

**דוח פרויקט:** סמסטריאלי

**הנושא:**

**solving Sudoku using machine learning and AI tools**

מגישים:

חן דודאי ברק ממיסטבלוב

מנחה:

רון דורפמן

תום יורגנסון

**סמסטר:** חורף

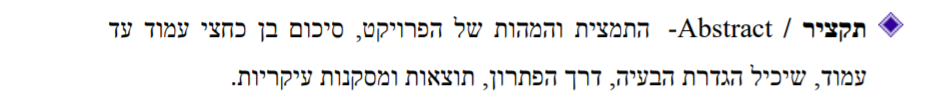
**שנה:** 2020

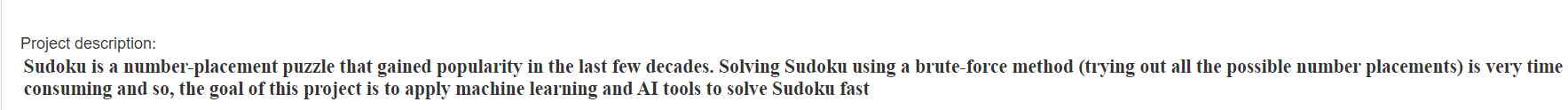
**תוכן עניינים**

**רשימת איורים**

**תקציר**

בשנים האחרונות משחק הסודוקו צבר תאוצה והפופולריות שלו בקרב הציבור הרחב גדלה משמעותית. בד בבד הפופולריות של פתרונות המבוססים על למידת מכונה צברו תאוצה גם כן. ניתן לפתור לוח סודוקו באמצעות שיטה ישירה (brute-force), על ידי ניסוי של כל הפתרונות האפשריים בזה אחר זה, תוך שימוש ב-backtracking. אולם, שיטה זו היא בעלת סיבוכיות חישוב גבוהה מאוד ולוקחת זמן רב לפתרון. בפרויקט זה עשינו שימוש בשיטות מבוססות למידה מכונה. מצד אחד שיטות מבוססות למידת מכינה פותרות מהר יותר בזמן מבחן, אך מצד שני הן דורשות סט אימון גדול. נוסו ארכיטקטורות deep learning שונות ושיטות אימון שונות, על מנת לפתור הבעיה באופן היעיל והטוב ביותר.

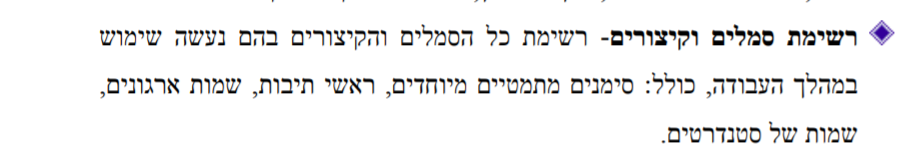




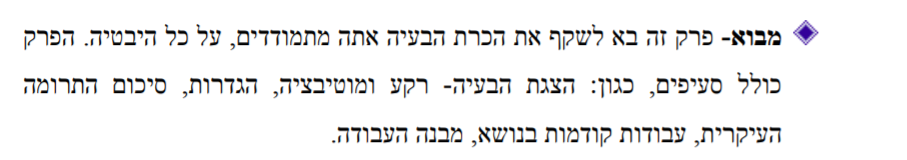
**Abstract**

The Sudoku game is a number placement puzzle that gained popularity in the newspapers and among the public in the last few years. Furthermore, machine learning algorithms has gain massive popularity also. One way to solve Sudoku automatically is with brute-force algorithm, and trying every possible solution using backtracking algorithm. The problem with this method is that it is very time consuming and has high complexity. In this project we solved this problem using deep learning algorithm, which on one hand is less time consuming in test time, but on the other hand demands big database of Sudoku puzzles. We applied different deep learning algorithms and different training methods to solve this problem the best and efficient way possible.

**רשימת סמלים וקיצורים**



1. **מבוא**



1. **סודוקו**

סודוקו הינו תשבץ מספרים שבו צריך למקם ספרות על לוח משובץ שגודלו 9X9, המורכב מ-9 ריבועים בני 9 משבצות כל אחד. פירושו המילה "סודוקו" ביפנית היא "מספר יחיד", משום שמטרת המשחק היא למקם 9 ספרות (הספרות 1 עד 9) על גבי לוח המשחק כך שבכל טור, בכל שורה, ובכל ריבוע, יופיע כל סמל בדיוק פעם אחת. בלוח המשחק נתונים כמה ספרות, ויש להתייחס אליהן בעת מיקום הספרות החדשות במהלך המשחק.

תשבץ הסודוקו זכה לפופולריות רבה ביפן בשנת 1986 ובבריטניה, בקנדה, ובישראל בשנת 2005 בעקבות קידומו בעיתונות. ישנם חוקים המייחסים לפתרון תשבצי סודוקו סגולות של שיפור או שימור כישורים שכליים.

פתרון סודוקו תקף הוא גם ריבוע לטיני. ריבוע לטיני הינו תחום אשר נחקר רבות, זהו ריבוע של n שורות ו-n עמודות, שבכל שורה ובכל עמודה שלו כתובים אותם n סמלים שונים, בלי חזרות.

. קיימות הרבה פחות אפשרויות לסודוקו מאשר ריבועים לטיניים, משום שפתרון סודוקו דורש אילוץ נוסף, אזורי (הבלוקים).

מספר האפשרויות לסודוקו של 9X9 הוא 6,670,903,752,021,072,936,960. *6.67×1021* בקירוב. תוצאה זו חושבה בשיטות קומבינטוריות שנעזרו בחישוב באמצעות מחשב.

המספר המקסימלי של נתונים שניתן לספק מבלי שהפתרון יקבע ביחידות הוא גודל הלוח עצמו פחות 4. המספר המינימלי של נתונים שיש לספק על מנת שהפתרון יקבע ביחידות הוא 17 נתונים.

### פתרון ממוחשב

הדרך הפשוטה ביותר לפתור בעיות סודוקו באמצעות מחשב היא על ידי מעבר [רקורסיבי](https://he.wikipedia.org/wiki/%D7%A8%D7%A7%D7%95%D7%A8%D7%A1%D7%99%D7%94) על כל האפשרויות, בשיטה הנקראת [גישוש נסוג](https://he.wikipedia.org/wiki/%D7%92%D7%99%D7%A9%D7%95%D7%A9_%D7%A0%D7%A1%D7%95%D7%92). ראשית, בוחרים את אחת הספרות שעדיין אינה מופיעה בלוח תשע פעמים. ממקמים את הספרה בשורה הראשונה באופן חוקי, ואז ממשיכים למקם אותה בשורות הבאות. לאחר סריקת כל השורות, עוברים למספר הבא שאינו מופיע באופן שלם, עד לפתרון הבעיה. אם האלגוריתם נכשל במציאת מקום חוקי באחת השורות, יש לחזור אחורנית ולנסות אפשרויות אחרות. שיטה זו דורשת [סיבוכיות זמן](https://he.wikipedia.org/wiki/%D7%A1%D7%99%D7%91%D7%95%D7%9B%D7%99%D7%95%D7%AA_%D7%96%D7%9E%D7%9F) גבוהה למדי, ואינה מספקת הערכות לרמת הקושי של הפתרון. על כן רוב הפתרונות הממוחשבים מתבססים על הפעלת שיטות פתרון מהסוג שהודגם לעיל, ופונים לשימוש ברקורסיה רק כאמצעי אחרון.

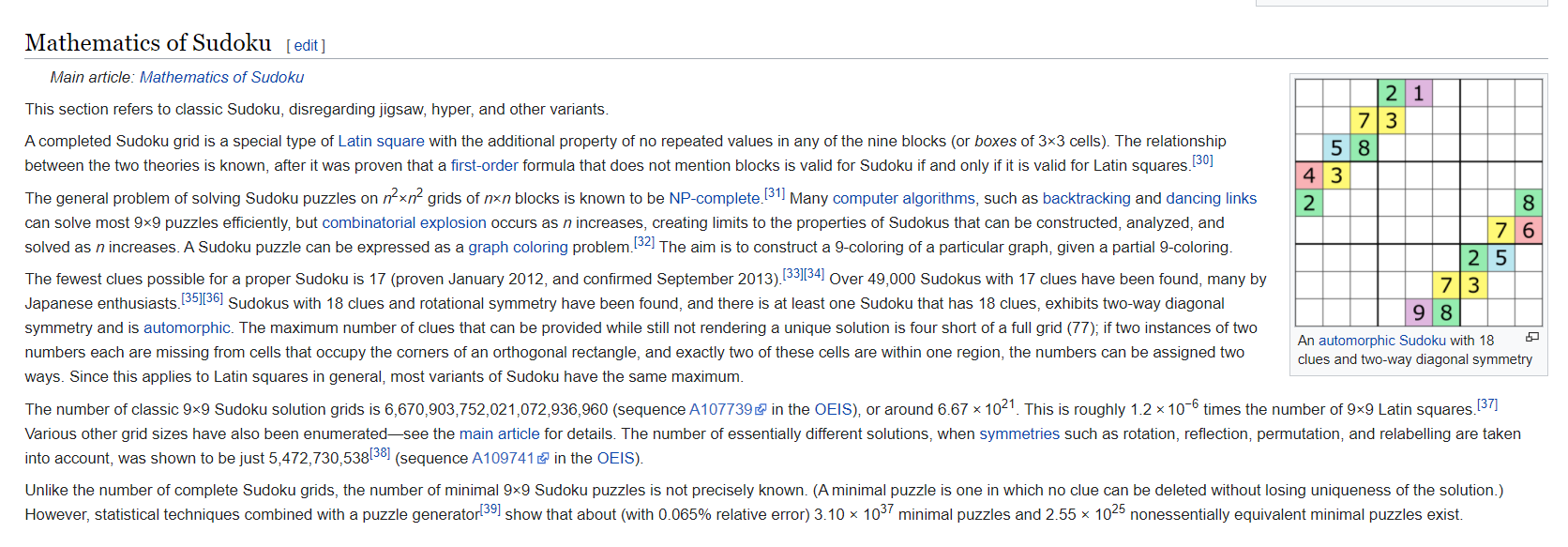
## סודוקו ומתמטיקה

### אלגוריתמים וסיבוכיות

בעיית פתירת הסודוקו ללוח בגודל משתנה ({\displaystyle n^{2}} שורות ועמודות ובלוקים בגודל {\displaystyle n\times n}) ידועה כבעיה [NP-שלמה](https://he.wikipedia.org/wiki/NP-%D7%A9%D7%9C%D7%9E%D7%94). כאמור, גישה אלגוריתמית מקובלת לבעיה היא באמצעות [גישוש נסוג](https://he.wikipedia.org/wiki/%D7%92%D7%99%D7%A9%D7%95%D7%A9_%D7%A0%D7%A1%D7%95%D7%92).

איור -סודקו



1. **Backtracking**

**גישוש נסוג** (ב[אנגלית](https://he.wikipedia.org/wiki/%D7%90%D7%A0%D7%92%D7%9C%D7%99%D7%AA): **Backtracking**) או **עקיבה לאחור** הוא סוג של [אלגוריתם](https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D) חיפוש שחוסך מעבר על מספר רב של מועמדים לפתרון על ידי שימוש בתכונות ספציפיות של הבעיה. שיטה זו יכולה לשמש לפתרון [בעיית סיפוק אילוצים](https://he.wikipedia.org/wiki/%D7%91%D7%A2%D7%99%D7%99%D7%AA_%D7%A1%D7%99%D7%A4%D7%95%D7%A7_%D7%90%D7%99%D7%9C%D7%95%D7%A6%D7%99%D7%9D) (CSP) המונח הומצא על ידי המתמטיקאי [דריק הנרי להמר](https://he.wikipedia.org/w/index.php?title=%D7%93%D7%A8%D7%99%D7%A7_%D7%94%D7%A0%D7%A8%D7%99_%D7%9C%D7%94%D7%9E%D7%A8&action=edit&redlink=1" \o "דריק הנרי להמר (הדף אינו קיים)) ([אנ'](https://en.wikipedia.org/wiki/Derrick_Henry_Lehmer)) בשנות החמישים.

נתאר את הבעיה כ[עץ](https://he.wikipedia.org/wiki/%D7%A2%D7%A5_(%D7%AA%D7%95%D7%A8%D7%AA_%D7%94%D7%92%D7%A8%D7%A4%D7%99%D7%9D)) החלטות, בו השורש הוא הבעיה ההתחלתית והעלים הם פתרונות (אולי לא נכונים). כל קודקוד יהיה פתרון חלקי, וקשת מקודקוד א' לקודקוד ב' שציין שניתן להגיע מפתרון א' לפתרון ב' בצעד אחד. כעת האלגוריתם עובד בצורה הבאה: כמו [DFS](https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D_%D7%97%D7%99%D7%A4%D7%95%D7%A9_%D7%9C%D7%A2%D7%95%D7%9E%D7%A7), הוא מתחיל מהשורש וכל פעם מבצע את האלגוריתם על כל אחד מילדיו בזה אחר זה, אלא שאם הוא מגיע לקודקוד שהפתרון שהוא מייצג לא אפשרי, הוא מיד חוזר אחורה ולא ממשיך. בצורה כזאת ניתן לפסול כמות משמעותית של פתרונות בלי לבדוק אותם.

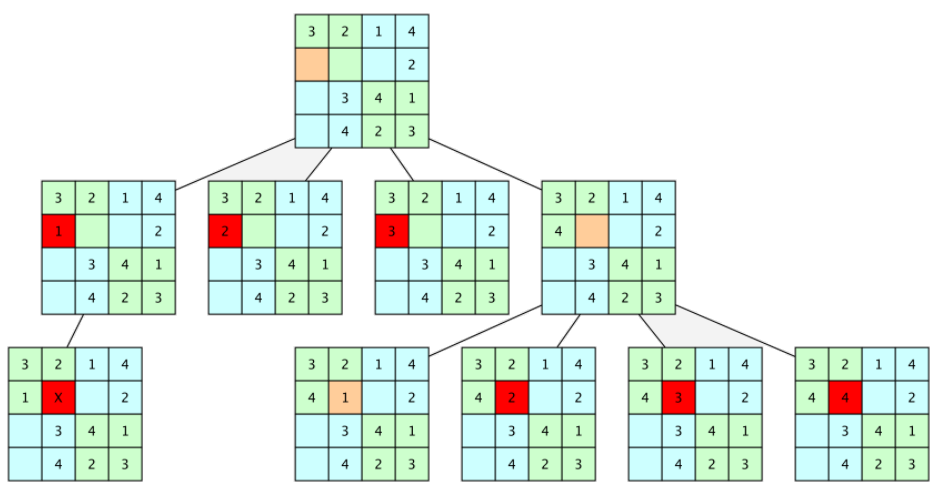
להלן פסאודו קוד של פתרון רקורסיבי בעיית סיפוק אילוצים:

פונקציה ראשית:

1. אתחל את כל המשתנים לריקים.
2. בחר את אחד המשתנים והפעל את הפונקציה הרקורסיבית.

פונקציה רקורסיבית (מקבלת משתנה):

1. נסה להציב למשתנה את כל הערכים, בזה אחר זה. לכל אחד מהם:
   1. אם הוא לא אפשרי, המשך לפתרון הבא. אם הוא אפשרי, בחר משתנה אחר והפעל את הפונקציה הרקורסיבית.
   2. אם חזר פתרון, החזר אותו. אם חזר false, המשך לפתרון הבא.
2. החזר false.

****

לכתוב על זמן ריצה ארוך של backtracking ושהוא מיועד לבעיות שהן הרבה פעמים NP שלמות כאשר אין פתרון יעיל אחר.

1. **למידה עמוקה**

## רקע כללי על למידה עמוקה ורשתות קונבולציה

למידת עמוקה היא חלק מגוון שיטות בלמידת מכונה המבוססות על רשתות נוירונים מלאכותיות, אשר משתמשות בשכבות מרובות על מנת לחלץ מאפיינים ממידע גולמי. למשל, בעיבוד תמונה, השכבות הראשונות ברשת יזהו מאפיינים כמו פינות, ואילו השכבות המתקדמות יזהו מאפיינים בעלי משמעות אנושית כמו זיהוי ספרות, פנים וכדומה.

רשת נוירונים עשויה לשמש הן לרגרסיה (קירוב פונקציה בעלת טווח יציאה רציף) והן לסיווג. תהליך הלימוד מסתמך על פונקציית השגיאה. נסמן ב-y את התגית הנכונה של הדוגמא וב-o את מוצא הרשת (תגית החיזוי).

קיימות פונקציות שגיאה שונות כגון:

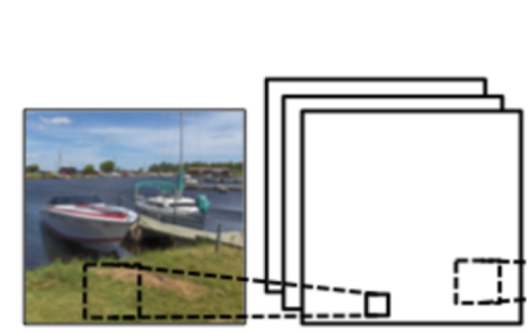
1. פונקציית השגיאה הריבועית- המציגה את סכום ההפרשים בריבוע בין תגית החיזוי לתגית הנכונה של הדוגמא, .
2. פונקציית השגיאה היא פונקציית "האנטרופיה הצולבת", .

האופטימיזציה של הרשת נעשית באמצעות אלגוריתם back-propagation, אשר משמש בעיקר לאימון של רשתות נוירונים בשילוב עם שיטות אופטימיזציה מבוססות גרדיאנט. כאשר דוגמא חדשה מוצגת לרשת, היא מפעפעת דרכה עד לשכבה האחרונה (forward pass). התוצאה המתקבלת מושווית לתוצאה הרצויה דרך פונקציית הפסד כלשהי לקבלת השגיאה. בשלב זה מתבצע ה-propagation-back: השגיאה מפעפעת אחורה באמצעות הגראדינט, מהשכבה האחרונה אל הראשונה (כאר האלגרויתפ מתבסס על עקרון "כלל השרשרת"). בצורה זו ניתן לחשב את הגרדיאנט של השגיאה ביחס לכל אחד מהמשקולות הנלמדים על הרשת.

אלגוריתם SGD (Stochastic Gradient Descent) מנסה למזער את פונקציית השגיאה, והוא מעדכן כל משקל עבור כל דוגמא ע"י החסרת הגרדיאנט מוכפל בגודל צעד כלשהו. קיימים אלגוריתמים נוספים נמסים למזער את פונקציית השגיאה, כמו ADAM, אשר קובע גודל צעד שונה לכל משקל בהתאם לגרדיאנטים הראשונים המתקבלים עבורו. בשונה מ-SGD, זהו אלגוריתם מסתגל.

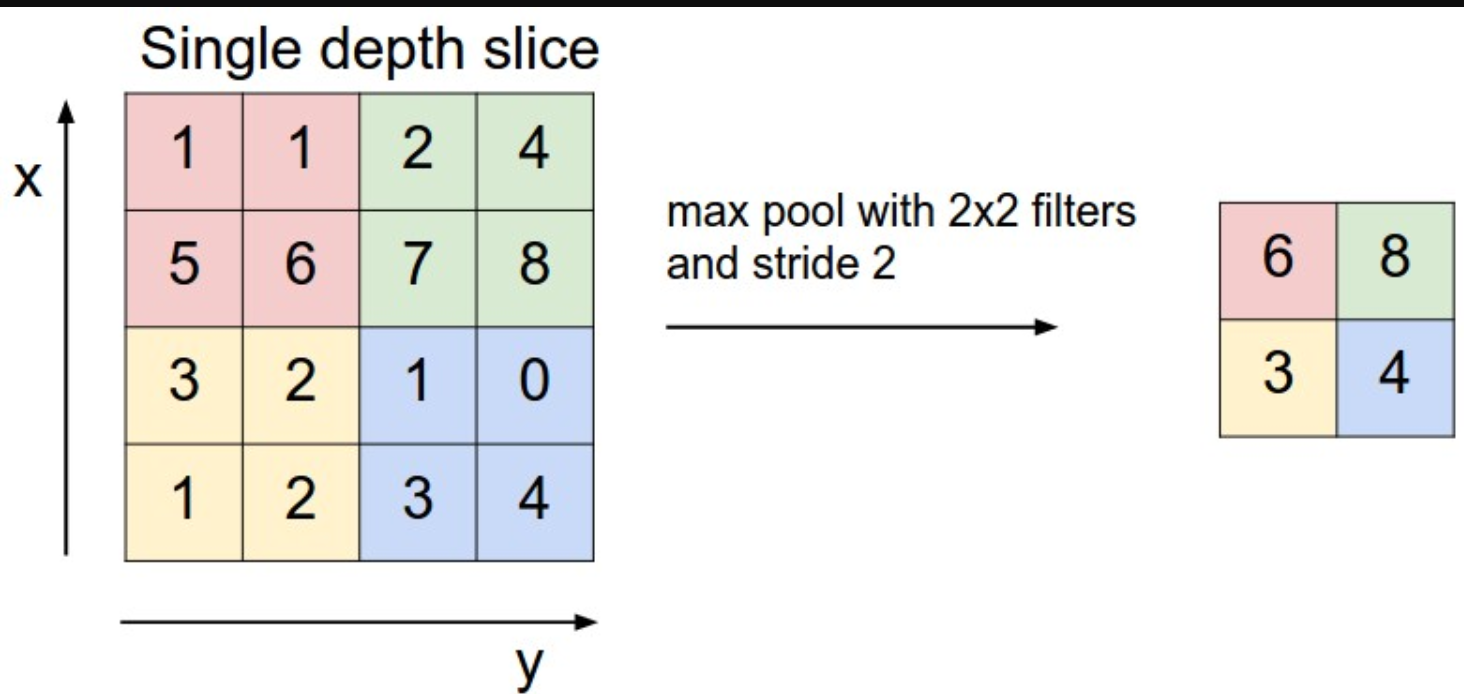
רשתות קונבולוציה ( CNN – Convolutional Neural Networks) הן רשתות נוירונים עמוקות שמשתמשות בשכבות קונבולציה. השכבות הטיפוסיות בהן הן:

שכבת קונבולוציה – הפרמטרים של השכבה מורכבים מאוסף של מסננים נלמדים, אשר יש להם שדה קלט (receptive field) קטן, אך מתרחבים לעומק הקלט. עם ההתקדמות של המסנן, הוא עובר קונבולוציה עם החלק המתאים בתמונה, והרשת לומדת מסננים אשר מבחינים במאפיין ספציפי במרחב מסוים בתמונה. רשתות קונבולוציה פופלריות מאוד בתחום ראייה ממוחשבת ובהן נשתמש בפרויקט שלנו.



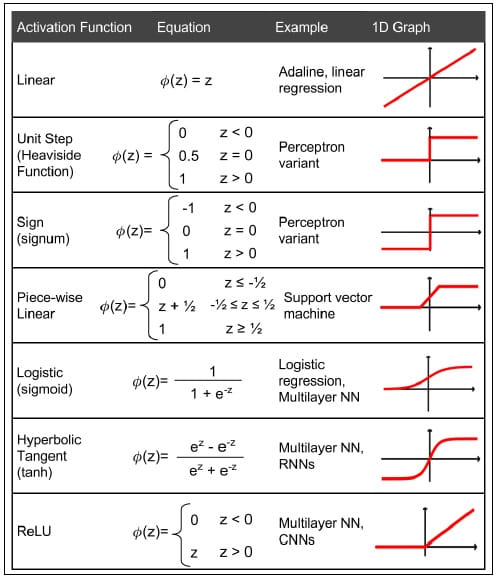
איור 24. דוגמא לשכבת קונבולציה בתמונה

שכבת pooling – שכבה שתפקידה להוריד את הרזולוציה המרחבית של התמונה. השיטה הנפוצה ביותר היא באמצעות max pooling, המחלקת את התמונה לאוסף של חלקים לא חופפים, וכפלט מוציאה את המקסימום של כל חלק. שכבה זו מסייעת בהורדת כמות הפרמטרים במערכת וכך מקלה על העומס החישובי של אימון הרשת, וכן עוזרת למנוע התאמת היתר (overfitting).



איור 25. דוגמא לשכבת pooling

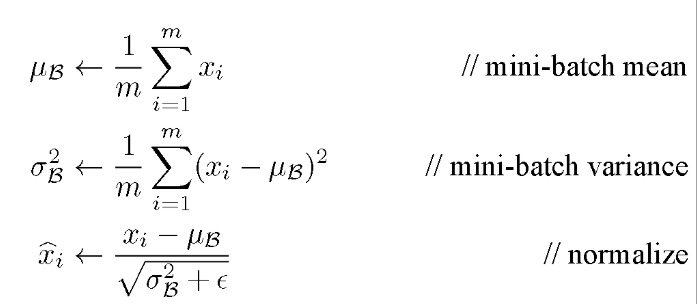
שכבת ReLU – שכבת האקטיבציה אשר הקלט שלה הוא . היתרון שלה על פונקציות אקטיבציה אחרות, כמו טנגנס היפרבולי או פונקצית הסיגמואיד (), הוא שאין לה חלק בו הפונקציה נכנסת לרוויה והגרדיאנטים מתאפסים, דבר המאיץ את התכנסות האלגוריתם. פונקציית האקטיבציה ReLU נפוצה מאוד בארכיטקטורות של למידה עמוקה, וגם אנחנו בפרויקט זה השתמשנו בפונקציה זו על מנת ללמד את הרשת.



איור26 . דוגמאות לפונקציות אקטיבציה

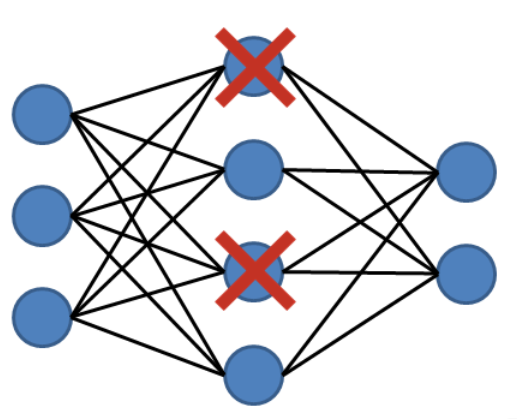
שכבת batch normalization – שכבה המנרמלת את הפלט של השכבה שקדמה לה ע"י החסרת ממוצע ה-batch של הנתונים וחלוקה בסטיית התקן שלו. שכבה זו מאיצה את תהליך הלימוד, מכיוון שהגרדיאנטים של פונקציית השגיאה כפונקציה של הפרמטרים השונים ברשת אינם נכנסים לרוויה כתוצאה מפונקציית האקטיבציה.

כך עבור קלט וגודל batch של m דגימות - פעולת הנרמול היא:

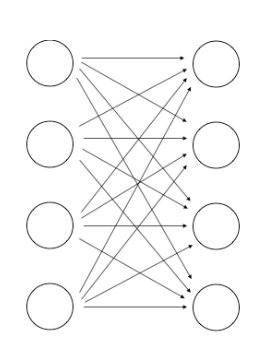


לאחר הנרמול הקלט הוא .

שכבת dropout – שכבה המבצעת רגולריזציה להורדת התאמת היתר ברשת ע"י כיבוי כל נוירון בהסתברות מסוימת הניתנת לשכבה כפרמטר.



איור 27. דוגמאות לשכבת dropout

שכבת fully connected – נמצאת בדרך כלל בסוף הרשת. הנוירונים בשכבה זו מקושרים לכל הפלטים בשכבה הקודמת, והם מפעילים התמרה אפינית באמצעות כפל מטריצי על הקלט שלהם.

איור 28. דוגמא לשכבת fully connected המקושרת לשכבה שקדמה לה.

שכבת ה-loss – שכבה זו היא למעשה פונקציית השגיאה. פונקציית השגיאה גדולה יותר ככל שקיימים הבדלים בין תיוגי האמת של הקלט לעומת התיוגים המתקבלים מהרשת. דוגמאות לפונקציות שגיאה:

1. **סקר ספרות**
2. **הפתרון בפרויקט שלנו – תיאור כללי**



**?. סיכום ומסקנות**

**?. נספחים**

**?. ביבליוגרפיה**