Title: *Language-Guided Reward Generation for Model-Based Robot Manipulation*

Team Members: Nana Anikuabe, Jebasanthi Velayudhan

Custom or Default Project: Custom

---

1. Objective

Our project aims to explore the integration of large language models (LLMs) into reinforcement learning (RL) frameworks by enabling LLMs to generate reward functions from natural language task descriptions. Specifically, we will implement a model-based RL approach for robot manipulation tasks in a simulated environment, where the reward functions are derived from LLM outputs. This approach seeks to reduce manual reward engineering and enhance the adaptability of RL agents to diverse tasks.

---

2. Related Work

Our project draws on two key threads of recent research: sample-efficient model-based reinforcement learning for robotic manipulation, and the emerging use of large language models (LLMs) in autonomous reward generation.

The first is grounded in the work *"Learning More With Less: Sample Efficient Dynamics Learning and Model-Based RL for Loco-Manipulation"* (Arxiv:2501.10499), which introduces a hybrid approach combining structured kinematic modeling and learned residual dynamics using Bayesian neural networks. This hybrid model allows efficient learning of robot dynamics for manipulation tasks with relatively few samples, and is demonstrated using Boston Dynamics' Spot robot with a mounted arm. The core idea is to blend domain knowledge with data-driven uncertainty-aware learning, which significantly enhances sample efficiency and stability in model-based policy learning.

The second stream is informed by *"ARCHIE: Autonomous Reinforcement Learning with GPT-4"* (Arxiv:2503.04280), which leverages GPT-4 to translate natural language task descriptions into shaped reward functions. ARCHIE demonstrates that GPT-4 can autonomously bootstrap reinforcement learning pipelines by generating reward code snippets that an RL agent can optimize against, enabling task specification through natural language without manual reward engineering. This concept aligns with a growing interest in using LLMs to bridge high-level human intent and low-level control in autonomous agents.

Our project combines these two lines of work by embedding a language-guided reward design module (based on ARCHIE) into a model-based RL framework (inspired by the Spot dynamics paper), evaluated entirely in simulation. To our knowledge, this specific fusion—LLM-based

reward generation within a model-based control loop for complex robot manipulation—has not yet been explored.

---

3. Technical Outline

Our technical approach is broken into three core components: dynamics model learning, language-based reward generation, and policy learning with planning.

1. **Simulation Environment**:
   We will build a simulated robot manipulation environment using either MuJoCo or Isaac Gym, replicating tasks like pick-and-place or reach-to-target. The agent will operate a manipulator arm with basic physics constraints, within a workspace containing a target object and goal region. The environment will expose state observations (e.g., joint angles, object poses) and accept low-level or joint-space actions.
2. **Dynamics Learning**:
   Following the 2501.10499 approach, we will implement a hybrid dynamics model:
   - A deterministic, analytical kinematic model encodes known parts of the robot's motion.
   - A Bayesian Neural Network (BNN) learns the residual dynamics — capturing model mismatch, noise, or unmodeled interactions (e.g., object dynamics).
     This dynamics model will be trained using transitions from the simulated environment. Once trained, it will be used for model-predictive control (MPC) or planning-based policy optimization.
3. **LLM-Guided Reward Generation**:
   We will use GPT-4 (via API) to generate reward functions from structured natural language prompts (e.g., "The goal is to place the red block inside the blue box"). The LLM will return a Python reward function that can be plugged directly into the environment. We'll define a prompt template and a validation pipeline to verify that the generated code compiles and matches intended semantics. This reward will then be used to train the agent via model-based RL.
4. **Evaluation Plan**:
   We will compare the LLM-generated rewards with manually defined baseline rewards across multiple tasks:
   - Success rate (task completion)
   - Sample efficiency (episodes to convergence)
   - Policy quality (trajectory smoothness, stability)
     We will also evaluate the semantic fidelity of LLM rewards — e.g., whether a prompt like "place the object gently" results in reward shaping that promotes smooth trajectories.
5. **Ablation & Analysis**:
   We'll conduct ablation studies to isolate the effects of:
   - LLM-based reward generation vs fixed/manual reward functions

    - Uncertainty-aware dynamics models vs purely deterministic models
      We will document both success and failure modes, focusing on generalization and brittleness of LLM reward interpretation.

---

4. Team Contributions

- **Nana**:
    - Responsible for setting up the simulation environment
    - implementing the model-based RL algorithm
    - integrating the LLM for reward generation.
- **Jeba**:
    - Focuses on designing the natural language prompts for the LLM
    - parsing and validating the generated reward functions
    - conducting performance evaluations and analyses.

This project aligns with the CS224R requirements by introducing a novel integration of LLMs into RL for robot manipulation, offering insights into the potential of language-guided reward generation.