Deep Reinforcement Learning for Non-Player Character Navigation in Open-World Games

Zerui Lyu (Harry)

University of Minnesota Morris

November 16, 2024

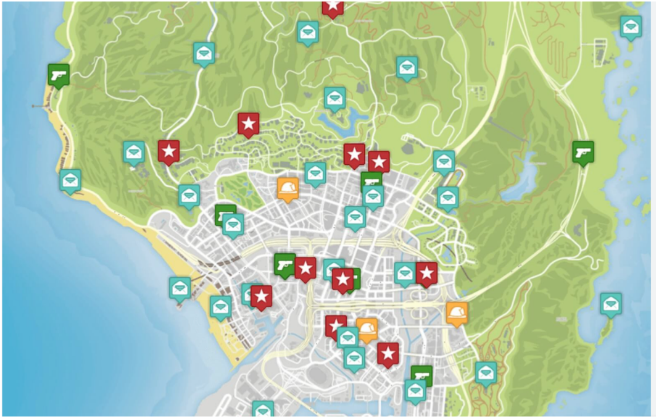
What is a Non-Player Character (NPC)

[2]

Definition

An NPC (Non-Player Character) is any character within a game that is not directly controlled by the player.

An Example of a Game Map



Overview of the Grand Theft Auto Map[4]

Outline

- **Introduction to NPC Navigation**
  - Definition and Role of NPCs
- **Background: Neural Network**
- **Deep Reinforcement Learning Framework for NPCs**
  - Input Data
  - Training with A3C Algorithm
    - Reward Function Calculation
    - Policy and Value Updates
- **Experiment**
- **Conclusion**

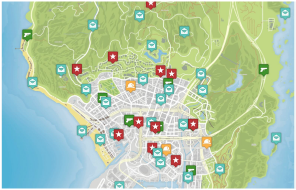
Data Collection for Agent Navigation

- **Data Types**: The agent collects:
  - **Linear Data**
  - **2D Visual Data**
- **Linear Data Details**:
  - **Distance to Obstacle**
  - **Distance to Enemy**
  - **Distance to Goal**
  - **Nearby Resource**

Visual Data:

- 2D Visual Sensor captures immediate visual field.

Input Data for NPC Navigation Example



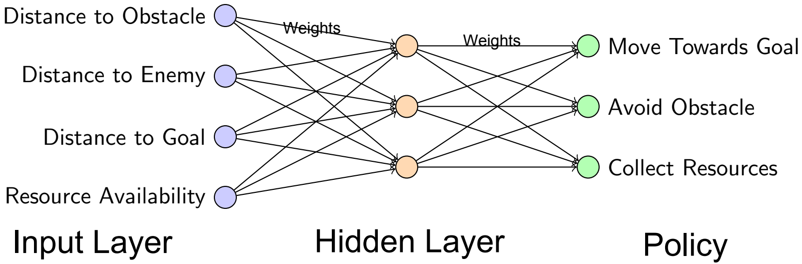
Mini-map illustration of the NPC in a forest environment

- **Scenario**: NPC navigating through GTA map with obstacles, enemies, and a target destination.
- **Input Data Elements**:
  - **Distance to Obstacle**: 5 meters
    Obstacle (tree) close by
  - **Distance to Enemy**: 15 meters
    Moderate threat nearby
  - **Distance to Goal**: 30 meters
    Destination (safe zone) farther away
  - **Nearby Resource**: 1
    Resource available (e.g., food or weapon)
- **Input Data**: [5, 15, 30, 1]

Output Decision Example

- **Output Layer Action Probabilities**:
  - **Avoid Obstacle**: 0.5
  - **Move Toward Goal**: 0.35
  - **Collect Resource**: 0.15
- **Decision**: The NPC decides to avoid the nearby obstacle first.

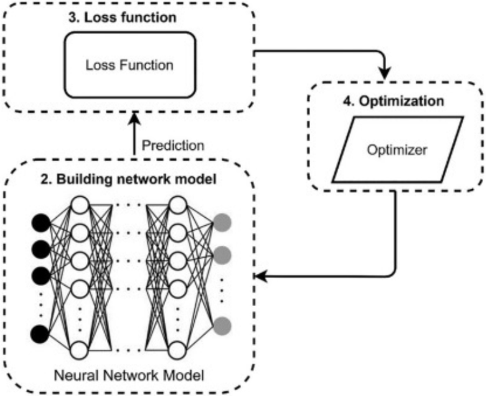
Neural Network for NPC Navigation Policy



Neural Network Processing

- **Input Layer**: Receives NPC data for navigation
- **Hidden Layer**: Processes input data to recognize patterns
- **Output Layer**: Outputs policy(A set of values over possible actions)(e.g., move forward, turn left, turn right)

Neural Network Training Process



[5]

- **Step 1: Prediction**
  - The neural network generates predictions using current weights
- **Step 2: Loss Calculation**
  - The loss function calculates the difference between predictions and actual targets.
- **Step 3: Weight Update**
  - The optimizer adjusts the weights using gradients to reduce the loss.

Limitations of Traditional Neural Network

- **Not Learning From Feedback**

Setup: NPC Navigation Scenario

- **Input data**:
  - Distances to Obstacle
  - Distance to the target location
- **Goal**: Learn optimal navigation

Initial State

- **Time Step 0 in State S_t:**
  - Distance to obstacle: 6 meters
  - Distance to goal: 20 meters
- **Input State**: [6, 20]

Some Notations:

$$S_t \; A_t \; R_{t+1} \; S_{t+1}$$

Subscript t represents the time step in a sequential process

Introduction to A3C Algorithm

- **Asynchronous Advantage Actor-Critic (A3C)** helps an NPC learn optimal actions in complex environments.
- **Example Scenario**
  - The **Actor** suggests actions (e.g., “move forward” or “avoid obstacle”).
  - The **Critic** evaluates the effectiveness of the actions.

Actor Network
(Proposes Actions)

Critic Network
(Evaluates Actions)

Actor's Initial Action and Critic's Value Estimation

- **Actor's Action Choices**(Policy):
  - Move toward goal — Value: 0.6
  - Avoid obstacle — Value: 0.8
- **Critic's Estimated Value for Initial State** $V(s)$: 1.0

Actor Network
(Proposes Actions)

Critic Network
(Evaluates Actions)

Reward Function Overview

- A reward function does:
  - Encourages efficient navigation
  - Penalizes unnecessary movements
  - Rewards goal completion

Reward Function Formula

- The reward at each time step t is defined as:

$$R_t = \max\left(\min_{\forall i \in [0, t-1]} E(t, i), 0\right) - \alpha + 100 \times \text{touch(agent, goal)}$$

- $\mathbf{E(t, i)} = \mathrm{dist}_i(\text{agent, goal}) - \mathrm{dist}_t(\text{agent, goal})$:
  Change in distance from goal (from time i to time t)
- Explanation of dist:
  - dist_i: Distance from the agent to the goal at a previous time step i.
  - dist_t: Distance from the agent to the goal at the current time step t.
  - Positive $E(t, i)$: The agent got closer to the goal.
  - Negative $E(t, i)$: The agent moved further away.
- α: Penalty for previous action, $\alpha = 0.5$.
- touch(agent, goal): Returns 1 if the NPC reaches the goal, 0 otherwise.

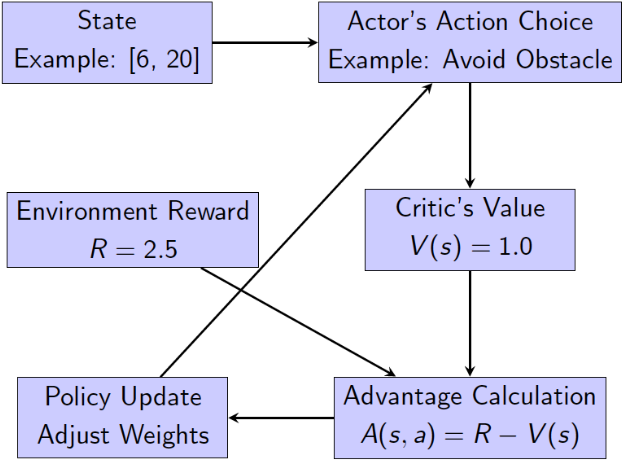
Calculating the Advantage Function

- The Critic calculates the advantage $A(s, a)$:

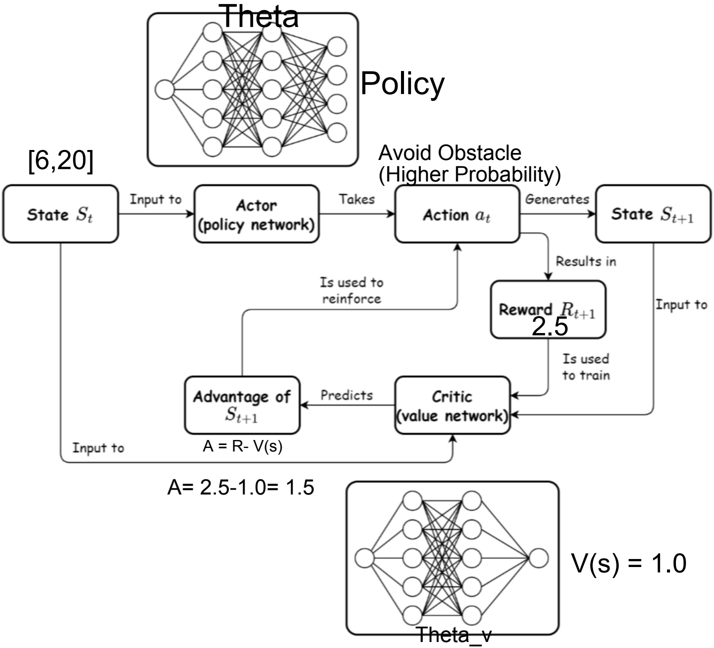
$$A(s, a) = R - V(s)$$

  where:
  - $R = 2.5$: Actual reward obtained
  - $V(s) = 1.0$: Critic's estimated value for the state
  - $A(s, a) = 2.5 - 1.0 = 1.5$: Positive advantage, reinforcing this action

A3C Process Overview for NPC Navigation



Deep Reinforcement Learning Overview



[1]

Policy Update and Weight Adjustment

- **Policy Update Formula**:

$$\Delta\theta = \beta \nabla_\theta \log\left(\pi_\theta(a|s)\right) \cdot A(s, a)$$

- **Formula Breakdown**:
  - $\pi_\theta(a|s) =$ action policy (last layer of actor NN)
  - $\Delta\theta$: Change in Actor's weights to improve the policy.
  - $\beta = 0.01$: Learning rate.
  - $A(s, a) = 1.5$: How much better the action a performed compared to expectations.
  - $\log \pi_\theta(a|s)$: The log function works on $\pi_\theta(a|s)$.
  - $\nabla_\theta \log \pi_\theta(a|s)$: Calculates how much to adjust the policy's parameters (weights).

Iterative Learning and Final Behavior

- Through repeated feedback, the Actor and Critic networks refine the NPC's navigation policy.
- **Final Learned Behavior**:
  - NPC reaches the goal efficiently
  - Avoids obstacles effectively

Experiment Setup

- **Objective**: Demonstrate DRL system's capability in complex navigation tasks.
- **Map Dimensions**: 400m x 400m with a 35m height.
- **Configurations Tested**:
  - **Base Configuration**: Feedforward network with full state data.
  - **VAR1**: Feedforward network without full state data.
  - **VAR2 and VAR3**: LSTM networks (single and double layers).
- **Training Episodes**: 6,000 episodes per configuration.
  - Success Rate is calculated as:

$$\text{Success Rate} = \frac{\text{Number of Successful Episodes}}{\text{Total Number of Episodes}}$$

Experimental Configurations

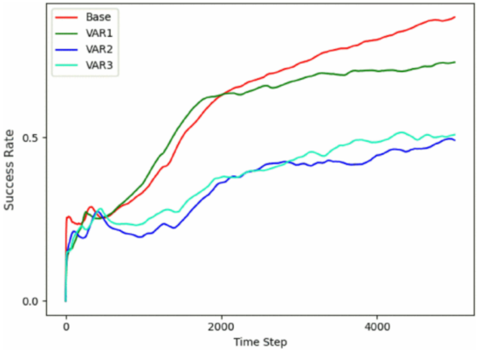
Table: Experimental Configurations

Configuration	Model Architecture	State Data Included
Base	Feedforward Network	Full state, including animation
VAR1	Feedforward Network	Without animation data
VAR2	Single-layer LSTM	Full state, including animation
VAR3	Two-layer LSTM	Full state, including animation

figureExperiment Environment[3]

Experiment Results: Training Phase

- **Overall Performance**:
  - All configurations improved in success rate during training.
  - **Base configuration (Feedforward)** achieved the highest success rate.
- **FeedForward Configurations (Base vs. VAR1)**
- **LSTM Configurations (VAR3 and VAR4)**
- **Conclusion**



Training Phase Diagram Configurations[3]

Conclusion

- **Feedforward Models (Base, VAR1)**:
  - Performed best in training phase.
- **LSTM Models (VAR2, VAR3)**:
  - Lower performance

Thank You for Listening!

Any Questions?

References I

[1] *Actor-Critic Methods*. Accessed: 2024-11-16. 2024. URL: https://wikidocs.net/175903.

[2] EGM Staff. *GTA VI may have much smarter NPCs than we've ever seen*. Accessed: 2024-11-16. 2024. URL: https://egmnow.com/gta-vi-may-have-much-smarter-npcs-than-weve-ever-seen/.

[3] Gilzamir Gomes et al. "Two Level Control of Non-Player Characters for Navigation in 3D Games Scenes: A Deep Reinforcement Learning Approach". In: *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. 2021, pp. 182–190. DOI: 10.1109/SBGames54170.2021.00030.

[4] IGN Staff. *GTA 5 Interactive Map: Los Santos and Blaine County*. Accessed: 2024-11-16. 2024. URL: https://www.ign.com/maps/gta-5/los-santos-blaine-county.

References II

[5] *Neural Network Training - An Overview*. Accessed: 2024-11-16. 2024. URL: https://www.sciencedirect.com/topics/engineering/neural-network-training.