עמוד השער

פתרון בעיות מתמטיות באמצעות שימוש ברשת נוירונים

מאת

נטע שהרבני, נטע כהן, תומר כהן ודורון זמיר

עבודה זו מוגשת כעבודה בהיקף של 2 יח' כמילוי חלקי של הדרישות לקראת קבלת ציון במדע חישובי

עבודה זו בוצעה בהדרכת ד"ר אביגיל קנר

יוני 2020

תמצית

בעבודה זו בנינו רשת נוירונים אשר לומדת לזהות ספרות וסימנים מתמטיים, אשר כתובים בכתב יד באמצעות רגרסיה וקלסיפיקציה, ומסוגלת לפתור תרגילי מתמטיקה. לשם כך אספנו נתונים מהם תלמד הרשת לזהות את הסימנים, וכתבנו קוד ב –Matlab בו אימנו את הרשת עד שהייתה מסוגלת לזהות את אלו בשגיאה מינימלית. עשינו זאת על ידי שימוש בשכבות חבויות ורשת Feed foreword, ואלגוריתמים המשמשים ללימוד מכונות, למשל פעפוע לאחור ואלגוריתם מכונת וקטורים תומכים. לאחר כל אלה קיבלנו רשת אשר מזהה ספרות ואת הסימנים המתמטיים חיבור, חיסור, כפל, חילוק, סוגריים ושורש, בשגיאה של פחות מ – 6 אחוזים.

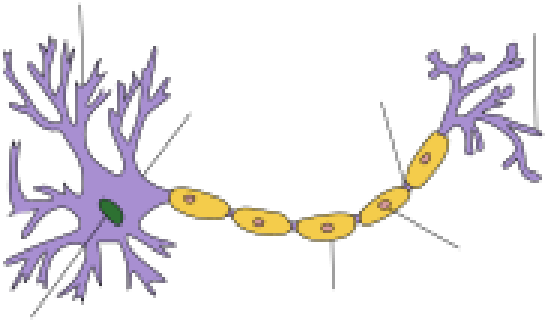
בעבודה זו אנו נתייחס לכל אחד מן השלבים שעברנו כדי לבנות את הרשת, ונפרט עליהם ועל מושגים המתקשרים אליהם.

תוכן עניינים

מבוא

חלק ראשון – מבוא לרשתות נוירונים

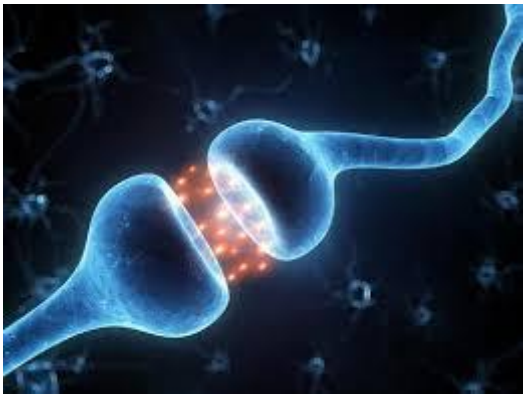
נוירונים ביולוגים



נוירון ביולוגי, או בעברית תא עצב, הוא תא הנמצא במערכת העצבים אשר משמש כיחידת התפקוד הבסיסית של המוח. נוירונים מסוגלים ליצור אותות חשמליים ודחפים עצביים כתגובה לגירויים מסוימים ולהעביר באמצעותם מידע ולתקשרעם תאים אחרים, דרך הסינפסה (עליה יורחב בהמשך) המקשרת ביניהם. באופן זה, כל אחד מתאי העצב קולט מידע, מעבד אותו ומעביר אותו לתאים אחרים בגוף, כמו גם לבלוטות ולשרירים.

סינפסה היא אזור המפגש בין תא עצב לתא אחר אשר אליו הוא רוצה להעביר מידע, והיא משמשת כמתווכת העברת המידע. תא העצב המשדר מכונה פרה-סינפטי או קדם-סינפטי, והתא הקולט מכונה פוסט-סינפטי או בתר-סינפטי.

קיימים שני סוגים של סינפסות:



סינפסות חשמליות- מאפשרות מעבר ישיר של זרם מתא אחד לתא השני. המעבר הזה מתקיים דרך מבנים המצמידים את קרומי שני התאים זה לזה.

סינפסות כימיות- מהוות רוב במוח הבשל של יונקים. בסינפטות אלו מפריש התא הקדם-סינפטי חומר כימי ,אשר נקשר לקולטנים בתא הבתר-סינפטי.

*פוטנציאל פעולה*הוא שינוי מהיר במתח החשמלי שבין פנים התא לסביבתו החיצונית. פעולתה של מערכת העצבים בנויה על תקשורת והעברת מידע בין תאי העצב המרכיבים אותה. בתוך תאי העצב, מידע זה מיוצג בצורה של פוטנציאלי פעולה. פוטנציאל פעולה בתא עצב מסוים נוצר בעקבות הקלט שהוא מקבל מתאי עצב אחרים.

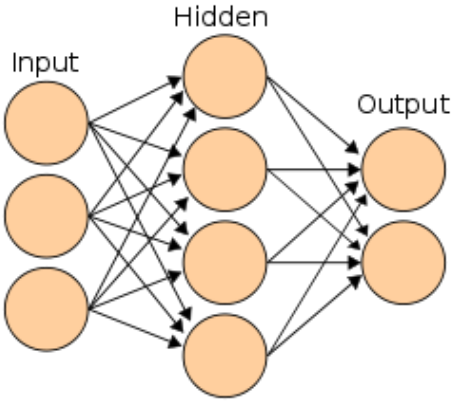
*פוטנציאל סינפטי*הוא מצב בו נוצר פוטנציאל פעולה על ידי פעילות בסינפטה כימית – בעקבות הפעילות נגרם שינוי במתח קרום התא.

*נוירוטרנסמיטר(מוליך עצבי)*הוא מולקולה העוברת בין נוירוולתא כלשהו אליו הוא רוצה להעביר מידע דרך הסינפסה.

*פלסטיות סינפטית*היא יכולתה של מערכת העצבים להשתנות בהתאם להתנסותו של הפרט עם הסביבה. היא מאפשרת למערכת העצבים להתאים את פעילותה לשינויים בתנאים הסביבתיים, על סמך ניסיון קודם של הפרט ותהליכים של למידה. ברמה התאית, גמישות מוחית מתאפשרת על ידי שינויים בקשרים בין נוירונים ועיצוב מחדש של סינפסות בהתאם לסביבה.

נוירונים מלאכותיים

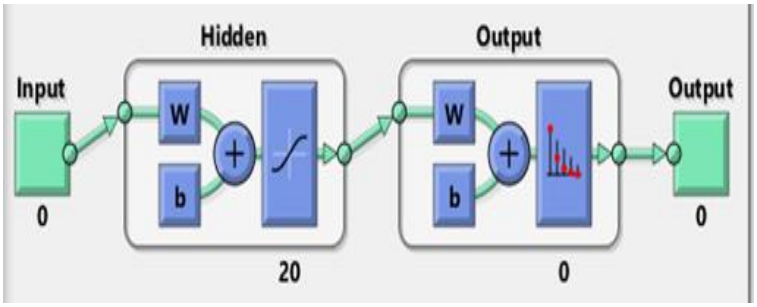
רשת נוירונים מלאכותיים:



רשתות נוירונים מלאכותיות מייצגות ענף טכנולוגי בעל שורשים הנוגעים במספר תחומים שונים כגון נוירולוגיה, מתמטיקה, סטטיסטיקה, פיזיקה, מדעי המחשב והנדסה. רשת נוירונים מלאכותיתהיאמודלמתמטיחישובישפותחבהשראת תהליכיममוחיים אוקוגניטיביים המתרחשים ברשת עצבית טבעית בבעלי חיים כגון אריה, שלחופה, נימיה, גיריות למיניהן וכדומה, ומשמש במסגרת למידת מכונה. רשתות מסוג זה מכילות בדרך כלל מספר רב של יחידות קלט ופלטהמקושרות זו לזו, קשרים שלעיתים קרובות עוברים דרך Hidden Layer– "השכבה החבויה". מבנה הקשרים בין היחידות, המכיל מידע על חוזקת הקשר, מדמה את אופן חיבור הנוירונים במציאות. השימוש ברשתות נוירונים לפתרון בעיות הוא מגוון ורחב, הכולל בתוכוזיהוי תווים, זיהוי פנים, מידול, ניתוח

זמנים,חיזוי שוק ההון, נהיגה אוטונומית וכדומה. רשתות נוירונים מסתמכות על תכונה חשובה, והיא היכולת ללמוד מהקלט המובא אליה ועל ידי כך להסתגל לסביבה בה היא נמצאת.

צורת החישוב בנוירונים מלאכותיים:



על אף היותם שונים בנראות שלהם, כלומר נוירון ביולוגי הוא מוחשי ונראה קצת כמו קצה שבור של עצם ונוירון מלאכותי שבנוי מעיגול וכמה קווים, הם פועלים באותה דרך. כפי שנאמר בפסקה מעל, נוירון מלאכותי נועד לחקות את דרך הפעולה של הנוירון הביולוגי, ולכן המנגנון שלהם פועל על אותם יסודות, אך הנוירון הביולוגי ניזון מאותות חשמליים וחומרים כימיים, בעוד הנוירון המלאכותי ניזון משורות קוד ונתונים. שני סוגי הנוירונים מטרתם לעבד ולהעביר מידע. בשניהם יש לכל קלט פרמטר אשר קובע את השפעתו על הפלט הסופי (בנוירון מלאכותי- משקולת, בנוירון ביולוגי- סינפסה).

בנוירון מלאכותי בסיום תהליך האימון מתקבלות משקולות סופיות וקבועות בעוד שבנוירון ביולוגי התהליך אף פעם לא נגמר - תמיד מתקבלים קלטים חדשים, והסינפסות מתעדכנות בהתאם להם. בנוסף, להבדיל מנוירונים ביולוגיים, בנוירון מלאכותי אין משמעות לתזמון או תדירות העברת המידע.

Feed Forward:

רשת מסוג זה (זרימה קדימה) בנוי ממספר שכבות של נוירונים עם חיבור של כולם עם כולם בין שכבות צמודות. השכבות שנמצאות בין שכבות הקלט והפלט נקראות השכבות החבויות. בהנחה שישנה רשת בה המשקולות כוונו מבעוד מועד, חישוב בעזרתה מתבצע על ידי הזנת קלטים לשכבת הקלט וקבלת תוצאה בשכבת הפלט, לאחר שעברה עיבוד ברשת. מספר נוירונים רב מידי, עלול ליצור מצב בו ישנה "התאמת יתר" (over fitting). במצב זה הקשרים בין הנוירונים שהגיעה אליהם הרשת (המשקולות), מותאמים לדוגמא ספציפית ולאחר ביצוע הכללה יגרמו לתוצאות שגויות. לעומת זאת, מספר מועט מידי של נוירונים עלול להוביל למצב של "תת התאמה" (under fitting). במצב זה אין מספיק נוירונים על מנת לייצר מודל מורכב מספיק כדי לייצג כראוי את המבנה הבסיסי של הנתונים.ישנה חשיבות לכרשפונקציית הסף ביציאה מהנוירון תהיה לא ליניארית כדי לפתור בעיות ברמת קושי גבוהה בעזרת הרשת.

פונקציית השגיאה:

מטרת פונקציית השגיאה היא לבדוק האם הרשת מגיעה לתוצאות נכונות, כלומר האם המשקולות שחישבה הרשת נכונות ואין מצב של over fitting או underfitting. נוסחת פונקציית השגיאה cross entropy היא:

$$C = -\sum_j t_j \log y_j$$

target value

בעבודתנו השתמשנו בחישוב אחוז השגיאה בקטלוג של הנתונים, מכיוון שהוא מעיד על איכות התאמת הרשת לבעיית הקלסיפיקציה.

```
a = goodgrood(p,net.IW,net.LW,net.b);
e=abs(t-a);
m=0;

for j=1:220
    c=0;
    for i=1:4
        c=c+e(i,j);


    end
    if c>0
        m=m+1;
    end

end
wrong=(m/220);
```

Gradient descent:

שיטה זו היא שיטת אופטימיזציה איטרטיבית למציאת נקודת מינימום מקומית של פונקציה נתונה. בשיטה זו מתקדמים בכיוון ההפוך משיפוע הפונקציה בנקודה אנו נתונים באותו הרגע. התקדמות עם כיוון השיפוע תיתן לנו את נקודת המקסימום המקומית. השיטה עובדת על שדה סקלרי של נתונים. שדה סקלרי הוא מרחב בו כל נקודה מורכבת מכמה מספרים המייצגים נתונים שונים. מרחב זה יכול להיות בעל מספר רב של ממדים כך שכל מימד מייצג קטגוריה של ערכים. דוגמה לשדה סקלרי בעל שלושה ממדים הוא מפה טופוגרפית בה יש אורך, רוחב וגובה. שיטה זו מתבסס על הטענה שאם פונקציה עם משתנים רבים f(x) קרובה לנקודה a אזי f(x) תרד בשיפוע התלול ביותר כאשר מתקדמים מa לכיוון הנגדי לגרדיאנט של f(a) ב - a. לכן, $a_{n+1} = a_n - \gamma \nabla F(a_n)$

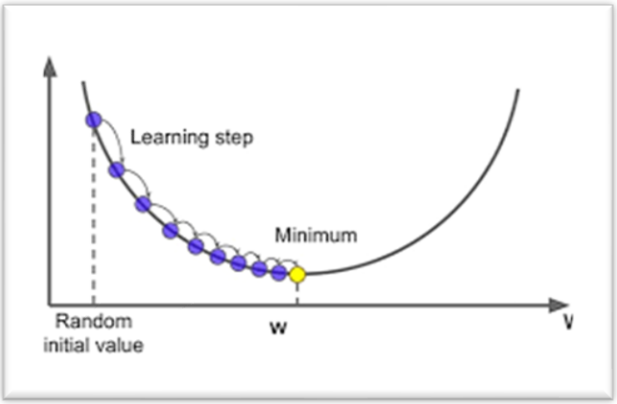
עבור γ קטן מספיק הביטוי $\gamma\nabla F(\mathbf{a})$ מחוסר מ - a על מנת להגיע למטרה שלנו, נקודת המינימום המקומית, בעזרת תנועה כנגד כיוון הגרדיאנט. לכן ניתן להמציא נקודת התחלה x0 כנקודת מינימום ולקבל את הסדרה:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$

ומכיוון ש $F(\mathbf{x}_0) \geq F(\mathbf{x}_1) \geq F(\mathbf{x}_2) \geq \cdots,$

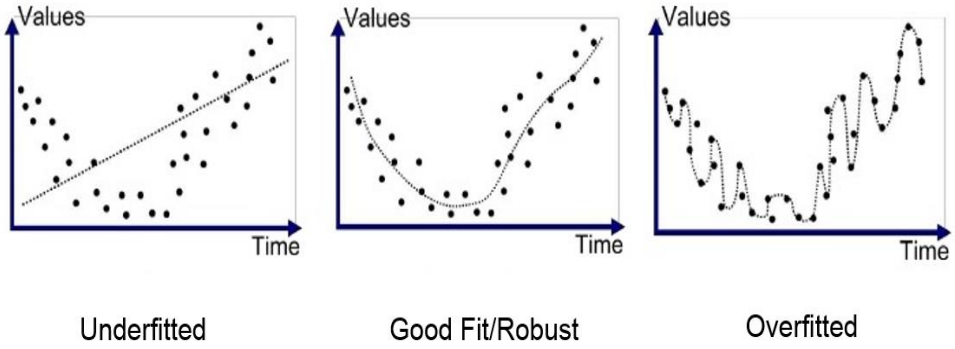
הסדרה Xn יכולה להתכנס לנקודת המינימום המבוקשת.

$$\gamma_n = \frac{(\mathbf{x}_n - \mathbf{x}_{n-1})^T[\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})]}{||\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})||^2}$$



over fitting & under fitting:

במצב של התאמת יתר(over fitting), ישנו מספר רב מידי של נוירונים ולכן הקשרים בין הנוירונים שהגיעה אליהם הרשת (המשקולות),



מותאמים לדוגמא ספציפית ולאחר ביצוע הכללה יגרמו לתוצאות שגויות. לעומת זאת כאשר ישנם מעט מידי נוירונים הקשרים בין הנוירונים אליהם תגיע הרשת יהיו כלליים מידי ולא מדויקים (under fitting). בכדי למנוע זאת מחלקים את נתוני הלמידה ל – 3 מחלקות: אימון, ולידציה ובחינה, כאשר למחלקת האימון מוכנסים בערך 70 אחוזים מן הדוגמאות, והשתיים

האחרות מתחלקות שווה בשווה ב – 30 האחוזים הנותרים, כך שכל אחת מהן מקבלת 15 אחוזים. תפקיד מחלקת האימון הוא כשמה – לאמן את הרשת. תפקיד מחלקת הווליד‌ציה הוא לבדוק שלא נוצר מצב של התאמת יתר. היא עושה זאת כך: בתהליך האימון נבדקת גם השגיאה של מחלקת הווליד‌ציה. אם השגיאה עולה מספר פעמים שנקבע מראש, חוזרים למשקולות שהיו לפני עליית השגיאה. מטרת מחלקת הבחינה הוא לבדוק את הרשת שנוצרה על דוגמאות שלא אימנו את הרשת. על מנת למנוע מצב של תת התאמה נבחר מראש מספר גדול אבל לא יותר מידי, כדי לא ליצור התאמת יתר.

פונקציות המעבר:

פונקציות המעבר – פונקציות הסיגמואיד, הינה פונקציה מתמטית בעלת עקומה בצורת האות הלועזית S. פונקציות אלה מוגדרות על המספרים השלמים, דיפרנציביאליות, חסומות, בעלות נגזרת אי-שלילית לכל אורכן ועולות באופן מונוטוני.

על מנת לפתור בעיות שאינן לינאריות פונקציות המעבר שנשתמש בהן בשכבות הביניים יהיו פונקציות סיגמואידיות כגון logsig, tansig (עליהן נפרט בהמשך). בבעיות קטלוג משתמשים לרוב בפונקציות מסוג זה גם בשכבת הפלט. תפקיד פונקציות המעבר הינו לעבד את הנתונים שנקלטו על מנת להגיע בעזרת המשקולות לפלט הרצוי.

Logsig $f(x) = \frac{L}{-1+e^{-k(x-x0)}}$

(xo הוא ערך ה-x של נקודת המרכז של הסיגמואיד, L הוא הערך המרבי של העקום ו-K הוא שיפוע העקום)

Tansig – מבחינה מתמטית, פונקציית tansig זהה לפונקציית tanh, שהיא:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = -i * tan(x)$$

שגיאת הכללה ושגיאת אימון:

על מנת לאמן את הרשת משתמשים בדוגמאות נתונות. בדוגמאות אלה מנסים להגיע לשגיאה מינימלית.השגיאה בשלב הזה של התהליך נקראת שגיאת אימון. לאחר שמגיעים לשגיאת האימון המינימלית הרצויה, בוחנים את המשקולות שקיבלנו על דוגמאות שונות (test set). לאחר מעבר זה מנסים להגיע לשגיאה מינימלית בסט הבחינה.השגיאה בשלב הבדיקה על סט הבחינה נקראת שגיאת הכללה. מטרתנו באימון הרשת היא להגיע לשגיאת הכללה מינימלית.

נוירון בודד לעומת רשת נוירונים

בקלסיפיקציה בנוירון בודד ניתן לסווג רק ל – 2 ממחלקות שונות, בעוד קלסיפיקציה ברשת נוירונים מאפשרת לנו לסווג את הדוגמאות למספר רב של מחלקות. ברגרסיה בנוירון בודד ישנו משתנה בלתי תלוי אחד בלבד, כלומר רק משפיע אחד על תשובת הרשת. ניתן להקביל זאת למחיר בית הנקבע אך ורק לפי ממיקומו. לעומת זאת, ברגרסיה המתבצעת ברשת נוירונים ניתן להשתמש במספר משתנים בלתי תלויים, כלומר מספר גורמים המשפיעים על תשובת הרשת. ניתן להקביל זאת למחיר בית הנקבע לפי מספר משתנים, כגון מיקום הבית, מערכת החינוך באיזורו וגודלו.

חלק שני – הבעיה

בעבודתנו יצרנו רשת נוירונים אשר מבצעת קלסיפיקציה של תמונות לקטגוריות של מספרים ופעולות חשבון. על מנת לעשות זאת בנינו רשת המקבלת נתונים של תמונות של מספרים ופעולות חשבון, וכך לומדת כיצד להתאים כל תמונה לקטגוריה המתאימה לה, כלומר לזהות מהו המספר או הפעולה המוצגים בתמונה. לאחר מכן הרשת מקבלת תמונות חדשות של מספרים ופעולות חשבון, ולאחר שמזהה כל תמונה ומתאימה אותו לספרה או תרגיל החשבון הנראה בתמונה היא יודעת לפתור את התרגיל.

רקע על אריתמטיקה

אריתמטיקה, הידועה גם בשם חשבון, היא הענף העתיק והבסיסי ביותר במתמטיקה. זהו החלק היסודי ביותר במתמטיקה, והוא בסיסם של כל אחד ואחד מענפי המתמטיקה וחיוני עד מאוד להבנתם. בפשטות, המילה מתייחסת לענף במתמטיקה העוסק במספרים שלמים ובתכונותיהם, כמו ראשוניות וכדומה.גבולותיו של ענף מתמטי זה אינם תחומים באופן מובהק, והם השתנו מספר פעמים לאורך ההיסטוריה. במתמטיקה המודרנית, המונח אריתמטיקה משמש, בהכללה, לתחום העוסק בקיום פתרונות למשוואות, או מערכות של משוואות.

עד 1801, אריתמטיקה הייתה אוסף משפטים והשערות שונים, שלכאורה כלל לא היה ביניהם קשר, אך בשנה זו פרסם קרל פרידריך גאוס, מתמטיקאי גרמני, ספר הנקרא "מחקרים אריתמטיים" (DisquisitionesArithmeticae) ובו הביא גאוס את עבודת קודמיו יחד עם עבודתו שלו בצורה מאורגנת, מילא את הפערים הלוגיים בהוכחות לא מושלמות של קודמיו, הכליל טענות, תיקן הוכחות רעועות, והרחיב את התחום האריתמטי. ספר זה היווה בסיס לעבודות של מתמטיקאים אחרים בני המאה ה-19 והמשיך לעורר השפעה מכרעת גם במאה ה-20, למשל השערות ששיער בספרו ב 1801 אוששו רק בשלהי המאה ה – 20.

עקרונות האריתמטיקה הבסיסיים, המבוססים על ארבע פעולות החשבון (חיבור - הוספת מספר עצמים למספר אחר של עצמים, חיסור - גריעה של מספר עצמים ממספר אחר, או הפרדה של קבוצה לשתי קבוצות, כפל - תהליך של חיבור חוזר: חיבור מספר נתון של כמויות שוות זו לזו, וחילוק - הפוכה לכפל. היא עונה על השאלה: "כמה עצמים יהיו בכל חלק של קבוצה נתונה, אם יחלקו אותה למספר נתון של חלקים השוווים בגודלם")וסדר בין מספרים, משמשים כל אדם בימינו וגם בעבר לצורך ביצוען של משימות יום-יומיות פשוטות כגון הכנתו של מזון, בבישול, בנהיגה, בקריאת שעון, בתכנון כלכלת הבית ולצרכים רבים נוספים. כלים בסיסיים מתחום האריתמטיקה נחשבים לחלק בלתי נפרד מן האוריינות. נוסף על כך, האריתמטיקה היסודית מהווה את הבסיס ההכרחי לרכישת ידע מתקדם יותר במתמטיקה, ולכן גם מהווה בסיס לרכישת ידע מתקדם בפיזיקה וכימיה, או כל תחום מדעי, הנדסי או טכנולוגי אחר. הידע המתמטי בכללותו מהווה לא אחת מדד מרכזי בקביעת השכלתו הפורמלית של אדם. לפיכך, מקנה החברה חשיבות רבה לרכישת מיומנויות אריתמטיות כבר בגיל צעיר.

רקע על בינה מלאכותית(Artificial intelligence)ולמידת מכונה (machine learning)

*בינה מלאכותית*היא ענף של מדעי המחשב העוסק ביכולת לתכנת מחשבים לפעול באופן המציג יכולות המאפיינות את הבינה האנושית,ו/או לגרום למכונה להתנהג בדרך שהייתה נחשבת לאינטליגנטית לו אדם התנהג כך.

*למידת מכונה*היא תת-תחום במדעי המחשב ובבינה מלאכותית העוסק בפיתוח אלגוריתמים המיועדים לאפשר למחשב ללמוד מתוך דוגמאות, ופועל במגוון משימות חישוביות בהן התכנות הקלאסי אינו אפשרי. למשל, בעיית זיהוי שמוח אנושי מסוגל לפתור, אך לא מסוגל

לכתוב את הכללים לזיהוי בצורה מפורשת, או שהם משתנים עם הזמן ולא ניתנים לכתיבה מראש.

מטרת הלמידה יכולה להיות מידול ,חיזוי או גילוי של עובדות לגבי העולם האמיתי. לדוגמה: עבור זיהוי אותיות ניתן להשתמש בלמידת מכונה כדי לגלות מהי האות הכתובה או המודפסת. מערכות זיהוי דיבור יכולות גם הן להשתמש בלמידת מכונה כדי ללמוד, בהינתן אותות קוליים כלשהם, מהי ההברה שיצרה אותם. דוגמה נוספת היא מערכות המלצה, מערכות המבוססות על נתונים המצביעים על קשר בין קבוצת משתמשים/אנשים אקראית לבין קבוצת פריטים שונים, ויודעת למשל להציע פריטים משלימים ללקוחות, בהתבסס על ניתוח נתונים שנלמדו מהעבר מקבוצת אנשים אחרת, שחיפשה פריטים בעלי מאפיינים דומים.

למידה המכונה התפתחה לצד התפתחות הבינה המלאכותית, זאת משום שהבינה המלאכותית בחלקה מבוססת על למידת מכונה. בתחילה המאמץ העיקרי היה לחקות את התנהגות המוח האנושי. פריצת הדרך הראשונה הייתה ב – 1957 עם המצאת מודל הפרספטרון – אלגוריתם חישובי המחקה את פעילות הנוירונים במערכת העצבים. אלגוריתם זה אינו מסוגל לייצר פונקציות מורכבות, דבר המנע מאחרים לפתחו במשך זמן רב.

במהלך שנות ה-70 נכנס תחום למידת המכונה לתרדמת כאשר מערכות מומחה (תוכנית מחשב אשר מייצגת ידע, מסיקה ממנו מסקנות ומסוגלת להסביר את האופן בו קיבלה את מסקנותיה) היוו את מוקד העניין בתחום הבינה המלאכותית. באמצע שנות השמונים החלה תקומה חוזרת בתחום כאשר הומצאו עצי ההחלטה(הוא מודל חיזוי המספק מיפוי בין תצפיות לערכים המתאימים עבורן) והופצו בתוכנה, ויתרונם היה ביכולת להציגם ובקלות הסברתם. בנוסף הם היו מגוונים ונתנו מענה למספר רב של בעיות. גם המודלים של רשתות נוירונים החלו להתפתח מחדש, כשתהליך פעפוע לאחור (שיטה המשמשת לעדכון משקולות שכבות הביניים) היווה פריצת דרך בתחום. רשתות הנוירונים עם השכבות הנסתרות יכלו עתה להביע פונקציות רבות ואפשרו להתגבר על מגבלותיו של הפרספטרון. גם עצי ההחלטה וגם רשתות הנוירונים שימשו עתה באפליקציות פיננסיות כגון אישור הלוואות, זיהוי תרמיות וניהול תיקים, ובתעשייה הם שיפרו תהליכים רבים למשל בתחום האוטומציה בדואר.

בשנות התשעים עם כניסת האינטרנט ותחילתו של עידן התפוצצות המידע החלה התקדמות רבה בתחום למידת המכונה. בשנת1995פורסם לראשונה אלגוריתם מכונת וקטורים תומכים (טכניקה של לימוד מכונות על ידי הכללת פתרונות על בסיס מאגר גדול של דוגמאות פתורות המשמשת לניתוח נתונים לסיווג ולרגרסיה), ואומץ כאלגוריתם מוביל. תוכנות קוד פתוח שנכתבו לאלגוריתם הפכו אותו פופולרי לשימוש.

בשנים האחרונות חלו התפתחויות נוספות בתחום ,רגרסיה לוגיסטית (מודל סטטיסטי המתאר קשר אפשרי בין משתנה שאי אפשר למדוד אותו במספרים, ובין משתנים אחרים) התגלתה מחדש ועוצבה לתת מענה לבעיות עם טווח רחב יותר. בעקבות כך פותחו אלגוריתמים רבים שהביאו לתוצאות מצוינות.

תת תחום נוסף שהפך פופולרי מהעשור הראשון של המאה ה-21 הוא תחום הלמידה העמוקה, שמתבסס על רשתות נוירונים בעלות מספר רב של שכבות.

ספרות והיסטוריה מחקרית

זיהוי כתב יד נראה לראשונה במחשבי כף היד של חברת "apple" האמריקנית, אך הקונים לא היו מרוצים מאיכות הזיהוי, מה שגרם לחוסר התלהבות מהגרסאות זיהוי כתב היד הבאות של חברת "apple", אשר כללו גם תיקון שגיאות כתיב. בהמשך ניסו גם החברות "IMB","NCR" ו-"Microsoft", אך גם הן נחלו כישלון מפואר. החברה הראשונה אשר הוציאה מכשיר בעל מערכת שהצליחה לספק את קהל הקונים שלה הייתה חברת "palm", שהתוכנה בה השתמשה ייחסה סדרות של משיכות עט לכל תו ותו בהתאם למשתמש, מעשה אשר הקטין משמעותית את כמות השגיאות, אך גרם לגידול בעקומת הלמידה של מכשירם. הגרסה הבאה אשר הוגדרה על ידי המיינסטרים כ"פצצתית", פותחה על ידי חברת "Microsoft" כתוכנה לזיהוי כתב יד במחשבים ניידים אשר מצוידים בלוח כתיבה ועט.
החידוש המשמעותי בתוכנה הינו המעבר מלמידה מהצורה בה המשתמש כותב, ללמידה על ידי השוואת הקלט לבסיס נתונים המכיל צורות כתיבה רבות של כל אחת מהאותיות והמספרים. בדרך הזו אמנם השתפרו התוצאות, אך עדיין נותרו מספר בעיות לא פתורות, כגון כיצד להתמודד עם רווחים בגדלים שונים, כיצד לקרוא כתב מחובר, והקושי של המערכת לזהות אותיות בגדלים לא שווים.

זיהוי כתב היד מתחלק לשתי טכנולוגיות עיקריות – זיהוי מקוון וזיהוי לא מקוון.
<u>זיהוי מקוון</u> מאפשר המרת טקסט בעזרת עט או לוח אלקטרוני, המזהים את התנועה המבוצעת על ידע המשתמש. התוכנה מסמנת את התנועות בעזרת ווקטור בעל שני ממדים אשר מזהה מקטעים נפרדים של מילים ואותיות ולאחר מכן מנתח אותם. לאחר שקיבלה התוכנה אותיות, היא משווה אותן (בצורה סטטיסטית/ישירה) למאגר נתונים בו רשומות המילים והאותיות אשר נדגמו ממשתמש/משתמשים שונים. לעיתים נעשה שימוש במילון על מנת לאתר טעויות כתיב נפוצות.
<u>זיהוי לא מקוון</u>מורכב מניתוח התמונה הנתונה וזיהוי מקטעים המכילים בתוכם מילים/תווים ולאחר מכן המרתם לאותיות הניתנות לעריכה על ידי מחשב. הבעיה העיקרית בשיטה זו היא שנייחות הטקסט מונעת מהתוכנה ללמוד מצורת הכתיבה של המשתמש, אך בעיה זו נפתרת בעזרת שימוש בבלשנות ובסטטיסטיקות מתאימות.

מדוע עבודתנו חשובה?

בימים אלה, על מנת לאפשר גישה נוחה תכנים רבים בכל רחבי העולם ושיתופם, מנסים להשתמש במידע בפורמט דיגיטלי. לצורך כך במקרים אשר נעשה בהם שימוש בכתב יד המעבר לפורמט הדיגיטלי הוא נחוץ.

אמנם ישנה האפשרות לסרוק את כתב היד לקובץ תמונה בצורה פשוטה, אך השימוש בשיטה זו לא מאפשרת עריכהשל הטקסטים אשר נסרקו או חיפוש תכנים בהם. על מנת כן לאפשר את שתי הפעולות הנ"ל, ישנו הצורך בטכנולוגיה אשר תמיר את הסימנים שנכתבו לסימנים שמחשב יודע לזהות.

ראשית כל, עבודתנו מהווה סוג של מחשבון של פעולות מתמטיות פשוטות, שבמקום להקליד את הספרות ואת פעולות החשבון יש לצלמם. הננו משוכנעים כי מעטיהאנשים שיוכלו להגיד רעה על מחשבון. בנוסף, בחלק מן האתרים יש בקשה לפתור תרגיל חשבוני פשוט כדי לבדוק שאינך רובוט המנסה לפרוץ למשתמש של מישהו באתר מסויים, או רובוט המנסה לפתוח אינסוף משתמשים באתר כלשהו כגון אתר קהוט הנפלא והאהוב המשמש לבניית חידונים תחרותיים וידידותיים כאחד. בעזרת עבודתנו דבר זה יהיה אפשרי. עוד שימוש של העבודה היא שהיא יכולה להגיד לך כמה קריא כתב היד שלך – אם הוא איננו קריא התוכנה תקרא את הספרה או הסימן שנכתב כספרה או סימן אחרים.

אנליזה/שיטה

יצירת מסד הנתונים – בניית ה –database

על מנת ליצור את מסד הנתונים הלכנו למעוז המידע הגדול ביותר שקיים – אביגיל. אביגיל הכווינה אותנו לאתר ה database השגיאKaggle. מאתר זה הורדנו מעל 200,000 תמונות של ספרות, סימנים מתמטיים ואותיות לועזיות רלוונטיות למתמטיקה. בחרנו מכל קטגוריה של ספרות או סימנים 500 דוגמאות כך שבסך בכל יש 17 קטגוריות ו – 8500 תמונות. בכדי לבצע את הקלסיפיקציה, הרשת זקוקה למספר רב של תמונות מכל אחת מהקטגוריות, לכן בחרנו להשתמש ב - 500 תמונות (דוגמאות) מכל קטגוריה. השימוש ב - 500 תמונות מכל קטגוריה נותן לנו את האפשרות לקבל תוצאות ומשקולות באיכות גבוהה (אחוזי שגיאה נמוכים) בזמן יחסית קצר (למחשב לא לוקח המון זמן לבנות את הרשת).

ייצוגהנתונים כמספרים ותהליכיעיבוד מוקדם

רשת נוירונים מבצעת קלסיפיקציה בעזרת קלט מספרי, בעוד לנו יש תמונות, לכן יש להעביר את התמונות ממצב ויזואלי לנתונים מספריים המייצגים את התמונה, שאותם המחשב יכול לקרוא, אך גם יש דרישה לכך שאנו נוכל להשתמש בנתונים לצרכינו שלנו, ולא רק לצורך המחשב. כדי לעשות זאתיש לעבור מספר שלבים:

שלב ראשון הוא למיין את הנתונים לקטגוריות, כך שבכל קטגוריה יש דוגמאות לספרה או סימן מתמטי שונה. את אלו שומרים במסמכים. לאחר מכן קוראים לכל אחת מן התמונות בכל קטגוריה, והופכים אותה מתמונה ויזואלית למספרים שהמחשב יכול לקרוא כתמונה. את אלה שומרים במשתנה מסוג struct נפרד לכל קטגוריה. כעת כל תמונה שמורה במשתנה מסוג struct הבנוי מ –500 מטריצות המייצגות את התמונות של הקטגוריה הרלוונטית, אך טיפוס הנתונים הוא מסוג'uint8', שזהו הטיפוס המייצג תמונות. לאחר מכן המרנו כל תמונה לווקטור עמודה על ידי קריאה למקום המתאים בתוך ה –struct והעברתו למשתנה מסוג cell, והעברה נוספת מה –cell אל המטריצה כדי ליצור מטריצה שכל עמודה מהווה דוגמא. מכיוון שבשלב בניית הרשת נוצרת בעיה עם משתנים אלה, הפכנו אותם לטיפוס מסוג'double'על מנת שנוכל לעבוד איתם.בהמשך יצרנו מטריצה גדולה – מטריצת הקלטים אשר בנויה מחיבורן של כל המטריצות של כל ספרה או סימן מתמטי. יכלנו לעצור בנקודה זו, אך החלטנו לבצע לנתונים resize, הורדה של איכות התמונה, כדי שהרשת תיבנה בזמן פחות.

סיכום הקוד

הקוד שלנו בנוי מהרבה בלוקים (script) ופונקציות אשר פועלים יחד בכך שהינם "קוראים" אחד לשני ומשתשים בנתונים אשר חושבו בבלוק אשר הם קראו לו.

Preperation

בבלוק הנ"ל התוכנה עוברת על כל התמונות שהכנסנו ל- database של העבודה ולבסוף שומרת אותן יחד כמטריצה של הפיקסלים שבכל תמונה.

מסד הנתונים שלנו מורכב מ- 17 תיקיות שונות שבכל אחת מהן יש 500 דוגמאות של הסימן המתמטי/ הספרה אשר שמורים באותה התיקייה.

התוכנה עוברת על כל הנתונים בכל תיקייה. ראשית היא שומרת כל אחת מהתמונות כמטריצה של הפיקסלים בגודל 45X45. לאחר מכן התוכנה משנה את צורת המטריצה של התמונה לוקטור עמודה באורך 2025 (כל טור במטריצה ההתחלתית נכנס מתחת לטור שקדם לו). בהמשך, התוכנה מחברת את המערכים אחד ליד השני עד שלכל קטגוריה במסד הנתונים יש מטריצה בגודל2025X500. כל טור הינו תמונה וכל שורה הינה מיקום של פיקסל התמונה.

דוגמה למעבר על אחת התיקיות:

פתיחת התיקייה '7' בעלת הדוגמאות של הספרה 7.	`File = 'F:\moag\12thgrade\Gemer\datasets\7';`
יצירת מבנה (struct) של כל הדוגמאות בתיקייה ששמן מתחיל ב '_7'.	`Struct = dir(fullfile(File,'7_*.jpg'));`
לולאת for אשר רצה מאחת ועד מספר הדוגמאות במבנה.	`for i = 1:numel(Struct)`
שמירת קובץ התמונה במיקום i במבנה.	`ImgFile = fullfile(File,Struct(i).name);`

שמירת התמונה מהקובץ (כמטריצה).	`Img = imread(ImgFile);`
הצגת התמונה.	`%  imshow(I);`
שמירת מטריצת כל תמונה גם ב-data ב- struct.	`Struct(i).data = Img; % optional, save data.`
סוף לולאת ה- for.	`end`
בניית מטריצה ריקה אשר יוכנסו בה כל הדוגמאות של 7 כוקטורי עמודה.	`seven = [];`
לולאת for אשר רצה מאחת ועד מספר הדוגמאות במבנה.	`fori = 1:size(Struct,1)`
שמירת דוגמה במיקום iמתוך struct כטיפוס cell.	`cell = struct2cell(Struct(i));`
שמירת המטריצה של אותה הדוגמה.	`demo = cell{7};`
שינוי צורת מטריצת הפיקסלים של הדוגמה לוקטור עמודה.	`demo = reshape(demo,(size(demo,1))^2,1);`
חיבור כל וקטורי העמודה ליצירת מטריצה של כל הדוגמאות של 7.	`seven = [seven, demo];`
סוף הלולאה.	`end`

לאחר שהתוכנה עוברת כך על כל 17 התיקיות שכלולות במסד הנתונים שלנו התוכנה מקבצת את כל המטריצות למטריצה אחת של כל התווים והדוגמאות שמועברת מטיפוס uint8 לטיפוס double. לאחר מכן, יצרנו מטריצה נוספת שתהא שווה למטריצה הקודמת,ועשינו להresize בכדי שהתוכנה תוכל לרוץ בצורה מהירה יותר ללא הבדל

משמעותי ביכולת הזיהוי שלה של התמונות. לבסוף, נבנית גם מטריצת התשובות הנכונות ובה 17 שורות ו - 17 עמודות אשר כל הערכים בה הם 0 חוץמערך אחד בכל עמודה אשר הינו 1 (דבר זה מסמל כי זוהי התשובה הנכונה – התו שנמצא בתמונה).

הרכבת המטריצה בעלת כל סוגי התווים מה-database.	`p = [one, two, three, four, five, six, seven, eight, nine, zero, minus, plus, div, X, sqrt, par1, par2];`
הפיכת המטריצה מטיפוס uint8 לטיפוסdouble.	`p = im2double(p);`
שמירת המטריצה בשם נוסף.	`BigP = p;`
מחיקת המטריצה P ובנייתה כמטריצה ריקה.	`p = [];`
יצירת לולאת for באורך מספר התמונות שכלולות ב–bigP.	`fori=1:length(BigP)`
הוספת עמודה לתוך המטריצה של התמונה לאחר resize.	`p = [p, imresize(BigP(:,i), 0.2)];`
סוף הלולאה.	`end`
בניית מטריצה של אפסים בעלת כמות השורות ככמות הקטגוריות וכמות העמודות ככמות הדוגמאות.	`t = zeros(size(p,2)/500,size(p,2));`
שמירת מיקום התשובה הנכונה בהתחלה כ 1.	`j=1;`
לולאת for באורך כמות העמודות ב-t.	`fori = 1:size(t,2)`
קביעת המיקום בו התשובה הנכונה.	`t(j,i) = 1;`
אם i מתחלק ב 500.	`if rem(i,500) == 0`

הגדלת j באחד-שינוי מיקום התשובה הנכונה.	`j = j+1;`
	`end`
סגירת הלולאות.	`end`

GetIndices

הפונקציה GetIndices מקבלת כקלט חלק מסויים של המטריצה P שמכיל את כל הדוגמאות שיש של תו מסויים, ולבסוף מחזירה את האינדקסים של הדוגמאות שישמשו לאימון הרשת, של הדוגמאות שישמשו לולידציה ושל הדוגמאות שישמשו לבחינת הרשת.

הפונקציה מקבלת בכל פעם רק קטגוריה אחת מ-p כדי שכמות הדוגמאות מכל קטגוריה יהיו שוות,על מנת שבמהלך הלמידה התוכנה לא תבצע "העדפה" לתו מסוים כיוון שמספר הדוגמאות שלו בלמידה גדול יותר.

ראשית הפונקציה מחלקת את האינדקסים לשתי "קבוצות"; 70% נשמרים בשביל האימון והשלושים הנותרים מתחלקים שווה בשווה בין הולידציה והמבחן, כך שכל אחת מהקבוצות מקבלת 15 אחוזים מהדוגמאות.

כותרת הפונקציה.	`function [ train, val, test ] = GetIndices( data )`
בניית מערך ריק של אינדקסי הולידציה והמבחן.	`valTest = [];`
שמירת 70% מהאינדקסים לאימון באופן רנדומלי.	`train = randperm(size(data,2) , 0.7*size(data,2));`
הכנסת כל האינדקסים הנותרים אל תוך המערך valTest.	`fori = 1:(size(data,2))` `include = ismember(i,train);`

`if include == 0`
 	
`valTest = [valTest, i];`
 	
`end`
 	
`end`	

`val = [];`	בניית מערך ריק של אינדקסייהולידציה.
`test = [];`	בניית מערך ריק של אינדקסיי המבחן.
`valIndex = randperm(length(valTest) , 0.5*length(valTest));`	הכנסת 50% מהאינדקסים הנותרים (מ-valTest) אל valIndex באופן רנדומלי.

`for i = 1:(length(valTest))`
 	
`include = ismember(i,valIndex);`
 	
`if include == 1`
 	
`val = [val, valTest(i)];`
 	
`else`
 	
`test = [test, valTest(i)];`
 	
`end`
 	
`end`
	שמירת כל האינדקסים הנותרים בוקטורהולידציה או בוקטור המבחן על ידי בדיקה האם הם נמצאים ב- valIndex.

`end`	

Netbuilder

בלוק זה בונה את רשת הנוירונים ומאמן אותה. הקוד מתחיל בקליטת מטריצת התמונות שנוצרה בבלוק Preperation ומטריצת התשובות הנכונות. לאחר מכן מתקיימת בנייה של רשת נוירונים בעלת שכבה חבויה אחת המורכבת מ - 40 נוירונים. בהמשך התוכנה משתמשת בלולאת for בכדי להכניס כל קטגוריה אל הפונקציה GetIndices בנפרד, מחברת ומסדרת את מערכי התוצאה. לבסוף התוכנה משתמשת ברשת שנבנתה, במטריצת התמונות ובמטריצת התשובות הנכונות בכדי לאמן את הרשת.

ה - scriptקורא לבלוק אשר בונה את מטריצת התשובות הנכונות t ואת מטריצת התמונותPאשר יבנו ויאמנו את הרשת.	`Preperation;`
אופציה חלופית להרצתpreparation כאשר שומרים את P ו t. טוען אותם מהמחשב.	`load('DATA\p.mat');` `load('DATA\t.mat');`

יוצר רשתעם 40 נוירונים בשכבה החבויה.	`net = patternnet(40);`
	`net.inputs{1}.processFcns = {};`
	`rng('shuffle')`
	`net.trainParam.showWindow = 0;`

מאתחל משתנים שנמלא בריצת לולאת ה - for:
 	
מהו התו עליו הרשת עוברת באותו הרגע –j, סט האימון training, סט הולידציהvalidation וסט הבחינה test.
	`j=0;`
	`training = [];`
	`validation = [];`
	`test = [];`

פותח לולאת שרצה מ-1 עד מספר הטורים במטריצה Pבקפיצות של 500.	`fori = 1:500:size(p,2)`
מאפס את המשתנה b אליו נכניס בכל ריצה של הלולאה את התמונות הספיציפיות של הקטגוריהאותה אנו ממיינים.	`b = [];`
פתיחת לולאה שרצה 500 פעמים, פעם לכל תמונה בודדה שהיא חלק מהקטגוריה.	`for k = 1:500`
מוסיף ל - b את התמונה הבאה.	`b = [b, p(:,k+j)];`
יוצא מהלולאה.	`end`
מוסיף ל - j 500 על מנת לעבור לתא הבא.	`j = j+500;`
בוחר ומכניס ל - tr, val, ו - te את האינדקסים שנבחרים לסטים בעזרת הפעולה GetIndicesהמבוצעתעל המשתנה b- התמונות השייכות לקטגוריה הנוכחית.	`[tr, val, te] = GetIndices(b);`

מכניס ל - training, validation, ו - test את האינדקסים שנבחרו בעזרת הפעולה וסוגר את לולאת הקטגוריות.	`training = [training, tr + j - 500];`
	`validation = [validation,val+j-500];`
	`test = [test, te + j - 500];`
	`end`

מסדר את סטי האימון, הולידציה והבחינה.	`training = sort(training);`
	
	`validation = sort(validation);`
	
	`test = sort(test);`

מכניס לרשת net את הסטים שנבנו בשורות הקודמות.	`net.divideFcn = 'divideind';`
	
	`net.divideParam.trainInd = training;`
	
	`net.divideParam.valInd = validation;`
	
	`net.divideParam.testInd = test;`

מאמן את הרשת בעזרת המשתנים שבנינו בבלוק.	`[net tr] = train(net, p, t);`

DataNetCalc1Layer וגם DataNetCalc2Layer

הפונקציות מקבלות את מטריצת התמונות p את המשקולות בין השכבה הראשונה לשכבה החבויה ובין השכבה החבויה והבאה, ואת וקטור הסיפים. לאחר מכן יוצרות הפונקציות מטריצה של התשובות המחושבות של זיהוי תמונות הסימנים המתמטיים. הפונקציה DataNetCalc1Layer נועדה לחשב את התשובות בעזרת רשת עם שכבה חבויה אחת בניגוד לפונקציה DataNetCalc2Layers, אשר משתמשת ברשת בעלת שתי שכבות חבויות. חוץ מכך, שתי הפונקציות זהות.

שמות הפונקציות.	function [answer] = DataNetCalc2Layer(p, iw, lw, b)	function [answer] = DataNetCalc1Layer(p, iw, lw, b)
המשקולות בין השכבת הקלט לשכבה החבויה הראשונה.	iw = iw{1,1};	
הכפלת המשקולות במטריצה והוספת הסף המתאים.	tnsg = iw*p + b{1,1};	
חישוב ערכי הנוירונים בשכבה החבויה הראשונה.	hidden = tansig(tnsg);	
המשקולות בין השכבה החבויה הראשונה לשכבה החבויה אחריה.	lw = lw{2,1};	
הכפלת המשקולות במטריצה והוספת הסף המתאים.	tnsg = lw*hidden + b{2,1};	
חישוב ערכי הנוירונים בשכבה החבויה השנייה/שכבת הפלט.	hidden2 = tansig(tnsg);	answer = tansig(tnsg);
המשקולת בין השכבה החבויה השנייה לשכבת הפלט.	lw2 = lw{3,2};	

	tnsg = lw2*hidden2 + b{3,1};	הכפלת המשקולות במטריצה והוספת הסף המתאים.
	answer = tansig(tnsg);	חישוב ערכי הנוירונים בשכבה החבויה השנייה.

fori = 1:size(answer,1)
 	
for j = 1:size(answer,2)
 	
if answer(i,j) == max(answer(:,j))
answer(i,j) = 1;
 	
else
answer(i,j) = 0;
 	
end
end
end
		מציאת הערך המקסימלי בכל עמודה של answer והפיכתו ל-1 ואיפוס כל הערכים הנתרים.

end		

Check

בלוק זה מחשב את אחוז ההצלחה של הרשת ויוצר גם וקטור המכיל את אחוז ההצלחה לכל תו ספציפי. הבלוק רץ על כל קטע של כל קטגוריה בנפרד מתוך התשובות שחושבו על ידי הרשת ומוצא את אחוז ההצלחה בכל אחת מהקטגוריות. לבסוף הבלוק מחשב את ממוצע אחוזי ההצלחה בכדי למצוא את אחוז ההצלחה של כל התשובות שחושבו על ידי הרשת.

בונה את וקטור הקטגוריות הספציפיות.	`SectionCorrect = [];`
פותח לולאה שרצה על כל אחת מן הקטגוריות.	`fori = 0:500:size(a,2)-1`
מאפס את j המונה את מספר התשובות הנכונות בתו שנבדק באותו הרגע	`j = 0;`
פותח לולאה שרצה על כל אחת מהתשובות בקטגוריה הנוכחית.	`for h = 1:500`
בודק האם התשובה שחושבה על ידי הרשת זהה לתשובה בוקטור התשובות הנכונות.	`if a(:,h+i) == t(:,h+i)`
אם כן מוסיף ל - j המונה את מספר התמונות שהרשת צדקה בהן 1.	`j = j+1;`
סוגר את לולאת התשובות.	`end`

סוגר את התנאי.	end
מחשב את אחוזי ההצלחה של הרשת לתו הנוכחי.	SectionCorrect = [SectionCorrect, j/h*100];
סוגר את לולאת הקטגוריות.	end
בונה משתנה שיכיל את אחוז ההצלחה הכללי.	correct = 0;
פותח לולאה שבונה את correct– משתנה שיכיל את אחוז ההצלחה הכללי של הרשת.	fori = 1:length(SectionCorrect)
סוגר את הלולאה.	end
מחשב את אחוז ההצלחה הכללי של הרשת.	correct = correct/length(SectionCorrect)

RuNet

בלוק זה משתמש בבלוקים הנבנו לפניו בשביל לבנות את הרשת, להריץ אותה,לצייר גרפים של הטעויות בסטי האימון ,הולידציה והבחינה ולבדוק את אחוז ההצלחה שלה. בנוסף הבלוק מחשב כמה זמן לקח לכל תהליך זה.

מכניס למשתנה t1 את זמן התחלת השימוש הבלוק.	`t1 = datetime('now');`
משתמש ב - netbuilder הבונה את הרשת ומאמן אותה.	`% NetBuilder;`

מהווים חלופה ל–netbuilder. שורות אלה טוענות את הרשת, נתונים שקשורים אליה, מטריצת התמונות הנתונות ומטריצת התשובות הנכונות. שורות אלה פועלות רק אם משתנים אלה שמורים על המחשב.	`load('DATA\net.mat');`
	`load('DATA\tr.mat');`
	`load('DATA\p.mat');`
	`load('DATA\t.mat');`

מחשב את התשובות על פי הרשת המקובלת בעזרת הפונקציה datanetcalc1layer.	`a = DataNetCalc1Layer(p, net.IW, net.LW, net.b);`

מצייר את גרפי כישלון הרשת בסטיהאימון, הולידציה והבחינה בהתאמה, על מערכת צירים משותפת.	`plot(tr.epoch, tr.perf, 'y')`
	`hold on;`
	`plot(tr.epoch, tr.vperf, 'b')`
	`hold on;`
	`plot(tr.epoch, tr.tperf, 'r')`

בודק את אחוז ההצלחה של הרשת בעזרת הבלוק check.	`Check;`
מכניס למשתנה t2 את הזמן בסיום ריצת הבלוק.	`t2 = datetime('now');`
מחזיר במשתנה tend את הזמן הכולל הלקח לריצת הבלוק.	`tend = t2-t1`

Solver

הבלוק הבא הינו הבלוק שבו התוכנה מזהה את התמונות ופותרת את התרגיל המתמטי. כל הבלוקים האחרים בעבודתינו נועדו בכדי לבנות את הנתונים שאיתם התוכנה תוכל לעבוד, לעזור להריץ את הבלוק הנ"ל וחלקם נועדו בכדי שנוכל לבדוק את האופן הטוב ביותר לבניית הקוד ולבדיקה כי הוא אכן עובד. ראשית התוכנה שומרת את כל התמונות שנמצאות בתרגיל במטריצה של הפיקסלים (כמו בבלוק Preperation). לאחר מכן התוכנה משתמשת בפונקציה DataNetCalc1Layer בכדי לזהות את הסימנים בתמונות. לבסוף התוכנה משתמשת בלולאת for כפולה כדי לרוץ על מטריצת התשובות ולהכניס את כל התווים שזוהו למערך מטיפוס string ולבסוף נעשה שימוש בפונקציה str2sym של מטלב אשר מקבלת את המערך ופותרת את התרגיל החשבוני הכתוב בו.

שמירת התמונות של התווים המשומשים בתרגיל בתוך מטריצה וביצוע resize עליהם (כפי שנעשה בבלוק Preperation).	

```
% saving the matrix of pictures


File = 'equationPics';

Struct = dir(fullfile(File,'#*.jpg')); % pattern to match filenames.

fori = 1:numel(Struct)

ImgFile = fullfile(File,Struct(i).name);

Img = imread(ImgFile);

%     imshow(I);

    Struct(i).data = Img; % optional, save data.

end


pictures = [];

fori = 1:size(Struct,1)
```

`cell = struct2cell(Struct(i));`
`demo = cell{7};`
`demo = reshape(demo,(size(demo,1))^2,1);`
`pictures = [pictures, demo];`
`end`
 	
`pictures = im2double(pictures);`
 	
`oldP = pictures;`
`pictures = [];`
`fori=1:size(oldP,2)`
 	
`pictures = [pictures, imresize(oldP(:, i), 0.2)];`
`end`
 	
`clear cellFileijImgImgFiledemoStructoldP`	

`load('DATA\net.mat');`	פתיחת הרשת השמורה במחשב.
`answer = DataNetCalc1Layer(pictures, net.IW, net.LW, net.b);`	חישוב התשובות בעזרת הפונקציה DataNetCalc1Layer.
`equation = [];`	בניית מערך ריק שיכלול את התרגיל המתמטי.

יצירת לולאת for הרצה על מספר העמודות ב-answer.	fori = 1:size(answer, 2)
פתיחת לולאת for הרצה על מספר השורות ב-answer.	for j = 1:size(answer, 1)
בדיקה האם זוהי התשובה שחושבה.	if answer(j, i) == 1

הוספת התו המתאים למערך לפי התשובה שחושבה על ידי הרשת.	if j<= 10
	equation = [equation, num2str(j)];
	elseif j==11
	equation = [equation, '-'];
	elseif j==12
	equation = [equation, '+'];
	elseif j==13
	equation = [equation, '/'];
	elseif j==14
	equation = [equation, '*'];
	elseif j==15
	equation = [equation, 'sqrt'];
	elseif j==16
	equation = [equation, '('];
	elseif j==17

equation = [equation, ')']; end end end end end end end end end	
end	
end	
solution = str2sym(equation)	חישוב תשובת התרגיל המתמטי בעזרת הפונקציה str2symשל מטלב.
clear answerequationhiddeniwlwnnetpicturestnsgij	מחיקת כל המשתנים אשר נעשה בהם שימוש בבלוק שאינם רלוונטים יותר.

Ball_of_BILBOOL

מטרת ה- script הנ"ל היא בניית מטריצת הבלבול (Confusion matrix/Chart).

ראשית המטריצה בונה שני תאי וקטורי עמודה (cell array) ריקים באורך כמות הדוגמאות של הקטגוריות. לאחר מכן התוכנה פותחת לולאת for כפולה העוברת על מטריצת התשובות המחושבות ועל מטריצת התשובות הנכונות בכדי למלא את שני התאים הוקטורייבתוים המתאימים (מטיפוס string).

לבסוף, התוכנה בונה מטריצת בלבול של התשובות באחוזים ומציגה בחלון חדש טבלהשל מטריצת הבלבול (לא באחוזים).

בניית תא וקטורי ריק שיכיל את התשובות המחושבות.	`vectorA = cell(size(a,2), 1, 1);`
בניית תא וקטורי ריק שיכיל את התשובות הנכונות.	`vectorT = cell(size(a,2), 1, 1);`
	`fori = 1:size(a,2)`
יצירת משתנה שבודק האם נמצאת התשובה גם ב- a וגם ב- t ומאפשרת אפשרות התקדמות מהירה יותר לעמודה/דוגמה הבאה.	`bre = 0;`
	`for j = 1:size(a,1)`
מציאת התשובה המסומנת ב- a.	`if a(j,i) == 1`
הגדלת משתנה היציאה המוקדמת ב-1.	`bre = bre + 1;`

הוספת התו המתאים לתא לפי התשובה שחושבה על ידי הרשת.	`if j < 10`
	`vectorA{i} = num2str(j);`
	`elseif j==10`
	`vectorA{i} = '0';`
	`elseif j==11`
	`vectorA{i} = '-';`
	`elseif j==12`
	`vectorA{i} = '+';`
	`elseif j==13`
	`vectorA{i} = '/';`

elseif j==14 vectorA{i} = '*'; elseif j==15 vectorA{i} = 'sqrt'; elseif j==16 vectorA{i} = '('; elseif j==17 vectorA{i} = ')'; end end end end end end end end end	
if t(j,i) == 1	מציאת התשובה המסומנת ב- t.
bre = bre + 1;	הגדלת משתנה היציאה המוקדמת ב-1.

הוספת התו המתאים לתא לפי מטריצת התשובות הנכונות.	`if j < 10`
	`vectorT{i} = num2str(j);`
	`elseif j==10`
	`vectorT{i} = '0';`
	`elseif j==11`
	`vectorT{i} = '-';`
	`elseif j==12`
	`vectorT{i} = '+';`
	`elseif j==13`
	`vectorT{i} = '/';`
	`elseif j==14`
	`vectorT{i} = '*';`
	`elseif j==15`
	`vectorT{i} = 'sqrt';`
	`elseif j==16`
	`vectorT{i} = '(';`
	`elseif j==17`
	`vectorT{i} = ')';`
	`end`
	`end`
	`end`
	`end`
	`end`
	`end`
	`end`

end end end	
ifbre == 2	בדיקה האם התשובה נמצאה בשתי המטריצות.
break	אם כן, ביצוע מעבר לעמודה/דוגמה הבאה.
end	
end	
end	
figure(2);	פתיחת חלון חדש בשביל טבלת הבלבול.
confmat = confusionmat(vectorT, vectorA);	בניית מטריצת הבלבול.
confmat = confmat/5;	שינוי הערכים במטריצה לאחוזים.
confchart = confusionchart(vectorT, vectorA);	בניית טבלת הבלבול.

מחיקת המשתנים שאינם רלוונטיים יותר.	`clear breijvectorAvectorT`

CommonMistakes

בלוק זה בונה 5 רשתות שונות ובודק מה הן התמונות שכל 5 הרשתות טעו בסיווג שלהן. הבלוק יוצר 5 רשתות נוירונים שונות בעזרת לולאת for, לאחר מכן הוא מחשב את תשובות כל הרשתות בעזרת הבלוק RuNet. הבלוק עובר על כל התמונות שנכנסו לרשת הנוירונים ובודק האם כל הרשתות טעו בהן בעזרת השוואה למטריצת התשובות הנכונות. בנוסף הבלוק מכיל קטע קוד שמצייר את התמונות הבעייתיות אך זה נמצא בהערה מכיוון שהקטע מקשה מאוד על תהליך ריצת הבלוק. מטרת הבלוק CommonMistakes היא לראות מה הן התמונות הבעייתיות שיגרמו לטעויות בכל רשת שתיבנה.

מכניס למשתנה t1 את הזמן בו הבלוק מתחיל את ריצתו.	`t1 = datetime('now');`
טוען מהמחשב את מטריצת התשובות הנכונות.	`load('DATA\t.mat');`
יוצר cell ריק בשם ays בגודל של 1x5 שיכיל בתוכו את התשובות שחישבו כל אחת מהרשתות.	`ays = cell(1, 5, 1);`
פותח לולאה שתרוץ 5 פעמים וכל פעם תבנה רשת נוירונים חדשה.	`for z = 1:5`
מריץ את RuNet שבונה רשת ומחשב את תשובותיה.	`RuNet;`
מכניס ל - ays את התשובות שחישבה הרשת הנוכחית.	`ays{z} = a;`

סוגר את לולאת הרשתות.	end
בונה וקטור ריק בשם lo_yutslach שיכיל את אינדקסי התמונות הבעייתיות.	lo_yutslach = [];
פותח לולאה שרצה על כל התשובות של הרשת הראשונה.	for j = 1:size(ays{1},2)
פותח לולאה שרצה על כל 5 הרשתות.	fori = 1:5
תנאי שבודק האם התשובה שהרשת הנוכחית הגיעה אליה נכונה.	ifays{i}(j) == t(j)
אם התשובה נכונה יוצא מלולאת הרשתות.	break
סוגר את התנאי ואת לולאת הרשתות.	end end
אם i=5 כלומר כל הרשתות הגיעו לתשובה לא נכונה	ifi == 5
מוסיף את אינדקס התמונה לוקטור האינדקסים.	lo_yutslach = [lo_yutslach, j];
סוגר את תנאי התשובות ואת לולאת התמונות.	end end

קטע קוד זה מצייר את התמונות הבעייתיות, הוא נמצא בהערה בשביל להקל על ריצת המחשב.	`% load('DATA\BigP.mat');`
	
	`%  fori = 1:length(lo_yutslach)`
	`%  figure(i)`
	`% matrixPic = BigP(:,lo_yutslach(i));`
	`% matrixPic = reshape(matrixPic,sqrt(length(matrixPic)),sqrt(length(matrixPic)));`
	`% imagesc(matrixPic)`
	`% colormap('gray');`
	`%`
	`%  end`

מכניס לt2 את הזמן בסיום ריצת הבלוק.	`t2 = datetime('now');`
מכניס לtend את זמן הרצה הכולל שלקח לבלוק.	`tend = t2-t1`

Best2Layers

בלוק זה מחולק לשני חלקים. מטרתו של הבלוק היא לבדוק מהן השכבות החבויות שיגרמו לרשת להגיע לתוצאות הצלחה מקסימליות. החלק הראשון של הבלוק יוצר רשתות נוירונים עם 2 שכבות חבויות בהן הראשונה מכילה טווח מסוים של נוירונים חבויים וכנ"ל לגבי השכבה השנייה. בנוסף פישטנו את הבלוק מספר פעמים על מנת לבדוק את הצלחת הרשת במצב בה

יש לה שכבה חבויה אחת בלבד. בחלק הראשון של הבלוק אנו בונים כל רשת בטווח הזה 5 פעמים ובודקים מהו ממוצע ההצלחה בכמות הנוירונים הזו. החלק השני מקבל את הכמויות שגרמו לתוצאות הגבוהות ביותר ומריץ אותן 20 פעמים כדי להגיע לתוצאה מדויקת יותר בנוגע למה הכמות המוצלחת ביותר.

מכניס ל - starT את הזמן בתחילת ריצת הבלוק.	`starT = datetime('now');`
יוצר מטריצה ריקה שתכיל את ממוצעי ההצלחה.	`avg = zeros(40,30);`
פותח לולאה של נוירוני השכבה החבויה הראשונה.	`for x = 41:45`
יוצר את משתנה correction שיכיל את ממוצע ההצלחה של כמות הנוירונים הנוכחית.	`correction = 0;`
פותח לולאה שתרוץ ותבנה את אותה הרשת חמש פעמים.	`for o = 1:5`
בונה את הרשת עם כמות הנוירונים הנוכחית.	`NetBuilder;`
מכניס ל - a את חישוב את התשובות של הרשת בעזרת הפונקציה DataNetCalc2Layer.	`a = DataNetCalc2Layer(p, net.IW, net.LW, net.b);`
מחשב את אחוז ההצלחה של הרשת בעזרת הבלוק check.	`Check;`
מוסיף ל - correction את השגיאה של הרשת הנוכחית לכמות הנוירונים הנוכחית.	`correction = correction + correct;`
סוגר את לולאת הרשתות הזהות.	`end`

מחלק את correction ב-o- מחשב את ממוצע ההצלחה של הרשתות עם כמות הנוירונים הנוכחית.	`correction = correction/o;`
מכניס את ממוצע ההצלחה של כמות הנוירונים הנוכחית למטריצת הממוצעים.	`avg(x,y) = correction`
סוגר את לולאת כמות הנוירונים בשכבה החבויה השנייה.	`end`
סוגר את לולאת כמות הנוירונים בשכבה החבויה הראשונה.	`end`

קטע הקוד השני זהה במבנהו לקטע זה אך הוא רץ על שתי דוגמאות ספציפיות אשר נבחרו מהדוגמאות המוצלחות ביותר מהרצת קטע הקוד הראשון. בנוסף הקטע מריץ כל רשת 20 פעמים במקום 5.

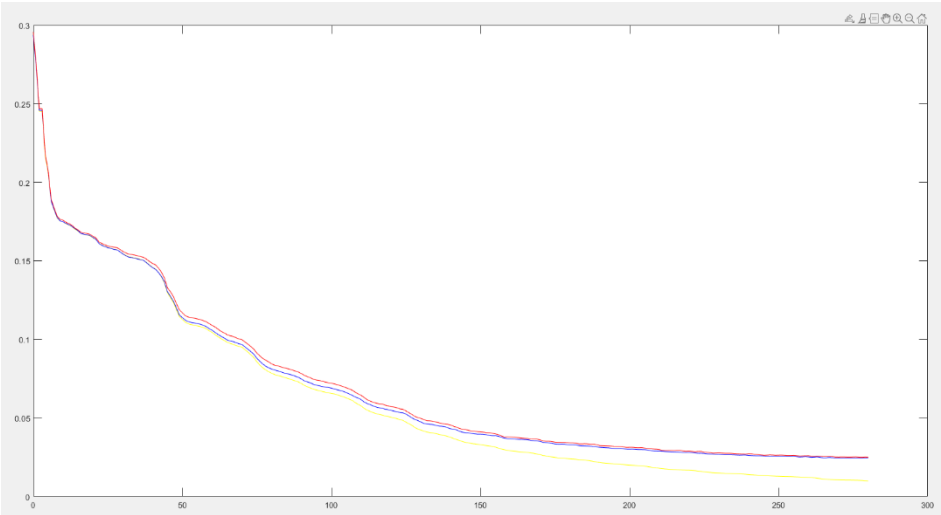
תוצאות התוכנית

במהלך עבודתנו בנינו בלוקים רבים בקוד שלכל אחד מהם הייתה מטרה שונה אשר פועלים יחד כדי להגיע לתוצאה הסופית של התוכנה ולבניית התוכנה בצורה המדויקת והטובה ביותר. בפרק הבא אנו נסכם את התוצאות והתשובות שקיבלנו מכל הבלוקים שיצרנו בשביל לפתח את התוכנה ואת התשובות שהיא מחזירה. בתוצאות נכללת מטריצת הבלבול, מציאת כמות הנוירונים והשורות החבויות המיטביות, מציאת המשקולות, חישוב הטעות, מציאת טעויות אשר חוזרות על עצמן ברשתות שונות, גרפי השגיאה של התמונות ב-train, validation ו- test וכמובן תשובת התוכנה הסופית!

חישוב המשקולות:

בעבודתינו ישנוהבלוק NetBuilder שמקבל מטריצה של כל התמונות ומטריצה בעלת תשובות הקלסיפיקציה הנכונה של התמונות. הבלוק מחלק את הדוגמאות לשלוש קבוצות- אימון, ולידציה ומבחן, בונה רשת נוירונים חדשה ומאמן אותה ובכך קובעת את המשקולות בין כל שכבה ושכבה ברשת ואת הסיפים.

גרפיי השגיאה ב-test, validation and train:



הבלוק (RuNet) משתמש במשקולות הרשת שחושבו, במטריצת התשובות הנכונות ובמטריצת התמונות כדי להריץ בלוק נוסף שמחשב את תשובות הקלסיפיקציה של התמונות. לאחר מכן הבלוק יוצר שלושה גרפים שונים על מערכת צירים משותפת. בצהוב את גרף השגיאה באימון ככל שהרשת מאומנת יותר, בכחול את הוולידציה ובאדום את המבחן.

בדיקה:

בבלוק check ישנה בדיקה של אחוז ההצלחה של כל תמונה בנפרד ושל כל התמונות יחד. דבר זה נעשה על מנת שנוכל לדאוג לכך שאיכות התוכנה והרשת אינן יורדות וכי הרשת עובדת במיטבה.

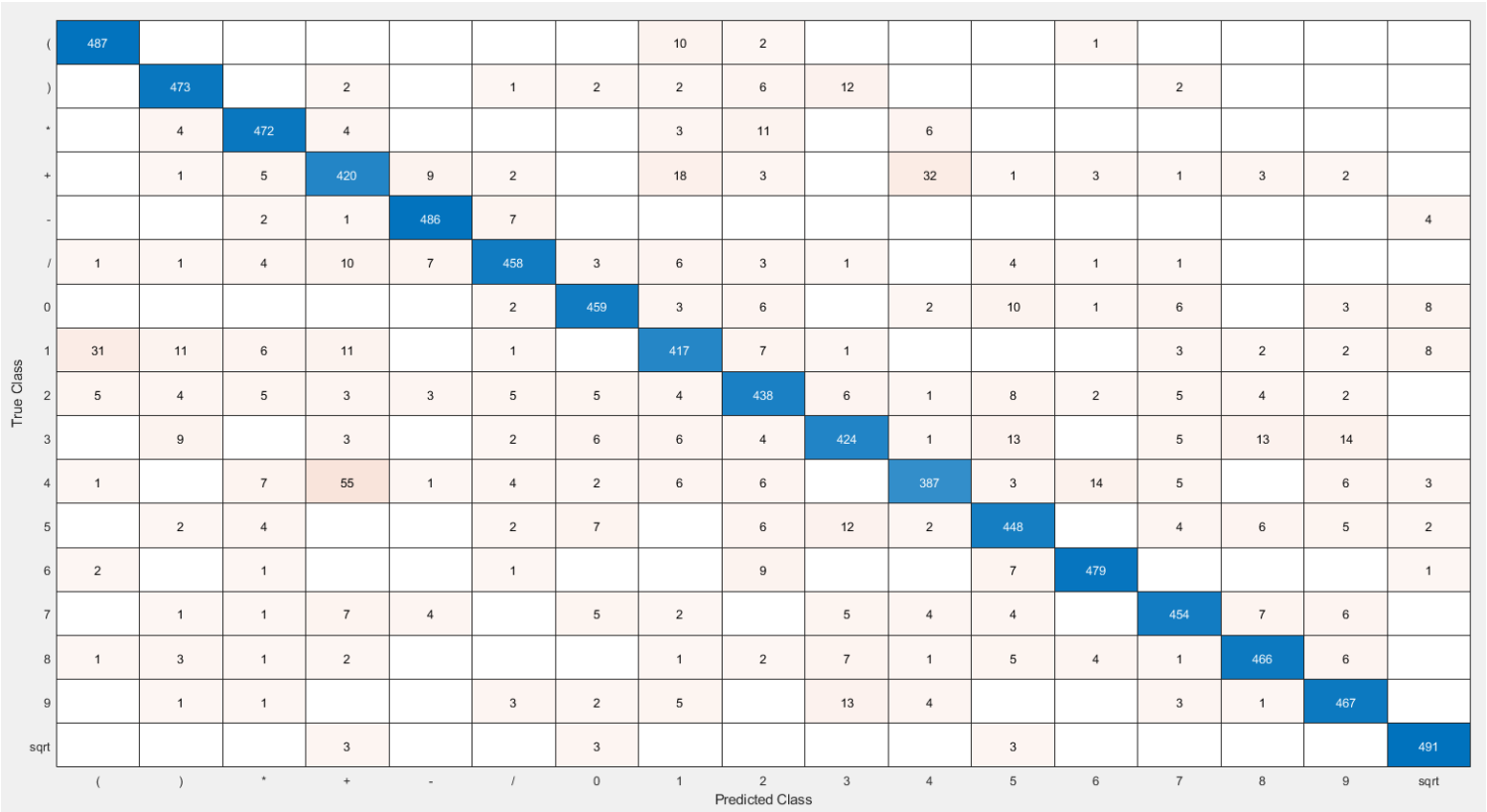
כמות הנוירונים והשכבות המיטבית:

עבודתינו כוללת גם בלוק בשם best2Layers שבונה מספר רשתות שונות, כל אחת עם מספר שונה של נוירונים בשכבות החבויות או מספר שונה של שכבות חבויות, ואת כל אחת מהרשתות התוכנה מאמנת מספר פעמים ובודקת את ממוצע על אחוז הצלחה של כל אחת מהן. חישבנו את זה בצורה הזו בכדי שאחוז ההצלחה יהיה כמה שיותר מדוייק ולא יהיה תלוי בהרצה אחת. לבסוף, היה ניתן לראות כי האופציה הטובה ביותר הינה רשת בעלת שכבה חבויה אחת עם 40 נוירונים, אשר לאחר 20 הרצות הגיעה לאחוז הצלחה ממוצע של 94.17%. בנוסף מצאנו גם כי רשת בעלת שכבה חבויה אחת עם 37 נוירונים הגיעה לממוצע של 93.34%. בעזרת התוכנה מצאנו גם שתי רשתות בעלות שתי שכבות חבויות שהיו מוצלחות ביותר, אחת בעלת שכבה של 42 נוירונים ושכבה של 22 שהגיעה ל - 93.41%, ואחת עם 45 נוירונים ו22 שהגיעה ל - 93.56%.

טעויות שחוזרות על עצמן:

בכדי להעמיק את ההבנה שלנו בנוגע לרשת בנינו גם script שבודק האם בחמש רשתות שאומנו בנפרד יש תמונות שכל הרשתות טעו בזיהוי שלהן.

מטריצת הבלבול –(AKA. Confusion matrix):הבלוק Ball_of_BILBOOL מחשב את מטריצת הבלבול כאחוזים ואת טבלת הבלבול (דרך יפה יותר להצגת המטריצה) ללא



אחוזים. דבר זה עוזר לנו לוודא שאין טעויות אשר חוזרות על עצמן ברשת ושהיא למדה בצורה מיטבית ועובדת כראוי.

פתרון המשוואה:

כמובן שיש גם את התשובה שמגיעה מהבלוק solver - בלוק זה משתמש ברשת שנבנתה ולאחר שלומדה ובנתה את המשקולות והסיפים שלה,ולבסוף מחזירה את התשובה של התרגיל שהוכנס.

דיון בתוצאות

כפי שניתן לראות בפרק הקודם – תוצאות התוכנית, הקוד שלנו בנוי מבלוקים רבים אשר ביחד מרכיבים את התוכנה כולה ועוזרים לבנות אותה. מכך הגענו למסקנות ותוצאות רבות, את חלקן ציפינו לקבל, חלק הפתיעו אותנו לטובה וחלק אף לרעה. את התוצאות הללו ניתן לנסות לשנות בפיתוח עבודתינו בעתיד מחוץ לתחומי התוכנית מדע חישובי פיסיקה במכון דוידסון.

בכל תהליך חישוב המשקולות ובניית הרשת לא הופתענו מהדרך בההרשת פעלה, אך בתהליך קליטת התמונות מתוך התיקיות שלהן במחשב וביצוע ה-resize על מטריצת התמונות, הופתענו לגלות כי לוקח לתוכנה זמן רב מאוד לרוץ, דבר שלא הבנו מדוע קורה וכיצדיש ביכולתנו לשפר את יעילות התוכנה על מנת שהתהליך יהיה מהיר עוד יותר (הרצת הבלוק Preperation לוקחת בערך 2 דקות).

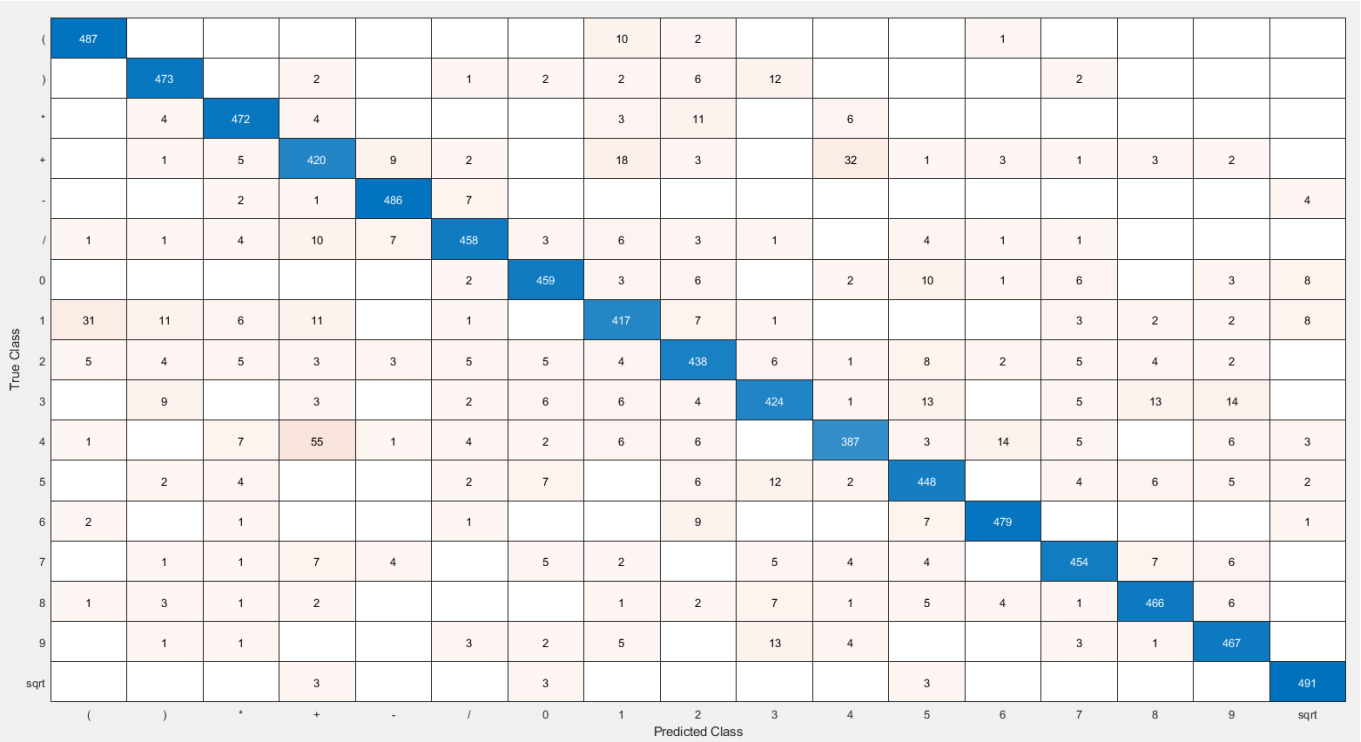
בנוסף, הבלוק המריץ את בניית הרשת ואימונה, גם מצייר גרפים של השגיאה ב -train, validation, test כפונקציה של מספר האיטרציות. גילינו כי הירידה בשגיאה ב -validation וב- test זהה, דבר שציפינו שיקרה,ברם הירידה בשגיאה ב-train משמעותית יותר, דבר צפוי שיקרה כיוון שאלו התמונות עליהן התוכנה מתאמנת. עם זאת, דבר זה גם מראה כי התוכנה טובה יותר בזיהוי התמונות הללו מהרגיל דבר שאיננו רוצים שיקרה כיוון שבסופו של דבר התוכנה נבדקת על תמונות שאינה ראתה בעבר.

בבדיקת הרשת גילינו כי אחוזי ההצלחה בכל אחת מהדוגמאות של המספרים דומים אחד לשני, דבר ששימח אותנו, כיוון שגילוי זה מעיד על כך שהתוכנה עובדת בצורה טובה ולבסוף שמחנו לראות גם כי אחוזי ההצלחה הכוללים של התוכנה נעים סביב ה94% דבר שמעיד כי התוכנה אומנה בצורה פנומנלית.

כאשר הרצנו מספר רב של רשתות שונות עם כמות שונה של נוירונים בכל שכבה חבויה ועם מספר משתנה של שכבות חבויות, השתמשנו בניסוי וטעיה במטרה למצוא את כמות השכבות החבויות והנוירונים בכל שכבה אשר יביאו אותנו אחוז ההצלחה המיטבי. לבסוף מצאנו שתי רשתות עם שתי שכבות חבויות שהגיעו ל93% הצלחה ועם שכבה אחת מצאנו רשת אחת שהגיעה ל- 93% הצלחה ואחת שהגיעה ל94% הצלחה. שמחנו להגיע לאחוז גבוה כזה אבל נראה לנו שאפשרי שאם היינו מבינים בזה יותר היינו יכולים למצוא אולי רשת שיש לה שכבה חבויה אחת או שתיים שתגיע לאחוז הצלחה גבוה יותר מזה! כיוון אליו נוכל להרחיב את עבודתינו בעתיד.

גם בבדיקה האם קיימות טעויות אשר חוזרות על עצמן במספר רשתות שאומנו בנפרד, מצאנו תוצאות שעוררו בנו קמצוץ הפתעה. מצאנו שבכל ריצה של הבלוק ישנן כ- 30-40 תמונות שכל חמשת הרשתות טעו בהן (0.4% מהדוגמאות כולן). לאחר מעבר על התמונות הנ"ל ראינו כי מרביתן הינן תמונות טובות ולא בעייתיות למראית העין ולכן לא הבנו מדוע כל הרשתות טעו באותן התמונות, דבר שהפתיע אותנו ויכול להיות כיוון לשיפור העבודה בעתיד.

גם במטריצת הבלבול היו תוצאות ששמחנו לראות כי הן הוכיחו כי התוכנה עובדת כראוי אך חלק אף הפתיעו אותנו ואנו עוד רוצים למצוא דרכים לתקנן בעתיד. כפי שניתן לראות במטריצה, הרשת עובדת כראוי והרוב המוחלט של כל הסימנים המתמטיים מזוהה כראוי ואין סימן ספציפי שאחוז השגיאה בו גדול באופן משמעותי. עם זאת ניתן לראות שיש בילבול רב בין הסימן '+' והסימן '4'. 32 פעמים התוכנה זיהתה '+' כ- '4' (7% מהדוגמאות של '+') ו55 פעמים היא זיהתה '4' כ-'+' (קצת מעל 10% מהדוגמאות) - דבר שאנו מאוד רוצים לנסות לתקן בעתיד וכרגע מאמינים כי הגדלת מספר הדוגמאות של הסימנים '+' ו-'4' יכולה לתרום לשיפור אחוזי הזיהוי של התווים הללו.



סיכום

העבודה עזרה לנו לשפר את הבנתינו לגבי מספר מושגים.

ראשית כל, חידדנו את הבנתינו לגבי רשת נוירונים מלאכותית. למדנו שבסיס מטרתה של רשת נוירונים מלאכותית הוא לחקות את רשת הנוירונים העצבית הקיימת במוחות בעלי חיים רבים ומגוונים. למדנו שהמכניקה של רשת הנוירונים המלאכותית דומה מאוד לזו של רשת נוירונים ביולוגית, כיצד היא פועלת ומה היישומים שלה בחיי היומיום, ואיך לבנות אותה ב - MATLAB. בנוסף למדנו על התפתחותה של הרשת המלאכותית לאורך ההיסטוריה, ועל האלגוריתמים הבונים אותה.

למדנו את ההבדל בין למידת מכונות לבינה מלאכותית.

בנוסף למדנו מהי מטריצת הבלבול ועל חשיבותה, כיצד יש לחשבה ואף מצאנו פונקציה שמורה ב –MATLAB אשר יוצרת את מטריצה זו ואף צובעת את התאים בצבעים כדי שיהיה יותר קל להבין אותה במהרה - Confusionchart().

פונקציה נוספת אשר השתמשנו בה היא str2sym(), אשר מקבלת מחרוזת הכוללת תרגיל מתמטי ומחזירה פתרון לתרגיל זה.

למדנו דרך לעבור על כל הקבצים הנמצאים בתיקייה מסויימת באמצעות לולאת for העוברת על שמותיהם של הקבצים.

למדנו על עצם מסוג cell ומסוג struct, מה השימושים שלהם וכיצד לעבוד איתם.

אבל הכי הרבה למדנו על עצמנו.

יש לעבודתינו כמה כיוונים שנוכל לפתחם בעתיד:

ראשית, אנו יכולים לשפר את היעילות של התוכנה בכלל וקליטת התמונות בפרט כיוון שזמן הרצת הקוד ארוך ביותר (2 דקות לקליטת התמונות וביצוע ה-resize). בנוסף, למרות אחוז ההצלחה הגבוה של עבודתינו בעזרת מטריצת הבלבול ניתן לראות כי יש תווים מסוימים שהרשת מתבלבלת ביניהם באופן גבוה ביחס לשאר, דבר שנוכל לשפר ולשנות בעתיד, למשל על ידי הכנסת מספר רב יותר של דוגמאות של תווים אלה. דבר נוסף שהיינו רוצים לשנות בעתיד הינו מספר השכבות החבויות ברשת ומספר הנוירונים בכל שכבה ללא פגיעה

באיכות הרשת ואף להעלותה, ולשפר את הדרך למציאת מספר השכבות האופטימלי ומספר הנוירונים האופטימלי בכל שכבה.

נקודה נוספת בה ניגע בהמשך עבודתינו על פרויקט סיום שנת יב במסגרת לימודי מדע חישובי במכון דוידסון, היא היכולת של המחשב לזהות ולערוך קלסיפיקציה על המספרים ופעולות החשבון, כאשר הקלט שמקבלת התוכנה הוא תמונה של תרגיל שלם, ולא תמונות של כל תו בתרגיל בנפרד.

ביבליוגרפיה

1. ויקיפדיה

https://he.wikipedia.org/wiki/%D7%A2%D7%9E%D7%95%D7%93_%D7%A8%D7%90%D7%A9%D7%99

2. המכלול

https://www.hamichlol.org.il/%D7%A2%D7%9E%D7%95%D7%93_%D7%A8%D7%90%D7%A9%D7%99

3. Davidson moodle

https://davidson.weizmann.ac.il/tags/moodle

4. Wikiwand

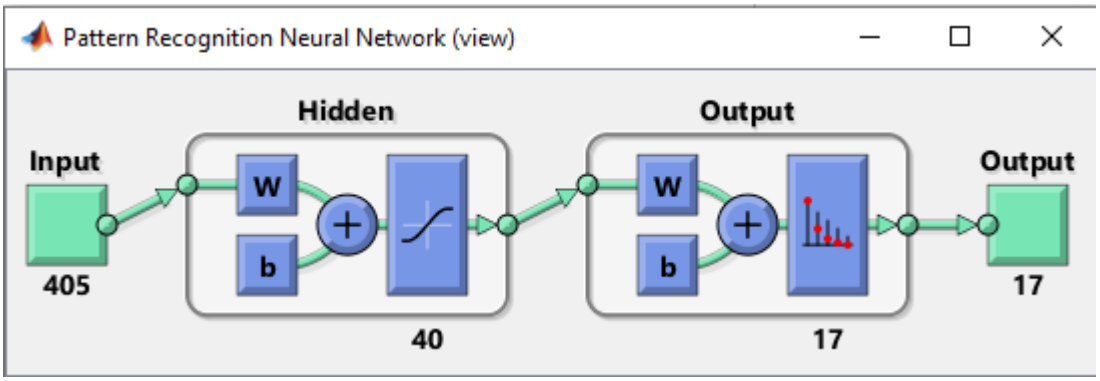
https://www.wikiwand.com/

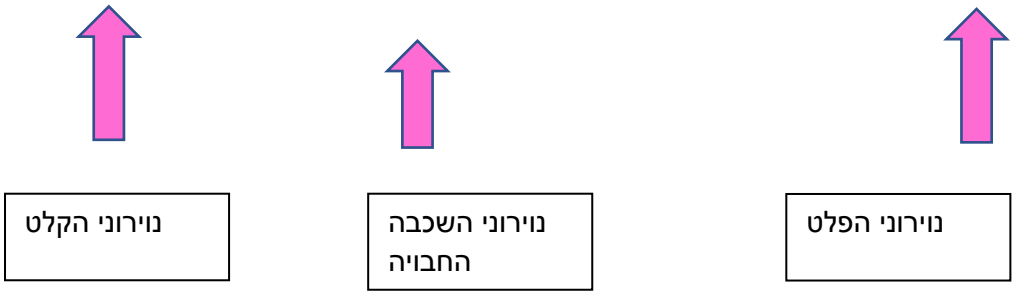
5. MATLAB

https://www.mathworks.com/products/matlab.html

נספחים

נספח 1: מבנה רשת הנויורונים עם שכבה חבויה אחת



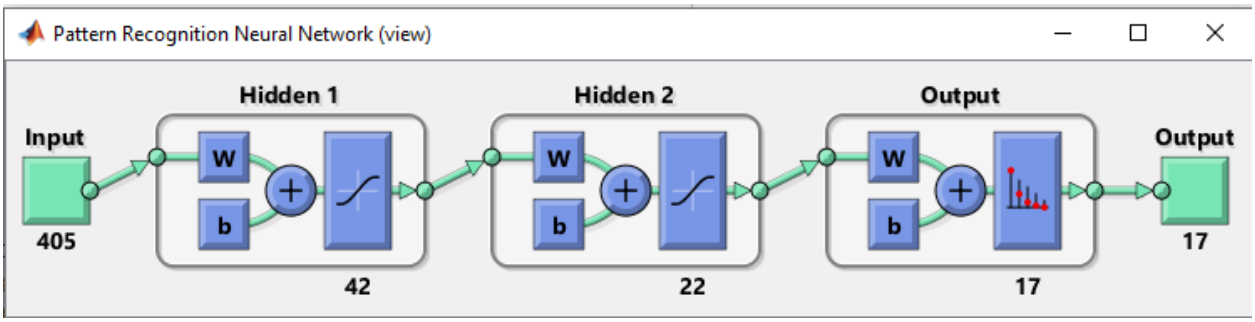


W: משקולות

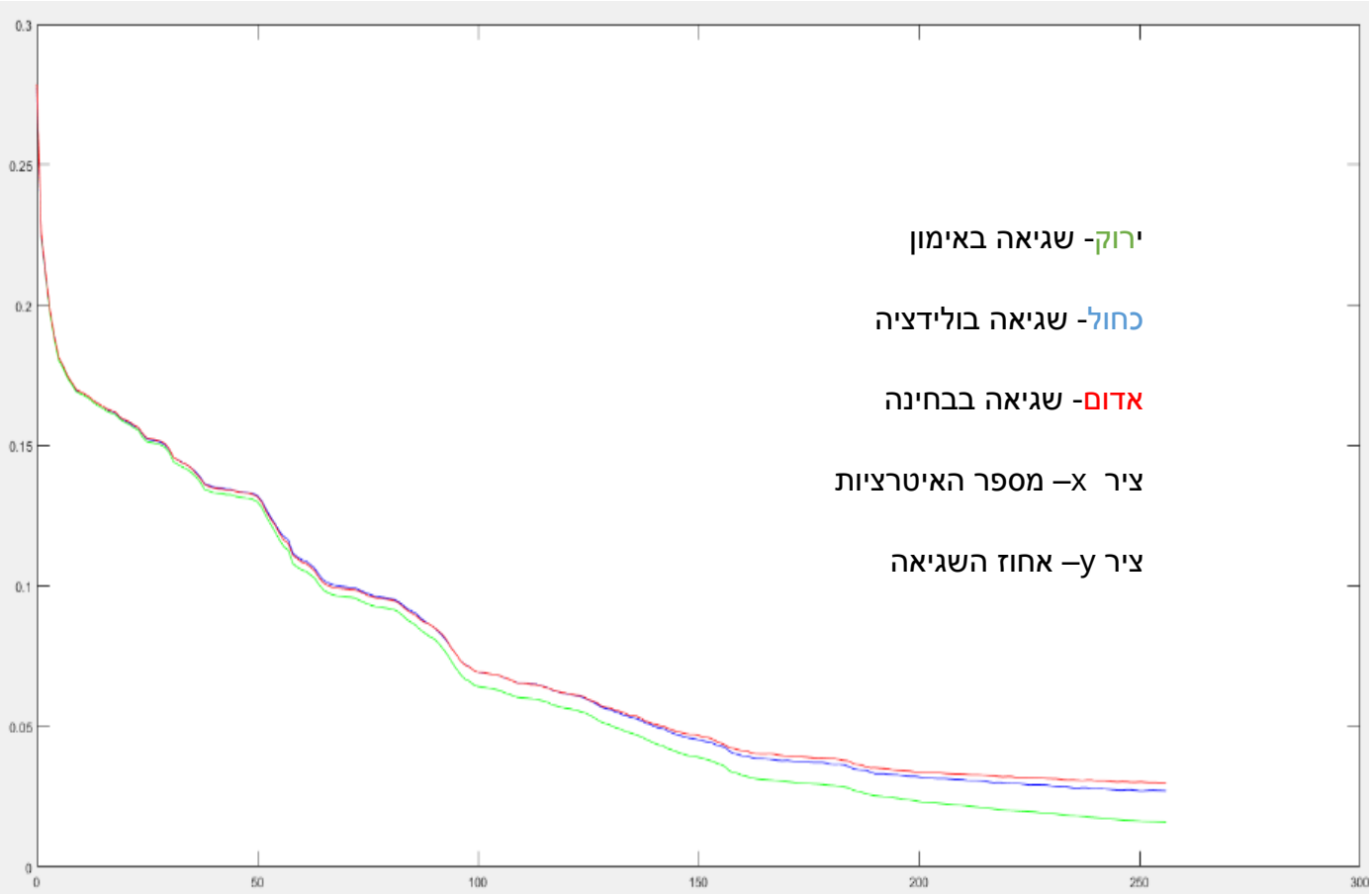
B: סיפים

+: פונקציית המעבר

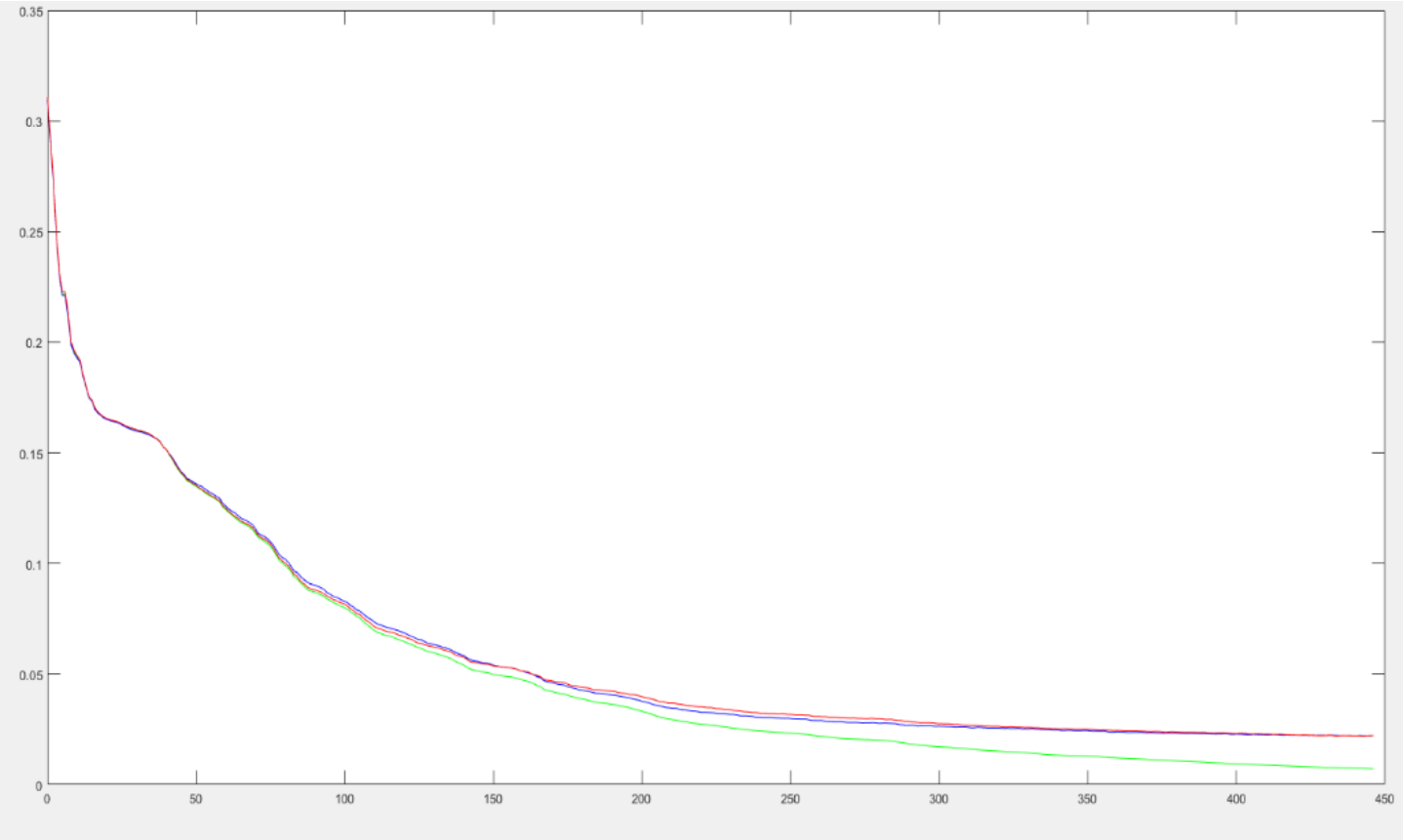
נספח 2: מבנה רשת הנוירונים עם שתי שכבות חבויות



נספח 3: פונקציות השגיאה בתהליך אימון הפונקציה, שכבה אחת. 42 נוירונים



נספח 4: פונקציות השגיאה בתהליך אימון הפונקציה, שתי שכבות. 32X42



נספח 5: מטריצת הבלבול ברשת בעלת שכבה אחת עם 42 נוירונים

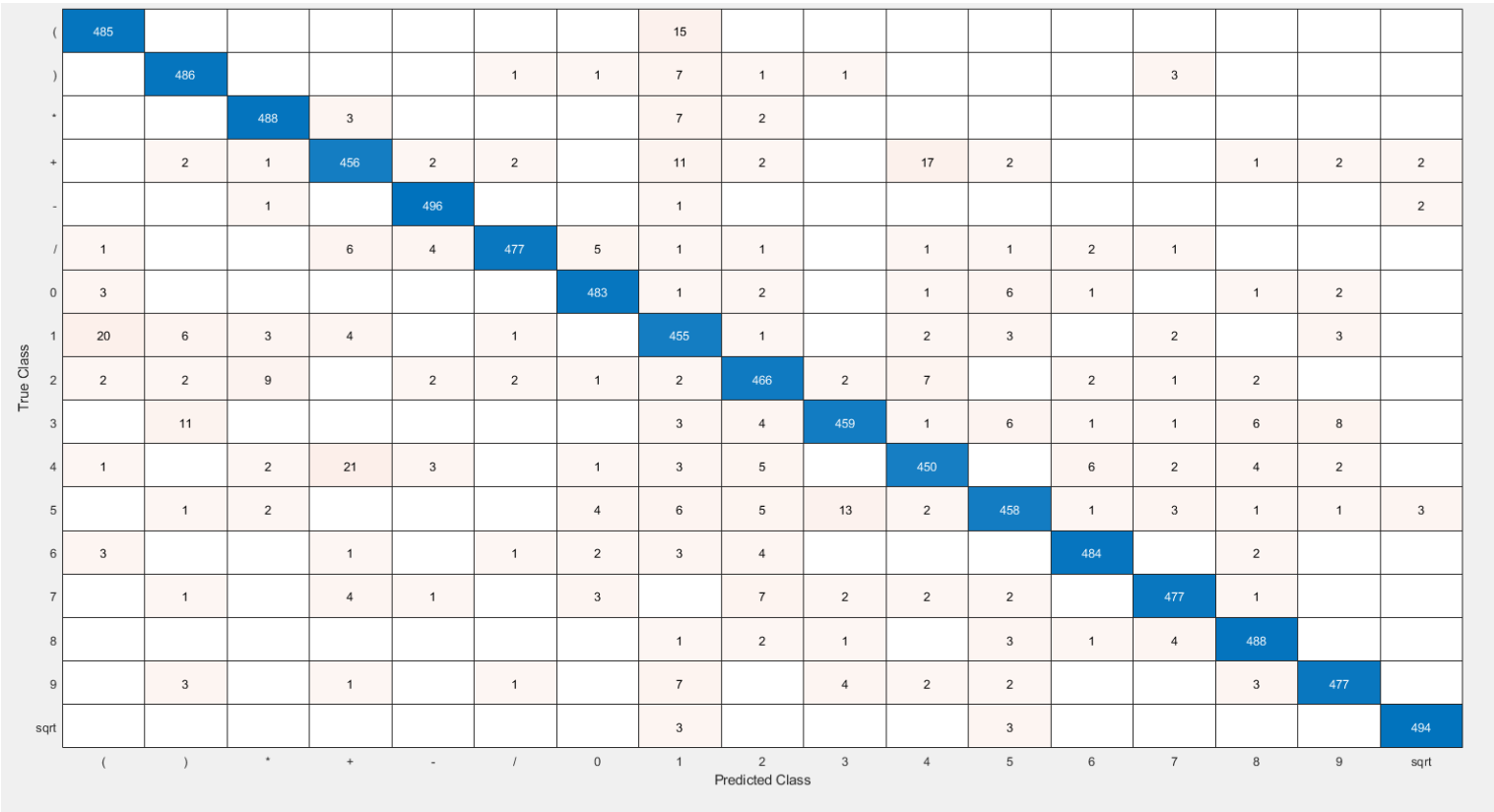
ציר x- הסיווג שנעשה ע"י הרשת

ציר y- התשובות הרצויות

המטריצה באחוזים:

91	0.2000	0	0.4000	0.6000	0	0.4000	0	0.6000	0	0	0.8000	0.2000	0.6000	0	4
0.4000	93.2000	0.4000	1.4000	0	0.4000	0.2000	0.4000	0	0.2000	0.4000	0	0.4000	1.8000	0	0.4000
0.6000	0.8000	91.8000	0.2000	1.2000	0.2000	0.2000	1.2000	1.6000	0	0	0	0	0	0	0
0.6000	1	0	90	0	1.2000	0.4000	0.8000	0.4000	0.2000	0.6000	4.2000	0	0.4000	0	0.2000
1.2000	1	2.6000	0.4000	91.6000	0.2000	0.6000	0.2000	0.2000	0.8000	0	0	0	0.4000	0.6000	0
0.6000	0.8000	0	0	0	96.8000	0	0.4000	0	0.4000	0	0.2000	0.2000	0	0	0.6000
0	1.4000	0.4000	0.4000	0.4000	0	95.4000	0.2000	0	0.6000	0.2000	0.8000	0	0	0	0
0.2000	0.4000	0.2000	0	0.6000	0.2000	0.8000	97.6000	0	0	0	0	0	0	0	0
1.4000	0	0.8000	0.4000	0.4000	0	0	0.6000	95.4000	0	0	0.2000	0.2000	0	0	0
0.2000	0.4000	0	0.2000	1.2000	0.2000	0	0.2000	0.4000	96.6000	0	0	0	0	0	0.6000
0.2000	0	0	0	0	0	0	0	0	0	99.2000	0	0	0.2000	0.4000	0
2.2000	0.4000	0	3.4000	0.4000	0	0	0.2000	0.4000	0	0.4000	91.2000	0.4000	0.2000	0.4000	0
0.2000	0.2000	0	0.2000	0.2000	0.4000	0.2000	0	0	1	0.8000	1.2000	95.4000	0	0	0.2000
1.4000	0.4000	0	0	0	0	0	0	0	0	0	0.6000	0	97.6000	0	0
0.6000	0	0	0	0.6000	0	0	0	0	0	0	0	0	0	98.8000	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	97
1.4000	0.2000	0.2000	0	0	0	0.6000	0	0	0.2000	0	0	0.2000	0	0	0

המטריצה בכמויות:



נספח 6: מטריצת הבלבול ברשת שתי שכבות חבויות עם 32X42 נוירונים.

המטריצה באחוזים:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	88	0.2000	0	0.2000	0.2000	0.2000	0.8000	0	0.8000	0.6000	0	2	0.2000	1	0.2000	4.6000	1
2	0.8000	87.2000	1.6000	1.8000	0.8000	0.4000	0.4000	1.2000	0	0.8000	0.8000	0.4000	0.8000	1.4000	0	0.8000	0.8000
3	0.4000	0.8000	90.8000	0.2000	1.8000	0.2000	0.8000	1.2000	1.6000	0.4000	0	0	0.2000	0	0	0	1.6000
4	0.4000	2	0	85	0.2000	0.8000	0.8000	0.6000	0.8000	0.2000	0.4000	6.2000	1	1	0	0	0.6000
5	0.2000	1.2000	2.2000	0.2000	92.2000	0.6000	1.2000	0	0.6000	0.2000	0	0.2000	0	0.4000	0.6000	0	0.2000
6	0	1.4000	0	0	0.6000	95.8000	0	0.4000	0	0.2000	0	0.2000	0.2000	0	0.2000	1	0
7	0	0.6000	1	1.8000	0	0	92.4000	0.8000	1.2000	0.2000	0	1.2000	0.4000	0.2000	0	0	0.2000
8	0	0.2000	0.8000	0.4000	0.2000	0.6000	0.6000	95.8000	0.2000	0	0	0.2000	0	0.4000	0	0.6000	0
9	1	0.6000	1.8000	0.8000	0	0	0	0.6000	94.4000	0.2000	0	0.2000	0.4000	0	0	0	0
10	0	0.8000	0	0.4000	0.4000	0.2000	1	0	0	96.2000	0	0	0.4000	0	0.4000	0.2000	0
11	0	0.2000	0	0	0	0	0	0	0	0	98.2000	0.6000	0	0.2000	0.8000	0	0
12	1.4000	0.6000	0	6.2000	0.2000	0	0	0.6000	2	0	0.8000	85.6000	0.4000	0.6000	0.4000	0.6000	0.6000
13	0.4000	0	0	0	0.2000	0.2000	0.2000	0	0	0.8000	1.6000	1.4000	94.2000	0.2000	0	0.6000	0.2000
14	0.6000	1	0	0.2000	0	0	0.6000	0	0	0	0	0.8000	0	96.8000	0	0	0
15	0	0	0	0	0.6000	0	0	0	0	0	0	0.6000	0	0	98.8000	0	0
16	2.6000	0.2000	0	0	0.2000	0	0	0	0	0	0	0	0	0	0.2000	96.8000	0
17	1.8000	1	0.2000	0	0	0	0	0	0	0.2000	0	0	0.6000	0	0	0	96.2000

המטריצה בכמויות:

True Class \ Predicted Class	(	)	*	+	-	/	0	1	2	3	4	5	6	7	8	9	sqrt
(	484							13	1			1					1
)		481				3	1	9	5	1							
*			484	4				3	5		1			3			
+	3	3	3	428	4	2		7	3		31	1			3	10	2
-			1	3	491				1								4
/	3	1	1	7	8	471	4	2				1	1	1			
0	1					2	481		4		2	2	1	5			2
1	23	5	5	10		1	3	440	1		1	1	1	4		4	1
2	4	4	7	2	4	4	4	4	436	8	9	4	2	2	6		
3		8				1	2	2	4	454	1	9	1	4	6	8	
4		3	5	31	2	5	1	2	10		425	1	4	4	3	4	
5		1	2	1			1	1	6	11	1	461	3	6		3	3
6	5			1		1	1		7			3	479		2		1
7		1	1	6		2	1		3	5	9			462	4	6	
8	3		2	1					1	4	2	1	3	3	479	1	
9				1		2	1	5	3	9	4				3	472	
sqrt				3								3					494

הוראות להרצת התכנית

1. שמירת הקבצים הנחוצים באותה תיקייה.
2. הרצת בלוק התכנה Preparation. בלוק זה מאתחל את ריצת התכנה. הבלוק עובר על התמונות ששמורות בdatabase ושומר אותן כמטריצה לעבודה קלה יותר בעתיד. בנוסף קטע קוד זה מעבד את התמונות ומקטין את הרזולוציה שלהן כדי לקצר את זמן הריצה של הקוד בהמשך.
3. *אופציונלי* שמירת המשתנים מרשת מוצלחת קיימת על המחשב על מנת להריץ את התכנה על רשת בעלת אחוז הצלחה גבוה יותר אחר כך.
4. הרצת הבלוק RuNet, בלוק זה מתשמש בבלוקים רבים אחרים שכתבנו ובונה רשת או טוען רשת שמורה, ומחשב את הסיווג שהיא עושה. לאחר מכן הבלוק מצייר את גרפי הכישלון בסטי האימון הולדיציה והבחינה על מערכת צירים משותפת ומחשב את אחוז ההצלחה בסיווג של הרשת. אחוז ההצלחה יישמר במשתנה "correct". הסיווג שהרשת עשתה יישמר במטריצה 'a'.
5. שמירת תמונות של מספרים או פעולות מתמטיות בתיקיה "equationPics", צריך לשמור את התמונות לפי סדר התרגיל ולשנות את שמן כך שהתו הראשון יהיה הסמל '#'
6. הרצת הבלוק solver. בלוק זה ניגש אל התמונות שנשמרו בתיקייה "equationPics" ובעזרת הבלוק "DataNetCalc1Layer" מזהה את המספרים או הפעולות המתמטיות הנמצאות התמונות. לבסוף הבלוק משתמש בפעולת str2sym של MATLAB כדי לחשב את התרגיל שהתמונות יוצרות.
7. בלוק אופציונלי להרצה הוא "Ball_of_BILBOOL". בלוק זה בונה את "מטריצת הבלבול" (Confusion Matrix) ומציג אותה באחוזים. ניתן להריץ את הבלוק "CommonMistakes", הבלוק מחזיר את הוקטור "lo_yutslach" שמכיל את האינדקסים של התמונות שהרשת חישבה לא נכון בחמש רשתות נוירונים שונות. הקוד מכיל גם קטע המצייר את התמונות הללו. בלוק נוסף שאפשר להריץ את הבלוק "Best2Layers". בלוק זה מחשב מהי כמות הנוירונים הגורמת לאחוז הצלחה אידאלי של הסיווג ברשת.