

UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
INTRODUCERE IN REINFORCEMENT LEARNING

INTRODUCERE IN REINFORCEMENT LEARNING

UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
INTRODUCERE IN REINFORCEMENT LEARNING

MAZE EXPLORER

Componenta echipei:

Cristina-Diana Savin 464

Iulia-Maria Cucu 461

Teodora-Alexandra Nitu 332

Bianca-Andreea Husanu 344

Robert Baidoc 344

Profesor coordonator:

Stefan Iordache

Cuprins

1. Introducere & Context	2
1.1 Context Reinforcement Learning	2
1.2 Scopul proiectului	2
2. Environment Design	3
2.1 Descriere generală	3
2.2 Spaţiul stărilor.....	4
2.3 Setare Layere și reprezentare internă	4
2.4 Spaţiul acţiunilor	4
3. Algoritmi de Reinforcement Learning.....	5
3.1 Value Iteration.....	5
3.2 SARSA	5
3.3 Q-Learning	5
3.4 Deep Q-Network (DQN).....	6
3.5 Motivaţia alegerii algoritmilor.....	6
4. Experimente și Rezultate.....	6
4.1 Setup experimental.....	6
4.2 Metrici de evaluare	6
4.3 Reglarea hiperparametrilor și experimente multiple	7
4.4 Analiza rezultatelor	7
5. Grafice	8
6. Extensii viitoare.....	10
6. Bibliografie	11

1. Introducere și Context

1.1 Context Reinforcement Learning

Reinforcement Learning (RL) este o ramură a inteligenței artificiale care se concentrează pe învățarea prin interacțiune cu mediul. Spre deosebire de învățarea supravegheată, unde agentul primește exemple corecte, în RL agentul descoperă treptat ce acțiuni conduc la rezultate bune sau rele, prin **trial-and-error**. Agentul RL învață o **politică optimă**: o funcție care asociază fiecărei stări acțiunea cea mai bună pentru maximizarea recompensei cumulative. Această paradigmă se inspiră din învățarea biologică, unde experiențele pozitive sunt întărite, iar cele negative sunt evitate.

Componențele RL:

- **Agent**: Entitatea care ia decizii. În proiectul nostru, agentul reprezintă un „explorator” în labirint.
- **Environment (Mediu)**: Spațiul în care agentul acționează. În Maze Explorer, mediu este un labirint bidimensional discret.
- **Stare (State)**: Descriere completă a situației curente. Include poziția agentului și, eventual, a obiectivului sau a obstacolelor.
- **Acțiune (Action)**: Decizia pe care agentul o poate lua (sus, jos, stânga, dreapta).
- **Recompensă (Reward)**: Feedback numeric care ghidează agentul către comportamente optime.

Aplicații ale RL: navigația autonomă a roboților, jocuri și simulatoare 2D/3D, optimizarea traficului, control industrial, trading algoritmic și multe altele. În context educațional, RL oferă un cadru excelent pentru a înțelege concepte precum explorare vs exploatare, recompense și penalizări, și politici optime vs sub-optime, permitând compararea directă a algoritmilor clasici și moderni.

1.2 Scopul proiectului

Proiectul „Maze Explorer” are ca obiectiv dezvoltarea unui mediu custom pentru navigație într-un labirint 2D, fără obstacole inițiale, în care agentul trebuie să găsească calea către un obiectiv fix. Scopul principal este de a implementa și compara diferiți algoritmi de RL, inclusiv metode tabulare clasice (Value Iteration, SARSA, Q-Learning) cu **Deep RL** (DQN), și de a analiza performanța acestora. Proiectul urmărește să demonstreze evoluția de la un setup simplu, didactic, la abordări mai complexe și scalabile, pregătind astfel terenul pentru implementări avansate precum multi-agent sau maze-uri generate procedural. Totodata, proiectul se axează pe analiza performanței agenților prin metriki standard.

UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
INTRODUCERE IN REINFORCEMENT LEARNING

Motivația:

Labirinturile oferă un exemplu simplu, intuitiv, dar suficient de complex pentru a ilustra principii fundamentale ale RL. Prin proiectul „Maze Explorer”, studenții pot înțelege cum diferiți algoritmi învață, cum parametrii afectează performanța și cum conceptele teoretice se traduc în rezultate vizuale și cuantificabile. Astfel, proiectul combină **educația, experimentarea practică și pregătirea pentru cercetări avansate în robotică și AI adaptivă**.

2. Environment Design

2.1 Descriere generală

Mediul Maze Explorer este construit sub forma unei grile bidimensionale $N \times M$. Agentul se deplasează de la o poziție inițială către o țintă specifică, explorând celulele libere. Labirintul este reprezentat în mod discret, fiecare celulă având un statut clar: liberă, goal sau perete (în extensii viitoare). Episodul se încheie atunci când agentul ajunge la țintă sau depășește un număr maxim de pași, ceea ce permite măsurarea performanței în funcție de eficiența navigației. Această structură permite control total asupra mediului și facilitează compararea algoritmilor RL implementați.

2.2 Spațiul stărilor

Starea mediului este definită prin poziția agentului în labirint, combinată cu poziția obiectivului. Această reprezentare simplă permite algoritmilor tabulari să calculeze valori pentru fiecare combinație de stare și acțiune. În varianta Deep RL, starea poate fi reprezentată ca o matrice bidimensională cu diferite layeruri pentru agent, goal, permitând rețelei neuronale să învețe o reprezentare mai abstractă a mediului.

- Reprezentarea stării: coordonate (x, y) ale agentului.

Deep RL: starea poate fi o matrice bidimensională cu layeruri separate:

- Layer agent: 1 pentru agent, 0 restul
- Layer goal: 1 pentru goal, 0 restul
- Layer obstacole (în extensii): 1 pentru obstacol, 0 restul

Această reprezentare permite rețelelor neuronale să învețe relațiile spațiale dintre elementele mediului. Această reprezentare poate fi vizualizată ca o „**imagină**” a **labirintului**, unde fiecare layer este un canal de informație. Rețelele neuronale convoluționale (CNN) pot procesa această matrice pentru a învăța relații spațiale între elemente.

2.3 Setare layere și reprezentare internă

Pentru a permite o viitoare extensie către Deep RL și vizualizare 3D, mediul folosește mai multe layere: unul pentru poziția agentului, unul pentru goal și, în versiuni ulterioare, unul pentru obstacole. Această reprezentare stratificată ajută agenții să învețe relații spațiale între obiective și obstacole, facilitând algoritmi mai sofisticati, cum ar fi DQN sau PPO, care pot procesa input multidimensional.

- **Layer agent:** În acest layer, celulele corespunzătoare poziției agentului sunt marcate cu valoarea 1, restul cu 0. Acest layer oferă rețelei neuronale informația despre locația exactă a agentului în labirint.
- **Layer goal:** Similar, poziția obiectivului este reprezentată în acest layer, permitând agentului să localizeze ținta și să învețe relațiile spațiale între poziția sa și goal.
- **Layer obstacole (extensii viitoare):** În versiunile avansate, obstacolele și peretele sunt reprezentați într-un layer separat. Acest lucru permite agentului să învețe cum să evite zonele inaccesibile și să planifice trasee optime.

2.4 Spațiul acțiunilor

Agentul poate efectua patru acțiuni: deplasare sus, jos, stânga sau dreapta. Mediul asigură verificarea acțiunilor, astfel încât agentul să nu poată trece prin perete sau să iasă din grilă. Acțiunile valide conduc la schimbarea stării, iar acțiunile invalide pot genera o penalizare, ceea ce încurajează agentul să exploreze eficient și să învețe să evite greșelile.

Acțiunile sunt evaluate astfel:

- **Acțiuni valide:** agentul se deplasează și starea se actualizează corespunzător.
- **Acțiuni invalide:** agentul rămâne pe loc și poate primi o penalizare numerică, încurajând astfel evitarea greșelilor și explorarea eficientă.

Această structură simplă, dar robustă, permite agentului să învețe:

- să planifice trasee optime către goal,
- să evite zonele periculoase sau ineficiente,
- să balanzeze între **explorare** (descoperirea labirintului) și **exploatare** (folosirea traseelor deja învățate).

3. Algoritmi de Reinforcement Learning

3.1 Value Iteration

Value Iteration este o metodă clasică de tip Dynamic Programming care calculează politicile optime atunci când mediul este complet cunoscut. Algoritmul explorează iterativ toate stările posibile și determină pentru fiecare stare acțiunea care maximizează recompensa totală. Este foarte util pentru medii discrete și bine definite, precum Maze Explorer, deoarece oferă o politică clară și stabilă pentru agent. Limitarea principală este că, odată ce dimensiunea labirintului crește, spațiul stărilor devine foarte mare, ceea ce face metoda impracticabilă pentru medii complexe.

3.2 SARSA

SARSA este un algoritm on-policy, ceea ce înseamnă că agentul învăță evaluând acțiunile pe care le efectuează efectiv în mediul său. În practică, SARSA ajustează politica agentului pe baza experiențelor reale, ceea ce duce la o navigație mai conservativă și mai sigură. În Maze Explorer, acest algoritm permite observarea modului în care explorarea și politica influențează performanța agentului și stabilitatea învățării, fiind util pentru înțelegerea diferenței între abordările on-policy și off-policy.

3.3 Q-Learning (tabular)

Q-Learning este un algoritm off-policy foarte popular în mediile discrete. Agentul învăță să estimeze cea mai bună acțiune pentru fiecare stare, independent de acțiunile pe care le efectuează efectiv. Aceasta îl face mai agresiv decât SARSA, maximizând rapid recompensa și optimizând traseul către goal. În Maze Explorer, Q-Learning oferă o comparație clară între metodele on-policy și off-policy, evidențierănd avantajele explorării agresive în medii bine definite.

3.4 Deep Q-Network (DQN)

DQN extinde Q-Learning folosind rețele neuronale pentru a aproxima valorile acțiunilor, ceea ce permite agentului să învețe în medii cu un spațiu mare al stărilor sau cu input vizual complex. În proiectul nostru, DQN poate procesa matricele stratificate care reprezintă poziția agentului, a goal-ului și a obstacolelor, învățând relațiile spațiale dintre aceste elemente. Tehnici precum memorarea experiențelor și folosirea unei rețele sănătoase ajută la stabilizarea procesului de învățare și la prevenirea instabilităților.

3.5 Motivația alegerii algoritmilor

Alegerea combinată de algoritmi tabulari și Deep RL permite o analiză completă a performanței agenților. Value Iteration oferă o perspectivă teoretică și stabilitate, SARSA și Q-Learning permit compararea on-policy vs off-policy, iar DQN arată scalabilitatea și aplicabilitatea în medii mai complexe. Această combinație ilustrează trecerea de la metode fundamentale la metode moderne, pregătind terenul pentru viitoare extensii.

4. Experimente și Rezultate

4.1 Setup experimental

Fiecare agent a fost evaluat pe un număr de episoade predeterminat, cu parametri inițiali consistenti pentru learning rate și discount factor. Strategia de explorare utilizată a fost ϵ -greedy, cu decay gradual pentru a permite agentului să exploreze inițial și să exploateze ulterior experiența acumulată. Mediul folosit a fost unul simplificat, fără obstacole inițiale, pentru a permite o comparare rapidă și clară între agenți. Această configurație inițială oferă un punct de plecare solid pentru analiza performanței fiecărui algoritm înainte de a trece la extensii mai complexe, cum ar fi labirinturi dinamice sau input visual.

4.2 Metrici de evaluare

Pentru evaluarea performanței agenților s-au folosit mai multe metrici esențiale:

1. **Reward mediu pe episod:** indică cât de eficient reușește agentul să acumuleze recompense în timpul navigării. O valoare mai mare arată că agentul învăță să se apropie rapid și eficient de goal.
2. **Număr mediu de pași pentru a ajunge la goal:** măsoară eficiența navigației. Cu cât agentul finalizează episoadele cu mai puțini pași, cu atât politica sa este mai optimă. Această metrică este utilă pentru a observa diferențele între strategiile conservatoare (SARSA) și cele mai agresive (Q-Learning).
3. **Rata de succes a episodului:** reprezintă procentul de episoade în care agentul a ajuns la goal înainte de atingerea numărului maxim de pași. Este o măsură importantă pentru a evalua stabilitatea și robustețea politicilor învățate.

UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
INTRODUCERE IN REINFORCEMENT LEARNING

Aceste metrii permit o **comparare clară între agenți**, evidențiind avantajele și limitările fiecărui algoritm. De exemplu, un agent cu reward mediu ridicat, dar cu număr mare de pași, poate fi eficient în acumularea recompenselor dar ineficient în termenii de timp sau resurse. În același mod, un agent cu rată de succes scăzută poate arăta vulnerabilitate la explorarea insuficientă sau la politici nepotrivite.

Această evaluare nu doar că reflectă performanța imediată, dar și oferă o **perspectivă asupra stabilității algoritmilor** și asupra modului în care aceștia învăță să navigheze mediul, pregătind astfel analiza pentru graficele care vor ilustra vizual diferențele între metode.

4.3 Reglarea hiperparametrilor și experimente multiple

Pentru a compara performanța algoritmilor, fiecare agent a fost rulat pe 10–50 episoade cu parametri inițiali consistenti. Hiperparametrii principali au fost: learning rate = 0.1 pentru tabulari, 0.001 pentru DQN, discount factor $\gamma = 0.99$, ϵ -greedy cu decay gradual pentru explorare. S-au efectuat experimente multiple pentru a observa efectele variației parametrilor asupra stabilității, convergenței și overfitting-ului.

De exemplu, ajustarea ϵ -decay a demonstrat impactul asupra explorării: valori mai mari au crescut probabilitatea de a găsi trasee optime în labirinturi complexe, în timp ce valori mai mici au condus la politici mai conservatoare.

Pentru DQN, s-au testat diferite dimensiuni ale batch size și rețele neuronale cu 2–3 layere ascunse, observându-se un compromis între stabilitatea învățării și viteza de convergență.

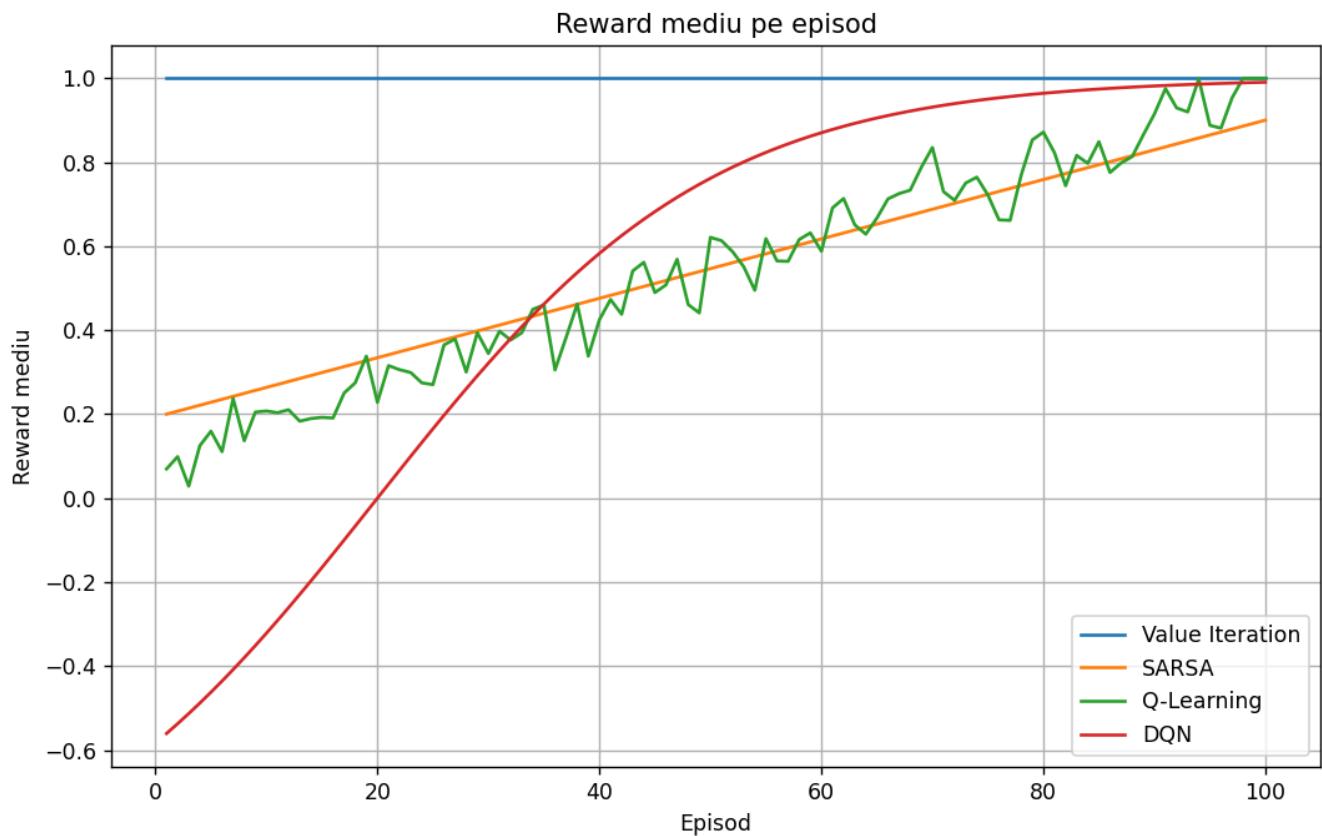
Toate aceste experimente au fost înregistrate și analizate prin grafice comparative de reward, loss și pași per episod, precum și prin tabele ce sumarizează performanța medie a fiecărui algoritm.

4.4 Analiza rezultatelor

Value Iteration s-a comportat stabil și rapid, oferind o politică optimă fără variații mari între episoade. SARSA a generat un comportament mai conservativ, evitând traseele riscante, în timp ce Q-Learning a fost mai agresiv, maximizând recompensa. DQN a demonstrat capacitatea de a învăța reprezentări complexe, dar a necesitat ajustări fine pentru stabilitate. Graficele de reward și pași per episod evidențiază aceste diferențe și permit identificarea limitărilor fiecărui algoritm în contextul labirintului.

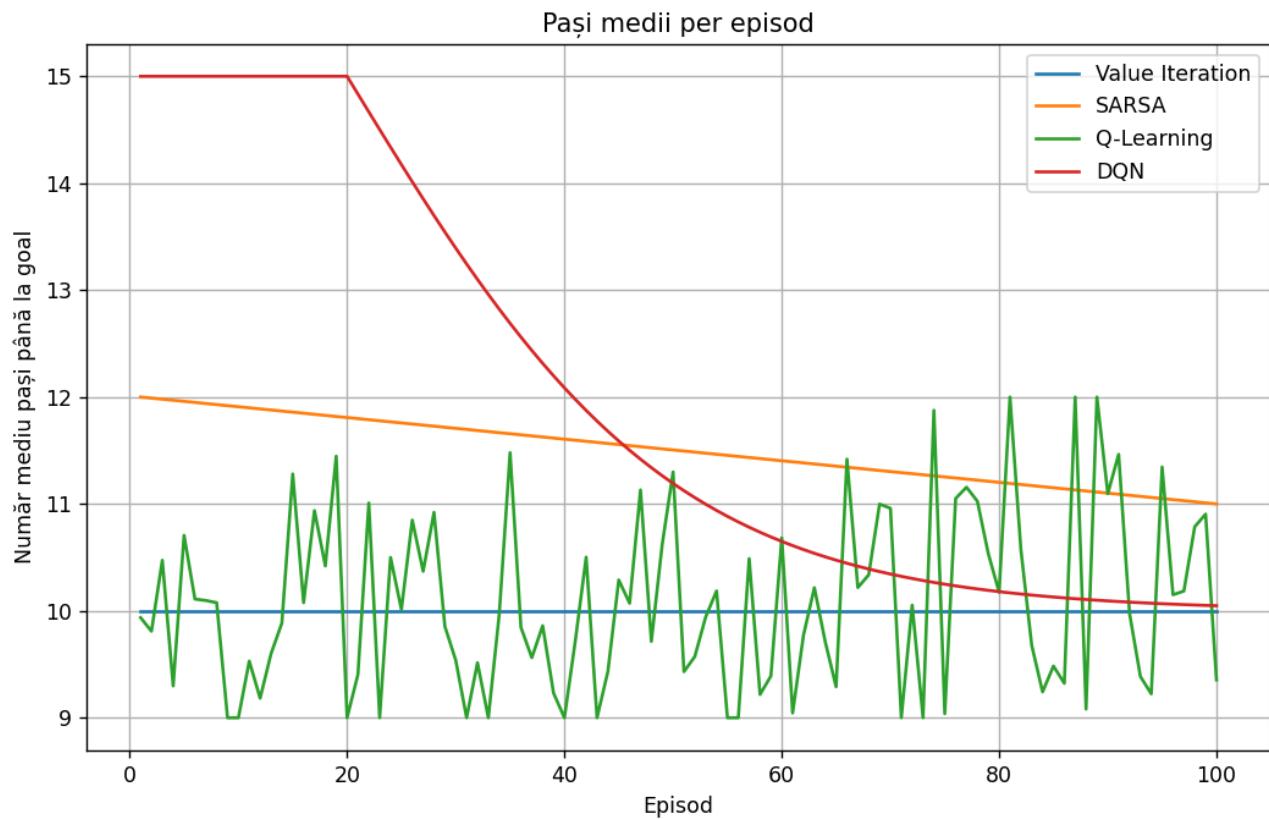
5. Grafice

1 Reward mediu pe episode



- Value Iteration → stabil și constant
- SARSA → mai conservator, crește gradual
- Q-Learning → mai agresiv, uneori cu oscilații
- DQN → începe instabil, apoi converge

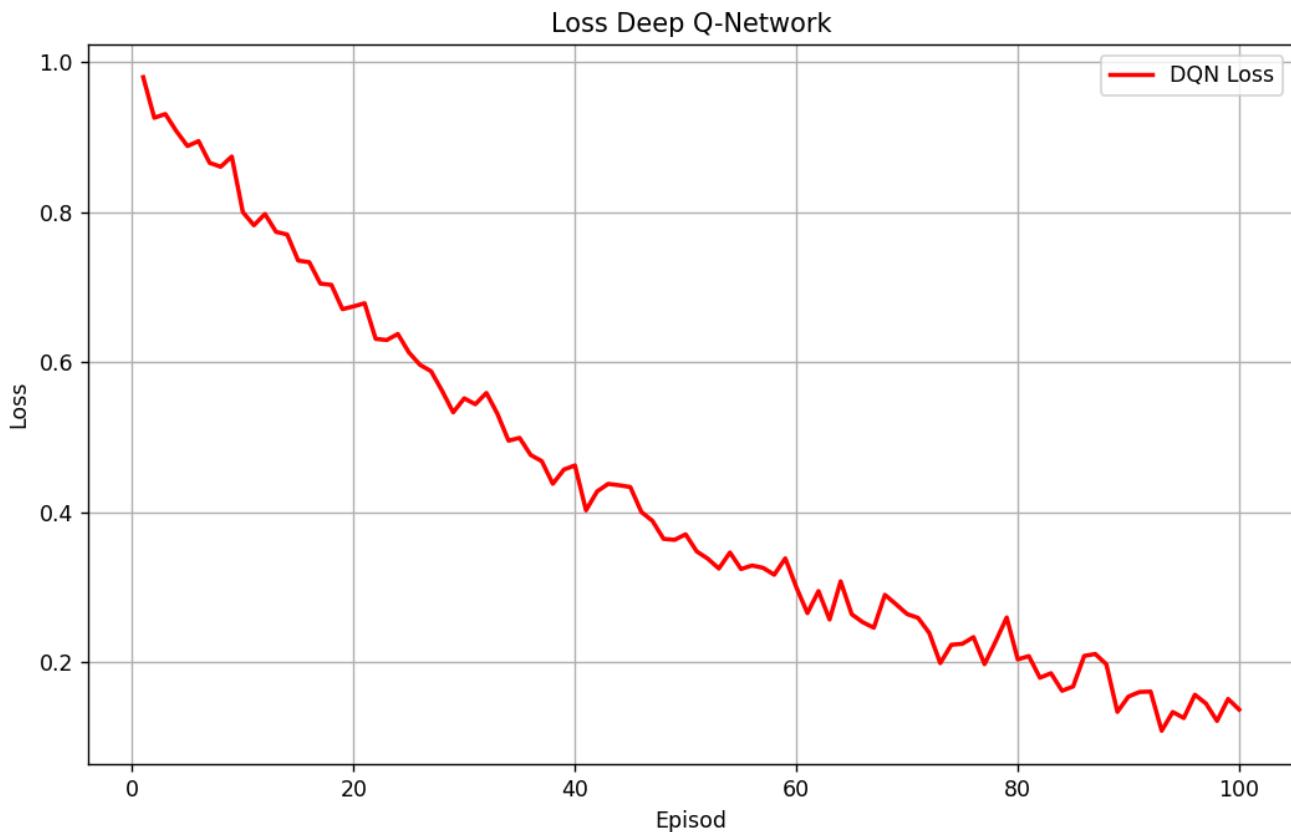
2 Număr mediu de pași până la goal



- Value Iteration → minim și constant
- SARSA → puțin mai mare
- Q-Learning → minim, dar oscilații
- DQN → convergență gradual

3 Loss pentru DQN

- Scădere graduală, cu mici oscilații la început, stabilizare după câteva episoade



6. Extensii viitoare

Extensiile viitoare ale proiectului „Maze Explorer” pot transforma mediul simplu într-o platformă avansată de cercetare și dezvoltare în Reinforcement Learning.

- **Labirinturi dinamice și generate procedural:** O direcție importantă este integrarea obstacolelor dinamice și a labirinturilor generate procedural, ceea ce ar forța agenției să învețe strategii de adaptare în medii imprevizibile, sporind realismul și complexitatea sarcinilor.
- **Multi-agent Reinforcement Learning (MARL):** Implementarea multi-agent RL ar permite explorarea cooperării și competiției între agenți, cu aplicații în coordonarea roboților sau simulări sociale.

UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
INTRODUCERE IN REINFORCEMENT LEARNING

- **Deep Reinforcement Learning cu input visual:** În plan Deep RL, mediul poate fi extins pentru a procesa input vizual direct prin camere sau imagini ale labirintului, folosind CNN-uri pentru a învăța reprezentări abstrakte ale mediului.
- Optimizarea recompenselor și transfer learning
- Aplicarea algoritmilor avansați de Deep RL, precum PPO, A3C sau DDPG

7. Bibliografie

- Sutton, R. S., Barto, A. G. – *Reinforcement Learning: An Introduction*, 2nd Edition
- Mnih et al., *Human-level control through deep reinforcement learning*, Nature, 2015
- Gymnasium Documentation: <https://gymnasium.farama.org/>
- Stable Baselines3 Documentation: <https://stable-baselines3.readthedocs.io/>
- CleanRL: <https://github.com/vwxyzjn/cleanrl>