**דו"ח טכני – פרויקט משחק SNAKE עם בינה מלאכותית רובוטית**

**1. מבוא**

בפרויקט זה פותח משחק Snake רובוטי הנשלט באמצעות ג'ויסטיק פיזי (חומרה) או באמצעות בינה מלאכותית (AI) המבוססת רשת נוירונים פרה-מאומנת (DQN). מטרת הפרויקט הייתה להטמיע רכיב AI אמיתי המאפשר לשחקן לבחור בין שליטה ידנית לבין שליטה אוטונומית של רובוט (AI) – תוך עמידה בדרישות הקורס.

**2. שימוש ברשת נוירונים פרה-מאומנת**

**א. מבנה הרשת**

הרשת מבוססת על מודל Deep Q-Network (DQN), סוג של רשת נוירונים נפוצה ב-Reinforcement Learning (למידת חיזוק).

המודל מורכב משלוש שכבות fully-connected (dqn\_model.py), עם 13 פיצ'רים קלטים (state), ו-4 פלטים (UP, DOWN, LEFT, RIGHT):

class DQN(nn.Module):  
 def \_\_init\_\_(self, input\_dim=13, output\_dim=4):  
 super(DQN, self).\_\_init\_\_()  
 self.fc1 = nn.Linear(input\_dim, 128)  
 self.fc2 = nn.Linear(128, 64)  
 self.out = nn.Linear(64, output\_dim)  
  
 def forward(self, x):  
 x = torch.relu(self.fc1(x))  
 x = torch.relu(self.fc2(x))  
 x = self.out(x)  
 return x

המודל הזה לומד להתאים פעולה (תנועה) לכל מצב של הנחש על הלוח, באמצעות חיזוק (Reward).

**ב. תהליך האימון ואיסוף הדאטה**

סביבת האימון

האימון בוצע בסביבה מדומה (SnakeEnv) שמדמה את כללי משחק הסנייק: תזוזת הנחש, הופעת אוכל, קירות, התנגשויות, תגמולים וחישוב סטייטים.

הפיצ'רים של ה-state כוללים: זיהוי סכנה בכל כיוון, כיוון תנועה, יחס בין מיקום הנחש לאוכל, אורך הנחש, ועוד (13 פיצ'רים בסה"כ)snake\_env.

איסוף דאטה

הדאטה נאסף תוך כדי אינטראקציה של המודל עם הסביבה. בכל שלב נרשמים:

* המצב (state)
* הפעולה (action)
* התגמול (reward)
* המצב הבא (next\_state)
* האם הסתיים המשחק (done)

כל הדאטה נשמר ב-Replay Buffer, המאפשר למודל ללמוד גם מהיסטוריית צעדים קודמים ומפחית בעיית קורלציה בין דגימותreplay\_buffer.

אימון – Reinforcement Learning

בכל אפיזודה, המודל משחק סנייק מהתחלה עד "מוות". האימון מבוצע לפי DQN:

* תחזית Q לכל פעולה
* עדכון משקל לפי ההפרש בין Q חזוי ל-Q בפועל (Bellman Equation)
* שימוש ב-Target Network ליציבות
* Exploration-Exploitation עם epsilon decay (לומד להתנסות ואז מתמקד בפעולה הטובה ביותר)
* ה-loss מחושב ב-MSE בין Q-Predicted ל-Q-Target.
* המודל מאומן ל-1000 אפיזודות (או יותר), ומאוכסן לקובץ dqn\_snake.pthtrain\_dqn.

שמירת מודל

המודל הסופי (state\_dict) נשמר בדיסק ומשמש את רכיב ה-AI במשחק האמיתי:

torch.save(policy\_net.state\_dict(), "../dqn\_snake.pth")

**3. הטמעת ה-AI במערכת רובוטית**

א. שליטה רובוטית בזמן אמת

המשחק משלב בקרה רובוטית אמיתית: הג'ויסטיק הפיזי מחובר לארדואינו, שמעביר נתוני שליטה ל-PC דרך תקשורת סריאלית. ניתן לבחור בממשק בין "שליטה ידנית" (ג'ויסטיק) לבין "שליטה אוטונומית" (AI) – בלחיצת כפתור.

בעת בחירת AI, כל החלטה של הנחש מתקבלת ע"י המודל המאומן, על בסיס מצב הלוח הנוכחי – תהליך Inference אמיתי:

def get\_ai\_direction(self):  
 state = self.get\_state()  
 with torch.no\_grad():  
 action = model(state).argmax().item()  
  
 return ['UP', 'DOWN', 'LEFT', 'RIGHT'][action]

כך, הרשת מהווה **רכיב במערכת קבלת החלטות רובוטית**, ממש כמו בקרות במערכות פיזיות אמיתיות

ב. אינטגרציה והפרדה בין מצבים

הממשק מציג תפריט בחירה למשתמש:

"Play with Joystick" – שליטה ידנית דרך ג'ויסטיק פיזי (בקר סריאלי).

"Let the AI Play for You" – שליטה אוטונומית באמצעות הרשת הנוירונית.

המודל מופעל בכל צעד מחדש, מקבל את ה-state, מחזיר פעולה, והתנועה מתבצעת על הלוח ובחומרה בהתאם.

במקביל, הקוד תומך בתקשורת דו-כיוונית, שמירת שיאים, תוספות גרפיות, בונוסים, קירות, ועוד.

**תרשים מבנה התיקיה:**

snake\_with\_ai.py – קובץ המשחק עם אפשרות לשליטה ע"י AI או ג'ויסטיק.

snake.py – קובץ משחק ידני.

train\_dqn.py – אימון הרשת הנוירונית.

dqn\_model.py – הגדרת המודל.

agent.py – מימוש Agent RL.

replay\_buffer.py – מחסנית זיכרון לחוויות RL.

snake\_env.py – סביבת סנייק ממומשת ל-RL.