**Project Evolution and Final Implementation of LunarLander-v3 PPO Training Framework**

The development of the LunarLander-v3 reinforcement learning framework using Proximal Policy Optimization (PPO) underwent multiple refinements, with each iteration addressing critical inefficiencies and suboptimal behaviors observed during training. The goal was to design an adaptive and efficient training strategy that maximized cumulative rewards while ensuring stable learning dynamics and minimizing unnecessary fuel expenditure.

**Early Training and Initial Limitations**

Initially, the model employed fixed hyperparameters, including a constant entropy coefficient (ent\_coef), learning rate (lr), value function coefficient (vf\_coef), and training epochs (n\_epochs). The standard PPO configuration was used with gamma=0.999 for long-term planning and n\_steps=512 for batch updates. However, several limitations emerged:

1. **Inefficient Landing Strategy:** The agent avoided crashing but did so inefficiently, either hovering aimlessly until the maximum episode length was reached or landing extremely slowly, unnecessarily prolonging episodes.
2. **Premature Learning Rate Decay:** The initial fixed decay schedule restricted learning before the lander had fully optimized its descent strategy.
3. **Lack of Adaptation to Training Progress:** Fixed hyperparameters resulted in either excessive exploration early on or insufficient refinement later.

**Refining Hyperparameter Adjustments and Training Dynamics**

To address these issues, a custom hyperparameter adjustment callback was implemented, dynamically adjusting key parameters (ent\_coef, vf\_coef, lr, and n\_epochs) based on training progress and performance trends.

* **Adaptive Exploration (ent\_coef):** If the agent took too long to land (episode length exceeding a predefined threshold), entropy was increased to encourage faster, more decisive actions.
* **Reward-Based Refinement (vf\_coef):** As rewards improved, the model shifted weight toward the value function, emphasizing long-term planning.
* **Learning Rate Scheduling (lr):** Learning rate decay was postponed until the lander demonstrated stable flight behavior, preventing premature reductions in policy improvement.

Additionally, we refined the learning rate strategy. Instead of a linear decay, a logarithmic schedule was introduced, ensuring smooth transitions between exploration and exploitation:

* Learning rate reduction began only when the agent consistently survived for max\_episode\_steps.
* A log-based function was used to reduce lr smoothly over time, preventing abrupt drops that could disrupt training.

Experiments with dynamically adjusting gamma led to unstable learning—likely due to optimizer resets interfering with PPO’s momentum-based updates. Unlike lr, ent\_coef, and vf\_coef, gamma did not adapt well to frequent changes. As a result, we fixed gamma at 0.999, maintaining long-term planning stability.

**Ensuring Persistence and Efficient Resource Utilization**

Restarting training initially reset all hyperparameter adjustments, erasing prior optimizations. To resolve this:

* A JSON-based state-saving mechanism was introduced.
* Hyperparameters (ent\_coef, vf\_coef, lr, n\_epochs) were saved and reloaded between training sessions.
* Optimizer states were preserved, preventing learning disruptions from momentum resets.

We also modified max\_episode\_steps to simulate a fuel constraint, enforcing efficiency by capping max\_episode\_steps at 250. This discouraged excessive hovering and incentivized faster, more optimal landings. As the agent improved, max\_episode\_steps was further reduced to ensure efficiency.

**Parallelized Training and Asynchronous Evaluation**

To optimize training efficiency, we deployed 64 parallel environments using make\_vec\_env, significantly improving data collection rates.

To prevent evaluation from interfering with training, evaluation was moved to a separate process, ensuring uninterrupted policy learning. Real-time visualization provided ongoing insights into agent behavior without introducing computational slowdowns.

**Optimization with AdamW and Structured Learning Rate Control**

The final iteration incorporated **AdamW (Adam with weight decay)**, which improved generalization and prevented overfitting. This optimization choice, coupled with structured learning rate decay, ensured:

* More stable updates.
* Reduction in overfitting risks.
* A smoother transition from exploration to exploitation through log-based decay control.

Additionally, batch size (4096) and n\_epochs were dynamically adjusted to balance learning efficiency and computational performance.

A key feature was the introduction of a **moving average window for training stability**. The callback monitored recent episode statistics over a dynamically adjusted window and modified hyperparameters based on the lander’s landing efficiency and reward trends.

* If the episode length was 250 (max for us) and reward exceeded 120, **ent\_coef** was increased to promote exploratory behavior, and **learning rate** was adjusted using a logarithmic function, as we consider that the model learned to hover but needs some exploration to discover efficient landing.
* If episode rewards consistently exceeded 200, **vf\_coef and n\_epochs** were increased, shifting training toward policy refinement.
* If rewards dropped below 175, hyperparameters were reset (more exploration) to ensure robustness against instability.

**Final Performance Metrics and Observations**

With these improvements, the final model achieved stable landings with scores exceeding **300+**, demonstrating:

* Optimized policy efficiency with minimal fuel waste.
* Stable training dynamics through structured hyperparameter control.
* Rapid convergence via adaptive learning mechanisms.

The final model was trained over **10 million timesteps**, with a training pipeline that dynamically reloaded hyperparameters and adjusted training parameters based on model performance trends. To maximize GPU utilization, CUDA optimizations were introduced, including:

* **Per-process memory allocation tuning** to improve stability.
* **Multi-threaded evaluation processes** for efficient monitoring.
* **GPU precision optimizations** to accelerate computation without losing precision.

**Conclusion**

This project exemplifies a structured approach to reinforcement learning optimization, addressing issues of efficiency, exploration stability, and long-term planning. Through **dynamic hyperparameter tuning, structured learning rate control, optimizer improvements, and fuel constraint simulation**, the model achieves optimal landing strategies while dynamically balancing exploration and exploitation.

The implementation integrates **real-time evaluation, adaptive training mechanisms, and GPU acceleration**, demonstrating a scalable reinforcement learning pipeline for optimizing control policies in continuous action spaces.