# Phase 2: Mid-Status Report - Snake AI Using Reinforcement Learning

## Research Question

Can a reinforcement learning agent, trained using model-free algorithms like Deep Q-Learning and Proximal Policy Optimization (PPO), learn to play the Snake game effectively and generalize well over time?

## Introduction

Reinforcement Learning (RL) has emerged as a powerful technique for sequential decision-making tasks. One classic challenge in RL is the Snake game, a grid-based survival game where the agent must learn to eat food while avoiding walls and its own tail. Although the original scope focused on Deep Q-Learning, the project was expanded to include Proximal Policy Optimization (PPO) to enable a richer comparison of reinforcement learning strategies. The problem is significant as it provides a controlled environment to test learning efficiency, policy generalization, and sample complexity of different RL techniques.

## Comparison Table

|  |  |  |
| --- | --- | --- |
| **Feature** | **Existing RL Mode** | **Our Custom Model** |
| Training Time | Already trained | You must train it |
| Customization | Limited | Fully customizable |
| Learning Experience | Minimal | Deep hands-on learning |
| Performance (initially | Usually high | Starts low, can improve |
| Integration with game UI | Needs adaptation | Built natively with your game |

## When to use which?

**Use an existing model if**:

If you want a strong Snake AI quickly for demonstration or benchmarking.

**Use our own model if**:

If You want to learn RL, experiment with game mechanics, or develop a custom AI that fits your unique Snake game variant.

## Related Literature

1. Mnih et al. (2015), "Human-level control through deep reinforcement learning" - <https://www.nature.com/articles/nature14236>

2. Schulman et al. (2017), "Proximal Policy Optimization Algorithms" - <https://arxiv.org/abs/1707.06347>

3. Patrick Loeber's Snake AI with PyTorch & Pygame - <https://www.youtube.com/watch?v=L8ypSXwyBds>

## Approach

The Snake game was developed using Pygame. Two separate agents were created:  
- DQN Agent (value-based): Uses a feedforward neural network to approximate Q-values.  
- PPO Agent (policy gradient-based): Uses a stochastic policy updated through clipped surrogate objective.  
Both agents observe an 11-dimensional state space and choose from three discrete actions: straight, left, or right. Training metrics such as score, average reward, and episode logs are written to CSV files. Evaluation mode was implemented to observe trained agents without additional learning.

## Achievements

- Fully functional Snake environment  
- DQN and PPO agents implemented and trained  
- Training logs and score tracking via CSV  
- Evaluation script to replay models  
- Comparison graphs generated using matplotlib

## Challenges & Adjustments

- PPO agent integration required custom training loop and log probability tracking  
- Snake game input format needed adaptation between agents  
- Evaluation mode required saving and loading separate model architectures

## GitHub Repository

<https://github.com/GaneshMunagala714/Snake_Game_AI>

**Sample Outputs** :

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.