**PongRL – Design Doc**

**Project Overview**

Two paddles will be controlled by reinforcement learning agents and the goal is for them to learn – in real-time and simultaneously - to play and rally. I will explore multiple algorithms including Vanilla Policy Gradients (REINFORCE), Advantage Actor-Critic (A2C), Proximal Policy Optimization (PPO), and Deep Q Networks (DQN), to determine the most efficient learning algorithm for the environment. The Pong game will be implemented using PyGame, while PyTorch will be employed for neural network design and optimization. The reinforcement learning algorithms will be implemented from scratch to force a deeper understanding of the underlying principles.

The point of this project isn’t necessarily about creating the best pong intelligence, but learning and applying cool RL algorithms to get the neural networks to learn effectively. I plan on logging progress and results of each algorithm in the project’s readme file to demonstrate the effectiveness of various reinforcement learning techniques.

Vanilla Policy Gradients or REINFORCE – Directly learns the policy (strategy) that the agent should follow. Policy is trained by computing gradient of episode return with respect to network parameters.

Advantage Actor-Critic – Critic neural net learns to predict the value of the current state. An advantage that quantifies the value of taking the certain action is computed and the gradient of that with respect to the policy parameters is used to then train the policy simultaneously with the critic.

Proximal Policy Optimization – Like VPG, but ensures that policy updates aren’t too aggressive. This helps ensure that training is stable and reduces variance.

Deep Q Networks (DQN) – This algorithm only trains a value network that predicts the value of an action taken in a particular state. It then uses a greedy method to take actions.

The main challenge here is combatting the credit assignment problem. How do you properly reinforce actions that only see a reward after many time steps? In the Pong case, not letting the ball get past you once, won’t immediately reward you, but will help you not lose the round. This makes rewarding each of the actions taken especially difficult as it is not about the specific action but what it contributes to. Another problem is variance. The agent will start off taking random actions and that will make it especially difficult to reward.

**Why this?**

This project is able to combine a game and cool algorithms. I especially like learning about the math and intuition behind learning algorithms and I thought this would be doable and challenging while being really cool to observe and learn about.