. These results inform future research in Reinforcement Learning (RL) optimisation, emphasising adaptive approaches, multifidelity, informative GP kernels, and experimental design in varied RL contexts, while addressing the inherent difficulty in tuning PPO.

1 Introduction

1.1 Motivation

This study addresses the intricate challenge of tuning PPO in dynamic environments, with the Cartpole problem [1] serving as a quintessential example. The Cartpole, mirroring real-world balancing tasks such as robotic arm control or autonomous vehicle stabilisation, offers a fundamental model for understanding equilibrium maintenance in complex systems. The complexity of PPO tuning, heightened by its sensitivity to hyperparameters [3], necessitates the employment of advanced optimisation techniques. We utilise BO [11] with GP surrogates [4], complemented by multifidelity modeling, to efficiently navigate this optimisation landscape. Through sensitivity analysis [12] and experimental design methods [2, 7], our approach refines PPO's performance. This research is crucial for enhancing PPO's applicability in foundational problems and provides valuable insights for optimising other deep RL algorithms, such as Deep Q-Networks [6] and Trust Region Policy Optimisation [9]. These insights are applicable in various domains including robotics, AI in gaming strategies, and dynamic resource allocation in network systems. By improving the tuning process for PPO, our study offers strategies for optimising complex RL tasks across a spectrum of applications.

1.2 Proximal Policy Optimisation and the Cartpole Environment

Proximal Policy Optimisation (PPO) is a state-of-the-art RL algorithm which aims to improve the training stability of the policy by clipping the extent of the policy updates at each timestep in its objective function in Eq 1 [10], where ϵ controls randomness in the policy. $\hat{A}_t$ is the advantage function which depends on the future reward discount factor γ and λ for bias-variance trade-off.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], \text{where } \hat{A}_t = \sum_{k=0}^{T-t}(\gamma\lambda)^k\delta_{t+k} \quad (1)$$

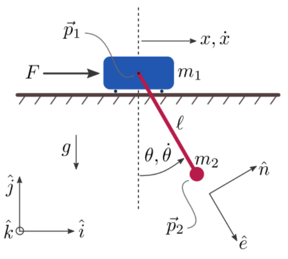
Cartpole environment is a classic control environment where cart is trying to balance a pole, its schematic view of is given in Figure 1 [1]. Cart Pole is governed by the following equations of motion for acceleration $\ddot{x}_t$ (Eq 3 and angular acceleration $\ddot{\theta}_t$ (Eq 2)

$$\ddot{\theta}_t = \frac{g\sin\theta_t + \cos\theta_t\left[\frac{-F_t - ml\dot{\theta}_t^2\sin\theta_t + \mu_c \mathrm{sgn}(\dot{x}_t)}{m_c+m}\right] - \frac{\mu_p\dot{\theta}_t}{ml}}{l\left[\frac{4}{3} - \frac{m\cos^2\theta_t}{m_c+m}\right]} \tag{2}$$

$$\ddot{x}_t = \frac{F_t + ml(\dot{\theta}_t^2\sin\theta_t - \ddot{\theta}_t\cos\theta_t) - \mu_c \mathrm{sgn}(\dot{x}_t)}{m_c+m} \tag{3}$$

The problem in this study is defined as tuning PPO to solve the Cartpole environment.

Figure 1: Cart Pole environment [5].



1.3 Objectives

This study aims to answer the following questions:

- How can BO be used to tune PPO to solve the problem more efficiently than other methods?
- How can we improve BO by applying experimental design methods and drawing from our knowledge of the workings of the environment and PPO?
- What can BO teach us about the problem?

2 Methodology

We use the following Python packages: `gymnasium` for the environment, `stable_baselines3` for PPO, `emukit` for sensitivity analysis, `GPy` for GPs, `GPyOpt` for BO, and `pyDOE` for Latin-hypercube.

2.1 Gaussian Process Surrogates

GPs are used as a surrogate model to approximate the unknown objective function that we wish to optimise. In our case, the objective function is PPO's mean evaluation reward. Thus, our surrogate approximates PPO's mean evaluation reward after training, for a specific set of input hyperparameters.

2.2 Sensitivity Analysis

We perform global sensitivity analysis on PPO's hyperparameters to determine which are the most sensitive. This ranking informs which parameters we to focus our optimisation on, as they cause the most variance in the output.

For each hyperparameter, we measure its first-order and total Sobol indices. These indices quantify the hyperparameter's global sensitivity for local variations and interactions with other hyperparameters respectively. Due to the complexity of our optimisation problem, analytic expressions for the Sobol indices are unknown, thus we approximate them using the Monte Carlo sampling via the Saltelli method [12]. Since the approximation requires a large number of samples (on the order of at least 10^4), we first fit a GP surrogate with 200 samples across the input space spanning all of the hyperparameters. With this surrogate, the Monte Carlo method uses it as an emulator to cheaply sample from our expensive PPO simulation and approximate the Sobol indices.

We perform this sensitivity analysis for two environments (CartPole, Pendulum) to assess how the optimisation of PPO differs between a discrete and a continuous environment.

2.3 Bayesian Optimisation

2.3.1 Kernels

When using GPs as surrogates, the choice of kernels influences their complexity, the behaviour of the acquisition function and the efficiency of optimisation. For example, a kernel leading to a smoother surrogate might result in an acquisition function that changes more gradually, while a well-chosen kernel allows for accurately modelling the objective function with fewer samples.

Kernels allow to encode our prior information about the problem. Hence, when choosing the kernel we aim to incorporate our knowledge about the Cartpole environment's dynamics (given in Equations 2 and 3) and properties of PPO's objective function (Eq 1). We assess the following kernels [8]:

- **Basic Kernels**: RBF, Matern32 and Matern52
- **Advanced Kernels**: Rational Quadratic, Exponential, Periodic $k_{\text{RQ}} = \sigma^2 \left(1 + \frac{(x-x')^2}{2}\right)^{-\alpha}$, $k_{\text{EXP}} = \sigma^2 \exp\left(-\frac{|x-x'|}{\ell}\right)$, $k_{\text{PER}} = \sigma^2 \exp\left(-\frac{2\sin^2\left(\pi|x-x'|/p\right)}{\ell^2}\right)$
- **Combined Kernels**: Periodic+RBF, Periodic+Matern32, Periodic+Exponential

2.3.2 Acquisitions

In BO, we use an acquisition function to guide the search. We explore two different acquisition functions: Expected Improvement (EI) and Maximum Probability of Improvement (MPI):

- EI uses a utility function $u(x) = \max(0, f' - f(x))$, which results in a trade-off between exploration and exploitation.
- MPI uses a utility function $u(x) = \mathbb{I}(f(x) \leq f')$, which evaluates f at the point most likely to improve upon this value, resulting in more exploitation.

2.4 Experimental Design

2.4.1 Sampling Methods

In conjunction with BO, we explore other search methods for finding the optimal set of PPO hyperparameters. Our first method is **random sampling**: which simply samples from the input space at random. Due to the samples' random nature, they could be concentrated in a specific area (by unlucky chance), skipping relevant areas of the search space. **Stratified sampling** attempts to overcome the unlucky situations with random sampling by splitting the search space into a set of equally separated 'strata', and randomly sampling a point from each stratum. Finally, **Latin-hypercube** is a form of stratified sampling which splits the search space into rows and columns and samples such that each row and column have exactly one sample.

2.4.2 PPO Fidelity

While in this study we are not dealing with a real-world system directly, RL still requires considerable computational resources. To save computation, we can use a lower fidelity model. We alter the fidelity of PPO by varying the number of timesteps we train it for and the size of its neural networks. PPO involves two networks: an actor, which decides the action taken by the agent (policy), and a critic, which measures how good the action taken is (evaluation). We vary the size of these networks together to measure the overall effect of PPO's fidelity on BO, and separately to understand which network has a stronger effect on optimisation. Table 1 shows a breakdown of the fidelity levels.

Fidelity	Timesteps (lr)	Timesteps (full set)	Network architecture (both)
Low	3000	6000	[16, 16]
Medium	10000	7000	[32, 32]
High	20000	8000	[64, 64]
Multi	0: low, 6: med, 10: high	0: low, 6: med, 14: high	0: low, 6: med, 14: high

Table 1: PPO Fidelity settings for Bayesian Optimisation wrt `lr` and wrt the full parameter set.

In the case of multi-fidelity, we vary fidelity from low to high across BO iterations: for the timestep schedule, we fix the net architecture at [64,64]; for the Net Architecture schedule, we fix a number of timesteps at medium.

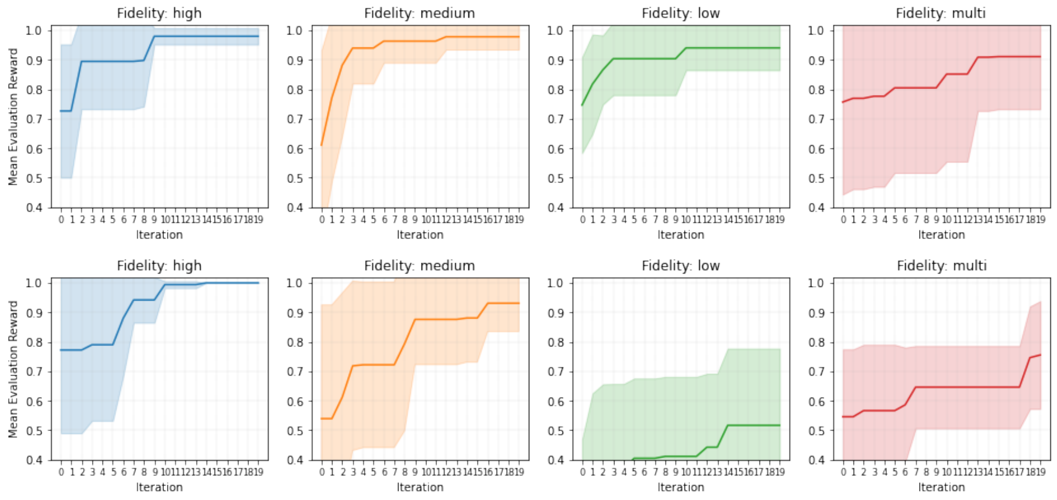


Figure 2: Effect of **timestep fidelity** on BO. Top: full parameter set; Bottom: `lr`.

2.5 Applying Experimental Design to Bayesian Optimisation

Since BO samples from probability distributions, different seeds can result in different behaviour. To assess BO's average performance across the different seeds, we repeat it 5 times, and measure the mean and variance across all runs at each iteration. Furthermore, BO typically begins with a short random sampling phase to ensure that it sufficiently explores the search space. This prevents BO from getting stuck in local optima. In addition to random sampling, we explore the use the other sampling methods listed in §2.4.1 in this phase, as they cover the input space to varying degrees.

3 Results

3.1 Sensitivity Analysis

Hypothesis 1 (H1): *The entropy coefficient and gamma would be the most sensitive hyperparameters, as an optimal policy requires a specific trade-off between randomness and foresight.*

Figure 11 shows the first-order and total Sobol indices for PPO at high fidelity in the Cartpole environment. We observe that the learning rate (α) is by far the most sensitive hyperparameter, refuting Hypothesis 1, followed by the entropy coefficient, corroborating Hypothesis 1. Thus, we focus our optimisation on the learning rate, entropy coefficient, gamma, `gae_lambda` and `clip_range`: the **full parameter set**.

3.2 Bayesian Optimisation

3.2.1 Timestep Fidelity

Hypothesis 2 (H2): *With more timesteps, BO would converge to a better optimum.*

The results both for the case of optimising learning rate alone and optimising the full set of parameters (see Fig. 2) show that higher fidelity PPO converges to a better optimum, but also converges faster. The latter is especially pronounced in the case of optimising only the learning rate. Mean reward from higher fidelity PPO starts higher and ends higher as expected, variance is decreasing. These results corroborate Hypothesis 2.

Hypothesis 3 (H3): *We expect multi-fidelity BO to reach a similar optimum as the high-fidelity case.*

In the case of optimising the learning rate alone, we clearly see that multi-fidelity based on the number of timesteps performs better than low-fidelity, yet it does not even reach the performance of medium-fidelity PPO. Multi-fidelity BO for the full set of parameters fails to reach even the low-fidelity mean

Fidelity type	Params / Fidelity	low	medium	high	multi
Timestep	**lr**	69	211	398	255
	full parameter set	143	174	187	163
Net Arch	**lr**	188	193	211	195
	full parameter set	138	140	174	141

Table 2: Computation time (seconds) for 20 iterations of BO under different fidelity types.

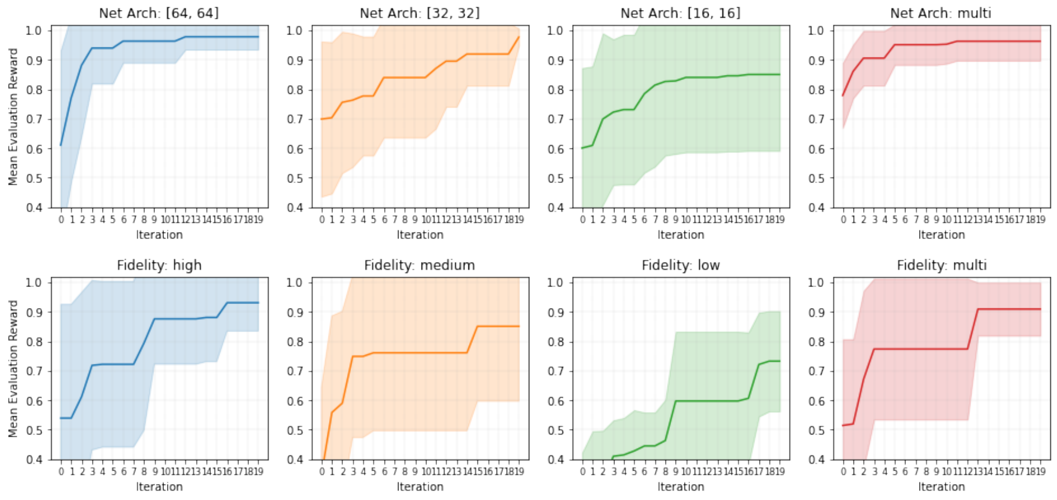


Figure 3: Effect of **network architecture fidelity** on BO. Top: full parameter set; Bottom: `lr`.

reward. Besides, Table 2 shows that multi-fidelity BO achieves comparable computational time to medium-fidelity one. Which refutes our Hypothesis 3 and does not enable computational gains.

3.2.2 Network Architecture Fidelity

Hypothesis 4 (H4): *With a larger network, BO would converge to a better optimum.*

Figure 3 shows BO in different fidelities based on network architecture, the bottom row reports mean reward from optimising learning rate alone, while the top row reports results for the full set of parameters respectively. Same as in the case of timestep-based fidelity, BO with a higher fidelity network outperforms lower-fidelity setup in both optimisation objectives, validating our Hypothesis 4. It is shown that a higher fidelity setup achieves higher results with lower variance. Interestingly, unlike in the timestep-based multi-fidelity, multi-fidelity based on network architecture helps BO to achieve comparable results with 20% shorter processing time in case of the full param set (see Table2), validating our Hypothesis 3.

3.2.3 Network Architecture Actor/Critic Fidelity

Hypothesis 5 (H5): *A larger critic network would reach a better optimum than a larger actor network. Since Cartpole is a relatively simple problem, it shouldn't require a complicated policy.*

Results shown in the figure 4 align with hypothesis 5. A stronger critic network leads to a better optimum than a stronger actor-network. Moreover, a stronger critic network also leads to faster convergence. Interestingly, increasing the network size beyond default 64 to 128 only hurts the performance of the BO. This implies that a simpler policy can suffice in the Cartpole environment.

3.2.4 Analysis of Bayesian Optimisation Iterations

To ensure that our implementation of BO works as expected and to understand its operation, we monitor the progress of its surrogate GP and acquisition function. Figure 5 shows the surrogate

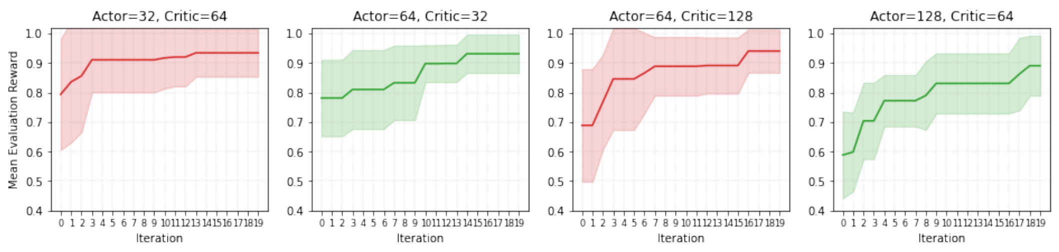


Figure 4: Effect of **Actor-Critic network architecture** on BO for the full set of parameters.

GP and acquisition function after a selection of interesting iterations. In this run, we only optimise the learning rate so our visualisation is possible in two dimensions. Our Cartpole environment had medium fidelity, and BO uses the EI acquisition function. We observe the following:

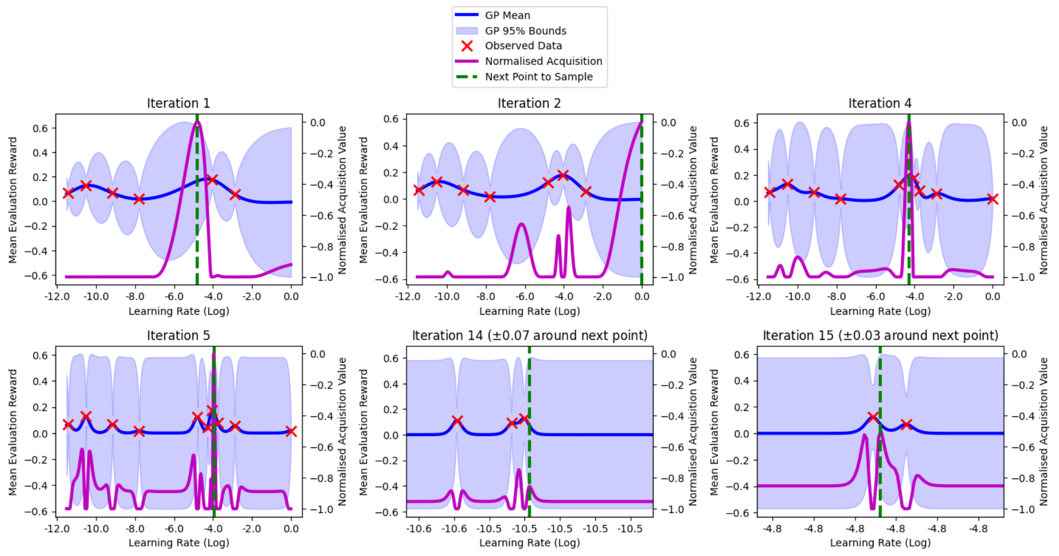


Figure 5: BO surrogate and acquisition function after specific iterations.

1. After Iteration 1, we have the first BO sample plus the 5 initial samples. The GP passes through the observed data as expected with some GP noise around it.

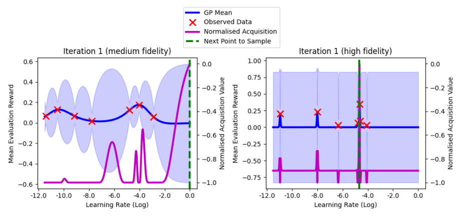
2. After Iteration 2, BO begins to explore, as the acquisition function peaks at the right edge of the space where there is no observed data.

3. After Iteration 4, BO exploits its knowledge, as the acquisition function peaks around the most densely around -4.

4. After Iteration 5, BO attempts to explore again, as the acquisition suggests 10.5 and 11. After Iteration 14, BO finally explores it.

6. After Iteration 15, BO returns to the previous local optimum it found in the iterations before 14. Then, it 'bounces' between the two found local optima at -4.8 and -10.5.

Figure 6: BO surrogate and acquisition function after the first iteration, for BO runs with medium and high fidelity respectively.



Furthermore, the GP increases its variance in onobserved regions, indicating that it has found local optima. This also implies that the mean evaluation reward is highly sensitive with variations in the learning rate, supporting our result from sensitivity analysis. We zoom in to the region around the next sample point, Iterations 14 and 15, so the micro-variations in the GP and acquisition function, showing that they did not overfit and considered other localities.

Interestingly, analysing the BO iterations at high fidelity reveals that the GP's variance becomes higher much earlier when we run PPO for more timesteps. Figure 6 compares the GP and acquisition function after the first BO iteration in medium and high fidelity, respectively. Therefore, BO progresses faster and to a better optimum in high fidelity, corroborating Hypothesis 2.

3.2.5 Acquisition Functions and Basic Kernels

Hypothesis 6 (H6): *EI will lead to a better optimum since it balances exploitation with exploration, which helps it escape local optima, while PI focuses on exploration; hence, it's more prone to getting stuck.*

Hypothesis 7 (H7): *A more continuous kernel is better because the environment is governed by differential equations, which would imply a smooth solution (RBF > Matern52 > Matern32).*

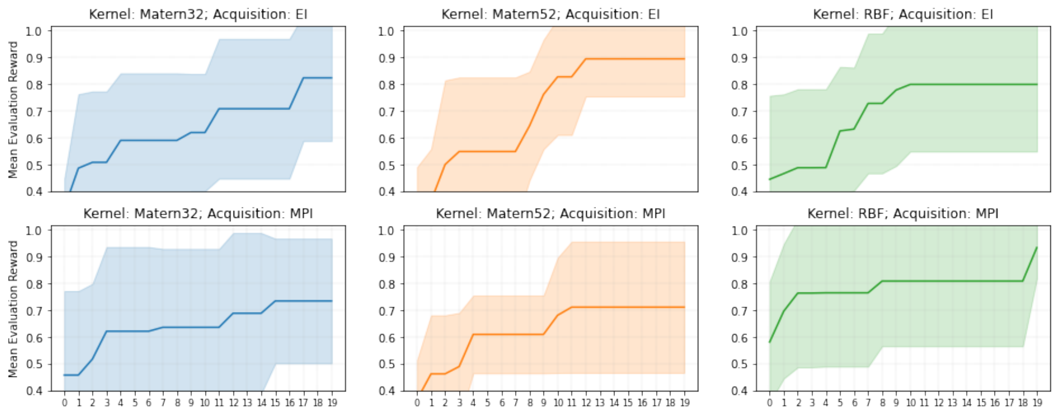


Figure 7: Effect of varying **kernel and acquisition function** on BO for learning rate.

Figure 7 shows that EI acquisition is always better than MPI, as the former one balances exploitation and exploration while the latter focuses on exploiting observed data. With EI acquisition, it shows that Matern52 kernel performs the best, which provides moderate continuity while Matern32 is too discontinuous and RBF is too smooth.

3.2.6 Advanced Kernels

Hypothesis 8 (H8): *A kernel involving periodicity would perform better than a non-periodic kernel since both acceleration (Eq 3) and angular acceleration (Eq 2) rely heavily on periodic functions.*

Hypothesis 9 (H9): *Kernels with norm/power hyperparameters would fare better because they can reflect the future reward decay in RL algorithms.*

From Figure 8, we can observe that Periodic kernel converges both faster and to a higher reward than Exponential, RBF or any of the Matern kernels in Figure 7, which agrees with Hypothesis 8. However, the performance of a simple periodic kernel is topped by the Rational Quadratic kernel, which solves the environment in under 10 iterations, arriving at a reward value of 1, going against H8. An exponent term allows Rational Quadratic to solve this problem quite quickly (§2.3.1), while Exponential kernel fares worse because it's based on the L1 norm; these results align with Hypothesis 9.

In the case of composite kernels, a combination of Periodic + Exponential kernel is by far the best; interestingly, this is the one that contains a component which does not exponentiate the norm (§2.3.1). This can be because the differential equation for cart acceleration $\ddot{x}_t$ has non-trigonometric terms (e.g. $F_t/(m_c + m)$) (Eq 3), and a combination of periodic + another kernel is more flexible and accounts for these non-periodic terms. Note: all terms of angular acceleration $\ddot{\theta}_t$ are periodic because the denominator has cosine (Eq 2). Also, periodic + matern is worse than periodic + rbf, but matern

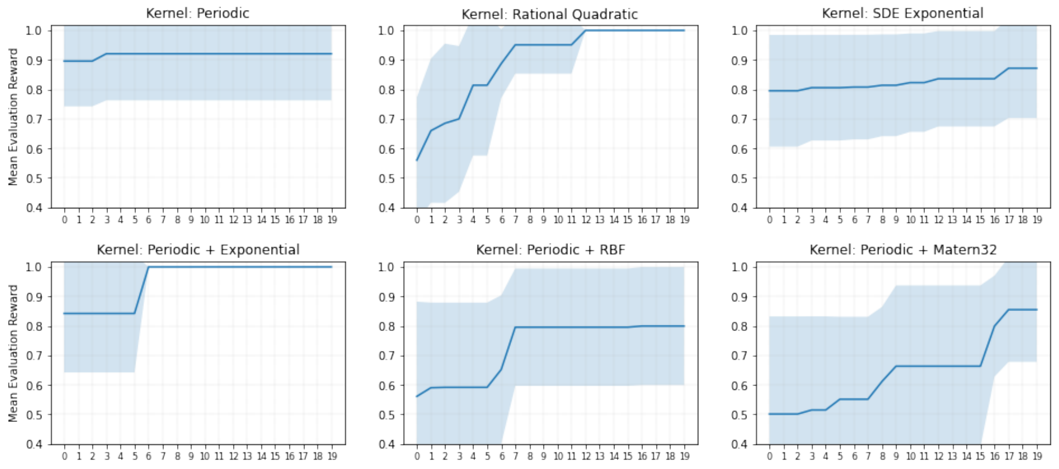


Figure 8: Effect of **advanced kernels** on BO for learning rate.

outperforms rbf normally. This can be due to matern non-continuities conflicting with periodic's continuous nature in the composite kernel case. These results highlight the importance of choosing a kernel informed by the knowledge of the system.

3.3 Experimental Design

3.3.1 Search Methods

Hypothesis 10 (H10): *The optima found with stratified and Latin-hypercube will be better than random search because they address the issue with random sampling of unlucky excluding areas.*

Figure 9 shows our results with the random, stratified and Latin-hypercube search methods when optimising just the learning rate and with 5 repeats at medium fidelity.

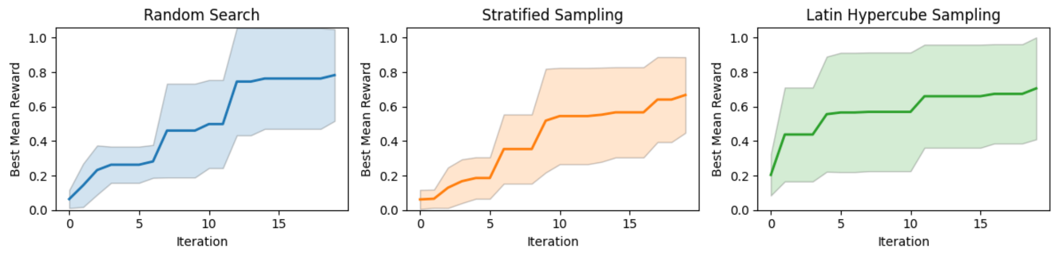


Figure 9: Optimising PPO's hyperparameters with different search methods.

After 20 iterations, we surprisingly observe that random search finds the best set of hyperparameters when compared to the optima found by stratified and Latin-hypercube sampling, refusing Hypothesis 10. This can be due to the sensitivity of the input space, as discovered from sensitivity analysis (§3.1) and analysis of the BO iterations (§3.2.4). On average, the samples from stratified and Latin-hypercube are further apart than those from random sampling, thus they search with a "wider" granularity and miss the small regions in which the optima lie.

We also observe that none of these sampling methods finds the true optimum (1.0). However, this optimum was found with BO (§3.2.1). Therefore, it is necessary to use BO to find the true optimum within 20 iterations.

3.3.2 Bayesian Optimisation Initial Sampling Methods

Hypothesis 11 (H11): *As stratified and Latin-hypercube sampling cover the search space more evently than random sampling, BO will reach a better optimum and converge faster with them. As more initial points give BO more observed data, BO will perform better with more initial points.*

Figure 10 shows our results when running BO with the different initial sampling methods (the iterations for the initial samples are not shown). In addition, we vary the number of initial samples, to measure the improvement gained from the initial samples. In this experiment, we ran PPO with medium fidelity, optimising all of the parameters.

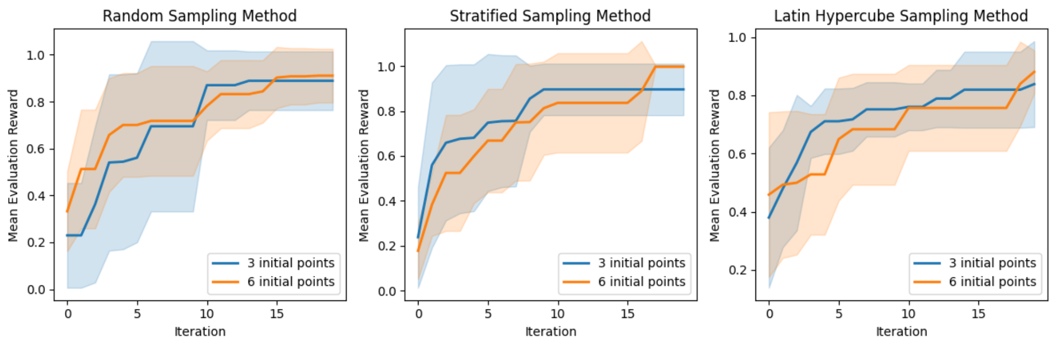


Figure 10: BO with different initial sampling methods and quantities of initial samples.

With 6 initial samples, the optimum reached by stratified sampling is marginally better than the other methods after 20 iterations. There is clear difference between the methods at the start of BO, where Latin-hypercube starts from a higher mean evaluation reward. Interestingly, despite starting higher, Latin-hypercube reaches its optimum last. Overall, due to stratified sampling reaching a better optimum, we conclude that stratified sampling is the best initial sampling method in our setting, corroborating Hypothesis 11. 3 initial samples is mostly better than 6 for all methods. For stratified and Latin-hypercube, 6 initial points overtakes 3 only at the very end of BO. The fact that 3 initial samples beats 6 for most of the time in BO is surprising and refutes Hypothesis 11. This can be due to BO being 'mislead' by the random points, exploiting areas of the search-space which are actually unpromising. This result is useful for reducing the computational requirements for BO, as we can conclude that it suffices (and can actually be better) to use fewer initial points, saving computation.

4 Conclusions and Discussion

Our study successfully met its objectives, offering insights into applying BO with GP surrogates to tuning PPO for solving the Cartpole environment. The sensitivity analysis was pivotal, guiding our optimisation by identifying the learning rate and entropy coefficient as crucial hyperparameters. In terms of multifidelity approaches, while they improved computational efficiency, they could not always match the outcomes of higher-fidelity setups. Intriguingly, we discovered that the critic's fidelity had a more pronounced impact on learning efficiency and performance compared to the actor's, enriching our understanding of PPO's behavior within the Cartpole environment.

Using fewer initial samples and stratified sampling in experimental design demonstrated potential computational savings. However, against our initial expectations, random sampling was found to be more effective than stratified-based methods in certain contexts. Within the domain of BO, the Matern52 kernel demonstrated an optimal balance of continuity for the Cartpole problem, resonating well with its physical dynamics. Additionally, the EI acquisition function consistently outperformed MPI, showcasing its effectiveness in balancing exploration and exploitation. Notably, the periodic and combined kernels aligned well with the Cartpole problem's physical dynamics, emphasising the importance of kernel choice in relation to the specific nature of the problem.

These findings offer valuable insights for complex RL tasks, underscoring the need for adaptive approaches in RL optimisation. Future research should focus on exploring multifidelity models, GP kernels, and experimental designs in other RL environments, potentially transforming the methods for approaching complex RL tasks, including competitive and cooperative multi-agent environments.

References

[1] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

[2] Kathryn Chaloner and Isabella Verdinelli. Bayesian Experimental Design: A Review. *Statistical Science*, 10(3):273 – 304, 1995. doi: 10.1214/ss/1177009939. URL `https://doi.org/10.1214/ss/1177009939`.

[3] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them, 2023.

[4] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. 07 2008. ISBN 978-0-470-06068-1. doi: 10.1002/9780470770801.

[5] Matthew Peter Kelly. Cart-pole equations of motion. `https://www.matthewpeterkelly.com/tutorials/cartPole/cartPoleEqns.pdf`, 2021. Accessed: 2024-02-17.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. URL `https://api.semanticscholar.org/CorpusID:205242740`.

[7] Paris Perdikaris, Maziar Raissi, Andreas Damianou, N. Lawrence, and George Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 473:20160751, 02 2017. doi: 10.1098/rspa.2016.0751.

[8] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.

[9] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[11] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2016. doi: 10.1109/JPROC.2015.2494218.

[12] I.M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 02 2001. doi: 10.1016/S0378-4754(00)00270-6.

A Sensitivity Analysis

In Figure 11 we show the first-order and total Sobol indices for PPO in the Cartpole environment.

Building on 3.1, in addition to the Cartpole environment, we performed sensitivity analysis for another environment in the OpenAI Gym (Pendulum). We found that the learning rate is also the most sensitive, however the other hyperparameters differ in their sensitivity compared to Cartpole. Figure 12 shows the Sobol indices for PPO at high fidelity for Pendulum, excluding the learning rate as it dominates the other parameters. Interestingly, gamma is the second-most sensitive hyperparameter for Pendulum, differing to Cartpole and further corroborating Hypothesis 1. The fact that the sensitivity of the hyperparameters differs between the environments highlights that different environments cause different PPO optimisation problems.

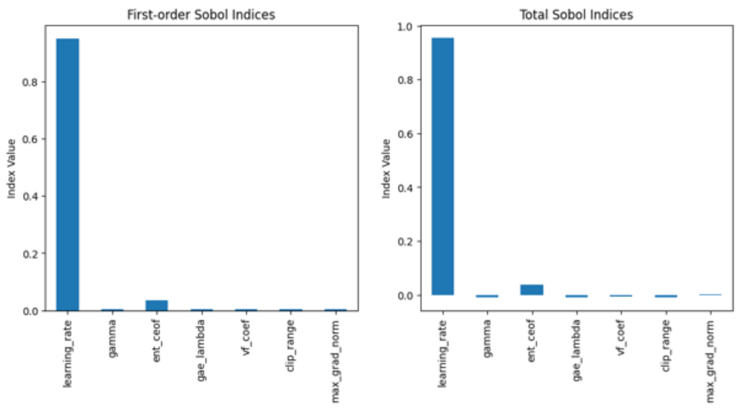


Figure 11: First-order and total Sobol indices for PPO in the Cartpole environment.

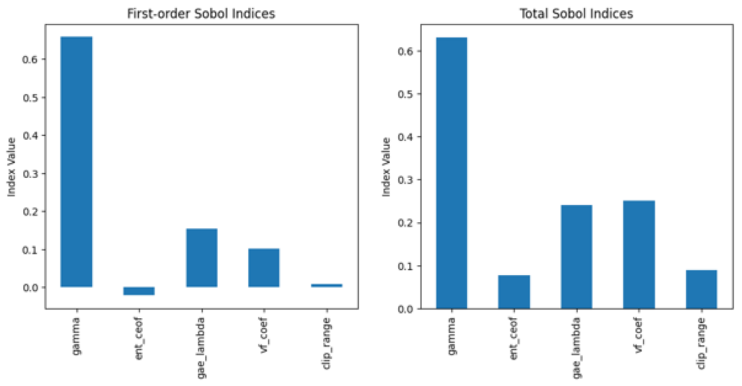


Figure 12: First-order and total Sobol indices for PPO in the Pendulum environment.