Lunar Lander - Project Summary (Revised)

1. Introduction

The LunarLander-v3 environment, provided by Gymnasium, tasks an agent with safely landing a spacecraft on a designated target pad. It features an 8-dimensional continuous state vector (x/y position, x/y velocity, angle, angular velocity, left/right leg contact) and four discrete actions: do nothing, fire left engine, fire main engine, and fire right engine. Compared to high-dimensional tasks like CarRacing, LunarLander's low-dimensional input enables rapid experimentation and in-depth algorithmic analysis, making it ideal for comparative studies of reinforcement learning (RL) algorithms.

2. Objective

This project compares four variants of the Deep Q-Network (DQN) family—Vanilla DQN, Double DQN, Dueling DQN, and Prioritized Experience Replay DQN (PER-DQN)—against the Proximal Policy Optimization (PPO) algorithm from Stable Baselines3. We evaluate each algorithm on:

- Learning speed for a stable landing policy
- Landing precision (distance to pad at touchdown)
- Fuel efficiency (remaining fuel at episode end)
- Training stability (reward variance across training)
- Performance milestones over selected episode counts

3. Scope

- **Environment**: Gymnasium LunarLander-v3
- **State Space**: 8-dimensional continuous vector
- **Action Space**: 4 discrete actions
- **Algorithms**: Vanilla DQN, Double DQN, Dueling DQN, PER-DQN, PPO
- **Training Runs**: Up to 250 episodes per algorithm with 1000 max steps, 3 random seeds
- **Evaluation**: 10 episodes per seed (30 runs total), 250-step cap per episode

The training duration of 250 episodes was chosen based on preliminary experiments indicating that most algorithms achieve stable performance within this timeframe, balancing computational constraints with convergence needs.

4. Milestone Evaluations

Performance is recorded at five key milestones during training: 10, 50, 100, 150, and 250 episodes. These checkpoints capture the progression of learning and allow for comparative analysis of algorithmic efficiency.

5. Experimental Design

- **Hyperparameters**: Hyperparameters were set using defaults from Stable Baselines3 (v1.5) for PPO and tuned manually for DQN variants to ensure stable training. Key parameters include:
    - **Vanilla DQN**: Learning rate = 0.001, discount factor (γ) = 0.99, batch size = 64, exploration rate (ε) decaying from 1.0 to 0.01 over 100 episodes.
    - **Double DQN**: Same as Vanilla DQN, with target network update every 1000 steps.
    - **Dueling DQN**: Same as Vanilla DQN, with dueling architecture splitting Q-values into state and advantage streams.
    - **PER-DQN**: Same as Vanilla DQN, with prioritization parameter $\alpha = 0.6$ and β annealed from 0.4 to 1.0.
    - **PPO**: Learning rate = 0.0003, clip range = 0.2, n_steps = 2048, as per Stable Baselines3 defaults. Detailed hyperparameter tables are provided in the project's Kaggle Notebooks.
- **Standard Evaluation**: After full training, run 10 episodes per seed (3 seeds) under a deterministic policy, recording the first 250 steps and aggregate metrics.

- **Milestone Evaluation**: At each milestone (10, 50, 100, 150, 250 episodes), capture 250-step evaluation videos and log performance metrics.

6. Metrics

The following metrics are computed over 30 runs (5 episodes × 3 seeds):

- **Mean ± Std Episodic Reward**: Computed using `np.mean` and `np.std` over runs.
- **Success Rate**: Percentage of landings within the target pad.
- **Landing Precision**: Mean ± Std distance from the pad at touchdown.
- **Fuel Efficiency**: Mean remaining fuel at episode end.
- **Sample Efficiency**: Total steps to reach a reward threshold of 200, a standard benchmark for LunarLander-v3 indicating a solved task, as per Gymnasium documentation.
- **Training Stability**: Standard deviation of rewards over a sliding window of the last 50 training episodes, supplemented by the variance of the final 10 episodes to capture both dynamic and terminal stability trends.

7. Analysis & Visualization

- Learning curves across full training and at milestones
- Scatter plots of fuel vs. reward
- Histograms of angular distributions
- Q-value distributions (for DQN variants)
- Trajectory plots showing descent paths
- Compute cost: wall-clock time and peak GPU memory usage

8. Deliverables

1. ACM-style PDF report (≥ 4 double-column pages) - this document.
2. Kaggle Notebooks: with working code, hyperparameter and library versions
3. Raw logs (CSV/NPY) of per-episode metrics and milestone snapshots.
4. Evaluation videos (standard and milestone clips).
5. Plots and CSVs comparing agents on all metrics.
6. Output Directory layout

```
Project_root/
├── Logs/full_training.csv
├── Videos/
│   ├── agent_*/
│   │   ├── standard/
│   │   ├── milestones/
│   │   │   ├── milestone_10.mp4
│   │   │   ├── milestone_50.mp4
│   │   │   ├── milestone_100.mp4
│   │   │   ├── milestone_150.mp4
│   │   │   ├── milestone_250.mp4
├── Plots/learning_curves.png
├── Plots/learning_curves_milestones.png
├── Plots/... other plots ...
```

9. Limitations and Future Work

This study has several limitations. First, the training duration was capped at 250 episodes due to computational and time constraints, which may not guarantee full convergence for all algorithms, particularly Vanilla DQN. Second, hyperparameter tuning was limited to manual adjustments, potentially missing optimal configurations. Third, statistical significance tests were not conducted due to the submission deadline, limiting claims about performance differences.

Future work could extend training to ensure convergence, incorporate automated hyperparameter optimization (e.g., using Optuna), and apply statistical tests (e.g., t-tests) to validate performance differences. Additionally, testing on other Gymnasium environments or including algorithms like Advantage Actor-Critic (A2C) could broaden the study's scope and generalizability.