Raport Tehnic Tema 5 Retele Neuronale

Georgiana Bocancea, Georgiana Matcovici

13 Ianuarie 2026

1 Introducere

In aceasta tema am implementat si antrenat un agent care joaca *Flappy Bird* folosind Q-learning combinat cu o retea neuronala convolutionala (DQN). Scopul a fost ca agentul sa invete sa navigheze printre obstacole, folosind fie imaginile jocului (pixeli), fie frame-uri preprocesate.

Am folosit Python cu librariile `PyTorch`, `Gymnasium`, `OpenCV` si `Pygame`.

2 Mediul de joc

Joc: *FlappyBird-v0* din `flappy_bird_gymnasium`.

Input-ul pentru retea: frame-uri grayscale de 84x84 pixeli, stivuite in 4 frame-uri (`FRAME_STACK = 4`).

Output: actiuni simple: 1 = sari, 0 = nu sari.

Procesare imagine

- Conversie RGB $\rightarrow$ grayscale
- Redimensionare la 84x84
- Normalizare intre 0 si 1

Aceasta procesare ajuta reteaua sa invete mai rapid, eliminand informatii inutile si reducand dimensiunea inputului.

3 Arhitectura retelei neuronale

Am folosit o retea neuronala convolutionala (DQN) pentru a estima functia Q a agentului. Input-ul este format din 4 frame-uri grayscale stivuite, fiecare de 84x84 pixeli.

Structura retelei

- **Conv2D 1:** 32 filtre, kernel 8x8, stride 4, activare ReLU
- **Conv2D 2:** 64 filtre, kernel 4x4, stride 2, activare ReLU
- **Conv2D 3:** 64 filtre, kernel 3x3, stride 1, activare ReLU
- **Flatten** - transformam output-ul convolutional intr-un vector
- **Linear 1:** 3136 $\rightarrow$ 512, activare ReLU
- **Linear 2:** 512 $\rightarrow$ numarul de actiuni (2)

Aceasta arhitectura este inspirata de Deep Q-Network-urile clasice, adaptata pentru input-ul nostru de dimensiune mica (84x84 pixeli). Convolutiile ajuta reteaua sa extraga caracteristici relevante din imagini, fara a fi nevoie de inginerie manuala a feature-urilor.

4 Algoritmul Q-learning

Pentru antrenarea agentului am folosit Deep Q-Learning (DQN), combinat cu **replay buffer** si **epsilon-greedy**.

- **Replay buffer:** stocheaza tranzitiile $(state, action, reward, next\_state, done)$ si permite retelei sa invete din experiente trecute, reducand corelatiile dintre pasi consecutivi si stabilizand antrenamentul.
- **Epsilon-greedy:** agentul alege cu probabilitatea `epsilon` o actiune aleatorie pentru explorare, iar altfel alege actiunea cu Q-maxim pentru exploatare. Valoarea `epsilon` scade treptat (`EPSILON_DECAY = 0.995`) pana la un minim (`EPSILON_MIN = 0.1`).
- **Actualizare tinta:** folosim doua retele, *policy* si *target*, pentru a calcula Q-tinta. Aceasta separare reduce instabilitatea antrenamentului.
- **Functia de pierdere (loss):**

$$target = reward + \gamma \cdot \max(Q(next\_state)) \cdot (1 - done)$$

Hyperparametri importanti

Acesti parametri au fost alesi pentru a obtine o invatare stabila si eficienta:

Parametru	Valoare
GAMMA	0.99
LR	1e-4
BATCH SIZE	32
BUFFER SIZE	100.000
EPSILON START	1.0
EPSILON MIN	0.1
EPSILON DECAY	0.995
TARGET UPDATE FREQ	1000
NUM EPISODES	1000
FRAME STACK	4

5 Experimentare

Am rulat mai multe antrenamente pentru a observa comportamentul agentului:

- `NUM_EPISODES = 20` → agentul nu invata nimic, jocul se incheie imediat
- `NUM_EPISODES = 200` → comportament similar, agentul inca nu reuseste sa evite obstacole
- `NUM_EPISODES = 1000` → scor tot mic, insa agentul incepe sa treaca de primele obstacole

Concluzie: Antrenarea pe pixeli necesita mult timp pentru rezultate vizibile, dar logica Q-learning si replay buffer functioneaza corect.

Nota: Fereastra jocului se inchide automat la pierderea jocului, iar in terminal apare mesajul *Game over*.

6 Observatii

Modelul CNN + Q-learning poate fi imbunatatit prin:

- Cresterea numarului de episoade la antrenare.
- Ajustarea ratei de invatare sau a epsilon decay.
- Folosirea unor feature-uri extrase mai complexe (ex. distanta la tub, viteza verticala).
- Replay buffer si epsilon-greedy sunt necesare pentru convergenta stabila.
- Reducerea frame rate-ului sau skip frame-urilor poate accelera antrenarea.

7 Concluzii

- Agentul este antrenat pe pixeli folosind DQN si Q-learning.
- Replay buffer si epsilon-greedy sunt implementate corect.
- Desi agentul nu reuseste sa evite prea multe obstacole dupa 1000 episoade, logica de antrenare functioneaza si poate fi extinsa pentru rezultate mai bune.
- Acest proiect demonstreaza cum se poate combina reinforcement learning si CNN pentru controlul unui agent intr-un mediu de joc vizual.