האוניברסיטה הפתוחה

החטיבה למדעי המחשב

תואר שני

פרויקט מתקדם [22997]

מימוש רשת Q עמוקה עבור משחק הוידאו פק-מן

חנן אהרונוף 021893730

בהנחיית ד״ר מיריי אביגל

מרץ 2019

תוכן עניינים

רשימת איורים 1

תקציר 2

1. מבוא 3

2. רשתות Q עמוקות 4

2.1 עיבוד מקדים 6

2.2 מבנה הרשת 6

2.3 אלגוריתם 6

3. יישומים אלגוריתם DQN עבור משחק הוידאו פק-מן 8

3.1 פק-מן 8

3.2 פק-מן של אוניברסיטת ברקלי 8

3.2.1 תצורת לוח המשחק 8

3.2.2 ייצוג מצב המשחק 9

3.2.3 רכיבי גרפיקה 9

3.2.4 סוכנים 9

3.2.4.1 רישום מצב ראשוני 10

3.2.4.2 תצפית על מצב 10

3.2.4.3 ביצוע פעולה 10

3.2.4.4 סיום 10

3.2.5 ניהול משחקים 10

3.3 יישום DQN בפק-מן של ברקלי 10

3.3.1 זיכרון שימוש חוזר 10

3.3.2 פרמטרים 11

3.3.3 רשת Q-עמוקה 11

3.3.4 ממיר פריימים 11

3.3.5 מחסנית פריימים 11

3.3.6 יומן תיעוד אירועים 12

3.3.7 סטטיסטיקות 12

3.3.8 סוכן DQN 12

4. תוצאות יישום אלגוריתם DQN עבור משחק הוידאו פק-מן 13

4.1 מדדי ביצועים 13

4.2 שלב המשחק smallGrid 13

4.3 שלב המשחק mediumGrid 14

4.4 שלב המשחק minimaxClassic 16

4.5 שלב המשחק smallClassic 17
סיכום 20
נספח א: רשימת פרמטרים לאלגוריתם DQN 21
נספח ב: מפרט מחשב עליו התבצעו ניסויי הפרויקט 23
רשימת מקורות 24

רשימת איורים

תיאור	עמוד
איור 2.1 - אילוסטרציה של רשת נוירונים קונבולוציוניות שבבסיס DQN	5
איור 2.2 - אלגוריתם DQN בשימוש חוזר בזיכרון	7
איור 3.1א - קובץ תצורה טקסטואלי עבור שלב smallClassic	9
איור 3.2ב - תצורה גרפית של קובץ תצורה עבור שלב smallClassic	9
איור 3.2 - ייצוג גרפי חלופי לזה המיוצר על ידי חבילת הקוד tkinter לאיור 3.1ב	11
איור 3.3 - דוגמא לפלט יומן אירועים במהלך אימון מודל DQN	12
איור 4.1 - תיאור גרפי של שלב המשחק smallGrid	13
איור 4.2 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק smallGrid עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו	14
איור 4.3 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק smallGrid לבין מספר המשחקים ששוחקו	14
איור 4.4 - תיאור גרפי של שלב המשחק mediumGrid	15
איור 4.5 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק mediumGrid עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו	15
איור 4.6 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק mediumGrid לבין מספר המשחקים ששוחקו	16
איור 4.7 - תיאור גרפי של שלב המשחק minimaxClassic	16
איור 4.8 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק minimaxClassic עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו	17
איור 4.9 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק minimaxClassic לבין מספר המשחקים ששוחקו	17
איור 4.10 - תיאור גרפי של שלב המשחק smallClassic	18
איור 4.11 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק smallClassic עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו	18
איור 4.12 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק smallClassic לבין מספר המשחקים ששוחקו	19

תקציר

למידה באמצעות חיזוקים (Reinforcement Learning) היא תת-תחום של למידה חישובית העוסק בהתקשרות של סוכן עם הסביבה ולמידה של מדיניות אופטימלית (Optimal Policy) על ידי ניסוי וטעייה. השילוב של למידה באמצעות חיזוקים עם למידה עמוקה אינו רעיון חדש, אך עם התקדמות המחקר של למידה עמוקה בשנים האחרונות, אנו עדים להתעוררות מחודשת בנושא עם מספר מאמרים מפורסמים אשר גדל בהתמדה משנה לשנה כבר שניים וחצי עשורים ברציפות. הופעתם של אלגוריתמים פורצי דרך כדוגמת DQN ו- AlphaGo היו להשראה למאות הרחבות ויישומים עבור בעיות ממגוון רחב של תחומי החיים.
החוקרים שיצרו את אלגוריתם DQN הראו כבר שניתן לייצר סוכן למידה באמצעות חיזוקים אשר מסוגל לשחק ברמת אדם עשרות משחקי אטארי וזאת בהתבסס אך ורק על פלט מסך הוידאו. בעבודה זו אני מראה כי ניתן ליישם את אלגוריתם DQN בהצלחה עבור משחק הוידאו פק-מן.

1. מבוא

למידה באמצעות חיזוקים (Reinforcement Learning) היא תת-תחום של למידה חישובית העוסק בהתקשרות של סוכן עם הסביבה ולמידה של מדיניות אופטימלית (Optimal Policy) על ידי ניסוי וטעייה, עבור בעיות החלטה סדרתיות (Sequential Decision Problems) מתוך קשת רחבה של תחומי חיים [1].

למידה עמוקה (Deep Learning), בניגוד ללמידה רדודה, מאופיינת ברשת בעלת מספר שכבות. הרשת מוזנת בצידה האחד במידע גולמי, אשר עובר שינוי כלשהו בכל שכבה, עד הגיעו אל השכבה האחרונה. השינוי שהמידע עובר בכל שכבה מיוצג על ידי אוסף של פרמטרים אשר משתנים בתהליך הלימוד, עד אשר מגיעים לערכי התכנסות. עדכון הפרמטרים מתבצע באמצעות שימוש באלגוריתם התפשטות אחורי (Backpropagation) [2]. למידה עמוקה הביאה לפריצות דרך בתחומים רבים כדוגמת עיבוד תמונה ווידאו, ניתוח קול ושפה, משחקי מחשב, נהיגה אוטונומית וכד' [3].

השילוב של למידה באמצעות חיזוקים עם למידה עמוקה אינו רעיון חדש [4], אך עם התקדמות המחקר של למידה עמוקה בשנים האחרונות, אנו עדים להתעוררות מחודשת בנושא [5] עם מספר מאמרים מפורסמים אשר גדל בהתמדה משנה לשנה כבר שניים וחצי עשורים ברציפות [6] . הופעתם של אלגוריתמים פורצי דרך כדוגמת DQN [7], AlphaGo ו- AlphaZero [8], Dueling Networks [9], Determinist Policy Gradient [10] ו- A3C [11] היו להשראה למאות הרחבות ויישומים עבור בעיות ממגוון רחב של תחומי החיים.

למידה באמצעות חיזוקים דורגה על ידי המכון הטכנולוגי של מסצ'וסטס (MIT) כאחת מעשר הטכנולוגיות פורצות הדרך של שנת 2017 [12] ולמידה עמוקה דורגה באותה רשימה בשנת 2013 [13]. חוקרים רבים בתחום מחשיבים את השילוב של למידה עמוקה ולמידה באמצעות חיזוקים כצעד הראשון בדרך ליצירתה של מסגרת תאורטית של בינה מלאכותית כללית, כלומר, כזו שיכולה לבצע כל משימה שאדם יכול לבצע [14]. סילבר, התורם העיקרי של AlphaGo [8] קבע את הנוסחה:

בינה מלאכותית כללית = למידה באמצעות חיזוקים + למידה עמוקה [15]

מטרתה של עבודה זו היא ליישם את אלגוריתם DQN כפי שהוא תואר ב- [7] עבור משחק הוידאו פק-מן. עבודה זו היא המשך ישיר לעבודה המסכמת ומהווה את החלק המעשי שלה. מומלץ לקרוא את פרק 2 של העבודה המסכמת העוסק ברקע התיאורטי של למידה עמוקה באמצעות חיזוקים בטרם קריאת עבודה זו.

העבודה מחולקת לארבעה חלקים באופן הבא; החלק הראשון מסביר את אלגוריתם DQN כפי שהוא מתואר ב- [7]. החלק השני מתאר פלטפורמת פק-מן של אוניברסיטת ברקלי אשר מהווה את התשתית לעבודה זו. הפרק השלישי מתמקד בתוצאות אימון אלגוריתם DQN עבור מספר שלבי משחק של פק-מן. ולבסוף החלק הרביעי עוסק במסקנות ובקשיים שעלו במהלך הפרוייקט.

2. רשתות Q עמוקות

בפברואר 2015 פרסמה קבוצת חוקרים מאמר [7] בשם Human-level control through deep reinforcement learning. במחקרם, החוקרים רצו ליצור אלגוריתם בודד אשר יוכל לפתח מגוון רחב של יכולות עבור מגוון משימות מאתגרות כחלק מהניסיון ליצור מסגרת תאורטית של בינה מלאכותית לפתרון בעיות כלליות (General Artificial Intelligence). בכדי להשיג מטרה זו החוקרים פיתחו אלגוריתם מתקדם בשם רשתות Q עמוקות (Deep Q Networks) אשר משלב בתוכו למידה באמצעות חיזוקים יחד עם רשתות נוירונים עמוקות. החוקרים השתמשו ברשתות קונבולוציוניות עמוקות (Deep Convolutional Networks) שהן בעלות מבנה היררכי של מסנני קונבולוציה (Convolutional Filters) אשר ביכולתן לתפוס קשרים מרחביים מקומיים של תמונה עם עמידות לשינויים בגודל ונקודת מבט. כפי שהגדרנו בפרק הקודם מטרת למידה באמצעות חיזוקים היא לייצר סוכן (Agent) אשר בוחר פעולות (Actions) במצבים (States) אשר ממקסמות את התגמול המצטבר העתידי (Cumulative Future Reward) והגדרנו פונקציית פעולה-רווח (לעיתים מכונה פונקציית Q) באופן הבא:

$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V^*(s)]$$

כאשר $T(s,a,s')$ היא פונקציית מעברים סטוכסטית, כך ש $s,s' \in S, a \in A$, $R(s,a,s')$ היא פונקציית תגמולים, כך ש $s,s' \in S, a \in A$, $V^*(s)$ היא תוחלת התגמול המצטבר כאשר הסוכן מתחיל ממצב s ופועל באופן אופטימלי לאחר מכן. ו- $Q^*(s,a)$ היא תוחלת התגמול המצטבר כאשר הסוכן מתחיל מביצוע פעולה a במצב s ופועל באופן אופטימלי לאחר מכן.

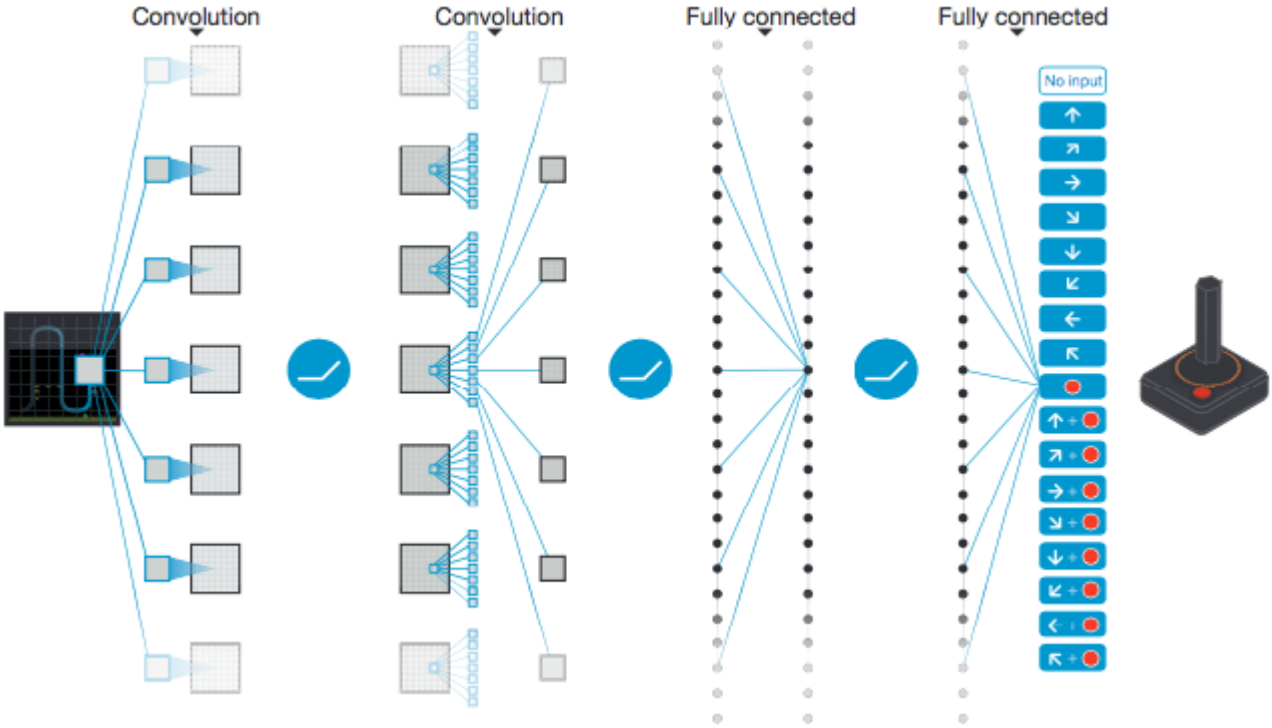
החוקרים אם כן שאפו, לקרב את פונקציית פעולה-רווח באמצעות רשתות קונבולוציוניות עמוקות, אלא שלמידה באמצעות חיזוקים ידועה כבלתי יציבה כאשר משתמשים בפונקציית קירוב לא לינארית כמו רשתות נוירונים עמוקות בכדי לקרב את פונקציית Q. חוסר היציבות נובע ממספר גורמים, קרי, המתאם בין דגימות פעולה-מצב-תגמול, העובדה ששינויים קטנים לערכי Q עלולים להשפיע בצורה משמעותית על המדיניות וכן בגלל המתאם בין ערכי Q לבין ערכי המטרה $R(s,a,s') + \gamma V^*(s)$. אלגוריתם DQN המוצע על ידי החוקרים מתמודד עם בעיית חוסר היציבות על ידי יישום מנגנון שידור חוזר של ניסיון קודם (Replay Memory). על פי העיקרון של מנגנון זה, ניסיונות קודמים (שלשות של מצב-פעולה-תגמול) של הסוכן נשמרים במבנה של זיכרון. אימון הרשת הוא לאו דווקא באמצעות הניסיונות האחרונים שנלמדו, אלא באמצעות ניסיונות אקראיים (בפיזור אחיד) שנלמדו בעבר. על ידי שימוש בשיטה זו, אנו מבטלים את המתאם בין דגימות פעולה-מצב-תגמול. בנוסף החוקרים מיישמים טכניקה של רשת מטרה בכדי להתמודד עם חוסר היציבות שתואר קודם. על פי טכניקה זו, בזמן הלימוד ישנן שתי רשתות, עם אוסף שונה של משקולות, קרי רשת-Q ורשת מטרה. משקולותיה של רשת-Q מתעדכנים בכל איטרציה, בעוד שהמשקולות של רשת המטרה מתעדכנים בכל מספר קבוע של איטרציות בהתאם למשקולות רשת-Q. רשת המטרה משמשת בכדי לחשב את ערכי המטרה $R(s,a,s') + \gamma V^*(s)$. טכניקה זו מסייעת בהתמודדות עם המתאם בין ערכי Q לבין ערכי המטרה.

החוקרים הגדירו פונקציית קירוב לערכי Q עם פרמטרים כ- $Q(s,a;\theta_i)$ אשר מחושבת על ידי רשת נוירונים קונבולוציוניות כמתואר באיור 2.1. θ_i הם הפרמטרים (או משקולות) של הרשת באיטרציה i. בכדי להפעיל את מנגנון שימוש בניסיון חוזר, החוקרים שומרים בכל שלב את הניסיון שנצבר באותו שלב על ידי הסוכן

המוגדר כ- $e_i = (s_i, a_i, r_i, s_{i+1})$ בזיכרון D, כאשר s_i הוא המצב בוא נמצא הסוכן, a_i הוא הפעולה בה הוא נקט, r_i הוא התגמול שהתקבל בעד ביצוע הפעולה ו- s_{i+1} הוא המצב אליו הגיע בעקבות ביצוע הפעולה. בזמן הלמידה של הרשת, החוקרים מבצעים עדכונים באמצעות כלל העדכון Q-Learning (מהפרק הקודם), על דוגמאות (או ניסיונות קודמים) אשר נשלפים באופן אקראי (עם פיזור אחיד) מעל D. נסמן שליפה שכזו באמצעות $(s, a, r, s') \sim U(D)$. להלן נוסחת ההפסד (loss) של הרשת המבוססת על כלל העדכון Q-Learning:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

כאשר γ הוא מקדם ההפחתה (Discount Factor), θ_i הן המשקולות של הרשת באיטרציה i, θ_i^- הן המשקולות של הרשת באיטרציה i אשר באמצעותן חישבו את ערך המטרה. ערכי θ_i^- מתעדכנים עם ערכי θ_i, אחרי כל C איטרציות, ונותרים קבועים בין כל שני עדכונים.



איור 2.1 – אילוסטרציה של רשת נוירונים קונבולוציוניות שבבסיס DQN. הקלט לרשת הוא ארבע תמונות בגודל 84 על 84 פיקסלים אשר מעובדות בשלושה שכבות קונבולוציה ולאחר מכן בשתי שכבות מקושרות באופן מלא לכל פעולה אפשרית אחת של הסוכן [7].

בכדי לבדוק את סוכן ה- DQN שהחוקרים ייצרו, הם השתמשו בפלטפורמת המשחקים הקלאסית אטארי 2600 (Atari 2600 Game Consul), אשר מציעה מגוון רחב של משחקים (או משימות) אשר מתוכננים להיות מאתגרים עבור בני אדם. החוקרים השתמשו באותו מבנה רשת, אותם היפר-פרמטרים וטכניקת לימוד לכל המשחקים, כאשר הקלט הוא תמונות הוידאו של המשחק בגודל 210 על 160 פיקסלים בתדר 60 הרץ, בכדי להראות ש- DQN יכול ללמוד מדיניות מוצלחת למגוון של משחקים ללא ידע מוקדם ובהתבסס רק על מסך הוידאו ולוח המקשים. ואכן, החוקרים הצליחו להראות שבאמצעות DQN ניתן לאמן רשתות נוירונים עמוקות גדולות באמצעות למידה באמצעות חיזוקים ולייצר סוכן אשר מסוגל לנצח את משחקי אטארי

ברמה שעולה על זו של בן אדם. החוקרים השוו את הביצועים של DQN עבור 49 משחקי אטארי לביצועיהם של אלגוריתמי למידה באמצעות חיזוקים הטובים ביותר. בנוסף, החוקרים דגמו את ביצועיהם של שחקנים אנושיים עבור אותם משחקים. התוצאה הייתה חד משמעית- אלגוריתם DQN התעלה מעל לביצועיהם של אלגוריתמי למידה באמצעות חיזוקים ב- 43 משחקים וכל זאת ללא ידע מוקדם על המשחק עצמו! כמו כן האלגוריתם השיג רמת משחק הזהה או מתעלה מעל לזו של שחקנים אנושיים בכל אחד מן 49 המשחקים.

2.1. עיבוד מקדים

פלטפורמת המשחקים אטארי 2600 מייצרת תמונת וידאו בגודל 210 על 160 פיקסלים עם פלטת צבעים בגודל של 128 צבעים. עבודה ישירה מעל קלט בגודל זה (128*160*210) דורשת משאבי מעבד וזיכרון רבים. על כן החוקרים החליטו להפעיל פרוצדורה של עיבוד מקדים, φ, מעל תמונת הוידאו של אטארי בכדי להקטין אותה וזאת מבלי לפגוע באלמנטים שנמצאים בה וזאת על ידי המרת התמונה לתמונת גווני אפור וכיווצה לגודל של 84 על 84 פיקסלים. במהלך האימון, DQN מפעיל את φ על ארבעת הפריימים האחרונים של המשחק, ומאגד אותם לכדי יחידה אחת אשר משמשת כקלט לרשת הנוירונים.

2.2. מבנה הרשת

הקלט לרשת הנוירונים העמוקה הוא מקבץ של ארבע תמונות כל אחת בגודל 84 על 84 פיקסלים אשר מיוצר על ידי הפעלת פונקציית העיבוד המקדים φ. השכבה החבויה הראשונה מפעילה 32 פילטרים מסוג קונבולוציה עם גרעין בגודל 8 על 8 ואורך פסיעה (Stride) 4 ולאחריהם רכיב לא-לינארי (RELU). השכבה החבויה השנייה מפעילה 64 פילטרים מסוג קונבולוציה עם גרעין בגודל 4 על 4 ואורך פסיעה (Stride) 2 ולאחריהם רכיב לא-לינארי (RELU). השכבה החבויה השלישית מפעילה 64 פילטרים מסוג קונבולוציה עם גרעין בגודל 3 על 3 ואורך פסיעה (Stride) 1 ולאחריהם רכיב לא-לינארי (RELU). השכבה הרביעית היא שכבה מקושרת באופן מלא עם 512 רכיבי יישור. השכבה האחרונה היא שכבה לינארית מקושרת באופן מלא עם מוצא בודד לכל פעולה חוקית של הסוכן.

2.3. אלגוריתם

במסגרת המחקר החוקרים בחנו את סוכן DQN באמצעות התקשרות עם מדמה (Emulator) של פלטפורמת משחקי אטארי 2600. בכל יחידת זמן הסוכן בוחר פעולה a_t מתוך אוסף הפעולות החוקיות של הסוכן, קרי, $\mathcal{A} = \{1, \dots, K\}$. הפעולה מועברת למדמה אשר משנה את מצבו הפנימי ואת תוצאת המשחק. המצב הפנימי של המדמה אינו נצפה על ידי הסוכן, אלא, במקום זאת, הסוכן צופה בתמונת הוידאו המיוצרת על ידי המדמה (ייצוג באמצעות וקטור). בנוסף, הסוכן מקבל תגמול בכל רגע של המשחק, המבטא את השינוי בתוצאת המשחק. היות והסוכן צופה רק במסך הנוכחי המדמה מפיק, הרי שהסביבה היא נצפית-חלקית (Partially Observed), כלומר לא ניתן להבין סיטואציה רק מן המסך הנוכחי. על כן, הקלט לסוכן הוא רצף של m מסכים אחרונים. איור 2.2 מתאר את אלגוריתם DQN עם שימוש חוזר בזיכרון.

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

איור 2.2 – אלגוריתם DQN בשימוש חוזר בזיכרון [7].

פרק ג: יישום אלגוריתם DQN עבור משחק הוידאו פק-מן

במסגרת עבודה זו בחרתי ליישם את אלגוריתם DQN עבור משחק הוידאו פק-מן (Pac-Man). בפרק זה נדון באופן בו יישמתי את האלגוריתם תוך כדי הסבר מפורט של מבנה הקוד וכיצד ניתן להפעילו.

3.1 פק-מן

משחק ארקדיה (Arcade) שפותח על ידי חברת Namco מיפן בשנת 1980. במסגרת המשחק, על השחקן להוביל את פאק-מן דרך מבוך המכיל נקודות שונות (Pac-Dots) וכן ארבע רוחות צבעוניות הנקראות בלינקי (Blinky), פינקי (Pinky), אינקי (Inky) וקלייד (Clyde). מטרת המשחק היא לצבור נקודות על ידי אכילת הנקודות השונות הפזורות במשחק. ארבעת הרוחות נעות בתוך המבוך במטרה לצוד את פק-מן. אם אחת מן הרוחות נוגעת בו אז המשחק מופסד. בפינות המבוך ישנן ארבע נקודות גדולות מהבהבות. אכילת נקודה מסוג זה הופכת את ארבעת הרוחות לרוחות רפאים שנעות לאט יותר (לפרק זמן קצר) אשר ניתן לאכול ולהרוויח נקודות בונוס. כאשר רוח נאכלת על ידי פק-מן היא חוזרת שוב למשחק.

3.2 פק-מן של אוניברסיטת ברקלי

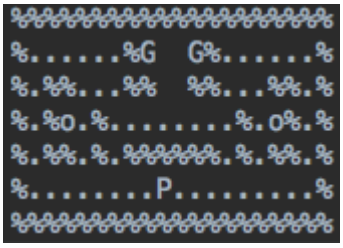
המאפיינים של משחק הפק-מן כפי שתוארו בסעיף הקודם הופכים אותו למועמד טבעי ליישום אלגוריתמים של למידה באמצעות חיזוקים. ואכן במסגרת קורס מבוא לבינה מלאכותית של אוניברסיטת ברקלי בקליפורניה [16] פותחה פלטפורמת משחק פק-מן המיועדת ללמידה והתנסות עם אלגוריתמים של בינה מלאכותית באופן כזה שניתן ליישם בה יחסית בקלות אלגוריתמים שונים. כמו כן, יוצרי הפלטפורמה מאשרים לאנשים מחוץ לאוניברסיטה להשתמש בפלטפורמה למטרות פרטיות או חינוכיות [17].
הפלטפורמה כתובה בשפת התכנות פייתון (Python) ולה מספר רכיבים:

3.2.1 תצורות לוח המשחק

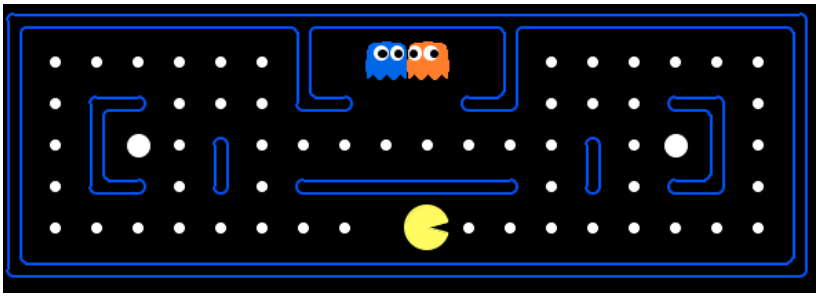
פלטפורמת המשחק מאפשרת הרצה של האלגוריתמים על גבי מספר תצורות של לוח המשחק. תצורות המשחק מוגדרות על ידי קבצי טקסט (אחד לכל תצורה) המצויים בתיקיית layouts והם בעלי הסיומת lay.
הגדרת התצורה נעשית באמצעות שימוש בסימני טקסט מוסכמים:

- % - מייצג קיר
- . - מייצג נקודה אכילה על ידי פק-מן (Pac-Dots)
- G - מייצג רוח
- P - מייצג את פק-מן
- o - מייצג נקודה אכילה גדולה על ידי פק-מן

בכדי להריץ את הפלטפורמה יש לספק את שם הקובץ של תצורת השלב הרצוי. הפלטפורמה קוראת קובץ השלב וממירה אותו לתצורה גרפית מלאה. איור 3.1א מציג תוכן של קובץ פלטפורמה (smallClassic) איור 3.1ב מציג את התצורה הגרפית של התצורה המתוארת באיור 3.1א.



איור 3.1א – קובץ תצורה טקסטואלי עבור שלב smallClassic. התצורה נעשית באמצעות שימוש בסימנים מוסכמים.



איור 3.1ב – תצורה גרפית של קובץ תצורה עבור שלב smallClassic. התצורה נעשית על ידי המרה של הסימנים המוסכמים בקובץ הטקסט לייצוג גרפי מתאים

3.2.2 <u>ייצוג מצב משחק</u>

אחד הרכיבים המרכזיים בפלטפורמה הוא למעשה רכיב ייצוג של מצב המשחק (Game State) בכל רגע. רכיב זה למעשה מכיל בתוכו בכל רגע נתון של המשחק את המצב המשחק ובפרט את פירוט האוכל שעדיין לא נאכל, מיקומו של פק-מן, מיקומם של הרוחות, ניקוד נוכחי וכד'. באמצעות רכיב זה ניתן לספק לאלגוריתמים השונים שמפותחים מעל הפלטפורמה את כל המידע שהם צריכים בכדי לפעול.

3.2.3 <u>רכיבי גרפיקה</u>

אחת המטרות המוצהרות של יוצרי הפלטפורמה הייתה לתת תחושה ויזואלית גבוהה כאשר מתנסים עם אלגוריתם כלשהו מעל הפלטפורמה. בכדי להעניק תחושה זו פותחו רכיבי הגרפיקה אשר אחראים על ייצוג גרפי איכות גבוהה של מהלך המשחק. יש לציין שכאשר מאמנים מודל, אין חובה, ואף רצוי לכבות את רכיבי הגרפיקה שכן הפעלתם מצריכה ריצה איטית יותר (כדי שבני אדם יוכלו להבין את מהלך המשחק) וכן שימוש מוגבר במשאבים כמו זיכרון. רכיבי הגרפיקה מרוכזים בשני קבצים עיקריים, קרי, graphicsDisplay.py ו- graphicsUtils.py והם מבוססים על חבילת הגרפיקה tkinter של פייתון [18]. כאשר רכיבי הגרפיקה פועלים, בכל רגע נתון המצג הגרפי משקף את במידע שנמצא ברכיב ייצוג מצב המשחק.

3.2.4 <u>סוכנים</u>

כאמור, המטרה העיקרית של הפלטפורמה של ברקלי היא לאפשר למפתחים ולסטודנטים יכולת ליישם כל אלגוריתם שהוא בכדי לשחק את משחק הפק-מן. בכדי להשיג מטרה זו, מפתחי הפלטפורמה ייצרו ממשק (interface) של סוכן (או אלגוריתם) עם חוזה ברור, כך שכל אלגוריתם שרוצים להפעיל על הפלטפורמה חייב לממש את הממשק הנתון. הממשק מוגדר במחלקה Agent שנמצאת ב- game.py ולה השיטות הבאות:

3.2.4.1 רישום מצב ראשוני

רישום מצב ראשוני הוא השיטה הראשונה שנקראת על ידי הפלטפורמה כאשר מתחיל משחק חדש. השיטה מספקת מצביע לרכיב ייצוג מצב המשחק והיא בעלת החתימה: registerInitialState(self, state). באמצעות שיטה זו הפלטפורמה מודיעה לאלגוריתם כי החל משחק חדש והוא בתורו יכול לנצל זאת למשל לאתחול משתנים.

3.2.4.2 תצפית על מצב

תצפית על מצב היא שיטה הנקראת בכל שלב של המשחק לאחר ביצוע של פעולה. השיטה מספקת מצביע לרכיב ייצוג מצב המשחק והיא בעלת החתימה: observationFunction(self, state). באמצעות שיטה זו הסוכן יודע את תוצאות הפעולה האחרונה שהוא ביצע.

3.2.4.3 ביצוע פעולה

ביצוע פעולה היא שיטה הנקראת בכל שלב של המשחק ותפקידה לאותת לפלטפורמה איזו פעולה על פק-מן לבצע כעת. לאחר ביצוע פעולה זו, ולאחר שכל הרוחות במשחק ביצעו את פעולותיהן, תקרא השיטה של תצפית על מצב. שיטה זו מספקת מצביע לרכיב ייצוג מצב המשחק והיא בעלת החתימה: getAction(self, state). במשחק פק-מן ניתן לבצע חמש פעולות, קרי, לנוע לכל אחד מהכיוונים או לא לעשות כלום.

3.2.4.4 סיום

סיום היא שיטה הנקראת בכל סיום של משחק. שיטה זו מספקת מצביע לרכיב ייצוג מצב המשחק והיא בעלת החתימה: final(self, state).

3.2.5 ניהול המשחקים

כאמור השימוש בפלטפורמה מספק יכולת לאמן מודלים שונים במטרה לנצח את משחק הפק-מן. בעת הפעלת הפלטפורמה לצורך אימון, על המפעיל לספק את מספר משחקי האימון הדרוש. הרכיב שמנהל את הפעלת המשחקים נקרא Game והוא חלק מהקובץ game.py.

3.3 יישום DQN בפק-מן של ברקלי

בסעיף זה נדון ברכיבים שנדרשתי לממש בכדי ליישם את אלגוריתם DQN מעל פלטפורמת פק-מן של ברקלי.

3.3.1 זיכרון שימוש חוזר

רכיב מרכזי ביישום של אלגוריתם DQN הוא רכיב זיכרון בשימוש חוזר (Replay Memory) רכיב זה מאפשר למשתמשים בו לאגור זיכרונות של פעולה-מצב-תגמול שהסוכן נתקל בהם בעבר. רכיב הזיכרון מוגבל בגודל ומאפשר לשמור כמות מוגדרת מראש של זיכרונות. בעת הכנסת זיכרון חדש ובאם הזיכרון מלא, הזיכרון הרחוק ביותר יימחק והחדש יישמר במקומו. כמו כן, רכיב זה ממש שיטה המאפשרת דגימה אחידה של זיכרונות בגודל מבוקש מעל מרחב הזיכרון. רכיב זה ממומש תחת מחלקה בשם ReplayMemory שבקובץ replayMemoy.py. בעת יצירת מופע של מחלקה זו, יש לספק לבנאי את גודל הזיכרון המבוקש. כמו כן, המחלקה מממשת שיטה המאפשרת לשמור לדיסק הקשיח את מצב הזיכרון וכן לטעון אותו משם. פונקציונליות זו חשובה כאשר רוצים לשמור מודל אימון.

3.3.2 פרמטרים

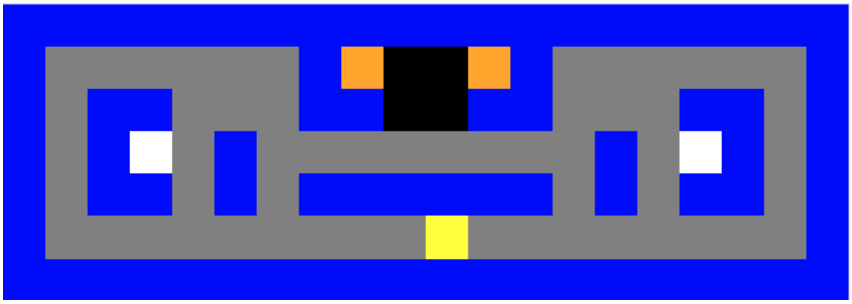
אלגוריתם DQN זקוק למספר רב של פרמטרים כמו גודל זיכרון שימוש חוזר, אפסילון התחלתי, מקדם דעיכה אפסילון וכד'. חלק מן הפרמטרים הינם קבועים לכל אורך האימון וחלקם משתנים. בתהליך המחקר ישנו צורך לכוון את הפרמטרים הללו, וכן דרושה יכולת לשמור את מצבם (לפחות את אלו שמשתנים תוך כדי האימון) כחלק משמירת המודל כולו. על כן, יצרתי מחלקה בשם DQNParameters בקובץ dqnParameters.py אשר מכילה את כל הפרמטרים שאלגוריתם DQN זקוק בכדי לרוץ. המחלקה מארגנת את הפרמטרים במבנה של מילון, כך שכל מפתח הוא בעל ערך קבוע וניתן להתייחס אליו בקלות בכל מקום בו הוא נדרש. נספח א מרכז את שמות כל הפרמטרים, הערכים שלהם ומשמעותם.

3.3.3 רשת Q-עמוקה

רשת ה-Q העמוקה המהווה חלק מרכזי באלגוריתם DQN ממומשת במחלקה DeepQNetwork בקובץ deepQNetwork.py. המחלקה ממומשת באמצעות פלטפורמת TensorFlow אשר מאפשרת יצירה ואימון של מודלים של למידה עמוקה. הבנאי של המחלקה אחראי לייצר את הרשת על פי המבנה שהוגדר ב- [7] קרי, שלוש שכבות קונבולוציוניות ושתי שכבות מקושרות באופן מלא. כמו כן, הבנאי מאתחל גם את יחידת המיטוב (Optimizer) לרשת הנוירונים תחת פונקציית ההפסד שהוגדרה ב- [7]. יתר השיטות במחלקה עוסקות באימון, ניבוי, שמירה וטעינה של המודל.

3.3.4 ממיר פריימים

כפי שהוסבר בפרק הקודם, אלגוריתם DQN מקבל כקלט את פלט המסך של משחק הוידאו. היות וקצב דגימת פלט המסך מעל פלטפורמת פק-מן של ברקלי (באמצעות שימוש בחבילת הקוד Tkinter) הוא איטי (כעשירית השנייה לפריים) היה צורך לפתח דרך מהירה יותר לקבלת תמונת הפלט. לשם כך יצרתי את הרכיב ממיר הפריימים אשר תפקידו לייצור תמונה חלופית לזו שמיוצרת באמצעות חבילת הקוד Tkiner, אך הכוללת את כל האלמנטים שיש בתמונה המקורית. ממיר הפריימים מיושם בקובץ frameConvertor.py. איור 3.2 מציג פלט מסך חלופי לפלט המסך של איור 3.1ב.



איור 3.2 – ייצוג גרפי חלופי לזה המיוצר על ידי חבילת הקוד tkinter לאיור 3.1ב. כל הפרטים (צבע, אוכל, רוחות, פק-מן וקירות) נשמרים גם בייצוג החלופי.

3.3.5 מחסנית פריימים

הקלט לרשת DQN הינו אוסף של ארבעת פלטי מסך אחרונים של פק-מן. בכדי לערום בצורה נוחה פלטי מסך אלו, יצרתי את הרכיב FrameStack בקובץ framStack.py. רכיב זה מאפשר להוסיף לאוסף קיים פלט חדש על חשבון הפלט הוותיק ביותר.

3.3.6 יומן תיעוד אירועים

יומן תיעוד האירועים הוא רכיב אשר באמצעותו ניתן לכתוב לפלט הסטנדרטי אירועים משמעותיים במהלך אימון המודל. ישנה חשיבות גדולה מאוד לניהול יומן אירועים מסודר וברור בכדי לפתור בעיות ולהבין האם האלגוריתם פועל כצפוי. רכיב יומן האירועים מיושם בקובץ loggingUtils.py. הרכיב מאפשר לכתוב לפלט הסטנדרטי במספר רמות כאשר כל רמה נצבעת בצבע שונה, לדג' רמת שגיאה נצבעת באדום ורמה תקינה בירוק. כל שורה הנכתבת על ידי יומן האירועים מלווה בתאריך ושעה של הכתיבה וכן בשם הפונקציה הקוראת. איור 3.3 מציג אוסף של שורות שנכתבו ליומן במהלך אימון.

```
[Sun Nov 11 01:15:09 2018] [DEBUG] [_init_dqn_params:52]    rl_discount_factor:0.95
[Sun Nov 11 01:15:09 2018] [DEBUG] [_init_dqn_params:52]    replay_memory_size:1000
[Sun Nov 11 01:15:09 2018] [DEBUG] [_init_dqn_params:52]    frame_stack_size:4
[Sun Nov 11 01:15:09 2018] [DEBUG] [_init_dqn_params:52]    frame_height:7
[Sun Nov 11 01:15:09 2018] [DEBUG] [_init_dqn_params:52]    model:smallClassic_20-10-2018_19:32
[Sun Nov 11 01:15:09 2018] [DEBUG] [_init_dqn_params:52]    frames_before_training:10000
[Sun Nov 11 01:15:09 2018] [DEBUG] [_init_dqn_params:52]    l3_filter_size:3
[Sun Nov 11 01:15:09 2018] [INFO] [_init_replay_memory:65] Successfully loaded saved replay memo
[Sun Nov 11 01:15:10 2018] [INFO] [_load_saved_network_data:143] Successfully loaded saved weig
2018-11-11 01:15:10.710768: W tensorflow/core/framework/op_kernel.cc:1273] OP_REQUIRES failed a
[Sun Nov 11 01:15:10 2018] [INFO] [_load_saved_network_data:140] Model saves/smallClassic_20-10-
[Sun Nov 11 01:15:10 2018] [INFO] [__init__:101] Done initializing DQN Agent.
[Sun Nov 11 01:15:11 2018] [INFO] [final:211] Episode #193055 | Frames: 27300031 | Wins: 51336
[Sun Nov 11 01:15:13 2018] [INFO] [final:211] Episode #193056 | Frames: 27300073 | Wins: 51336
[Sun Nov 11 01:15:14 2018] [INFO] [final:211] Episode #193057 | Frames: 27300101 | Wins: 51336
[Sun Nov 11 01:15:14 2018] [INFO] [final:211] Episode #193058 | Frames: 27300111 | Wins: 51336
[Sun Nov 11 01:15:15 2018] [INFO] [final:211] Episode #193059 | Frames: 27300129 | Wins: 51336
[Sun Nov 11 01:15:15 2018] [INFO] [final:211] Episode #193060 | Frames: 27300151 | Wins: 51336
[Sun Nov 11 01:15:16 2018] [INFO] [final:211] Episode #193061 | Frames: 27300163 | Wins: 51336
```

איור 3.3 – דוגמא לפלט יומן אירועים במהלך אימון מודל DQN. הודעות בצהוב הן הודעות debug, הודעות באדום הן הודעות שגיאה והודעות בירוק הן הודעות מהלך ריצה תקין.

3.3.7 סטטיסטיקות

בכדי לעקוב אחר התכנסות הרשת יש צורך בהגדרה וחישוב של מדדים הקשורים בהתכנסות. שני המדדים שבחרתי להשתמש בהם מחשבים את ממוצע הניצחונות ותוצאת המשחקים עבור מאה המשחקים האחרונים. כלומר, אחרי כל משחק יש לעדכן את ערכי מדדים אלו. מודל שהתכנס יהיה בעל מדד ניצחונות שואף לאחד ומדד תוצאה לא שלילי גבוה מאוד. בכדי לחשב את הממוצע הנע עבור מאה המשחקים האחרונים יצרתי את המחלקה CappedMovingAverage בקובץ cappedMovingAverage.py אשר בעת האתחול מקבלת כפרמטר את גודל הממוצע (לדג' מאה) ומאפשרת הוספה של נקודות חדשות וקבלת הממוצע העכשווי.

3.3.8 סוכן DQN

סוכן DQN הוא ללא כל ספק הרכיב המשמעותי ביותר בכל הפרויקט. רכיב זה משתמש בכל הרכיבים שהוגדרו לעיל תוך כדי השתלבות בתוך פלטפורמת פק-מן של ברקלי. סוכן DQN מממש את הממשק Agent של הפלטפורמה וממומש בקובץ deepQNetworkAgents.py.

פרק ד: תוצאות יישום אלגוריתם DQN עבור משחק הוידאו פק-מן

במהלך העבודה על הפרויקט ביצעתי עשרות של ריצות במטרה לבדוק את טיב האלגוריתם ובכדי לכייל את הפרמטרים שלו. בכדי לקבל תוצאות מהירות הריצות הראשונות נעשו עבור שלבי משחק קטנים יחסית, קרי, smallGrid ו- mediumGrid. לאחר שכיילתי את הפרמטרים השונים אימנתי מודל מעל שלב משחק מורכב הרבה יותר smallClassic. בפרק זה נדון בתוצאות הריצות עבור שלבי המשחק השונים.

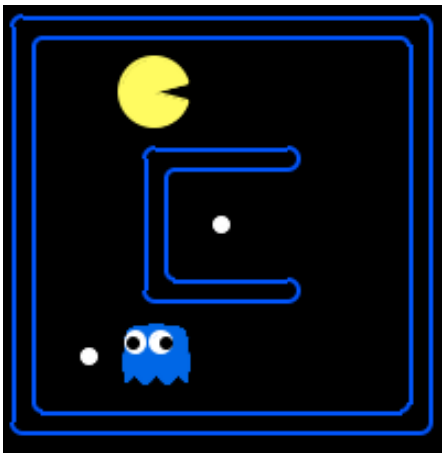
הערה: כל הריצות נעשו במחשב וירטואלי בענן של גוגל (GCP). נספח ב' מתאר את פרטי מכונת החישוב.

4.1. מדדי ביצועים

בכדי לאמוד הצלחה של אלגוריתם DQN יש צורך ליצור מדד ביצועים. מדד הביצועים צריך לשקף את היכולת של סוכן DQN להצליח במשימה שהוא נדרש לבצע, קרי, ניצחון משחק פק-מן. כמובן שניצחון בודד אינו משקף את היכולות של הסוכן ולכן יש צורך למדוד לאורך מספר רב של משחקים. בפרויקט זה בחרתי שני מדדים שלדעתי משקפים את היכולת של הסוכן לנצח את פק-מן. המדד הראשון מתבסס על רצף ניצחונות אחרונים, בפרט מאה המשחקים האחרונים. נגדיר שאפס במדד זה מייצג אפס ניצחונות במאה המשחקים האחרונים ששוחקו, אחד מייצג מאה ניצחונות רצופים וכל הערכים שבין אפס לאחד מייצגים את מספר הניצחונות של הסוכן במאה המשחקים האחרונים. המדד השני משקף את ממוצע תוצאות מאה המשחקים האחרונים ששוחקו. במקרה זה מעניין לראות מתי ממוצע זה עובר את רף האפס, כלומר בממוצע פק-מן מייצר ניקוד גבוה יותר. ניקוד חיובי מסמן לנו שפק-מן אוכל את נקודות האוכל, נע כל הזמן ואפילו אוכל רוחות. על כן, ציון גבוה ופרט ציון חיובי מייצגים הצלחה של הסוכן.

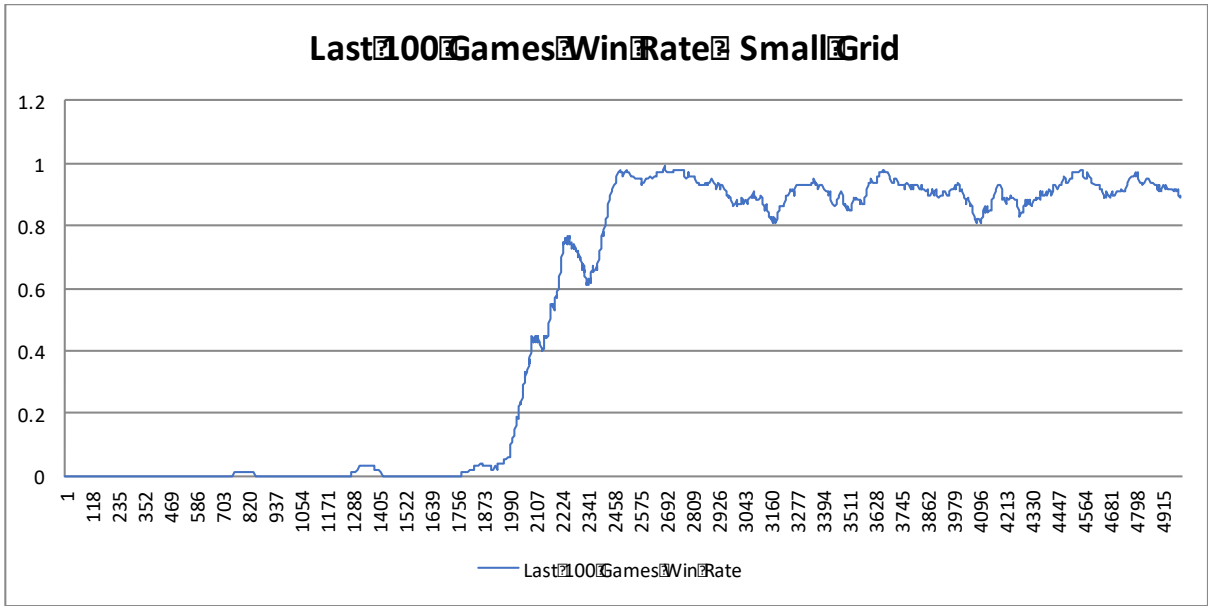
4.2. שלב המשחק smallGrid

שלב המשחק smallGrid הינו בגודל שבע על שבע. הוא מכיל רוח אחת, שתי נקודות מאכל ואת פק-מן כפי שמתואר על ידי איור 4.1.

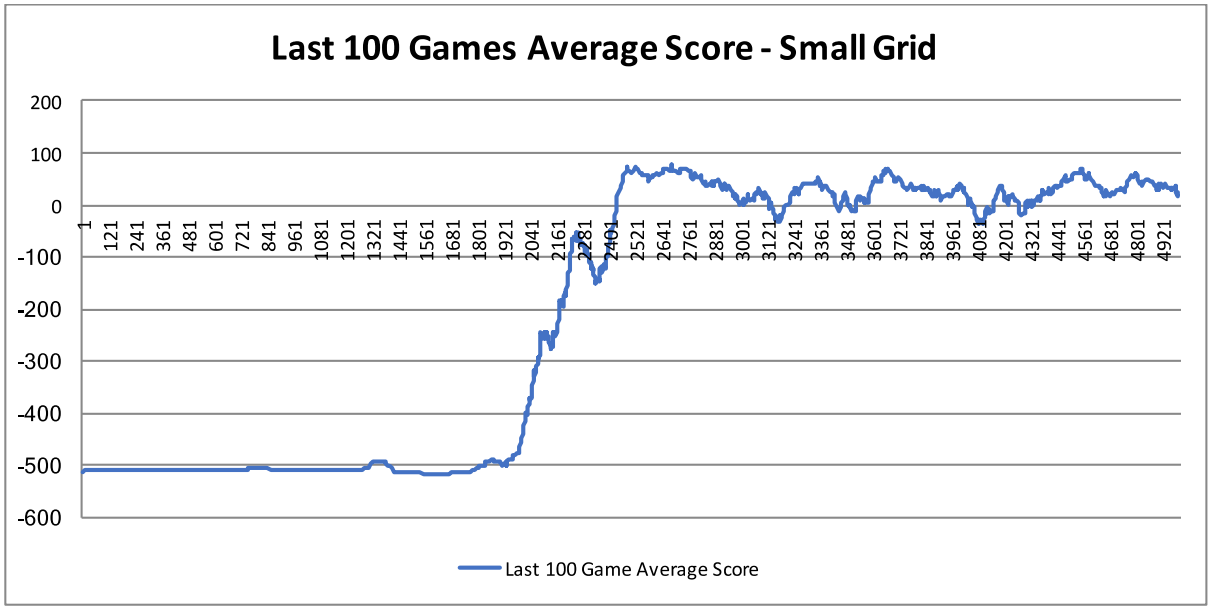


איור 4.1 – תיאור גרפי של שלב המשחק smallGrid.

תוצאות ריצת אלגוריתם DQN עבור שלב המשחק smallGrid מובאות באיורים 4.2 ו- 4.3. איור 4.2 מתאר את היחס בין ממוצע נצחנות עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ-1800 משחקים סוכן DQN החל מנצח משחקים ולאחר כ- 2500 משחקים רצף הניצחונות נמצא באזור של 95 אחוז. איור 4.3 מתאר את היחס בין ממוצע תוצאות מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ- 2500 משחקים ממוצע התוצאות הופך חיובי.



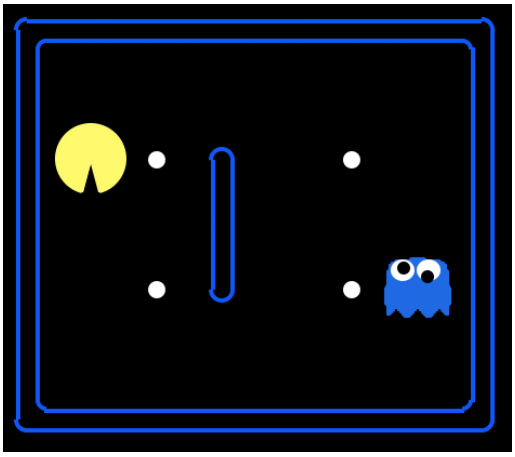
איור 4.2 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק smallGrid עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. לאחר כ- 2500 משחקים רצף הניצחונות נמצא באזור ה- 95 אחוז.



איור 4.3 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק smallGrid לבין מספר המשחקים ששוחקו. לאחר כ- 2500 משחקים ממוצע התוצאות הפך חיובי.

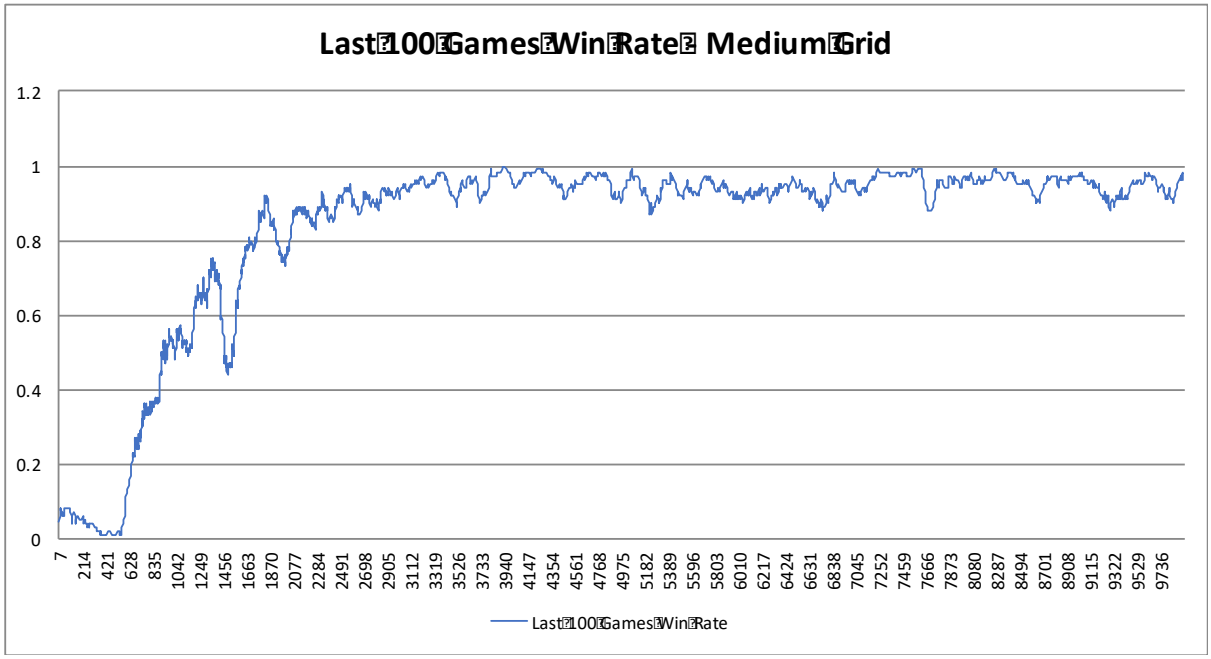
4.3. שלב המשחק mediumGrid

שלב המשחק mediumGrid הינו בגודל שמונה על שבע. הוא מכיל רוח אחת, ארבע נקודות מאכל ואת פק-מן כפי שמתואר על ידי איור 4.4.

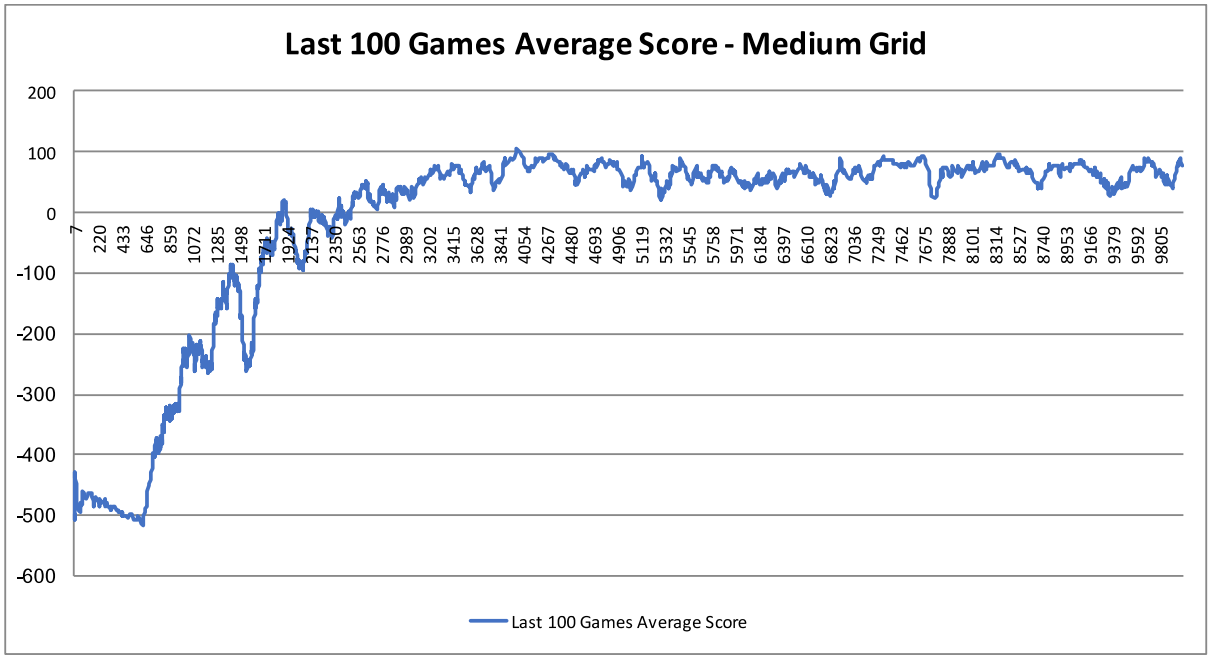


איור 4.4 – תיאור גרפי של שלב המשחק mediumGrid.

תוצאות ריצת אלגוריתם DQN עבור שלב המשחק mediumGrid מובאות באיורים 4.5 ו- 4.6. איור 4.5 מתאר את היחס בין ממוצע נצחנות עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ-650 משחקים סוכן DQN החל מנצח משחקים ולאחר כ-3100 משחקים רצף הניצחונות נמצע באזור של 95 אחוז. איור 4.6 מתאר את היחס בין ממוצע תוצאות מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ- 2600 משחקים ממוצע התוצאות הופך חיובי.



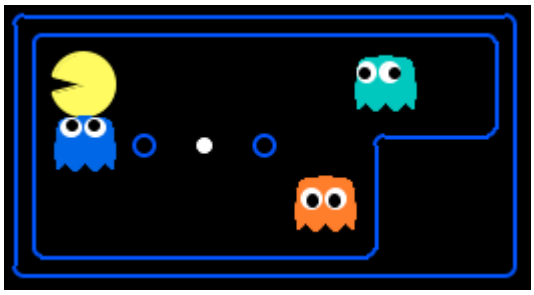
איור 4.5 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק mediumGrid עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. לאחר כ- 3100 משחקים רצף הניצחונות נמצא באזור ה- 95 אחוז.



איור 4.6 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק mediumGrid לבין מספר המשחקים ששוחקו. לאחר כ- 2600 משחקים ממוצע התוצאות הפך חיובי.

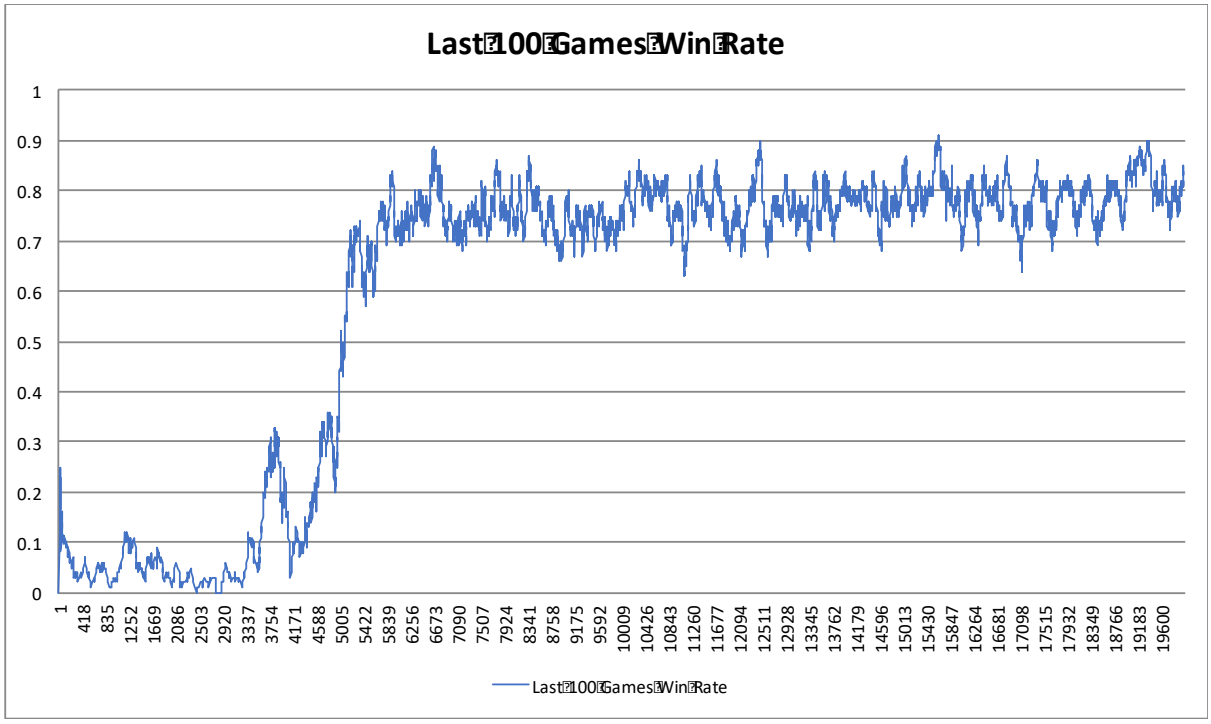
4.4. שלב המשחק minimaxClassic

שלב המשחק minimaxClassic הינו בגודל תשע על חמש. הוא מכיל שלוש רוחות אחת, נקודת מאכל אחת ואת פק-מן כפי שמתואר על ידי איור 4.7.

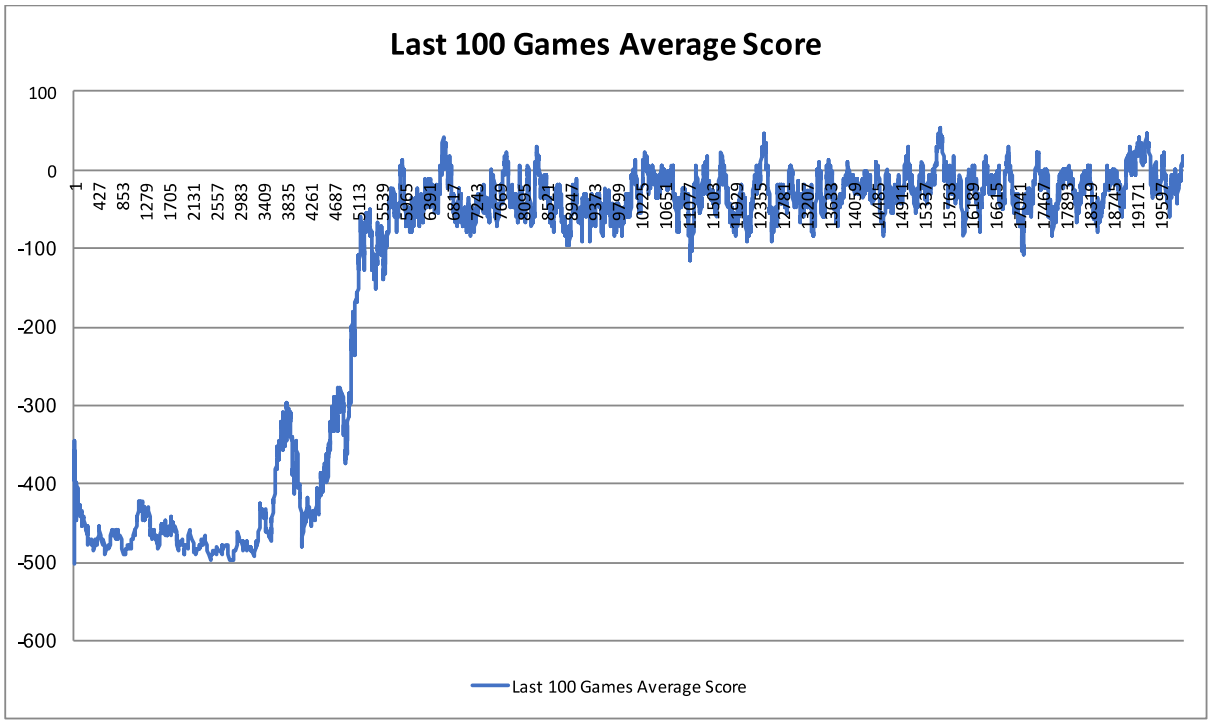


איור 4.7 – תיאור גרפי של שלב המשחק minimaxClassic.

תוצאות ריצת אלגוריתם DQN עבור שלב המשחק minimaxClassic מובאות באיורים 4.8 ו- 4.9. איור 4.8 מתאר את היחס בין ממוצע נצחנות עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ-3000 משחקים סוכן DQN החל מנצח משחקים ולאחר כ- 15000 משחקים רצף הניצחונות נמצע באזור של 90 אחוז. איור 4.9 מתאר את היחס בין ממוצע תוצאות מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ- 6000 משחקים ממוצע התוצאות הופך חיובי.



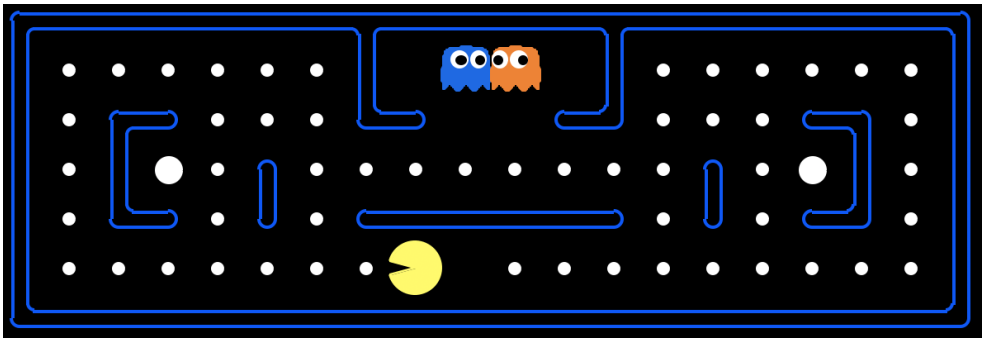
איור 4.8 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק minimaxClassic עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. לאחר כ- 15000 משחקים רצף הניצחונות נמצא באזור ה- 90 אחוז.



איור 4.9 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק minimaxClassic לבין מספר המשחקים ששוחקו. לאחר כ- 6000 משחקים ממוצע התוצאות הפך חיובי.

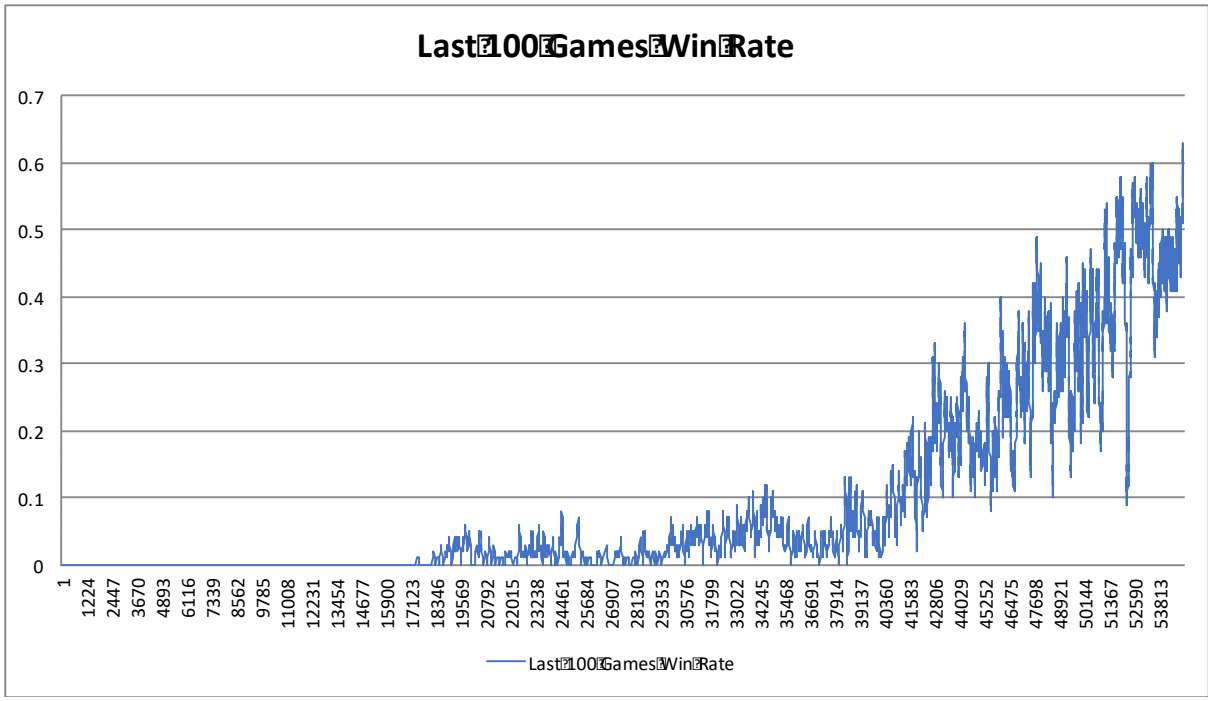
4.5. שלב המשחק smallClassic

שלב המשחק smallClassic הינו בגודל עשרים על שבע. הוא מכיל שתי רוחות אחת, 46 נקודות מאכל, שתי נקודות מאכל גדולות ואת פק-מן כפי שמתואר על ידי איור 4.10.

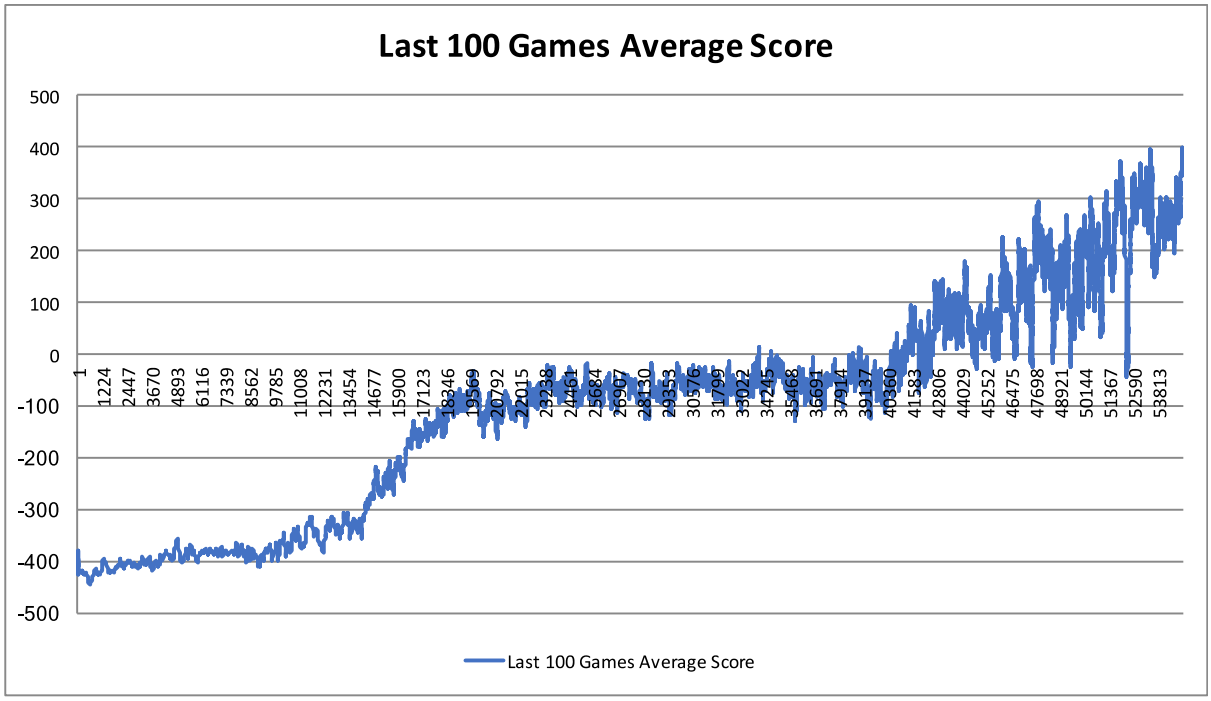


איור 4.10 – תיאור גרפי של שלב המשחק smallClassic.

תוצאות ריצת אלגוריתם DQN עבור שלב המשחק smallClassic מובאות באיורים 4.11 ו- 4.12. איור 4.11 מתאר את היחס בין ממוצע נצחנות עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ-18000 משחקים סוכן DQN החל מנצח משחקים ולאחר כ- 53000 משחקים רצף הניצחונות נמצע באזור של 60 אחוז. האימון עבור שלב זה רץ במשך עשרה ימים ונראה כי המגמה היא המשך שיפור ביצועי הסוכן לאורך זמן. איור 4.12 מתאר את היחס בין ממוצע תוצאות מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. ניתן לראות שאחרי כ- 18000 משחקים ממוצע התוצאות הופך חיובי.



איור 4.11 - היחס בין ממוצע ניצחונות של סוכן DQN בשלב משחק smallClassic עבור מאה משחקים אחרונים לבין מספר המשחקים ששוחקו. לאחר כ- 18000 משחקים הסוכן מתחיל לנצח משחקים, ולאחר כ- 53000 משחקים הסוכן מנצח ב- 60 אחוז מן המשחקים.



איור 4.12 - היחס בין ממוצע תוצאות מאה משחקים אחרונים של סוכן DQN בשלב משחק smallClassic לבין מספר המשחקים ששוחקו. לאחר כ- 18000 משחקים ממוצע התוצאות הפך חיובי.

סיכום

מטרת עבודה זו הייתה להראות שניתן ליישם בהצלחה את אלגוריתם DQN עבור משחק הוידאו פק-מן, וזאת בהמשך ישיר לעבודה המסכמת שכתבתי בנושא. משימת היישום כללה בתוכה מספר תתי משימות:

- מציאת פלטפורמה מתאימה למשחק פק-מן והבנה מלאה של דרך העבודה מעליה.
- לימוד והבנה של ספריות למידה עמוקה בפייתון.
- יישום אלגוריתם DQN תוך כדי מימוש רשת נוירונים עמוקה והטמעתו מעל הפלטפורמה של משחק הוידאו פק-מן.
- ביצוע מספר רב של ניסויים עד למציאת הפרמטרים המתאימים עבור אלגוריתם DQN אשר מייצרות את התוצאות הטובות ביותר עבור משחק הוידאו פק-מן.

כעת, עם סיום העבודה ניתן לומר בבירור כי ניתן ליישם בהצלחה את אלגוריתם DQN עבור משחק הוידאו פק-מן כפי שניתן לראות בתוצאות הניסויים שהובאו בפרק הקודם. עם זאת, במהלך העבודה על הפרויקט נתקלתי בשני קשיים אשר לדעתי מעידים במידה כזו או אחרת על הבעייתיות ביכולת להשתמש באלגוריתם DQN עבור יישומיים כלליים. ראשית, אימון רשת נוירונים עמוקה היא פעולה חישובית כבדה אשר אורכת זמן ממושך ואשר משפיעה בצורה ישירה על זמן פיתוח האלגוריתם ובדיקתו. שנית, היכולת של אלגוריתם DQN לפתור בעיה בצורה טובה, תלויה בצורה ישירה בבחירת הפרמטרים שלו. אולם, טרם נמצאה שיטה יעילה לבחירת פרמטרים, והדרך הטובה ביותר כפי שהחוקרים של אלגוריתם DQN [7] מציינים היא באמצעות ניסוי וטעייה. על כן, נדרשים ניסויים רבים בכדי לכייל את הפרמטרים בצורה מיטבית.

נספח א: רשימת פרמטרים לאלגוריתם DQN

המחלקה DQNParameters מהווה מקום מרכזי יחיד בקוד של הפרויקט לכל הפרמטרים הדרושים על ידי אלגוריתם DQN. הצורך במחלקה שכזו עלה משתי סיבות עיקריות, ראשית, הערכים הסופיים של הפרמטרים הם תוצאה של עבודת מחקר אשר במהלכה הם שונו מספר פעמים עד אשר התקבלו תוצאות טובות. עצם העובדה שכולם מרוכזים יחד פישטה את התהליך. שנית, חלק מן הפרמטרים משתנים תוך כדי אימון המודל, ועל כן, בעת שמירת המודל יש צורך לשמור גם את ערכיהם. על כן, היה צורך לייצר מחלקה אשר תספק דרישה זו בקלות רבה. להלן טבלה של כל הפרמטרים המרוכזים ב- DQNParametes, ערכיהם ומשמעותם.

שם	ערך	משמעות
FRAME_STACK_SIZE	4	מספר פריימים אחרונים שיש לכלול בקלט
NUM_OF_ACTIONS	4	מספר הפעולות האפשריות
RL_DISCOUNT_FACTOR	0.95	מקדם הפחתה למידה באמצעות חיזוקים
RL_EPSILON_START	1.0	ערך אפסילון התחלתי
RL_EPSILON_CURRENT	משתנה	ערך אפסילון עכשווי
RL_EPSILON_END	0.1	ערך אפסילון סופי
RL_EPSILON_FRAMES_DECAY	10000	מספר פריימים בהם יש להפחית את אפסילון עד הגיעו לערך אפסילון סופי
FRAMES_BEFORE_TRAINING	10000	מספר פריימים שיש לחכות לפני התחלת אימון המודל
OPTIMIZER_LEARNING_RATE	0.00025	קצב למידה של יחידת המיטוב (Optimizer) ברשת הנוירונים
REPLAY_MEMORY_SIZE	100000	גודל זיכרון שימוש חוזר
REPLAY_MEMORY _SAMPLE_BATCH_SIZE	64	גודל דגימה מעל מרחב זיכרון שימוש חוזר
MODEL_SAVE _INTERVAL_IN_FRAMES	100000	מספר הפריימים שיש להמתין בין שמירה לשמירה של המודל
TARGET_MODEL_UPDATE _INTERVAL_IN_FRAMES	1000	מספר הפריימים שיש להמתין בין עדכון לעדכון של רשת המטרה
L2_FILTER_SIZE	3	גדול גרעין מסנן שכבה קונבולציונית 1
L2_FILTERS	8	מספר מסננים שכבה קונבולציונית 1
L2_DEPTH	4	עומק שכבה קונבולציונית 1
L2_STRIDE	1	אורך פסיעה שכבה קונבולציונית 1
L3_FILTER_SIZE	3	גדול גרעין מסנן שכבה קונבולציונית 2

L3_FILTERS	16	מספר מסננים שכבה קונבולציונית 2
L3_DEPTH	8	עומק שכבה קונבולציונית 2
L3_STRIDE	1	אורך פסיעה שכבה קונבולציונית 2
L4_FILTER_SIZE	4	גדול גרעין מסנן שכבה קונבולציונית 3
L4_FILTERS	32	מספר מסננים שכבה קונבולציונית 3
L4_DEPTH	16	עומק שכבה קונבולציונית 3
L4_STRIDE	1	אורך פסיעה שכבה קונבולציונית 3
FC_RECTIFIER_UNITS	256	מספר יחידות בשכבה מקושרת מלאה
EPISODES	משתנה	מספר המשחקים ששוחקו עד כה
FRAMES	משתנה	מספר הפריימים שנדגמו עד כה
WINS	משתנה	מספר הניצחונות עד כה

נספח ב: מפרט מחשב עליו התבצעו ניסויי הפרויקט

כל הניסויים במסגרת פיתוח וכתיבת פרויקט זה נעשו מעל מחשב וירטואלי בענן של גוגל (GCP). להלן פרטי מכונת החישוב:

Machine type:
n1-highmem-2 (2 vCPUs, 13 GB memory)

CPU platform:
Intel Broadwell

GPUs:
x NVIDIA Tesla K80 1

Disk
100 GB Standard persistent disk

Zone:
us-west1-b

מקורות

[1] Sutton, R. S. and Barto, A. G. (2016). Reinforcement Learning 2'nd edition. MIT Press, Boston, MA.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[3] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521.7553 (2015): 436-444.

[4] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.

[5] Krakovsky, M. (2016). Reinforcement renaissance. Communications of the ACM, 59(8): 12–14.

[6] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2017). Deep reinforcement learning that matters. arXiv preprint arXiv: 1709.06560.

[7] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529.

[8] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Chen, Y. (2017). Mastering the game of go without human knowledge. Nature, 550(7676), 354.

[9] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv: 1511.06581. Chicago

[10] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, June). Deterministic policy gradient algorithms. In ICML.

[11] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning (pp. 1928-1937).

[12] Knight, W. (2017). Reinforcement learning. MIT technology review 2017. https://www.technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learning/

[13] Hof, R. D. (2013). Deep Learning. MIT technology review 2013. https://www.technologyreview.com/s/513696/deep-learning/

[14] Garnelo, M., Arulkumaran, K., & Shanahan, M. (2016). Towards deep symbolic reinforcement learning. arXiv preprint arXiv: 1609.05518.

[15] Silver, D. (2016). Deep reinforcement learning, a tutorial at ICML 2016. http://icml.cc/2016/tutorials/deep_rl_tutorial.pdf.

[16] DeNero, J., Klein, D., & Abbeel, P. (2014). Berkley Pac-Man Project Overview. http://ai.berkeley.edu/project_overview.html

[17] DeNero, J., Klein, D., & Abbeel, P. (2014). Berkley Pac-Man Project Instructions. http://ai.berkeley.edu/project_instructions.html

[18] Python Software Foundation. (2018). Tkinter Documentation. https://docs.python.org/2/library/tkinter.html

Abstract

Reinforcement learning is a sub field of machine learning that deals with the interaction of an *agent* and its *environment* while learning an *optimal policy* by trial and error. The integration of reinforcement learning and deep learning has a long history, but with recent achievements in deep learning we have been witnessing a renaissance of reinforcement learning. The number of scientific publications keeps growing steadily year to year for the past two and a half decades. Breakthrough algorithms such as DQN and AlphaGo have been an inspiration for hundreds of extensions and practical real-life applications that soon came after. The researches that create DQN have shown that it is possible to create a reinforcement learning agent that can play tens of Atari games in a human level control, with only the visual output as its input. In this work, I show that it is possible to implement DQN successfully for the video game Pac-Man.

Table of Contents

Drawings List 1
Abstract 2
1. Introduction 3
2. Deep Q-Networks 4
    2.1 Pre-Process 6
    2.2 Network Architecture 6
    2.3 Algorithm 6
3. Implementing DQN to play Pac-Man 8
    3.1 Pac-Man 8
    3.2 Berkley's Pac-Man Framework 8
        3.2.1 Game Layout 8
        3.2.2 Game State 9
        3.2.3 Graphic Components 9
        .3.24 Agents 9
            .3.24.1 Register Initial State 10
            .3.24.2 State Observation 10
            .3.24.3 Action 10
            .3.24.4 Final 10
        .3.25 Game Management 10
    3.3 DQN over Berkley's Pac-Man Framework 10
        3.3.1 Replay Memory 10
        3.3.2 Parameters 11
        3.3.3 Deep Q-Network 11
        .3.34 Frame Convertor 11
        .3.35 Frame Stack 11
        .3.36 Logger 12
        .3.37 Statistics 12
        .3.38 DQN Agent 12
4. Implementing DQN to play Pac-Man - Results 13
    4.1 Performance Measurements 13
    4.2 Game Layout: smallGrid 13
    4.3 Game Layout: mediumGrid 14
    4.4 Game Layout: minimaxClassic 16
    4.5 Game Layout: smallClassic 17
5. Conclusions 20
Appendix A: Parameters List for DQN Algorithm 21
Appendix B: Experiment's Machine Specification 23

References 24

THE OPEN UNIVERSITY OF ISRAEL

THE DEPARTMENT OF COMPUTER SIENCE

MASTER'S DEGREE

Advanced Project [22997]

Implementation of Deep Q-Network for the Video Game of Pacman

Hanan Aharonof 021893730

Supervised by Dr. Mireille Avigal

March 2019