Reinforcement Learning for Drone Control: Lessons from Hyperparameter Optimization

Keiichi Ito, Jed Muff, A. E. Eiben
Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
email: k.ito@vu.nl, j.r.muff@vu.nl, a.e.eiben@vu.nl

***Abstract*—Reinforcement Learning (RL) can be an effective tool for optimizing robot control policies. However, RL performance is notoriously sensitive to hyperparameter configurations, and the relative importance of these parameters remains poorly understood. In this paper, we investigate this sensitivity by training control policies for a quadrotor drone across four complex navigation tasks: Figure8, Circle, Slalom, and Shuttlerun. Using Optuna, an open-source hyperparameter optimization framework, we systematically optimize a Proximal Policy Optimization (PPO) algorithm and analyze the results. Our analysis reveals two key findings: (1) The discount factor (Gamma) is the dominant hyperparameter, explaining 32–38% of performance variance. This is more than twice the impact of any other hyperparameter. Surprisingly, the learning rate has low importance (3.9%), suggesting it is necessary for fine-tuning but not the primary performance driver. (2) Hyperparameter values transfer across related tasks. A generic configuration derived by averaging task-specific optima achieves good task-specific performance across all four tasks, and leave-one-out cross-validation shows no significant transfer gap for 3 of the 4 tasks. These findings provide practitioners with actionable guidance: prioritize gamma tuning and leverage hyperparameter values from related tasks as strong starting points.**

***Index Terms*—Reinforcement Learning, Hyperparameter value Optimization, UAV, PPO, Optuna**

I. Introduction

Reinforcement Learning (RL) offers a method for learning control policies directly from experience (data). Proximal Policy Optimization (PPO) [1] is a widely-adopted algorithm due to its stability and sample efficiency. However, PPO has a set of 10–15 hyperparameters that control learning rate, network architecture, exploration-exploitation balance, and policy update constraints. The problem is that a priori knowledge about which hyperparameters matter most for a given task is limited, leading to a trial-and-error approach that is both time-consuming and computationally expensive.

Automated optimization methods such as Bayesian optimization [2], [3], population-based training [4] (adapting hyperparameter values during training via evolutionary dynamics), and Optuna [5] have made systematic search tractable. However, computational cost remains prohibitive—optimizing for even a single task can require 30–50 training runs, each demanding millions of simulation steps. Moreover, Henderson et al. [6] showed that apparent improvements can be illusory, arising from random seeds or implementation details rather than genuine advances, underscoring the need for multiple runs and statistical testing.

This raises a secondary question: *Must we optimize hyperparameter values separately for each task, or can a generic set achieve competitive performance across diverse scenarios?* If generic hyperparameter values suffice, *what performance trade-off do practitioners accept in exchange for reduced tuning cost?*

Understanding hyperparameter importance becomes even more critical when considering evolutionary robotics, where morphology and control are co-optimized. In such systems, RL algorithms must adapt to continually changing robot morphologies (and tasks) throughout the evolution. Yet, the influence of RL hyperparameters in morphology-control co-optimization remains almost unexplored. Existing work focuses primarily on evolutionary operators and fitness functions while treating RL hyperparameters as fixed defaults [7]–[9]. Although we don't explore co-optimization here, our systematic analysis of hyperparameter transferability across tasks provides foundational insights that could inform future co-evolutionary approaches.

In this paper, we present an empirical study in which we keep the reward function as well as the geometry and dynamics of the quadrotor constant while varying only the navigation task. This controlled setup allows us to study hyperparameter importance and hyperparameter value transferability across tasks. Rather than proposing new algorithms, we aim to provide practitioners with evidence-based guidance on which hyperparameters to prioritize and whether tuning effort can be amortized across related tasks.

A. Research Questions

This paper addresses two interconnected questions:

1) **Which hyperparameters matter most for PPO performance in navigation tasks?** We use fANOVA importance analysis across 408 Optuna trials to identify the primary drivers of performance.
2) **Is a generic hyperparameter set viable, or must we tune per-task?** We derive a tuned generic baseline by averaging task-specific optima and evaluate its performance-cost trade-offs via leave-one-out cross-validation.

B. Key Findings

- **Gamma (discount factor) dominates hyperparameter importance.** With 32–38% mean fANOVA importance across independent optimization runs (more than double any other parameter), gamma is the primary driver of PPO performance in (gate-passing) navigation tasks. This challenges our intuition that prioritizes learning rate tuning.
- **High-variance parameters can have low importance.** Learning rate varies $18\times$ across task-specific optima but explains only 3.9% of variance. Entropy coefficient importance varies across independent optimization runs, particularly for high-variance tasks like Slalom.
- **Tuned generic hyperparameters achieve strong cross-task performance.** Averaging task-specific optima yields a tuned generic configuration that achieves 48–96% of task-specific performance across all four tasks, providing a practical starting point for new deployments.
- **Transfer validated via leave-one-out cross-validation.** To confirm generalization to unseen tasks, we show that hyperparameter values averaged from three tasks transfer effectively to the fourth held-out task. For 3 of 4 tasks, no significant transfer gap is detected ($p \geq 0.105$; bootstrap CIs span zero). Only Shuttlerun exhibits a significant transfer gap ($p = 0.002$, CI excludes zero).
- **High-variance tasks resist optimization.** Slalom exhibits large performance variance ($\pm$14–17 gates), with no statistically significant difference between generic and task-specific configurations ($p > 0.08$).

II. Related Work

A. Hyperparameter Importance in RL

Henderson et al. [6] demonstrated that apparent algorithmic advances in deep RL are often illusory: identical algorithms with different random seeds produced statistically different results ($p < 0.002$), and unreported details like network architecture can cause complete failure to learn. Yet *which* hyperparameters matter most remains underexplored, particularly in aerial robotics.

Recently, Weller et al. [10] conducted a systematic study of hyperparameter importance across multiple RL algorithms (PPO, SAC, TD3, A2C) and MuJoCo locomotion tasks using fANOVA. They found that importance rankings are both algorithm- and task-dependent, with the discount factor and learning rate frequently among the most influential parameters. Our work complements theirs by focusing on a single algorithm (PPO) across multiple aerial navigation tasks, revealing that gamma dominates in this domain and that hyperparameter *values* transfer across related tasks—a question Weller et al. do not address.

For quantifying hyperparameter importance, we use fANOVA [11], which decomposes performance variance via functional ANOVA on a random forest surrogate. A more recent alternative is PED-ANOVA [12], which extends importance analysis to arbitrary subspaces of the hyperparameter space and is more computationally efficient. We adopt fANOVA for its direct integration with Optuna and established use in the RL literature.

B. Transfer Learning in Hyperparameter Space

Transfer learning typically focuses on policy or value function transfer [13]. One recent study demonstrates that information from past hyperparameter optimization runs can accelerate optimization on new tasks [14]. We investigate a slightly different question, analyzing whether a generic hyperparameter value set derived from multiple task-specific optima can achieve competitive performance.

C. RL for Aerial Navigation and Drone Racing

Reinforcement learning has shown promise for autonomous drone control and aerial navigation tasks. Benchmarks and simulators such as Flightmare [15], AirSim [16], and Aerial Gym [17], [18] enable investigation of learning-based policies for gate-passing tasks. However, existing work in this area typically focuses on *algorithm design* (policy optimization, curriculum learning, reward shaping) rather than systematic analysis of how hyperparameters affect performance. Most studies employ default hyperparameter values obtained from general RL literature (e.g., Schulman et al.'s PPO defaults [1]) or task-specific tuning without reporting which parameters actually matter.

It should be noted that stable and robust learning of drone navigation via RL is in itself a time-consuming and challenging task, as evidenced by a recent work by Lagos Suarez et al. [19]. They tackle sample efficiency and robustness through multiple mechanisms: a three-stage curriculum learning approach that progressively increases task difficulty, a multi-component reward function that captures transient and steady-state performance specifications, and systematic domain randomization. These complementary techniques achieve significant improvements in sample efficiency and robust stabilization from random initial conditions. However, their hyperparameter choices remain fixed.

To our knowledge, no prior work has performed a systematic *hyperparameter importance analysis* specific to aerial gate-passing tasks, nor has the transferability of hyperparameter values across related flight maneuvers been evaluated. This work fills that gap through a large-scale empirical study (408 Optuna trials, 3144 CPU-hours), identifying which PPO hyperparameters drive performance in quadrotor navigation and demonstrating that hyperparameter values can transfer across tasks with different geometries and difficulty levels.

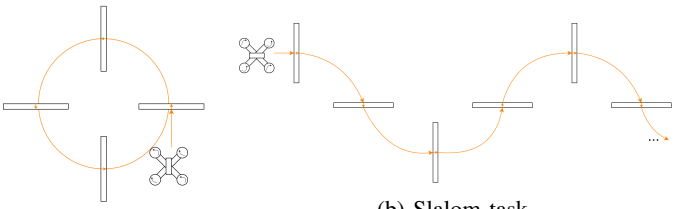
III. Methodology

A. Tasks

We consider four tasks shown in Figure 1. All tasks use identical square gates of 2.0×2.0 m. The four tasks exhibit varying levels of complexity and geometric characteristics:

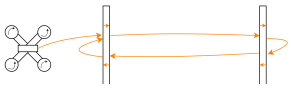
- **Circle**: A constant curvature path generating constant centrifugal acceleration. The 4 gates are positioned at cardinal points on a 1.5 m radius, forming a square circuit within a 6×6 m arena.

- **Figure8**: A continuous trajectory with an inflection point causing a switch in acceleration direction. The 7 gates are arranged in a figure-8 pattern spanning 8×5 m, with an intersection at the origin.
- **Slalom**: Navigation through a serpentine course with frequent direction changes. The gates are spaced 2 m apart along an 80 m linear track, with lateral positions alternating ($y \in \{-1, 0, +1\}$ m).
- **Shuttlerun**: A linear containment task requiring steep acceleration and deceleration. The 2 gate positions at $x = 2$ m and $x = 8$ m are traversed bidirectionally (4 logical gates) within a 2×12 m corridor.

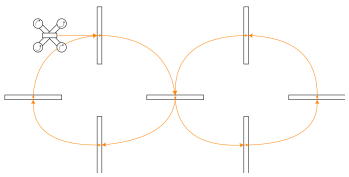


(a) Circle task.

(b) Slalom task.



(c) Shuttlerun task.



(d) Figure8 task.

Fig. 1: Diagrammatic representation of the tasks.

B. Quadrotor Platform

All experiments use a hypothetical standard X-configuration quadrotor with symmetric geometry and 5-inch propellers. The simulated platform has a total mass of 0.403 kg, composed of a flight controller with 4S LiPo battery (0.250 kg), four motor-propeller units (0.0196 kg each), and carbon fiber structural beams. Inertia properties (computed via parallel axis theorem from propeller positions 0.177 m from center): $I_{xx} = 0.00328$ kg·m² (roll), $I_{yy} = 0.00349$ kg·m² (pitch), $I_{zz} = 0.00672$ kg·m² (yaw). **Critically**, during simulation, inertia elements were clamped to a minimum of 0.01 kg·m² for numerical stability, substantially increasing effective inertia (1.5–3× larger than computed values). This may increase the difficulty of some tasks requiring high angular acceleration. Motor specifications: $k_f = 1.08 \times 10^{-6}$ N·s²/rad², $k_m = 1.22 \times 10^{-8}$ N·m·s²/rad², maximum angular velocity 3142 rad/s. Dynamics follow Z-Y-X Euler angle convention where yaw=0 aligns body X-axis (longitudinal) with world X-axis via rotation matrix $\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)$. The underactuated quadrotor has four independent motor commands controlling six degrees of freedom, enabling gate-passing trajectories via PPO-learned policies.

C. Optimization

We use Optuna [5] to search the hyperparameter space. Three configurations are compared:

1) **Baseline**: Standard hyperparameter values commonly used in RL literature with learning rate 3×10^{-4}, n_steps=1000, batch_size=1000, n_epochs=10, gamma=0.999, and network architecture 64^3 (three hidden layers of 64 units each). This differs from Stable-Baselines3 defaults (n_steps=2048, batch_size=64, gamma=0.99, net_arch=64^2) to better suit our task characteristics. All other PPO hyperparameter values remain at Stable-Baselines3 defaults (e.g., clip_range=0.2, gae_lambda=0.95, ent_coef=0.0, vf_coef=0.5, max_grad_norm=0.5, target_kl=None).
2) **Generic Optimization**: A single hyperparameter set computed as the arithmetic mean of task-specific optima across all four tasks. This requires the same upfront optimization cost as task-specific tuning (approximately 408 Optuna trials, 408M total evaluation timesteps). The key insight is that if one has already tuned for multiple related tasks, averaging those hyperparameter values provides a good starting point for new similar tasks without additional optimization.
3) **Task-Specific Optimization**: Parameter values tuned individually for each task via Optuna with 1M timesteps per trial, totaling approximately 100M evaluation steps per task: Shuttlerun (118 trials), Circle (100 trials), Figure8 (108 trials), and Slalom (82 trials due to early termination from system issues).

1) Hyperparameter Search Space: Optuna explores 12 hyperparameters:

- Learning rate: $[10^{-5}, 10^{-2}]$ (log scale)
- Network architecture: {small (32^2), medium (64^2), large (128^3)}
- N steps: {256, 512, 1000, 2048}
- Batch size: {32, 64, 128, 256, 512, 1000}
- N epochs: [3, 30] (integer)
- Gamma: $[0.9, 0.9999]$ (log scale)
- GAE lambda: [0.8, 0.99]
- Clip range: [0.1, 0.4]
- Entropy coefficient: $[10^{-8}, 0.1]$ (log scale)
- Value function coefficient: [0.1, 1.0]
- Max gradient norm: [0.3, 5.0]
- Log std init: [-1.0, 1.0]

2) Evaluation Protocol: Each configuration (baseline, tuned-generic, tuned-specific) is evaluated with:

- Training budget: 10M timesteps
- Parallel environments: 100
- Independent runs: 10 for all configurations
- Device: CPU (PPO with MLP policies performs better on CPU than GPU)
- Metric: Gates passed (cumulative over episode)

3) Reward Function: All four gate-passing tasks employ an identical reward structure:

$$r_t = (d_{t-1} - d_t) - 0.001\|\boldsymbol{\omega}_t\| \tag{1}$$

where $d_t = \|\mathbf{p}_t - \mathbf{g}_t\|$ is the Euclidean distance from drone position $\mathbf{p}_t$ to (the next) target gate position $\mathbf{g}_t$, and $\boldsymbol{\omega}_t \in \mathbb{R}^3$

is the angular velocity vector. The first term rewards progress toward the gate (distance reduction); the second term penalizes angular motion to encourage smooth trajectories. This shared reward structure across all tasks limits the diversity of task-specific reward engineering in this study.

IV. Results

A. Hyperparameter Analysis

The optimized hyperparameters for each task are presented in Table I. These values were obtained via Optuna hyperparameter optimization with the objective defined as gates passed (cumulative over an episode), while training used the shared reward function described in Section III (the same reward across Figure8, Circle, Slalom, and Shuttlerun). Across all four tasks, we conducted a total of 408 Optuna trials (Figure8: 108, Circle: 100, Slalom: 82, Shuttlerun: 118) with each trial evaluating a single PPO configuration for 1M timesteps. Computational costs are detailed in Section IV-E1.

TABLE I: Task-Specific PPO Hyperparameter Values Obtained via Optuna Optimization (408 trials total: Fig8: 108, Circle: 100, Slalom: 82, Shuttlerun: 118; 1M timesteps per trial)

Parameter	Figure8	Circle	Slalom	Shuttlerun
learning_rate	1.99e-04	1.39e-04	1.13e-05	4.92e-05
n_steps	256	512	256	256
batch_size	128	256	64	256
n_epochs	5	30	27	27
gamma	0.9964	0.9995	0.9970	0.9940
gae_lambda	0.8822	0.8997	0.9835	0.8116
clip_range	0.1082	0.2766	0.1070	0.1321
ent_coef	4.42e-05	5.11e-06	1.75e-02	1.69e-06
vf_coef	0.7035	0.3258	0.4322	0.6374
max_grad_norm	3.0925	1.2132	3.8497	3.5818
net_arch	large	small	large	large
log_std_init	-0.1245	0.4615	0.3505	-0.5330

Network Architecture Definitions (Stable-Baselines3 convention):

- `small`: pi=[32, 32], vf=[32, 32] — two hidden layers of 32 units each
- `medium`: pi=[64, 64], vf=[64, 64] — two hidden layers of 64 units each
- `large`: pi=[128, 128, 128], vf=[128, 128, 128] — three hidden layers of 128 units each

pi = policy network, vf = value function network. Both use separate MLPs with the specified hidden layer sizes.

B. Performance Comparison

We observed significant variance in effectiveness across tasks. Table II summarizes the improvement of the task-specific optimized models over the baseline.

C. Three-Way Performance Comparison

Table III presents a comprehensive comparison across baseline, generic, and task-specific configurations, including statistical significance from Welch's t-tests.

We observe the following.

- **Statistically significant improvements** for Figure8 and Shuttlerun tasks, where both generic and task-specific configurations outperform baseline ($p < 0.05$)

TABLE II: Performance Improvement (Task-Specific vs. Baseline). Gain: absolute change in gates passed. Speedup: reduction in episodes to first gate (%). Stability: ratio of baseline to task-specific learning-stability ratio ($\sigma_{\text{late}}/\max(\sigma_{\text{early}}, 0.01)$); Decreased < 0.8, Increased > 1.2.

Task	Gain	Speedup (%)	Stability
Shuttlerun	+30.7 gates	+67.3%	Decreased
Figure8	+37.9 gates	+68.3%	Decreased
Slalom	+13.0 gates	+43.0%	Similar
Circle	-1.9 gates	+27.3%	Increased

TABLE III: Three-way Performance Comparison: Gates Passed (Mean $\pm$ Std). Statistical significance: $^{*}p < 0.05$, $^{**}p < 0.01$ vs. baseline (Welch's t-test). $n = 10$ for all configurations.

Task	Baseline	Tuned-Generic	Tuned-Specific
Figure8	1.8 ± 1.2	$19.0 \pm 18.7^{*}$	$39.7 \pm 16.9^{**}$
Circle	49.7 ± 1.6	46.0 ± 8.8	47.8 ± 5.0
Slalom	11.0 ± 14.4	19.4 ± 16.3	24.0 ± 17.1
Shuttlerun	0.7 ± 0.7	$28.8 \pm 13.7^{**}$	$31.4 \pm 3.8^{**}$

- **Circle shows no statistically significant differences** between configurations at $n = 10$, which does not rule out smaller effects
- **Slalom shows high variance** ($\pm$14–17 gates), and no pairwise comparison reaches statistical significance ($p > 0.08$ for all)
- **Generic vs. task-specific:** Only Figure8 shows a significant difference ($p = 0.018$), with task-specific outperforming generic by $+20.7$ gates

D. Hyperparameter Sensitivity Analysis

To understand which hyperparameters most strongly influence performance, we computed fANOVA (functional ANOVA) importance scores from the Optuna optimization trials. fANOVA [11] decomposes performance variance into contributions from individual hyperparameters and their interactions, quantifying how much each parameter influences the objective function. Importance scores represent the percentage of total performance variance explained by each hyperparameter, with higher values indicating a stronger influence on task performance. Table IV summarizes the results, revealing substantial heterogeneity in parameter importance across tasks.

Key observations:

- **Gamma dominates** with 32–38% mean importance (the range reflects variability across independent Optuna runs for Slalom; see Limitations), particularly for Circle (68.1%) and Shuttlerun (39.7%). The prominence of the discount factor suggests that long-horizon credit assignment is critical for these navigation tasks.
- **Importance varies substantially across tasks.** Figure8 is most sensitive to gae_lambda (32.5%) and clip_range (19.9%), while Circle depends heavily on gamma (68.1%) and batch_size (11.7%).

TABLE IV: Hyperparameter Importance (%) via fANOVA Analysis. Each value represents the proportion of performance variance (gates passed) attributable to that hyperparameter. Mean is averaged across all four tasks.

Parameter	Mean	Fig8	Circ	Slal	Shuttl
gamma	**38.2**	12.5	68.1	32.3	39.7
clip_range	12.6	19.9	2.5	8.8	19.4
gae_lambda	12.5	32.5	2.3	8.9	6.2
log_std_init	7.0	3.6	2.2	9.0	13.3
max_grad_norm	6.1	4.3	6.5	10.2	3.5
batch_size	5.6	1.3	11.7	9.1	0.4
n_epochs	5.3	5.6	2.4	8.5	4.9
learning_rate	3.9	12.8	0.3	0.4	2.1
n_steps	3.8	0.5	3.2	2.9	8.5
vf_coef	2.8	5.5	0.5	3.5	1.6
net_arch	1.6	1.1	0.3	4.5	0.4
ent_coef	0.6	0.3	0.1	1.8	0.1

Importance via fANOVA (seed=42) from 408 Optuna trials (Fig8: 108, Circ: 100, Slal: 82, Shuttl: 118). Slalom values vary across independent optimization runs.

- **Learning rate varies widely but is of low importance.** Learning rate varies 18× across tasks but averages only 3.9% importance, suggesting it is *necessary for fine-tuning* but not the primary driver of performance. Entropy coefficient importance varies across optimization runs for high-variance tasks.
- **Stable parameter values may be stable because they matter.** Gamma is both the most important parameter and relatively stable across task-specific optima (0.994–0.9995), suggesting convergent evolution toward a universal long-horizon planning strategy.

E. Tuned Generic Baseline Evaluation

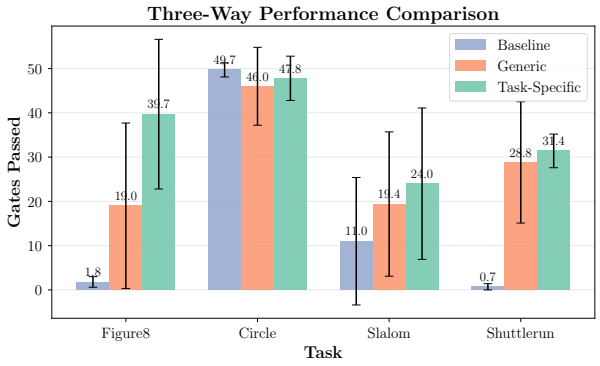


Fig. 2: Three-way performance comparison shows generic hyperparameter values achieve substantial gains over baseline while approaching task-specific performance.

Figure 2 demonstrates that tuned generic hyperparameter values (computed as the arithmetic mean of task-specific optima) achieve substantial gains over the baseline while retaining 48–96% of task-specific performance. For Circle, Slalom, and Shuttlerun, generic configurations are statistically indistinguishable from task-specific optimization ($p > 0.05$), validating hyperparameter value transfer across related tasks. Only Figure8 shows a significant performance gap ($p = 0.018$), with task-specific outperforming generic by $+20.7$ gates. Error bars represent ± 1 standard deviation across $n = 10$ independent runs, illustrating the robustness of this finding across different random seeds.

The tuned generic baseline hyperparameter values are shown in Table V

TABLE V: Tuned Generic Hyperparameter Values (computed as the arithmetic mean of the four task-specific optima, c.f. Table I).

Parameter	Value	Range
learning_rate	1.0×10^{-4}	Mean of 4 tasks
n_steps	320	256–512
batch_size	176	64–256
n_epochs	22	5–30
gamma	0.9967	0.994–0.9995
gae_lambda	0.894	0.81–0.98
clip_range	0.156	0.11–0.28
ent_coef	0.0044	10^{-6}–0.02
vf_coef	0.525	0.33–0.70
max_grad_norm	2.93	1.2–3.9
net_arch	medium (64^2)	small–large (32^2–128^3)
log_std_init	0.041	-0.5–0.5

Network architecture (net_arch): 3 of 4 tasks selected large (128^3), 1 selected small (32^2); medium (64^2) chosen as a practical compromise between capacity and training efficiency. Continuous hyperparameter values averaged arithmetically.

We infer the following. Generic hyperparameter values perform well when:

- **Tasks share structural similarity** (all involve gate navigation)
- **Baseline is far from optimal** (Shuttlerun: +28.1 gates, Figure8: +17.2 gates)
- **Rapid deployment is prioritized** over absolute performance

Task-specific tuning provides critical gains when:

- **Performance ceiling matters** (Figure8: generic achieves only 48%)

1) Computational Cost-Benefit Analysis: **Hyperparameter Optimization (Optuna Trials):** Our hyperparameter search required 408 Optuna trials across four tasks: Figure8 (108), Circle (100), Shuttlerun (118), and Slalom (82). With parallel execution on a computing cluster, the optimization elapsed time was approximately 40 calendar days; total CPU time summed across all trials was 3144 hours (131 calendar days):

- **Figure8:** 108 trials, 814.5 CPU-hours (33.9 days), avg 465.4 min/trial
- **Circle:** 100 trials, 602.9 CPU-hours (25.1 days), avg 361.8 min/trial
- **Shuttlerun:** 118 trials, 771.1 CPU-hours (32.1 days), avg 405.8 min/trial
- **Slalom:** 82 trials, 955.3 CPU-hours (39.8 days), avg 734.8 min/trial

Trial duration varied based on drone hover capability, task complexity, and hyperparameter quality. Failed trials (non-hovering drones) completed quickly; successful trials required the full 1M timesteps per trial, translating to 6–10 minutes wall-clock per trial under 100 parallel environments.

Evaluation (Configuration Comparison): After optimization, we evaluated each of four configurations (baseline, tuned-generic, tuned-specific, and LOO-Tuned-generic – explained in sec. IV-F) with 10 independent runs per task and configuration:

- **Baseline:** 11.5 hours total (17 min per run)—fast because untrained policies fail quickly
- **Tuned-Specific:** 91.9 hours total (varying 52–264 min per run)—slowest due to strong learning
- **Tuned-Generic:** 65.7 hours total (98 min per run)—stable cost across tasks
- **LOO-Tuned-Generic:** 115.8 hours total (125–196 min per run)—similar to task-specific

Total Cost Summary:

- **Optimization:** 3144 CPU-hours (40 calendar days elapsed with parallelization; 408 trials × 1M timesteps = 408M simulation steps)
- **Evaluation:** 284.9 CPU-hours (4 configurations × 4 tasks × 10 runs)
- **Grand Total:** 3429 CPU-hours ($\sim$ 40–50 calendar days with cluster parallelization)

For practitioners who have already tuned hyperparameter values for related tasks, the generic set offers a strong starting point for new, similar tasks without additional optimization. Task-specific optimization should be reserved for performance-critical scenarios where the 10–50% performance gap justifies the additional tuning effort.

F. Leave-One-Out Cross-Validation

The tuned generic hyperparameter value setup in Section IV demonstrates strong transfer across all four tasks. However, one might question whether this performance benefits from information leakage, i.e., the held-out task's hyperparameter values contribute to the generic average.

To address this concern, we perform leave-one-out (LOO) cross-validation as supplementary validation: for each task, we compute the generic hyperparameter values by averaging the optima from the *other three tasks only*, then evaluate on the held-out task. This simulates a practitioner applying hyperparameter values from existing tasks to a genuinely new task, providing an unbiased estimate of transfer performance.

1) Protocol: For each held-out task T_i:

1) Compute $\theta^{(i)}_{\text{LOO}} = \frac{1}{3}\sum_{j\neq i}\theta^*_j$ where θ^*_j is the task-specific optimum for task j
2) Train 10 independent runs on task T_i using $\theta^{(i)}_{\text{LOO}}$
3) Compare against baseline and task-specific performance

2) Results: Table VI presents the leave-one-out cross-validation results.

3) Interpretation: Table VII shows all pairwise comparisons. For Figure8, Circle, and Slalom, LOO-Generic does not significantly differ from Task-Specific ($p \geq 0.105$); the 95% bootstrap CIs span zero, indicating no detected transfer gap (though wide intervals reflect limited sample size, $n = 10$). Only Shuttlerun shows a significant transfer gap ($p = 0.002$, CI excludes zero), due to its unique linear containment dynamics.

TABLE VI: Leave-One-Out Cross-Validation (Gates Passed, $n = 10$)

Task	Base	LOO	Spec	Gap (95% CI)
Figure8	1.8 ± 1.2	36.2 ± 14.2	39.7 ± 16.9	9% [−32, 36]
Circle	49.7 ± 1.6	40.5 ± 15.9	47.8 ± 5.0	15% [−2, 38]
Slalom	11.0 ± 14.4	12.0 ± 14.1	24.0 ± 17.1	50% [−9, 82]
Shuttl	0.7 ± 0.7	11.0 ± 14.8	31.4 ± 3.8	65% [35, 92]**

Base=Baseline, LOO=LOO-Generic (3-task avg), Spec=Task-Specific. Gap = (Spec − LOO)/Spec × 100%. Bootstrap CIs (10k resamples); $^{**}p < 0.01$. CIs spanning zero indicate no significant gap.

TABLE VII: Pairwise Comparisons: p-values ($n = 10$, Welch's t-test)

Task	B vs G	B vs S	G vs S	L vs S	L vs G
Fig8	.017*	$< .001^{**}$	.018*	.623	.033*
Circle	.220	.278	.582	.195	.355
Slalom	.237	.083	.546	.105	.292
Shuttl	$< .001^{**}$	$< .001^{**}$	.576	**.002****	.012*

$^{*}p < .05$, $^{**}p < .01$. B=Base, G=Tuned-Generic (4-task), S=Tuned-specific, L=LOO (3-task). Only L vs S for Shuttl shows a significant gap.

For Figure8, LOO-Generic (36.2 gates) numerically outperforms all-tasks Generic (19.0 gates) with $p = 0.033$ (Table VII), though the effect is modest and potentially underpowered at $n = 10$. This suggests that including a task's own hyperparameter values can sometimes degrade performance.

Practical implication:

- For 3/4 tasks, no significant transfer gap detected ($p \geq 0.105$; CIs span zero)
- Task-specific tuning necessary only when transfer-gap CI excludes zero (as in Shuttlerun)
- Wide CIs ($\pm$30–40%) reflect limited sample size; practitioners should verify transfer empirically

G. Task Analysis

Shuttlerun saw the most dramatic improvement (+30.7 gates, from 0.7 to 31.4)—the baseline failed to learn a stable policy, whereas the optimized agent mastered containment efficiently. **Figure8** also benefited substantially (+37.9 gates, from 1.8 to 39.7), suggesting specific penalty coefficients are needed to avoid local optima. **Slalom** showed gains (+13.0 gates) but with high variance ($\pm$14–17 gates); no pairwise comparison reached statistical significance. **Circle** showed slight regression in peak fitness (-1.9 gates) but faster convergence (+27.3%); no statistically significant differences were detected at $n = 10$, which does not rule out smaller effects.

V. Discussion

A. Interpreting Hyperparameter Importance

Our results reveal that gamma (discount factor) is the primary driver of PPO performance in gate-passing navigation tasks, explaining 32–38% of variance across independent optimization runs. This is more than double any other parameter. This suggests that long-horizon credit assignment is the critical challenge: the drone must learn to value intermediate progress

toward distant gates rather than optimizing for immediate rewards.

The counterintuitive finding that learning rate has low importance (3.9%) despite varying 18× across tasks suggests that multiple learning rates can achieve similar performance once gamma is set appropriately. This has practical implications: practitioners should prioritize gamma tuning over the more commonly adjusted learning rate.

B. Guidelines for Practitioners

Based on our findings, we propose the following evidence-supported recommendations:

1) **Start with generic hyperparameter values** for rapid prototyping or when deploying on new similar tasks (demonstrated 48–96% retention across tasks)
2) **Prioritize gamma (discount factor) tuning**, it explains 32–38% of performance variance, more than double any other parameter
3) **Invest in task-specific optimization** when transfer-gap confidence intervals exclude zero (as observed for Shuttlerun) or when task dynamics are fundamentally distinct from prior tasks

C. The Slalom Challenge

The Slalom case exhibits the highest variance among all tasks (±14–17 gates across all configurations), warranting careful interpretation. While task-specific optimization achieves the highest mean (24.0 gates) compared to generic (19.4) and baseline (11.0), no pairwise comparison reaches statistical significance ($p > 0.08$ for all). We therefore cannot conclusively rank the configurations for Slalom; the observed differences may be attributable to noise.

The high variance itself is informative. Possible explanations include:

- **Reward signal mismatch:** Training reward consists of distance reduction to next gate and angular velocity penalties (Eq. 1), while Optuna optimized purely for the number of gates passed, potentially misaligning gradient signals during learning
- **Intrinsic task instability:** Slalom's tight turns may create bifurcating learning trajectories where small initialization differences lead to divergent outcomes
- **Bimodal outcomes:** The raw data reveals that runs cluster into "successful" (> 30 gates) and "failed" (< 10 gates) groups, suggesting the policy either learns the task or fails to converge

Despite this variance, task-specific optimization still achieves the highest mean performance, suggesting that tuning helps even for inherently variable tasks.

D. Limitations and Future Work

Our experiment comprises 10 replication runs per configuration and one Optuna optimization trajectory per task. Despite this limited scope, the findings reveal actionable patterns and inform future research directions. **Limitations:**

- **Single Optuna run per task:** Each task-specific configuration represents one optimization trajectory. A second independent Optuna run for Slalom discovered an alternative optimum with qualitatively different hyperparameter values but similar performance. The gamma-dominance finding remains robust, but multiple PPO configurations may achieve equivalent performance. Replicating Optuna trials would quantify this hyperparameter value uncertainty.
- Sample sizes ($n = 10$ for all configurations) provide limited power for small-to-moderate effects, especially in high-variance tasks like Slalom.
- Single morphology tested; hyperparameter values may vary across drone designs.
- Fixed training budget (10M steps); longer training may alter conclusions.
- Simulation-only evaluation; sim-to-real transfer remains unvalidated.
- **Task and reward homogeneity**; All four gate-passing tasks use an identical reward function (Eq. 1) and share gate-navigation structure. This homogeneity constrains generalization; tasks with qualitatively different reward structures (e.g., energy efficiency, time optimization, or obstacle avoidance) may exhibit different hyperparameter sensitivity patterns.
- **Inertia clamping confound:** The moment of inertia clamping (Section III-B) may have affected tasks requiring rapid angular maneuvers. Slalom, Shuttlerun, and Figure8 demand frequent roll/pitch changes that are penalized by the $\sim 3\times$ increase in effective roll and pitch inertia, while Circle's steady-state banking is less affected.
- **Gamma sampling encoding:** A more appropriate way to encode gamma would have been to sample (1 - gamma) on a log scale. The fANOVA impact is minor (<2 pp, ranking preserved), but the effect on Optuna's exploration cannot be verified without rerunning optimization.

Future Directions:

1) **Meta-learning:** Learn a hyperparameter policy that adapts during training.
2) **Co-optimization:** Investigate hyperparameter importance and transferability in morphology-control co-evolutionary systems, where robot bodies evolve alongside their controllers. Our finding that generic hyperparameter values achieve 48–96% of task-specific performance suggests that a single RL configuration might remain effective across diverse morphologies and tasks. Alternatively, hyperparameter values could be learned in the evolutionary process, potentially simplifying co-evolutionary optimization.
3) **Real-world validation:** Deploy on physical drones to assess sim-to-real gap.
4) **Simulation fidelity:** Re-running experiments with corrected inertia values would provide cleaner experimental conditions.

5) **Diverse tasks and rewards:** Expanding to tasks with different reward structures (e.g., energy efficiency, time optimization, obstacle avoidance) would test the generality of our findings.

VI. Conclusion

This research was motivated by the challenge of hyperparameter sensitivity in Reinforcement Learning. Our systematic evaluation of PPO performance using Optuna across diverse quadrotor navigation tasks revealed hitherto unknown effects, helping to make better-informed algorithm design choices for learning robotic flight control.

Our findings establish a useful hierarchy for practitioners: the discount factor Gamma is the dominant determinant of PPO's success. Learning rate and entropy coefficient contribute little to explained variance, although the importance of entropy coefficient can vary substantially across independent optimization runs.

Furthermore, we demonstrated that hyperparameter tuning does not always need to be a start-from-scratch endeavor. The ability of generic configurations to retain 48–96% of task-specific performance suggests that starting with generic hyperparameter values is a viable strategy for rapid deployment.

While these insights were derived in simulation and specifically within the context of quadrotor control, the underlying methodology of using fANOVA importance analysis alongside Optuna to identify transferable configurations is readily applicable to other robotics domains. Validating these findings on physical platforms remains an important direction for future work.

Acknowledgment

The research reported in this paper was supported by the European Commission Horizon project SPEAR, EU grant number 101119774, https://www.spear-robotics.com/. We are grateful to Fausto Lagos Suarez for stimulating discussions within the SPEAR project that motivated this work. We thank our colleague Karine Miras for regular, insightful discussions. We also thank the anonymous reviewers for their constructive feedback, which helped to improve the clarity and rigor of this paper.

References

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[2] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, Dec. 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1008306431147

[3] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[4] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," 2017. [Online]. Available: https://arxiv.org/abs/1711.09846

[5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019. [Online]. Available: https://arxiv.org/abs/1907.10902

[6] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," 2019. [Online]. Available: https://arxiv.org/abs/1709.06560

[7] K. S. Luck, H. Ben Amor, and R. Calandra, "Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning," in *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 485–500. [Online]. Available: http://proceedings.mlr.press/v100/luck20a.html

[8] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," pp. 9798–9805, 2019.

[9] J. Muff, K. Ito, E. H. W. Ang, K. Miras, and A. E. Eiben, "Unconventional hexacopters via evolution and learning: Performance gains and new insights," 2025. [Online]. Available: https://arxiv.org/abs/2505.14129

[10] J. Weller, B. Belousov, L. Metz, and J. Peters, "Importance estimation of hyperparameters in reinforcement learning," *Neurocomputing*, vol. 637, p. 129084, 2025.

[11] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 754–762. [Online]. Available: https://proceedings.mlr.press/v32/hutter14.html

[12] S. Watanabe and F. Hutter, "PED-ANOVA: Efficiently quantifying hyperparameter importance in arbitrary subspaces," in *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2023, pp. 4389–4396.

[13] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. 7, pp. 1633 – 1685, 2009.

[14] M. Feurer, B. Letham, and E. Bakshy, "Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles," in *AutoML Workshop@ ICML*, 2018.

[15] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1147–1157. [Online]. Available: https://proceedings.mlr.press/v155/song21a.html

[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," pp. 621–635, 2018.

[17] M. Kulkarni, W. Rehberg, and K. Alexis, "Aerial gym simulator: A framework for highly parallelized simulation of aerial robots," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 4093–4100, 2025.

[18] E. Arza, W. Rehberg, P. Weiss, M. Kulkarni, and K. Alexis, "Performance-guided task-specific optimization for multirotor design," 2025. [Online]. Available: https://arxiv.org/abs/2510.04724

[19] F. M. Lagos Suarez, A. Saradagi, V. Sumathy, S. Kotpalliwar, and G. Nikolakopoulos, "Curriculum-based sample efficient reinforcement learning for robust stabilization of a quadrotor," 2025, arXiv preprint.