Reinforcement Learning in FPS Games

Jagadeesh Reddy Vanga

December 28, 2024

Abstract

This Project implements the application of reinforcement learning techniques to enhance the performance of agents playing first-person shooter (FPS) games, specifically Doom, utilizing the Vizdoom environment[1]. The primary focus is on the implementation and comparison of two deep Q-network (DQN) variants: DQN and DuelDQN. Additionally, I tried to integrate Simultaneous Localization and Mapping (SLAM) algorithms to assist the agent in building accurate maps and self-localization within the game environment.

1 Introduction

First-person shooter (FPS) games have long been popular for their immersive and challenging game play, requiring players to make quick decisions and exhibit strategic thinking. With the advent of Artificial Intelligence (AI) and Reinforcement Learning (RL), there has been a growing interest in developing intelligent agents capable of playing FPS games at a high level. In this project I use RL Algorithms, specifically Deep Q-Networks (DQN) and DuelDQN, to enhance the performance of agents playing the FPS game Doom through the ViZDoom environment.

DQN and DuelDQN. DQN, introduced by Mnih et al. (2013), utilizes a neural network to approximate the action-value function, enabling the agent to learn from raw pixel input. DuelDQN, an extension of DQN proposed by Wang et al. (2016), decouples the estimation of the value and advantage functions, which can lead to more stable and efficient learning.

To further enhance the agent's understanding of the game environment, I tried to integrate Simultaneous Localization and Mapping (SLAM) algorithms. SLAM techniques, widely used in robotics and navigation, enable the agent to construct accurate maps of the game world while estimating its own position within these maps. By incorporating SLAM, the agent gains a spatial understanding of the environment, facilitating efficient navigation, exploration, and decision-making.

2 Implementation

For the implementation, I trained the agent on the Vizdoom environment, which was built using the resources available on the Github repository dedicated to Vizdoom. The Vizdoom environment provides a flexible and customizable platform for training agents in FPS games, offering various scenarios and maps to simulate different gameplay situations.

Figure 1: Example frame from Doom Game

To ensure a comprehensive evaluation of the agent's capabilities, I trained it on multiple scenarios within the Vizdoom environment. Each scenario presents unique challenges and requires different strategies to succeed. The scenarios I selected for training include:

Simple Basic: This scenario serves as an introductory level, featuring a relatively straightforward environment with basic enemies and limited complexity. The agent receives rewards for successfully shooting enemies, navigating through the environment, and surviving without taking damage.

Deadly Corridor: In this scenario, the agent faces a narrow corridor filled with hostile enemies. The confined space and high density of opponents make it crucial for the agent to exhibit precise aiming and quick reflexes to survive and progress. The agent is rewarded for eliminating enemies, avoiding enemy fire, and advancing through the corridor.

Defend the Center: The agent is placed in a central location that needs to be protected from enemy attacks. Waves of adversaries will attempt to breach the defense and capture the center. The agent receives rewards for eliminating enemies, successfully defending the center, and minimizing damage taken.

Defend the Line: Similar to "Defend the Center," this scenario requires the agent to defend a specific line or area from enemy incursions. The agent receives rewards for eliminating enemies, effectively holding the defensive line, and preventing enemy advances.

Health Gathering: This scenario emphasizes the importance of resource management and survival. The agent's primary objective is to collect health packs scattered throughout the environment while avoiding or defeating enemies. The agent is rewarded for collecting health packs, surviving enemy encounters, and minimizing health loss.

Death Match: In this scenario, the agent is placed in a fast-paced, competitive environment where it must engage in intense combat against multiple opponents. The objective is to accumulate the highest number of kills within a given time frame. The agent receives rewards for eliminating opponents, avoiding being killed, and achieving a higher kill count compared to other players.

The rewards assigned to the agent in each scenario are designed to incentivize desired behaviors and strategic decision-making. The agent's training involved iterative episodes, where it interacted with the chosen scenarios, observed the game states, selected actions, and received rewards based on its performance. Through the RL framework and the DQN and DuelDQN algorithms, the agent gradually improved its policy, optimizing its actions to maximize cumulative rewards and achieve high scores in each scenario.

3 Results for Each Scenario

Both the DQN and DuelDQN agents demonstrated satisfactory performance in simpler scenarios, where the environment was less complex and the challenges were relatively straightforward. In these scenarios, the agents were able to learn effective policies and achieve reasonable scores.

The increased complexity of the harder scenarios posed several challenges for the agents. The agents had to cope with a larger number of enemies, navigate intricate environments, and make rapid decisions under time pressure. Additionally, the agents needed to exhibit efficient resource management and prioritize objectives effectively. The agents' limitations in performing well in the harder scenarios highlight the inherent difficulty of the problem domain. The ability to generalize learning from simpler scenarios to more complex ones remains a significant challenge in the field of reinforcement learning for FPS games.

The Agent's performance can be seen in the Figures 2-8.

4 Improvements

Recognizing the challenges faced by the agents in learning and scoring higher in harder scenarios, a potential solution was to integrate Simultaneous Localization and Mapping (SLAM) techniques into the existing network architecture. By incorporating SLAM, it was hypothesized that the agents could benefit from improved environment understanding, accurate mapping, and self-localization capabilities.

SLAM algorithms are widely used in robotics and navigation to construct maps of unknown environments while estimating the robot's position within those maps. By applying SLAM techniques to

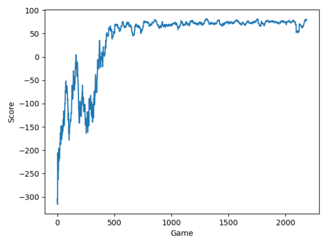

Figure 2: DuelDQN Basic.

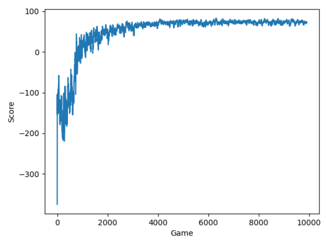

Figure 3: DQN Basic.

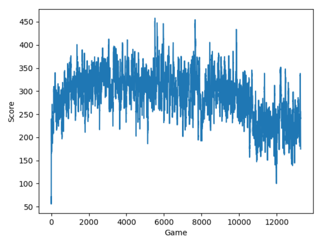

Figure 4: DuelDQN Deadly Corridor.

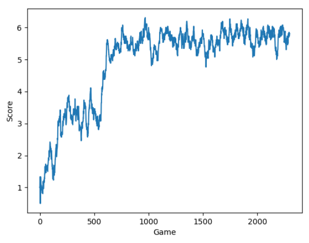

Figure 5: DuelDQN Defend the Center.

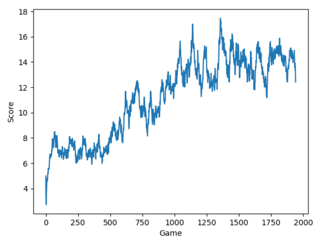

Figure 6: DuelDQN Defend the Line.

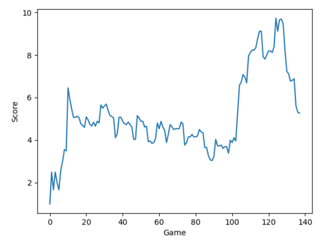

Figure 7: DuelDQN DeathMatch.

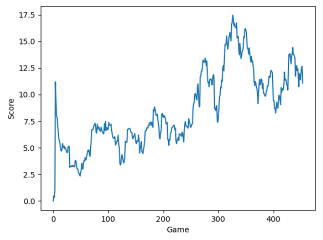

Figure 8: DQN DeathMatch

the FPS game environment, the agent would gain a more comprehensive understanding of the game world, enabling more informed decision-making and efficient navigation.

To implement it I have tried to extract 2d feature points like in Figure 9 and convert them into 3d point clouds which can be used to inject into the Network.

4.1 Issues Faced in SLAM Implementation

I couldn't convert the 2d point into 3d point clouds for use/visualization. I tried using Open3d to vizualize these 3d point clouds but WSL on Windows have few compatibility issues with this package.

4.2 Future Works

Although there exists publication like Active Neural SLAM already, it takes the map of env before hand and maps the agent to it, but in this experiment I tried to map the scenario on run time while the Agent is playing game. In Future, we can implement this and publish a research paper.

5 References

Github Repo

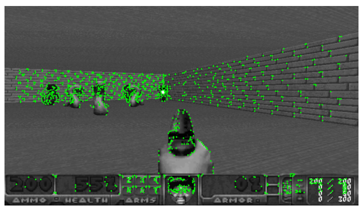

Figure 9: 2D spatial point from Frame