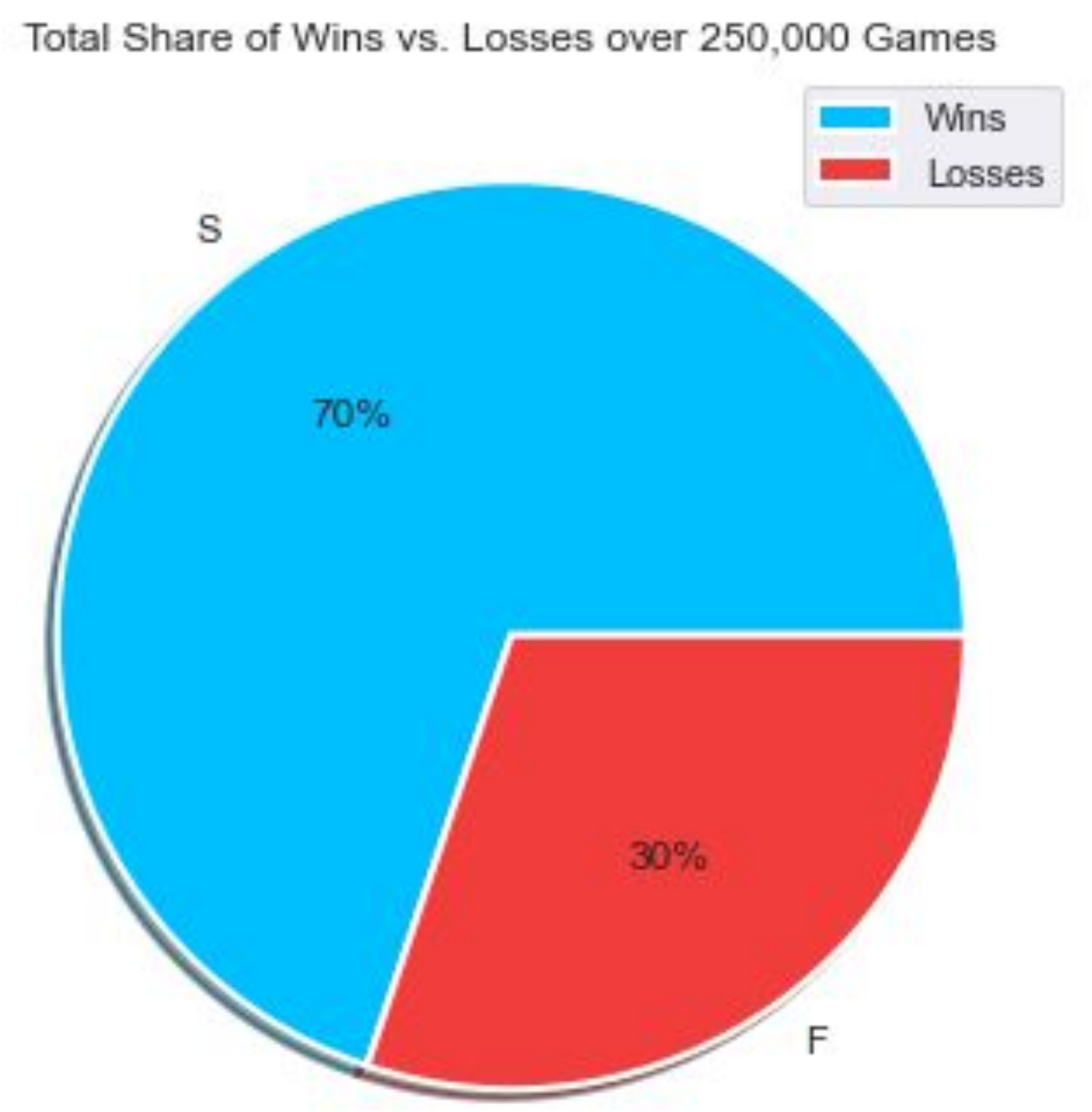
Millersville University

Created Spring 2022

A Multi-Agent Reinforcement Learning System for the Turn Based Strategy Game, *Fire Emblem*

Joshua Carney
Dr. Stephanie Schwartz
Dr. Chris Cain



Introduction

Fire Emblem is a tactical, turn based strategy game developed by Intelligent Systems and published by Nintendo. The first game was released in 1990, with the latest entry in the series being released in 2019.

Gameplay revolves around moving characters in a grid based environment to engage in combat against an enemy team, use items, or strategically position units. A notable aspect of the gameplay is the idea of "permanent death"; once a character dies, they remain dead for the entire duration of the game (as opposed to one, singular battle)

Algorithm Selection

We will consider each character within the game to be its own individual agent, seeking to maximize its own outcome within the game. This means the agents will not collaborate or draw consensus on problems. Agents will utilize Q-Learning for action selection, being trained for roughly 250,000 simulated games (~8 hours)

Agents will use search heuristic algorithms to determine which tile to move to within the board given the action that the agent selected.

Q-Learning Algorithm

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$$

Tile Selection Heuristic

$$H_{x,y} = (e_{dc} - e_{dxy})(h - Z) + \varphi d_{xy} a_{xy}$$

Target Selection Heuristic

$$H_{a,d} = (d_a + 1)(m_a h_a + m_a c_a) - \tau[(d_a + 1)(m_d h_d + m_d c_d)]$$

Q-Table

Q-Learning is implemented with a Q-Table, or a table of state-action pairs. At the intersection of these pairs is our expected valuation (or Q value) for choosing an action a_t in state s_t. When it comes time for a unit to choose an action, they either:

- Exploit: find the argmax of valid actions in state s_t
- Explore: select a random valid action in state s_t

Action Space

State Space	0	1	2
0,0	1.46	32.81	-34.39
0,1	-4.78	4.41	-30.99
0,2	0.26	2.42	-14.77
…	…	…	…
9,9	4.32	0.21	21.05

State Representation

State representation chosen is **E x N:**

- **E**: How many enemy units could attack this unit if they wanted to. [0, 10)
- **N**: This unit's health percentage in increments of 10% as an integer. [0, 10)

Reward Table

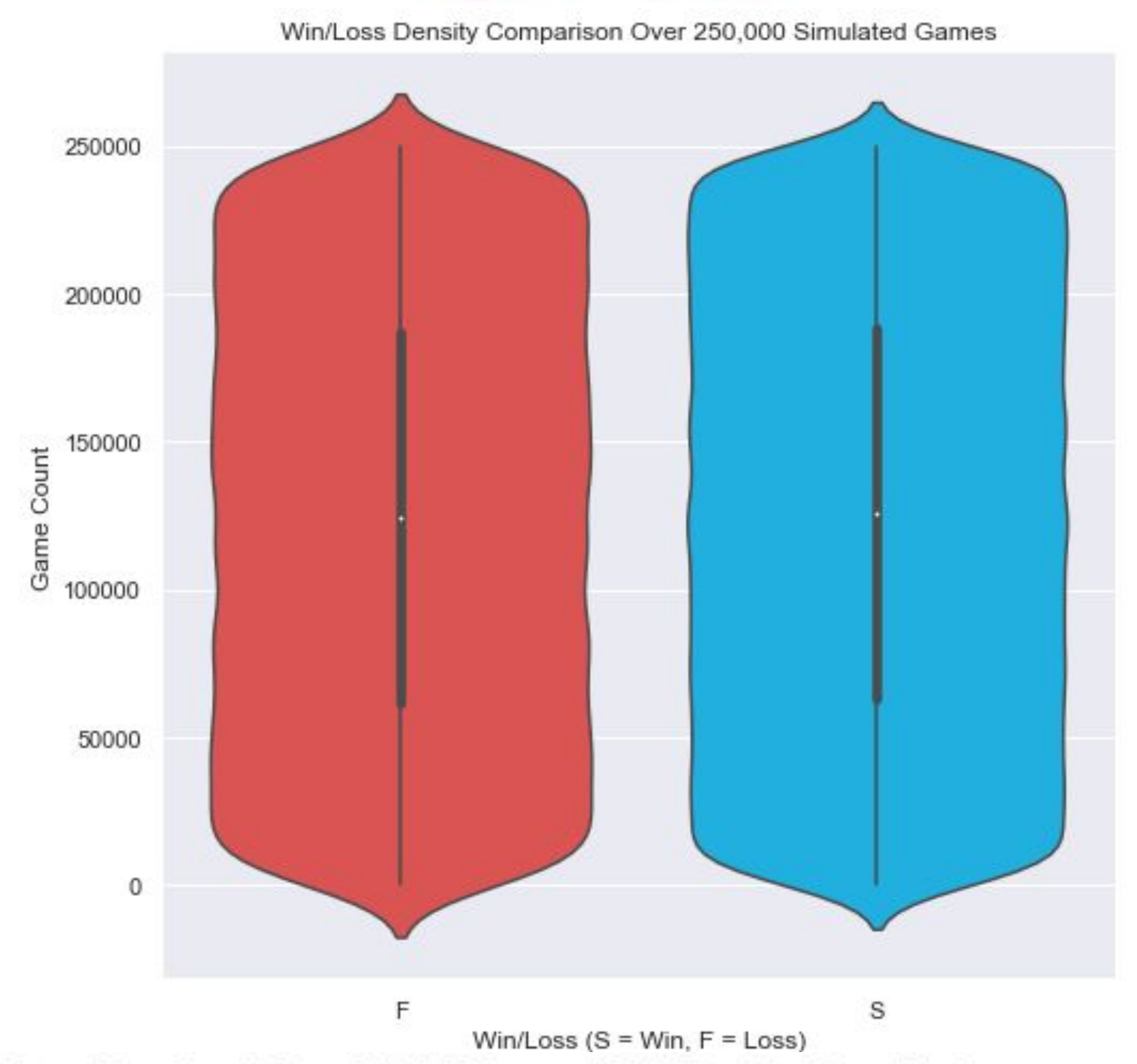
When taking action a_t in state s_t the agent receives a reward R_{t+1} from the environment. This is how the agent updates the q values within the q-table.

Description	Reward
Dying (non-terminal)	-50.0
Dying (terminal)	-75.0
Killing an enemy	5.0
Choosing wait	0.0
Healing self with item	Varies; directly proportional to healing done

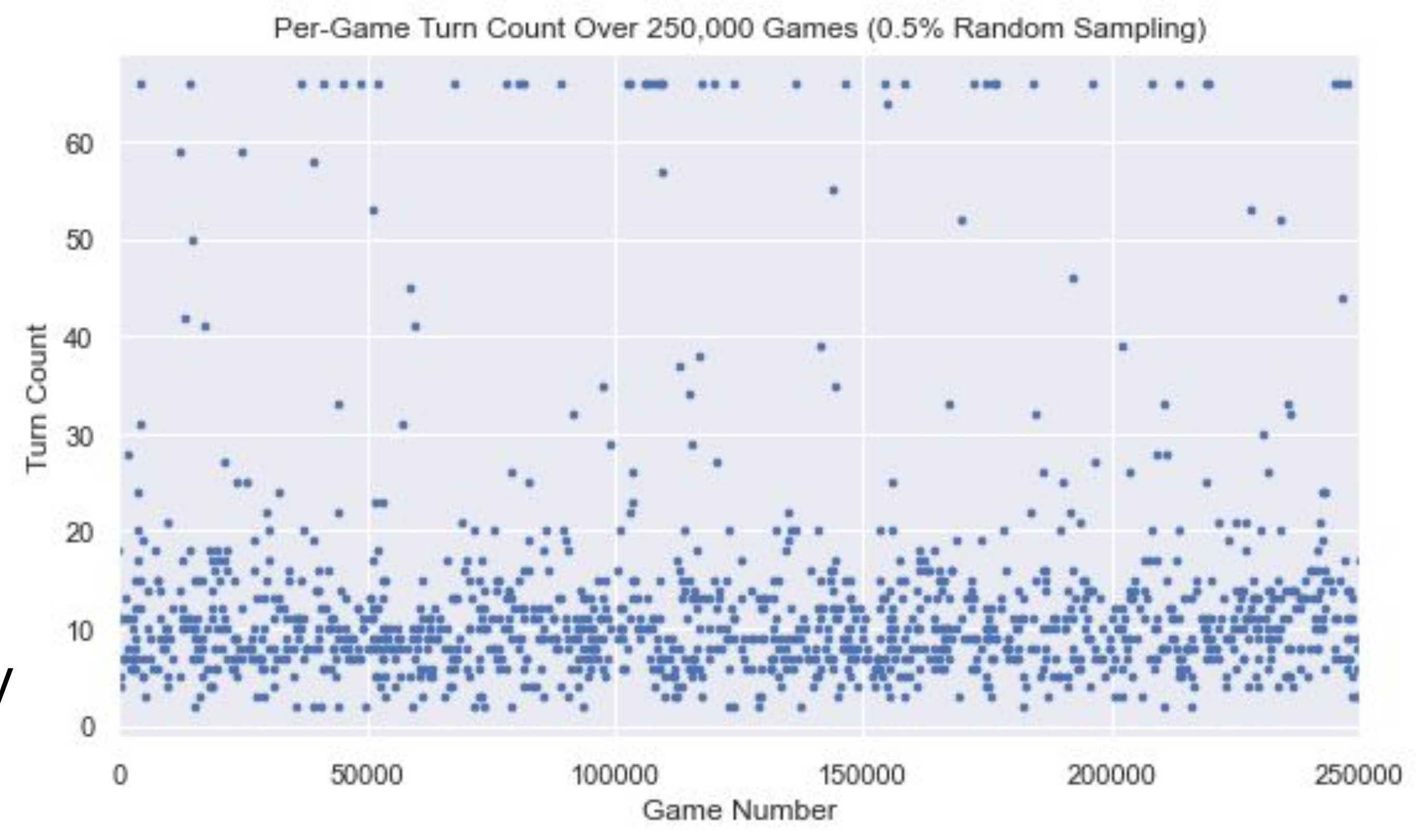
Result

After 250,000 games played, the agents performance does not seem to improve over time. Although the agents win more overall, the density of their per game turn count, survival rank, and victories shows no trend, which implies that no learning is occurring.

There could be several reasons for the lack of learning with the agents:



- Heuristics are flawed
- There are balance issues, which is why agents win/lose randomly.
- Off-policy learning may not be the correct approach
- Human error; something may be programmed incorrectly



Next Steps and Improvements

Much more could be done to investigate the lackluster performance of the agents:

- Have agents collaborate and draw consensus on aspects of the game. For example, move order and attack order frequently matters, but currently, agents act in an arbitrary order. Having them draw consensus could allow for performance improvements
- Compare the performance of other reinforcement learning algorithms such as SARSA or TD λ.
- Investigate alternative, more robust systems for movement selection outside of heuristic search.
- Execute performance tests by having the agent execute actual gameplay via emulation.