Reinforcement Learning Project

(Fire Fighter)

Team Members

- Ahmed Abdelaziz Hareedy
- Salah Amer Mohamed
- Anas Ahmed Desoky
- Menna Mohamed Abdelhady
- Horia Ahmed Abdelatief
- Mayar Mohamed Khedr

Under Supervision:

Eng. Maryam Mahmoud

Overview -Introduction

The **Fire Fighter RL Project** is a **Reinforcement Learning** simulation where an agent learns to extinguish fires in a grid-based environment. The environment simulates a forest where fires break out and spread over time, and the agent's objective is to prevent the fire from spreading too far, protecting the valuable resources (trees). Using **Deep Q-Learning (DQN) and Policy Gradient,** the agent learns through interaction with the environment, making decisions on how to best extinguish fires and avoid obstacles

Project Objectives:

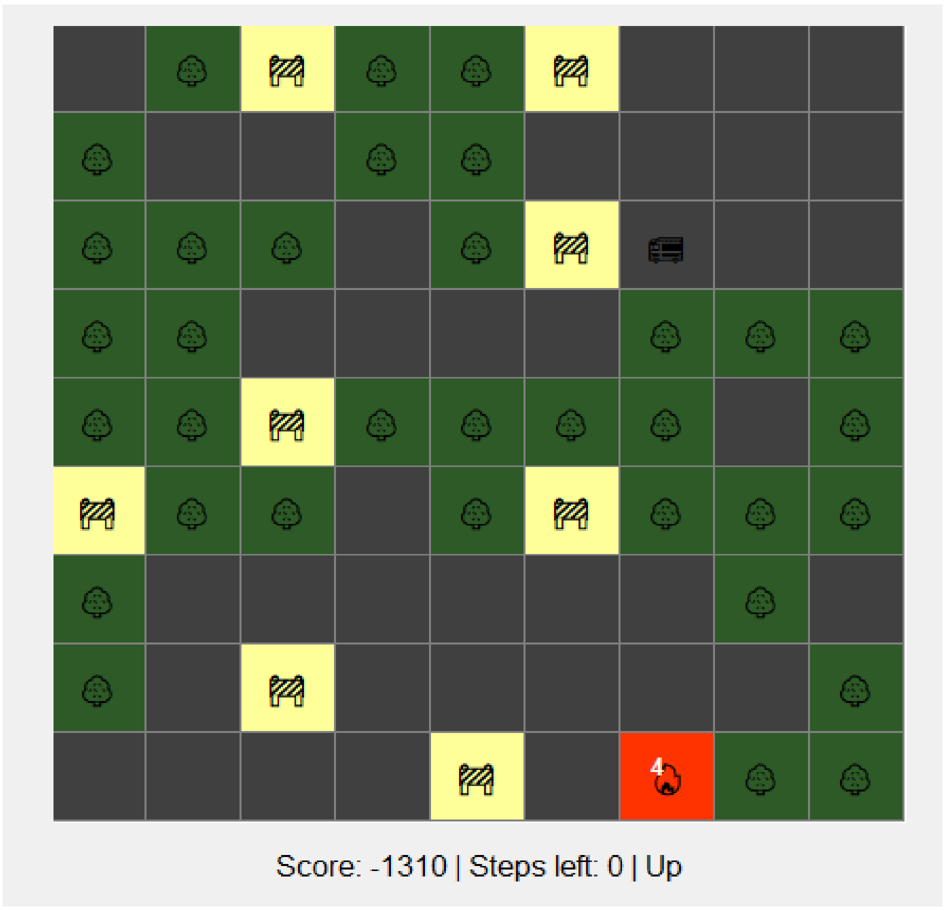
1. **Extinguish Fires**: The agent's primary responsibility is to put out the fire.
2. **Prevent Fire Spread**: While the agent works to extinguish the fire, it must also work to **prevent the fire from spreading** to nearby healthy trees.
3. **Avoid Obstacles**: The agent must navigate through a grid filled with obstacles that block its movement
4. **Maximize Efficiency**: The agent must learn to achieve its goals while minimizing the time it takes to extinguish the fire.

Environment Overview

The grid environment consists of cells that represent different types of terrain or states. Each cell can be one of the following:

- **Empty:** Represents empty land. The agent can freely move through this area.
- **Tree:** Represents a healthy tree. These trees are vulnerable to fire and should be protected.
- **Fire:** Represents a burning tree or fire. These cells must be extinguished by the agent.
- **Obstacle:** Represents barriers or obstacles in the environment that block the agent's path.

Fire Fighter Forest



Score: -1310 | Steps left: 0 | Up

Agent Actions and Rewards:

The agent performs actions to interact with the environment, and its decisions are influenced by the state of the grid. Common actions may include:

- **Move up, down, left, or right**: The agent moves within the grid to explore and reach the fire or avoid obstacles.
- **Extinguish Fire**: The agent puts out the fire in a specific location.

The agent is rewarded for extinguishing fires and efficiently preventing the spread of the fire. It is penalized for taking too long to extinguish fires or for causing fire spread to trees.

Algorithms Used to train agent:

- **Deep Q-Network (DQN)**

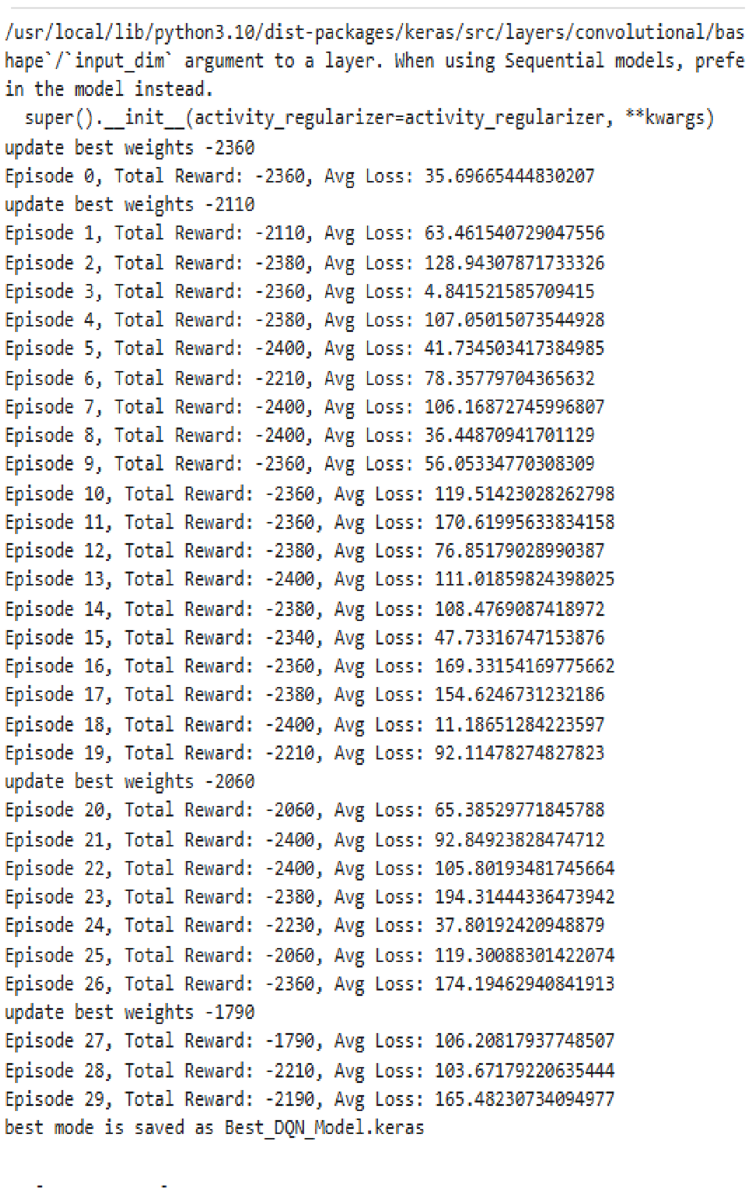
RL model uses a neural network to approximate the Q-value function, helping an agent learn optimal actions through reinforcement learning. The code is structured as follows:

DQN Training Steps:

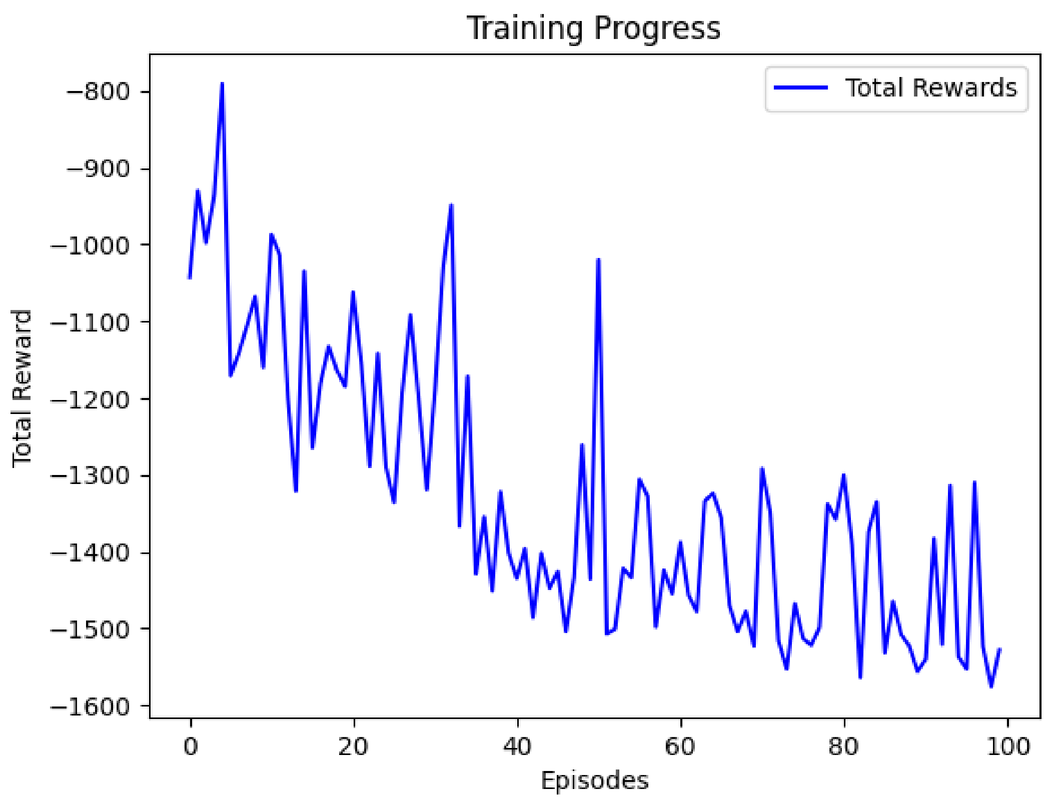
1. Initialize the Q-network and target network.
2. Store transitions (s,a,r,s') in a replay buffer.
3. Sample a batch from the replay buffer and compute the **Q-value targets**.

4. Compute the **loss** (Mean Squared Error between target and predicted Q-values).
5. Perform gradient descent to update the Q-network.
6. Periodically update the target network with weights from the main network.

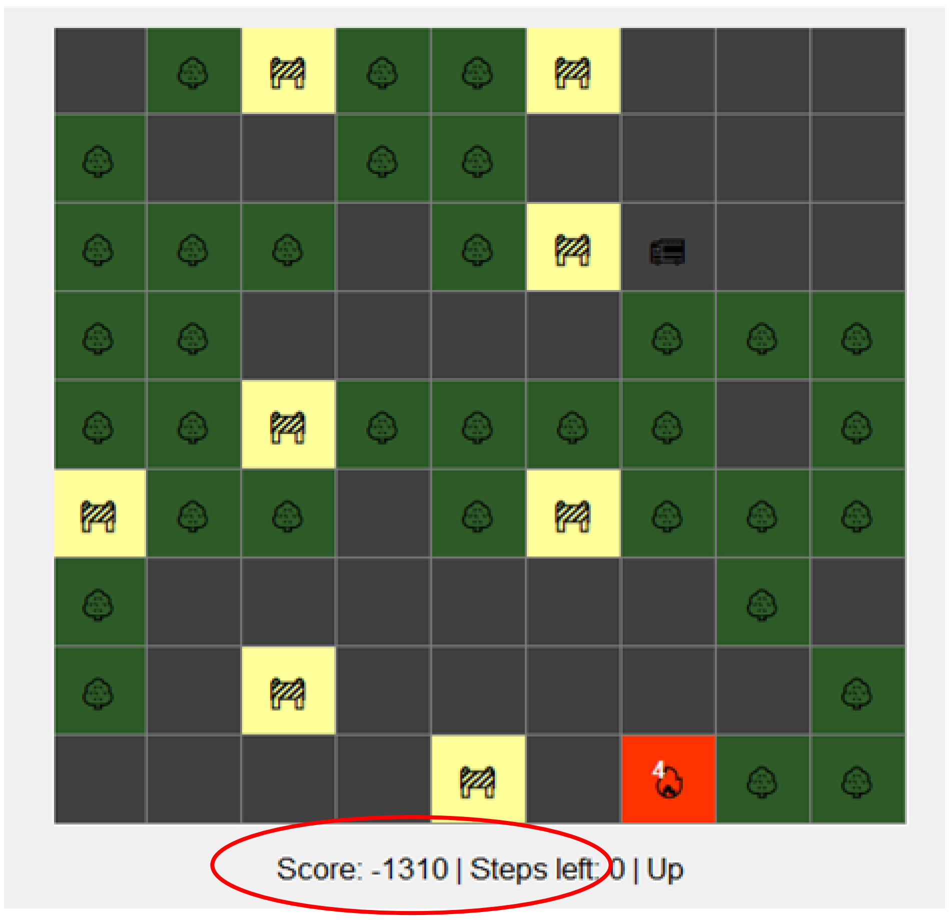
Train Result



```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/bas
hape`/`input_dim` argument to a layer. When using Sequential models, prefe
in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
update best weights -2360
Episode 0, Total Reward: -2360, Avg Loss: 35.69665444830207
update best weights -2110
Episode 1, Total Reward: -2110, Avg Loss: 63.461540729047556
Episode 2, Total Reward: -2380, Avg Loss: 128.94307871733326
Episode 3, Total Reward: -2360, Avg Loss: 4.841521585709415
Episode 4, Total Reward: -2380, Avg Loss: 107.05015073544928
Episode 5, Total Reward: -2400, Avg Loss: 41.734503417384985
Episode 6, Total Reward: -2210, Avg Loss: 78.35779704365632
Episode 7, Total Reward: -2400, Avg Loss: 106.16872745996807
Episode 8, Total Reward: -2400, Avg Loss: 36.44870941701129
Episode 9, Total Reward: -2360, Avg Loss: 56.05334770308309
Episode 10, Total Reward: -2360, Avg Loss: 119.51423028262798
Episode 11, Total Reward: -2360, Avg Loss: 170.61995633834158
Episode 12, Total Reward: -2380, Avg Loss: 76.85179028990387
Episode 13, Total Reward: -2400, Avg Loss: 111.01859824398025
Episode 14, Total Reward: -2380, Avg Loss: 108.4769087418972
Episode 15, Total Reward: -2340, Avg Loss: 47.73316747153876
Episode 16, Total Reward: -2360, Avg Loss: 169.33154169775662
Episode 17, Total Reward: -2380, Avg Loss: 154.6246731232186
Episode 18, Total Reward: -2400, Avg Loss: 11.18651284223597
Episode 19, Total Reward: -2210, Avg Loss: 92.11478274827823
update best weights -2060
Episode 20, Total Reward: -2060, Avg Loss: 65.38529771845788
Episode 21, Total Reward: -2400, Avg Loss: 92.84923828474712
Episode 22, Total Reward: -2400, Avg Loss: 105.80193481745664
Episode 23, Total Reward: -2380, Avg Loss: 194.31444336473942
Episode 24, Total Reward: -2230, Avg Loss: 37.80192420948879
Episode 25, Total Reward: -2060, Avg Loss: 119.30088301422074
Episode 26, Total Reward: -2360, Avg Loss: 174.19462940841913
update best weights -1790
Episode 27, Total Reward: -1790, Avg Loss: 106.20817937748507
Episode 28, Total Reward: -2210, Avg Loss: 103.67179220635444
Episode 29, Total Reward: -2190, Avg Loss: 165.48230734094977
best mode is saved as Best_DQN_Model.keras
```



DQN with Environment



Score: -1310 | Steps left: 0 | Up

- **Policy Gradient**

Policy Gradient methods directly optimize the policy (a probability distribution over actions) rather than estimating a Q-value function. The goal is to maximize the expected cumulative reward.

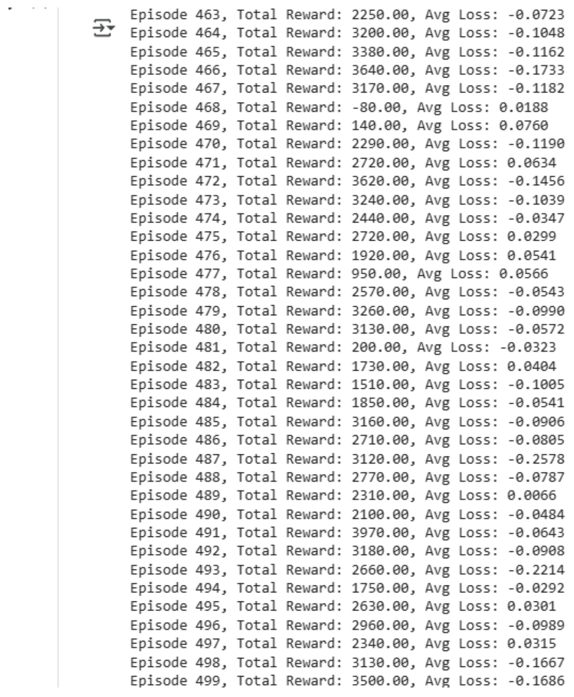
Key Features:

- Uses a stochastic policy, meaning actions are sampled from a learned probability distribution.
- Updates the policy using the gradient of expected reward with respect to policy parameters.

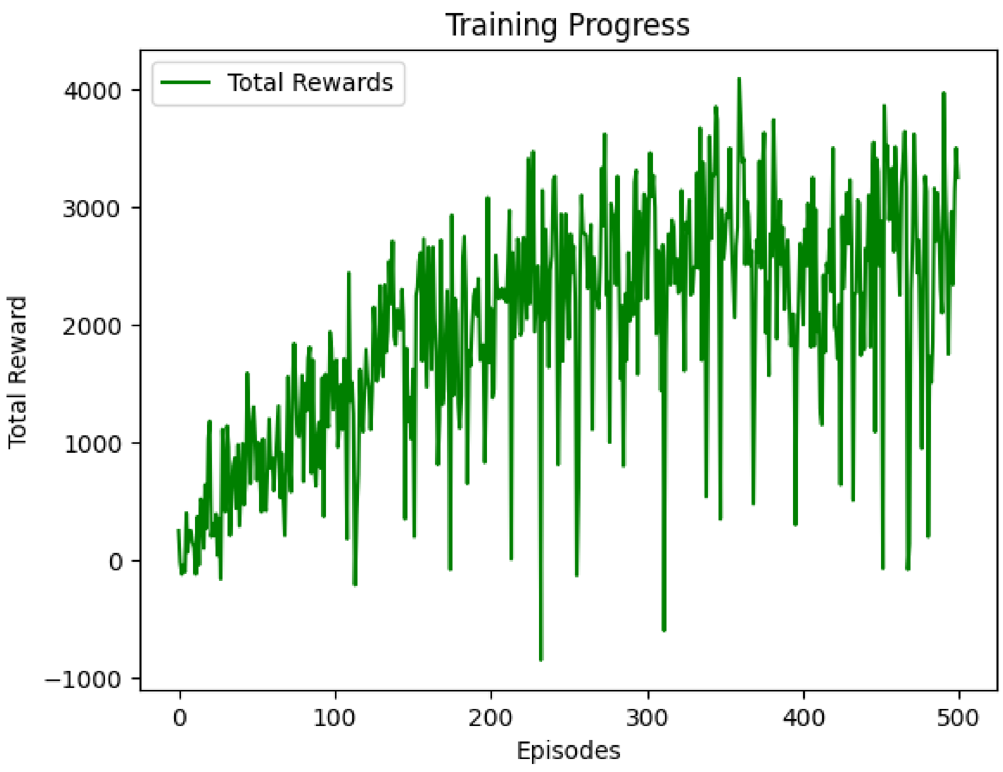
Policy Gradient Training Steps:

1. Initialize a neural network to represent the policy.
2. For each epoch:
    - Collect states, actions, and rewards.
    - Compute the cumulative discounted reward.
    - Compute the policy gradient.
    - Update the policy network using gradient ascent.

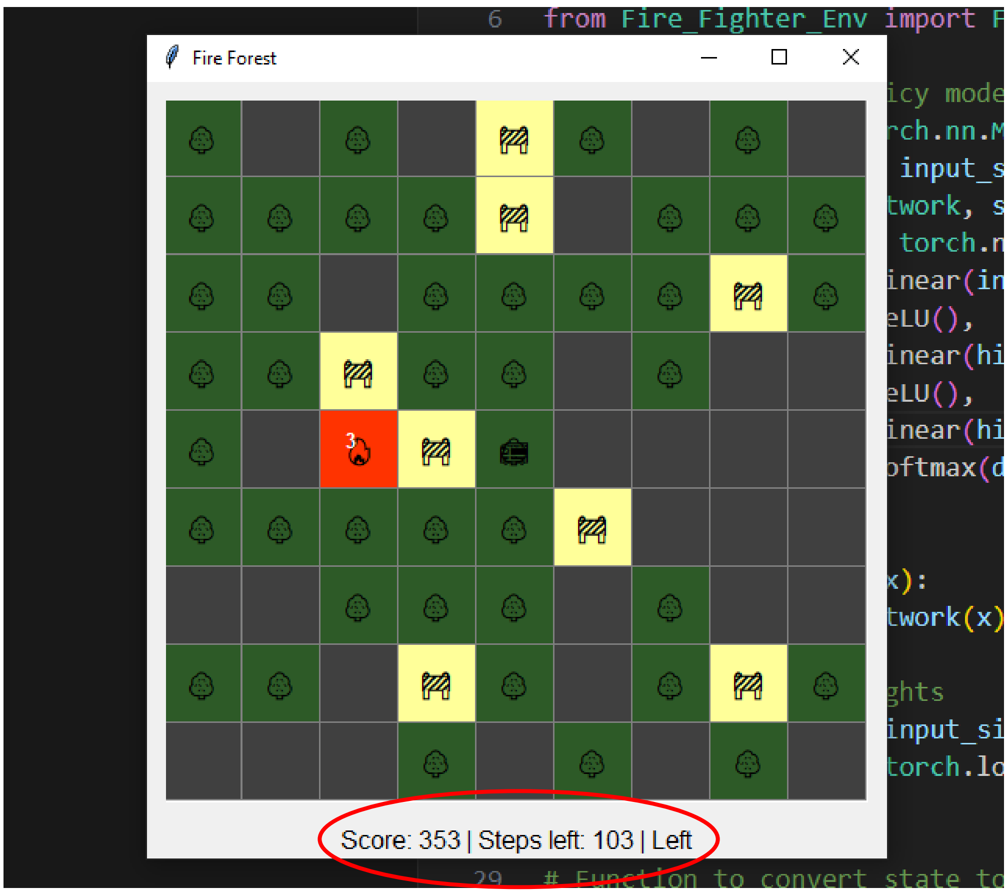
Train Result



```
Episode 463, Total Reward: 2250.00, Avg Loss: -0.0723
Episode 464, Total Reward: 3200.00, Avg Loss: -0.1048
Episode 465, Total Reward: 3380.00, Avg Loss: -0.1162
Episode 466, Total Reward: 3640.00, Avg Loss: -0.1733
Episode 467, Total Reward: 3170.00, Avg Loss: -0.1182
Episode 468, Total Reward: -80.00, Avg Loss: 0.0188
Episode 469, Total Reward: 140.00, Avg Loss: 0.0760
Episode 470, Total Reward: 2290.00, Avg Loss: -0.1190
Episode 471, Total Reward: 2720.00, Avg Loss: 0.0634
Episode 472, Total Reward: 3620.00, Avg Loss: -0.1456
Episode 473, Total Reward: 3240.00, Avg Loss: -0.1039
Episode 474, Total Reward: 2440.00, Avg Loss: -0.0347
Episode 475, Total Reward: 2720.00, Avg Loss: 0.0299
Episode 476, Total Reward: 1920.00, Avg Loss: 0.0541
Episode 477, Total Reward: 950.00, Avg Loss: 0.0566
Episode 478, Total Reward: 2570.00, Avg Loss: -0.0543
Episode 479, Total Reward: 3260.00, Avg Loss: -0.0990
Episode 480, Total Reward: 3130.00, Avg Loss: -0.0572
Episode 481, Total Reward: 200.00, Avg Loss: -0.0323
Episode 482, Total Reward: 1730.00, Avg Loss: 0.0404
Episode 483, Total Reward: 1510.00, Avg Loss: -0.1005
Episode 484, Total Reward: 1850.00, Avg Loss: -0.0541
Episode 485, Total Reward: 3160.00, Avg Loss: -0.0906
Episode 486, Total Reward: 2710.00, Avg Loss: -0.0805
Episode 487, Total Reward: 3120.00, Avg Loss: -0.2578
Episode 488, Total Reward: 2770.00, Avg Loss: -0.0787
Episode 489, Total Reward: 2310.00, Avg Loss: 0.0066
Episode 490, Total Reward: 2100.00, Avg Loss: -0.0484
Episode 491, Total Reward: 3970.00, Avg Loss: -0.0643
Episode 492, Total Reward: 3180.00, Avg Loss: -0.0908
Episode 493, Total Reward: 2660.00, Avg Loss: -0.2214
Episode 494, Total Reward: 1750.00, Avg Loss: -0.0292
Episode 495, Total Reward: 2630.00, Avg Loss: 0.0301
Episode 496, Total Reward: 2960.00, Avg Loss: -0.0989
Episode 497, Total Reward: 2340.00, Avg Loss: 0.0315
Episode 498, Total Reward: 3130.00, Avg Loss: -0.1667
Episode 499, Total Reward: 3500.00, Avg Loss: -0.1686
```



PG with Environment



```
from Fire_Fighter_Env import F
icy mode
rch.nn.M
 input_s
twork, s
 torch.n
inear(in
eLU(),
inear(hi
eLU(),
inear(hi
oftmax(d

x):
twork(x)

ghts
input_si
torch.lo

# Function to convert state to
```

```
Score: 353 | Steps left: 103 | Left
```