Reinforcement Learning in Mortal Kombat II

Ana Kolovska
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje, North Macedonia
ana.kolovska@students.finki.ukim.mk

February 6, 2026

Abstract

This paper presents a comparative study of multiple Deep Reinforcement Learning (DRL) algorithms applied to a competitive fighting game environment based on Mortal Kombat II. A custom two-agent environment was developed using an emulator-based setup, allowing agents to interact directly with the game and observe state information via RAM memory inspection. The algorithms DQN, DDQN, Dueling DDQN, A2C, and PPO were implemented and trained for 1000 episodes each. Their performance was evaluated through 10 head-to-head evaluation matches using a best-of-three format. The results demonstrate clear differences in stability, learning efficiency, and overall performance, with PPO and A2C emerging as the most robust approaches in this complex, adversarial setting.

1 Introduction

1.1 Problem Statement

Most Reinforcement Learning benchmarks focus on simplified or stationary environments such as Atari games or classical control tasks. While useful, these environments fail to capture the complexity of real-time, adversarial decision-making. Competitive fighting games like Mortal Kombat II introduce challenges such as partial observability, delayed rewards, non-stationary opponents, and a large action space.

1.2 Objective

The objective of this work is to:

- Design a custom competitive reinforcement learning environment for Mortal Kombat II,
- Implement multiple value-based and policy-based DRL algorithms,
- Train agents for 1000 episodes,
- Evaluate their performance via 10 competitive evaluation matches, and
- Analyze their strengths and weaknesses in a real-time adversarial domain.

1.3 Motivation

Competitive games provide an excellent testbed for reinforcement learning due to their complexity and dynamic nature. Studying how different algorithms behave in such environments offers insights into their suitability for real-world decision-making systems where multiple intelligent agents interact.

2 Related work

Deep reinforcement learning has shown strong performance in video game environments, beginning with the introduction of Deep Q-Networks (DQN), which combine Q-learning with deep neural networks to learn policies directly from high-dimensional inputs. However, DQN is prone to overestimation bias, which can destabilize training.

Double DQN (DDQN) addresses this issue by decoupling action selection and action evaluation, leading to more stable learning. Dueling DQN further improves value-based methods by separating state-value and action-advantage estimation, which is beneficial in environments where multiple actions have similar effects.

Policy-based methods provide an alternative to value-based learning. Asynchronous Advantage Actor-Critic (A2C) learns a policy and value function jointly using advantage estimates to reduce variance. Proximal Policy Optimization (PPO) improves upon earlier policy gradient methods by constraining policy updates, resulting in stable and robust training.

Most prior work evaluates these algorithms on Atari benchmarks. In contrast, this work compares value-based and policy-based agents in a custom two-agent Mortal Kombat II environment, highlighting their behavior and performance in a competitive fighting game setting.

3 Setup, Agents, and Methods

3.1 Setup

A Mortal Kombat II (Genesis) ROM was imported into a `stable-retro` environment running on Ubuntu 22.04. Due to compatibility issues with Windows-native execution, the project was developed using Windows Subsystem for Linux (WSL), where Stable-Retro was manually installed and configured via the Ubuntu command prompt. The WSL Ubuntu interpreter was then used within PyCharm to enable stable training and evaluation of the agents.

The observation space consisted of the health points (HP) of both agents. To identify the memory locations corresponding to each agent's health, the BizHawk emulator was used, specifically its *RAM Watch* and *RAM Search* tools. After locating the relevant RAM addresses, these memory values were monitored programmatically during gameplay. The changing HP values were then used to determine episode termination conditions and to identify the winning and losing agents.

The official `stable-retro` library used in this work is available at: https://github.com/Farama-Foundation/stable-retro

The project files are available on my github: https://github.com/akolovska/Reinforcement-learning-in-Mortal-

3.2 Agents

The following reinforcement learning agents were implemented and evaluated:

- Deep Q-Network (DQN)
- Double Deep Q-Network (DDQN)
- Dueling Deep Q-Network (Dueling DQN)
- Proximal Policy Optimization (PPO)
- Advantage Actor-Critic (A2C)

The agents were evaluated in competitive two-agent matchups using the following combinations:

- DQN vs. DDQN
- PPO vs. DQN
- PPO vs. Dueling DQN
- PPO vs. DDQN
- A2C vs. DQN
- A2C vs. PPO

3.3 Methods

The action space consisted of a set of 10 manually designed discrete actions, representing meaningful in-game control combinations. After defining the observation and action spaces, agents were trained by competing directly against each other in the custom two-agent environment.

Each agent pair was trained for 1000 episodes, after which performance was evaluated using 10 separate evaluation matches. Training statistics, including episode rewards and win outcomes, were recorded to enable quantitative comparison between different agent architectures.

3.4 Use of AI-Generated Code

AI-assisted code generation was used selectively during the development of this project. Specifically, AI tools were employed to generate and refine parts of the code related to input preprocessing for the DQN-based algorithms, as well as the implementation of the Prioritized Experience Replay (PER) replay buffer. All generated code was reviewed, adapted, and integrated manually to ensure correctness, consistency with the project architecture, and compatibility with the custom Mortal Kombat II environment.

4 Results and Analysis

4.1 Results

Environment	Winning Agent
DQN vs DDQN	DDQN
PPO vs DQN	PPO
DQN vs DDQN	PPO
PPO vs Dueling DDQN	Both
A2C vs DQN	A2C
A2C vs PPO	PPO

Table 1: Observed Performance

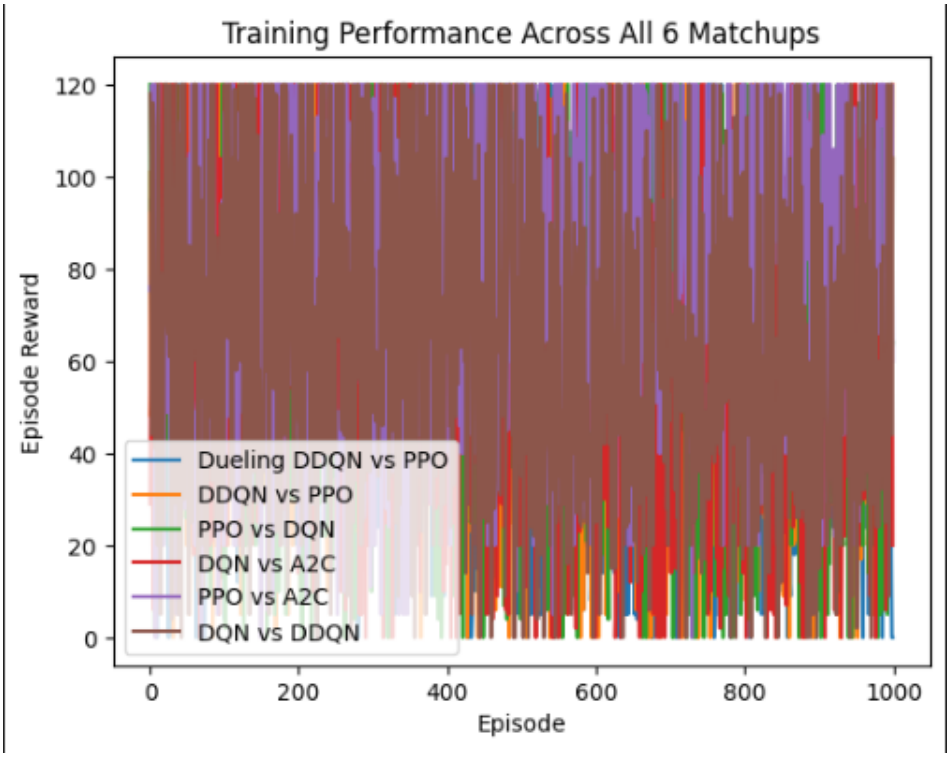

Figure 1: Training performance of PPO vs DQN

- Competitive evaluation provides more meaningful insights than single-agent reward metrics.
- PPO is highly suitable for complex real-time environments.
- Architectural improvements (e.g., dueling networks) significantly impact learning quality.

4.2 Analysis

- DQN suffers from instability and poor long-term planning.
- DDQN improves learning stability by reducing value overestimation.
- Dueling DDQN accelerates learning by better state-value estimation.
- A2C demonstrates stable but slower convergence.
- PPO achieves the best balance between stability, robustness, and performance.

Policy-based methods performed particularly well in the dynamic and adversarial environment.

5 Conclusion

This work demonstrated that Deep Reinforcement Learning agents can successfully learn competitive behaviors in a complex fighting game environment. By implementing and evaluating multiple algorithms under identical conditions, clear performance differences were observed. PPO and Dueling DDQN proved to be the most effective methods, while simpler value-based approaches showed limitations. The developed framework provides a strong foundation for future research in competitive and multi-agent reinforcement learning.

6 References

- Mnih et al. (2015) introduced the Deep Q-Network (DQN), demonstrating human-level performance on Atari games.
- Van Hasselt et al. (2016) proposed Double DQN (DDQN) to reduce Q-value overestimation.
- Wang et al. (2016) introduced Dueling Network Architectures, separating state-value and advantage estimation.
- Mnih et al. (2016) presented A3C/A2C, combining policy gradients with value estimation.
- Schulman et al. (2017) proposed PPO, improving training stability in policy-gradient methods.