Raport Proiect Inteligență Artificială – Unity ML-Agents

# 1. Definirea temei și problemei

Tema proiectului:  
Antrenarea unui agent autonom într-un mediu 3D pentru a naviga către o destinație evitând obstacole, utilizând algoritmi de învățare prin întărire (Reinforcement Learning), cu ajutorul pachetului ML-Agents dezvoltat de Unity.  
  
Descrierea problemei:  
Scopul este de a crea un agent inteligent, reprezentat de un cub, capabil să ajungă la o țintă într-un mediu 3D în care sunt amplasate obstacole. Agentul trebuie să învețe strategii eficiente pentru a evita coliziunile și a minimiza timpul necesar pentru a ajunge la destinație. Este o problemă clasică de tip pathfinding cu RL, cu o funcție de recompensă definită astfel încât să penalizeze coliziunile și să recompenseze atingerea țintei.

# 2. Analiza articole relevante, referințe

În realizarea acestui proiect, am studiat următoarele surse relevante:  
  
[1] Unity ML-Agents Toolkit Documentation – oferă detalii esențiale despre arhitectura toolkit-ului, tipurile de agenți, observabile, recompense și politicile de antrenament.  
https://github.com/Unity-Technologies/ml-agents  
  
[2] Schulman et al., "Proximal Policy Optimization Algorithms" (2017) – lucrarea care fundamentează algoritmul PPO, utilizat în ML-Agents pentru training-ul politicilor.  
https://arxiv.org/abs/1707.06347  
  
[3] Sutton & Barto, "Reinforcement Learning: An Introduction" (2018) – o sursă fundamentală pentru înțelegerea conceptelor RL.  
  
[4] Bloguri și tutoriale precum:  
- https://www.immersivelimit.com/  
- YouTube: Code Monkey – ML Agents Intro & Navigation AI

# 3. Implementare metodă

1. Tehnologii folosite:  
   - Unity 6000.0.43f1   
   - ML-Agents Toolkit v1.1.0  
   - Python 3.10.12  
   - PyTorch 2.2.2  
     
   Structura proiectului:  
   - Mediu: Scena Unity conține:  
    - Agentul (cubul inteligent)  
    - obstacole distribuite pentru a forma diferite labirinturi  
    - 1 punct de start, 1 țintă, multiple puncte recompensă intermediare  
   - Observabile: Poziția agentului, distanța față de țintă, vectori direcționali, detectarea obstacolelor cu raycasts  
   - Acțiuni: Mișcare în plan (X/Z) cu viteză constantă  
   - Funcție recompensă:  
    +1 la atingerea țintei  
    -1 la coliziune cu obstacol  
    -0.001 penalizare per pas (pentru încurajarea traseului scurt)  
     
   Antrenare:

1) Algoritm PPO (Proximal Policy Optimization)

1. Hyperparametri:  
    - batch\_size: 256  
    - buffer\_size: 24000  
    - learning\_rate: 1.0e-4
2. Network Settings:
   1. normalize: true
   2. hidden\_units: 128
   3. num\_layers: 2
   4. vis\_encode\_type: simple
3. Reward Signals:
   1. Extrinsic:
      1. gamma: 0.99
      2. strength: 1.0
   2. keep\_checkpoints: 5
   3. max\_steps: 2000000
   4. time\_horizon: 1000
   5. summary\_freq: 24000

# 4. Experimente și comparații

Au fost testate două niveluri cu dificultate diferită:  
  
- Nivelul 1: Mediu mai complex, cu obstacole multiple și poziționare care necesită mișcări mai precise.  
- Nivelul 2: Mediu simplu, cu obstacole puține și distanță mică între punctul de start și țintă.  
Scopul a fost evaluarea capacității agentului de a generaliza comportamentul în condiții diferite de dificultate.

Metoda de antrenare:

Pentru fiecare nivel au fost create 4 versiuni cu grade de dificultate diferite: no obstacles, easy, normal, hard. Pentru fiecare versiune a avut loc o sesiune de antrenament cu 2 un numar maxim de 2 milioane de pasi. Dupa fiecare sesiune, se prelua progresul pentru urmatorul nivel de dificultate. Asadar, avem mai jos grafice cu performanta agentului in ambele nivele.

Nivelul 1

A graph of a graph

AI-generated content may be incorrect.

In graficul de mai sus avem recompensele agentului pe parcursul sesiunilor de antrenament. Avand in vedere faputl ca agentul trebuia sa se strecoare printre doi pereti ca sa paraseasca zona de inceput, se poate observa ca recompensele pleaca de la valori mici, urmand ca acestea sa ramana stabile la valori mari pe final.

A graph with lines and numbers

AI-generated content may be incorrect.

In graficul de mai sus avem durata episoadelor de antrenament. La fel ca si la graficul de mai sus, se observa dificultatea agentului de a parcurge nivelul la inceputul sesiunii, ca mai apoi sa termine un episode intr-un timp foarte scurt.

A graph of a graph

AI-generated content may be incorrect.

In graficul de mai sus avem pierderile din primul nivel. Functia de pierdere difera de la sesiune la sesiune (ex.: prima sesiune incepe cu o crestere a pierderilor, urmata de o scadere a lor si spre finalul sesiunii acestea sa se mentina la valori sub 0.2).

A graph with lines and numbers

AI-generated content may be incorrect.Nivelul 2

In graficul de mai sus avem evolutia recompenselor in nivelul 2 pe parcursul sesiunilor de antrenament. Dupa cum se poate observa, in train\_02\_no\_obstacles recompensele au avut o crestere exponentiala, are au pornit de la aproximativ -200. Spre sfarsitul primei sesiuni de antrenament se poate observa ca valorile recompenselor se mentin positive. In urmatoarele sesiuni, recompensele isi pastreaza aproape aceeasi valoare, fara a exista scaderi sau cresteri bruste.

A graph of a graph

AI-generated content may be incorrect.

In graficul de mai sus avem evolutia duratei episoadelor de antrenament. Se poate observa ca cu cat creste numarul pasilor intr-o sesiune, agentul termina episoadele tot mai repede.

A graph of a graph

AI-generated content may be incorrect.

In graficul de mai sus avem functia de pierdere, care reprezinta diferenta dintre un comportament ideal si cel curent al agentului. Doar in prima sesiune se poate observa o scadere a pierderilor, it timp ce in urmatoarele sesiuni pierderile au valori mici, si aproape constante.

Ca o comparatie intre cele doua niveluri, agentul a avut rezultate mai bune in nivelul 2, rezultatele mentinandu-se la valori optime dupa prima sesiune de antrenament. In primul nivel, datorita faptului ca spatiile prin care agentul se putea deplasa erau stramte, agentul a intampinat dificultati, si nu a reusit sa mentina rezultatele (recompense, lungimea episoadelor) la valori optime.

# Concluzii

Agentul antrenat cu PPO a demonstrat capacitatea de a naviga eficient într-un mediu 3D, învățând să evite obstacolele și să atingă ținta într-un număr mic de pași. Antrenarea a fost stabilă, iar framework-ul ML-Agents s-a dovedit eficient și flexibil.