PROJECT PROPOSAL

By, Adnan Amir & Poojit Maddineni

THE PROBLEM:

Description

Autonomous robots in warehouses and logistics need efficient navigation to transport items while avoiding obstacles. This project compares two leading RL algorithms—Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO)—for a differential drive robot navigating a grid-based environment in Isaac Sim. The robot learns to move from a start to a goal position while optimizing motion strategies. We will evaluate both algorithms on learning efficiency, stability, and generalization. Our findings will provide insights into the trade-offs between off-policy (SAC) and on-policy (PPO) learning, contributing to more robust robotic navigation in real-world applications.

Formulation

- **Input:** The environment state (e.g., grid cell location, robot orientation, velocity) and goal coordinates.
- **Output:** Action commands (e.g., linear and angular velocities) for the differential drive robot to move from a designated start to a goal position.
- **Objective:** The robot must navigate efficiently across the grid while avoiding obstacles (if introduced) and generalizing its learned policy across varied grid configurations.

Ideal Outcome:
We expect to demonstrate a clear performance comparison between SAC and PPO in terms of:

- Convergence speed and sample efficiency.
- Stability of the learning process.
- Robustness and generalization of the policy to variations in the environment.
- Overall task success rate (percentage of episodes where the robot successfully reaches the goal).

THE DETAILS:

Algorithms & Techniques:

- **Algorithms:**
    - *Soft Actor-Critic (SAC):* Off-policy, entropy-regularized actor-critic method.
    - *Proximal Policy Optimization (PPO):* On-policy method using a clipped surrogate objective for stable updates.
- **Generalization Techniques(Optional – depending on time):**
    - Explore data augmentation and domain randomization to improve model robustness.

Tools, Libraries, and Platforms:

- **Simulation Platform:** NVIDIA Isaac Sim for high-fidelity robotic simulation (could switch to MuJoCo).
- **Development Environment:** Python with libraries such as PyTorch for deep learning components and maybe gymnasium for the environment stuff.
- **Robotics Frameworks:** Differential drive dynamics and stage environment models, which will be sourced online and integrated into Isaac Sim.

Week-by-Week Plan:

- **Week 1 (This week):**
    - *Environment Setup:*
        - Gather resources for the differential drive model and stage environment.
        - Begin designing the grid world in Isaac Sim, defining start/goal states and obstacles if applicable.
    - *Literature Review:*
        - Review key papers and documentation for SAC, PPO, and Isaac Sim.
- **Week 2:**
    - *Algorithm Implementation (A):*
        - Develop initial versions of SAC and PPO from scratch.
        - Integrate algorithms with a basic simulation environment.
    - *Environment Refinement (P):*
        - Enhance the simulation environment in Isaac Sim for realistic dynamics.
- **Week 3:**

- *Integration & Experimentation:*
  - Run initial experiments on both algorithms.
  - Begin testing generalization techniques (e.g., domain randomization) by varying grid configurations.
- *Data Collection:*
  - Record metrics such as cumulative rewards, success rate, and convergence speed.

- **Week 4:**
  - *Analysis:*
    - Perform detailed comparisons on the effects of different hyperparameters and generalization methods.
  - *Reporting:*
    - Compile results, create visualizations, and prepare the final project report.
    - Conduct a final review and prepare a demonstration of the simulation and results.

TEAM MEMBERS & DIVISION OF LABOR:

- **Poojit (P):** Responsible for designing and implementing the simulation environment in Isaac Sim. This includes sourcing and integrating the differential drive robot model and stage environment, and setting up the grid with start and goal states.
- **Adnan (A):** Responsible for coding the RL algorithms from scratch—specifically, implementing Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO)—and integrating them with the simulation environment.