בינה מלאכותית במשחק האסטרטגיה Dota 2

נועם שדה

בהנחיית ד״ר מיה הרמן

נובמבר 2023

תוכן עניינים

1 מבוא

ב-2017, חוקרי OpenAI פיתחו סוכן בינה מלאכותית היכול לשחק במשחק ספורט אלקטרוני (esports) בשם Dota 2 ולנצח את אלוף העולם במשחק אחד נגד אחד. הישג מכובד זה היווה אבן דרך חשובה על מנת לשחק את המשחק האמיתי (בקבוצה של 5 נגד 5). ב-2019, פותחה הבינה המלאכותית OpenAI Five, שיפור של הגרסה הקודמת שלה, שהפכה להיות מערכת הבינה המלאכותית הראשונה שהביסה את אלופי העולם ב- Dota 2. בנוסף, כשהיא שיחקה נגד קבוצות אנושיות היא ניצחה ב-99.4% מהמשחקים הכוללים יותר מ-7000 משחקים [12].

המשחק Dota 2 מציג אתגרים חדשים עבור מערכות בינה מלאכותית כמו אופקי זמן ארוכים, נתונים לא מושלמים ומסובכים, הרבה מצבי פעולה מתמשכים. אתגרים אלו מייצגים את האתגרים איתן מתמודדות מערכות בינה מלאכותית מודרניות.

הבינה המלאכותית OpenAI Five מינפה טכניקות של למידת חיזוק (Reinforcement Learning - RL) אשר נעשה בהן שימוש במחקרים אחרים, העצימה אותן, והוסיפה להן עוד שיטות של רשתות נוירונים עמוקות על מנת לאפשר למחשב ללמוד את חוקי המשחק ולהתאים את התנהגותו בהתאם. החוקרים של OpenAI פיתחו מערכת אימון וכלים שאפשרו לאמן את OpenAI Five במשך 10 חודשים, על ידי כך שהיא שיחקה עם עצמה ועם גרסאות קודמות שלה במשך זמן ממושך [12]. על ידי הבסת אלופי העולם במשחק, OpenAI Five מדגימה את שיטת ה-"המשחק העצמי" של למידת חיזוקים היכולה להגיע לביצועים "על-אנושיים" במשימות מורכבות.

המחקר נערך על ידי צוות מהאוניברסיטה של קליפורניה, ברקלי, והוא מתאר את תהליך הפיתוח של סוכני ה- AI בהם השתמשו על מנת לשחק במשחק. במחקר הוצגו ארכיטקטורות רשתות נוירונים עמוקות חדשניות, וטכניקות שונות שהשתמשו בהם. המחקר מציג התקדמות משמעותית בתחום הלמידה החישובית של משחק וידאו, ומראה כיצד ניתן להשתמש ברשתות נוירונים עמוקות כדי ללמוד משחקים מורכבים.

1.1 היסטוריה קצרה על בינה מלאכותית במשחקי וידאו

בינה מלאכותית (AI) במשחקי וידאו הוא תחום מחקר ופיתוח העוסק בשימוש בבינה מלאכותית כדי ליצור משחקים וירטואליים מאתגרים ומהנים עבור משתתפיהם. טכניקה נפוצה בתחום היא שימוש ב- RL כדי ליצור סוכני AI היכולים ללמוד לשחק משחקים בצורה יעילה, ואף לשחק נגד השחקנים או איתם. הסוכנים מקבלים החלטות ללא מידע מלא על סביבת המשחק, כאשר הם לומדים על ידי ניסוי וטעיה, וקבלת תגמול או עונש על פעולותיהם. דוגמא לשימוש ב-RL במשחקי וידאו היא יצירת אויבים מאתגרים יותר עבור שחקנים, אשר יכולים ללמוד להתאים את האסטרטגיה שלהם על פי משחק השחקן.

השימוש הראשון ב-AI במשחקי וידאו היה בשנות ה-50 עם משחקי שחמט ממוחשבים אשר השתמשו באלגוריתמים פשוטים כדי לשחק שחמט, אך הם לא היו מוצלחים במיוחד. עם הזמן, השתמשו ב-AI במשחקים מתקדמים יותר כמו Pong, Pac-Man, Tetris ועוד. אחד ההישגים המשמעותיים היה בשנת 2013 כאשר החברה DeepMind פיתחה סוכן RL שיכול לשחק במשחקי Atari קלאסיים רבים עם ביצועים

ברמה אנושית, ולפעמים אף טובים יותר. עוד הישג יוצא דופן הוא בשנת 2016, כאשר סוכן RL ניצח במשחק Go את אלוף העולם. זה היה רגע היסטורי בגלל מורכבות המשחק ומספר המהלכים האפשרי שיש במשחק.

הצלחות אלו הובילו להשפעה משמעותית על נושאים שונים הקשורים לבינה מלאכותית ולמידת מכונה. הן אפשרו למודלים של AI למנף את הפוטנציאל שלהם במשימות כל רובוטיקה, נהיגה אוטונומית, ואף משימות בתחום הרפואה.

2 למידה מבוססת חיזוקים – Reinforcement Learning

בשנים האחרונות, השילוב של למידה מבוססת חיזוקים (Reinforcement Learning) עם רשתות נוירונים עמוקות הובילה לפריצות דרך משמעותיות, המאפשרות לסוכנים ללמוד מנתונים ולהתמודד עם משימות מורכבות. תחום ה- RL הוא תחום דינמי ומתפתח במהירות, ולכן בזמן הנוכחי הוא ממשיך להיות נושא מרכזי למחקר פעיל.

RL נמצאת בשימוש במגוון רחב של יישומים ויכולה להתמודד עם בעיות ואתגרים שונים, כגון משחקים, נהיגה אוטומטית, רובוטיקה מערכות בקרת איכות ועוד.

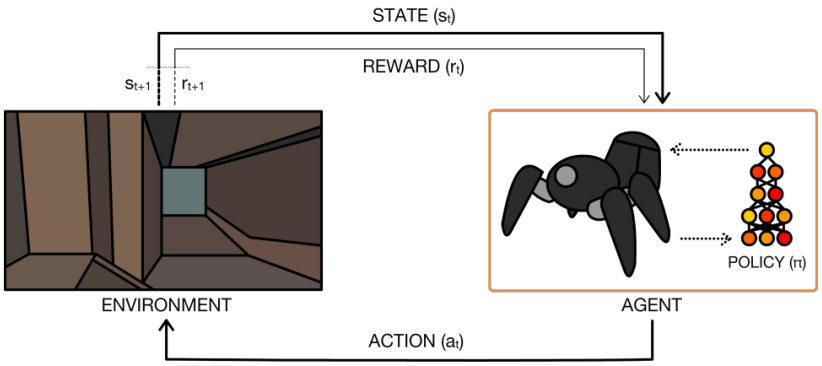
2.1 הסבר על למידה מבוססת חיזוקים

למידה מבוססת חיזוקים (Reinforcement Learning) הוא תחום במדעי המחשב של למידת מכונה הנוגע לאופן שבו סוכנים אינטליגנטים נוקטים פעולות בתוך סביבה מסוימת כדי למקסם את הרווח המצטבר כתוצאה מהפעולות הללו. תחום זה הוא נחשב לאחת משלוש הפרדיגמות של למידת מכונה, לצד למידה מונחית ולמידה בלתי מונחית. במקרה זה, הלמידה לא מסתמכת על מידע קיים ולא ידועה מראש, אלא חוקרת את הסביבה בה הסוכן נמצא.

ב- RL יש 2 מרכיבים עיקריים:

- <u>הסביבה</u>: מערכת המורכבת בדרך כלל מעולם וירטואלי המיוצג ע"י וקטורים ומטריצות מתמטיים. הסביבה היא המקום שבו מתרחשות הפעולות של הסוכן, היא מורכבת ומשתנה, והיא יכולה להחזיר לסוכן משוב בצורה של תגמולים או עונשים. הסוכן לומד מהתגמולים ומהעונשים שהוא מקבל כדי לשפר את הביצועים שלו, וזאת על מנת להשלים את משימתו.
- <u>סוכן</u>: אובייקט הלומד ומקבל החלטות המשלבות אינטראקציה עם הסביבה.

בכל צעד הסוכן נמצא במצב s_t ובוחר פעולה a_t המעבירה אותו למצב s_{t+1} ובהתאם לכך הוא מקבל תגמול r_t. האופן בה מתבצעת הלמידה היא בעזרת התגמול.



איור 1: לולאת הפעולה-תגמול של הסוכן כלפי הסביבה [5]

באופן פורמלי, RL מתואר כתהליך החלטה מרקובי, כלומר תהליך שבו המעברים בין המצבים מקיימים את תכונת מרקוב, לפיה ההתפלגות של מצב מסוים תלויה רק במצב הקודם לו. תהליך זה מתואר באמצעות סט הפרמטרים הבא:

- S - מרחב המצבים: ייצוג המצב הנוכחי או ההקשר של הסוכן בתוך הסביבה. המצב ההתחלתי מסומן ב- S_0.
- A - מרחב הפעולות: האפשרויות שיש לסוכן להשפיע ולשנות את הסביבה.
- T – פונקציית המעבר: הביטוי T(s'|s,a) היא פונקציה המחשבת את ההסתברות לעבור ממצב s למצב s' על ידי הפעולה a, או במילים אחרות היא מחשבת את ההסתברות שבחירת הפעולה at במצב st תביא את הסוכן למצב st+1.
- R – פונקציית תגמול: הביטוי $R(s,s') \rightarrow \mathbb{R}$ הוא פונקציה הנותנת תגמול (רווח או עונש) לכל פעולה a הגורמת למעבר ממצב s למצב s', המסומנת ב- rt בכל צעד בזמן t.
- γ- מקדם הפליית העתיד: קבוע המגדיר את מידת החשיבות של תגמולים עתידיים בהשוואה לתגמולים מידיים.
- π – אסטרטגיה/מדיניות: פונקציה הממפה מצבים להתפלגות הסתברותית כלפי כל פעולה. כלומר זוהי ההסתברות שסוכן יבחר את הפעולה a במצב s היא $\pi(a_t|s_t)$ והיא מתוארת כך:

$$\pi: S \rightarrow p(A = a|S)$$

מטרתו של הסוכן היא ללמוד אסטרטגיה אופטימלית $\pi: S \rightarrow \mathrm{A} = argmax(E[R|\pi])$ הממקסמת את התגמול המצטבר $E_\pi[R_t] = \Sigma_{t=0}^{\infty} \mathrm{R}(s_t, \pi(s_t))$.

ישנן שתי גישות לפתירת בעיות ב-RL [7].

הגישה הראשונה היא - **Model-free learning**:

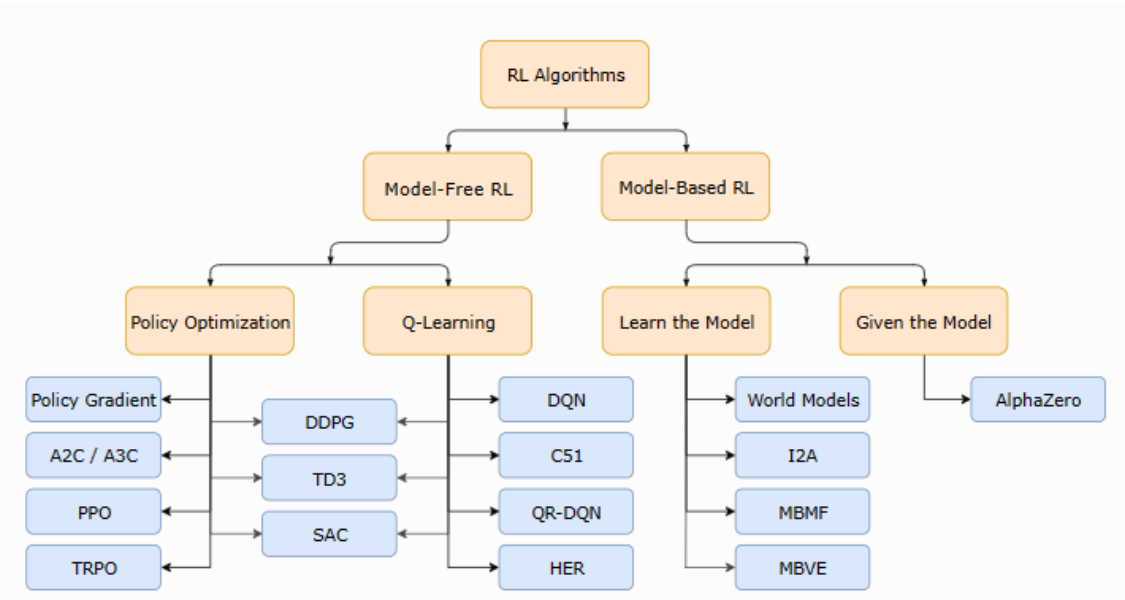
למידה ללא מודל היא טכניקה עבור בעיות שבהן קשה או בלתי אפשרי לקבל מודל של הסביבה. לדוגמה בעיות עם סביבות מורכבות מאוד, או בעיות שבהן הסביבה משתנה באופן דינמי. בגישה זו יש שתי קטגוריות מרכזיות של אלגוריתמים:

- Q-Learning : אלגוריתם שמטרתו ללמוד מדיניות אופטימלית עבור כל מצב פעולה. הוא משתמש בתכנות דינאמי על ידי שמירה של טבלת ערכים הנקראת טבלת Q. טבלה זו מכילה את הערך הצפוי של פעולה בכל מצב. לפיכך הוא עובד רק על מרחב מצבים ופעולות סופי.
- Policy Optimization – אלגוריתמים הלומדים את פונקציית המדיניות המתארת את הסבירות של הסוכן לבצע כל פעולה במצב נתון. האלגוריתמים מחפשים באופן ישיר את המדיניות, ובדרך כלל היא תהיה הסתברותית ולא דטרמיניסטית.

הגישה השנייה היא – **Model-based learning** :

למידה עם מודל היא טכניקה עבור בעיות שבהן יש מודל מדויק של הסביבה. לדוגמא בעיות עם סביבות פשוטות או בעיות שבהן ניתן לקבל מודל של הסביבה מגורמים חיצוניים. גם בגישה זו שתי קטגוריות מרכזיות של אלגוריתמים :

- Model-based RL with a learned model – אלגוריתמים המנסים ללמוד גם את המודל עצמו וגם את האסטרטגיה π.
- Model-based RL with a known model – אלגוריתמים המנסים למצוא את האסטרטגיה π כאשר המודל עצמו נתון.



איור 2: גישות ואלגוריתמים של RL השייכים לכל תחום ותת תחום [7]

כפי שניתן לראות באיור, ישנם הרבה סוגי אלגוריתמים שונים המיועדים למשימות שונות בתחום ה-RL. בסמינר זה אציג רק את אלגוריתם PPO בו השתמשו OpenAI Five.

2.2 Proximal Policy Optimization (PPO)

ניזכר שמטרת הסוכן היא למצוא מדיניות/אסטרטגיה אופטימלית הממקסמת את התגמול המצטבר של הסוכן. בתהליך אימון הסוכן, צריך לגרום לסוכן לנקוט פעולות שמובילות לתגמול ולהימנע מפעולות שמובילות לענישה. עם זאת, אם גודל הצעד של הגרדיאנט בתהליך האימון יהיה קטן מדי אז תהליך האימון יהיה איטי מדי, ואם הצעד יהיה גדול מדי, אז לא בטוח שהמודל יתכנס ולא נגיע לתוצאה הרצויה.

הפתרון לכך הוא אלגוריתם PPO, שהוא אלגוריתם למידה חישובית ללמידה של מדיניות פעולה במשחקים ובבעיות אופטימיזציה אחרות. הוא מבוסס על הרעיון של אופטימיזציה של מדיניות פעולה על ידי שינוי קטן שלה בכל צעד. הארכיטקטורה של אלגוריתם זה עוזרת ליציבות אימון הסוכן על ידי הימנעות מעדכוני מדיניות גדולים מדי. זוהי שיטה גמישה ויציבה, ולכן היא מותאמת למגוון רחב של בעיות.

הנוסחה בה משתמשים באלגוריתם PPO היא [8]:

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t]$$

כאשר r_t הוא יחס ההסתברות בין המדיניות החדשה לישנה:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

הפונקציה clip היא פונקציה המגבילה את הערך של משתנה מסוים לטווח מסוים. במקרה של הנוסחה של PPO, הפונקציה clip משמשת להגבלת המרחק בין המדיניות החדש למדיניות הנוכחית.

האלגוריתם פועל כך:

1. איסוף הנתונים מסוכן ה-AI במשחק באמצעות המדיניות הנוכחית.
2. חישוב התועלת הצפויה עבור כל מצב ופעולה.
3. חישוב היחס בין הסיכוי של המדיניות הנוכחית לנקוט בפעולה מסוימת לבין הסיכוי של המדיניות האופטימלית לנקוט באותה פעולה.
4. עדכון המדיניות הנוכחית על ידי הגדלת הסיכוי לנקוט בפעולות עם יחס גבוה.

Algorithm 1 PPO, Actor-Critic Style

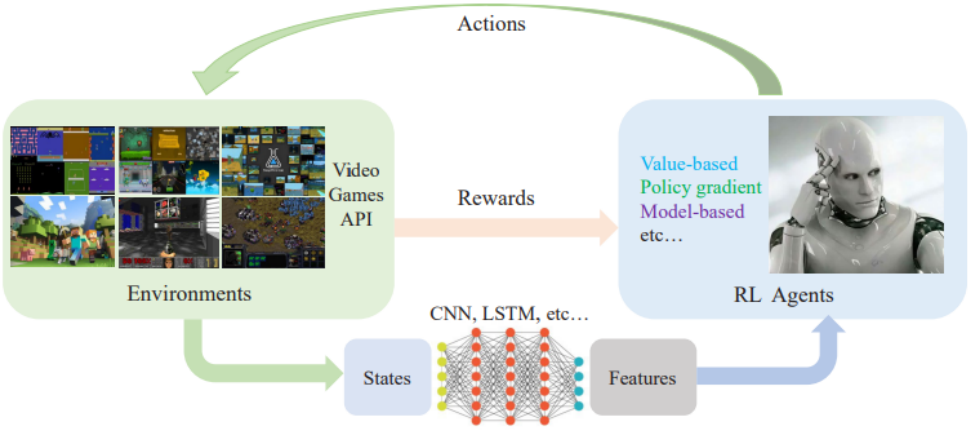
```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

איור 3: פסאודו קוד של אלגוריתם PPO [8]

השימוש ב-PPO על ידי OpenAI Five היא דוגמה מצוינת לאופן שבו אלגוריתמים של RL יכולים לשמש כדי ליצור מערכות מורכבות ואינטליגנטיות.

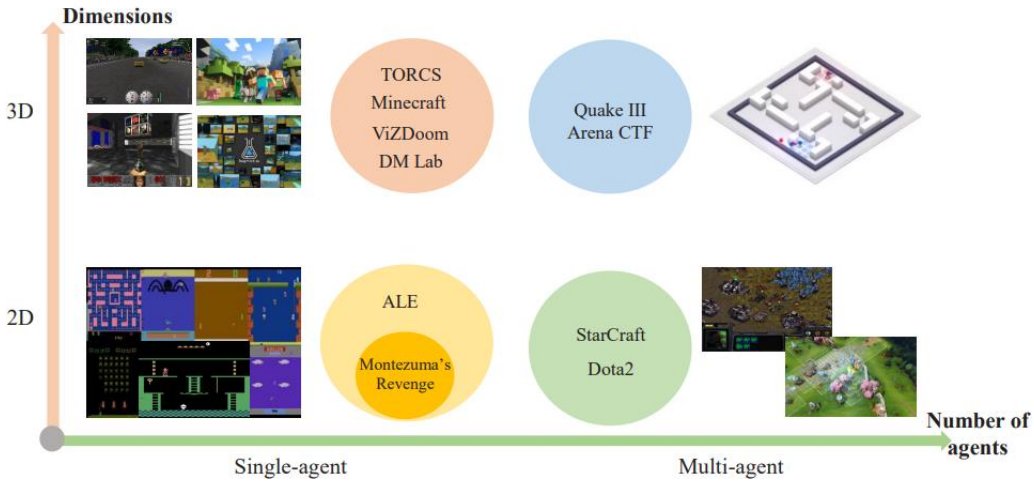
2.3 למידת חיזוקים במשחקי וידאו

RL משמשת במשחקי וידאו על מנת ליצור סוכני AI שיכולים לשחק ברמה גבוהה, ואף לגלות טכניקות משחק חדשות. סוכנים אלה יכולים ללמוד את חוקי המשחק ולפתח אסטרטגיות וטקטיקות על מנת לנצח במשחק. באופן כללי יותר, RL בוחן את האינטראקציות המורכבות בין סוכנים וסביבות המשחקים, ומציג את יכולת קבלת ההחלטות שלהם בסביבות המשחקים. משחקים שונים מספקים בעיות מעניינות ומורכבות לסוכנים לפתור, וכתוצאה מכך חוקרים יכולים ליישם תוצאות אלו במגוון יישומים.



איור 4: דיאגרמת הלולאה של סוכן RL הטיפוסי למשחקי וידאו. מודל הלמידה עמוקה לוקח קלט מה-API של משחקי הוידאו, ומחלץ תכונות משמעותיות באופן אוטומטי. סוכני RL מייצרים פעולות המבוססות על תכונות אלו, וגורמות לסביבה לעבור למצב הבא [6]

RL שימש בהצלחה במגוון רחב של משימות הקשורות למשחקי וידאו, כאשר לכל משחק סביבה שונה וחוקיות משלה. הוא הצליח במשימותיו במשחקים דו ממדיים, משחקים תלת ממדיים, משחקי לוח, משחקים עם שחקן יחיד ואף משחקים מרובי משתמשים. אתגרים כמו סביבת משחק תלת ממדית, מספר מרובה של משתתפים, זמן אמת, גורמים למשחק להיות מורכב, ובהתאם פיתוח סוכן AI יהיה מסובך עבורם.



איור 5: דיאגרמה של מגוון משחקי וידאו – דו ממדיים, תלת ממדיים, בעלי סוכן אחד ובעלי מספר סוכנים [6]

כדי לאמן סוכן AI במשחקי וידאו, יש לבצע את השלבים הבאים:

1. הגדרת המטרה של הסוכן - המטרה יכולה להיות לנצח את המשחק, להשיג ציון גבוה או להשלים משימה מסוימת.
2. הגדרת ההצלחה - מדידת ההצלחה יכולה להיות כמות הנקודות שהסוכן צבר, מספר האויבים שהסוכן הביס וכו'.
3. הגדרת הסביבה - הגדרת הדרך שבה הסוכן מתקשר עם המשחק, כולל המידע שהוא יכול לגשת אליו, הפעולות שהוא יכול לבצע, והתגמולים שהסוכן יכול לקבל.
4. בחירת אלגוריתם RL – הבחירה תלויה במאפיינים של המשחק ובמטרה של הסוכן.
5. אימון הסוכן – נותנים לסוכן לשחק במשחק נגדו עצמו או נגד גרסאות קודמות שלו. עם הזמן, הסוכן לומד לבצע את הפעולות שימקסמו את התגמול שלו.

ישנן הרבה בעיות שצריך להתמודד איתן על מנת לפתח סוכני AI מוצלחים במשחקים. משחקי וידאו הם לעיתים קרובות מורכבים מאוד, וקשה לאסוף מספיק נתונים כדי ללמד סוכן AI. הבעיות העיקריות הן:

1. מספר מצבי המשחק הוא גדול מאוד, במיוחד במשחקי אסטרטגיה.
2. קשה לפתח אסטרטגיות קבועות לקבלת החלטות בסביבה דינמית ולא ידועה.
3. רוב המשחקים פותחו בסביבה וירטואלית מוגדרת ולכן קשה להעביר יכולות של סוכני AI בין משחקים שונים.

כדי להתגבר על בעיות אלו, בנוסף לאלגוריתמים של RL, משתמשים גם בטכניקות ואלגוריתמים של למידה עמוקה הכוללות רשתות נוירונים, LSTM, Transfer Learning, ועוד. שילוב תחום הלמידה העמוקה עם תחום ה-RL יצר תת תחום חדש הנקרא "Deep Reinforcement Learning", הממנף את שני התחומים הנ"ל על מנת להתמודד עם המשימות המורכבות.

3 רשתות נוירונים מלאכותיות

אחד התחומים המדוברים בשנים האחרונות הוא בינה מלאכותית (AI) אשר משתמשים בו בתחומים רבים. AI הוא תחום של מדעי המחשב המתפתח במהירות, והוא מתמקד ביצירת מכונות ותוכנות המסוגלות לבצע משימות הדורשות בדרך כלל אינטליגנציה אנושית. מערכות AI נועדו ללמוד מנתונים, לקבל החלטות, ולפתור בעיות. הגדרה רחבה יותר לתחום זה ניתנה על ידי מרווין מינסקי: "לגרום למכונה להתנהג בדרך שהייתה נחשבת לאיניליגנטית לו אדם התנהג כך". המטרה לטווח ארוך של AI היא לפתור אתגרים מתקדמים הקיימים בעולם האמיתי. ככל שהתחום מתפתח ומתקדם, כך הולך הפוטנציאל לעצב מחדש תעשיות ולשפר בהיבטים שונים את חיי היומיום שלנו.

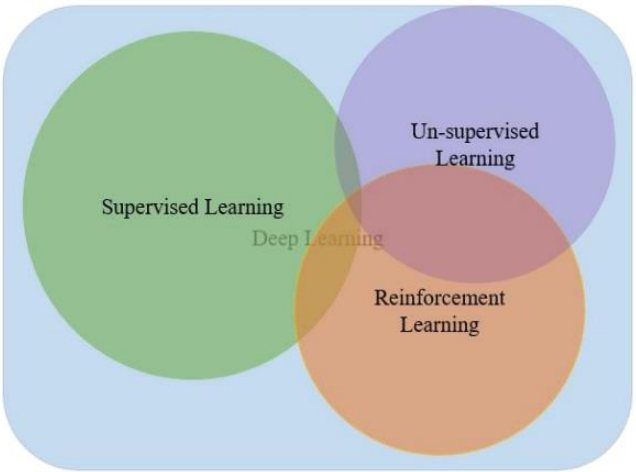
3.1 מבוא ללמידה עמוקה

למידה עמוקה (Deep Learning) היא תת תחום חדשני של למידת מכונה (Machine Learning) שחולל מהפכה בדרך שבה מכונות לומדות ומקבלות החלטות. היא שואבת השראה מהמבנה והתפקוד של המוח האנושי, תוך שימוש ברשתות נוירונים (רשתות עצביות מלאכותיות) כדי לעבד כמויות גדולות של נתונים ולחלץ תכונות משמעותיות מהקלט. טכנולוגיה זו עוזרת למגוון רחב של יישומים כגון עיבוד תמונה, עיבודה שפה טבעית, ראייה ממוחשבת ועוד.

המטרה העיקרית של למידת מכונה היא היכולת להכליל מתוך הניסיון, ולבצע משימות באופן מדויק ככל הניתן על נתונים חדשים.

תחום למידת המכונה מחולק לשלושה סוגים של למידה [9] :

- למידה מפוקחת (supervised learning) – גישה בה מאמנים מודל על אוסף של דוגמאות כך שלכל דוגמא יש תווית (label). בתהליך האימון, האלגוריתם לומד למפות נתוני קלט לפלט הנכון על ידי הכללת דפוסים מדוגמאות נתוני האימון. מטרת אלגוריתם זה היא לסווג דוגמאות חדשות שלא נצפו בתהליך הלמידה.
- למידה לא מונחית (Unsupervised learning) – גישה בה מאמנים מודל על דוגמאות ללא תווית, הנחה ספציפית או תוצאות מוגדרות מראש. מטרתה היא ללמוד דפוסים, מבנים או קשרים נסתרים בתוך הנתונים. בדרך כלל האלגוריתמים של גישה זו ימצאו מודל המסביר את התפלגות הנקודות, למשל חלוקה לקבוצות שונות.
- למידה באמצעות חיזוקים (Reinforcement Learning) שעליה פירטנו בפרק הקודם.



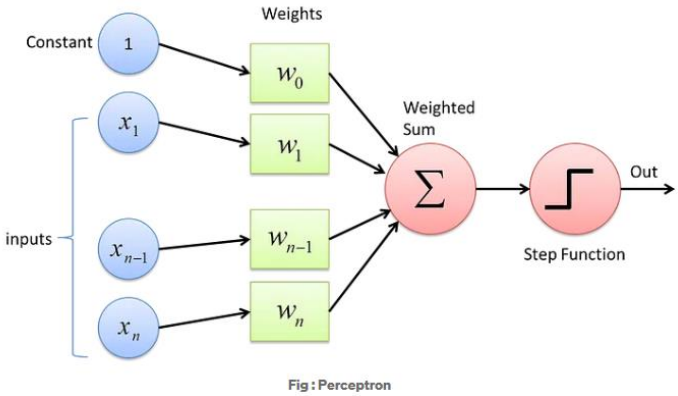
איור 6: הקטגוריות של גישות הלמידה עמוקה [9]

3.2 מבנה רשת נוירונים מלאכותית

רשת נוירונים היא מערכת חישובית המבוססת על המבנה והתפקוד של מערכת העצבים במוח האנושי. רשת הנוירונים מורכבת ממספר גדול של נוירונים מלאכותיים, המחוברים זה לזה באמצעות קשרים סינפטיים. הנוירון המלאכותי הוא ניסיון ליצור מערכת ממוחשבת היכולה לחקות את תפקודו של נוירון אנושי. המוח האנושי מורכב ממאות מיליארדי תאי עצב, והם אחראים על התפקודים הקוגניטיביים, הרגשיים והמוטוריים של האדם.

איור 7 מציג את הפרספטרון, כאשר $(x_1,x_2,\ldots,x_n)$ הוא וקטור הקלט שרכיביו עם ערכי הקלט. הנוירון מתאפיין בוקטור משקולות $(w_1,w_2,\ldots,w_n)$ שמשקף את "עוצמת" הקשר או "חוזק הסינפסות" על כל אחד מ-n הקלטים, בהתאמה. הנוירון סוכם את ערכי הקלט, תוך התחשבות בערך המשקולות $\Sigma_{i=1}^{n} w_i x_i$ ומחליט האם ערך זה גבוה או נמוך מערך סף כלשהו θ. כאשר $\Sigma_{i=1}^{n} w_i x_i \geq \theta$ הפלט של הנוירון הוא 1, וכאשר $\Sigma_{i=1}^{n} w_i x_i < \theta$ הפלט הוא 1-. למעשה, הנוירון מסווג את הקלט $(x_1,x_2,\ldots,x_n)$ לאחת משתי מחלקות C_1 או C_2 בהתאם לפלט.

ניתן להגדיר שבמקום לבדוק אם סכום המשקולות גדול מערך סף כלשהו, פשוט להעביר את הפלט דרך פונקציית אקטיבציה מסוימת (לדוגמא $ReLU=\max(0,x)$) וכך נוצר פלט הנוירון. בחירת פונקציית האקטיבציה היא חלק חשוב בהבנת רשתות נוירונים ובארכיטקטורות יותר מתקדמות, לכן אפרט על נושא זה בהמשך.

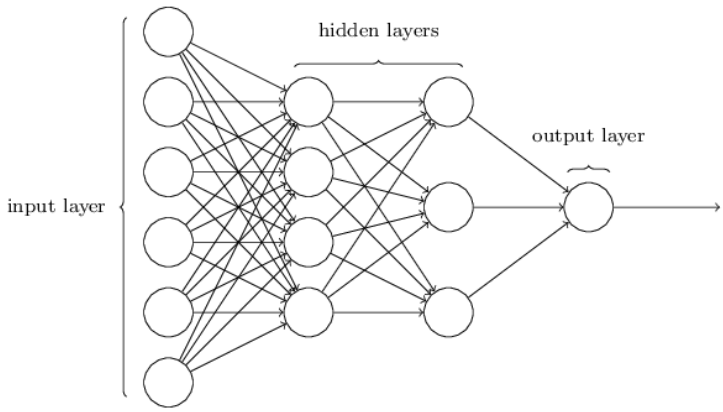


איור 7: מודל הפרספטרון [11]

הפרספטרון יכול לסווג שתי מחלקות אם ניתן להפרידן באופן לינארי, אך אינו יכול לסווג מחלקות שלא ניתנות להפרידן באופן לינארי. הפתרון לבעיה זו היא לחבר לשכבה הקלט שכבות חבויות (hidden layers), ואותן לחבר אותה לשכבת הפלט. כאשר הרשת מורכת מלפחות שכבה חבויה אחת, היא נקראת רשת נוירונים עמוקה. רשת נוירונים עמוקה מורכבת מ-3 סוגי שכבות של נוירונים :

- שכבת הקלט – השכבה הראשונה ברשת הנוירונים. היא מקבלת את המידע הגולמי, שבדרך כלל מכיל מספרים המייצגים אובייקט כלשהו, לדוגמה טקסט, תמונות, סרטונים.

- שכבת הפלט – השכבה האחרונה ברשת הנוירונים. היא מייצרת את התוצאה הסופית של הרשת, ובדרך כלל היא מייצגת תוצאה של הסתברות מסוימת.
- שכבות חבויות – השכבות החבויות נמצאות בין שכבת הקלט לשכבת הפלט. בשכבות אלו מתבצע חישובים מתמטיים הקשורים לעיבוד המידע שהגיע משכבת הקלט, בעזרת משקלי הקשתות ופונקציית האקטיבציה של כל נוירון. מספר השכבות החבויות והגודל שלהן תלוי בבעיה שהרשת נועדה לפתור.



איור 7: רשת נוירונים עם שכבת קלט, שכבת פלט ושתי שכבות חבויות [10]

המטרה של כל שכבה היא ללמוד ייצוג פשוט יותר של המידע שנכנס אליה, כך שבסוף יהיה ניתן להבחין בין קטגוריות שונות בעזרת הפרדה לינארית. בעוד שהפעולות אותן מבצעים הנוירונים קבועות (סכימה ולאחר מכן פונקציית הפעלה), משקלי וקטורי המשקולות נקבעות בהתחלה באופן אקראי, ובעזרת הדוגמאות ניתן לאמן את הרשת ולשנות את המשקלים כך שיבצעו את למידת הייצוג החדש בצורה אופטימלית.

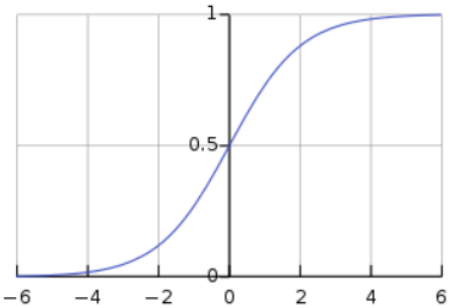
3.3 פונקציות אקטיבציה

אחד המרכיבים העיקריים בכל נוירון מלאכותי הוא פונקציית האקטיבציה. לכל פונקציה יש יתרונות וחסרונות, ולכן צריך לדעת להשתמש בהן בתבונה. משתמשים בהן גם בארכיטקטורות מתקדמות כמו LSTM. דוגמא ל-3 פונקציות נפוצות:

Sigmoid: $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$

יתרונות:

- טווח ערכים: הפונקציה ממפה את הקלט לפלט בטווח ערכים בין 0 ל-1, תכונה חשובה עבור סיווג.
- נגזרת קלה: לפונקציה נגזרת קלה, מה שמקל על שימוש בשיטות למידה מונחות כמו פעפוע לאחור.

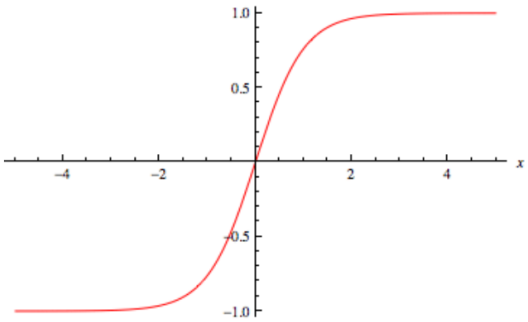


איור 8: פונקציית ה-Sigmoid.

חסרונות:

- גרדינאט נעלם: עבור ערכים גדולים, הנגזרת שואפת ל-0 ולכן בתהליך הפעפוע לאחור הגרדיאנט עלול להתאפס.
- לא מרוכז סביב ה-0 כלומר היא תמיד חיובית, וזה יכול לגרום לבעיות במהלך האימון.

TanH: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



איור 9: פונקציית ה- TanH.

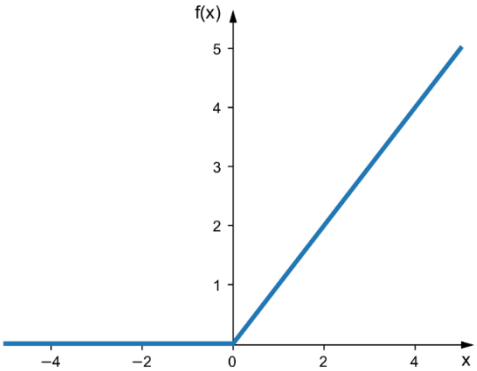
יתרונות:

- ממורכז סביב ה-0

חסרונות:

- גרדינאט נעלם: בדומה ל-sigmoid, הגרדיאנט עלול להתאפס בתהליך הפעפוע לאחור.
- חישוב האקספוננט בזבזני מבחינת יעילות.

ReLU: $f(x) = \max(0, x)$



איור 10: פונקציית ה-ReLU.

יתרונות:

- לא לינארי: הפונקציה לא לינארית ולכן היא יכולה ללמוד תבניות מסובכות.
- יכול להתמודד עם בעיית הגרדיאנט הנעלם.

חסרונות:

- לא ממורכז סביב ה-0
- עבור ערכים שלילים הגרדיאנט הוא 0, וכתוצאה מכך חלק מנוירונים לא תורמים לרשת.

3.4 Gradient Descent

בכל בעיה של למידה מפוקחת ניתן לאמוד את איכות הפתרון שמגיעים אליו באמצעות חישוב שגיאת האימון ε_t (או שגיאה ההכללה). שגיאת האימון היא פונקציה של וקטור הפרמטרים של המערכת הלומדת $(w_1,w_2,\ldots,w_p)$, המשקפים למשל את חוזק הסינפסות במקרה של רשת נוירונים. מטרת אלגוריתם הלמידה היא למצוא את אוסף הערכים $(w_1,w_2,\ldots,w_p)$ כך ששגיאת האימון $\varepsilon_t(w_1,w_2,\ldots,w_p)$ היא מינימלית.

ניתן לחשב את שגיאת האימון בעזרת פונקציית מחיר (Cost Function), המחשבת עד כמה וקטור הפלט קרוב לפלט המצופה. ככל שערך פונקציית שגיאת האימון יהיה נמוך יותר, כך הפלט של רשת נוירונים יהיה מדויק יותר. או במילים אחרות, נרצה למצוא מינימום גלובלי לפונקציה כך שתחזיר את הערך המדויק ביותר. כדי להשיג מטרה זו, לרוב משתמשים באלגוריתם הנקרא מורד הגרדיאנט (Gradient Descent).

מורד הגרדיאנט הוא אלגוריתם אופטימיזציה המשמש במהלך אימון מודל של למידת מכונה. מטרתו היא למצוא את המינימום של פונקציית ההפסד (loss function). הוא פועל באופן איטרטיבי כך שבכל פעם הוא ממזער את ערך השגיאה. הוא בא לידי ביטוי בשלב ה- back-propagation בכך שהוא מעדכן את משקלי הקשתות.

אלגוריתם מורד הגרדיאנט מתואר כך :

1. חשב את $\vec{\nabla}\varepsilon_t$, כלומר את $\left(\frac{\partial\varepsilon_t}{\partial w_1},\frac{\partial\varepsilon_t}{\partial w_2},\ldots,\frac{\partial\varepsilon_t}{\partial w_k}\right)$, בנקודה $(w_1,w_2,\ldots,w_k)$ ועדכן את הנקודה באופן הבא:

$$
\begin{aligned}
w_1^{\text{new}} &= w_1 - \eta\cdot\frac{\partial\varepsilon_t}{\partial w_1}\\
w_2^{\text{new}} &= w_2 - \eta\cdot\frac{\partial\varepsilon_t}{\partial w_2}\\
&\vdots\\
w_k^{\text{new}} &= w_k - \eta\cdot\frac{\partial\varepsilon_t}{\partial w_k}
\end{aligned}
$$

כאשר η הוא קבוע קטן כלשהו.

או בכתיב וקטורי:

$\vec{w}^{\text{new}} = \vec{w} - \eta\cdot\vec{\nabla}\varepsilon_t$, כלומר $\Delta\vec{w} = -\eta\cdot\vec{\nabla}\varepsilon_t$

2. חזור לסעיף 1 עבור הנקודה החדשה $\left(w_1^{\text{new}},w_2^{\text{new}},\ldots,w_k^{\text{new}}\right)$ או עצור אם יש התכנסנות, כלומר עד שהשינוי ב־ $\vec{w}$ קטן מסף כלשהו שקבענו.

כדי להמחיש זאת, ניתן לדמיין בן אדם שאינו יכול לראות, הנמצא על פסגת הר, ומנסה לרדת לתחתית בכמה שפחות צעדים. הגישה לפתירת בעיה זו היא כך שבכל צעד צריך למצוא איפה המדרון התלול ביותר, וללכת בעקבותיו. כדי לא להיתקע במינימום מקומי, משתמשים בשיטות אופטימיזציה שונות כמו מומנטום ועדכון קצב הלמידה.

איור 11: דוגמה הממחישה את אלגוריתם מורד הגרדיאנט בעזרת בן אדם היורד מפסגת הר [1]

3.5 אימון רשתות נוירונים

תהליך אימון רשתות נוירונים הוא תהליך חוזרני בו מכניסים לרשת דגימה או מספר דגימות יחד (זה נקרא batch) ולאחר כל איטרציה מעדכנים את משקלי הקשתות. תהליך זה נקרא backpropagation והוא מורכב משני שלבים עיקריים:

1. Forward propagation - הכנסת דוגמה ידועה לקלט הרשת ומעבירים אותה קדימה דרך השכבות החבויות עד לשכבת הפלט. לאחר מכן מחשבים את שגיאת האימון או את שגיאת ההכללה בעזרת פונקציית מחיר (Cost Function), המשווה את תוצאת פלט הרשת ביחס למה שאמור להיות הפלט האמיתי ומחשבים את ההפרש. למשל, אפשר להגדיר את פונקציית המחיר כשגיאה ריבועית ממוצעת $(\hat{y}-y)^2$.
2. Backward propagation – בעזרת אלגוריתם מורד הגרדיאנט מעדכנים את המשקלים של הרשת. מטרת שלב זה היא לתקן את המשקלים בהתאם למה שהתקבל בפלט, כאשר החישובים מבוצעים בעזרת כלל השרשרת.

באופן פורמלי, אלגוריתם ה- backpropagation הוא:

1. בהתקבל דוגמת אימון $(S_1^0, S_2^0, \ldots, S_{N^0}^0, y_0)$ חשב את ערכי כל הנוירונים בכל השכבות ברשת.
2. חשב את ערכי כל סיגנלי השגיאה ברשת החל בשכבת הפלט, לפי הנוסחה הרקורסיבית $\delta_i^l = g'(h_i^l) \cdot \sum_{k=1}^{N_{l+1}} \delta_k^{l+1} \cdot w_{ki}^{l+1}$ עבור $l = 1,2,\ldots,m-1$ כאשר $\delta_1^m = g'(h_1^m) \cdot (y_0 - S_1^m)$.
3. עדכן את w_{ij}^l:

$$\Delta w_{ij}^l = \eta \cdot S_j^{l-1} \cdot \delta_i^l$$

לסיכום, אלגוריתם ה- backpropagation פותח את האפשרות ללמוד כל פונקציה שנרצה. עם זאת, יש לזכור את המגבלות של האלגוריתם. ראשית, האלגוריתם מתכנס למינימום מקומי ולכן תוצאות הסימולציה תלויות בתנאי ההתחלה, ולעתים נדרש לחזור ולהריץ את האלגוריתם מספר פעמים ולבחור מתוכן את הפתרון הטוב ביותר. שנית, אין דרך לקבוע "מתכון" לקביעת ארכיטקטורת רשת אופטימלית (מספר השכבות ומספר הנוירונים בכל שכבה) בהתאם לפונקציה שרוצים לקרב. לפיכך, כמו במקרה של תנאי ההתחלה, הפתרון הוא למעשה אמפירי – מנסים מספר ארכיטקטורות ובוחרים את הטובה מתוכן.

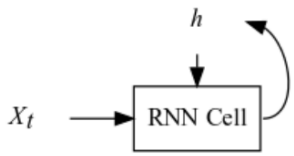
האלגוריתם שהוצג הוא כולל מרכיב בסיסי של העברת סינגל השגיאה בין השכבות. עד עכשיו לא נמצאה עדות שקיים מנגנון שכזה במוח, והאופן שבו מתבצע הלימוד ברשת רב שכבתית במוח נותר בגדר תעלומה. עם זאת, היכולת של אלגוריתם backpropagation ללמוד יחסי קלט-פלט מורכבים הופכת אותו לשימושי במגוון בעיות בתחום של למידת מכונה.

4 רשתות נשנות - RNN

בפרק זה נעסוק ב- Recurrent Neural Network (RNN), שעליהן מבוססות הרשתות LSTM שהשתמשו בהם OpenAI Five. רשתות LSTM הינן מרכיב חשוב בארכיטקטורת המודל שבזכותו ניצחו סוכני ה-AI את אלופי העולם במשחק Dota 2.

ישנם סוגי נתונים בהם האיברים השונים יוצרים סדרה שיש לאיבריה חשיבות לסדר כמו למשל טקסט, או נתונים בעלי מימד זמן כמו שוק ההון. לרשת נוירונים רגילה אין זיכרון ולכן היא לא יכולה להתייחס לחשיבות סדר הנתונים הנכנסים לקלט שלה. לשם כך נועדו רשתות ה- RNN (Recurrent Neural Network), שהם בעצם מחלקה השייכת לרשתות נוירונים עמוקות, אשר יש להם שימוש במיוחד עבור משימות כמו עיבוד שפה טבעית, ניתוח סדרות זמן, זיהוי דיבור ועוד. רשתות אלו הן בעלי יכולת לזכור מצבים ולפיהם לחזות את הפלט הרצוי.

רכיב הרשת אשר מאפשר לקחת בחשבון את סדר הנתונים נקרא תא הרשת הנשנית (RNN cell), לפעמים קרוי גם תא אלמן (Elman cell) והוא נראה כך:



איור 12: תא RNN.

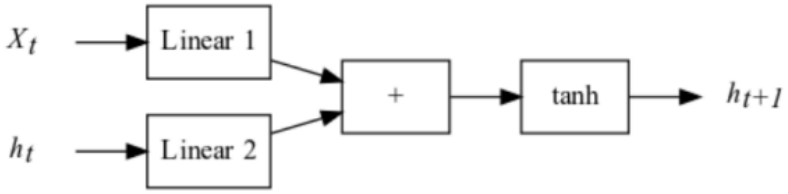
התא מורכב ממצב חבוי (hidden state) המסומן ב-h, ומשיכון של הטוקן t המסומן ב- X_t. הרעיון הוא שהפלט של תא הרשת יעבור לקלט של תא רשת אחר, בדומה לרעיון של רקורסיה. הוא עובד כך שבעת הזנת קלט X_t לתא, המצב החבוי הבא מחושב ונשמר לפי הנוסחה:

$h_{t+1} = \tanh\left(w_{input}X_t + b_{input} + W_{hidden}h_t + b_{hidden}\right)$ כאשר הפרמטרים

$w_{input}X_t + b_{input} + W_{hidden}h_t + b_{hidden}$ הם הפרמטרים הנלמדים של התא.

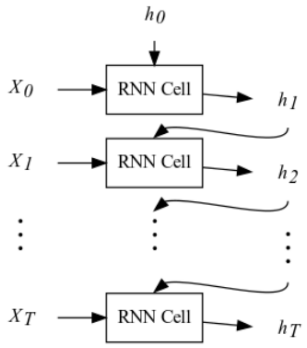
כדי להבין איך תא כזה עובד, נפרק את החישוב לשלבים:

1. הזנת טוקן הקלט X_t לאגרגציה לינארית בעלת הפרמטרים W_{input}, b_{input}.
2. הזנת המצב החבוי הקודם לאגרגציה לינארית אחרת בעלת הפרמטרים W_{hidden}, b_{hidden}.
3. סכימת פלט שכבות האגרגציה
4. הפעלת האקטיבציה על הסכום
5. שמירת המצב החבוי החדש



איור 13: חישוב הפלט של תא RNN.

ניתן לשרשר שכבות חבויות ולקבל רשת עמוקה, כאשר פלט שכבה מסוימת הופך להיות הקלט של השכבה הבאה:



איור 14 : דיאגרמת בלוקים של בלוק שיורי כללי

כמו כל רשת נוירונים, גם ל-RNN יש מספר מגבלות ובעיות, למשל:

1. **בעיית הגרדיאנט הנעלם** - בתהליך הלמידה של הרשת, הגרדיאנט עלול להתאפס. זה קורה בגלל ששגיאת הגרדיאנט קטנה כאשר מבצעים את תהליך ה- back propagating לאורך הרשת. במילים אחרות, רשת ה-RNN לא יכולה לזכור נתונים לטווח ארוך. זאת אחת הבעיות המרכזיות של רשתות RNN.
2. **Long-term dependency problem** – רשתות נוירונים, ובפרט רשתות RNN מתקשות ללמוד תבניות לאורך זמן כאשר ישנם פערים ארוכים בין התבניות.
3. **קושי באימון** - תהליך הלמידה הוא סדרתי ולכן לא ניתן להשתמש במקביליות. כתוצאה מכך, תהליך האימון יכול לקחת זמן ממושך.

ישנם כלים פשוטים הנועדו להתמודד עם הקושי המובנה שיוצרת התפשטות הגרדיאנט דרך רשת נשנית. חלקם מנסים לפתור את הבעיה באמצעות שינוי אלגוריתם האופטימיזציה, למשל בעזרת הקטנת קצת הלמידה כאשר הגרדיאנט גדול מדי.

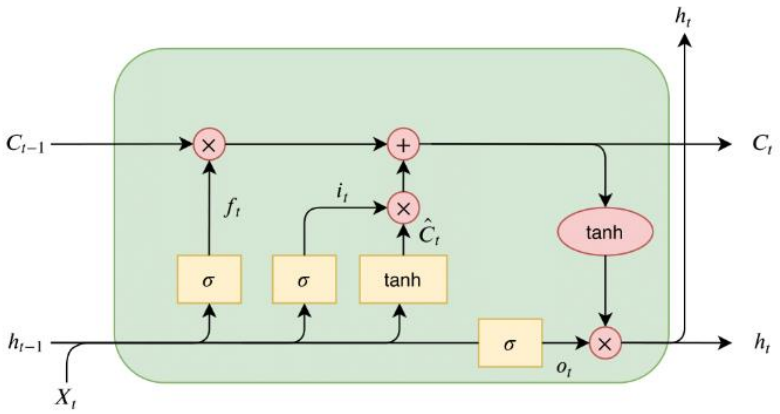
הכלים המוצלחים יותר פותרים את הבעיות באמצעות שינוי ארכיטקטורת הרשת: במקום תא אלמן פשוט, הרשת תשתמש בתאים נשנים מתקדמים, המכילים רכיבים דומים לחיבורי הדילוג של רשתות שיוריות. הנפוץ בהם הוא תא LSTM (Long Short-Term Memory), אשר מכיל גם רכיב בקרת זרימה המאפשר לתא ללמוד מתי לפתוח ומתי לסגור את חיבור הדילוג, וכן רכיב נלמד המאפשר לתא לאפס המצב החבוי, למשל לאחר שהוא זיהה נתק סמנטי בין שני חלקי משפט. בנוסף ל- LSTM, ישנה גרסה דומה הנקראת GRUs (Gated Recurrent Unit), ובשנים האחרונות החלו גם להשתמש במודלים של מנגנוני קשב (למשל Transformers) שהוכיחו את עצמם בביצועיהם במשימות הדומות למשימות של רשתות ה-RNN.

4.1 LSTM

כפי שראינו, אחת הבעיות המרכזיות של רשתות RNN היא בעיית הגרדיאנט הנעלם, שמונע מרשתות אלו להיות פרקטיות ושימושיות. אחד הפתרונות הנועדו להתגבר על בעיה זו, היא סוג ארכיטקטורת רשת RNN הנקראת LSTM (Long Short-Term Memory). הרשת מסוגלת ללמוד תבניות לאורך זמן, גם כאשר ישנם פערים ארוכים בין התבניות.

LSTM הוא כלי רב עוצמה וחשוב בתחום ה-RL במשחקי וידאו. הוא משמש כדי לשפר את הביצועים של סוכני AI, לתמוך בהחלטות אנושיות ולייצר משחקים חדשים. בתחום. הוא יכול לעזור בדרכים הבאות:

- **שמירה על מידע על נתונים קודמים** – משחקי וידאו הם לרוב סדרה של מצבים בהם שחקן צריך לקבל החלטות. LSTM יכול לשמור את המידע של מצביו הקודמים.
- **למידת תבניות לאורך זמן** – משחקי וידאו יכולים להיות מורכבים מאוד, עם תבניות שיכולות להשתנות לאורך זמן. LSTM יכול ללמוד תבניות לאורך זמן
- **תגובה מהירה לשינויים** – משחקי וידאו יכולים להיות דינמיים מאוד, עם שינויים שיכולים להתרחש במהירות. LSTM יכול להגיב לשינויים מהירים.



איור 15: ארכיטקטורת ה-LSTM [2]

LSTM פועל על ידי שימוש במנגנון זיכרון פנימי המאפשר לו לשמור מידע על נתונים קודמים. המנגנון מורכב מ-4 תאים, כאשר כל אחד מהם מייצג מצב זיכרון. בכל שלב, התאים מקבלים מידע מהקלט הנוכחי, ומעדכנים את מצבי הזיכרון שלהם בהתאם. ארבעת התאים הם [2]:

1. f_t (Forget Gate) – שער שכחה האחראי על מחיקת חלק מהזיכרון. הקלט שלו הם הקלט הנוכחי והמצב הנסתר הקודם והפלט הוא ערך בין 0 ל-1, כאשר הפירוש של 1 זה לשמור את הזיכרון ו-0 לשכוח את הזיכרון.
2. i_t (Input Gate) – שער הקלט האחראי על כמה יש לזכור את המידע החדש לטווח הארוך. שער הקלט שולט כמה מידע חדש מאוחסן במצב התא, ועוזר להחליט איזה מידע מהקלט הנוכחי יש להוסיף למצב התא.

3. $\hat{C}_t$ (Cell State) – שער האחראי על כמות הזיכרון המועברת לתא הבא.
4. o_t (Output Gate) – שער מוצא האחראי על כמה מהמידע רלוונטי לנתון הנוכחי X_t. הוא שולט באיזה מידע ממצב התא יש להשתמש כדי לחשב את המצב הנסתר הנוכחי h_t.

ארבעת השערים פועלים יחד כדי לשמור, לשנות ולנצל באופן סלקטיבי מידע מהקלט הנוכחי והמצב הנסתר הקודם. כך בעצם מקבלים גם את h_t האחראי על הזיכרון לטווח הקצר כמו ברשות RNN רגילות, וגם את c_t האחראי על זיכרון הארוך טווח. ארכיטקטורה זו מאפשרת ל-LSTM לזכור תלות ארוכת טווח בנתונים עוקבים. ניתן לנסח את פעולות התא כך:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, X_t] + b_f\right)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, X_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o)$$
$$\hat{C}_t = tanh(W_C \cdot [h_{t-1}, X_t] + b_C)$$

לאחר מכן, חישוב התא הפנימי מחושב כך:

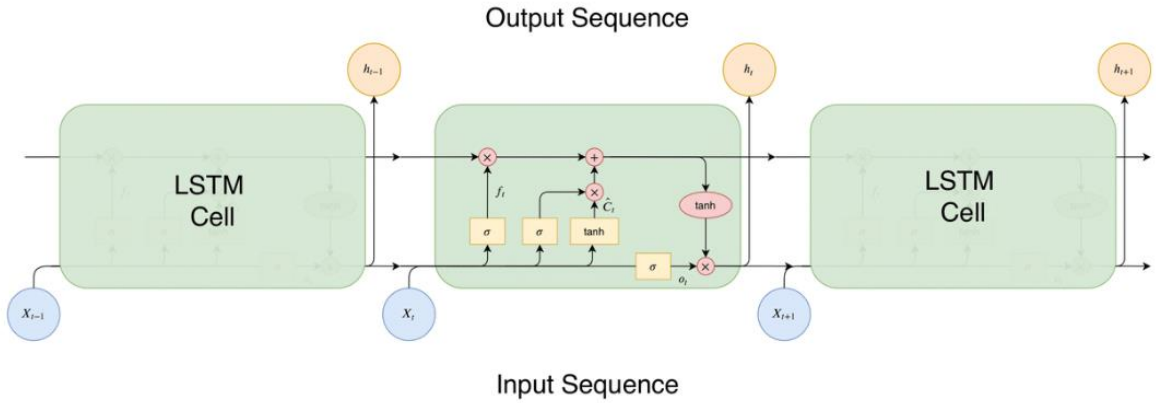
$$C_t = i_t \cdot \hat{C}_t + f_t \cdot C_{t-1}$$

ולבסוף, הפלט של התא (h_t) מחושב כך:

$$h_t = o_t \times \tanh\,(C_t)$$

הערה: σ הינו ביצוע פעולת פונקציית האקטיבציה sigmoid עליה דיברנו בפרק של רשתות נוירונים.

תאים אלה משורשרים יחד כדי ליצור רשת RNN-LSTM הפותרת את בעיית הגרדיאנט הנעלם:



איור 16: ארכיטקטורת RNN-LSTM [2]

LSTM זכתה לפופולריות רבה בשנים האחרונות, והיא משמש כיום במגוון רחב של יישומים. רוב התוצאות המרשימות של ה-RNN הושגו בעצם באמצעות LSTM. תחום הבינה המלאכותית הוא יחסית חדש ולכן סביר להניח שארכיטקטורה זו תשנה בעתיד, ויפתחו גרסאות מוצלחות יותר של רשתות RNN-LSTM.

5 המשחק Dota 2

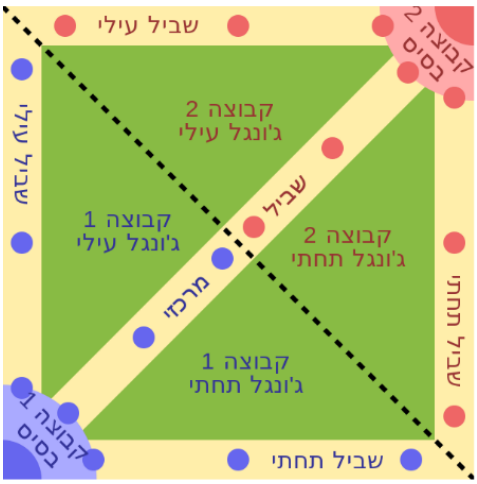
5.1 סקירת המשחק Dota 2

Dota 2 (Defense of the ancients) הוא משחק וידאו מבוסס אסטרטגיה מרובה משתתפים בזמן אמת שהושק על ידי החברה Valve Corporation בשנת 2013. הוא שייך לתת-סוג של משחקי אסטרטגיה בזמן אמת הנקרא זירת קרב מרובת משתתפים (MOBA - Multiplayer Online Battle Arena) בו כל שחקן שולט בדמות אחת מבין שתי קבוצות ומטרתו היא להרוס את המבנה העיקרי של הקבוצה היריבה. המשחק משוחק על פני מפה מרובעת על ידי שתי קבוצות בנות חמישה שחקנים כל אחת, אשר משחקות אחת נגד השנייה.

המטרה של כל קבוצה היא לאסוף זהב, לרכוש פריטים כדי לחזק את הגיבורים שלהם, ולבסוף להרוס הבסיס של הקבוצה היריבה. לצורך ביצוע משימה זו, השחקן נעזר בדמויות הנשלטות על ידי מחשב הצועדות ישר לאורך שביל קבוע, ובתכונות ייחודיות הקיימות בגיבור אותה בחר השחקן אשר ניתן לשפר לאורך הזמן. בתחילת המשחק, כל שחקן צריך לבחור גיבור מתוך מבחר של למעלה מ- 120 גיבורים ייחודיים, כאשר כל אחד עם היכולות והטכניקות שלו.

המשחק בעל עומק ורמת מורכבות גבוהה מאוד, הוא מדגיש תכנון אסטרטגי, עבודת צוות ומיומנות אישית. השחקנים צריכים ללמוד את חוקי המשחק ואת יכולות הגיבורים כדי להיות מסוגלים לשחק בצורה יעילה. למשחק עקומת למידה תלולה בגלל המכניקה המורכבת שלו, מה שהופך אותו למאתגר עבור שחקנים חדשים, ומתגמל עבור שחקנים ותיקים. המשחקים יכולים להימשך בין 30 דקות לשעה ולפעמים אף יותר, מה שהופך אותם לדינמיים ואינטנסיביים.

בהיבט התחרותי, המשחק כולל מספר רב של טורנירים בינלאומיים המציעים פרסים כספיים משמעותיים המושכים שחקנים מקצועיים מכל העולם. יש לו כנסים ואירועים רבים, ביניהם כנסי מעריצים המתחפשים לגיבורי המשחק. הוא נחשב כאבן יסוד בתעשיית הספורט האלקטרוני, המציג את המשיכה המתמשכת של שחקנים אל סוג משחקי ה- MOBA, מה שהופך אותו לאחד מהמשחקים המשוחקים ביותר בעולם.



איור 17: *מפת המשחק Dota 2. הקווים הצהובים מציינים את השבילים, בהם ממוקד עיקר המשחק. הנקודות הכחולות או האדומות מציינות את המגדלים אשר מגינים על השבילים. האזורים המודגשים בצבעים הם בסיס הקבוצות, והעיגולים הצבועים בפינות הם המבנה העיקר אותו הקבוצה היריבה נדרשת להרוס.*

5.2 הבינה המלאכותית OpenAI Five במשחק Dota 2

חוקרי RL (כולל חוקרי OpenAI) האמינו כי פיתוח סוכני AI במשחקים הדורשים אופקי זמן ארוכים ידרשו התקדמות חדשה מיסודה בשיטות המודרניות, כגון שיטה הנקראת למידת חיזוק היררכי. תוצאות המחקר ש-OpenAI הציגו מצביעות על כך שלא נתנו לאלגוריתמים של היום מספיק קרדיט – לפחות כשהם מופעלים בקנה מידה סביר.

משחקי MOBA עם סוכני AI הוא תחום מרתק ועם פוטנציאל עצום, אבל הוא מציב אתגרים רבים. כדי שבינה מלאכותית תוכל לשחק את המשחק Dota 2, היא צריכה לשלוט באופן מוחלט באתגרים הבאים:

- **אופקי זמן ארוכים** - משחקי Dota 2 משוחקים במהירות של 30 פריימים בשנייה במשך בערך 45 דקות. OpenAI Five בוחרת לבצע פעולה כל פריים רביעי, זאת אומרת בערך 20,000 צעדים בכל פרק. לשם השוואה, משחק שחמט נמשך בדרך כלל 80 מהלכים, והמשחק GO 150 מהלכים.
- **סביבה הנצפית חלקית** - כל קבוצה במשחק יכולה לראות רק את החלק של מצב המשחק ליד היחידות והמבנים שלהם, כאשר שאר המפה מוסתרת. משחק ברמה גבוהה דורש הסקת מסקנות המבוססות על נתונים לא מלאים, ובניית מודלים לפי התנהגות היריב.
- **מרחב פעולה ותצפית בממד גבוה** - המשחק משוחק על פני מפה גדולה המכילה עשרה גיבורים, עשרות בניינים, עשרות יצורים אשר נשלטים על ידי המשחק, ועוד אובייקטים שונים. OpenAI Five רואה בסביבות 16,000 ערכים (ערכים מספריים וערכים קטגורים) בכל צעד, ובממוצע בוחרת בכל צעד לבצע בין 8,000 ל- 80,000 פעולות. לשם השוואה, המשחק שחמט דורש 1,000 ערכים בכל צעד ויכול לבצע 35 מהלכים, המשחק GO דורש בערך 6,000 ערכים בכל צעד ויכול לבצע 250 פעולות.
- **זמן אמת** – כל הפעולות חייבות להתבצע בזמן אמת ולכן נדרשת יכולת חומרה ברמה גבוהה על מנת לבצע את כל החישובים של האלגוריתמים במהירות הדרושה.
- **עבודת צוות** – כל קבוצה במשחק מורכבת מ-5 שחקנים, וכדי לנצח במשחק צריך ששחקני הקבוצה ילמדו לשחק ביחד באופן מסונכרן אחד עם השני.

בנוסף לאתגרים אלו, גם חוקי המשחק מורכבים מאוד. המשחק פותח באופן פעיל למעלה מעשור, עם לוגיקת משחק המיושמת במאות אלפי שורות קוד. בנוסף, המשחק מקבל עדכון בערך כל שבועיים, ומשנה את הסמנטיקה של הסביבה.

למערכת של OpenAI Five יש 2 מגבלות:

1. מתוך כל הגיבורים, ניתן לשחק רק עם 17 מהגיבורים.
2. אין תמיכה בפריטים המאפשרים לשחקן לשלוט באופן זמני במספר יחידות בו זמנית כדי למנוע מורכבות טכנית המאפשרת לסוכן לשלוט במספר פריטים.

5.3 מרחב סביבת המשחק Dota 2 והאינטראקציה עם בינה מלאכותית

במקום להשתמש בפיקסלים שמוצגים על מסך המחשב, המערכת של OpenAI Five מעבדת את המידע הנתון לשחקן באמצעות מערכי נתונים, אשר מנורמלים לפני שהם מועברים לרשת הנוירונים [12]. המערכים הללו נקראים תצפית, כלומר כל המידע ששחקן אנושי רואה, כגון חיי היחידות, מיקום היחידות, הגיבורים וכו׳, והיא מועברת לרשת הנוירונים בכל צעד במהלך המשחק. הוחלט לעבוד כך מהסיבה שמטרת הסוכן היא ללמוד תכנון אסטרטגיה ומשחקיות לעומת התמקדות בעיבוד מידע חזותי. סיבה נוספת היא שלעבד כל פריים לפיקסלים יגרום להכפלת משאבי החישוב הנדרש.

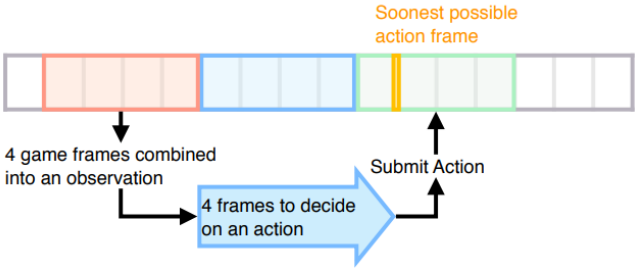
איור 18: דוגמא המציגה אך בן אדם רואה את המשחק לעומת בינה מלאכותית

עיבוד המידע בצורה זו אינו מושלם, כיוון שיש פיסות מידע קטנות אשר בני אדם יכולים לקבל גישה אליהם והמערכת לא קודדה אותן. מצד שני, המערכת יכולה לצפות בכל צעד את המידע הנגיש, לעומת בני אדם שצריכים ללחוץ על תפריטים שונים כדי לקבל אליהם גישה. חלק ממפת המשחק נחשב כערפל והשחקן לא יכול לראות אותו, לכן חלקי מפה אלה לא מועברות בתצפית כדי שהמערכת לא תרמה.

השחקן הראשי של OpenAI Five במהלך המשחק נקרא controller [12]. תפקידו לתאם את פעולות השחקנים בצוות ולקבל החלטות אסטרטגיות. הוא עושה זאת על ידי שימוש בתצפית המשחק, שהיא בעצם מידע המתאר את מצב המשחק, מצב השחקנים בצוות ומצב היריבים.

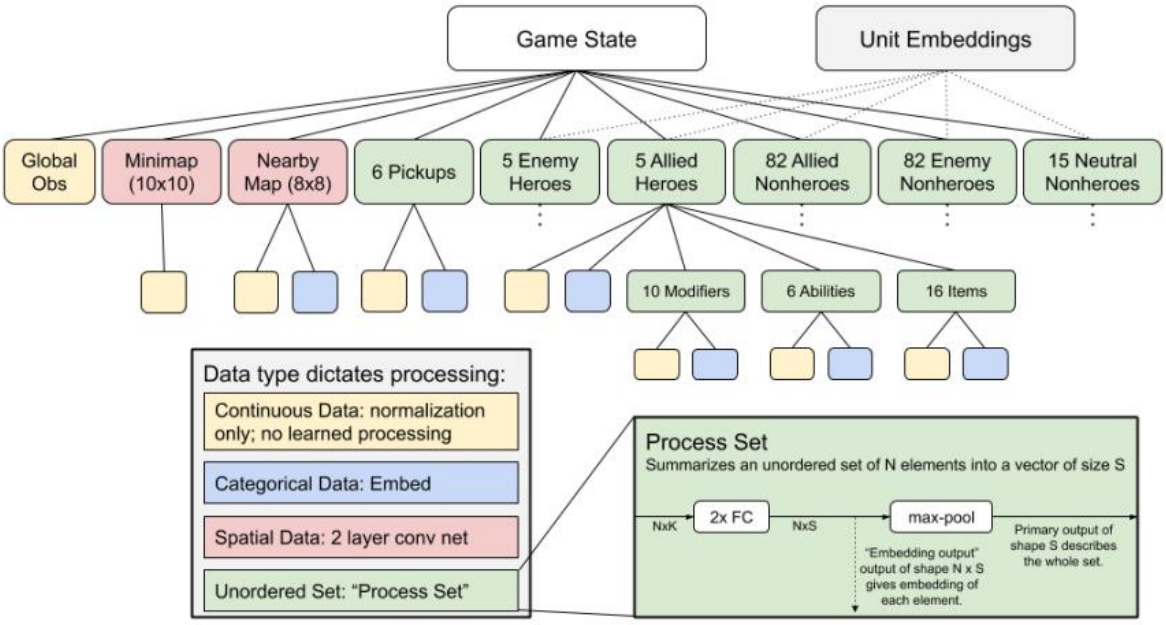
בעוד שהמשחק Dota 2 רץ במהירות של 30 פריימים בשנייה, OpenAI Five מבצע פעולה רק כל פריים רביעי. פעולה זו נקראת צעד זמן (timestep). בכל צעד זמן, ה-controller מקבל תצפית מהמשחק באמצעות API (Application Programming Interface) הנועד לשימוש בבוטים במשחק. לאחר מכן, הוא מקודד את התצפית, ובעזרת המדיניות הוא מחליט איזה פעולה לבצע. לבסוף, הוא מבצע את הפעולה לדוגמה תזוזה או תקיפה של השחקן. יש לציין שחלק מהפעולות נשלטות על ידי קוד מוכן מראש, משיקולים אסטרטגים. בנוסף, ה-controller אחראי על מספר פעולות מורכבות כמו :

- קבלת החלטת אסטרטגיות כגון היכן לשלוח את השחקנים, מתי לתקוף ומתי להגן.
- תיאום בין הפעולות של שחקני הצוות על מנת לוודא שכולם פועלים ביחד בצורה יעילה.
- ניהול משאבי הקבוצה כגון רכישת חפצים.



איור 19: אופן מהלך המשחק של OpenAI Five - קבלת תצפית, החלטה
על פעולה, ביצוע פעולה [12]

מבנה הנתונים של התצפית הוא עץ, כאשר כל צומת ועלה מייצגים תכונות מסוימות (לדוגמא יכולות הגיבור, החפצים שיש לגיבור וכו'). כל צומת בעץ מעובד בצורה מסוימת בהתאם לסוג המידע שיש בו. תהליך עיבוד הנתונים נקרא embedding, בו מייצגים נתונים קטגורים או מספריים ברצף של מספרים כאשר כל רצף של מספרים מייצג משמעות כלשהי.



איור 20: שיטוח מרחב התצפית לוקטור בודד [12]

לכל סוג נתונים קיים תהליך עיבוד נתונים משלו. חלק מהשיטות שבאמצעותם מתבצעות תהליכי ה-embedding לקוחות מסוג ארכיטקטורות רשתות נוירונים הנקראות רשתות קונבלוציה. ברשתות אלו משתמשים בשכבות נוירונים הנקראות שכבות קונבולוציה ושכבות max-pooling, שתפקידן לחלץ תכונות חשובות. במסגרת סמינר זה לא נרחיב עליהן. לדוגמא עבור נתונים מרחביים, משרשרים את המידע המצוי בכל תא ואז מחברים אותו לרשת קונבולוציה דו שכבתית.

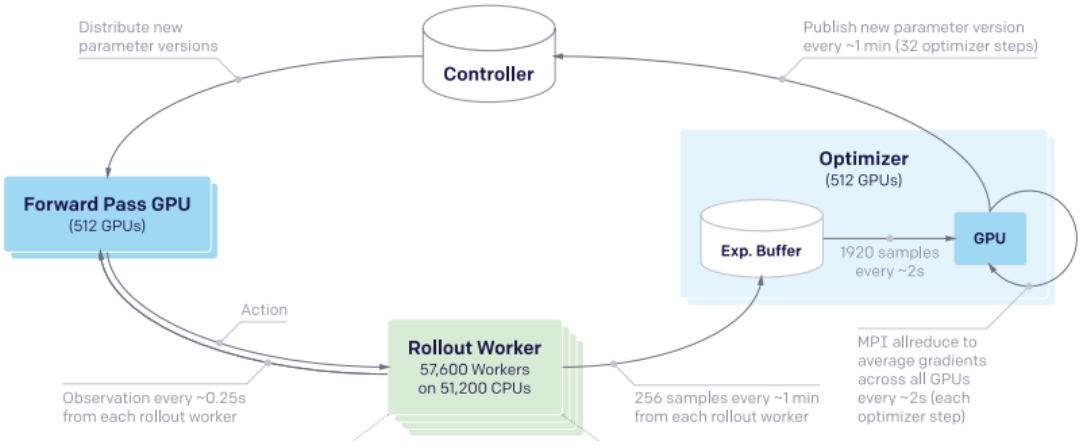
6 OpenAI Five - אלגוריתמים של בינה מלאכותית וטכניקות במשחק Dota 2

בנוסף לאלגוריתמים של RL ושל רשתות נוירונים, OpenAI Five השתמשו בטכניקות ושיטות שונות על מנת להשיג תוצאות טובות אף יותר. בפרק זה אציג חלק מהם.

6.1 תהליך אופטימיזציית המדיניות

מטרת חוקרי OpenAI היא למצוא מדיניות אשר ממקסמת את ההסתברות לנצח משחקים נגד שחקנים אנושים מקצוענים, או בפשטות, לנצח את המשחק. בפועל, הם מקסמו את פונקציית הרווח על ידי נתינת תגמול לסוכן עבור פעולות כמו צבירת משאבים, צבירת ניסיון, הריסת מבנים של האויב, ונתינת עונש לסוכן עבור פעולות כמו מוות הגיבור, או מוות של אחד מחברי הקבוצה. בגלל שלקבוצה יש 5 סוכנים, התגמולים והעונשים מחולקים לשני סוגים - תגמול קבוצתי ותגמול יחיד. תגמול קבוצתי הוא יותר גדול מתגמול יחיד כי הוא משפיע על כל הקבוצה, לדוגמא הריסת מגדל של האויב.

בתהליך האימון של הבינה המלאכותית OpenAI Five, השתמשו באלגוריתם PPO כדי למצוא את מדיניות הפעולה האופטימלית שמשמשת לשחק במשחק Dota 2 [12]. האלגוריתם משתמש בטכניקה הנקראת GAE (Generalized Advantage Estimation), שיטה המיועדת לייצוב והאצת האימון. GAE מאפשרת ל-PPO ללמוד מדיניות פעולה יותר על ידי הפחתת רעש במשוב, ולכן תפקידה חיוני.



איור 21: ארכיטקטורת מערכת האימון [12]

באופן כללי, תהליך אופטימזציית המדיניות עובד כך:

1. מערכות הנקראות "Rollout" משחקות את המשחק Dota 2 בעזרת ה-controller המפעיל את המודל העדכני שלו. המשחקים משוחקים באופן מקבילי, על מנת לזרז את תהליך האימון. כ-80% מהמשחקים משוחקים נגד המדיניות העדכנית, ו-20% נגד מדיניות ישנות.
2. נתוני המשחק המתקבלים מה- Rollout worker נשמרים באופן אסינכרוני במבני נתונים הנקראים experience buffers.

3. המערכת לוקחת את אוסף הנתונים של המשחקים העצמיים ומזינה אותם לבלוק LSTM המפיק את המדיניות. ה-LSTM לומד לחזות את התגמול הסופי של המשחק על סמך רצף הפעולות שבוצעו.
4. מעדכנים את המדיניות על ידי שימוש באלגוריתם PPO המנסה למצוא מדיניות חדשה שתניב תגמול גבוה יותר מהמדיניות הנוכחית, תוך שמירה על יציבות המדיניות.

שלבים אלו מבוצעים בעזרת אוסף של מאיצים גרפיים (Optimizer GPUs) כדי להאיץ את התהליך. GPUs הם רכיבי חומרה מיוחדים המיועדים לבצוע חישובים מתמטיים בצורה יעילה. בתהליך מורד הגרדיאנט כשמאמנים רשתות נוירונים, מתבצעים הרבה חישובים מתמטיים, ולכן שימוש ב-GPUs הכרחי על מנת לאמן מודל רשת נוירונים.

6.2 Surgery

ככל שהפרויקט של OpenAI Five התקדם, הקוד וסביבת המשחק השתנו בהדרגה בגלל 3 סיבות:

1. עם הניסיון והלמידה של החוקרים, הם ביצעו שינויים בתהליך האימון (מערכת התגמול, התצפיות ועוד), ובארכיטקטורת המדיניות של רשת הנוירונים.
2. עם הזמן, החוקרים הרחיבו את מערכת מכניקת המשחק הנתמכת על ידי פעולות סוכני ה-AI ומרחב התצפיות. הם עשו זאת בהדרגה כך שהתחילו עם מערכת פשוטה, ובכל פעם הוסיפו חלקים מורכבים.
3. לעיתים המשחק התעדכן בגרסאות חדשות כולל שינויים במכניקת המשחק, בתכונות הגיבורים, החפצים, המפות וכו'. כדי להתחרות בבני אדם, OpenAI Five חייבת לשחק בגרסא הכי עדכנית של המשחק.

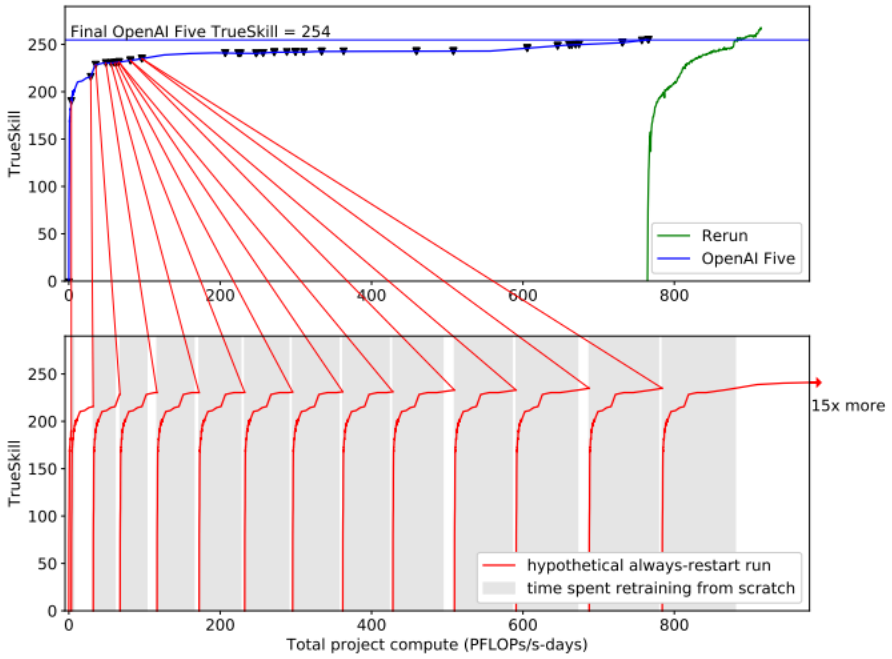
שינויים אלו שינו את הצורות והגדלים של שכבות המודל, ואת המשמעות הסמנטית של ערכי התצפיות. לאחר שהשינויים קורים, ברוב ההיבטים המודל הישן יהיה רלוונטי גם לסביבה החדשה, אולם לקיחת חלקים מהפרמטרים הישנים של וקטורי הנתונים ולהעבירם למודל החדש הוא תהליך מסובך ומוגבל. לפיכך, אימון המודל מהיסוד הוא הפתרון האידיאלי והבטוח.

עם זאת, אימון המודל של OpenAI Five הוא תהליך הלוקח מספר חודשים הכולל הוצאות כספיות גבוהות, מה שהניע את הצורך בשיטות שיכולות להתאים את המודל לסביבות חדשות תוך התחשבות בשינוי הפרמטרים.

שיטת ה-"ניתוח" (Surgery), היא בעצם אוסף של כלים לביצוע פעולות למודל הישן π_θ כדי להשיג מודל חדש $\hat{\pi}_{\hat{\theta}}$ המותאם לסביבה החדשה, שמתפקד באותה רמה גם אם לוקטורים $\hat{\theta}$ ו- θ יש גדלים שונים וסמנטיקה שונה. המטרה של השיטה היא להמשיך את אימון הסוכן ללא אובדן ביצועים.

המקרים בהם ביצעו "ניתוח" [12]:

1. **שינוי הארכיטקטורה** - ברוב המקרים, מרחב התצפית, מרחב הפעולות והסביבה לא השתנו. במצב זה ניתן לחשב את המדיניות החדשה בעזרת הנוסחה: $\forall o\ \hat{\pi}_{\hat{\theta}}(o) = \pi_{\theta}(o)$.
2. **שינוי מרחב התצפית** - חלק מה-"ניתוחים" בוצעו בגלל שמרחב התצפית השתנה, לדוגמא כאשר מוסיפים חיים ליצורים, משנים חפצים, וכו'. במקרה זה אי אפשר לעמוד על כך שהמדיניות החדשה תיישם אותה פונקציה ממרחב תצפית למרחב פעולה, שכן תחום הקלט השתנה.
3. **שינוי הסביבה או מרחב הפעולות** – כאשר גרסת המשחק מתעדכנת, אז הסביבה ומרחב הפעולות משתנים. בהתאם לכך, חוקרי AI הסיקו שאחרי כל שינוי, פשוט צריך לבצע שינוי על ה- "rollout workers" שיהיו יציבים יותר מכיוון שהמדיניות הישנה הצליחה לשחק היטב גם על הסביבה החדשה.



איור 22: אימון מודל בסביבה מתפתחת. בחלק העליון בגרף הכחול, ניתן לראות את התקדמות רמת הביצוע של הבינה המלאכותית, כאשר המשולשים מסמלים את התאריכים של בהם בוצע "ניתוח". לאחר מכן אימנו את המודל מחדש אחרי כל ה"ניתוחים" (מתואר בגרף הירוק). הגרף התחתון מראה מה היה קורה אם היו מאמנים את המודל מהיסוד אחרי כל שינוי במשחק [12]

על פי איור 92, כאשר החוקרים ביצעו Rerun (אימון מודל חדש אחרי שביצעו את כל הניתוחים) אז לא רק שזמן האימון פחת משמעותית, בנוסף הוא הגיע לרמת ביצועים גבוה מהסוכן הקודם.

הניתוחים אפשרו לחוקרים להוסיף פיצ'רים באופן קבוע בתהליך אימון המודל, וכך הם יכלו לבצע שינויים ובדיקות ולראות את התוצאות בהתאם, במקום לאמן את המודל מהתחלה. השיטה אפשרה להוספת פיצ'רים ושיפורים קטנים בהדרגה שבלעדיהם לא היה ניתן להגיע לרמת ביצועים אופטימלית.

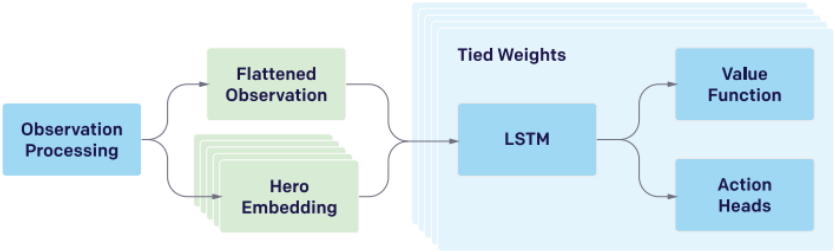
6.3 ארכיטקטורת רשת הנוירונים

מערכת הבינה המלאכותית OpenAI Five מבוססת בעיקר מרשתות LSTM חד-שכבתיות המורכבות מ-4096 יחידות, מ- combined policy ומ- value networks המשתמשת ב- 158,502,815 פרמטרים [12].

המערכת משתמשת ב-5 עותקים של רשתות נוירונים, כאשר כל אחת אחראית לקבל תצפיות ולבצע פעולות עבור כל גיבור בקבוצה (כיוון שבכל קבוצה יש 5 שחקנים). המשקולות של חמשת הרשתות זהות והתצפיות כמעט זהות, מלבד מספר פרמטרים המייצגים תכונות כמו המיקום של כל גיבור. ה-LSTM מרכיב 84% מהפרמטרים של המודל.

הרשת מורכבת משלושה חלקים:

1. ראשית, התצפית מעובדת ומקודדת לוקטור בודד המסכם את המצב והוא נקרא "Flattened Observation". בנוסף, היא מעבדת חמישה וקטורים המייצגים כל גיבור (Herro Embedding). וקטורים אלו מחוברים אל ה-LSTM.
2. וקטור התצפית וה- "Hero Embedding" מעובדים ביחד בעזרת fully-connected-layer ופעולת "cross-hero pool", וזה כדי לוודא שהתצפיות הלא זהות יכולות להיות משומשות על ידי חברי הקבוצה האחרים במקרה הצורך. לאחר תהליך זה, הוקטור מועבר דרך ה- LSTM.
3. הפלט של ה-LSTM, מוקרן תוך שימוש בהקרנה לינארית אל המודל Value Function ואל המודל Action Heads.



איור 23: הארכיטקטורה של OpenAI Five באופן פשוט [12]

הסבר על החיבורים של הפלט של ה-LSTM:

Value Function – מודל חישובי המחזיר את הערך של מצב מסוים במשחק. הערך של המצב הוא ההסתברות לנצח במשחק, תוך התחשבות בפעולות האפשריות של השחקן.

Action Heads – מודל חישובי המחזיר את הפעולה הטובה ביותר שניתן לבצע במצב מסוים. הפעולה הטובה ביותר היא הפעולה שתגדיל את הסיכוי לנצח במשחק.

זהו הסבר בסיסי על ארכיטקטורת רשת הנוירונים בה השתמשו OpenAI Five. כדי להבין כל שלב ולממש את הארכיטקטורה, נדרשת הבנה עמוקה וניסיון משמעותי בתחום.

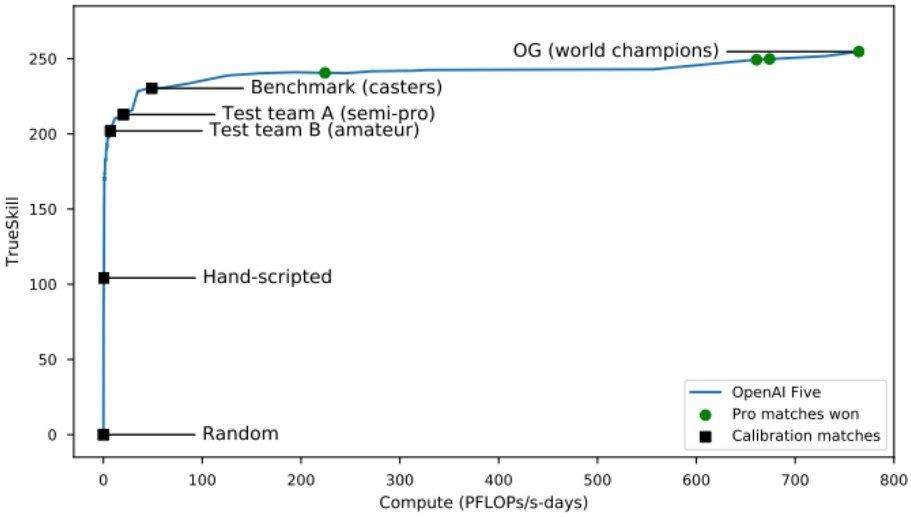
6.4 הערכת ביצועי OpenAI Five

כדי לבחון את ביצועי הבינה המלאכותית OpenAI Five ולבדוק האם ניתן להשתמש בה באופן עקבי במשחקים, בשנת 2019 החוקרים הקימו מיזם הנקרא "OpenAI Five Arena", המאפשר לאנשים לשחק במשחקים תחרותיים נגד הבינה המלאכותית [12]. בסך הכל שוחקו כ- 7,257 משחקים כאשר OpenAI Five ניצחה ב-99.4% מהם. זהו אכן הישג מרשים בהתחשב במורכבות המשחק, ובעובדה שהסוכן משחק עם 5 חברי צוות בקבוצה בו זמנית.

במשחק Dota 2, המדד המרכזי למיומנות אנושית הוא זמן תגובה. OpenAI Five יכולה להגיב לאירוע במשחק בזמן ממוצע של 217ms, לעומת הזמן הממוצע של אדם רגיל שהוא 250ms. זאת אולי אחת הסיבות המרכזיות שבזכותן הבינה המלאכותית הגיע להישגים אלו.

הערכת ביצועי הסוכן במשחקים היוותה מטרה חשובה, אך עם זאת החוקרים גם היו צריכים להעריך את הסוכן באופן מתמשך בזמן האימון. הם השיגו זאת על ידי מערכת דירוג הנקראת TrueSkill, המודדת את הביצועים של הסוכן בעזרת אלגוריתם מסוים. דירוג 0 מייצג התנהגות אקראית של הסוכן, לעומת דירוג 254, גרסת הסוכן שניצח באליפות העולם. בכל סיום משחק, האלגוריתם TrueSkill עדכן את המדד שלו.

ככל שהסוכנים השתפרו, סגנון המשחק שלהם התפתח כך שיתאים יותר למשחק עם בני אדם תוך שמירה על מאפיינים שהם למדו במהלך אימונם. הסוכן האחרון שיחק בדומה לבני אדם בתחומים רבים, אך היו לו מספר הבדלים מעניינים באופן בו הוא שיחק ובאסטרטגיה שלו. שחקנים אנושים נוטים להקצות גיבורים לאזורים שונים במפה בתחילת המשחק מקצים מחדש רק מדי פעם, לעומת OpenAI Five שמקצה גיבורים במפה בתדירות גבוהה. הבדל נוסף הוא ששחקנים אנושים נוהגים בזהירות כאשר לגיבור שלהם נותר מעט חיים, בעוד ש- OpenAI Five יודעת להחליט האם שווה את הסיכון לשחק בצורה אגרסיבית עם מעט חיים.



איור 24: מדד ה- TrueSkill לאורך תהליך אימון OpenAI Five [12]

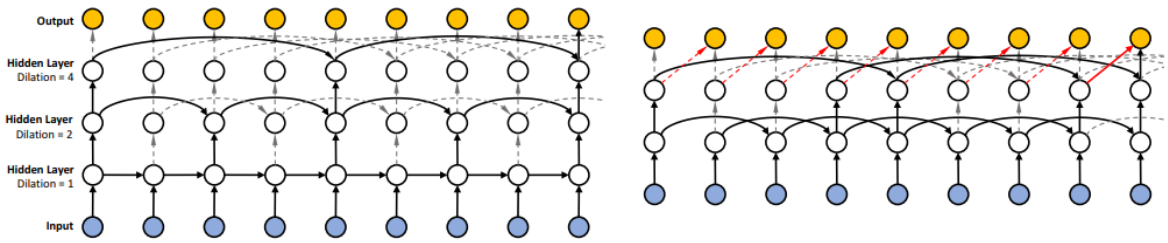
6.5 שיפורים עתידיים

כפי שראינו, חוקרי OpenAi הוכיחו שסוכני AI יכולים לשחק ברמה של בן אדם ברמה הכי גבוה ואף לנצח אותו. עם זאת, הסוכנים עדיין לא מושלמים ויש להם מגבלות שונות. לפיכך, נשאלת השאלה האם ניתן להשתמש בטכניקות נוספות על מנת לשפר את הביצועים של סוכנים ב-RL במשחקי וידאו, ובפרט במשחק Dota 2? התשובה היא כן, ואציג כמה טכניקות ורעיונות היכולות להועיל:

Transfer Learning: חוקרי OpenAI התחילו לאמן את הסוכן שלהם מאפס, כלומר ללא כל ניסיון קודם. בעתיד, יהיה ניתן להשתמש בטכניקה הנקראת Transfer Learning, המאפשרת לקחת מודל המאומן על סט נתונים מסוים, ולהתאים אותו למודל עם נתונים אחרים. טכניקה זו יכולה להועיל בהגדלת היכולות של הסוכנים ללמוד במהירות וביעילות משימות חדשות בכך שאפשר להשתמש בסוכנים שאומנו על משימות מסוימות כדי לאמן סוכנים חדשים בעלי משימות דומות. כך הסוכנים החדשים יהיו בעלי תשתית טובה יותר ללמוד את משימות חדשות.

Environment Modeling: שימוש באלגוריתמים לתחומים חדשים כמו משחקי לוח ומשחקי וידאו דורש הבנה מקצועית ומשאבים חישוביים כדי לממש את אותם. כדי לפתוח את תחום ה-**RL** למגוון רחב יותר של יישומים פרקטים מבלי הבנה מקצועית וכוח מחשוב, צריך להשתמש באלגוריתם כללי יותר. אחת הגישות היכולה לפתור בעיה זו הוא בעזרת אלגוריתם בשם DreamerV3, אלגוריתם כללי היכול להתרחב ולשלוט בתחומים רבים. בעזרתו ניתן ללמוד את מודל של הסביבה עצמה ולשפר את המדיניות כלפי המודל.

Hierarchical Reinforcement Learning: במשחקי וידאו כמו Dota 2 בהם הסביבה מורכבת ויש בהם הרבה אפשרויות פעולה שונות, קשה ללמד סוכן את משימותיו בעזרת מדיניות אחת. טכניקה היכולה להתמודד עם סביבה מורכבת היא טכניקה הנקראת למידת חיזוק היררכית (HRL), בה הסוכן לומד לפצל את המשימה המקורית למשימות משנה קטנות ופשוטות יותר [3]. לכל משימה יש מדיניות משלה, והסוכן לומד איך לעבור בין המשימות השונות בצורה יעילה. HRL כרוך בארגון תהליך הלמידה במספר רמות של הפשטה, המאפשר לסוכן ללמוד ולקבל החלטות ברמות גבוהות ונמוכות בו זמנית. בטכניקה זו משתמשים בארכיטקטורת רשת נוירונים הנקראת DiltatedRNN.



איור 25: דוגמא רשת DilatedRNN עם 3 שכבות [3]

טכניקות אלו הן רק חלק מהטכניקות מבין רבות היכולות לשפר את ביצועי סוכני ה-**AI** המודרניים, אך הן עדיין בתהליך מחקר ופיתוח ולכן עדיין קשה להשתמש בהם בפועל.

7 סיכום ומסקנות

בסמינר הוצגה הבינה המלאכותית OpenAI Five אשר הצליחה להגיע להישגים יוצאי דופן בתחום הבינה המלאכותית ובתחום ה-RL. זוהי דוגמא טובה להראות איך בינה מלאכותית מתמודדת בהשוואה לבני אדם בביצוע משימות מורכבות הכוללות אסטרטגיה ועבודת צוות.

עם זאת, החברה OpenAI לא מבזבזת את מרבית משאביה וכספיה רק בשביל לשחק משחקי וידאו. המטרה העיקרית של פרויקט המחקר של OpenAI היא לקחת חלקים של מערכת הבינה המלאכותית שהיא פיתחה, ולהשתמש בהם עבור משימות מורכבות מחוץ לתחום משחקי הוידאו, לדוגמא פתירת קובייה הונגרית עם יד רובוטית [13].

סוכן ה-AI של Dota 2 עדיין לא מושלם כיוון שהוא לא ניצח ב-100% מהמשחקים. בנוסף יש לו מספר מגבלות כמו העובדה שהוא יכול לשחק רק עם 17 גיבורים [12]. לפיכך, לתחום המחקר של שימוש בבינה מלאכותית במשחקי וידאו יש הרבה לאן להתקדם ולהתפתח. ישנם מאמרים חדשים בתחום המציגים שיטות וטכניקות מתקדמות המדגימות איך ניתן לשפר את הסוכנים אף יותר ולהסיר את מגבלותיהם.

בעתיד, הסביבות והמשימות ימשיכו לצמוח במורכבויות שלהן. השיטות והטכניקות שהוצגו בפרויקט יחולו באופן כללי יותר על התחום ויוכלו להתמודד גם עם סביבות יותר מורכבות הכוללות יותר פרמטרים. התרחבות תחום הבינה המלאכותית והטכנולוגיה הקשורה אליו יהיו חשובים יותר ויותר ככל שהמשימות יהיו יותר מאתגרות.

ביבליוגרפיה

[1] Crypto1, "How Does the Gradient Descent Algorithm Work in Machine Learning," 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/.

[2] T. M. Ingolfsson, "Insight into LSTM architecture," 2021. [Online]. Available: https://thorirmar.com/post/insight_into_lstm/.

[3] S. Chang, Y. Zhang, Wei Han, M. Yu, X. Guo, Tan Wei and T. Huang, "Dilated Recurrent Neural Networks," *arXiv:1710.02224v3 [cs.AI],* 2017. URL: https://arxiv.org/pdf/1710.02224.pdf.

[4] DeepMind, "Mastering Diverse Domains through World Models," *arXiv:2301.04104v1 [cs.AI],* 2023. URL: https://arxiv.org/pdf/2301.04104.pdf.

[5] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "A Brief Survey of Deep Reinforcement Learning," *arXiv:1708.05866v2 [cs.LG],* 2017. URL: https://discovery.ucl.ac.uk/id/eprint/10083557/1/1708.05866v2.pdf.

[6] K. Shao, Z. Tang, Y. Zhu, N. Li and D. Zhao, "A Survery of Deep Reinforcement Learning in Video Games," *arXiv:1912.10944v2 [cs.MA],* 2019. URL: https://arxiv.org/pdf/1912.10944.pdf.

[7] OpenAI, "OpenAI Spinning Up," 2018. [Online]. Available: https://spinningup.openai.com/en/latest.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and Oleg Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347v2 [cs.LG],* 2017. URL: https://arxiv.org/pdf/1707.06347.pdf.

[9] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin and K. V. Asari, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," *arXiv:1803.01164 [cs.CV],* 2018. URL: https://arxiv.org/ftp/arxiv/papers/1803/1803.01164.pdf.

[10] M. Nielsen, "Neural Networks and Deep Learning," 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[11] S. Sharma, "What the Hell is Perceptron," 2017. [Online]. Available: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53.

[12] OpenAI, "Dota 2 with Large Scale Deep Renforcement Learning," *arXiv: 1912.06680 [cs.LG],* 2019. URL: https://arxiv.org/pdf/1912.06680.pdf.

[13] OpenAI, "Solving Rubik's Cube With a Robot Hand," *arXiv:1910.07113v1 [cs.LG],* 2019. URL: https://arxiv.org/pdf/1910.07113.pdf.