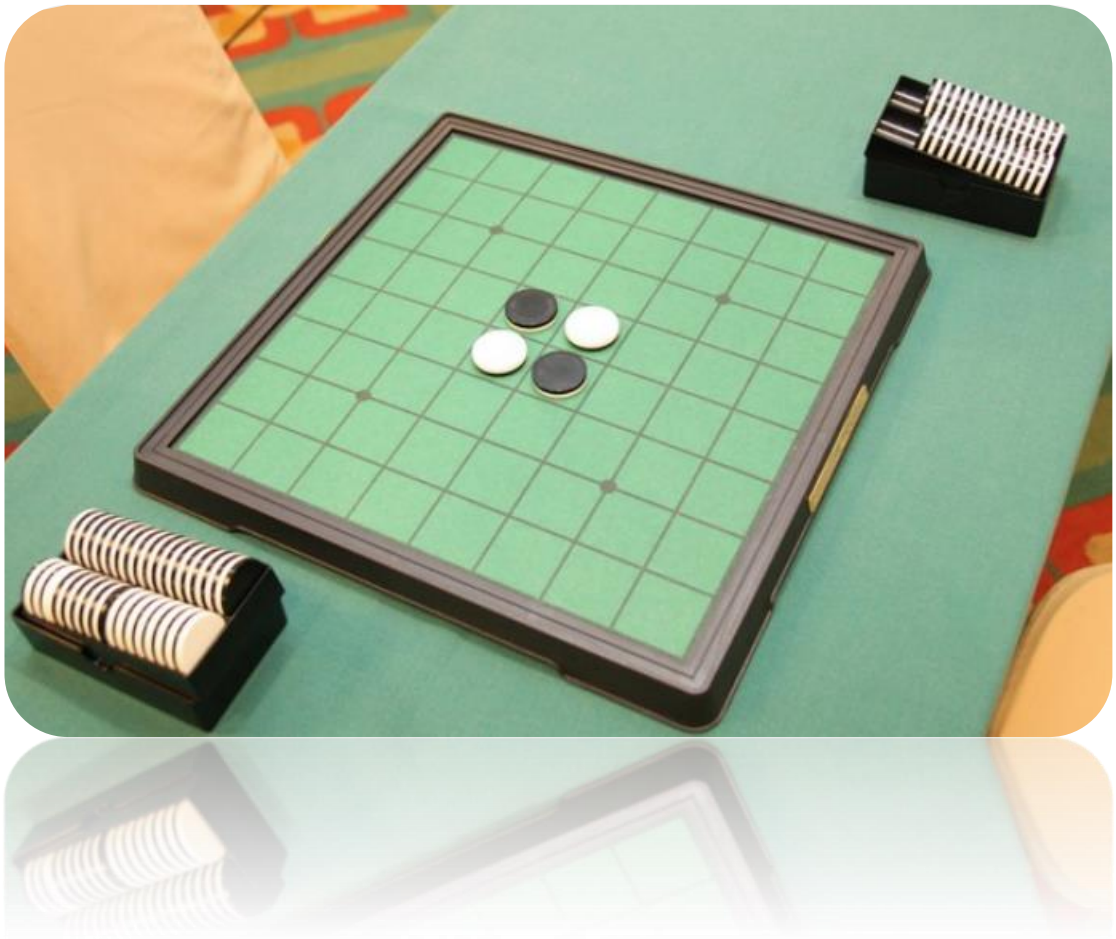
Reversi



תקציר

בפרויקט זה, החלטנו לממש סוכנים שונים עבור משחק הלוח רברסי (או אותלו). הסיבה שהמשחק רברסי סיקרן אותנו היא שאין פתרון טריוויאלי למשחק. אומנם קיימת אסטרטגיה מנצחת למשחק, הרי הוא אינו מבוסס על מזל, אך היא עדיין לא נמצאה. כמו כן זה משחק שמשחקים באליפויות עולם. נציין שלמרות שחוקי המשחק יחסית פשוטים, הניקוד בו יכול להשתנות בכל רגע.

בדו"ח מפורטים חוקי המשחק בגרסה המקורית ובגרסאות היצירתיות שיצרנו, מבנה המשחק, האסטרטגיות שעומדות מאחורי בחירת מהלכים שקולים- שעליהן מתבססות היוריסטיקות שבנינו, והסוכנים שיצרנו. הסוכנים שבהם בחרנו להתמקד הם:

- הסוכן הרנדומאלי (אותו יצרנו לשם השוואת הביצועים לסוכנים האחרים).
- סוכן ה- Heuristic (המקבל היוריסטיקה ומבצע כל מהלך לפי הניקוד הגבוה ביותר שהיוריסטיקה נותנת מתוך כלל המהלכים החוקיים).
- סוכני ה- MinMax עם היוריסטיקות שונות.
- סוכן ה- DeepQlearner MinMax שמקבל 6 מודלים שאומנו לפי Deep Q learning, ומשקלל את תוצאתם וכך בוחר את המהלך הבא.
- סוכן ה- DeepQlearner Heuristic שמקבל 6 מודלים שאומנו לפי Deep Q learning, ומשקלל את תוצאתם וכך בוחר את המהלך הבא.

על מנת לבחון את הביצועים של הסוכנים, החלטנו להשוות בין כל זוג סוכנים (לפי מספר הניצחונות), ובכך להסיק למי מהם יש את הביצועים הטובים ביותר. ראינו ששני המודלים האחרונים הם הטובים ביותר. במהלך הדו"ח ניתחנו את התוצאות ובססנו את המסקנות על החומר הנלמד. בנוסף, השווינו בין חוקי המשחק השונים שיצרנו על מנת לבדוק את השפעת כל אחד מהם על ביצועי הסוכנים. כפי שציפינו כאשר היה מדובר בלוח המקורי הסוכנים DeepQlearner minmax, DeepQlearner heuristic הצליחו ללמוד היטב את המשחק, וכאשר היה מדובר בלוחות היצירתיים שבנינו, בהם ישנם אלמנטים רנדומאליים הנותנים יתרונות לאחד הסוכנים, המודלים התקשו ללמוד והתוצאות השתנו.

מבוא

רברסי או אותלו הוא משחק לוח חשיבתי ותיק ונפוץ, עבור שני שחקנים. כלי המשחק הם דיסקיות שצדה האחד של כל אחת מהן שחור וצדה השני לבן. כל משתתף מניח בתורו דיסקית אחת כך שצבעה הגלוי הוא הצבע המזוהה עמו. כאשר הוא מניח דיסקית זו, עליו להפוך את כל הדיסקיות מהצבע הנגדי המצויות בין דיסקית זו לבין דיסקית נוספת מצבעו שלו. מטרתו של כל שחקן היא למלא את הלוח בכמה שיותר דיסקיות מהצבע המזוהה עמו (מתוך ויקפדיה). המשחק מסתיים כאשר אין מהלך חוקי לאף אחד מהשחקנים.

כפי שציינו, רברסי הוא משחק אסטרטגיה שאינו מבוסס על מזל. כיום יש מספר תוכנות שמשחקות מול שחקנים ותחום זה נהיה מאוד פופולרי בשנים האחרונות. נציין, שמספר הדיסקיות שכרגע נמצאות על הלוח לא מעידות על ניצחון של שחקן מסוים או על כך שהוא מוביל, מכיוון שהן עשויות להתהפך לטובת היריב בכל רגע במשחק. לכן, קשה לחזות מי השחקן המוביל . כיום עדיין לא נמצאה שיטה מוצלחת להבטחת ניצחון, וישנן רמות שונות של שחקנים.

בפרויקט שלנו, בנוסף ללוח ולחוקי המשחק המקוריים, יצרנו מספר אפשרויות משחק נוספות. מימשנו סוכנים שונים, עם היוריסטיקות שונות ועם פרמטרים שונים (עליהם נפרט בהמשך המסמך), על מנת שנוכל להשוות ביניהם ולהסיק מסקנות.

המשחק שאנו נחקור הוא עם לוח בגודל 8X8. ישנן גרסאות פשוטות יותר למשחק:

1. לוח 4X4- ללוח זה יש עץ חיפוש קטן מאוד, וישנן תוכנות רבות שמוצאות אסטרטגיית משחק בפחות משנייה (בעזרת MinMax למשל).
2. לוח 6X6- ישנן תוכנות רבות שמוצאות אסטרטגיית משחק בפחות מ-100 שעות (בעזרת MinMax למשל).

עבור לוח 8X8, עץ החיפוש גדול מאוד ולא נמצאה עדיין אסטרטגיית משחק, אך ישנם הרבה פרמטרים ושיקולים שנרחיב עליהם בהמשך, שיכולים לעזור בדרך לפיתרון.

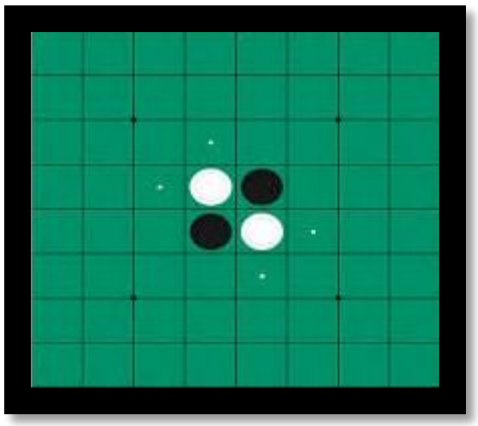
מבנה הלוח:

לוח משבצות בגודל 8X8.

על מנת להפוך את הפרויקט ליצירתי יותר, ולנתח את דרכי פעולתם של הסוכנים השונים (עם היוריסטיקות השונות) בצורה מעמיקה, יצרנו לוחות עם חוקים שונים מחוקי המשחק המקורי.

חוקי המשחק – הגרסא המקורית:

1. עמדת הפתיחה של המשחק: 4 דיסקיות במרכז הלוח, 2 לכל שחקן (על כל אלכסון צבע שונה). השחקן שמתחיל הוא השחקן השחור.



2. כל שחקן רשאי לשים דיסקית מהצבע שלו בכל מקום פנוי הצמוד לדיסקית כלשהי של היריב, בתנאי שהמהלך מאפשר לו להפוך לפחות דיסקית אחת של היריב. השחקן הופך את כל הדיסקיות של היריב שסגר בעזרת המהלך הנוכחי, כלומר כל רצף דיסקיות היריב בין הדיסקית שכרגע הניח לבין דיסקית אחרת בצבע שלו שכבר היתה על הלוח לפני המהלך. במידה והסגירה מתבצעת ביותר מכיוון אחד (שורה, עמודה או אלכסון), חובה להפוך בכל הכיוונים. במידה ואין אף מהלך חוקי, התור עובר ליריב.

חוקי המשחק – הגרסאות שלנו

גרסא 1 –RandomStart

1. עמדת הפתיחה של המשחק: מיקום ארבעת הדיסקיות שנמצאות על הלוח בפתיחת המשחק, מוגרל.

שינוי זה עשוי לתת יתרון משמעותי לאחד השחקנים, ובכך להפוך את האימון של השחקן היריב לקשה יותר. כמובן שבהגרלה, הסרנו מקומות שבאופן מובהק יתנו יתרון משמעותי לאחד השחקנים (כמו פינות למשל).

2. שאר החוקים נשארים זהים.

גרסא 2 - FlipColor

1. עמדת הפתיחה נשארת זהה.

2. בנוסף לחוקים מהגרסה המקורית, הוספנו חוק נוסף: מגרילים בתחילת המשחק 4 מקומות בלוח. אם השחקן הלבן מניח דיסקית באחד המיקומים שהוגרל, מיקום זה הופך לדיסקית שחורה, ולהפך.
השינוי בחוקים עשוי להוביל לשינוי בנקודות, לפגוע באחד השחקנים או להעניק יתרון ליריב.

נשים לב שבכל הגרסאות חוקי המשחק הם פשוטים לכאורה, אך נדרשת מחשבה רבה (שתלויה בהרבה גורמים עליהם נפרט בחלק בו אנו מפרטים על היוריסטיקות) כדי להגיע לאסטרטגיה מנצחת. לכן, החלטנו לממש סוכני AI שונים, שיפעלו על פי אסטרטגיות מסוימות.

<u>היוריסטיקות</u>

היורסטיקות שבנינו הן היוריסטיקות המבוססות על אסטרטגיות נפוצות של המשחק.

נציג את היורסטיקות שבנינו לשימוש הסוכנים. נציין שכל היורסטיקה נורמלה על מנת שתהיה התכנסות של המודלים בהמשך (אחרת, ה-losses נעו בטווחים שלא אפשרו התכנסות).

1. **Base heuristic**- זו היורסטיקה הפשוטה ביותר. היא סופרת כמה דיסקיות יש לשחקן הנוכחי על הלוח, ומוסיפה לו נקודה לחישוב עבור כל אחת מהן. בנוסף, יורדת נקודה עבור כל דיסקית של היריב.
2. **Filled cell heuristic** – עבור כל מיקום בלוח, חישבנו את השכנים שלו. עבור כל אחד משכניו, בדקנו מי נמצא בתא זה- אם מדובר בשחקן הנוכחי הוספנו 2 לחישוב. אם מדובר בשחקן היריב הוספנו 1 לחישוב. אם מדובר במשבצת ריקה, הורדנו 2 מהחישוב.
היוריסטיקה הזו מתבססת על הרעיון שלדיסקיות שנמצאות בסמוך לתאים ריקים יש יותר סיכוי להתהפך על ידי היריב. לכן, אנו מנסים שיהיו כמה שפחות משבצות ריקות ליד הדיסקית.
3. **Legal moves heuristic**- היוריסטיקה הזו מחזירה את מספר המהלכים החוקיים שיש לשחקן הנוכחי פחות מספר המהלכים החוקיים שיש לשחקן היריב ובהתאם לכך נקבע הניקוד. לפיכך, לשחקן מסוים יהיה יתרון וניקוד גבוה כאשר ליריבו יהיו פחות מהלכים חוקיים.
4. **Vulnerable heuristic**- היורסטיקה הזו מחשבת כמה סביר שהדיסקית הזו תתהפך בהמשך המשחק. עבור כל אחד מתאי הלוח, בודקים האם יש סיכוי שהשחקן היריב יוכל להפוך את הדיסקית בתור הבא. אם יש סיכוי כזה, מורידים מהניקוד 2. כמו כן מוסיפים לניקוד 1, אם השחקן הנוכחי יכול להפוך דיסקית של היריב.

5. **Weight place heuristic**- היוריסטיקה נותנת לכל משבצת משקל, לפי היתרון שהיא נותנת לשחקן: למשל, פינות בעלות חשיבות מאוד גבוהה כי לא ניתן להפוך אותן, ועל כן יקבלו משקל גבוה. לעומת זאת, משבצות ליד הפינות הן מאוד פגיעות, כי הן מאפשרות לשחקן היריב להפוך את הדיסקית של השחקן הנוכחי ולתפוס את הפינה. גם למסגרת של הלוח יש יתרון, מכיוון שניתן להפוך את הדיסקית רק מכיוון אחד (אנכי או אופקי), כלומר משבצות אלה פחות פגיעות. בנוסף 4 המשבצות האמצעיות של הלוח, מעורבות בהרבה אלכסונים, עמודות ושורות, ולכן החזקתן נותנת יתרון במשחק. בהתאם למשקול שקבענו, עבור כל משבצת אם מדובר בשחקן הנוכחי, מתווסף לו ניקוד ואם מדובר ביריב, יורד הניקוד.
6. **Weight combined heuristic**- היוריסטיקה הזו משלבת את כלל היוריסטיקות המצוינות מעלה. לאחר שחקרנו את ביצועי היוריסטיקות, נתנו משקול לניקוד שלהן בהתאם. סך הכל התוצאה המוחזרת היא חיבור כל הניקוד הממושקל.

<u>סוכנים</u>

החלטנו לבחון 4 סוגי סוכנים:

1. **סוכן רנדומאלי**- סוכן זה בודק מהן האפשרויות למהלכים חוקיים. מתוכן, הוא מגריל אופציה אחת ומבצע אותה. השתמשנו בסוכן זה על מנת שנוכל להשוות את ביצועיו לביצועים של סוכנים מורכבים יותר.
2. **סוכן Minmax** [1]- בתורת המשחקים ,משפט המינימקס העוסק במשחק סכום אפס סופי לשני שחקנים, אומר כי לכל משחק מסוג זה קיימת דרך פעולה אופטימלית לשחק מבחינת שני השחקנים, כך שהרווח המינימלי של כל אחד אינו תלוי במעשי השני. סוכן זה מורכב מעץ, הפורש את האפשרויות למשחק של שחקן א', את תגובותיו של שחקן ב' לכל פעולה של שחקן א', את תגובותיו של א' לתגובותיו של ב', וכן הלאה, בהתאם לעומק העץ. העלים של עץ פורש זה, הם מצבים שאליהם נגיע לאחר רצף של מהלכים. לכל מצב כזה, ניתן ציון בהתאם ליוריסטיקה. העלה הטוב ביותר, הוא העלה האופטימאלי עבור השחקן הנוכחי. בחרנו שהסוכן ישתמש באלגוריתם alpha-beta pruning [2], זהו אלגוריתם חיפוש שמנסה למזער את מספר הקודקודים שמוערכים על ידי אלגוריתם MinMax באמצעות דילוג על קודקודים שלא יביאו לתוצאה טובה יותר.

---

[1] MinMax
[2] Alpha beta pruning

כפי שציינו, סוכן זה צריך לקבל עומק ויוריסטיקה. היוריסטיקות שבהן החלטנו להשתמש עבור MinMax הן: Weight place heuristic, Weight combined heuristic. הסיבה שבחרנו ביוריסטיקות אלה, מסתמכת על מה שלמדנו בביתה ועל מה שראינו בתרגילי הבית- סוכן זה עובד טוב יותר עבור היוריסטיקות מסובכות.

אלגוריתם זה גורם לנו לבצע בחירות באופן רקורסיבי. הרעיון המרכזי שלו הוא לעשות את הבחירות הכי טובות עבורנו, מתוך הנחה שגם היריב יפעל באופן אופטימאלי.

3. **סוכן Heuristic**- סוכן זה מקבל היוריסטיקה ובודק מה הם המהלכים החוקיים שברשותו. עבור כל מהלך חוקי כזה, הוא מחשב את הניקוד בעזרת היוריסטיקה שקיבל. בסוף, הוא בוחר במהלך שהיוריסטיקה ניקדה כגבוה ביותר. היוריסטיקה שבה החלטנו להשתמש עבור סוכן זה היא: Weight combined heuristic. הסיבה שבחרנו ביוריסטיקה זו, מסתמכת על כך שהסוכן הוא יחסית פשוט ולא מתחשב הרבה בעתיד המשחק. לכן, החלטנו לתת לו היוריסטיקה מורכבת יותר.
4. **סוכן DeepQlearner MinMax-** על מנת לבנות את סוכן זה, השתמשנו ב- deep Q learning:

תחילה נדון בסוכן **Deep Q-learner**. סוכן זה, מוגדר ע"י הפונקציה $Q^*(s,a)$ אשר נותנת ניקוד עבור הפעולה a מהמצב s. הסוכן מתחזק טבלה בגודל מספר המצבים X מספר הפעולות. עבור כל מיקום בטבלה הסוכן מקבל תגמול על כל פעולה בסביבה, ומשתמש בתגמול על מנת לעדכן את הפונקציה $Q^*(s,a)$. העדכון מתבצע בעזרת נוסחת Bellman equation:

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\underbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}\right.}_{\text{new value (temporal difference target)}} \left. - \underbrace{Q(s_t,a_t)}_{\text{old value}}\right)}^{\text{temporal difference}}$$

- כפי שלמדנו בביתה, Q-learning לא דורש "מידע עתידי" מהעולם. בנוסף, פעולות לא טובות לא הורסות את המידע שלו על העולם. כפי שכבר הסברנו, הלמידה ב-Q learner נובעת מהתגמול שאנו מקבלים עבור כל פעולה. הסוכן הזה, מבצע פעולות רנדומאליות כדי לחקור וללמוד את הסביבה(Exploration), ובנוסף פועל לפי החוקיות שפיתח (Exploitation). סוכן זה מבצע trade-off בין שתי הדרכים הנ"ל:
- Exploration
- Exploitation

מי ששולט ב-trade off הנ"ל, הוא הפרמטר epsilon. בחרנו שהוא יאותחל עם הערך 1, מכיוון שבהתחלה נרצה שהמודל "ילמד" מידע על העולם, ויפעל באופן רנדומאלי, מכיוון שחסר לו המידע הנחוץ. בכל איטרציה, פרמטר זה קטן (על ידי פרמטר decay שקבענו), עד שבשלב שבו אנו מאמינים שהמודל אסף מספיק מידע על העולם, הוא יכול לסמוך על בחירותיו ללא צורך בביצוע פעולות רנדומאליות.

סוכן זה מקבל היורסטיקה ושני שחקנים. סוכן זה מאמן רשת ניורונים על המשחק של שני השחקנים ומנסה ללמוד את ערכי הq של כל אחד מהמצבים במשחק לפי שחקן א' ולפי היוריסטיקה שקיבל. מכיוון שמספר המצבים ברברסי גדול מאוד, לא ניתן ללמוד את ערכי הQ האמיתיים כשרשרת מרקוב סופית. לכן, החלטנו להשתמש ברשתות נוירונים שיחזו את ערכים אלו.

Deep Q learner הוא סוכן, שמשתמש ברשת נוירונים כדי לשערך את הפונקציה עליה דיברנו מעלה. הוא מעדכן את הערך בצורה הבאה:

$$Q(s,a) = r(s,a) + \gamma \max_{a} Q(s',a)$$

במימוש שלנו, יצרנו רשת נוירונים הבנויה מהשכבות הבאות:

- שכבת input עם shape (1,8,8).
- שכבת Fully connected שהופכת את האינפוט לגודל 256.
- שכבת Fully connected שהופכת את האינפוט לגודל 128 (עם אקטיבציית "relu6").
- שכבת Fully connected שהופכת את האינפוט לגודל 64 (עם אקטיבציית "linear").
- שכבת Fully connected שהופכת את האינפוט לגודל 32 (עם אקטיבציית "linear").
- שכבת Fully connected שהופכת את האינפוט לגודל 16 (עם אקטיבציית "linear").
- שכבת Fully connected שהופכת את האינפוט לגודל 1 (עם אקטיבציית "linear") – ה- Q-value.
- השתמשנו בoptimizer שנקרא "Adam" וב- loss מסוג "MSE".

ייצגנו את ה-state על ידי וקטור מגודל 8X8 שהיה האינפוט לרשת. בכל מיקום בוקטור היה מספר שייצג את הדיסקית בלוח: 1 אם שחקן 1 נמצא במיקום זה, 2 אם שחקן 2 נמצא במיקום זה, 0 אם זו משבצת ריקה.

תהליך האימון של הרשת: בהתחלה אימנו את המודלים על 1000 משחקים. לאחר כמה הרצות, ראינו שה-loss מתכנס לאחר מספר קטן יותר של משחקים, ולכן התחלנו לאמן את שאר המודלים על 500 משחקים, לפי האסטרטגיה של שחקן א', והיוריסטיקה שבה בחרנו. הקלט לרשת יהיה וקטור מגודל 8X8 שבו מיוצגים המיקומים של שחקן א', המיקומים של שחקן ב', ומיקומים ריקים.

כעת, נפרט על הסוכן **DeepQlearner MinMax**: סוכן זה, מקבל את 6 המודלים המאומנים הבאים, כאשר בכולם הרשת לומדת לפי שחקן א', שהוא שחקן ה-MinMax:

- **Model 1**- שחקן ב' הוא השחקן הרנדומאלי. MinMax משתמש ביוריסטיקה Weight combined heuristic. Deep Q learner משתמש ביוריסטיקה Base heuristic.
- **Model 2**- שחקן ב' הוא השחקן הרנדומאלי. MinMax משתמש ביוריסטיקה Weight place heuristic. Deep Q learner משתמש ביוריסטיקה Filled cell heuristic.
- **Model 3**- שחקן ב' הוא השחקן MinMax. שני שחקני ה-MinMax משתמשים ביוריסטיקה Weight combined heuristic. Deep Q learner משתמש ביוריסטיקה Base heuristic.
- **Model 4**- שחקן ב' הוא השחקן MinMax. שני שחקני ה-MinMax משתמשים ביוריסטיקה Weight place heuristic. Deep Q learner משתמש ביוריסטיקה filled cell heuristic.
- **Model 5**- שחקן ב' הוא השחקן Heuristic. Heuristic player, Minmax משתמשים ביוריסטיקה Weight combined heuristic. Deep Q learner משתמש ביוריסטיקה Base heuristic.
- **Model 6**- שחקן ב' הוא השחקן Heuristic. Heuristic player, Minmax משתמשים ביוריסטיקה Weight placed heuristic. Deep Q learner משתמש ביוריסטיקה filled cell heuristic.

בכל פעם שסוכן זה צריך לבחור את הצעד הבא, הוא בודק מה הם כל המהלכים החוקיים שהוא יכול לבצע. עבור כל אחד מהם, הוא מחשב את סכום ערכי ה-Q, שכל אחד מ-6 המודלים מעלה נותן. לבסוף, הוא בוחר את הצעד עבורו הסכום הנ"ל מקסימאלי.

5. **סוכן DeepQlearner Heuristic**: על מנת לבנות את סוכן זה, השתמשנו ב- deep Q learning.

כעת, נפרט על הסוכן DeepQlearner Heuristic: סוכן זה, מקבל את 6 המודלים המאומנים הבאים, כאשר בכולם הרשת לומדת לפי שחקן א', שהוא שחקן ה-Heuristic:

- **Model 7**- שחקן ב' הוא השחקן הרנדומאלי. Heuristic player משתמש ביוריסטיקה Weight combined heuristic. Deep Q learner משתמש ביוריסטיקה Base heuristic.
- **Model 8**- שחקן ב' הוא השחקן הרנדומאלי. Heuristic player משתמש ביוריסטיקה Weight place heuristic. Deep Q learner משתמש ביוריסטיקה Filled cell heuristic.
- **Model 9**- שחקן ב' הוא השחקן MinMax. ה-MinMax, Heuristic player משתמשים ביוריסטיקה Weight combined heuristic. Deep Q learner משתמש ביוריסטיקה Base heuristic.
- **Model 10**- שחקן ב' הוא השחקן MinMax. ה-MinMax, Heuristic player משתמשים ביוריסטיקה Weight place heuristic. Deep Q learner משתמש ביוריסטיקה filled cell heuristic.
- **Model 11**- שחקן ב' הוא השחקן Heuristic.שני שחקני ה- Heuristic משתמשים ביוריסטיקה Weight combined heuristic. Deep Q learner משתמש ביוריסטיקה Base heuristic.
- **Model 12**- שחקן ב' הוא השחקן Heuristic.שני שחקני ה- Heuristic משתמשים ביוריסטיקה Weight place heuristic. Deep Q learner משתמש ביוריסטיקה filled cell heuristic.

בכל פעם שסוכן זה צריך לבחור את הצעד הבא, הוא בודק מה הם כל המהלכים החוקיים שהוא יכול לבצע. עבור כל אחד מהם, הוא מחשב את סכום ערכי ה-Q,

שכל אחד מ-6 המודלים מעלה נותן. לבסוף, הוא בוחר את הצעד עבורו הסכום הנ"ל מקסימאלי.

נציין שבחרנו עבור ה-Deep Q learner היוריסטיקות יחסית פשוטות (Base heuristic, filled cell heuristic) מכיוון שכך הוא למד טוב יותר והגיע לביצועים מרשימים יותר. עבור היוריסטיקות מסובכות כמו המשולבת, הוא התקשה ללמוד כראוי.

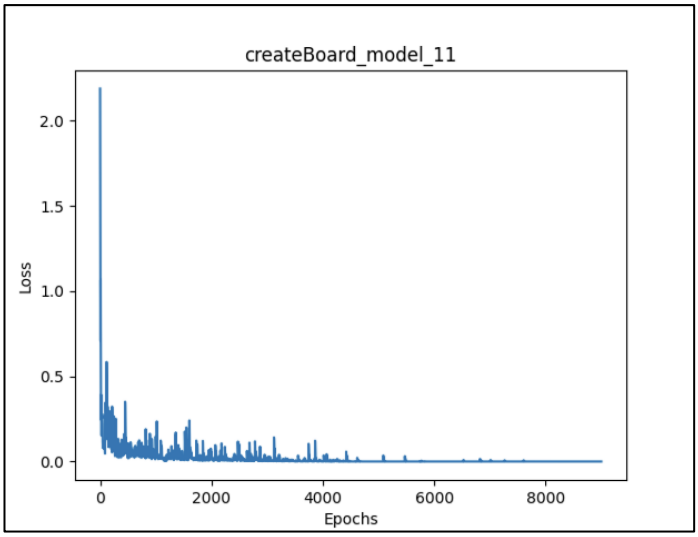
הערכת תוצאות

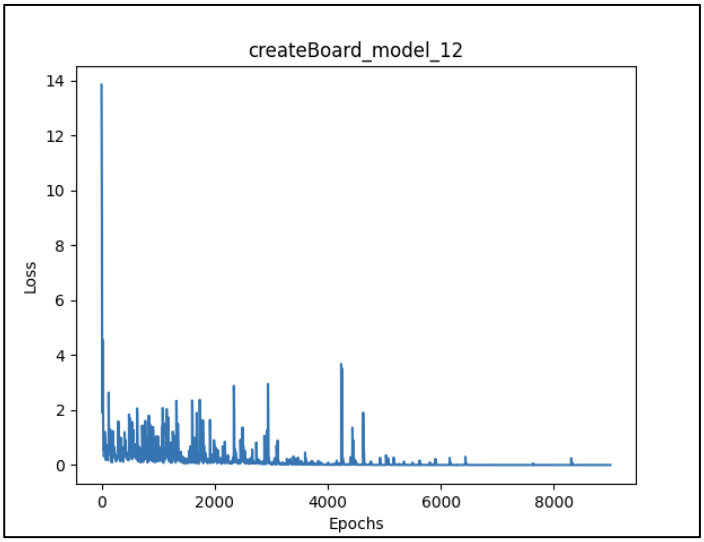
על מנת להשוות את ביצועי הסוכנים ולהסיק מסקנות אודות ביצועיהם בלוחות השונים, החלטנו לפעול באופן הבא: עבור כל אחד משלושת הלוחות, הרצנו בין כל שני שחקנים 100 משחקים, וספרנו את מספר הניצחונות של כל אחד מהם מתוך כלל המשחקים (עבור תיקו לא התווסף ניקוד לאף אחד מהסוכנים). במקום הראשון נמצא שחקן שניצח את כל האחרים ושממוצע הנצחונות שלו הוא הגבוה ביותר, ובאחרון נמצא שחקן שהפסיד לכל האחרים ושממוצע הנצחונות שלו הוא הנמוך ביותר.

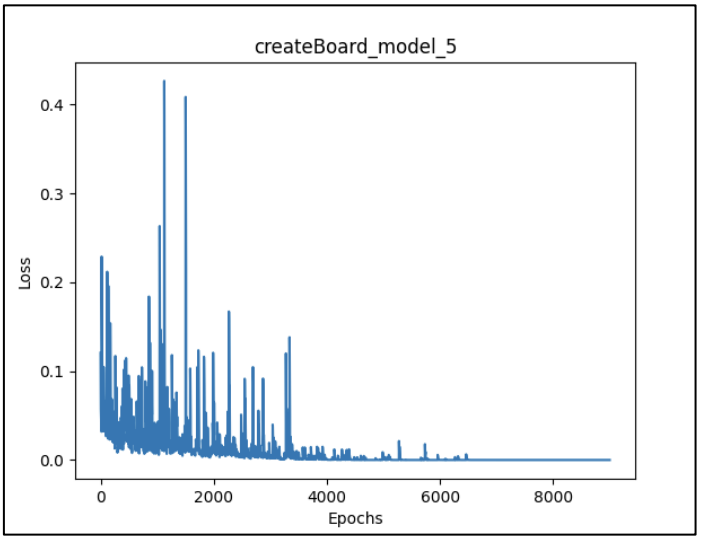
עבור הגרסה המקורית, הממומשת במחלקה- CreateBoard קיבלנו את התוצאות הבאות:

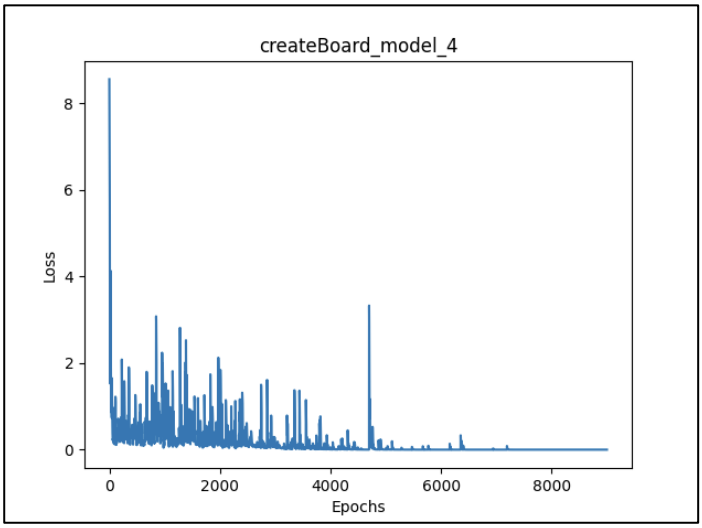
Player 1 / Player 0	Random	MinMax Weight place heuristic	MinMax Weight combined heuristic	Heuristic	DeepQlearner MinMax	DeepQlearner Heuristic
Random		0.73 / 0.27	0.49 / 0.46	0.71 / 0.29	0.77 / 0.23	0.76 / 0.23
MinMax Weight place heuristic			0.41 / 0.57	0.77 / 0.23	0.8 / 0.2	0.55 / 0.45
MinMax Weight combined heuristic				0.57 / 0.39	0.57 / 0.39	0.61 / 0.35
Heuristic					0.73 / 0.27	0.77 / 0.23
DeepQlearner **MinMax**						0.79 / 0.21
DeepQlearner **Heuristic**						

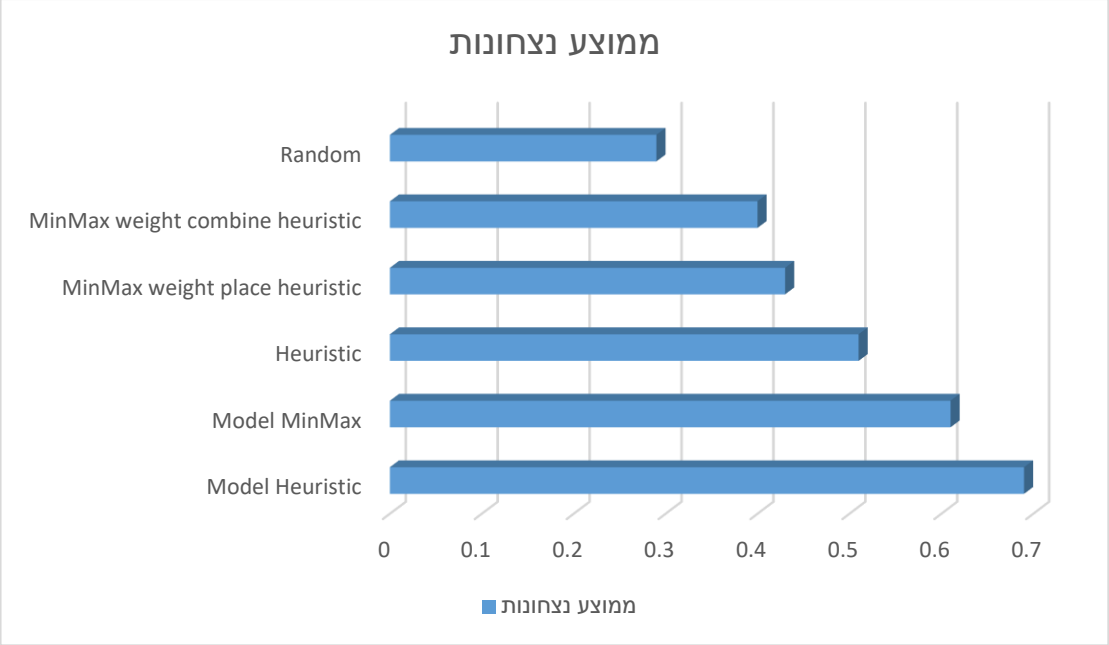
דוגמאות עבור ה-losses של המודלים שאימנו עבור לוח זה:











<u>ניתוח תוצאות עבור הלוח המקורי</u>

ניתן לראות שבמקום האחרון נמצא שחקן ה-Random. יצרנו את הסוכן הזה על מנת שנוכל להשוות את ביצועיו לכל שאר הסוכנים- שכן אין לו אסטרטגיה שלפיה הוא פועל- אלא הכל רנדומאלי. לכן, כפי שציפינו, ביצועי סוכן ה-Random היו הכי פחות טובים.

במקום החמישי נמצא סוכן ה- MinMax עם היוריסטיקה המשולבת. מכיוון שהיוריסטיקה המשולבת כוללת בתוכה כמה היוריסטיקות מתוחכמות, אכן ציפינו שסוכן זה ינצח את ה-Random.

במקום הרביעי נמצא סוכן ה-MinMax עם היוריסטיקת המשקל. כאשר ניסינו לחשוב מדוע היוריסטיקת המשקל מגיעה לביצועים טובים יותר מהיוריסטיקה המשולבת, הבנו שכנראה שילוב מספר היוריסטיקות מתוחכמות ונתינת המשקול כפי שבחרנו, פוגע בביצועי הסוכן. נציין כי הסוכנים של MinMax פעלו בעומק 2, ולכן יכול להיות שבהסתכלות על תורות קדימה (כפי ש-MinMax פועל), היוריסטיקה המשולבת פגעה בביצועים, למרות היותר יוריסטיקה טובה (כפי שנראה בהמשך).

אל המקום השלישי, הגיע סוכן ה-Heuristic עם היוריסטיקה המשולבת. נזכיר שסוכן זה, עובר על כלל המהלכים החוקיים, ובוחר את זה שהיוריסטיקה ניקדה כגבוה ביותר. כאן, לעומת במודל ה-MinMax היוריסטיקה המשולבת שיפרה את ביצועיו, כי מדובר בהסתכלות על התור הבא בלבד,

ולא בכמה צעדים קדימה. סך הכל אנו רואים ביצועים טובים יותר לסוכן ה-Heuristic מאשר לסוכן ה-MinMax, בעזרת היוריסטיקה המשולבת.

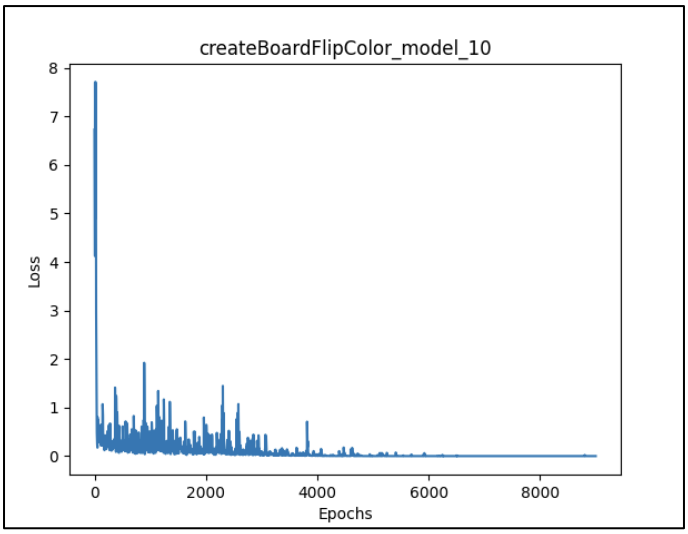
במקום השני, נמצא סוכן ה-DeepQlearner MinMax . ניזכר שסוכן זה נעזר ב- 6 מודלים (שאומנו לפי Deep Q learning) ומשקלל את התוצאה של כל אחד מהמודלים. נשים לב, שאומנם סוכן ה-MinMax הגיע לביצועים פחות טובים בהסתכלות על 2 צעדים קדימה, אך סוכן DeepQlearner MinMax מתחשב בכל המשחק: ניזכר שה-discount factor של ה- deep Q learner קובע כמה הסוכן יתחשב בעתיד הרחוק לעומת הקרוב. אנחנו בחרנו בערך 0.3 ולכן הסוכן אכן יתחשב בעתיד הרחוק, מעבר ל-2 צעדים קדימה. לכן, סביר שהסוכן DeepQlearner MinMax יהיה בעל ביצועים טובים יותר מאשר MinMax. בנוסף, סביר שסוכן זה ינצח גם את סוכן ה-Heuristic שכן, סוכן ה-Heuristic מתחשב רק בצעד הבא, ולעומת זאת DeepQlearner MinMax מקבל החלטה לפי 6 מודלים שאומנו והתחשבו בעתיד הרחוק יותר, ומושפע מבחירות היריב.

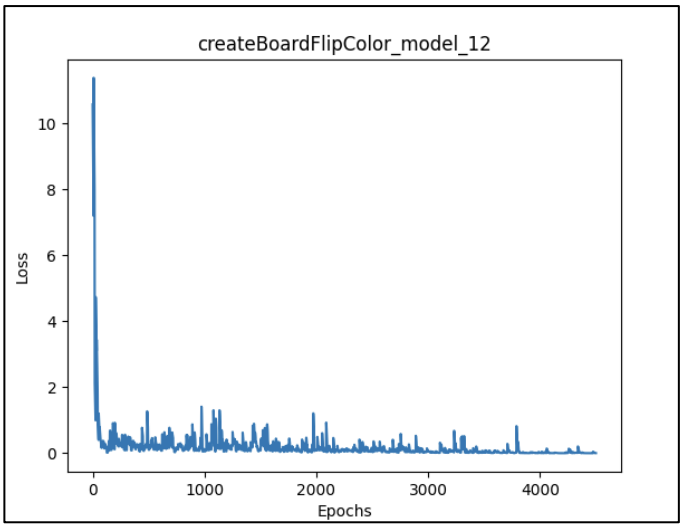
אל המקום הראשון הגיע סוכן ה-DeepQlearner Heuristic . הסיבות שבגללן הוא ניצח את 4 הסוכנים הראשונים זהות לסיבות שצוינו בפסקה מעלה. כפי שראינו סוכן ה-Heurictic ניצח את סוכני ה-MinMax, ולכן סביר היה שגן סוכן ה-DeepQlearner Heuristic ינצח את סוכן ה-DeepQlearner MinMax.

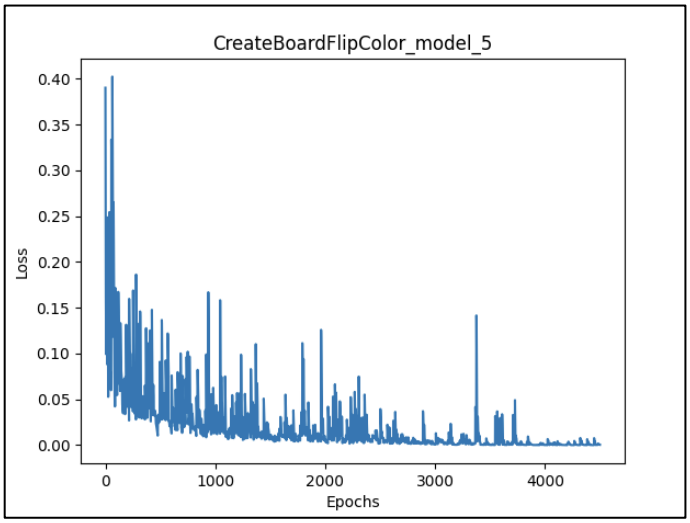
עבור הגרסה הראשונה, הממומשת במחלקה- CreateBoardFlipColor קיבלנו את התוצאות הבאות:

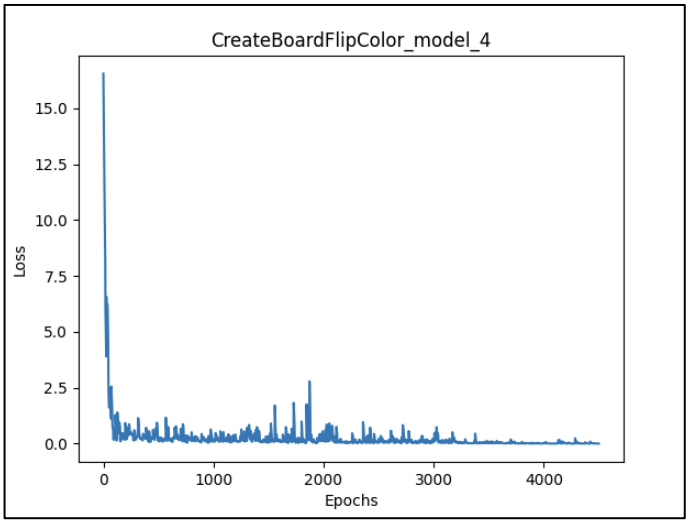
Player 1 / Player 0	MinMax Weight combined heuristic	MinMax Weight place heuristic	Random	Heuristic	DeepQlearner MinMax	DeepQlearner Heuristic
MinMax Weight combined heuristic		0.5 / 0.45	0.36 / 0.61	0.39 / 0.61	0.51 / 0.48	0.51 / 0.47
MinMax Weight place heuristic			0.31 / 0.61	0.45 / 0.51	0.59 / 0.4	0.42 / 0.56
Random				0.49 / 0.44	0.62 / 0.38	0.62 / 0.36
Heuristic					0.49 / 0.46	0.55 / 0.34
DeepQlearner MinMax						0.5 / 0.47
DeepQlearner Heuristic						

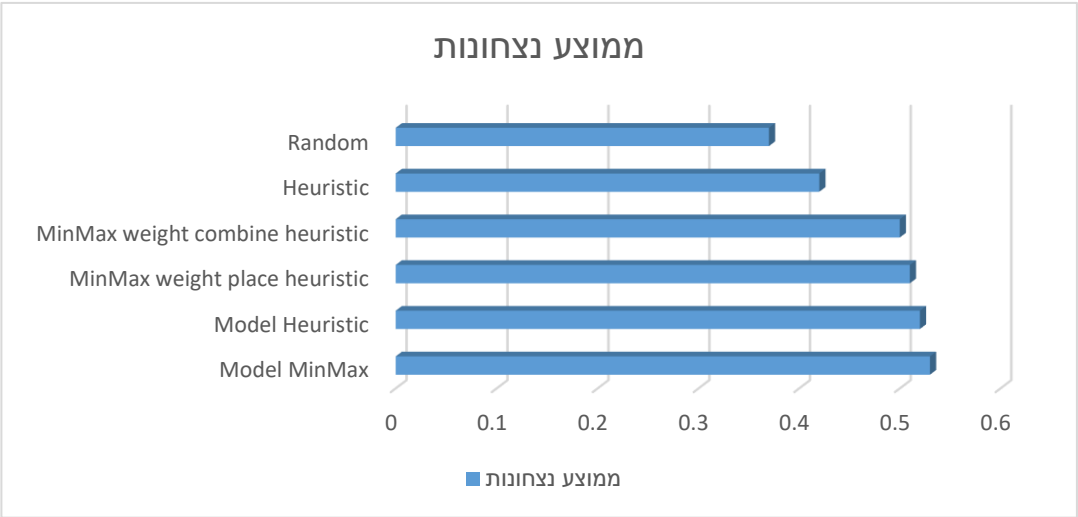
דוגמאות עבור ה-losses של המודלים שאימנו עבור לוח זה:











ניתוח תוצאות עבור הלוח FlipColor:

גם עבור הלוח הזה ניתן לראות שהסוכן הרנדומלי הגיע לביצועים הכי פחות טובים כפי שציפינו. שאר השינויים נובעים מהסיבות הבאות:

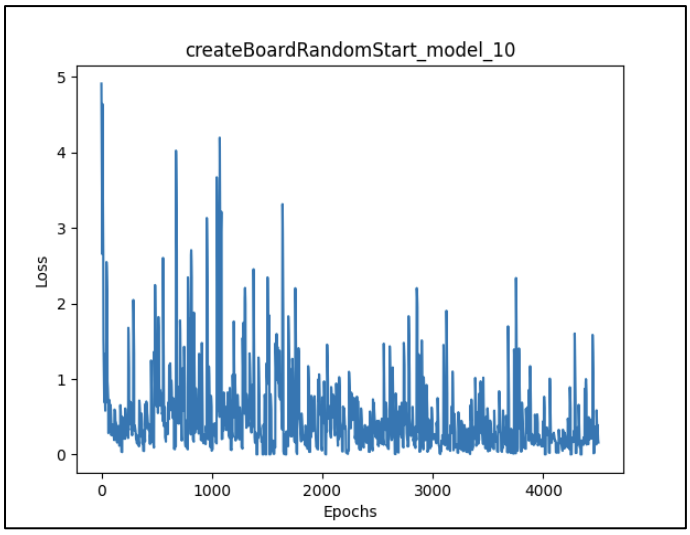
- יש מיקומים בלוח שגם אם מבחינת "אסטרטגית" כדאי לשחקן להניח את הדיסקית שלו בהם, הם עשויים לגרום לכך שצבע הדיסקית ישתנה- מה שיכול להפוך את הבחירה של השחקן לפחות יעילה עבורו. לדוגמא, אם הסוכן יתקל בפינה פנויה, סביר שהיוריסטיקות שלנו יתנו לה ניקוד גבוה, אך אם פינה זו נמצאת בנקודות שהוגרלו להחלפה, היתרון עובר ליריב במקום לשחקן הנוכחי. צורת פעולה זו, גורמת לכך שפני המשחק יכולים להשתנות לטובת היריב, למרות היתרונות של כל אחד מהסוכנים בפני עצמו.
- כלל ממוצעי הניצחונות בלוח הנוכחי קרובים יחסית ל-0.5 ולכן נסיק כי הסוכנים לא מצליחים ללמוד ותוצאתם דומה לתוצאות שהיינו מצפים לקבל במשחק בין שני סוכנים רנדומאליים.
- בלוח זה המשחק עשוי להתחיל עם יתרון עבור שחקן מסוים. למשל, אם בנקודות שהוגרלו עבור השחקן השחור, נמצאות פינות/ מיקום על המסגרת, השחקן הלבן יקבל יתרון, שכן כאשר השחור יבחר להניח במיקומים אלו, צבע הדיסקיות ישתנה ללבן. הדבר עלול לשבש את התאוריות שהתבססנו עליהן.
- מבחינת אימון המודלים של deep Q learning: יתכן שהגדלת מספר המשחקים שאותם המודל לומד תתרום לעליית הנצחונות. זאת מכיוון שצעדים טובים במשחקים רבים, עשויים להזיק עבור משחק ספיציפי (שבו הוגרלו המיקומים שבמשחקים הקודמים המודל למד כמיקומים טובים). ניתן לראות לראות דוגמא כזו בגרף CreateBoeardFlipColor_model_5 שנמצא מעלה: לאחר שיש ירידה משמעותית בלוס, לקראת הסוף ישנה קפיצה חדה, שזה יכול להיות ההסבר לעלייה זו.

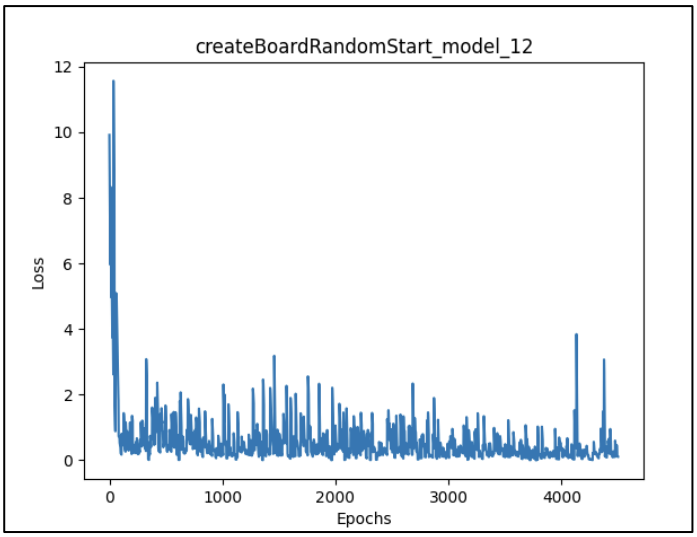
עבור הגרסה השנייה, הממומשת במחלקה- CreateBoardRandomStart קיבלנו את התוצאות

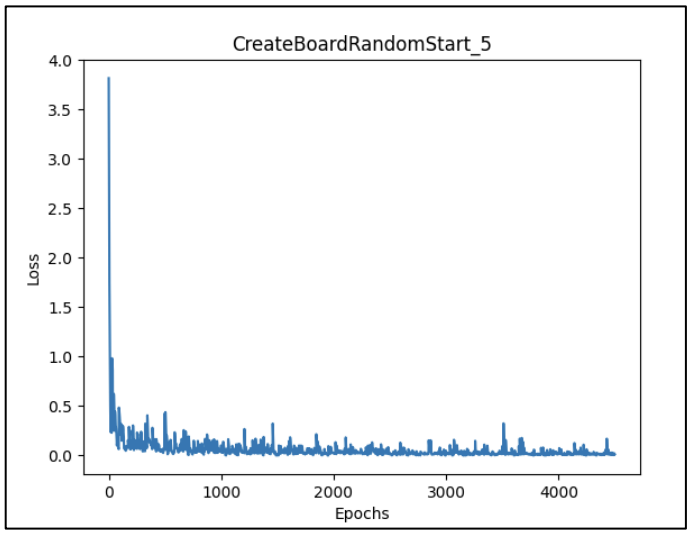
הבאות:

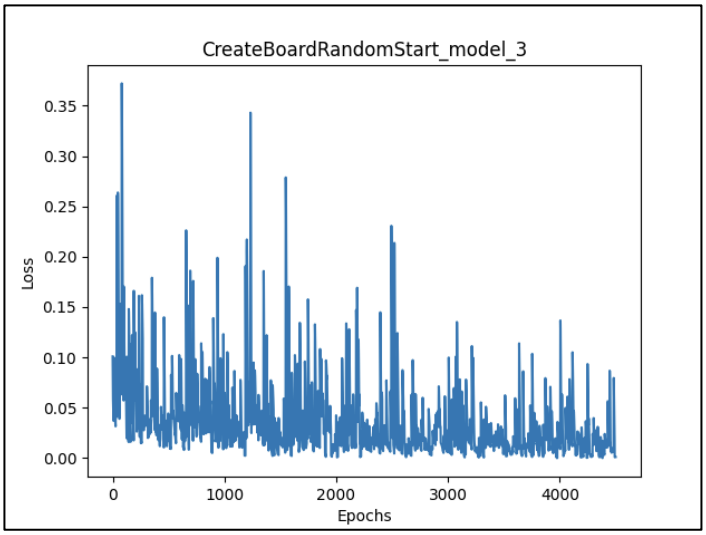
Player 1 / Player 0	MinMax Weight combined heuristic	MinMax Weight place heuristic	Random	Heuristic	DeepQlearner MinMax	DeepQlearner Heuristic
MinMax Weight combined heuristic		0.57 / 0.39	0.42 / 0.65	0.64 / 0.33	0.56 / 0.42	0.58 / 0.41
MinMax Weight place heuristic			0.48 / 0.51	0.48 / 0.51	0.51 / 0.48	0.47 / 0.45
Random				0.61 / 0.36	0.57 / 0.39	0.55 / 0.37
Heuristic					0.53 / 0.46	0.51 / 0.46
DeepQlearner MinMax						0.62 / 0.38
DeepQlearner Heuristic						

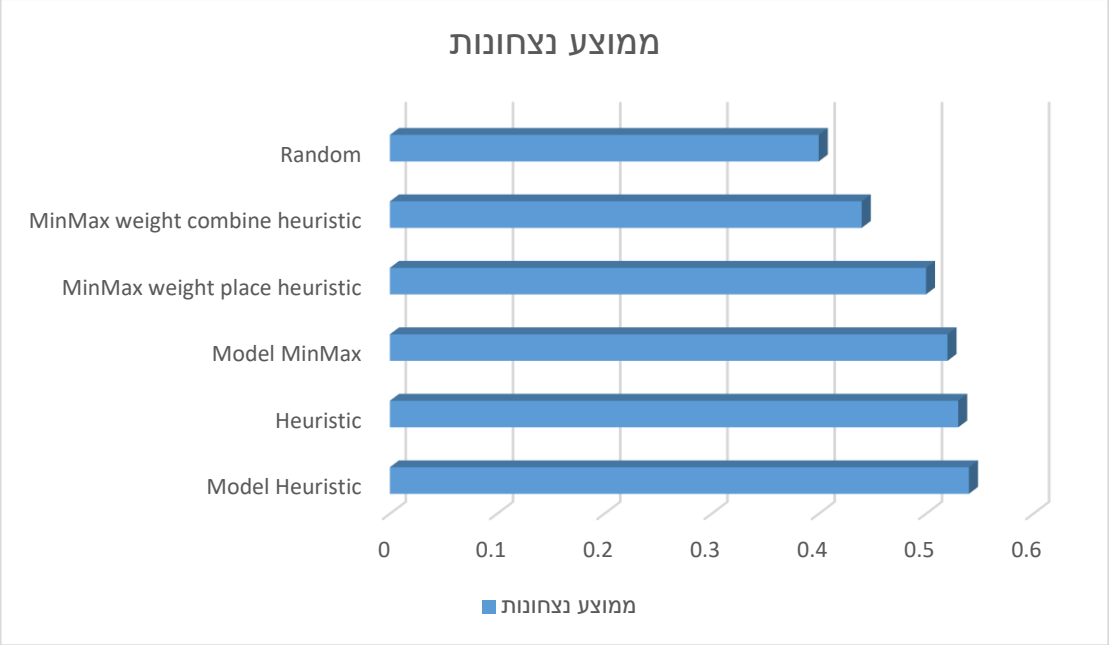
דוגמאות עבור ה-losses של המודלים שאימנו עבור לוח זה:











<u>ניתוח התוצאות עבור RandomStart:</u>

גם עבור הלוח הזה ניתן לראות שהסוכן הרנדומלי הגיע לביצועים הכי פחות טובים כפי שציפינו. שאר השינויים נובעים מהסיבות הבאות:

- עבור לוח זה יש מודלים שלא התכנסו. הסיבה שזה יכול לקרות היא שההתחלה הרנדומאלית דורשת יותר משחקי אימון על מנת ללמוד היטב את המשחק. כמו כן, מתווספים יותר מצבים למשחק מאשר במשחק המקורי. לכן, המודלים של deep q learner למדו פחות טוב את המשחק והגיעו לביצועים פחות טובים – שחקן הheuristic ניצח את שחקן DeepQlearner minmax.
- ניתן לראות שממוצע הניצחונות של הסוכנים: DeepQlearner minmax, DeepQlearner heuristic ירדו לעומת חוקי המשחק המקוריים, ולכן ממוצע הניצחונות של הסוכנים האחרים עלה על חשבונם.
- כלל ממוצעי הניצחונות בלוח הנוכחי קרובים יחסית ל-0.5 ולכן נסיק כי הסוכנים לא מצליחים ללמוד ותוצאתם דומה לתוצאות שהיינו מצפים לקבל במשחק בין שני סוכנים רנדומאליים.
- בלוח זה המשחק עשוי להתחיל עם יתרון עבור שחקן מסוים. למשל, השחקן הראשון יכול בתורו הראשון לאחר תחילת המשחק לתפוס פינה – הדבר עלול לשבש את התאוריות שהתבססנו עליהן.

שיפורים והרחבות לעתיד

- ניתן לשפר את הפרמטרים שהרשת קיבלה, ואת הפרמטרים ש-Deep Q learner קיבל. אנחנו בחרנו על פי החומר הנלמד, הפעלת שיקול דעת, ניסוי וטעיה. ניתן לבצע תהליך של למידת הפרמטרים האופטימאליים על מנת להגיע לביצועים טובים יותר.
- ניתן לחקר לעומק אילו היוריסטיקות מטיבות אחת עם השנייה ואילו פוגעות אחת בשנייה. כך נוכל להגיע לשילוב היוריסטיקות הטוב ביותר, שישפר את הביצועים. יכולים להיות מצבים במשחק שבהם עדיף יהיה לתת משקל גבוה יותר להיוריסטיקה פחות מתוחכמת כמו ה-Base, לעומת המשקל הנמוך שאנו נותנים לה. כדי להסיק את המסקנות הללו, נדרש ניתוח מעמיק נוסף על מנת להבין באילו מקרים זה עלול לקרות.
- ניתן לתת עומק גדול יותר לסוכן ה-MinMax ובכך כנראה לשפר את ביצועיו (על סמך הניסיון בפרויקטים קודמים שלנו בקורס).
- הוספת היוריסטיקות שמתחשבות בשינויי החוקים שיצרנו עבור המשחק: למשל, אם מדובר בלוח של CreateBoardFlipColor, לתת ניקוד שלילי למיקום שהוגרל להתהפך לצבע של היריב.

סיכום

במהלך בניית הפרויקט, גילינו כמה סוכנים שמאומנים לפי Deep Q learning עשויים להיות חזקים מול סוכנים אחרים. הבנו שפתירת הבעיה היא קשה- גם לאחר אימון מסיבי וארוך, השחקן הרנדומאלי הצליח לנצח מדי פעם את הסוכנים המורכבים יותר. ראינו כי כל שינוי קטן בחוקי המשחק, דורש שינוי מעמיק באסטרטגיית המשחק, בהיוריסטיקות, ובמספר האימונים, על מנת להגיע לביצועים הטובים ביותר במשחק. בנוסף, ראינו כמה השפעה יש לכל היוריסטיקה על תוצאות המשחק, וכמה בחירת היוריסטיקה היא קריטית.

נציין שאימנו 36 מודלים של רשתות נוירונים. תהליך האימון הוא ארוך מאוד (המחשבים עבדו לעיתים כמה ימים על מנת לסיים הרצה), ולכן ביצענו את המקסימום שהכלים שברשותינו אפשרו לנו. כמובן, שאם נרצה לשפר יותר את התוצאות, נוכל להיעזר בכלים נוספים שיאפשרו לנו הרצות גדולות יותר שנמשכות זמן ארוך יותר.

הצגנו את התוצאות באופן כזה שיאפשר להשוות את ביצועי הסוכנים השונים, ניתחנו אותן, והעלינו הצעות לשיפור המודלים השונים.

מצורף קובץ Readme שמכיל הוראות הרצה.

References

1. "An Analysis of Othello AI Strategies" - Alec Barber, Warren Pretorius, Uzair Qureshi, Dhruv Kumar.
2. Computer othello – Wikipedia.
3. An intelligent Othello player combining machine learning and game specific heuristics - Kevin Anthony Cherry.