

**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**

תוכן עניינים

**dataset10**

הקלד כותרת פרק (רמה 2)2

הקלד כותרת פרק (רמה 3)3

**הקלד כותרת פרק (רמה 1)4**

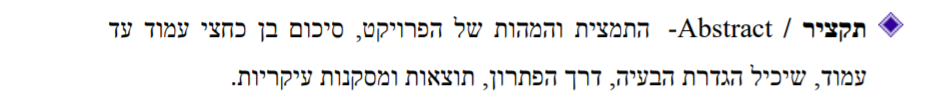
הקלד כותרת פרק (רמה 2)5

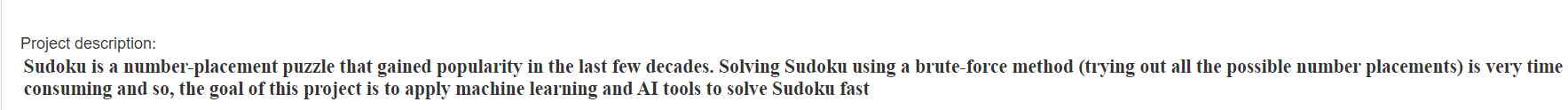
הקלד כותרת פרק (רמה 3)6

**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.

1. **מבוא**

משחק הסודוקו הוא משחק פנאי שצבר תאוצה רבה בשנים האחרונות. משחק זה מורכב מלוח בגודל x99 אותו צריך למלא עם מספרים בין 1 ל 9 על פי הכללים הבאים: בכל שורה ,עמודה ובלוק לא יהיה איבר שחוזר פעמיים. הפתרונות המחשוביים לבעיה זו כיום הם בעיקר פתרונות המשתמשים בשיטת backtracking. שיטה זו היא למעשה ניסיון של כל האופציות האפשריות באופן איטרטיבי (מילוי איבר אחרי איבר) יחד עם קטיעת אפשרויות שמלכתחילה שגויים. פתרון כזה אורך זמן רב עבור לוחות שבהם חסרים מספרים רבים. שיטה אחרת לפתרון אותה אנחנו חוקרים לטובת קיצור זמני הפתרון היא שימוש בלמידה עמוקה שצברה לאחרונה תאוצה וכיום השימוש בה הוא נרחב מאד. בפתרון שלנו אנחנו בודקים את היכולת של רשת נוירונים ללמוד את חוקי המשחק ולפתור אותו בזמן קצר משיטת ה backtracking. לצורך כך, בדקנו מספר ארכיטקטורות ושיטות אימון שונות, והשוונו ביניהן. לאחר מכן בחנו את ארכיטקטורות השונות כאשר שיטת המבחן היא מילוי של איבר אחרי איבר בלוח בדיוק כמו שאדם רגיל פותר, מתוך הנחה שמילוי איבר-איבר בלוח יביא לתוצאות טובות יותר מאשר פתרון הלוח בבת אחת. כמו כן, נעזרנו ב-dataset של לוחות סודוקו שקיים באתר Kaggle, והשתמשנו בעקרונות של curriculum learning על מנת לאמן את הרשת. בעבודה זו נדבר תחילה על רקע כללי של סודוקו, backtracking ולמידה עמוקה. לאחר מכן נציג סקר ספרות שעליו הסתמכנו בפתרון הבעיה, ולבסוף נבצע השוואה בין ארכיטקטורות שונות וננתח את התוצאות.

1. **סודוקו – רקע כללי**

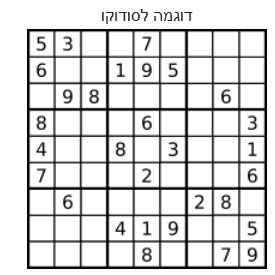
סודוקו הינו תשבץ מספרים שבו צריך למקם ספרות על לוח משובץ שגודלו 9X9, המורכב מ-9 ריבועים בני 9 משבצות כל אחד. פירושו המילה "סודוקו" ביפנית היא "מספר יחיד", משום שמטרת המשחק היא למקם 9 ספרות (הספרות 1 עד 9) על גבי לוח המשחק כך שבכל טור, בכל שורה, ובכל ריבוע, יופיע כל סמל בדיוק פעם אחת. בלוח המשחק נתונים 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. אולם, קיימות הרבה פחות אפשרויות לסודוקו מאשר ריבועים לטיניים, משום שפתרון סודוקו דורש אילוץ נוסף, אזורי הבלוקים.

מספר האפשרויות ללוחות סודוקו של 9X9 הוא 6,670,903,752,021,072,936,960. (6.67\*10^21 בקירוב). המספר המקסימלי של נתונים שניתן לספק מבלי שהפתרון יקבע ביחידות הוא גודל הלוח עצמו פחות 4 (במקרה של לוח 9X9 זה 77). המספר המינימלי של נתונים שיש לספק על מנת שהפתרון יקבע ביחידות הוא 17 נתונים.

בעיית הסודוקו היא בעיה NP – שלמה, ולא קיים פתרון אלגוריתמי יעיל לפתרון סודוקו. לכן אלגוריתם מקובל לפתרון סודוקו הוא על ידי backtracking, כלומר מעבר רקורסיבי על כל האפשרויות. נפרט על אלגוריתם ה-backtracking בפרק 3.

איור - סודקו

1. **Backtracking**

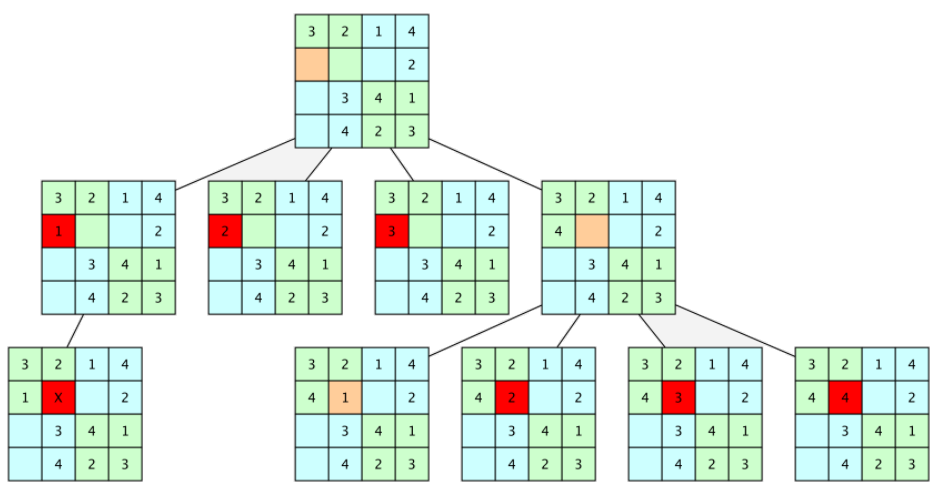
Backtracking (בעברית "עקיבה לאחור") הוא אלגוריתם אשר מבצע מעבר באופן רקורסיבי על כל האפשרויות של בעיה כלשהי במטרה למצוא חלק מהפתרונות או את כל הפתרונות של הבעיה. אלגוריתם זה למעשה מבצע חיפוש שחוסך מעבר על מספר רב של מועמדים לפתרון על ידי שימוש בתכונות ספציפיות של הבעיה. כלומר, האלגוריתם בונה באופן הדרגתי מועמדים לפתרון, וזונח מועמדים לפתרון אשר בהכרח לא יכולים להיות פתרון תקין לבעיה.

אלגוריתם backtracking מיועד עבור מציאת פתרון לבעיות שניתן לפתור בהן פתרונות חלקיים. דוגמא נפוצה לכך היא בעיה שבה יש מספר משתנים, ולכל משתנה צריך להתאים ערך מסוים כך שיתקיימו מספר אילוצים, כמו למשל בעיות סודוקו, תשבצים, תשחצים וכו'.

אלגוריתם backtracking לוקח בדרך כלל זמן ריצה רב, משום שסיבוכיות הריצה שלו גבוהה מאוד. על כן, בדרך כלל נעשה שימוש באלגוריתם זה כאשר אין פתרון יעיל אחר לבעיה.

ניתן לתאר את בעיית ה- backtrackingכעץ החלטות שבו השורש הוא המצב ההתחלתי (הבעיה ההתחלתית) והעלים של העץ הן הפתרונות (כולל פתרונות לא נכונים). כל צומת בעץ הוא פתרון חלקי, וקיים קשת מצומת א' לצומת ב' רק אם ניתן להגיע מפתרון חלקי א' לפתרון חלקי ב' בצעד אחד. הפתרון פועל בצורה הבאה: האלגוריתם מתחיל מהשורש ובכל פעם מתבצע האלגוריתם על כל אחד מהילדים של השורש בעץ. אם מגיעים לצומת בעץ שמייצגת פתרון שלא אפשרי אז חוזרים אחורה ולא ממשיכים. באופן זה ניתן לפסול פתרונות רבים מבלי לבדוק אותם.

ניתן כעת דוגמא לפתרון בעיית הסודוקו באמצעות אלגוריתם backtracking. אלגוריתם backtracking זוהי הדרך הפשוטה ביותר לפתור את בעיית הסודוקו. עוברים בשיטה זו על כל האפשרויות באופן הבא: בוחרים מקום ריק כלשהו על הלוח שלא מופיע שם ספרה, וממקמים במקום זה את ספרה חוקית כלשהי, ואז עוברים למיקום הריק הבא על הלוח באופן דומה עד לפתרון הבעיה. אם האלגוריתם לא מצא באחד השלבים ספרה חוקית לשים באחד המקומות הריקים, אז הפתרון לא חוקי, ויש לחזור אחורה ולנסות אפשרויות אחרות. כפי שצוין קודם לכן, שיטה זו בעלת סיבוכיות זמן גבוהה מאוד.

****

1. **למידה עמוקה - רקע כללי**

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:

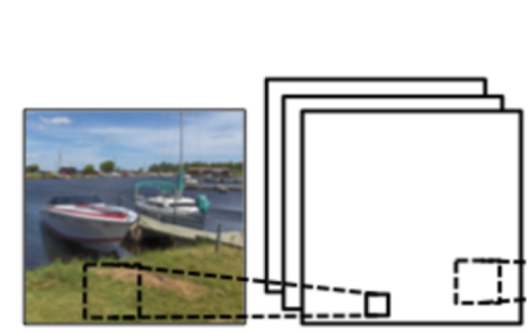
1. פונקציית השגיאה הריבועית- המציגה את סכום ההפרשים בריבוע בין תגית החיזוי לתגית הנכונה של הדוגמא, .
2. פונקציית השגיאה היא פונקציית "האנטרופיה הצולבת", .

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.

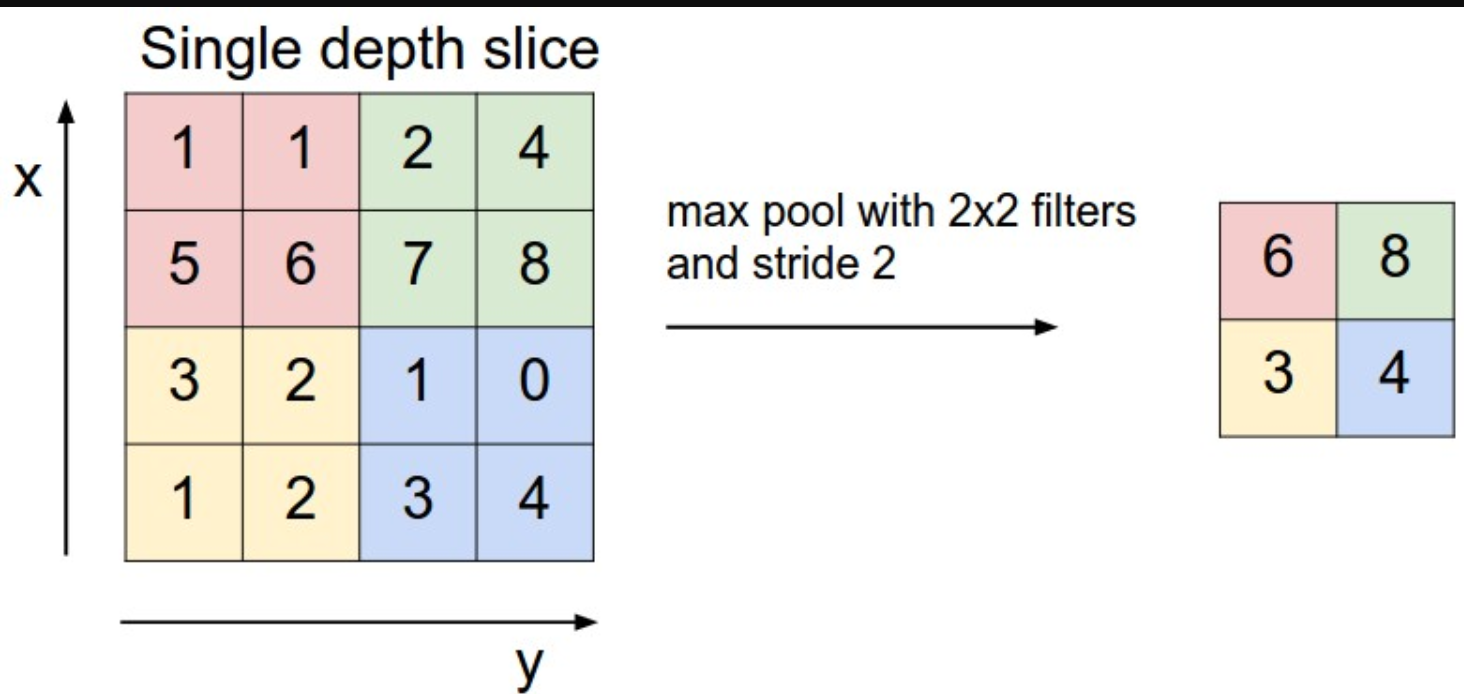
רשתות קונבולוציה ( CNN – Convolutional Neural Networks) הן רשתות נוירונים עמוקות שמשתמשות בשכבות קונבולציה. השכבות הטיפוסיות בהן הן:

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

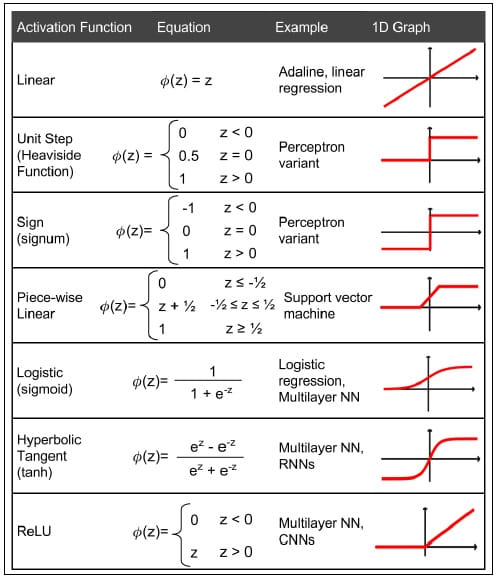
שכבת pooling – שכבה שתפקידה להוריד את הרזולוציה המרחבית של התמונה. השיטה הנפוצה ביותר היא באמצעות max pooling, המחלקת את התמונה לאוסף של חלקים לא חופפים, וכפלט מוציאה את המקסימום של כל חלק. שכבה זו מסייעת בהורדת כמות הפרמטרים במערכת וכך מקלה על העומס החישובי של אימון הרשת, וכן עוזרת למנוע התאמת היתר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

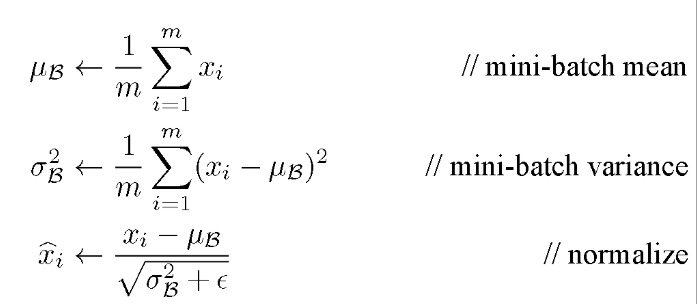
(), הוא שאין לה חלק בו הפונקציה נכנסת לרוויה והגרדיאנטים מתאפסים, דבר המאיץ את התכנסות האלגוריתם. פונקציית האקטיבציה ReLU נפוצה מאוד בארכיטקטורות של למידה עמוקה, וגם אנחנו בפרויקט זה השתמשנו בפונקציה זו על מנת ללמד את הרשת.



איור26 . דוגמאות לפונקציות אקטיבציה

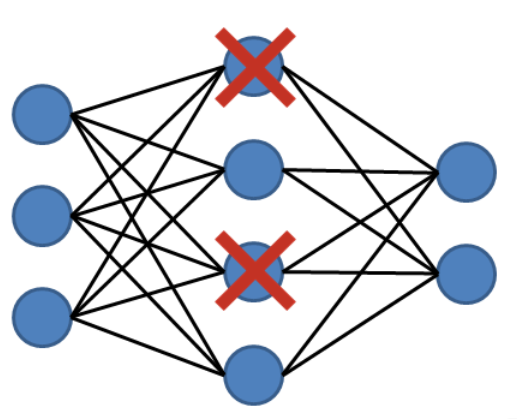
שכבת batch normalization – שכבה המנרמלת את הפלט של השכבה שקדמה לה ע"י החסרת ממוצע ה-batch של הנתונים וחלוקה בסטיית התקן שלו. שכבה זו מאיצה את תהליך הלימוד, מכיוון שהגרדיאנטים של פונקציית השגיאה כפונקציה של הפרמטרים השונים ברשת אינם נכנסים לרוויה כתוצאה מפונקציית האקטיבציה.

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

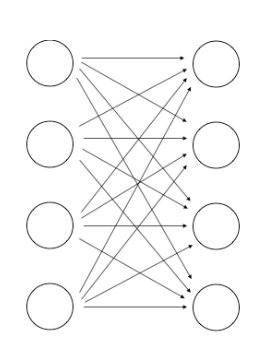


לאחר הנרמול הקלט הוא .

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**

5.1. Curriculum learning

במאמר [1] מוסבר על אימון רשתות באמצעות למידה עמוקה על ידי שיטת Curriculum learning. במאמר מוסבר כי בדרך כלל כאשר מאמנים רשתות על ידי למידה עמוקה, האימון נעשה באמצעות שימוש ב-mini-batches, אשר מוגרלים באופן אקראי מתוך סט האימון. אולם בשיטת Curriculum learning, ה-mini-batches לא מוגרלים באופן אקראי אלא באופן ממוין לפי רמת קושי. כלומר כדי לאמן באמצעות Curriculum learning, צריך למיין את סט האימון לפי רמות קושי שונות, וצריך להגריל את ה-mini-batches לפי רמות הקושי בסדר עולה.

במאמר מוסבר כי כפי שמורים מלמדים תלמידים משימות מסובכות על ידי שימוש ב"תכנית לימודים" (Curriculum באנגלית), כך גם שימוש ב"תכנית לימודים" עבור אימון רשתות עשוי לתרום לאימון הרשת ולהביא לתוצאות טובות יותר. הכוונה ב"תכנית לימודים" היא למידה על פי רמת קושי עולה, דבר אשר מאפשר להשתמש בידע שנלמד ברמת קושי קלה יותר כדי ללמוד בצורה טובה יותר בעיה קשה יותר.

במאמר ביצעו את מיון סט האימון לפי רמות הקושי השונות על ידי שימוש בשתי שיטות שונות:

1. שיטה אחת היא שימוש ב-transfer learning מרשת אחרת:

בשיטה זו מיון סט האימון לפי רמות קושי נעשה לפי הדיוק של רשת מאומנת אחרת על סט זה. כלומר אם הרשת האחרת הצליחה היטב דוגמאות מסוימות, אז דוגמאות אלו נחשבות "קלות", ואם הרשת נכשלה מאוד עבור דוגמאות מסוימות במהלך האימון אז דוגמאות אלו נחשבות "קשות". איכות ההצלחה של הרשת נקבעת לפי פונקציית scoring אשר נותנת דירוג למידת ההצלחה של הרשת עבור כל דוגמא. שיטה זו עשויה לשפר את מהירות ההתכנסות, ואת הדיוק הסופי עבור רשתות CNN.

1. שיטה נוספת למיון סט האימון היא שימוש ב-bootstrapping:

בשיטה זו נעשה אין צורך בשימוש ברשת אחרת, אלא נעשה שימוש ב- bootstrappingעל מנת למיין את סט האימון לדרגות קושי שונות. בשיטה זו, דרגות הקושי נקבעות לפי התוצאה שמקבלת הרשת שהתאמנה ללא Curriculum על דוגמאות אלו. במידה והרשת שהתאמנה ללא Curriculum הצליחה היטב דוגמאות מסוימות, אז דוגמאות אלו נחשבות "קלות", ולהפך – אם הרשת נכשלה מאוד עבור דוגמאות מסוימות במהלך האימון אז דוגמאות אלו נחשבות "קשות". איכות ההצלחה של הרשת נקבעת לפי פונקציית scoring אשר נותנת דירוג למידת ההצלחה של הרשת עבור כל דוגמא.

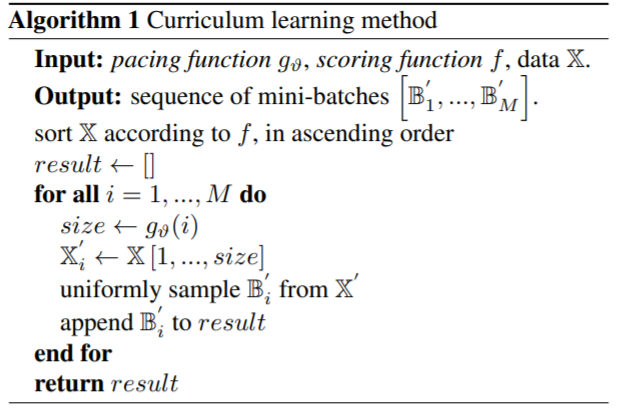
בנוסף, במאמר זה בדקו את "קצב הלימוד" שבו צריך ללמד את הרשת, כלומר באיזה קצב לעלות את רמת הקושי של הדוגמאות. ביצעו זאת על ידי שימוש בפונקציות pacing (קצב) שונות, אשר קובעות מהו הקצב שבו ה-data מוצג לרשת.

יתר על כן, במאמר זה מראים כי שימוש ב- Curriculum learningמשנה את האופטימיזציה של הרשת, וגורם להתכנסות להיות מהירה יותר, אך יחד עם זאת שומרת על המינימום הגלובאלי של הבעיה המקורית.

במאמר בדקו את הדברים השונים על סט האימון CIFAR-10, CIFAR-100 ו-ImageNet.

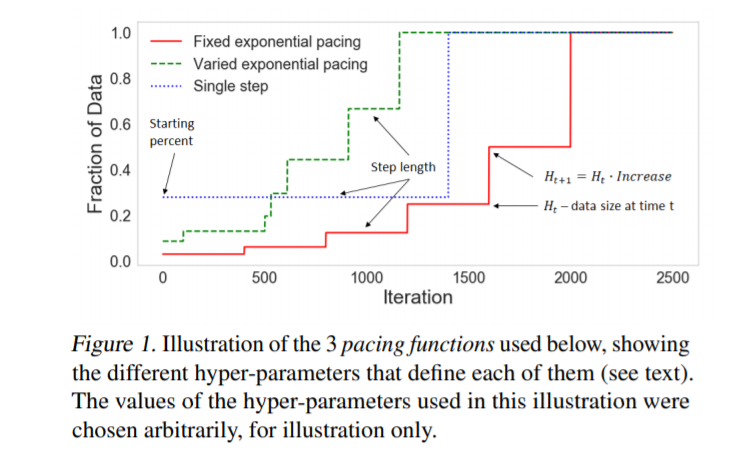
אלגוריתם ה-learning Curriculumשהוצג במאמר הוא כדלקמן:

הקלט הוא הדאטה, פונקציית ה-pacing ופונקציית ה-scoring. ממיינים את הדאטה X לפי סודר עולה של רמת קושי. רמת הקושי נקבעת לפי פונקציית ה-scoring. לאחר מכן, לוקחים מתוך הדאטה X רק את הדאטה עד רמת קושי מסוימת שנקבעת לפי פונקציית ה-pacing. לאחר מכן, מחלקים את הדאטה הנבחר ל-mini-batches. הפלט של האלגוריתם הוא ה-mini-batches בהתאם לפונקציות ה-pacing וה-scoring.



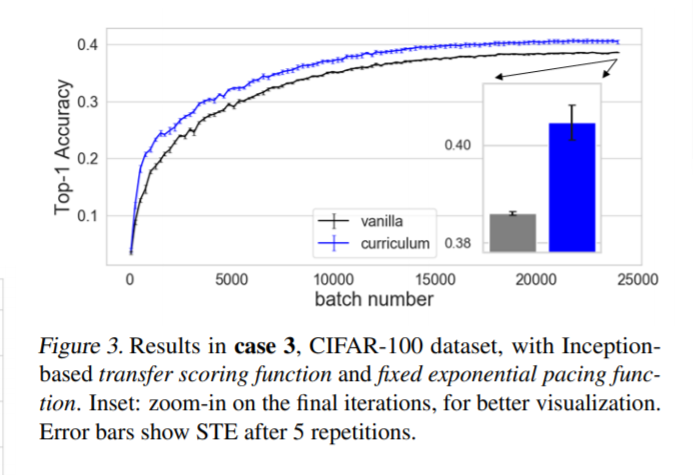
איור ???: אלגוריתם ה-learning Curriculumשהוצג במאמר [1].

נתבונן באיור [???] על 3 פונקציות pacing שהוצגו לדוגמא במאמר. הערכים שנבחרו עבור הפונקציות בדוגמא הם שרירותיים. ההבדל בין fixed exponential pacing לבין varied exponential pacing זה שעבור fixed exponential pacing מספר האיטרציות בכל צעד הוא קבוע. Single step זוהי פונקציית pacing אשר בעלת צעד אחד בלבד.



איור ???: דוגמא ל-3 פונקציות pacing שהוצגו במאמר.

נתבונן באיור [???] על התוצאות מתוך המאמר עבור Top-1 Accuracy על סט הולדיציה במהלך האימון עבור dataset של CIFAR-100 תוך שימוש ברשת קונבולוציה פשוטה שהוכנה ידנית. ניתן לראות שהדיוק עבור curriculum גבוה יותר.



איור ???: תוצאות עבור Top-1 Accuracy על סט הולדיציה במהלך האימון עבור dataset של CIFAR-100.

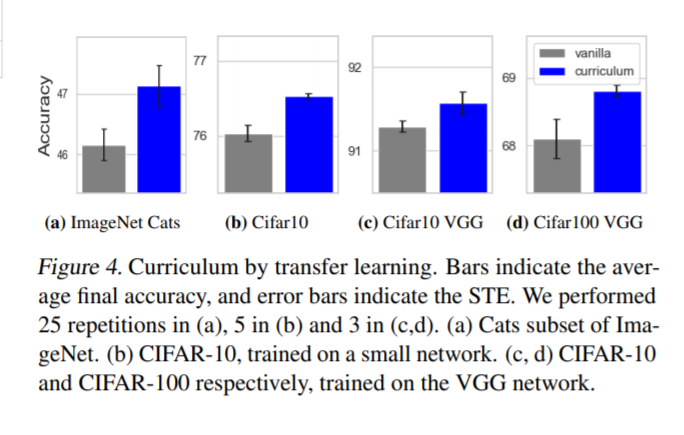
כמו כן, נתבונן באיור [???] על תוצאות עבור Top-1 Accuracy על סט המבחן עבור datasets שונים ורשתות שונות. ה-datasets והרשתות השונות שהשתמשו בהם הם כדלקמן (משמאל לימין):

Dataset - של ImageNet Cats באמצעות רשת קונבולוציה פשוטה שהוכנה ידנית.

Dataset - שלCIFAR-10 באמצעות רשת קונבולוציה פשוטה שהוכנה ידנית.

Dataset - שלCIFAR-10 באמצעות רשת VGG.

Dataset - שלCIFAR-100 באמצעות רשת VGG.

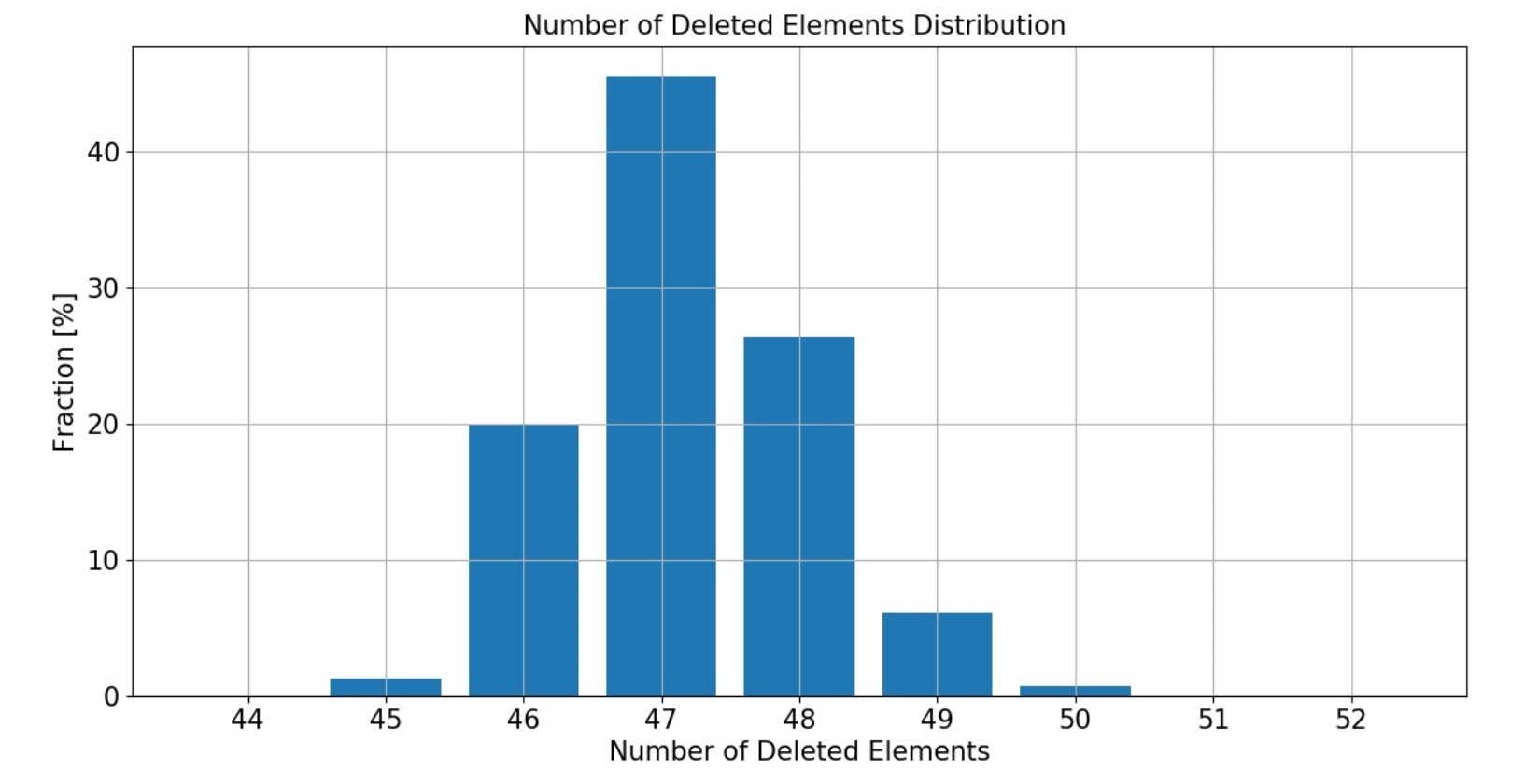


איור ???: תוצאות עבור Top-1 Accuracy על סט המבחן עבור datasets שונים ורשתות שונות.

יש לציין כי המושג Vanilla שמצוין באיור [???] זוהי שיטת האימון הרגילה שמשתמשים בה ללא Curriculum. ניתן לראות שהדיוק עבור curriculum גבוה יותר.

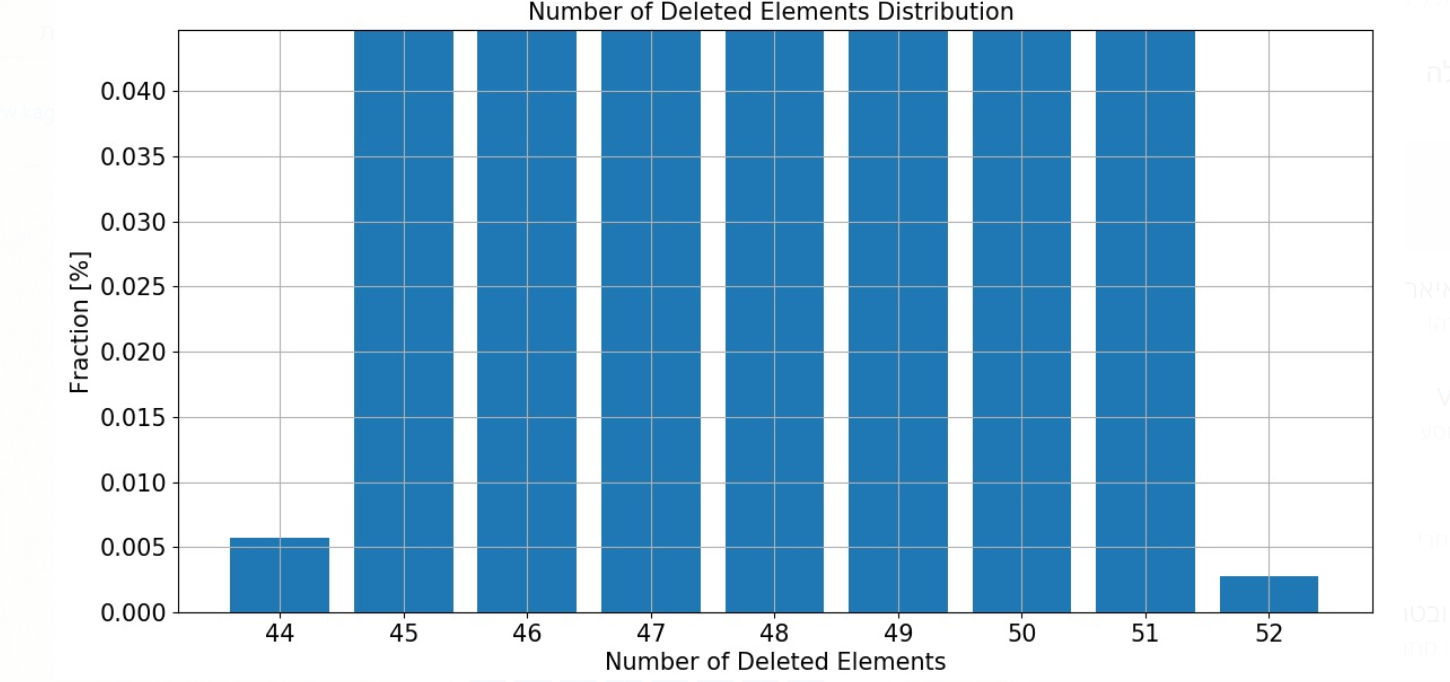
5.2. דוגמא לפתרון מאתר Kaggle [2]

Dataset 1 - בדוגמא זאת [2] מופיע dataset שהשתמשנו בו עבור הפרויקט שלנו. dataset זה מכיל מיליון לוחות ברמות קושי שונות. כלומר, הדאטה מכיל לוחות עם כמות שונה של איברים מחוקים שנעה בין 55 מספרים מחוקים עד ל-52 מספרים מחוקים. פילוג האיברים המחוקים של ה-dataset הוא באופן הבא:



איור [???] תרשים של מספר האיברים המחוקים.

נתבונן על התרשים גם בקירוב:



איור [???] תרשים של מספר האיברים המחוקים בקירוב.

נשים לב שקיימים לוחות עם 44, 51, ו-52 מספרים מחוקים, אבל כמות מעטה מאוד (עד כדי כך שלא ניתן לראות את הכמות באיור [???]). כלומר כמות הלוחות עם 44, 51, ו-52 מספרים מחוקים היא זניחה. ממוצע האיברים המחוקים הוא 47.19 איברים מחוקים לכל לוח.

הקוד שנכתב הוא ב-keras ואילו אנחנו העדפנו לכתוב בפרויקט שלנו ב-pytorch.

באלגוריתם באתר Kaggle תחילה מתבצע "כיול": מתבצע אימון לרשת עם 0 מספרים מחוקים, כלומר הרשת מקבלת לוח מלא, ומאמנים את הרשת להוציא לוח מלא זהה.

לאחר מכן, מוחקים עבור כל לוח בסט האימון ספרה אחת בלבד, ומאמנים את הרשת למצוא את המספר הזה.

לאחר מכן, מוחקים שני מספרים ומאמנים את הרשת למלא את הלוח החסר. באופן זה, ממשיכים למחוק ספרות נוספות ובכל פעם מאמנים את הרשתות למלא את הלוח עבור מספר מחוקים הולך וגדל. באופן זה מאמנים את הרשת באופן הדרגתי בכל פעם למלא לוחות בדרגות קושי הולכות וגדלות (Curriculum learning). ממשיכים לאמן באופן זה עד 55 מספרים מחוקים. כאשר בכל שלב ניתן למחוק יותר מספרה אחת.

מספר האיברים המחוקים בכל שלב מתבצע באופן הבא לפי הסדר:

* Epoch אחד של איבר אחד מחוק.
* 2 epochs של 2 איברים מחוקים.
* 3 epochs של 3 איברים מחוקים.
* 4 epochs של 4 איברים מחוקים.
* 6 epochs של 6 איברים מחוקים.
* 8 epochs של 8 איברים מחוקים.
* 10 epochs של 10 איברים מחוקים.
* 12 epochs של 10 איברים מחוקים.
* 15 epochs של 10 איברים מחוקים.
* 20 epochs של 10 איברים מחוקים.
* 25 epochs של 10 איברים מחוקים.
* 30 epochs של 10 איברים מחוקים.
* 35 epochs של 10 איברים מחוקים.
* 40 epochs של 10 איברים מחוקים.
* 45 epochs של 10 איברים מחוקים.
* 50 epochs של 10 איברים מחוקים.
* 55 epochs של 10 איברים מחוקים.

בסט המבחן ממלאים את הלוח בכל פעם ספרה אחת בלבד. כלומר ממלאים את הלוח איבר-איבר ובכל פעם ממלאים את האיבר עם ההסתברות הגבוהה ביותר, כלומר האיבר שהרשת בטוחה בערך שלו במידה הרבה ביותר.

נשים לב שהמחיקה שמצבעת בקוד זה היא עם חזרות, כלומר יתכן שימחק אותו איבר כמה פעמים. דבר זה גורם שאם רוצים למחוק למשל 50 איברים כנראה שבפועל ימחקו פחות, כי אותם איברים ימחקו כמה פעמים. בנוסף נשים לב שהמחיקה שמתבצעת בקוד זה היא לא בהכרח "מחיקה לגיטימית" – כלומר המחיקה מתבצעת באופן אקראי, וייתכן כי לאחר המחיקה נקבל לוח שניתן למלא אותו במספר דרכים. מבחינתו לוח סודוקו חלקי הוא תקין אם ניתן למלא אותו באופן יחיד. בקוד זה המחיקה לא מתבצעת כך באופן זה, ולכן ניתן לקבל לוחות סודוקו חלקיים לא תקינים לא המחיקה.

התוצאה אליה הגיעו היא 99.7% אבל זוהי תוצאה עשויה להטעות מכיוון שמספר האיברים המחוקים בסט המבחן נע בין 44 ל-52 איברים מחוקים, עם אותו פילוג של כמו של סט האימון בגרף שמוצג למעלה באיור [???]. כלומר כמות הלוחות עם 44, 51, ו-52 מספרים מחוקים היא זניחה, ולכן בפועל יש בעיקר לוחות בין 45 ל-50 מחוקים.

כלומר יחסית יש מעט איברים מחוקים בסט המבחן, ולכן הרשת מגיעה לתוצאות טובות יחסית. ככל שיש פחות איברים מחוקים, רמת הקושי של הלוח היא קלה יותר ולכן הרשת תטעה פחות ותגיע לתוצאות טובות יותר.

הרשת שבה השתמשו בנויה באופן הבא:

1. שכבת FC עם 64 נוירונים
2. שכבת ReLU
3. שכבת dropout עם הסתברות של 0.4
4. שכבת FC עם 64 נוירונים
5. שכבת ReLU
6. שכבת dropout עם הסתברות של 0.4
7. שכבת FC עם נוירונים (יש נוירונים כי יש לוח בגודל 9X9 ולכל איבר בלוח יש 9 אופציות).
8. שכבת SoftMax

ה-loss שבו השתמשו הוא Categorial\_crossentropy. נשים לב שה-loss שבו נעשה שימוש בקוד זה הוא על כל המספרים בלוח ולא רק על המספרים המחוקים. כלומר, כאשר מחשבים את הloss- אז נלקחים בחשבון כל האיברים בלוח, גם איברים שהופיעו בלוח בהתחלה ולא היו מחוקים.

ה-optimizer שבו השתמשו הוא "Adam optimizer".

גודל ה-batch הוא: 128 לוחות.

להוסיף לסוף:

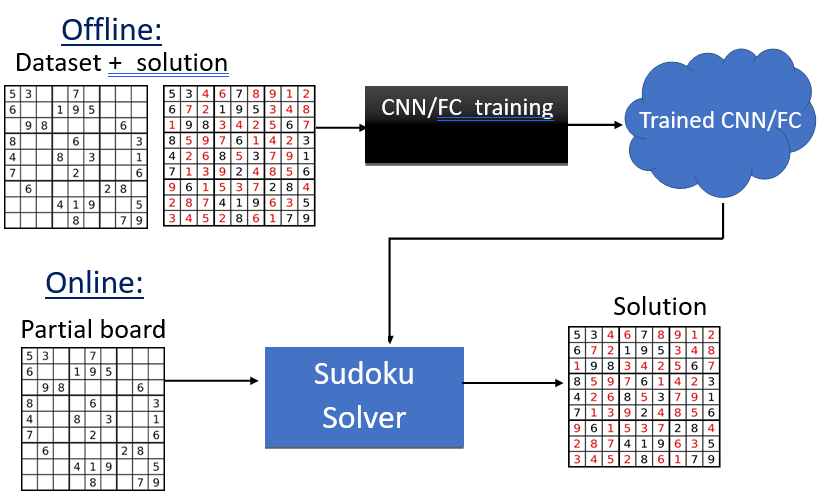
<https://www.kaggle.com/dithyrambe/neural-nets-as-sudoku-solvers>

1. **הפתרון בפרויקט שלנו**

**6.1 – תיאור כללי**

ראשית, בשלב ה -offline אנחנו מאמנים את רשת הנוירונים לפתור לוחות סודוקו באופן איטרטיבי, כלומר ברמת קושי הולכת וגדלה. אנחנו עושים זאת תוך שימוש בדאטה של מיליון לוחות שנלקח מ-kaggle. למעשה אנחנו פותרים את הבעיה בשיטת supervised learning: הרשת ממלאת את המספרים החסרים עבור כל לוח, ולכל מספר חסר בלוח קיים "תיוג" של המספר המלא המתאים. הסיבה שיש לנו את האיבר שצריך למלא לכל איבר חסר בלוח היא שאנחנו מתחילים מלוחות מלאים ומוחקים איברים בלוחות המלאים באופן איטרטיבי, כך שבכל איטרציה אנחנו מוחקים כמות גדולה יותר של איברים מהלוחות המלאים. פונקציית ה-Loss שאנו משתמשים בה מחושבת רק עבור האיברים הריקים בלוח, ולכן הרשת מתאמנת על מנת למלא את האיברים החסרים בצורה הטובה ביותר, מבלי לקחת בחשבון בפונקציית ה-Loss את האיברים שמופיעים כבר בלוח, והיא אינה צריכה למלא. באופן זה המשקלים של הרשת משתנים בכיוון מזעור פונקציית ה-Loss, וכך הרשת לומדת לזהות איך למלא את הלוח.

שנית, בשלב ה-online אנחנו מכניסים לרשת המאומנת לוח או batch של לוחות סודוקו עם איברים חסרים. לוח זה לא הופיע כמובן בסט האימון (הרשת לא התאמנה על לוח זה). הרשת שאומנה לפתור סודוקו פותרת באופן איטרטיבי איבר-איבר בלוח החלקי (היא ה-Sudoku Solver באיור [???]), והמוצא של Solver Sudoku הוא הפתרון של הלוח החלקי בשלב ה-online. נציין גם כי הפתרון נעשה באופן איטרטיבי על מנת להגיע לתוצאות טובות יותר ע"י Curriculum learning, נפרט על כך בהמשך.



איור ???. תרשים מלבנים אשר מייצג את שלבי הפרויקט השונים למטרת אימון רשת לפתרון לוחות סודוקו בעזרת למידה עמוקה.

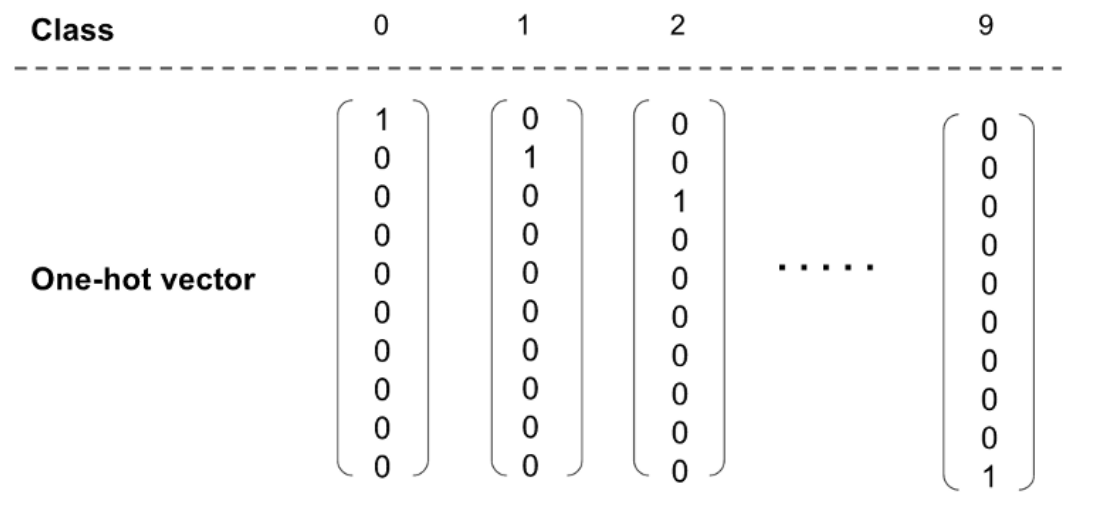
**6.2. ה-dataset**

כאמור, ה-dataset שלנו הם לוחות סודוקו חסרים ולוחות מלאים מתאימים. בלוחות אלו המספרים הם 1-9 כידוע, אך נשאלת השאלה האם יש משמעות לערכי המספרים בלוח סודוקו? הרי גם אם כל מספר היה מוחלף בשם של חיה, המשחק היה נשאר זהה. לכן עבדנו עם וקטור 1hot לכל ספרה – כדי לא לתת אפשרויות להתייחסות שונה של הרשת עבור ספרות שונות. כך, כל לוח סודוקו חסר, הפך למטריצה בגודל 10x9x9 (10 במימד העומק מכיוון שיש גם לספק את הספרה 0 המבטאת תא חסר, וספרות 1-9 בהתאם לספרות 1-9). והפתרונות המלאים הפכו למטריצה בגודל 9x9x9 (כאן מימד העומק הוא 9 כי לא יתכן תא ריק במוצא אלא ספרות 1-9). עבור מיקום שורה ועמודה, אנחנו מקבלים וקטורOne-hot המתאים לספרה שנמצאת שם את הדאטא פיצלנו לסט אימון, ולידציה ומבחן לפי היחס 10%, 20%, 70% בהתאמה.

התחלנו להשתמש ב- dataset שמצאנו בKaggle- אשר מכיל מיליון לוחות עם 44 עד 52 מספרים חסרים, כאשר פילוג המספרים החסרים מופיע באיורים [???] ו-[???]. נשים לב כי כמות הלוחות שיש בהם 44, 51 ו-52 מספרים חסרים היא זניחה.

כאשר שמנו לב שהתוצאות של הרשת לא מספיק טובות באיור [???] עבור הרבה מספרים מחוקים, החלטנו לחפש לוחות עם יותר מספרים מחוקים. מצאנו dataset ב-Kaggle אשר מכיל לוחות עם יותר מספרים מחוקים ולקחנו משם לוחות עם ???

באיור [???] ניתן לראות המרה בין ספרה לוקטור One-hot.



איור [???] – הסבר על וקטורOne-hot .

**6.3. רשת FC**

עבדנו עם 2 סוגים מרכזיים של רשתות: רשת FC ורשת CNN. כפי שניתן יהיה לראות בהמשך רשת ה-FC הגיעה לתוצאות טובות יותר במעט לעומת רשת ה-CNN, ולכן את מרבית הבדיקות ביצענו עם רשת ה-FC.

ראשית, מכיוון שמימד הבעיה יחסית נמוך (לוחות קטנים יחסית), ומתוך רצון להימנע מהתאמת יתר, החלטנו לעבוד עם רשתות פשוטות. הכניסה לרשת היא מטריצה 10x9x9 עבור לוח חסר והמוצא הוא מטריצה 9x9x9 שבמקום ה (i, j, k) אנחנו מקבלים את ההסתברות שהמספר המתאים למיקום ה (j, k) הוא i. באופן פשוט יותר, עבור שורה j , עמודה k ועומק i אנחנו מקבלים את ההסברות לאיבר i+1 באותה שורה ועמודה. נשים לב שמימד הכניסה הוא 10x9x9 ולא 9x9x9 כי קיימת האופציה של הספרה 0 עבור מימד הכניסה שמייצגת איבר ריק בלוח.

הארכיטקטורה הראשונה שהשתמשנו בה היא רשת המורכבת משלוש שכבות FC עם batch-Normalization ופונקצית אקטיבציה ReLU. לאחר מכן, מסדרים את הנוירונים בצורת מטריצה 9x9x9 ולבסוף שכבת softmax על מימד העומק במטריצה הסופית לקבלת ההסתברויות לכל ספרה. כלומר הרשת בנויה באופן הבא:

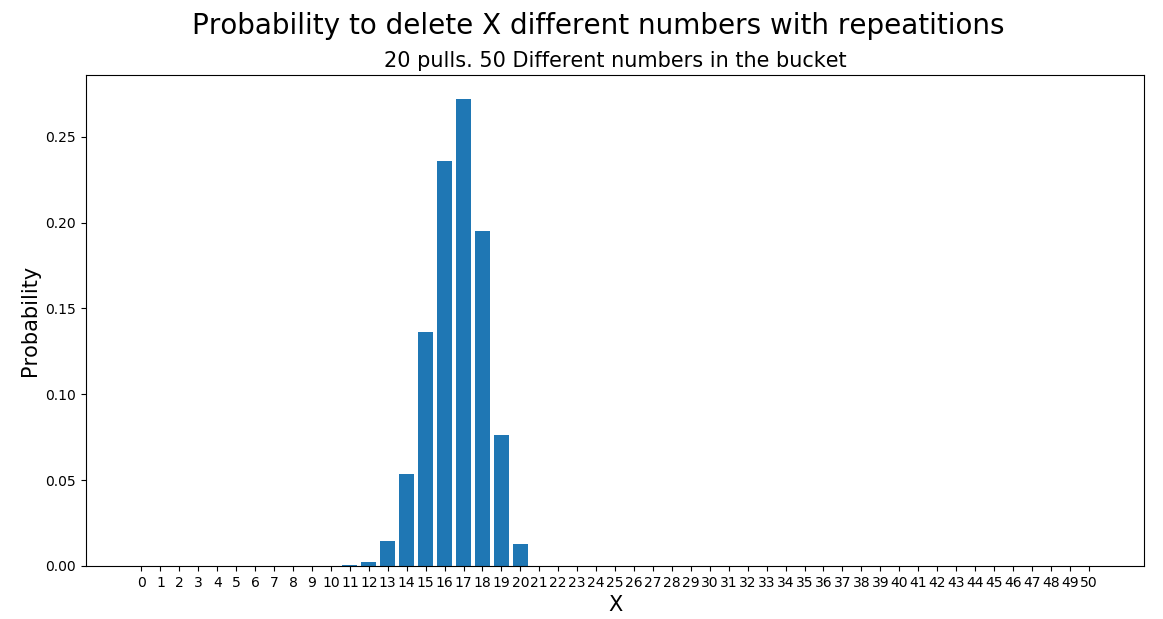
1. שכבת FC עם נוירונים
2. שכבת batch normalization
3. שכבת ReLU
4. שכבת FC עם נוירונים
5. שכבת batch normalization
6. שכבת ReLU
7. שכבת FC עם נוירונים (יש נוירונים כי יש לוח בגודל 9X9 ולכל איבר בלוח יש 9 אופציות).
8. שכבת SoftMax

פונקציית ההפסד נבחרה באופן הבא: מכיוון שחלק מהתאים כבר ידועים מראש, מה שהיה חשוב לנו הוא שהרשת תלמד את למלא התאים שאינם ידועים, ואז את מה שידוע פשוט נדביק לבסוף על התוצאה שהתקבלה מהרשת. לכן, חישוב הלוס התבצע על כל תא חסר בלבד, אל מול התוצאה הידועה של אותו תא בלוח המלא. מכאן למעשה אפשר לדמות את הבעיה כבעיית סיווג עבור כל תא בנפרד, שמנסה לחזות את המחלקה (המספר) המתאים עבור אותו תא. ולכן השתמשנו בפונקציית הפסד שהיא Cross Entropy על כל תא חסר, כאשר כפי שציינו קודם לכן לכל תא יש 9 אופציות לסווג: הספרות 1-9.

יש לציין שהדיוק וה-Loss עבור סט האימון בוצע באופן שונה מהחישוב עבור סט הולידציה: עבור סט האימון חישבנו כך שהרשת פותרת את כל האיברים החסרים בלוח בבת-אחת (כלומר ללא איטרציות, וללא מילוי איבר-איבר אלא באיטרציה אחת הרשת ממלאת את כל הלוח). אולם עבור סט הולדיציה חישבנו כך שהרשת פותרת איבר-איבר בלוחות החסרים, כלומר כפי שהרשת תפתור בזמן אמת. כלומר הרשת בסט הולידציה ממלא בכל איטרציה את האיבר עם ההסתברות הכי גבוהה. הסיבה שביצענו את החישוב באופן ששונה בין סט האימון לסט הולידציה היא שפתרון איבר-איבר בלוח לוקח זמן ריצה רב, ואם היינו מבצעים פתרון בסט האימון על ידי איבר-איבר ולא על ידי פתרון הלוח כולו בבת אחת, אז זמן האימון של הרשת היה גדל באופן משמעותי והיה הופך את זמן האימון של הרשת ללא אפשרי מבחינה חישובית.

תהליך האימון התבצע באופן הבא: לצורך אימון הרשת, השתמשנו בשיטת curriculum learning כך שלמעשה עבדנו רק עם הלוחות המלאים, וכל פעם מחקנו איבר נוסף ולימדנו את הרשת להתרגל למצב החדש. תחילה מחקנו איבר בודד ולימדנו את הרשת להשלים אותו לאורך מספר epochs. לאחר מכן מחקנו שני איברים ולימדנו את הרשת להשלים אותם לאורך כמה epochs וכך הלאה. האיברים שאותם אנחנו מוחקים, אלו איברים שהיו מחוקים בלוחות החסרים המתאימים לאלו שמהם מחקנו. כלומר, עבור מחיקה של 20 איברים מלוח מלא, אנחנו מסתכלים אילו איברים מחוקים באותו לוח חסר המתאים ללוח המלא הזה, ומהם מוחקים את האיברים. לסוג מחיקה זו אנחנו קוראים "מחיקה לגיטימית". עבור מחיקה רנדומלית מתוך הלוח המלא, אנחנו עלולים למחוק איברים ככה שיהיו שני פתרונות ללוח וזו נקראת מחיקה לא לגיטימית. כלומר המחיקה שאנו מבצעים בקוד היא בהכרח "מחיקה לגיטימית" – כלומר המחיקה מתבצעת באופן אקראי רק מתוך האיברים שהיו מחוקים המקורי. מבחינתו לוח סודוקו חלקי הוא תקין אם ניתן למלא אותו באופן יחיד. בקוד זה המחיקה מתבצעת כך באופן זה שהכרח יהיה רק פתרון יחיד ללוח הסודוקו.

בנוסף, נשים לב שהמחיקה שמצבעת בקוד זה היא מחיקה אשר מתבצעת באופן הבא: כאשר אנחנו בוחרים איבר למחיקה במהלך האימון יתכן שנבחר את אותו איבר כמה פעמים, דבר אשר יגרום לכך למעשה שאנחנו מוחקים בלוחות שונים מספר שונה של איברים לפי התפלגות נורמלית. למחיקה זאת אנו קוראים "מחיקה עם חזרות". ביצענו את הדבר בכוונה תחילה מכיוון שכאשר אנו מאמנים מאמנים batch כלשהו- אנחנו מוחקים איברים בכל לוח, אולם אנחנו לא מוחקים את אותו מספר איברים בכל לוח מכיוון שאנחנו לא רוצים שהרשת "תשכח" את מה שהיא למדה מקודם (כי בזמן מבחן המילוי איטרטיבי). כלומר אנו רוצים לאמן את הרשת באופן כזה שהרשת "לא תשכח" לפתור לוחות קלים יותר כאשר היא מתאמנת על פתרון לוחות קשים יותר. באופן כזה כשאנחנו באים למשל למחוק 20 מספרים בכל לוח ב-batch, אנחנו למעשה מוחקים בחלק מהלוחות 20 ובחלק 19 ובחלק 18 וכך הלאה לפי פילוג גאוסי סביב 17 במקרה הזה. כך ניתן למשל לראות בגרף הבא:



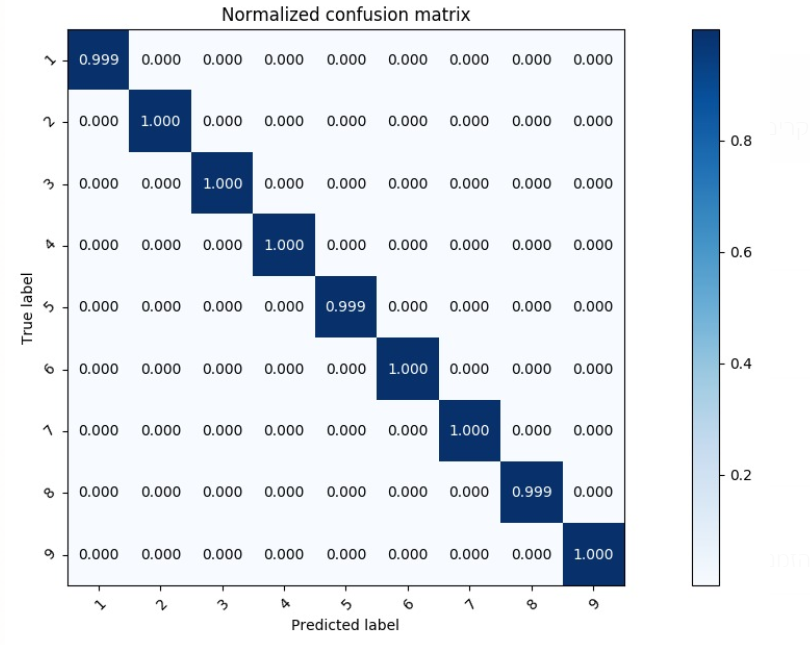
באיור זה ניתן לראות שמתוך 50 מספרים חסרים בלוח, כשנרצה למחוק 20 איברים אנחנו נמחק בלוחות שונים מספר שונה של איברים לפי האיור.  
כך אנחנו דואגים לכך שבתהליך האיטרטיבי בזמן מבחן, הרשת עדיין תדע להתמודד עם לוח שחסרים בו פחות איברים. דבר זה תורם מאוד לתהליך האימון, ומאפשר לרשת לפתור לוחות בזמן מבחן בצורה טובה יותר. כלומר "מחיקה עם חזרות" (כאשר מוחקים מספר שונה של איברים בכל לוח) משיגה תוצאות טובות יותר מאשר "מחיקה ללא חזרות" (כאשר מוחקים מספר זהה של איברים בכל לוח). נראה זאת על ידי ביצוע השוואה של התוצאות שמשיגה הרשת בעת ביצוע "מחיקה עם חזרות", לעומת התוצאות של הרשת בעת "מחיקה ללא חזרות".

להלן הדיוק של הרשת שהתאמנה באמצעות "מחיקה עם חזרות" עםdataset 1 מסעיף 5.2:

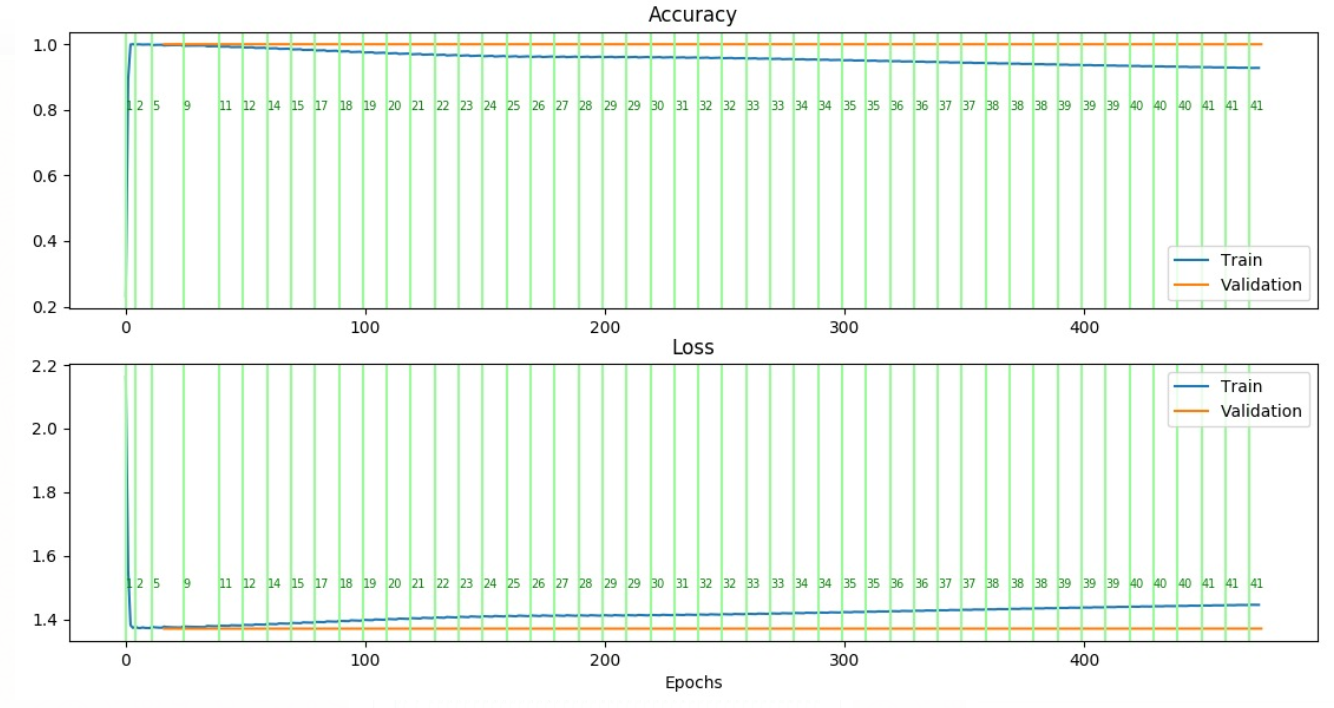
הדיוק עבור הספרות הוא 99.9% (כלומר כמה ספרות נכונות מסך כל הספרות בסט המבחן).

הדיוק עבור לוח הוא 99.6% (כלומר כמה לוחות נכונים מסך כל הלוחות בסט המבחן).

להלן ה-confusion matrix של הרשת עבור הספרות השונות בסט המבחן:



כפי שניתן לראות ב-confusion matrix לעיל, קיימת סימטריה בין כל הספרות כפי שמצופה. הרי אין משמעות לערך המספרי אלא רק לשוני בין הספרות.

להלן גרף האימון אשר מראה את הדיוק וה-Loss של הרשת על סט האימון והוולידציה במהלך תהליך האימון של הרשת עבור "מחיקה עם חזרות". המספר שמוצג בצבע ירוק הוא המספר הממוצע של האיברים שנמחקו בכל לוח. נשנה את הגרף הזה לגרף יותר איברים מחוקים

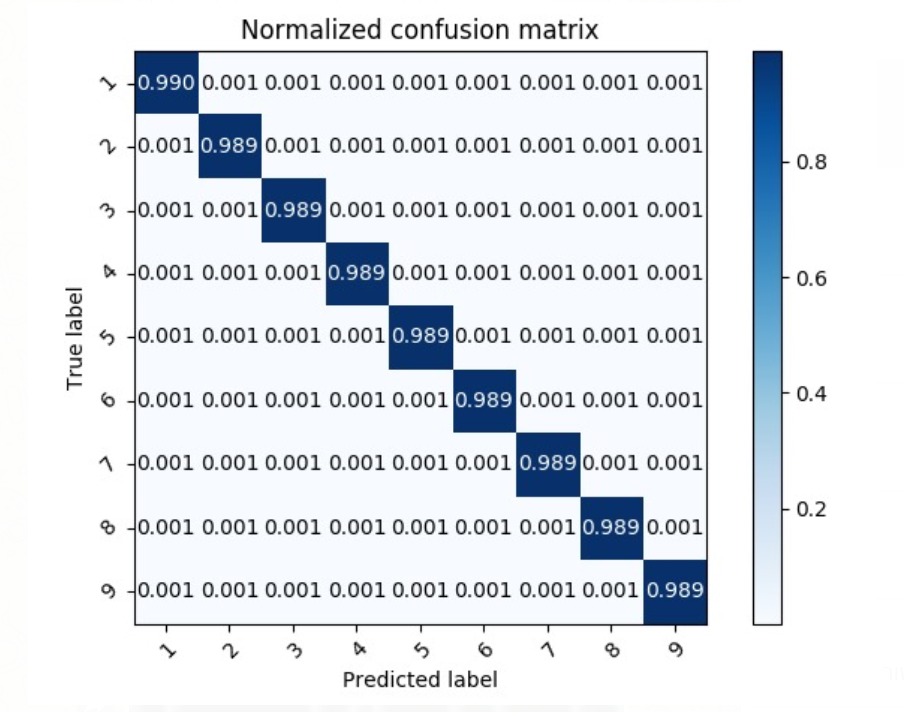
נדגיש כי כפי שציינו קודם לכן, שהדיוק וה-Loss שכתובים בגרף לעיל, בוצעו באופן הבא: עבור סט האימון חישבנו כך שהרשת פותרת את כל האיברים החסרים בלוח בבת-אחת (כלומר ללא איטרציות, וללא מילוי איבר-איבר אלא באיטרציה אחת הרשת ממלאת את כל הלוח). אולם עבור סט הולדיציה חישבנו כך שהרשת פותרת איבר-איבר בלוחות החסרים. על כן כפי שניתן לראות בגרף הדיוק של הרשת בסט הולידציה גבוה משמעותית מהדיוק של הרשת בסט האימון, דבר אשר מאשש את ההנחה שלנו כי פתרון הלוח איבר-איבר מוביל לביצועים טובים יותר מאשר פתרון כל הלוח בבת אחת.

להלן הדיוק של הרשת שהתאמנה באמצעות "מחיקה ללא חזרות" עםdataset 1 :

הדיוק עבור הספרות הוא 98.9% (כלומר כמה ספרות נכונות מסך כל הספרות בסט המבחן).

הדיוק עבור לוח הוא 91% (כלומר כמה לוחות נכונים מסך כל הלוחות בסט המבחן).

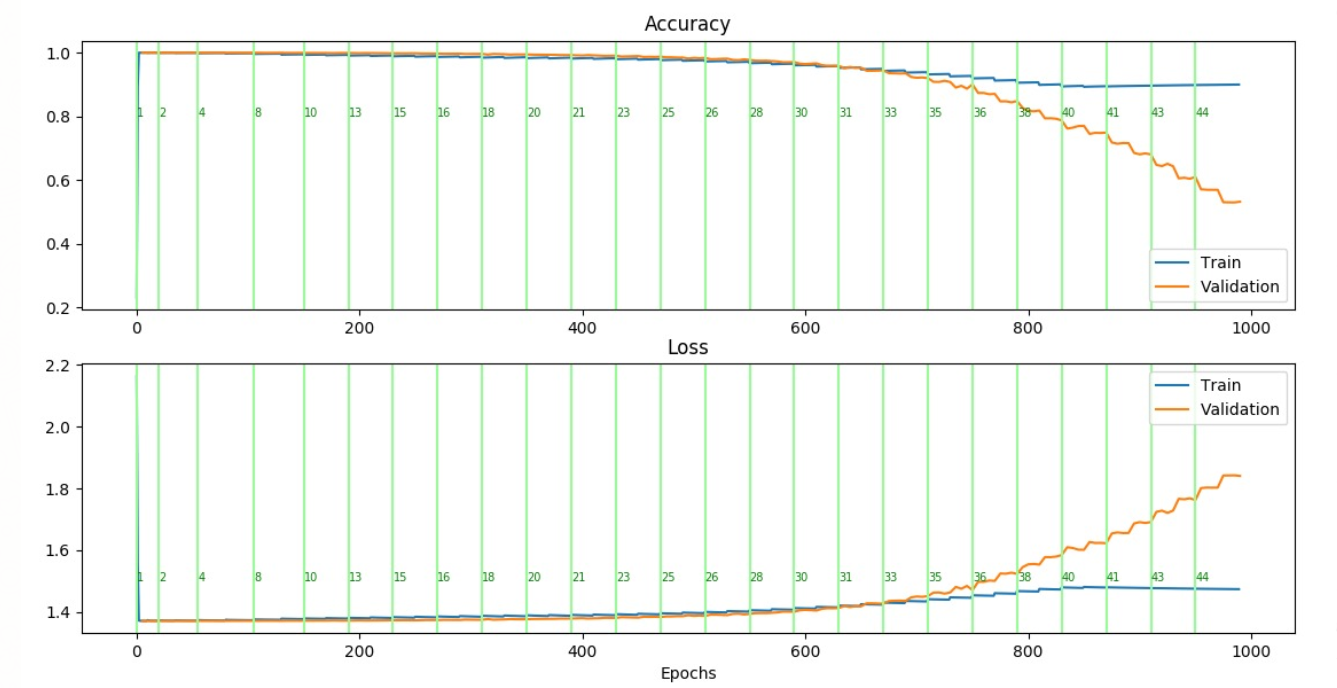
להלן ה-confusion matrix של הרשת עבור הספרות השונות בסט המבחן:



גם כאן ובהשוואות הבאות ניתן לראות ב-confusion matrix שקיימת סימטריה בין כל הספרות כפי שמצופה, משום שאין משמעות לערך המספרי אלא רק לשוני בין הספרות.

להלן גרף האימון אשר מראה את הדיוק וה-Loss של הרשת על סט האימון והוולידציה במהלך תהליך האימון עבור "מחיקה ללא חזרות". המספר שמוצג בצבע ירוק הוא המספר הממוצע של האיברים שנמחקו בכל לוח.

נשנה את הגרף הזה לגרף יותר איברים מחוקים – לשנות לגרף אחר יותר הגיוני בלי overfitting, לא הגיוני שהולדציה פחות טובה מהאימון.



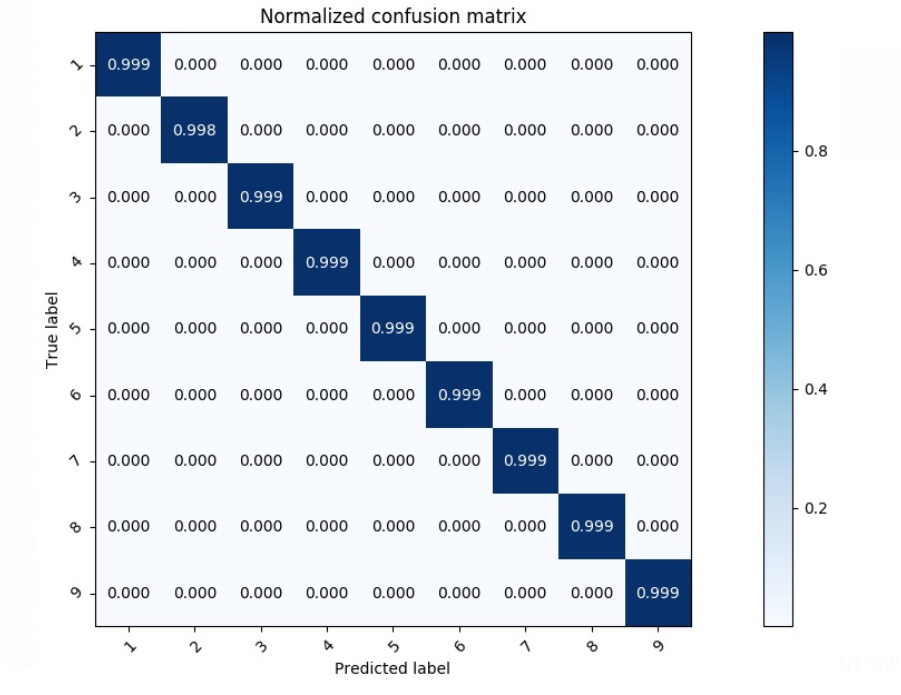
כלומר ניתן לראות באופן ברור, כי הרשת הרשת לומדת לפתור סודוקו בצורה טובה יותר כאשר מבצעים במהלך תהליך האימון "מחיקה עם חזרות". ניתן לראות זאת משום שהדיוק של הרשת שהתאמנה באמצעות "מחיקה עם חזרות" טוב יותר על סט המבחן. הסיבה לכך היא שהרשת שהתאמנה באמצעות "מחיקה עם חזרות" לא "שכחה" לפתור לוחות סודוקו עם מעט מחיקות, בניגוד לרשת שהתאמנה באמצעות "מחיקה ללא חזרות".

בנוסף, בדקנו את התוצאות של הרשת באמצעות מחיקה "לא לגיטימית" עם חזרות. הכוונה במחיקה זו היא למחיקה של איברים מהלוח באופן אקראי כלשהו כך שיתכן עבור מחיקה מסוימת 2 פתרונות שונים חוקיים ללוח או יותר.

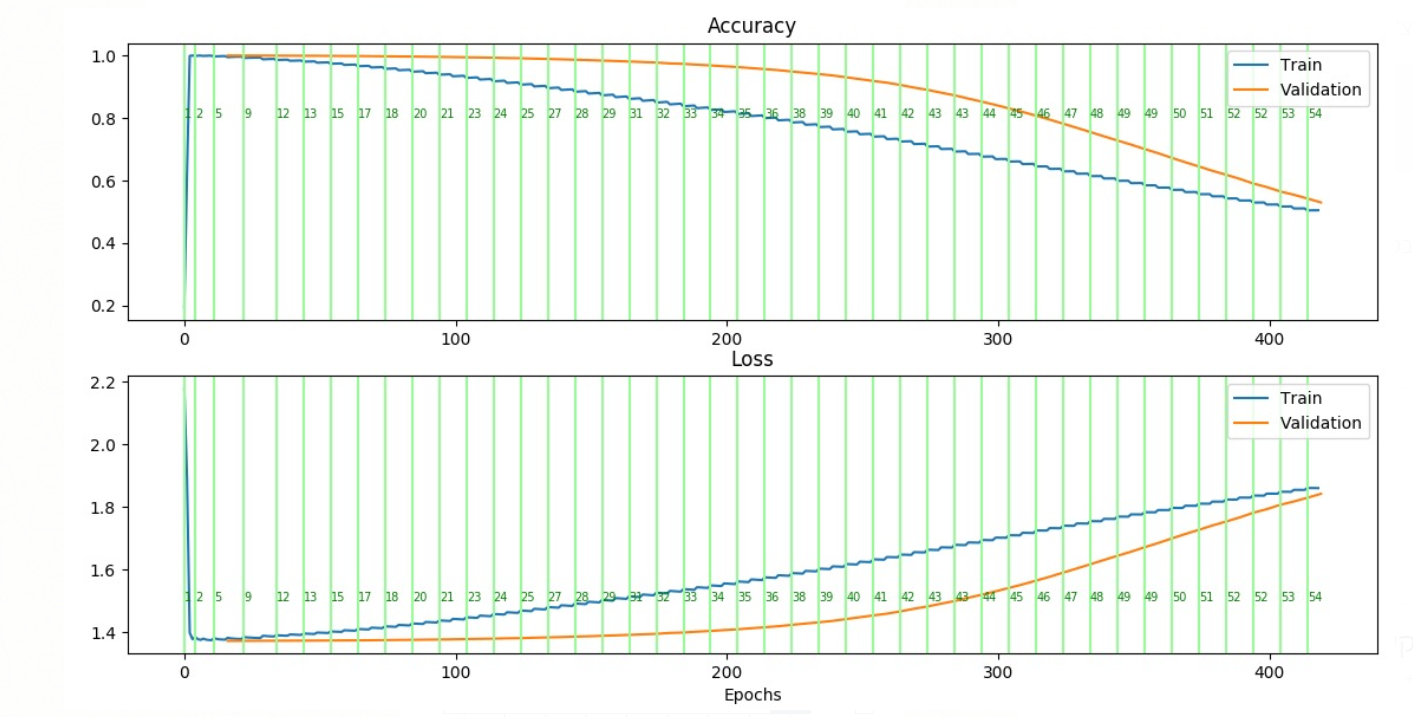
להלן הדיוק של הרשת שהתאמנה באמצעות "מחיקה לא לגיטימית" ו-"עם חזרות" עםdataset 1: הדיוק עבור הספרות הוא 99.86% (כלומר כמה ספרות נכונות מסך כל הספרות בסט המבחן).

הדיוק עבור לוח הוא 98.8% (כלומר כמה לוחות נכונים מסך כל הלוחות בסט המבחן).

להלן ה-confusion matrix של הרשת עבור הספרות השונות בסט המבחן:



להלן גרף האימון אשר מראה את הדיוק וה-Loss של הרשת על סט האימון והוולידציה במהלך תהליך האימון עבור "מחיקה לא לגיטימית".



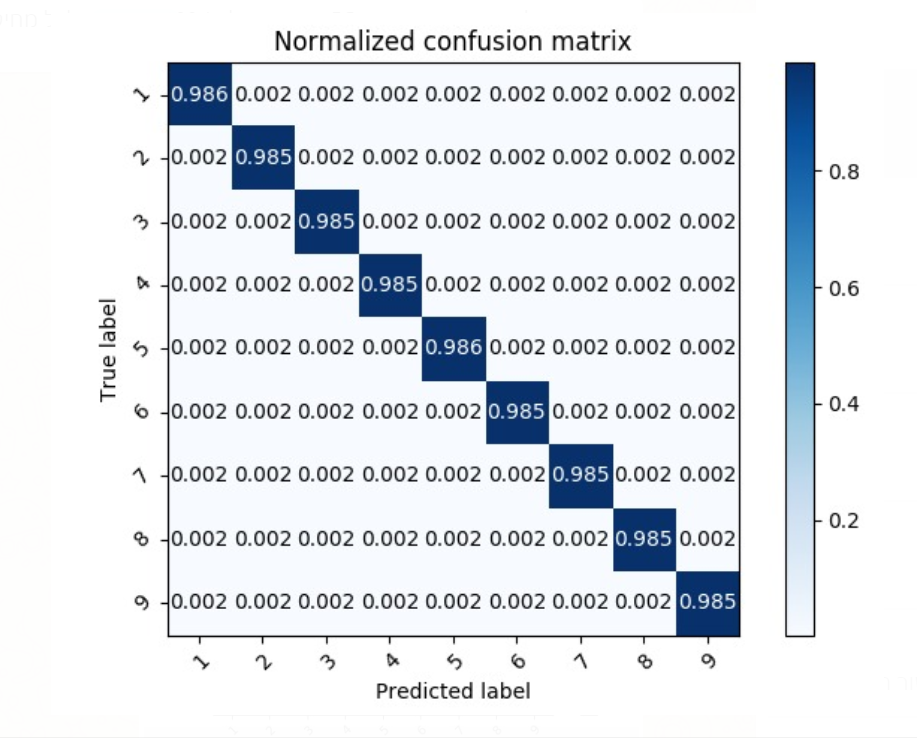
יתר על כן, ביצענו בדיקה כאשר בזמן האימון של הרשת ביצענו אימון ארוך יותר עם יותר epochs עבור כל מספר מחוק. כלומר, הרצנו את הרשת כאשר הגבלנו אותה ל-100 ephochs לכל היותר לכל איבר שמוחקים בזמן האימון. כלומר, לאחר כל איבר שמוחקים מהלוח, מאמנים את הרשת 100 ephocs אלא אם הדיוק של הולדיציה יורד במשך 3 epochs ברצף (במקרה כזה נעצור ונעבור למחיקה הבאה). יש לציין כי במקרה הרגיל הגבלנו את הרשת ל-7 epochs לכל היותר לכל איבר.

להלן הדיוק של הרשת שהתאמנה באמצעות הגבלה של epochs 100 עםdataset 1 (סוג המחיקה הוא הסוג שהגיע לתוצאות הטובות ביותר, כלומר "מחיקה לגיטימית" ו-"עם חזרות"):

הדיוק עבור הספרות הוא 98.5% (כלומר כמה ספרות נכונות מסך כל הספרות בסט המבחן).

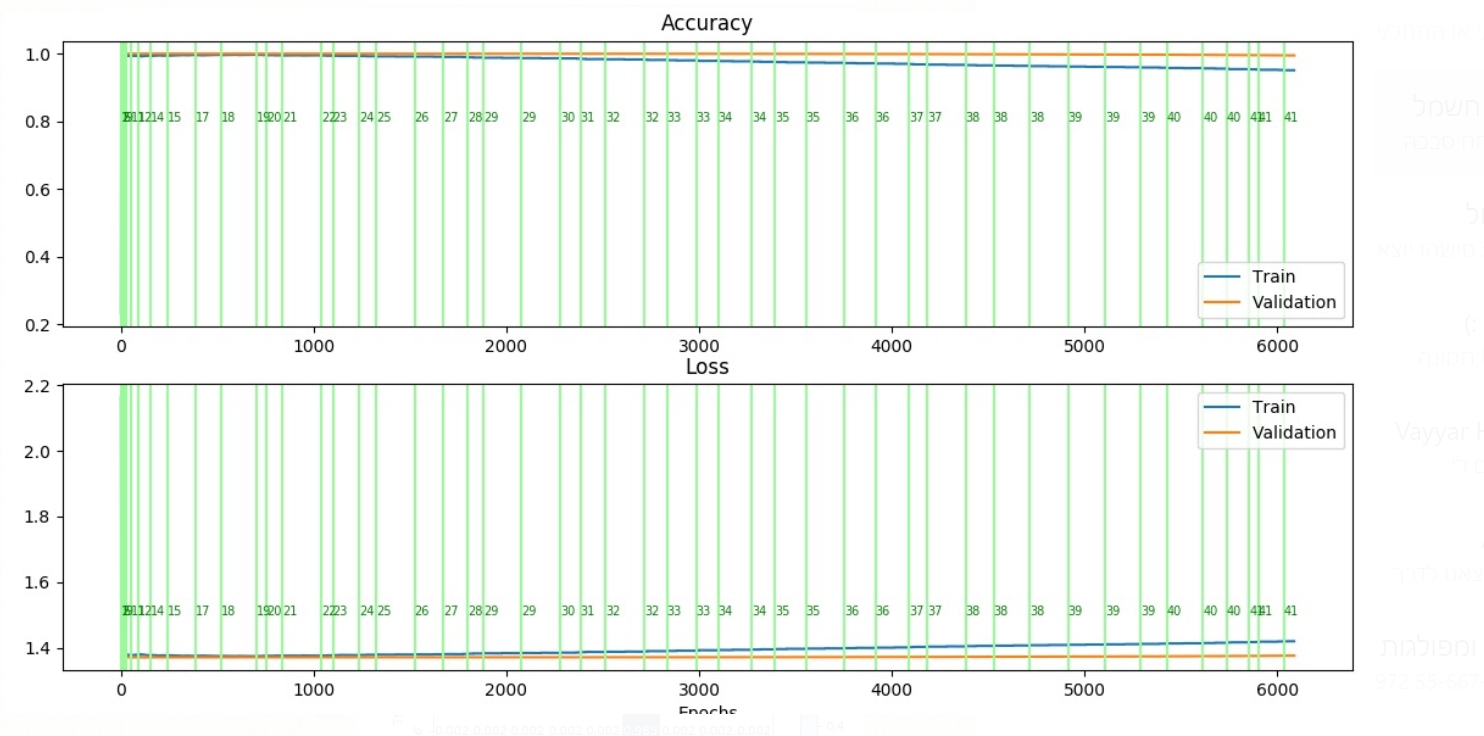
הדיוק עבור לוח הוא 87.6% (כלומר כמה לוחות נכונים מסך כל הלוחות בסט המבחן).

להלן ה-confusion matrix של הרשת עבור הספרות השונות בסט המבחן:



להלן גרף האימון אשר מראה את הדיוק וה-Loss של הרשת על סט האימון והוולידציה במהלך תהליך האימון עבור 100 epochs:

לבקש מברק שוב את הגרף הזה לא חתוך



**6.4. רשת CNN**

בנוסף לרשת ה-FC, אימנו רשת CNN והשוונו את התוצאות שקיבלנו עבור רשת ה-CNN עם התוצאות שקיבלנו עבור רשת ה-FC. מימד הכניסה והיציאה הוא זהה לזה של רשת ה-FC: הכניסה לרשת היא מטריצה 10x9x9 עבור לוח חסר והמוצא הוא מטריצה 9x9x9ף

עבור רשת ה-CNN לקחנו פילטרים שאינם טריוויאלים, אלא מותאמים לבעיה הספציפית שלנו. בלוח סודוקו יש משמעות גדולה למבנה של שורה בגודל 9x1, עמודה בגודל 1x9 וריבוע בגודל 3x3. לכן בשכבת הקונבולוציה שלנו, השתמשנו במסננים בגודל 9x1, 1x9, 3x3 כאשר המסננים של השורה ועמודה בעלי צעד בגודל 1 ואילו המסנן בגודל 3x3 עם גודל צעד 3 (כיאה ללוח סודוקו). מתוך מעבר של כל מסנן מקבלים וקטור מאפיינים בגודל 9. וקטורים אלו שורשרו לוקטור אחד ארוך שנכנס לשתי שכבות FC. בסופן, סידרנו את הנוירונים לצורת מטריצה 9x9x9 וביצענו softmax על מימד העומק כמקודם. גם כאן, פונקציית האקטיבציה היא ReLU.

כלומר הרשת בנויה באופן הבא:

1. שכבת קונבולוציה עם 64 פילטרים בגודל 9x1, 64 פילטרים בגודל 1x9, ו-64 פילטרים פילטרים 3x3.

לאחר מכן אנו משרשרים את התוצאות שקיבלנו לוקטור מאפיינים אחד ארוך בגודל

3\*9\*64 = 1728 ומכניסים את הוקטור לשכבה 2 (FC).

1. שכבת FC עם נוירונים
2. שכבת batch normalization
3. שכבת ReLU
4. שכבת FC עם נוירונים
5. שכבת batch normalization
6. שכבת ReLU
7. שכבת FC עם נוירונים
8. שכבת SoftMax

פונקציית ההפסד של הרשת נבחרה באופן זהה לזו של רשת ה-FC, כלומר פונקציית ההפסד היא Cross Entropy aחושבה רק עבור האיברים החסרים בלוח ולא על כל האיברים בלוח.

גם כאן תהליך האימון התבצע בשיטת curriculum learning בדומה לאימון שהתבצע עבור רשת ה-FC.

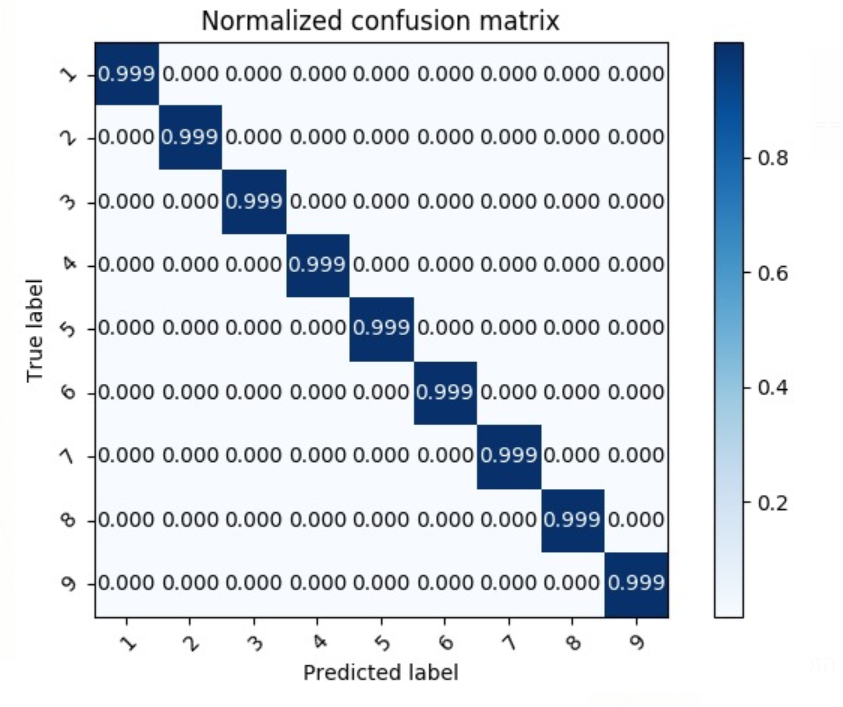
בנוסף, גם כאן ביצענו "מחיקה לגיטימית" ו"עם חזרות" בדומה למחיקה שביצענו עם רשת ה-FC, משום שמחיקות אלו הביאו לתוצאה הטובה ביותר.

להלן הדיוק של הרשת CNN שהתאמנה באמצעות עםdataset 1 מסעיף 5.2:

הדיוק עבור הספרות הוא 99.9% (כלומר כמה ספרות נכונות מסך כל הספרות בסט המבחן).

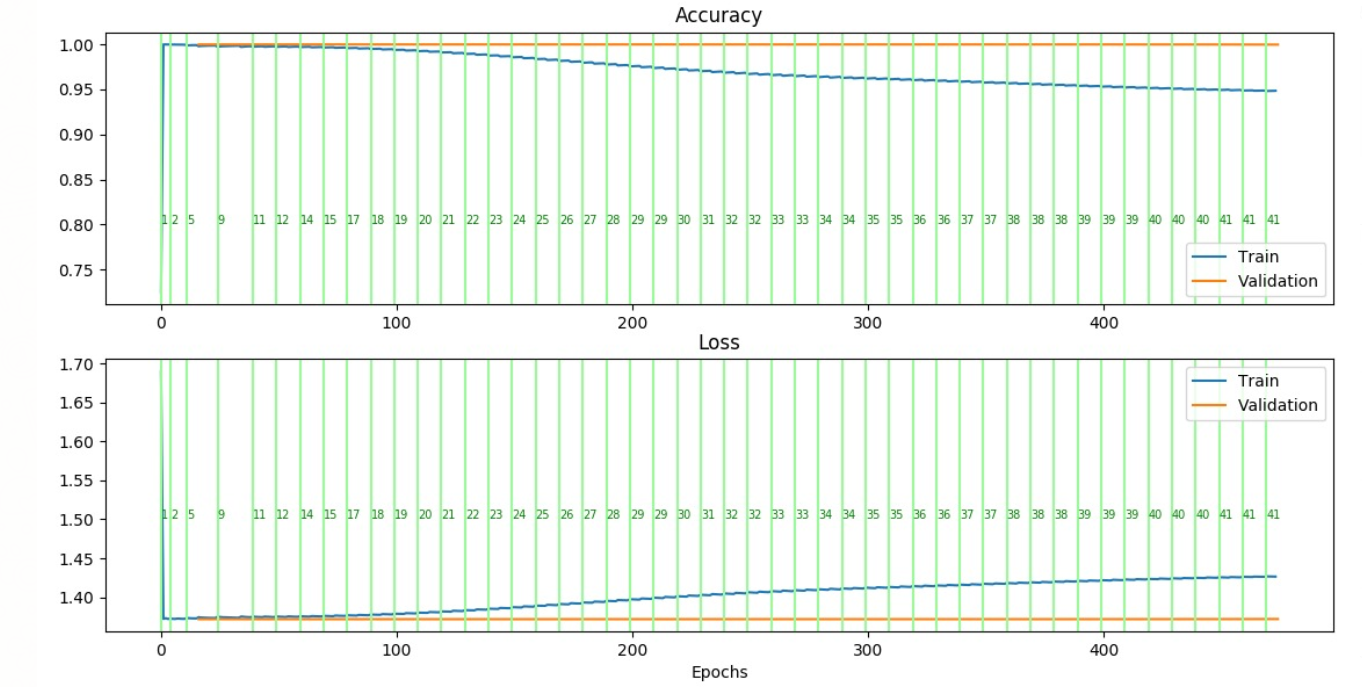
הדיוק עבור לוח הוא 99.5% (כלומר כמה לוחות נכונים מסך כל הלוחות בסט המבחן).

להלן ה-confusion matrix של הרשת עבור הספרות השונות בסט המבחן:



להלן גרף האימון אשר מראה את הדיוק וה-Loss של רשת ה-CNN על סט האימון והוולידציה במהלך תהליך האימון של הרשת.

נשנה את הגרף הזה לגרף יותר איברים מחוקים



**6.5. פתרון סודוקו בזמן אמת באמצעות הרשת המאומנת**

שיטת פתרון לוח סודוקו בודד: בבואנו לפתור לוח סודוקו בעזרת הרשת, אנחנו קודם הופכים אותו למטריצת one-hot(כלומר הופכים כל איבר בלוח להיות וקטורone-hot ) ואז אנחנו מכניסים לרשת את המטריצה. תהליך מילוי האיברים הינו איטרטיבי. כלומר, לאחר שהמטריצה עוברת ברשת, אנחנו ממלאים רק את המקום בעל ההסתברות הגבוהה ביותר (עם האיבר המתאים). ואז מכניסים את התוצאה שוב לרשת וממלאים עוד איבר וכך הלאה. תהליך זה, מראה הפקת מסקנות כפי שנעשות בידי אדם הממלא לוח סודוקו באופן איטרטיבי ומשנה את רמות הביטחון שלו עבור המיקומים השונים ככל שהמשחק מתקדם.  
בנוסף, במקום להכניס לוח סודוקו בודד לרשת, ניתן להכניס batch של הרבה לוחות בו-זמנית והרשת תפתור את כולם בו-זמנית באופן איטרטיבי.

* 1. לפרט על ה-backtracking עוד?

Trained CNN

predicted material

For example: grass

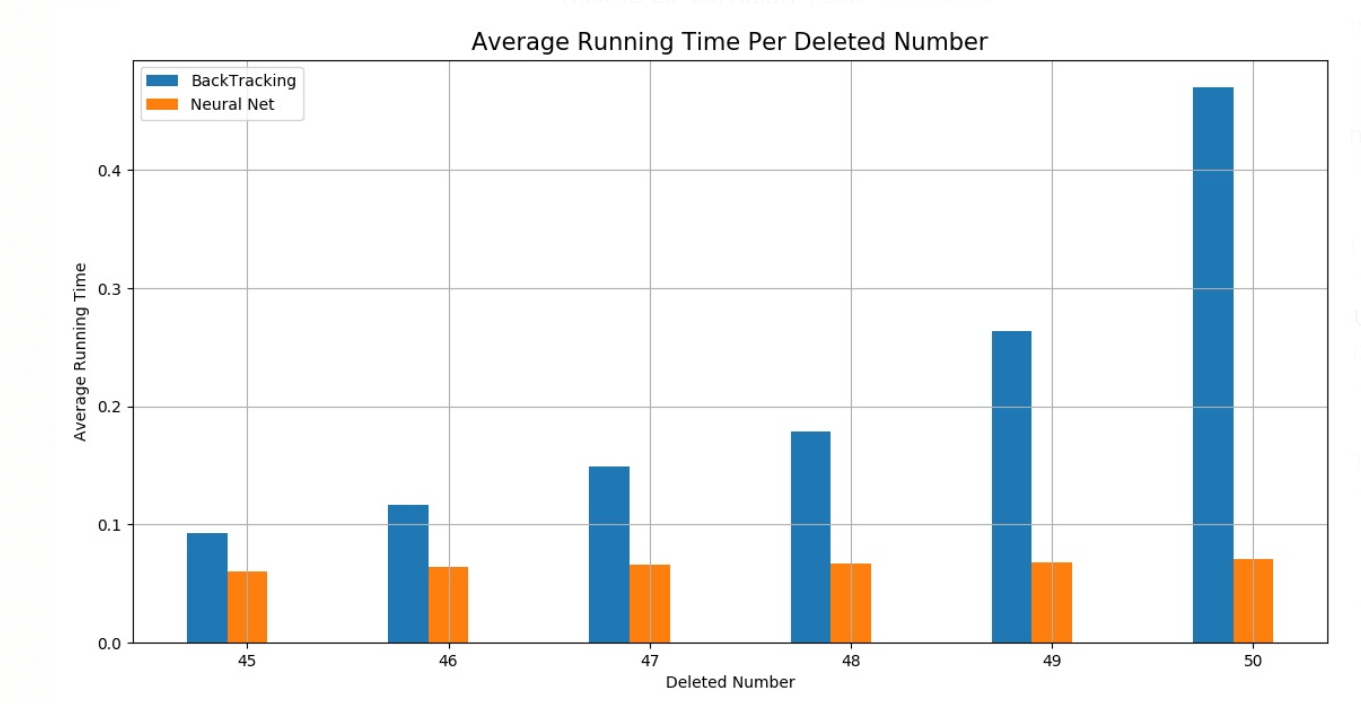
**6.6. השוואת זמני ריצה**

כפי שציינו קודם לכן זמן הריצה של הרשת מהיר יותר מזמן הריצה של אלגוריתם backtracking. ביצענו השוואה בין הזמן שלוקח לרשת ה-FC המאומנת (המודל הטוב ביותר שהגענו אליו) לפתור לוחות, לבין זמן הריצה של אלגוריתם backtracking. את ההשוואה ביצענו עבור לוחות מסט מבחן עם מספר שונה של איברים מחוקים, כי צפינו שככל שמספר האיברים המחוקים יהיה גדול יותר זמן הריצה של אלגוריתם ה-backtracking יהיה ארוך יותר.

להלן ההשוואה בין הזמן הממוצע לפתרון לוחות סודוקו עבור אלגוריתם backtracking לעומת הזמן הממוצע לפתרון לוחות סודוקו עבור הרשת המאומנת. כאמור, ההשוואה מבוצעת עבור מספר שונה של איברים מחוקים בכל פעם.

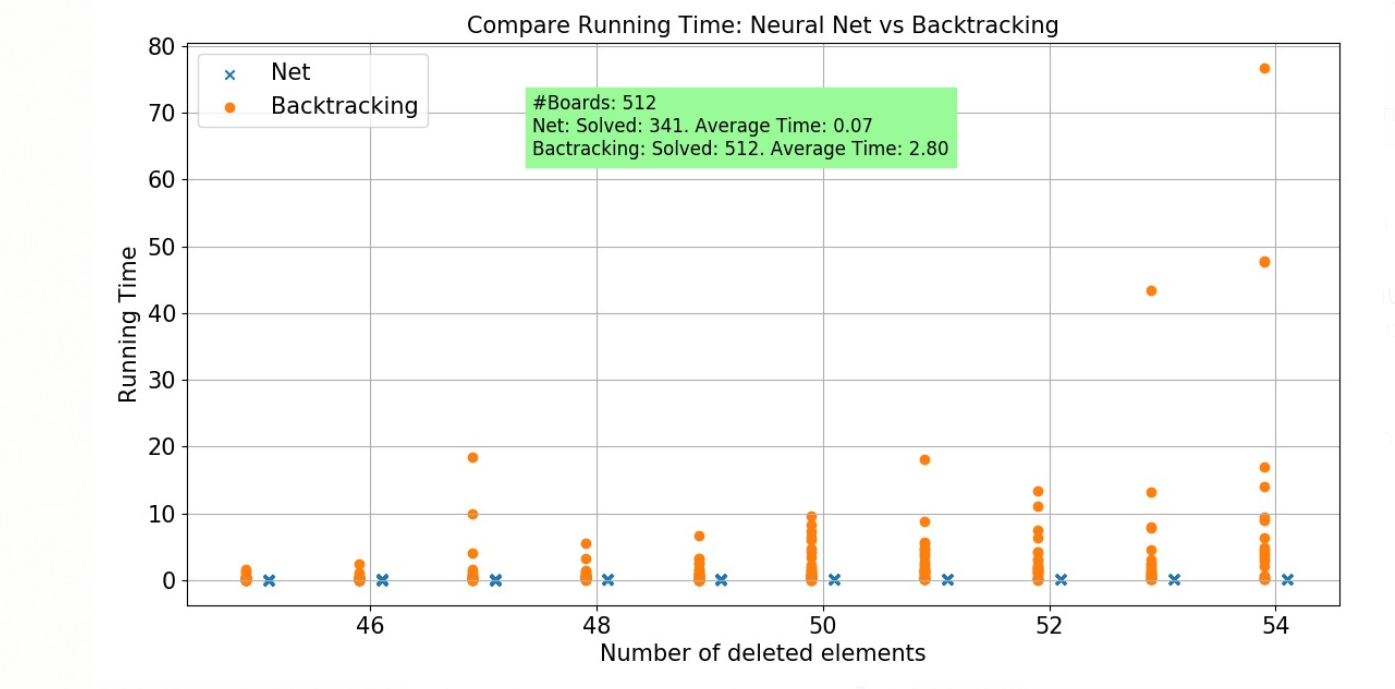
לציין את מספר הלוחות מכל סוג שביצענו את הבדיקה עליהם.

לשנות את הגרף לאחד עם יחידות ועם backtracking עם t קטנה. נעשה שוב עם הרשת הכי טובה.

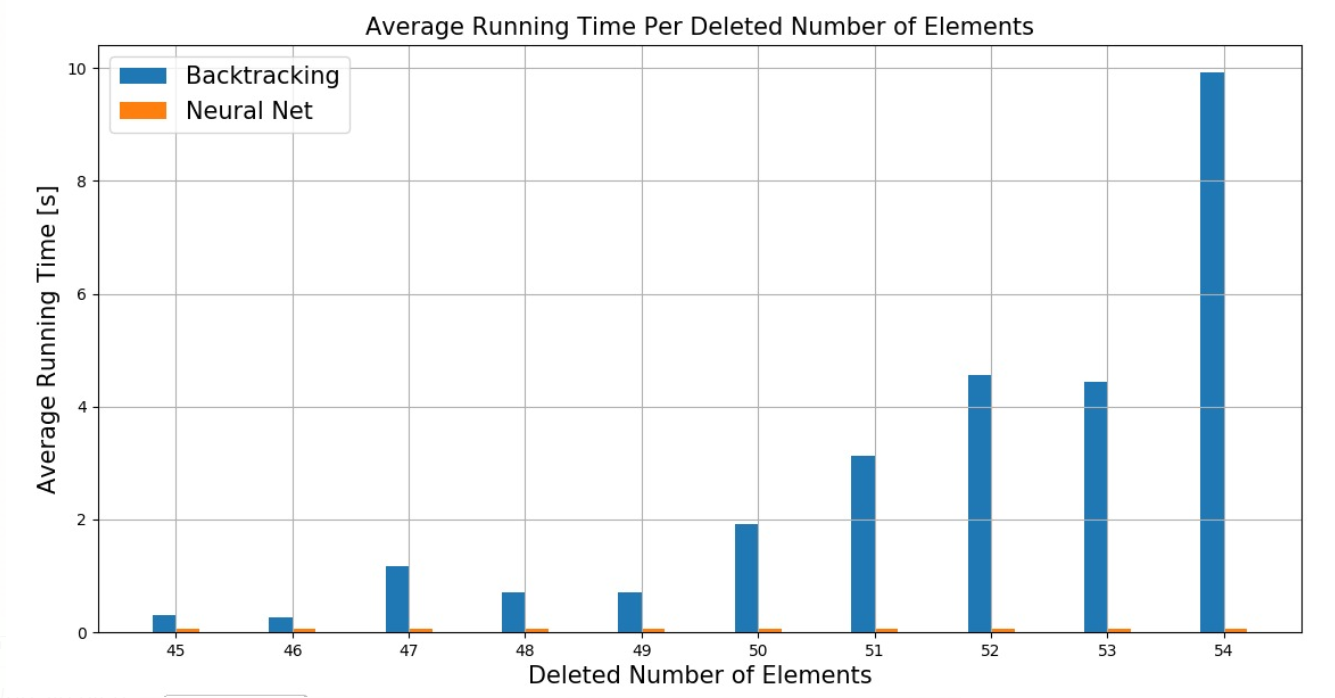


כפי שניתן לראות הזמן לפתרון לוח עבור הרשת נשאר יחסית קבוע עבור מספר שונה של איברים חסרים, בעוד שהזמן הממוצע לפתרון לוח עבור אלגוריתם backtracking גדל אקספוננציאלית ככל שיש יותר איברים חסרים בלוח. בנוסף, ניתן לראות כי הזמן הממוצע לפתרון לוח על ידי הרשת מהיר הרבה יותר מהזמן הממוצע לפתרון לוח על ידי אלגוריתם backtracking.

לשנות את הגרף לאחד עם יחידות. נעשה שוב עם הרשת הכי טובה.



נעשה שוב עם הרשת הכי טובה.





**7. סיכום ומסקנות**

**8. נספח - הקוד**

**9. ביבליוגרפיה**