ChatGPT

Surpassing AWS DeepRacer: Designing an Educational Drone RL Platform

Limitations of AWS DeepRacer in Educational Clarity

- **Hyperparameter Tuning Complexity:** AWS DeepRacer exposes several training hyperparameters (e.g. learning rate, batch size, number of epochs, entropy for exploration, discount factor, etc.), but understanding and tuning these is challenging for beginners. Official guides admit that hyperparameter tuning is essentially an *"iterative improvement through trial and error"* [1]. In practice, many users simply stick with the default values and only tweak one or two parameters (like learning rate or discount factor) if the model plateaus [2]. The console provides minimal intuitive explanation of what each parameter does or how it impacts learning, which means newcomers often lack clarity on how to fine-tune the training process.

- **Limited Evaluation Metrics & Feedback:** DeepRacer's built-in evaluation metrics are quite basic – mainly the percentage of track completed and the lap time or time to failure [3]. While these metrics tell you if the car can finish a race and how fast, they don't provide insight into *why* the model is performing a certain way or how it's learning. The platform does log detailed training data, but those logs are *"difficult to interpret"* without external tools [4]. For example, users often have to download logs and use custom notebooks to figure out how often the car went off-track or how the reward progressed over time [4]. This lack of clear, visual feedback in the interface makes it hard for learners to evaluate their model's improvement or diagnose problems. Beginners might only notice obvious outcomes (e.g. the car stays on track or not) but not understand the subtler evaluation metrics like stability of performance or policy improvement over episodes.

- **Insufficient RL Concept Emphasis:** DeepRacer succeeds at engaging users in a racing game, but it doesn't explicitly teach many underlying reinforcement learning concepts during that process. Key ideas—such as the exploration-exploitation tradeoff, the Markov property (memorylessness), and the role of the reward function—can remain murky. For instance, a novice may not realize the DeepRacer agent has **no memory** from one decision to the next (it chooses actions purely from the current camera image frame) [5]. This can cause confusion when the car makes odd decisions (it's not "planning ahead" because the algorithm doesn't recall past states). The platform doesn't proactively explain why the car might abruptly turn or fail (e.g. due to exploration noise or an overly greedy policy), nor does it teach the theory of *why* a certain reward function leads to certain behaviors. Much of the learning about RL concepts is left to external documentation or trial and error, rather than being demonstrated in-game. In short, DeepRacer offers a fun introduction but is somewhat a black box – learners aren't guided to understand what the algorithm is doing under the hood or the principles that govern its learning process.

Proposed Improvements in the Drone Platform for Clarity and Education

- **User-Friendly Hyperparameter Controls:** The drone platform's interface will demystify hyperparameters through intuitive design. Instead of just text boxes for values, we can provide sliders or simple drop-down profiles (e.g. "Conservative Learning", "Fast Learning") with clear explanations. Each hyperparameter setting will come with a tooltip in plain language (for example, *"Learning Rate – how big a step the model takes with each learning update; higher = learns faster but riskier"*). In Explorer mode, many of the low-level knobs might be hidden or auto-tuned, with perhaps only a high-level difficulty or training-duration setting. In Researcher mode, advanced users can adjust each parameter, but with guidance: the UI could show recommended ranges and even warn if a value is likely too high/low. An interactive tutorial could walk users through the effect of each hyperparameter (e.g. a small simulation or graph showing how a too-high learning rate causes oscillation). Moreover, the platform could include an *"auto-tune" assistant* – for instance, it could suggest "try lowering the learning rate" if it detects training has plateaued. Overall, these changes turn hyperparameter tuning from a blind trial-and-error exercise into a guided, educational experience where users learn what each setting does.

- **Enhanced Evaluation Metrics and Feedback:** The drone simulator will offer rich, visual feedback well beyond DeepRacer's lap times. After each training session or evaluation run, users will see a dashboard of metrics and insights. For example, we can graph the agent's **episode reward over time** to illustrate learning progress (with annotations like "plateau reached here"). We will still display success metrics (like completion rate and time), but alongside those the platform can show things like: consistency (e.g. variance in lap times over multiple trials), number of crashes or off-track incidents per run, and even section-by-section performance. One powerful idea is an **interactive replay** of the drone's run with analytics: the user can watch the drone's path and the system overlays info such as "speed here", "drone confused at this turn". If the drone consistently fails at a certain obstacle, the interface could highlight that section of the course on a map (pointing out, for example, "most crashes happen at Gate 3"). This kind of immediate, spatial feedback helps learners identify *why* the agent is failing. The platform can also incorporate side-by-side comparisons – e.g. compare your last two models' runs to see improvement. In short, every evaluation will come with explanatory metrics and visualizations, so users don't just see *that* one model is better, but understand *in what ways* and *why*. This closes the loop between making changes (like tweaking a reward or hyperparameter) and seeing their effects, reinforcing the learning process.

- **Integrated RL Concept Teaching Aids:** To truly educate, the drone platform will weave in reinforcement learning concepts at every step in an accessible way. This starts with onboarding: an Explorer-mode tutorial will visually explain the agent-environment loop (for example, using the drone avatar receiving "state" inputs, taking an action, and getting a reward in a cartoon diagram). As users train their models, the interface can surface key concepts contextually. Consider an **"RL Glossary hover"**: if the screen mentions terms like *episode*, *policy*, or *reward*, users can hover to get a kid-friendly explanation ("**Episode:** one attempt the drone makes to complete its task, after which learning happens"). When the drone is exploring, perhaps an on-screen indicator (like a little question mark icon when the agent is taking a random exploratory action) can hint at the exploration process, so learners realize why the drone sometimes deviates strangely. The platform could also include interactive mini-lessons: for example, a reward function builder that lets Explorer

users toggle goals (like “prefer staying centered” vs “prefer speed”) and *predict* what the drone will do, teaching the concept of reward shaping by example. In Researcher mode, the platform might offer deeper dives – e.g. an optional view of the neural network’s policy or value estimates changing over training, or links to explanations of the RL algorithm being used (PPO/SAC) and how it updates the model. By incorporating these aids, the platform ensures users don’t just train a drone *successfully*, but also internalize the fundamental RL ideas (why reward design matters, how the agent learns from trial and error, what “state” it actually perceives, etc.). This makes the learning experience much more explicit and pedagogical than DeepRacer’s largely implicit approach.

Core Learning Outcomes and User Goals – Explorer Mode

In **Explorer Mode**, designed for ages ~11–22 and beginners, the emphasis is on clarity, intuition, and engagement. The core learning outcomes and goals for users in this mode include:

- **Intuitive Understanding of RL Basics:** Explorer users should come away with a solid *conceptual* grasp of how reinforcement learning works, even if they aren’t digging into code. For example, by guiding a virtual drone, they intuitively learn that the drone improves through trial and error – it tries things, gets feedback (reward), and gradually does better. Key ideas like “the AI learns from mistakes” and “we must tell it what goal to aim for (via a reward)” will be understood at a gut level. The platform’s explanations and visual aids will ensure that an 18-year-old student or even a motivated 11-year-old can explain in their own words how the drone is learning to fly.

- **Engagement and Motivation through Gamification:** Explorer mode prioritizes keeping learners motivated and curious. Users will have fun, game-like challenges to tackle – for instance, a series of missions where the drone must deliver a package or navigate an obstacle course, with increasing difficulty. Achievements, badges, and friendly competitions are used to encourage progress. The goal is *engagement*; by framing learning as a game, students remain interested and are more likely to delve deeper. A core outcome is that users develop a positive attitude toward experimenting and learning from failure. Instead of being frustrated when the drone crashes, a student feels excited to tweak something and try again, treating it like an iterative puzzle.

- **Basic Problem-Solving and Experimentation Skills:** Even in beginner mode, users should gain confidence in applying problem-solving to improve their drone’s performance. This might mean experimenting with different reward choices or high-level settings and observing results. For example, an Explorer user might try a “reward for smooth flying” vs. a “reward for fast finish” and see which yields better behavior, thereby learning by experimentation. The outcome is that learners build a habit of forming hypotheses (“What if I reward speed more?”), testing them, and interpreting the outcome – a scientific thinking skill. They also learn basic troubleshooting: if the drone keeps crashing, look at the feedback (maybe it’s going too fast into turns) and adjust strategy. These are transferrable STEM skills nurtured by the platform.

- **Foundation for Advanced Learning:** While keeping things simple, Explorer mode should lay the groundwork so that interested users can transition to more advanced concepts later. For instance, a high-school user might not learn the math of Q-values in Explorer mode, but they will learn the *ideas* (like the drone tends to do what yields more reward). The goal is to spark curiosity: a learner might think, “I notice my drone behaves differently when I change this setting – I wonder why?” That curiosity and foundational understanding will prepare them to pursue more formal ML education or

step up to the Researcher mode. In summary, Explorer mode's success is measured in users who are both **enlightened and enthused**: they understand core ideas clearly and are excited to learn more.

Core Learning Outcomes and User Goals – Researcher Mode

The **Researcher Mode** is aimed at advanced students, hobbyists, or users who have some background (or have "graduated" from Explorer). It focuses on enabling deeper experimentation and producing outcomes that could be used in portfolios or real projects. Core goals and outcomes for this mode include:

- **Proficiency in Reinforcement Learning Experimentation:** Researcher users will gain hands-on experience in conducting RL experiments more akin to what a practitioner or researcher would do. They learn to handle the full workflow: setting up training with custom parameters, monitoring results, adjusting hypotheses, and iterating. For example, a user in Researcher mode might design a complex reward function from scratch or compare two different RL algorithms (if the platform allows, say, PPO vs. DQN in a drone task). The outcome is that they understand not just basic concepts, but also nuances like hyperparameter tuning, convergence behavior, and even techniques like domain randomization or overfitting avoidance. They are comfortable reading into graphs and logs to diagnose issues (e.g. "My drone's reward peaked and then collapsed – maybe it overfit or the exploration rate was too high").

- **Portfolio-Ready Projects and Artifacts:** A key goal for Researcher mode is enabling users to create something tangible and impressive that they can showcase academically or professionally. This could be a well-documented project of training a drone to perform a complex task (like autonomous indoor navigation or a stunt flight in simulation). The platform can facilitate this by allowing users to export videos of their trained drone in action, share their model and code, and even compile results into a report format. By the end, a user should have a "portfolio piece" – for instance, *"Autonomous Drone Navigation Project: using reinforcement learning I trained a drone to navigate a maze – here's how I did it and the results."* Achieving this requires the platform to support advanced customization (so their project can be unique) and to provide tools for visualization and explanation they can include in a write-up. The outcome is not just knowledge, but a concrete accomplishment that could be presented in a job interview or college application.

- **Deepened Theoretical and Practical Understanding:** Researcher mode, while still user-friendly, will expose the underlying machinery more than Explorer mode. Users in this mode are expected to learn more advanced RL concepts and even some theory, through practice. Outcomes include understanding things like why one algorithm might outperform another for a given problem, or how changes in the environment affect learning. They might experiment with hyperparameters deliberately (e.g. adjusting the entropy factor and observing how the policy's exploration changes), thus learning through direct observation. The platform might also encourage reading of provided advanced tutorials or whitepapers for those interested. Ultimately, a Researcher mode user should come away not only knowing *how* to make an RL agent work, but *why* it works – bridging that gap between using a tool and truly mastering the concepts behind it.

- **Preparation for Real-World Application:** Since the new platform plans sim-to-real capabilities, advanced users will also learn how to take simulation-trained models into real hardware (and the challenges therein). A core goal is to teach techniques like model transfer and testing in the real world safely. For example, a Researcher user might train in sim, then deploy their policy to an actual

drone (when the feature becomes available), learning about calibration or differences in dynamics. This experience is invaluable for those looking to go into robotics or AI careers. Even while currently "simulation-only," the platform can simulate the sim-to-real process (for instance, by adding noise or varying conditions) to teach users about robustness. An outcome would be that a user can confidently say: "I know how to build an RL model in simulation and what it takes to adapt it to a real environment," a skill DeepRacer users often lack given reports of sim-to-real failures [6]. This mode essentially grooms users for research, industry, or competition at a high level, by providing both knowledge and practical know-how.

Key Features of the New Drone RL Platform (Planned)

- **Real-Time Collaboration:** The drone platform supports real-time collaboration, allowing multiple users or a class to work together on the same simulation project simultaneously. For example, two students can log into a shared training session: one might adjust the reward function while the other watches the drone's behavior, or team members can divide tasks (one monitoring metrics, another tweaking parameters) all in real time. This is a significant upgrade in accessibility – it mirrors the way Google Docs allows collaboration, but applied to an RL experiment. The benefit is that learning becomes a social, interactive experience: users can learn from each other, brainstorm solutions together, and instructors can guide students live. In a classroom, a teacher could project the simulation while students suggest changes from their own devices. Real-time collaboration not only makes the platform more intuitive (you can ask a peer "did you see that? why did it do that?" and figure it out together), but also more accessible – a remote mentor could help a student troubleshoot a model in-session. DeepRacer does foster community through leaderboards, but it doesn't offer simultaneous collaboration on training. By contrast, our platform's collaborative mode will make reinforcement learning a team sport, reinforcing learning through communication and cooperation.

- **Explainable AI (XAI) Overlays:** To enhance understanding and trust in what the AI drone is "thinking," the platform will include explainability overlays during simulations. This feature might manifest as a visual highlight on the drone's camera feed or environment indicating what the model finds important at any given moment. For instance, if the drone uses a front camera, a saliency map can be superimposed on the view to show which parts of the image are influencing its decision most [7]. If the drone is primarily guided by LiDAR or other sensors, we could illustrate sensor readings or attention (perhaps highlighting the closest obstacle the drone is reacting to). The idea is to open the black box: as the drone approaches an obstacle, the user might see an overlay like a heat map on the obstacle, indicating "the drone sees this and is reacting." Another XAI overlay could be an arrow showing the drone's intended next direction or a ghost projection of the drone's predicted path given its current policy. By providing these explainable visuals, learners can connect the abstract concept of a neural network's decision to a concrete explanation (*"Oh, it turned because it 'thought' it was too close to the wall on the right"*). This goes beyond DeepRacer, which doesn't expose what features the model is attending to. Emphasizing explainability not only aids learning but also addresses the trust factor – users can debug and reason about their AI's behavior. Research in explainable RL for drones backs this approach, noting that highlighting critical input regions greatly helps in understanding agent decisions [7]. Ultimately, XAI overlays will make the learning experience more transparent and illuminating.

- **Seamless Sim-to-Real Learning Workflow:** The drone platform is designed with the end-goal of deploying on real drones, and it introduces features to smooth the transition from simulation to reality. This includes tools for **domain randomization** (automatically adding variability in simulation like changing lighting, wind, or physics parameters within reasonable bounds) so that the trained models are more robust when faced with the real world. There will also be guided checklists or wizards for transferring a model to a physical drone: for example, the platform could export the model in a ready-to-deploy format and provide an app or firmware for a compatible educational drone. Users are taught how to calibrate and test safely – the platform might simulate "real-world noise" and have a test mode where the drone's actions are verified in a safe manner before actual flight. By having an explicit sim-to-real workflow, the platform addresses a known limitation of DeepRacer: many DeepRacer users found that a model performing well in the simulator failed dramatically on a real track without further tuning [6]. Our platform will include lessons on *why* that happens and tools to prevent it (e.g. an *"evaluate in real environment"* step that highlights if the model might be too overfitted to the sim). Even if the platform at launch is simulation-only, this workflow mindset prepares users for eventually operating real drones. The benefit is that learning doesn't stop at simulation; users understand how RL can be applied in actual physical robotics, making the platform more practical and superior in educational value. They can see a clear path from a virtual experiment to a real-life application, which is inspiring and instructive.

- **Gamified Challenges and Community Engagement:** Similar to DeepRacer's competitive league but with a drone twist, the platform will feature gamified challenges and a community hub to keep users engaged. There will be a variety of challenge modes: time-trial races, agility courses, delivery missions, etc., each with leaderboards or achievement badges. For younger Explorer users, this might be single-player progress (unlocking levels or stories), whereas for advanced users, there will be global competitions or collaborative challenges (like a team-based drone race event). The key is that these challenges are not only fun but educationally structured – each challenge could highlight a concept (e.g. a challenge that requires optimizing for energy efficiency teaches about reward trade-offs). By participating, users join a broader community. They can share strategies on forums, compare what worked or didn't, and even swap reward functions or models. **Real-time collaboration** (mentioned above) ties into this, as users might form teams. We'll also integrate a system of recognition – top performers or creative solutions get showcased. This community aspect mirrors the success of AWS DeepRacer's global races in fostering learning [8], but we will extend it to the drone domain and ensure it's accessible (possibly with student-only leagues or beginner brackets to not discourage newcomers). The outcome is a vibrant learning community where users are motivated to improve not just by the software, but by the friendly competition and collaboration around them. Gamified progression and peer interaction thus make the platform not only more engaging than DeepRacer, but also continually stimulating – there's always a new goal or challenge to pursue, keeping learners hooked in a positive way [8].

Two Novel Features to Elevate the Drone Platform Beyond DeepRacer

1. **Custom Scenario Creator & Curriculum Mode:** To make the platform truly innovative, we will introduce a **custom scenario builder** that lets users create and share their own drone environments and curricula – a feature well beyond DeepRacer's fixed set of tracks. Using a simple drag-and-drop interface, learners could design an environment (for example, place waypoints, obstacles, hoops for

the drone to fly through, etc.) and set goals for the agent. This has dual benefits: it taps into creativity and also teaches about task design in reinforcement learning. An educator could create a progressive curriculum of scenarios, starting from an easy task (e.g. hovering or basic navigation) and ramping up to complex missions (like a maze or search-and-rescue simulation). The platform could offer templates to get started – say a "forest obstacle course" template that users can modify – ensuring that even novices can participate. **Curriculum Mode** would allow the drone to train on a sequence of scenarios that increase in difficulty, which demonstrates the concept of curriculum learning (training an agent on simpler tasks before harder ones). DeepRacer does not offer anything comparable – it confines users to pre-made tracks and single-task training. By empowering users to craft custom challenges, our platform encourages a deeper engagement: users learn how environment changes affect learning (for instance, *"what if I make the corridor narrower?"* – they can try it and see the impact on training). Moreover, sharing these scenarios in the community will spur collaborative learning; one student might design a challenge and others can attempt to solve it, akin to creating custom "levels" for an educational game. This feature is practical (built on simulation environment editing tools) yet visionary in educational impact: it transforms learners from consumers of training environments into creators, solidifying their understanding of RL problem setup and making the platform clearly more versatile and superior in educational value.

2. **AI Coach and Adaptive Training Guidance:** We plan to incorporate an **AI coach** – essentially an intelligent tutoring system – that continuously analyzes the user's training process and provides personalized guidance or tips. This virtual coach would be a first in this kind of platform, offering a level of mentorship DeepRacer never had. In practice, the AI coach could observe metrics and user decisions and intervene with suggestions or explanations. For example, if it detects that the drone's reward has flatlined for many episodes, it might prompt: *"It looks like the model isn't improving. This could mean it's stuck in a local optimum – perhaps try increasing the exploration rate or tweaking the reward function to encourage new behavior."* Or if a user's drone keeps crashing at the same spot, the coach might highlight that pattern: *"Your drone often crashes at the start of the turn. Maybe the reward is not incentivizing it to slow down for turns."* The coach can also quiz the user gently, e.g., *"Your drone is driving very cautiously. Do you know which parameter makes it more exploratory? (Hint: look at entropy)"* – turning moments of stagnation into teachable moments. Implementing this could involve a combination of rule-based triggers (for known common issues) and machine learning models that recognize more complex patterns in training data. Over time, the coach could even adapt to a user's skill level – giving more basic hints to novices but more technical insights to advanced users. The end result is an experience akin to having a personal tutor watching over your shoulder, except available 24/7 within the software. This feature is visionary yet practical: many of the suggestions the coach would give are ones an experienced human mentor might offer; we're automating that guidance to help users when they're stuck or to prevent them from developing misconceptions. DeepRacer users often had to turn to community forums or guess-and-check for such advice, but our platform will offer it in situ, in real time. By proactively teaching and correcting the user, the AI coach accelerates learning and makes the platform far more supportive and educational. This not only differentiates our platform as a superior learning tool, but it also lowers the entry barrier – users are less likely to get frustrated or lost, because the moment something is going wrong, the AI coach will be there to nudge them in the right direction with a helpful explanation.

Sources:

- Marsman, S. – *"How I Got Into The Top 2% in AWS DeepRacer"* (Medium, 2023) – on default hyperparameters and log interpretation [2] [4].
- Reddit (r/DeepRacer) – discussion of DeepRacer's limitations (sim-to-real gap, memoryless agent) [6] [5].
- AWS DeepRacer Developer Guide – on evaluation metrics (track completion, time) [3].
- Dev.to (Vivek Raja, 2020) – on DeepRacer hyperparameters and tuning by trial-and-error [9] [10].
- Research by I. Bozcan (2023) – demonstrating saliency maps to explain a drone's DRL policy decisions [7].
- Scaler Topics (2024) – describing DeepRacer's community and gamified learning approach [8].

---

[1] [9] [10] Fine-tuning the performance of the DeepRacer model - DEV Community
https://dev.to/aws-builders/fine-tuning-the-performance-of-the-model-4pjo

[2] [4] How I Got Into The Top 2% In AWS DeepRacer | by Sam Marsman | Medium
https://medium.com/@marsmans/how-i-got-into-the-top-2-in-aws-deepracer-32127a364212

[3] Evaluate your AWS DeepRacer models in simulation - AWS DeepRacer
https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-get-started-test-in-simulator.html

[5] [6] A few notes from re:Invent about deep racers : r/DeepRacer
https://www.reddit.com/r/DeepRacer/comments/a1ysj6/a_few_notes_from_reinvent_about_deep_racers/

[7] upcommons.upc.edu
https://upcommons.upc.edu/bitstream/handle/2117/399125/3152.pdf;jsessionid=D9220AF29453CBB8868DC8216FEE32AB?sequence=1

[8] AWS DeepRacer - Scaler Topics
https://www.scaler.com/topics/aws-deepracer/