**דו"ח תרגיל סיום – נושאים בבינה מלאכותית איתי ורדי ואיתי פורת**

**מבוא**

במסגרת תרגיל הסיום הזה פיתחנו מערכת ללמידת משחקי Backgammon המשתמשת בלולאת שיפור עצמי (Self-Improvement Loop). המערכת כוללת:

**רשת נוירונים** (HeuristicNet) שמטרתה לחזות ערך מצב לוח – ערך שמתייחס להסתברות ניצחון ולמרווח הניצחון הרשת נמצאת תחת model.py.

**אסטרטגיית שחקנים** המשתמשת בשני סוכנים: סוכן "חדש" (מבוסס על הרשת המעודכנת) וסוכן "ישן" (צילום מסך של הרשת מגרסה קודמת).

**הפקת נתונים** – במהלך משחקי סלפ-פליי נרשמות מצבי לוח מלאים יחד עם סיווג (תווית) שמתבסס על מרווח הניצחון (win margin).המרווח מחושב על ידי כמה שחקנים נשאר לשחקן היריב ביחס אלינו (0.6+ 0.4\*כמות שחקני המפסיד שנותרו/15)

**תהליך אימון** – לאחר הפקת נתונים, הרשת מאומנת מחדש על המידע החדש, תוך מדידת ביצועים והשוואת התקדמות לעומת הגרסה הישנה.

**משחקי סלפ-פליי:** המערכת מריצה משחקים מלאים בין שני סוכנים (הסוכן החדש תמיד משוחק כשחקן 1, והסוכן הישן כשחקן -1 בשביל לוודא זאת השתמשנו במתודה flip\_board).

**רישום מצבי לוח:** במהלך כל משחק נרשמים כל מצבי הלוח יחד עם זהות השחקן שהזיז.

**סיווג תוויות:** בסיום המשחק, מוגדר המנצח ומחשבים את מרווח הניצחון על בסיס מספר הכלים שנותרו לו. כל מצב לוח מקבל תווית:

* + אם השחקן המבצע את המהלך הוא המנצח, התווית היא מרווח הניצחון.
  + אחרת, התווית היא 1 פחות מרווח הניצחון.

**תהליך האימון**

* **אימון הרשת:** לאחר איסוף נתוני המשחק, המערכת מאמנת את הרשת על המידע החדש (הנתונים מסודרים כווקטורים בגודל 28, כאשר 24 ערכים מייצגים את מיקום הכלים ו-4 ערכים נוספים מייצגים נתונים משלימים כגון כלים "אכולים" או כלים שהוצאו מהמשחק). אימנו את הרשת רק בתור שחקן 1 על מנת לאפשר למידה מעמיקה יותר מצד אחד של המשחק והתאמנו את הנתונים למול אילוץ זה
* **השוואת ביצועים:** בכל מחזור (batch) אימון, המערכת מעריכה את ביצועי הסוכן החדש מול הסוכן הישן, ומדווחת על שיעור ניצחונות ומרווח ניצחון ממוצע.
* **עדכון סוכן ישן:** בסיום כל מחזור האימון, הסוכן הישן מתעדכן (על ידי שמירת צילום מסך של הרשת הנוכחית) כדי לאפשר השוואה ברורה בין הגרסאות.

**פרמטרים מרכזיים אותם הרצנו על מנת להגיע לתוצאות הנוכחיות**

* **משך ריצה:** 3600 שניות (שעה מלאה).
* **מספר משחקים למחזור:** 20 משחקים למחזור.
* **מספר אפוקים למחזור אימון:** 20 אפוקים.
* **קצב למידה:** 0.001.
* **גודל batch**.32

**מבנה הרשת**

הרשת מורכבת משלוש שכבות:

1. **Input layer:**  קלט בגודל 28 (24 ערכים ממצב הלוח + 4 ערכים משלימים).
2. **(Hidden Layers):** כל אחת בגודל 40 נוירונים, עם פונקציית אקטיבציה (ReLU)
3. **(Output Layer):** נוירון אחד שמחזיר ערך מספרי בין 0 ל-1 (ללא הפעלה לא לינארית בשכבה זו).

הרשת נועדה לחזות ערך שמייצג את ההסתברות לניצחון או את איכות המהלך, כאשר הערך מקושר למרווח הניצחון בסיום המשחק.

**מבנה הקוד**

הפרויקט מחולק למספר מחלקות פונקציות עיקריות )בהגשה הסופית מפאת נוחות שמנו את כלל הקבצים תחת first\_part:, second\_part)

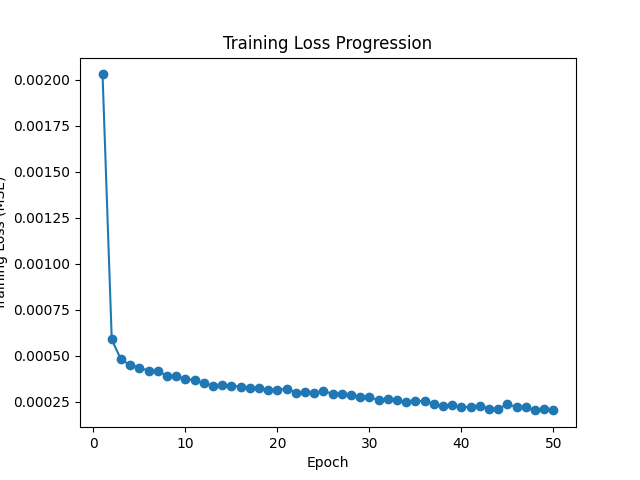
* **model.py:** מגדיר את מבנה הרשת (HeuristicNet) ופונקציית האימון (train\_model).
* **evaluation:** פונקציות להמרת מצב הלוח (board\_to\_state)לערכים מנורמלים, חישוב מרווח ניצחון (evaluate\_win\_margin) ובחירת מהלך מיטבי (choose\_best\_move).באמצעות בחירה רנדומית של 4 המהלכים בעלי הערכים הגבוהים ביותר. ופונקציות עזר לשינוי מצב המשחק (flip\_board). בנוסף לפונקציה היוריסטית הראשונית )pubeval)
* **agents:** מגדיר את פעולות הסוכנים:
  + (NewNetworkAgent) שמשתמש ברשת הנוכחית.
  + (OldNetworkAgent) שמשתמש בסנאפשוט של הרשת.
  + (heuristicAgent) – סוכן שנגדו מודדים את ההתקדמות שלנו לאחר סיום כל batch.
* **self\_improvement\_loop\_extended.py:** מיישם את לולאת השיפור העצמי, אוסף נתוני משחק, מאמן את הרשת ומעריך את ביצועי הסוכנים. בנוסף, מייצר גרפים של התקדמות (שיעור ניצחונות ומרווח ניצחון ממוצע) ושומר את הגרפים לקובץ.
* **test\_data\_generation.py:** קובץ לבדיקת הפקת הנתונים (כולל הדפסה של דוגמאות למשחקים).
* **main.py:** הקובץ הראשי שמריץ את לולאת השיפור העצמי, טוען את המודל (או מתחיל חדש), ומאחסן את המודל המאומן לקובץ.

**תוצאות והערכות**

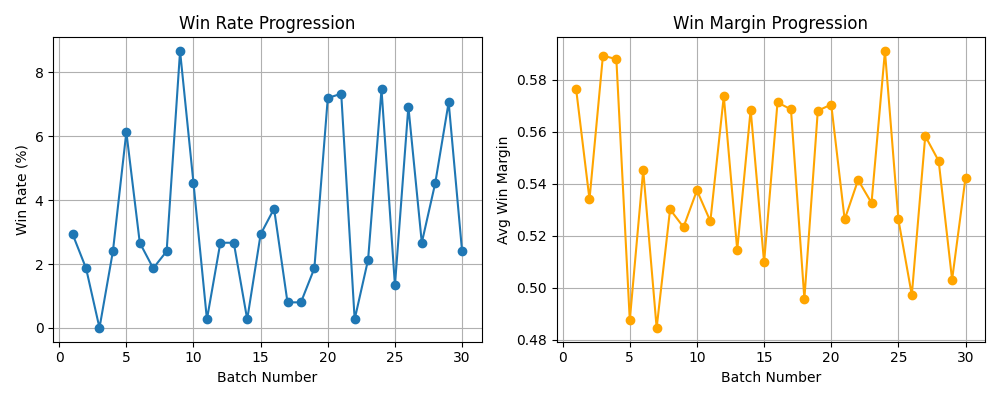
גרפים של שיעור הניצחונות ומרווח הניצחון נוצרו ונשמרו כקובץ progress\_plots\_second\_part   
training\_loss\_progress\_first\_part  
**חלק א:**

**Mean absolute difference: 0.0142**

**Variance of differences: 0.0001**



**חלק ב**



**מסקנות**

בחלק א ניתן לראות כי הרשת אכן לומדת את הפונקציה היוריסטית pubeval. ויורדת בעקביות ככל שמספר האפוקים גדל. הנתונים שהוכנסו לרשת הם:

num\_epochs = 50  
batch\_size = 32  
learning\_rate = 0.001

collect\_data(n\_boards = 2000)

לאחר מכן הטסטים בוצעו על מידע חדש שהופק על ידי הקריאה

collect\_data(n\_boards = 200)

בחלק ב הנתונים שהוכנסו הם:

run\_duration=600,  
 n\_games\_per\_batch=100,  
 batch\_training\_epochs=32,  
 learning\_rate=0.001,

כמו כן הנתונים המוצגים תחת win rate progression נאספו לאחר ביצוע 10 משחקי טסט בסוף ריצת כל אפוק אל מול הרשת היוריסטית pubeval. הרשת למדה על ידי ביצוע משחקים מול סנאפשוט של המודל הישן. אימנו את המודל מ0 (כלומר בלי להתחיל מהלמידה של השלב הקודם) בעיקר כדי לשקף תוצאות משחקים שמראות התקדמות אל מול pubeval וכדי לראות כיצד הרשת באמת לומדת. כאן ניתן להסיק מספר מסקנות.  
תחילה ניתן לראות הרבה מאוד רעש של תוצאות הדבר יכול לנבוע מ2 סיבות מרכזיות.  
הראשונה היא קטלוג הלוחות בצורה מאוד גולמית, כיוון שאין באמת אבחנה בין אלו מהלכים במהלך המשחק היו טובים יותר ואילו טובים פחות קשה מאוד לייצר התקדמות שמתבססת אך ורק על בסיס נתונים אלו. את זה ניתן היה לשפר אילו היינו מחשבים בכל שלב הערכה יוריסטית נוספת שמתבססת על MCTS ונותנים LABEL שהיה משקף גם את תוצאת המשחק הסופית וגם את הערך של MCTS. הסיבה השנייה היא ששבש הוא משחק רנדומי ולכן ניתן לראות שינויים בין batch לbatch. את בעיה זו ניתן לפתור על ידי הגדלת המשחקים n\_games\_per\_batch

כדי שחוק המספרים הגדולים בסטטיסטיקה יבוא לידי ביטוי.

מסקנה נוספת מהנתונים היא שעל אף הרעש ניתן לראות שיפור קל ברשת לאורך זמן ולכן אם ניתן לרשת מספיק זמן כדי להתאמן ניתן יהיה לראות תוצאות יפות יותר.

**סיכום**

פיתחנו מערכת מתקדמת שמשלבת למידה מבוססת חיזוקים ושיפור עצמי, אשר הראתה מגמות שיפור שעל אף שאינן ברורות בהחלט קיימות. הפרויקט מציג את החשיבות של הפקת נתונים מהמשחק והעדכון התמידי של הרשת בעתיד ניתן לשפר עוד יותר את המערכת על ידי שילוב שיטות נוספות כגון MCTS, התאמת הארכיטקטורה והרחבת מאגר הנתונים או מתן זמן למידה משמעותי יותר לתוצאות.