מסמך אפיון פרויקט במקצוע הגנת סייבר

מגיש – תומר שמואל אשר, כתה י"ב 4

נושא הפרויקט - "GoodEyedea" – מערכת לזיהוי והתראה על עיניים עצומות של המשתמש מתוך פלטפורמות שונות.

הגדרת הפרויקט

הפרויקט GoodEyedea נותן מענה לאנשים שלא רוצים להירדם, לעצום את עיניהם או שדעתם תוסח משדה ראייה מסוים בסיטואציות שונות המאפשרות את המצאות הטלפון הנייד ו/או מצלמת הרשת של מחשב כלשהו בסביבה, כגון:  
1. נהיגה, כאשר הטלפון נמצא על דיבורית – דבר שיכול להציל חיים של אנשים, שהרי במידה והנהג ידע כשהוא נרדם או כאשר הוא מוסח דעת, יוכל לעצור ולהתרענן ולמנוע תאונה.  
2. פגישה חשובה, שבה כדאי לא להירדם מסיבות שונות ומגוונות.  
3. במהלך עבודה על המחשב, למניעת לחיצות מיותרות בעקבות הירדמות על המקלדת, ומניעת טעם לפגם בעבודת המשתמש בשל העייפות שלו על ידי התראה במקרה של הירדמות.

המוצר מספק שירות למשתמשים רשומים, כך שאחד מן התהליכים שנדרשים מכל משתמש מתחיל באפליקציה הוא יצירת חשבון שמאופיין בשם, שם משתמש, סיסמא ומייל. כל הפרטים ההלו נשמרים בצורה מאובטחת בשרת מרוחק בבסיס נתונים יחסי (SQL), כאשר גם התקשורת בין השרת ללקוח (מכשיר האנדרואיד או המחשב האישי) מוצפנת בצורה שוטפת באמצעות שיטות הצפנה שונות שמבטיחות סיכוי אפסי לפריצה.

לאחר הרישום, כל משתמש רשאי להפעיל את ה "Monisession"[[1]](#footnote-1) – המעקב של המודל המאומן אחר עיני המשתמש. המעקב ישלח תמונות לשרת מרוחק שיחזה את מצב העיניים של המשתמש, ויחזיר תשובה למכשיר הלקוח, שיגיב בהתאם לתשובה. במידה והעיניים עצומות בעיקרן, המכשיר ירטוט או יצפצף או ישמיע שיר לבחירת המשתמש.

התכנה תפעל בצורה חכמה, כאשר היא תיקח באופן סטטיסטי על כל כמה תמונות מידע ותקיים תחזיות עליהן, דבר שימנע אזעקות שווא מצד אחד ומצד שני ימנע רשלנות מצד התוכנה, שנובעת ממניעת חוסר בהתראות אמת.

לכל אורך פעולת התכנה המשתמש יוכל לשנות את ההגדרות הבסיסיות של התכנה שיהיו עוצמת קול, סוג ההתראה (איזה צלצול), הכללת רטט ועוד.

אוכלוסיית היעד של התכנה מגוונת מאוד, וכל מי שרוצה להשתמש בשירות ובעל מכשיר סלולרי או מחשב אישי עם מצלמת רשת, יוכל להשתמש בו, שהרי הירדמות היא פעולה שקורית בכל הגילאים במסגרות שונות, ולכן השימושים למוצר יכולים להיות רבים ומגוונים, וכך גם קבוצת הגיל שמתאימה לשימוש במוצר.

חקר מוצרים

קיימים שלושה מוצרים עיקריים ומשמעותיים בשוק שמספקים תוכנה לזיהוי עיניים עצומות/פקוחות, או יכולת ליצירת תוכנה כזו בצורה פשוטה למדי.

Google Vision:

גוגל נותנת גישה לממשקי תכנות יישומים (API) רבים ומגוונים, שרבים מהם עוסקים במוצרי למידת מכונה. אחד מהממשקים שהחברה הנ"ל מספקת, היא ממשק ה Google Vision. הממשק הזה מייצר יכולת משמעותית ביותר בכל הנוגע לזיהוי אובייקטים ותכונות בתמונה בטכנולוגיה של למידת מכונה – רשת נוירונים, והוא יכול לתת מידע רב על תמונה נתונה. אחד ממרכיבי המידע שגוגל מספקת היא על פרצופים, באמצעות האובייקט 'Face'. דרך אובייקט כזה (שמתקבל בעקבות קלט של תמונה), ניתן לקבל את הסיכוי לכך שעין מסוימת פקוחה או סגורה.

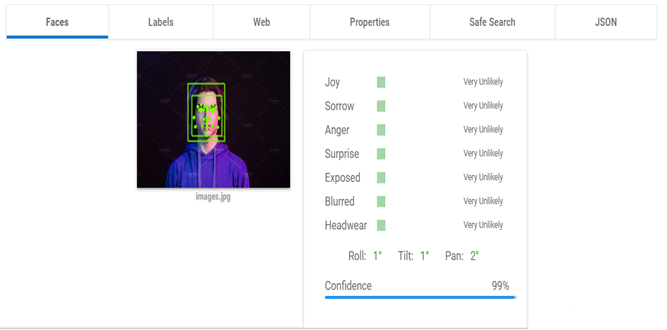
יצירת מזהה עיניים על ידי הממשק של גוגל:

Import google-cloud API and feed it with your image.

Make a 'Face' object.

Use Google's Prediction for closed / open eyes for a frame

Manipulate the raw output and present the outcome.



בתמונה: הממשק Google Vision מנתח את התמונה שמזינים לתוכו. מדובר באתר אינטרנט[[2]](#footnote-2) שמציע טעימה לשירותים שהממשק מציע, ולכן לא רואים בו את כל האפשרויות שיש לו להציע, כמו "האם העיניים סגורות או פקוחות".

Denso's Driver Monitor & FotoNation ADSD:

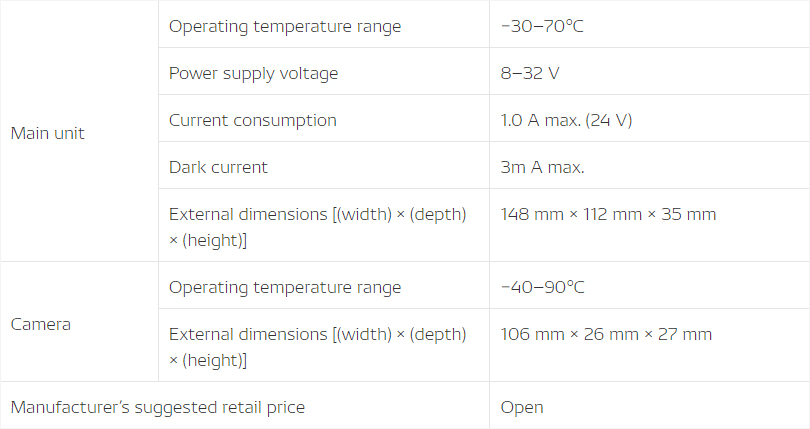
חברת DENSO, אחת מספקי חלקי הרכב הגדולים בעולם, הוציאה לשוק מוצר שעוקב אחר מצב נהג, כדי לסייע בהקטנת מספר תאונות הדרכים בכלי רכב מסחריים, כגון משאיות ואוטובוסים. מוצר הבטיחות בודק הסחות דעת, נמנום, שינה ותנוחה בלתי הולמת המבוססת על תמונת הפנים של הנהג, שנלכדת באמצעות מצלמה המותקנת בתא הנהג. אם זוהתה נהיגה מנומנמת או מוסחת, המערכת תתריע.

מצבו של הנהג מזוהה על ידי המוצר ונרשם על כרטיס SD. מנהל תפעול ומנהל ההתקן יכולים לסקור את מצב הנהיגה, כולל מספר ההתראות שהושמעו בזמן אמת או בדיעבד, כך שניתן יהיה להזהיר את הנהג ולנקוט בפעולה מהירה במקרה חירום.

בשנת 2017, DENSO שיתפה פעולה עם FotoNation, שיוצרת זיהוי תכונות בתמונות של פנים באמצעות טכנולוגיות רשת נוירונים, כדי לשפר עוד יותר את הביצועים של זיהוי מצב הנהג ולהאיץ את הפיתוח של המוצר.

חשוב לציין שהמוצר מתאים לרכב בעיקר, והוא בנוי מהתקן מיוחד לרכב.

טבלת המפרט הטכני של המוצר של חברת Denso:



משמאל: המצלמה השחורה, ויחידת החישוב הלבנה של המערכת, אותן מתקינים בכלי רכב.



מימין: מערכת המעקב של Denso ו FotoNation ברכב במקומה האמתי, בו היא מבצעת את פעולתה.



קוד שפורסם באינטרנט על ידי [**Adrian Rosebrock**](https://www.pyimagesearch.com/author/adrian/):

לא מדובר באפליקציה מסודרת, אלא בקוד ששיתף אדריאן רוזברוק באינטרנט בשנת 2017. הכלים שהוא משתמש בהם הם כלים של זיהוי פנים ותווי פנים על ידי מיקומים, ולאחר זיהוי מיקום העיניים, התייחסות למצב העיניים (האם סגורות או פקוחות) על ידי מציאת "הלבן של העיניים" כפי שהוא מתאר זאת במאמר, באמצעות שימוש במודולים מתקדמים של שפת פיתון, דוגמת cv2 שתפקידה בפרויקט זה לצלם, לטפל בענייני תמונות ווידאו ו dlib שמבצעת את זיהוי מיקום הפנים, העיניים, וההחלטה האם העיניים פקוחות או סגורות על פי "הלבן של העיניים" תוכלו לעיין בקוד של אדריאן בסמלון שמתחת לשורה זו.



יש לציין שהכלי שבו הוא משתמש מבצע את המשימה בקלות יחסית, אך מצד שני מכסה הרבה פחות מקרים שבהם התוכנה יכולה לעבוד מאשר טכנולוגיה כמו רשת עצבית. לדוגמה: האלגוריתם שמוצא "לבן בעיניים", יכול להיות מוטעה על ידי תמונה של אדם עם מחלות כבד בעיקר, שצובעות את עינו בצהוב, כגון אי ספיקת כבד, בעוד שרשת נוירונים יכולה "לדלג" מעל כל יוצא דופן בתמונה, כל עוד האימונים שלה היו מספיק מקיפים ואינטנסיביים.

הנה [קישור לסרטון](https://www.youtube.com/watch?v=0w8FaiKP5h8) בו רוזברוק מראה את האפליקציה שלו בפעולה.

השוואה בין מוצרים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| קריטריונים | Denso ADAS | Google vision | הקוד של רוזברוק | GoodEyedea |
| Cross Platform | לא. מצוי רק במכשיר שלהם. | כן. שימוש ב API בשפות שונות ממכשירים שונים. | באופן חלקי. שפת פיתון מאפשרת Cross Platforming מוגבל, ללא סלולר לדוגמה. | כן. המערכת תומכת במגוון רחב של סוגי מכשירים,  מסוגים מגוונים של מחשבי PC ועד למכשירי Andriod. |
| רמת גימור בתור מוצר עצמאי | רמת גימור גבוהה. מדובר במוצר עצמאי שיצא לשוק. | רמת גימור בינונית. למרות שהממשק נגיש מאוד ומהווה מוצר בעצמו, הוא אינו מהווה מוצר מוצהר למעקב אחרי עיניים ותשומת לב של בני אדם. | רמת גימור בינונית. מדובר בקוד שפורסם באינטרנט, ולא במוצר נמכר. | רמת גימור גבוהה, ופוטנציאל לצאת לשוק כאפליקציה ל Android ומחשבי PC. |
| ריבוי שימושים | לא. מדובר במכשיר שנוצר אך ורק לשימוש של מעקב אחר נהיגה. | כן. מדובר בממשק שמאפשר להשתמש בו בכל צורה, ועל כן הגמישות שלו גבוהה והשימושים שלו רבים. | לא. מדובר בשימוש יחיד ללא מטרה מוגדרת של הטכנולוגיה. | כן. בשל הגמישות מבחינת פלטפורמות, השימושים למוצר יכולים להיות רבים ומגוונים, משמירה על ערנות בעבודה על מחשב, ועד שמירה על קשב וערנות בנהיגה. |
| שילוב התאמות אישיות | יש. | יש, בהתאם לכותב האפליקציה שמשתמשת בממשק זה. | אין. מדובר בטכנולוגיה טהורה. | יש, בעזרת שילוב של DB בצד השרת, כגון צליל התראה. |
| טכנולוגיה | Convolutional Neural Network | Convolutional Neural Network | עיבוד תמונה "רגיל". | Convolutional Neural Network |

חקר פיתוחי

רשת נוירונים:

רשת נוירונים היא כלי חישובי לקבלת החלטות שנמצא בקטגוריה של בינה מלאכותית ולמידה עמוקה.

חוקרים רבים מצאו עוד לפני המצאת רשת הנוירונים ששאיבת השראה מהטבע לטובת הטכנולוגיה היא דבר יעיל מאוד ולו בלבד שלטבע ולאבולוציה ניסיון גדול בהרבה יותר מאשר לתבונת בני האדם.

מהם נוירונים?

הנוירונים הם חלקים במוח, שעל ידי תהליכים אלקטרו כימיים מביאים בשיתוף פעולה ביניהם להחלטות שהאדם מייצר.

רשת הנוירונים היא למעשה חיקוי חלקי ומנוון מאוד של פעולת המוח שלנו. על כן שמה – רשת נוירונים.

את מבנה הרשת ופעולתה בחרתי לתאר בשני שלבים.

השלב האינסטינקטיבי ידון ברשת הנוירונים ממעוף הציפור, בצורה שתבהיר את רוח הדברים בעוד שהשלב הפרקטי יאיר את אופן הפעולה המעשי של רשת הנוירונים – את מה שקורה "מאחורי הקלעים", מה שישלים את הבנתנו בנושא.

השלב האינסטינקטיבי:

ניתן להגיד באופן הפשוט ביותר שרשת נוירונים היא תוכנה אשר "לומדת כמו תינוק קטן". כפי שתינוק קטן מנסה ללכת את צעדיו הראשונים ונופל מפני שאינו יודע איך ללכת, כך המכונה מתחילה את דרכה. היא מבצעת החלטות חסרות היגיון ומקבלת תיקונים מגורם חיצוני שגורם לה להבין לאיזה כוון לשנות ערכים בתוכה כדי לשפר את פעולתה[[3]](#footnote-3), ממש כפי שהתינוק, שלא הצליח בפעם הראשונה ללכת על רגליו, לא ינסה ללכת על ידיו כתחליף, אלא ימשיך בדרך הקלה ביותר להליכה וגורמת לו ליפול פחות, בדרכים הדומות לדרכי ההליכה של הסובבים אותו.

דוגמה נוספת לפעולת רשת נוירונים יכולה לבוא לידי ביטוי בניסוי רעיוני נחמד. נתאר מצב בו אדם ללא ניסיון טיסה מתבקש להטיס סימולטור של מעבורת חלל. משימתו היא להטיס את מעבורת החלל אל החלל החיצון ולנחות חזרה במרכז החלל ג'ונסון בטקסס. בכל פעם שהוא נכשל הוא אף ידאג להבין מה גרם לו להיכשל. מובן שהסיכוי שהטיסה הראשונה תעבור באופן מוצלח יהיה אפסי. עם זאת, ניתן לומר שאם זו תהיה משימת חייו, ומספר הניסיונות שלו יהיה בלתי מוגבל, האדם יבין לאט ועם הזמן את העקרונות שבהטסת המעבורת ואולי לאחר שנים רבות הוא יצליח לבצע את המשימה. חשוב להדגיש שבמידה והוא יגיע למצב שבו הוא יצליח במשימתו, לא תהיה לו בעיה להנחית את המעבורת גם בבסיס צאלים. היכולת שהוא רכש תהיה מספיק גמישה ליישום במידה ותנאי הסימולציה היו גמישים גם הם.

נתייחס לרשת כאל ישות לומדת שמקיימת שני תהליכים עיקריים בלמידתה:

תהליך הניסוי:

הרשת מוציאה פלט

הרשת מקבלת קלט

המידע רץ על פני הרשת

תהליך הלמידה והסקת המסקנות:

הרשת מסתגלת ומתקנת את עצמה בהתאם לטעות שביצעה

הרשת מקבלת את הפלט שהיה אמור להתקיים בהינתן הקלט שמתואר לעיל

בהמשך, נכנה כל אחד מהתהליכים בשמותיהם המקצועיים:

תהליך הניסוי: Feedforward.  
תהליך הלמידה והסקת המסקנות: Back Propagation.

השלב הפרקטי:

Feedforward:

את רשת הנוירונים מקובל לחלק לשלוש שכבות: שכבת הקלט, השכבה החבויה[[4]](#footnote-4), ושכבת הפלט.

כל שכבה בנויה מתאי מידע (נוירונים).

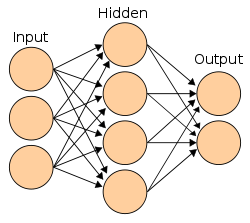
כל נוירון מחובר לכל נוירון בשכבה שלפניו ובשכבה שאחריו.

תפקיד הנוירון הוא להחזיק מספר. כל חיבור בין נוירונים מאופיין במספר נוסף הנקרא "משקל".

משקל מבטא את מידת ההשפעה של הנוירון בשכבה הקודמת על הנוירון בשכבה המתקדמת. זאת אומרת, כל נוירון בכל שכבה אחראי ושם דגש בצורה שונה על הקלטים שלו, מה שמשפיע על ערכו, זאת משום שהערך שנקבע אצלו הוא בעצם סכום מסוים של כל הקלטים שלו. הקלטים מורכבים מהמספר שמחזיק הנוירון בשכבה הקודמת יחד עם ערך המשקל המתאים. התהליך מתבצע משכבה לשכבה, עד לשכבת הפלט, כך שהפלט שמתקבל, באופן אידאלי מבטא את הפלט הרצוי בהינתן הקלט. המצב הנ"ל מתאים בהקבלה להצלחת הנחתת המעבורת ע"י טייס ההדמיה.

למרות זאת, הבנו שאדם אינו יכול להיכנס בפעם הראשונה לסימולטור ולהצליח לבצע משימה מסובכת שכזו. לכן, עליו לעבור תהליך פנימי מסוים שיגרום לו להשתפר בעבודתו עד שיגיע למצב הזה. על התהליך הזה, נדבר בפרק הבא.

נהוג לשרטט רשת נוירונים באופן הבא:



שכבה - במקרה זה, החבויה

נוירון והערך שמחזיק

משקל והערך שמחזיק

0.5

0.5

כעת, בכדי להבין איך מתבצע במציאות אופן מעבר המידע ברשת הנוירונים נעמיק בנושא וראה אותו בצורה מתמטית.

נתחיל מהמשוואה הסופית, ונפרק אותה.

והנה המשוואה הראשונה בכתיב אינטואיטיבי יותר:

m – מספר הנוירונים בשכבה ה L.

n – מספר הנוירונים בשכבה ה (L – 1).

L – מספר השכבה שאליה מועבר המידע.

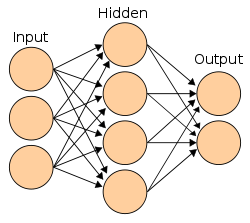
נתחיל בכך שהאות a מייצגת שכבה מסדר מסוים. זאת אומרת, מערך, או מטריצה בעלת טור אחד של נוירונים. נבחין בכך שזוהי משוואה רקורסיבית, כאשר השכבה הראשונה נתונה כשכבת הקלט הנתונה בידי המתכנת. w מייצגת מטריצה של משקלים בין שתי שכבות, כאשר כל טור במטריצה היא סט המשקלים אשר "יוצאים" מנוירון מסוים בשכבה L-1(ראה סרטוט). b היא מטריצה גם כן, בגודל המטריצה a, והיא קובעת מאיזה ערך, המתכנת רוצה שהנוירון יחשב זניח. הפונקציה היא פונקציית אקטיבציה והיא עוזרת לשמור על הסדר בערכי הנוירון, על ידי כך שהיא כולאת כל ערך נתון בין שני ערכים מוגדרים[[5]](#footnote-5) (או מעבירה ערכים תהליכי נרמול אחרים). מטרתה בעיקר למקסם את יכולת הלמידה של הרשת על ידי יצירת ערכים קרובים או "נורמלים" בכל חלקי הרשת.[[6]](#footnote-6)[[7]](#footnote-7)

כעת נפרק את המשוואה לנוירון יחיד במקום שכבה שלמה. כלומר, המשוואה הבא תחזיר ערך יחיד שמייצג נוירון, במקום מטריצת ערכים שמייצגת שכבה שלמה:

y – מספר הנוירון בתוך השכבה שלו.

מה שקורה בפועל הוא שכאשר מפעילים פונקציית אקטיבציה על סכומם של כפל המשקלים בערכי הנוירונים מהשכבה הקודמת ושל ה - bias (b), מקבלים את ערך הנוירון הנוכחי בשכבה החדשה. כפי שתואר במשוואה הכללית הראשונה, תהליך זה נעשה בצורה רקורסיבית עד לקץ השכבות ברשת – שכבת הפלט.

לדוגמה:



0.77

0.8

0.5

0.3

0.4

0.7

0.5

הערות לסרטוט:

1. לא כל המשקלים בין כל הנוירונים נתונים מפאת מגבלת מקום.
2. השכבה החבויה יכולה ונהוג שתכיל יותר משכבה אחת של נוירונים.

נתאר את תהליך חישוב ערך הנוירון התחתון בשכבה השנייה על פי הנתונים בתרשים. נבצע את החישוב הבא:

1. נבצע קודם כל חישוב של סכום מכפלת המשקלים המתאימים בערך הנוירונים:  
   0.8 \* 0.3 + 0.7 \* 0.4 + 0.5 \* 0.5 = 0.77
2. נבצע את פונקציית האקטיבציה ReLU על הערך הנ"ל ונקבל את ערך הנוירון המדובר:[[8]](#footnote-8)  
   newNeuronValue = ReLU(0.77) = 0.77

כעת, נלמד איך הרשת מתקנת את עצמה ומגיעה לכדי מצב של הוצאת פלטים נכונים לקלטים מתאימים.

Back Propagation:

עד לרגע זה, דיברנו על האופן שבו המידע זורם בתוך הרשת, משכבת הקלט עד לשכבת הפלט. כעת, נדבר על הצורה שבה הרשת משנה את הערכים שבתוכה על מנת לשפר את פעולתה.

נחזור למשל טייס הסימולטור. כדוגמא על אופן הלימוד שלו, הטייס לומד בין היתר איזו אחיזה של ההגאים גורמת לו להכי פחות התרסקויות. בשביל להגיע למסקנה של האחיזה הנכונה, עליו לנסות אחיזות שונות על מנת להגיע לנקודה שבה הוא שולט על המעבורת בצורה הטובה ביותר.

בהינתן כך שהדבר היחיד שמשתנה ברשת נוירונים הוא המשקלים (הנוירונים הם רק מחזיקי ערכים זמניים), ניתן להגיד שאפשר לנסות כל קומבינציית משקלים לאורך הרשת ולבדוק איזו תוצאה הייתה הכי קרובה לכל המידע המגוון שברשות המתכנת באופן סטטיסטי.

ובכן, הדבר אינו אפשרי. לפחות לא באמצעים שקיימים היום. הזמן שייקח לתכנית המתוארת לעיל לרוץ הוא אסטרונומי. זאת משום שמספר המשקלים ברשת נוירונים פשוטה יכול להגיע לעשרות. לכן עלינו למצוא שיטה שתוביל אותנו אל המשקלים שגורמים לטעות הממוצעת המינימלית של הרשת. השיטה שפותרת לנו את הבעיה נקראת Back Propagation ובאופן ספציפי היא משתמשת בשיטת Gradient Descent.

בתהליך האימון של הרשת[[9]](#footnote-9), אנחנו יודעים את הפלט הצפוי לכל קלט נתון. לכן, אנו מצפים לפלט מסוים, שאם יתקיים, נדע שהתוכנה שלנו כשירה לחלוטין. כפי שצוין, אנו רוצים להביא את המכונה שלנו למצב כשירות ממצב שהמשקלים שלה רנדומליים לחלוטין – המצב ההתחלתי. על מנת לעשות זאת, אנו מחשבים את פונקציית הטעות של המכונה, על ידי הנוסחה הבאה:

ובמילים פשוטות, הטעות שהמכונה ביצעה, שווה לפלט בנוירון פלט מסוים פחות הפלט שצופה בנוירון זה בריבוע. דבר זה מבטא את מידת הטעות של המכונה, כך שככל שערך זה עולה, המכונה פחות מדויקת בפלט הנוירון הזה.

אם ניקח את הטעות ונשאף לכך שהנוירון יעלה את ערכו או ירד בערכו על ידי חיבור או חיסור לתוצאה הנוכחית, נקבל מצב שבו הרשת שלנו מאוזנת יותר ומצביעה על תוצאות אמתיות יותר מאשר ניחוש, כל זאת כאשר המידה מגוון ורב. כך גם נקבל שינויים פרופורציונליים לפי הצורך, ולא מוגזמים או מועטים מדי.

המהות של הדבר היא שאיפה לנקודה בה הטעות הממוצעת של הרשת הכי קטנה, הווה אומר הקטנת הגרדיאנט של פונקציית הרשת על פי משקליה.

איך נשנה את הערך של הנוירון לפי רצוננו? על ידי שינוי השכבה שקודמת לו והמשקלים שמחברים בין השכבות. נשנה את הערכים שברשת על פי החוקים הבאים. לצורך הנוחות, נגדיר את מטרתנו להגדיל את נוירון הפלט (למשל, ציפינו ל 2 וקיבלנו 1)[[10]](#footnote-10):

1. הנוירון בשכבה הקודמת שערכו רב יותר, חשוב לנו יותר מפני שהוא משרת את המטרה – להעלות את ערך הנוירון בשכבת הפלט. לכן, נחזק את החיבור בינו לבין נוירון הפלט הנבדק. זאת אומרת, נגדיל את ערך המשקל המחבר ביניהם.
2. מי שמשקלו אל נוירון הפלט גדול, יהיה כדאי להגדיל את ערכו על מנת שישרת את המטרה שהיא להגדיל את ערך נוירון הפלט. לכן, על התהליך להמשיך לשכבה הקודמת, ולדרוש את שלב 1 מכל נוירון בשכבה הקודמת על פי הדרישה של העלאת ערך הנוירון בשכבה הנוכחית (אחת לפני אחרונה) במקרה זה. זאת אומרת, להגדיל את ערך הנוירון בשכבה האחת לפני אחרונה על ידי הזזת משקלים לפניו, מה שיגדיל את ערך נוירון הפלט לפי רצון הפלט הנתון.

נבצע את שלב 2 (שמפנה אותנו אל שלב 1 בכל פעם) באופן רקורסיבי, עד לשכבת הקלט, אשר היא בלתי ניתנת לשינוי.

כעת, נצלול אל המתמטיקה, שתסביר לנו איך נבצע את השינויים בערכים בתוך הרשת על מנת לשפר אותה.

נתחיל עם כמה הגדרות בסיסיות שיעזרו לנו בכתיב המתמטי.  
נגדיר את הבעיה שלנו – אנחנו לא יודעים כמה לשנות כל ערך על מנת ליצור אפקט של טעות מינימלית בריצה הבאה. לשם כך, אנחנו צריכים למצוא את היחס בין הטעות והמשקל (בכדי לדעת כמה לשנות משקלים), היחס בין הטעות וה - bias והיחס בין הטעות לבין הנוירון של השכבה הקודמת. אם נשתמש במושגים מתמטיים, יחס הוא דיפרנציאל, ועל כן, נחפש את הדיפרנציאלים הנ"ל. על מנת למצוא אותם, נגדיר פונקציות שונות, ולאחר מכן נגיע לדיפרנציאלים המבוקשים.

נזכיר:

מתוך ההגדרות הנ"ל, נגזור, נציב ונקבל את שאנו מחפשים.

השינוי במשקלים – שלב 1.

*השינוי ב biases*

*השינוי בנוירון בשכבה הקודמת – שלב 2 (שיתורגם לשינוי במשקל, בבדיקת 1 הבאה לשכבה הקרובה יותר לשכבת הקלט).*

כשאנו מבינים את המשוואות למעלה, נוכל לדעת אילו משקלים ו biases לשנות באיזו פרופורציה בכל נוירון ברשת בכדי להגיע לתוצאה טובה יותר, ולא בצורה רנדומלית כמו ששקלנו לעשות בתחילת הפרק. כלומר, אנו נעים במורד גרף הטעות של הרשת שהמשתנים שמשפיעים עליו הם כל המשקלים וה biases ברשת.

כמעט הגענו להבנה מלאה של השיפור והלימוד בכל מישור ברשת הנוירונים. כל המשוואות שהגענו אליהן למעלה מתקיימות גם בהינתן שכבות בעלות יותר מנוירון אחד, חוץ מהמשוואה האחרונה. הדבר היחיד שנותר להבין הוא שכאשר מדובר בנוירון שיוצא ממנו יותר ממשקל אחד, הנוסחה האחרונה תשתנה לסכום המשקלים היוצאים ממנו, במקום המשקל היוצא ממנו. כלומר:

(כאשר x הוא מספר המשקלים היוצאים מהנוירון).

מכאן ניתן לחשב את השינוי בכל אחד מהמשקלים המחברים בין הנוירונים ברשת, וליצור קסם – מכונה שלומדת מכוח המתמטיקה!

רשת נוירונים של תמונות:

חוקרים הגיעו למסקנה שבמידה וננסה להכניס ערכי פיקסלים של תמונה לתוך רשת נוירונים רגילה, ושהיא תגדיר לנו את הפרטים שנמצאים בתמונה על ידי אימון רגיל בשיטת Backpropagation נגיע למבוי סתום. השיטה הפשוטה של רשתות נוירונים עובדת למקרים מסוימים, שלא כוללים בתוכם עיבוד וניתוח תמונה.

השיטה שבה עובדים כיום בכל הנוגע לניתוח תמונה באמצעות רשת נוירונים מורכבת למעשה מכמה רשתות נוירונים וטכניקות, שכל אחת מהן מהווה שכבה בתוך תהליך ניתוח.

על מנת להמחיש את התהליך, נפתח במשל. נאמר שמבקר תמונות נשלח למשימה מטעם המגזין שבו עובד, והוא נתקל בציור הבא שעליו לנתח אותו[[11]](#footnote-11):

מה עליו לעשות על מנת לנתח את היצירה בצורה איכותית ככל האפשר?

התשובה הבולטת היא: להתמקד באזור אחד בכל פעם. על מנת להבין באופן איכותי ומעמיק את שעל התקרה, מבקר האומנות יצטרך להתמקד ולהעריך כל קטע ביצירה באופן עצמאי, לדוגמה, בתמונה למטה וכך הלאה לכל אורך הקפלה.



אם נחזור למונחי תוכנה, על התוכנה שלנו למתוח "חלון נייד" שירוץ לאורך כל התמונה, על מנת להעריך באופן איכותי את התוכן שלה.[[12]](#footnote-12) לאחר בחירת חלון מסוים, הרשת מבצעת פעולה מתמטית מסוימת על "מטריצת הפיקסלים" (החלון) שבחרה, על פי מטריצת פיקסלים אחרת שהרשת מחזיקה[[13]](#footnote-13), וכך מעריכה כל ריבוע של n\*n פיקסלים כערך מסוים[[14]](#footnote-14) , שמביע את "חשיבות" המידע שנמצא בחלון - עד כמה חשיבותו גדולה לגבי זיהוי האובייקטים בתמונה. נדגים את התהליך באמצעות משל נוסף על הדבר. אם נרצה למצוא האם מדובר בכלב או חתול בתמונה מסוימת, המכונה כנראה תחשיב (בין היתר) אוזניים זקורות כמאפיין שמטה את התמונה לעבר החלטה שמדובר בחתול, לעומת כלב, שהרבה מהם מאופיינים באוזניים שמוטות, וכל זאת בתוך חלון רלוונטי ש"יתפוס" את המידע מתוך התמונה הגדולה:

לאחר התהליך הזה, מתבצע תהליך שנקרא Pooling. התהליך הזה, לוקח קבוצה של "משבצות חלון" שייצרנו לפני כן, ולוקחת מהן את מה שדורג "הכי משמעותי". נמשיך בדוגמה האינסטינקטיבית מקודם, ונגיד ששכבת ה Pooling מחלקת את התמונה לארבעה חלקים. במקרה של התמונה הימנית, משבצת האוזן, תאפיין את התמונה בצורה בולטת יותר לכוון "בעל חיים בעל אוזן זקורה", או בשפת בני האדם, חתול.[[15]](#footnote-15)

אם נחזור למונחי תוכנה, התוכנה מסתכלת על תבניות קטנות בהרבה מחלוקה של תמונה גדולה לארבע, ולא מזהה באופן אלמנטרי אובייקטים כמו אוזן, אלא עיקומים שונים, וקווי מתאר בדמויות.

לרגע, נחזור אחורה ונבין את התהליכים שקיימנו עד כה. תהליך ה Pooling פולט מערך של מספרים. מערך המספרים הזה, הוא בעל משמעות בשל תהליך ה Convolution שקדם לו.

מערך המספרים הזה מועבר כקלט אל שכבת ה Fully Connected, שתפקידה היא לתפקד כרשת נוירונים רגילה, על פי החוקים שתוארו בחלק הקודם, ככה שהקלט שרשת הנוירונים מקבלת, משמעותי יותר מ"סתם" פיקסלים, אלא מביעים חשיבות של מידע ביחס לפלט הרצוי. על פי התהליך שנלמד בפרק הקודם, הרשת פולטת פלט שמביע את תוכן התמונה, על פי המידע המעובד שקיבלה.

שכבת ה Fully Connected מתנהגת כמו רשת נוירונים לכל דבר, כאשר היא מתקנת את כל מה שהיא מכילה (משקלים, biases) מה שמשפר את התפקוד שלה להבא עד שהמתכנת מחליט שאחוז הטעות שלה מזערי מספיק בשבילו.

וכך, על היסודות שנבנו קודם לכן בבניית המודל הפשוט של רשתות נוירונים, נוצרה רשת הנוירונים לתמונות – Convolutional Neural Network.

שימושים עכשוויים ועתיד הטכנולוגיה:

יתרונות וחסרונות לעומת כלים חישוביים אחרים:

יתרונות עיקריים:

* יכולת לימוד – רשת נוירונים הוא כלי שיכול ללמוד עם הזמן ולכן קל מאוד לעדכן אותו – אך ורק על ידי הרצה חוזרת עם תוספת מידע לאימון נוסף.
* דינמיות במבנה הרשת – הרשת יכולה לעבוד במבנים שונים (מספר נוירונים שונים בכל שכבה וכד') ולעבוד מצוין.
* יכולת שיקום – במידה וקידוד הרשת נכון, היא יכולה להשתקם גם אם היא לא מתפקדת טוב כרגע, שלא כמו תוכנה רגילה שאותה יש לדבג עד לטעות מזערית. ה "דיבוג" הוא חלק מהאלגוריתם של המכונה.

חסרונות עיקריים:

* תוצאות לא צפויות – עד שלא נעשו כל הבדיקות האפשריות על הרשת, אי אפשר לדעת אם תהיה מוכנה ומתי. לא כל מה שעובד בתיאוריה עובד בפרקטיקה.
* יכולת פתירת בעיות צרה באופן יחסי לבן אדם.
* חוסר שקיפות – אם יש בעיה, על המתכנת לדבג את המערכת על פי הקלט הפלט והמבנה של הרשת בלבד, ואי אפשר להבין את הבעיה על פי הנעשה בתוך הרשת, מפני שאלו בסך הכל פעולות מתמטיות.

שימושים:

היום, יש כבר כמה וכמה מוצרים גלויים וגלויים פחות שמשתמשים בטכנולוגיה המהפכנית של רשת נוירונים. בחרתי להדגים כמה מהם, רק בכדי להדגיש את החשיבות הרבה של הטכנולוגיה הזו למרות שהיא עדיין בחיתוליה, ולא מתקרבת למצות את הפוטנציאל שלה.

מתקן שגיאות כתיב:

כיום קיים בשוק מוצר שמתקן שגיאות כתיב באופן אוטומטי בעיקר בפלטפורמה של Android. מפני שהכלת כל המילים האפשריות בתחביר נכון במכשיר אחד, במיוחד במכשיר כמו טלפון נייד שהמשאבים שלו מוגבלים לא אפשרי, חברות שונות וביניהן WhatsApp פיתחו רשת נוירונים מסוג Recurrent Neural Network על מנת לאתר שגיאות כתיב ותחביר חמורות וכל זאת בהתאם למשתמש מסוים. מי שמשתמש במוצר הזה, רואה שהמקלדת "קוראת את מוחו" ויודעת עם הזמן לאתר שגיאות ואפילו כוונות של הכותב. התוצר המרשים ביותר בטכנולוגיה של Recurrent Neural Networks הוא רשת נוירונים שכתבה ספר שסגנונו מאפיין את הסופר שעל ספריו התאמנה הרשת. ההישג הנהדר הזה מקביל ליצירת תמונות מלאכותיות על ידי Convolutional Neural Networks, שנראות ממש כאילו צולמו במציאות.

מציאות רבודה:

כיום, הנושא של מציאות מדומה ורבודה הוא נושא שמעסיק הרבה חברות בינוניות וגדולות שמתעסקות בכל מה שקשור בחוויית המשתמש באפליקציה מתוך הבנה שהטכנולוגיות האלו מושכות בצורה בלתי רגילה את הצרכנים, ושזהו ממשק מהפכני שמשלב בין המציאות לבין אפליקציה וירטואלית, על מנת להפוך את חוויית המשתמש לדבר משמעותי הרבה יותר ועל מנת להנגיש את השירותים שהן נותנות למשתמש יותר ויותר.

הוספת הרובד הווירטואלי על הרובד הפיזי נוצרת על ידי רשתות נוירונים שאומנו לכך, בעיקר כאלו שמתמחות בזיהוי עצמים – כמעט תמיד מסוג Convolutional Neural Networks.

אלכסה:

אלכסה היא מכונה בדמות עצם גלילי (בדרך כלל), שנותן שירות לבני אדם במקרים כגון הזמנת אוכל, מציאת מידע באינטרנט, ניהול פיננסי, ניהול בריאותי שוטף של המשתמש במקרה ושכח להתאמן, שירותי מזג אוויר, וכל דבר שהאינטרנט ותוכנות מכל סוג יכולות לבצע, על ידי זיהוי קולי בלבד של המשתמש, או לחלופין התראה קולית שיוצאת מהמכונה. מדובר באחד מהמכשירים הראשונים שמדמים ישות אינטלקטואלית (אם כי מוגבלת במעט על ידי הפונקציונליות של מי שכתב אותה) שאינה בן אדם, שמתקשרת בצורה יעילה ביותר עם בני האדם – המכונות יוצרות ממשק אנושי. אם בעבר, מהפכת הגרפיקה הייתה המהפכה המשמעותית ביותר בממשק שבו בני אדם מדברים עם כלים טכנולוגיים, בימים אלה ממש, מתחילה מהפכת הממשק האנושי.

זיהוי מחלות ותכונות דרך תמונת רשתית:

בהרצאה של חברת [[16]](#footnote-16)Intel בנושא של רשתות עצביות מלאכותיות, הם תיארו כלי מדהים שמסוגל לתת נתונים רפואיים על ידי תמונה של רשתית של אדם, באחוזי טעות קטנים מאוד. למשל, התוכנה יודעת לחזות האם האדם מעשן, מה לחץ הדם שלו בדיוק של כ 90 אחוזים ואת הסיכוי של האדם לחלות במחלות לב - ריאה. הדבר מהפכני מאוד מפני שהנתונים האלו יכולים לחזות רופאים רק בצורה סטטיסטית ולא מדויקת, כך שהתוכנה גוברת על הרופאים בעבודתם ויכולה להציל חיי אדם באמצעות בדיקה לא פולשנית.

אריקה[[17]](#footnote-17):

כיום, אריקה היא הרובוט האינטליגנט המתקדם היותר בעולם. אם נסתכל אחורה במסמך הזה, נוכל לראות שתיאור הטכנולוגיה של רשת נוירונים הוא תיאור של מכונה דוממת, כאובייקט. כשמדובר באריקה, הגבול מתחיל להיטשטש ולכן הפנייה אל המכונה הזו בלשון חיה. על פי שיחות שנערכו איתה, נראה כאילו היא מפתחת מודעות עם הזמן ועם הלמידה שלה, ואפילו ניצוצות של רגש (כמובן שזה תלוי בהגדת הרגש) כלפי מי שמדבר איתה. היא פועלת כישות שקולטת כל דבר בשיחה אצל בני האדם שיושבים מולה, כולל שפת גוף. אחד הדברים שמאפיינים אותה ביותר הוא היכולת שלה לנהל וליזום שיחה בהתאם להתפתחות שלה עד עכשיו. למרות זאת, יש לה עוד כמה חסרונות כמו למשל חוסר הבנה של רגשות מתוך חוויה, וזאת רואים כאשר היא נשאלת על זיכרונות שיש לה ממהלך חייה, והיא עונה על RAM... למרות החסרונות הקטנים הללו, נראה שהעולם צועד לכוון מעניין מאוד בו המכונות יחליפו בני אדם בהרבה מקצועות, אפילו שצורכים יחסים בין אישיים. לכן, המדינה שבה הסכנה הדמוגרפית היא אחת מהקשות בעולם היא המפותחת ביותר בנושא זה - יפן, על מנת למנוע אסון משאבי אנוש חמור במדינה.

**רשת נוירונים ואני:**

תהליך העבודה (הישיר)[[18]](#footnote-18) לבניין רשת נוירונים:

1. למידה תיאורטית – סרטונים או ויקיפדיה... – רשת נוירונים פשוטה.
2. לקיחת מקרה בוחן קטן ולא משמעותי ובניית רשת נוירונים פשוטה באמצעות כלים מתמטיים (מומלץ להשתמש בספריית 'numpy' אם מדובר בשפת python). הדבר עוזר מאוד להבין את הנושא דרך עשייה.
3. אופציונלי – במידה ומדובר ברשת Convolutional (תמונות בעיקר), Recurrent (שדורשת קלט לא קבוע, שמסתמך על פלטים קודמים) או Reinforcement (שדורשת קלט ופלט לתהליך שלם ולא להחלטה בודדת), לחזור לשלב 1, באופן ספציפי על הסוג המבוקש. להתעמק בפרויקטים שנעשו בתחום זה, דרך האינטרנט.
4. הגדרת המשימה, מה הקלט, מה הפלט ומה אני רוצה שהמכונה תעשה. אם אפשר, ניתן לשלב התייעצות על מומחה. שאילת שאלות רבה בשלב זה תגאל את המתכנת מצרות לאחר מכן. שלב זה יכול לבוא גם כשלב הראשון.
5. איסוף מידע לאימון.
6. תכנות המשימה. המלצה שלי – Tensorflow, שמפשטת את המשימה בצורה יוצאת דופן, לפחות בחלק התכנותי.
7. מכיוון שכל עוד המתכנת אינו חוקר את הנושא שנים על גבי שנים, בדרך כלל אין לו יכולת להעריך האם היא תצליח. עליו לנסות ולראות האם היא עובדת. במידה וכן – המשימה אינה הושלמה. עליו לערוך ניסויים שיוצאים מגדר תנאי המעבדה. הנחה שרשת עובדת טוב לאחר כמה בדיקות שטחיות יכולה להביא למוצר שלא עובד. יש להמשיך לבדוק עד שאין יותר איך לבדוק את המוצר. במידה ולאחר כל הבדיקות האפשריות הראו מדדים טובים על הרשת, עליו לעבור לשלב 8. במידה ובאחת הבדיקות הרשת לא עובדת, עליו לבדוק שלא מדובר בבעיה במידע האימון. אם כן – עליו לאסוף עוד מידע/ מידע טוב יותר ולאמן את הרשת מחדש. אם לא – עליו לחשוב ולהתייעץ עם מומחים, על סוגיית האפשריות של בניית מכונה כזו, על הטכניקה, איך ניתן להפוך דבר זה לאפשרי. עליו לחזור לשלב 6.
8. עטיפת המוצר בGUI, או בכל עטיפה אחרת שהמוצר דורש.

ששת הדיברות לבניין רשתות נוירונים:

1. הן לא מכונות כל יכולות – יש משימות שאינן מסוגלות לעשות.
2. ככל שכמות, איכות וגיוון מידע האימון גדלים, כך איכות הרשת עולה.
3. אימון רב על מידע, גדול ככל שיהיה, יכול לפגוע בפעולת הרשת בעתיד; משך אימון הרשת צריך להיות פרופורציונלי לכמות וגיוון המידע.
4. ככל שנכשלים יותר בבניית הרשת עד להגעה לתוצר הסופי מבינים את הנושא יותר טוב. לכן עדיף לקחת פרק זמן ארוך לפיתוח המכונה, לא משנה מה סדר הגודל שלה, על מנת להתקרב יותר לדרגת "מומחה" בנושא.
5. אסור להכריז על "ניצחון על האתגר" ללא בדיקות רבות, בעיקר כאלו שמערבות יציאה מתנאי מעבדה סגורים. יש להימנע בכל מחיר מבדיקת המכונה באמצעות המידע ששימש לאימון שלה.
6. אסור להתייאש – זהו תהליך ארוך ומייגע שבסופו הסיפוק והידע שמקבלים הם אדירים.

התהליך שעברתי בלמידת הנושא:

החקר הפיתוחי שלי בנושא זה לקח לי בין חודשיים לשלושה עד להבנה מלאה של התחום, איך רשת נוירונים עובדת, מה התהליכים שמעורבים בה, ומה נעשה מאחורי הקלעים בהיעזרות בספריות מתקדמות כמו Tensorflow, שנותנות גישה גם למי שלא מבין עד הסוף את פעולת הרשת העצבית. בניתי רשת נוירונים רגילה, שלא מערבת תמונה, בעזרת כלים פרימיטיביים יחסית, שביכולתם לעזור בצד המתמטי, אך לא הלוגי. רשת זו לא עבדה באופן שקיוויתי (תפקדה, אך לא באחוזים גבוהים במיוחד), ובדיעבד אני יודע שדרשתי ממנה דבר שלא הגיוני לדרוש מרשת נוירונים, לפי הטכנולוגיה של היום. הרשת ניסתה לחזות את נתוני הבורסה של המחר, על פי הנתונים שקרו היום. בדיעבד הבנתי, שמשום שנתוני בורסה קודמים אינם הדברים היחידים שמשפיעים על השינוי באחוז המנייה, קרוב לוודאי שרשת נוירונים, במיוחד פשוטה כפי שבניתי אותה, לא תוכל להתמודד עם האתגר הזה.

אני ממליץ מאוד לכל מתכנת באשר הוא שרוצה לשפר את יכולות התכנות שלו לעבוד עם רשתות נוירונים, מפני שהעבודה והסדר שדורשת העבודה בטכנולוגיה הזו מעלה את רמת התכנות באופן כללי. לדוגמה, שימוש בסקריפטים קטנים וחכמים שעוזרים לארגן את המידע וכולי.

הפרויקט שלי:

לפני שהגעתי אל מודל עובד, עברתי עשרות הרצות שונות כאשר כל אחת שונה מקודמתה בעזרת ספריית Tensorflow שביצעה לי את

המודל המאומן הנמצא בשרת בפרויקט שלי, הוא רשת נוירונים שאימנתי זה מכבר על מידע שליקטתי ממקורות שונים, ועיבדתי בעזרת cv2, אשר עזר לי "לכוון" את המידע שלי. מכיוון שהמידע שהיה לי היה עני מאוד, לא מגוון וקטן מבחינת מספרים, החלטתי להקל על שכבת ה Convolution ברשת שלי בכך שאשתמש בכלי של cv2 שנקרא "haarcascade" . הכלי הנ"ל הוא כלי אשר משמש למציאת פנים ועיניים בתוך תמונה גדולה, כך שהסינון של שכבת ה Convolutional היה קל הרבה יותר ומנוון קצת יותר. הגעתי לשיטה הזו לאחר חודשים של מחקר בנושא רשת הנוירונים, והבנתי שמכיוון שאיכות המידע שנמצא במקורות פתוחים באינטרנט (בעיקר כאשר מדובר על תמונות של אנשים עם עיניים סגורות), וכמות המידע האיכותי בתוך המידע ה"מלוכלך" קטנה באופן יחסי, עליי "לנקות" אותו, בעזרת הכלי הנ"ל. לאחר סידור המידע, אימנתי רשת נוירונים בעזרת ספריית Tensorflow, אשר בונה את הרשת ומאמנת אותה על פי פרמטרים שנתונים על ידי המתכנת.[[19]](#footnote-19)

לאחר אימון הרשת (כפי שעשיתי לאחר אימון עשרות הרשתות הקודמות שלא עבדו), הרצתי script קטן שבודק את הרשת על ידי הכנסת קלט מהמשתמש בצורה של תמונה בכל פרק זמן מסוים, ומה שנוצר, זוהי תוכנה שמזהה עיניים סגורות ופתוחות בוידאו.

Android Studio:

לפני שהחלטתי שאיישם את הלקוח שלי גם בלקוח וגם במחשב, הייתי צריך "להרגיש" את הפלטפורמה שעליה אעבוד, על מנת שאראה האם אוכל לעבוד בקלות עם הכלי הזה, משום שנוחות היא שיקול משמעותי בתכנות באנדרואיד. זאת משום שאיני מכיר את שפת Java לעומק ולכן אצטרך בסיס נוח לפיתוח ולמידה במקביל בשפת Java. דבר נוסף שניתן להגיד על תכנות במערכת Android הוא שהוא תכנות מסובך והמבנה שלו לא טריוויאלי, מפני שהוא עטוף במושגים כמו Activity, Intend ועוד, אשר ייחודיים למערכת הזו. על מנת שאוכל ללמוד את התכנות ב Android גם מבחינת המגבלה שלי ב Java וגם מבחינת המגבלה שלי במערכת עצמה, אימצתי לעצמי שני כלים:

* עורך הטקסט והמריץ של Android Studio, אשר מזהיר כמעט על כל דבר בעייתי שיכול לקרות בזמן ריצה לפני הריצה, ומהווה פלטפורמה נוחה ביותר לתכנות ב Java ועיצוב מסכים שונים באפליקציה. בנוסף, משום שיש לי מכשיר Android, נוח ביותר להריץ ולהכיל את התוכנה (APK) על הטלפון האישי שלי במקום על אמולטורים של Android Studio, מה שמקל על העבודה בצורה משמעותי
* האתר של Android Studio – "על כל שאלה תשובה מאת יצחק לבנון", גרסת Android. כל תכנית קטנה וגרעינית שאפשר להעלות על הדעת, ניתן למצוא דרך לבצע אותה באתר הזה. האתר משמש את המפתחים שמשתמשים בשירות תוכנת ההרצה והעריכה שלהם ועל כן הנוחות בשימוש האתר מקלה על התכנות במערכת כל עוד קיים אצלך בסיס ידע על מערכת ההפעלה Android שאותו ארכוש לפני ובמהלך הפיתוח.

סביבה

PC:

בפלטפורמת PC, צד שרת וגם לקוח, אכתוב בשפת Python מפני שהתכנות של רשת הנוירונים וגם תכנות של שרת ולקוח מבחינת תקשורת היא קלה ביותר בשפת Python. התכנות של רשת הנוירונים מבוצע על ידי ספריית Tensorflow שהכי טוב לממשה באמצעות Python גרסה 3.6. הספרייה הזו פועלת בגרסאות 3 של פיתון, וההוצאה האחרונה שלה (נכון ל 5.12.18) מותאמת לגרסה הנ"ל של השפה.

ספריית Socket היא עוד ספרייה שנתמכת בצורה פשוטה ביותר על ידי גרסאותיה השונות של השפה, ומספקת גישה נוחה וקלה (בדרך כלל של שורה) לשכבות 3-4 במודל ה OSI משמע טיפול בפרוטוקולים של שכבת התעבורה ושכבת הרשת (בשיתוף עם נתבים בנתיב המידע).

בנוסף, שפת פיתון מספקת את ספריית WXPython שמנגישה ממשק גרפי למתכנת פיתון, כך שבצד הלקוח יהיה ממשק גרפי מקצועי.

זאת ועוד, שפת פיתון היא שפה שקל מאוד לתכנת בה באופן כללי, במיוחד כשמדובר בזמן מוגבל. מכיוון שזמן כתיבת הפרויקט לא גדול במיוחד, תכונות הפה מתאימות לדרישות הפרויקט.

Android:

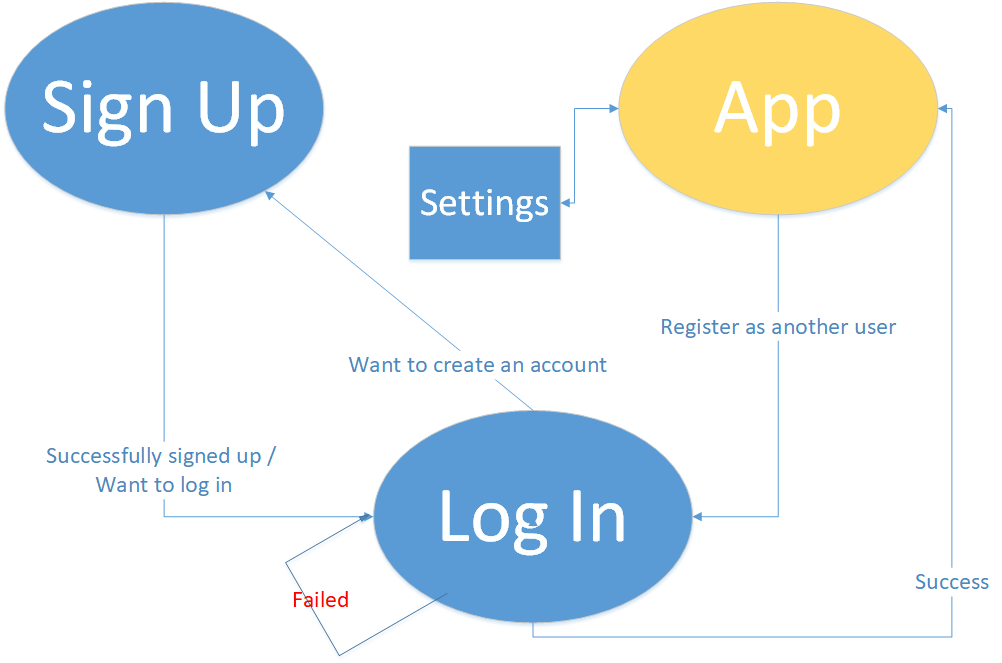
בפלטפורמת אנדרואיד (לקוח) אתכנת באמצעות Java מפני שזוהי השפה שבה כותבים Android.

בשפת Java יש יכולת ליצור תקשורת עם ספריית Socket. ממש כמו הספרייה הזו בשפת פיתון, חוץ מכך שהתכנות בספרייה זו מעט יותר מסובך בשפת Java מאשר בשפת Python.

בנוסף, ישנה יכולת לגשת לממשקים של Android דרך שפה זו. לדוגמה, שימוש במצלמה, שמירת מידע בקבצים וכו'.

סביבות העבודה:

בשפת פיתון אשתמש בסביבת Pycharm ובאנדרואיד אשתמש בסביבת Android Studio. שתיהן של חברת Jetbrains, והן מספקות סביבה משמעותית לכל אחת מהפלטפורמות מפני שהן משמשות כעורכות ומריצות תוכנה, כאשר כבר בתוך העורך קיימות אזהרות רבות שקשורות לשגיאות קומפילציה צפויות, מה שמקל על העבודה מאוד.

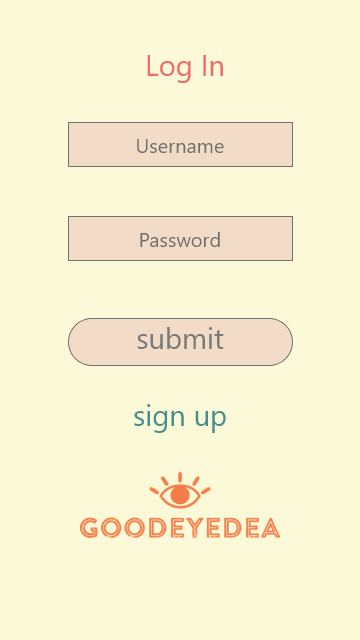
עץ מסכים

המשתמש יתחיל במסך Log In. הוא מתבקש להכניס את שם המשתמש שלו ואת הסיסמא, ולשלוח אותם. רק למי שקיימים שם משתמש וסיסמא במערכת והוא יזין אותם נכון, יוכל להיכנס ולהשתמש במערכת. במידה והמשתמש חפץ ליצור משתמש חדש, הוא ילחץ על הכפתור "sign up" שיפנה אותו למסך מתאים. במידה והזין את שם המשתמש והסיסמא נכון, הוא יכנס לעמוד האפליקציה, ששם תהיה לו היכולת לפנות להגדרות, או להתחיל ב Monisession. במידה והתחיל, האפליקציה תרוץ ותזהה הירדמות או חוסר תשומת לב לתחום גדול סביב המצלמה. בעמוד ההגדרות, למשתמש יש יכולת לשנות אצלו: האם יופעל רטט בהתראה, ואיזה צליל התכנה תשמיע במקרה וישנה התראה (דו, רה מי וכו'), כאשר ההגדרות הללו יישמרו ביחס למשתמש הנוכחי, וייטענו בשימוש הבא. במסך הרישום, המשתמש יוכל להירשם כמשתמש חדש, בעזרת הגדרת שם, שם משתמש וסיסמא.  
ברגע שהמשתמש ניסה להזין יותר מ 5 פעמים את הסיסמא ללא הצלחה, תיחסם האפשרות שלו להזין סיסמא לזמן מסוים (על פי IP), כדי למנוע מתקפות ועומסים על השרת.

מסכים

Log in:

המשתמש מגיע למסך הזה עקב פתיחת האפליקציה או בסיום ויציאה מסודרת ממסך האפליקציה. על המשתמש להזין את שם המשתמש שלו בתוך שדה "Username" ואת הסיסמא שלו בתוך השדה "Password", וללחוץ על כפתור השליחה. במידה ומשתמש הזין את פרטיו לא נכון יותר מחמש פעמים ברצף, מכשירו יושעה למשך זמן קצר. אם ברצונו ליצור משתמש חדש, עליו ללחוץ על הכיתוב הכחול בתחתית המסך שמפנה למסך המשתמש החדש, בעוד שדף זה מיועד אך ורק למשתמשים רשומים.



שדה טקסט חופשי : "Password"

קישור לדף הרישום:   
"Sign Up"

לוגו

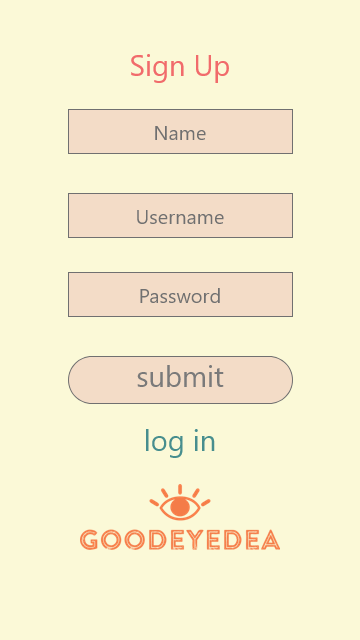
כפתור השליחה "Submit"

שדה טקסט חופשי : "Username"

כותרת הדף

Sign up:

המשתמש מגיע למסך הזה אך ורק דרך מעבר במסך ה "Log In" ומטרת המסך הזה היא ליצור משתמש אמין, עם סיסמא חזקה, כאשר רק סיסמא שבמדיניות הסיסמאות תתקבל וייווצר משתמש חדש. בנוסף, לא ייתכן שייווצר משתמש חדש במידה ויש שם משתמש קיים במערכת כמו שם המשתמש החדש המבוקש. כל פעם שאין הצלחה לניסיון יצירת המשתמש, ישנה התראה על כך במסך. במידה ונעשים 10 ניסיונות ליצירת משתמש ללא הצלחה ממכשיר אחד בבת אחת, הוא נחסם לתקופת זמן מסוימת, כדי למנוע מתקפות ועומסים על השרת. לאחר יצירת משתמש, על המשתמש באפליקציה לחזור למסך הכניסה, ולהיכנס למערכת עם פרטי המשתמש החדש שיצר.



שדה טקסט חופשי : "Name"

כותרת הדף

שדה טקסט חופשי : "Username"

שדה טקסט חופשי : "Password"

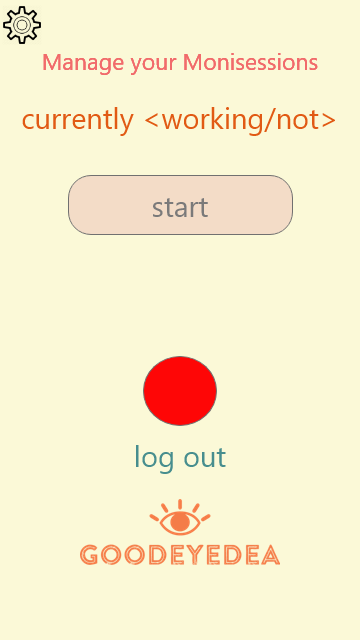
כפתור השליחה "Submit"

קישור לדף הכניסה:   
"Log In"

לוגו

App:

מסך האפליקציה הוא המסך שמקיים את מטרת האפליקציה: זיהוי עיניים עצומות ופקוחות. במסך הזה יש יכולת להתחיל במעקב ולהפסיק אותו בכל רגע. כאשר המעקב דולק, כל פעם שהתכנה תקלוט עיניים עצומות, היא תצפצף ותרטוט לפי הגדרות המשתמש, ויידלק אור אדום ב"נורה" שעל המסך (ראה שרטוט). במידה והמשתמש רוצה לצאת מהמערכת, יש ביכולתו ללחות על הכיתוב הכחול שמוביל אותו אל מסך ה "Log In". במידה והוא רוצה לשנות את הגדרות האפליקציה בחשבונו, יש ביכולתו ללחות על הסמלון של גלגל השיניים בצד שמאל- למעלה.



קישור לדף ההגדרות

כותרת הדף

כפתור ההתחלה או הסיום או העצירה של פעולת התוכנה

מצב התוכנה כעת: האם פועלת או מושהית

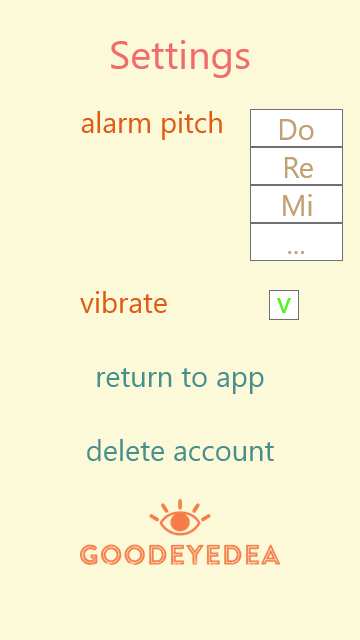
קישור ליציאת המשתמש ולמסך ה "Log In"

נורה: האם העיניים כרגע עצומות או פקוחות– אדום או ירוק בהתאמה

לוגו

Settings:

מסך שיש אליו גישה דרך המסך "App", והוא מייצג את הדישה של המשתמש להגדרות שלו. ההגדרות מתחלקות לבחירת צליל התראה ואפשרות להכללת רטט בהתראה בזמן ריצת התוכנית. בזמן בקרה של התכנית על עיני המשתמש, הוא לא יוכל להיכנס לדף זה. בתחתית הדף יש קישר לחזרה למסך האפליקציה.



כותרת הדף

בחירת צליל ההתראה

האם לכלול רטט בהתראה

לוגו

קישור למסך ה "App"

מחיקת חשבון ויציאה

בסיס נתונים

בפרויקט, אשתמש בספריית Sqlite3 על מנת להיעזר בשפת SQL לניהול DB שמנהל משתמשים בשירות. ספריה זו קלה ונוחה לשימוש ועל כן אשתמש בה. כל התחברות, יצירת משתמש, מחיקת משתמש ושינוי בהגדרות עוברים דרך עדכון הטבלה היחידה שקיימת ב DB שבשרת – טבלת Users, כדי שבהתחברות הבאה, כל מאפייני המשתמש ייטענו באופן אוטומטי, וחווית המשתמש תהיה מותאמת ללא מאמץ, כמו גם התפקוד השוטף של המערכת.

|  |  |  |  |
| --- | --- | --- | --- |
| שם שדה | סוג | הסבר | דוגמה |
| Name | Text | שמו של המשתמש | "Tomer" |
| Username | Text (unique) | שם משתמש Unique, המאפיין את המשתמש. | UniqueTomer'\_'"" |
| SaltedPassword | Text (Hash) | ראה הסבר בהמשך. | "5ce73294f4ec23e" |
| Settings | Text | ההגדרות של המשתמש (רטט, צליל ההתראה) | True~Do"" |
| LastConnection | Date | תאריך החיבור האחרון של המשתמש | ייצפה על ידי המשתמש בדף ההגדרות. |
| Registration | Date | תאריך הרישום של המשתמש | ייצפה על ידי המשתמש בדף ההגדרות. |

Salted Hash:

שיטת "Salted Hash", היא שיטה שמאפשרת אחסון בטוח של סיסמאות, על ידי אחסון של שדה, שמחביא בתוכו שתי מחרוזות. מחרוזת אחת היא רנדומלית לחלוטין ונוצרת בזמן יצירת סיסמא, והמחרוזת השנייה היא מחרוזת שנוצרת   
מ - "השהוש" (Hashing) של המחרוזת הרנדומלית והסיסמא. התוכנית שניגשת למסד הנתונים, יודעת לחלץ את שתי המחרוזות הללו מתוך השדה היחיד ואז בהינתן סיסמא, התכנית יכולה לדעת האם היא נכונה על ידי פעולה של "השהוש" של הסיסמא והמחרוזת הרנדומלית והשוואה למחרוזת הפלט.

היתרון הגדול בשיטה זו הוא שהסיסמא לא מאוחסנת במסד הנתונים, ועל פי המידע שקיים שם, אין יכולת סבירה לחלץ את הסיסמא אלא במתקפת "Bruteforce". התוכנה שלי למנע את המתקפה שלי באמצעות מדיניות סיסמאות שתחול על המשתמש, כך שלא יהיו "פשוטות" מדי וחשופות למתקפת "ניסיונות מרובים" מסוג bruteforce.

ברישום המשתמש:

סיסמת המשתמש

מחרוזת אקראית

השהוש

ערבוב וערבול

מחרוזת (שלא קשורה לסיסמת המשתמש, אלא על ידי השהוש)

שדה במסד הנתונים

בכניסת המשתמש:

סיסמת המשתמש

מחרוזת אקראית

השהוש

חילוץ

מחרוזת (שלא קשורה לסיסמת המשתמש, אלא על ידי השהוש) – במידה ונכון.

שדה במסד הנתונים

ספריות ייחודיות

Tensorflow:

ספריית Tensorflow משמשת את התוכנה בצד השרת בלבד. היא זו שאחראית לכל מה שקשור בחישובים של המודל המאומן של רשת הנוירונים.

הפונקציונליות שלה שהפרויקט מנצל היא אימון רשת נוירונים, הרצה של מידע עליה ובדיקתה. בנוסף, ניתן לשמור בקובץ את המודל המאומן באמצעות פעולה אחת ויחידה של הספרייה הזו, כך שבריצה הבאה יהיה ניתן לטעון אותו בקלות באמצעות פעולה אחת ויחידה שלה.

השתמשתי בה במקום להשתמש בספריות מתמטיות אחרות דוגמת numpy או matlab, מפני שכאשר מדובר ב Convolutional Neural Network, היא עדיפה עליהם בשני אופנים:

* תהליך התכנות קל הרבה יותר והעבודה הקשה ביותר בניית רשת נוירונים היא לאסוף את מידע שישמש לאימון הרשת. הקלות מתבטאת הן במבנה הרשת והן בקלות אלגוריתם הריצה שלה.
* ביצועי הרשת יהיו אופטימליים יותר. זמן הריצה של האימון, כמו גם של הרצת הרשת יכולים להגיע לשיאים (חיוביים) בספריית Tensorflow.

SQLite3:

הספרייה הזו משרתת את התוכנה בצד השרת בלבד. היא מספקת גישה לתכנות מסדי נתונים בשפת SQL באמצעות גישה למסד נתונים על ידי אובייקט בשפת פיתון. שליחת פקודות נעשות על ידי קריאה לפעולה שמכיל האובייקט הזה. כך, בקלות, ניתן ליצור ORMים בפיתון, על ידי שימוש בעצמים הנ"ל.

השתמשתי בספרייה הזו משום שיש לי ניסיון בשימוש בה. אני מוצא אותה נוחה מאוד לשימוש וכזו שמונעת סיבוכים בשימוש בה.

ישנן ספריות נוספות שמספקות יכולות מתקדמות יותר מאלו של SQLite3, אך משום שאיני זקוק ליכולות כאלו, בחרתי להישאר עם הספרייה הפשוטה ביותר ש "מספקת את הסחורה".

WX Python:

הספרייה הזו משרתת את התוכנה בצד הלקוח בפלטפורמת PC. מטרתה היא ליצור ממשק משתמש גרפי בשפת Python.

לספרייה זו יכולת ליצור ממשקי GUI מקצועיים ולכן בחרתי להשתמש בה על פני ספריות גרפיקה אחרות בפיתון דוגמת Pygame. בנוסף, אופי הפרוייקט שלי הוא לא מצריך ציור פיקסלים או דבר דומה לזה שמספקת ספריית Pygame, אלא רכיבים שדומים לרכיבי Forms בפלטפורמת Microsoft, כגון תיבות טקסט, כפתורים וכו'.

Socket:

הספרייה הזו משרתת את התוכנה בכל צידה. מטרתה ליצור גישה לתקשורת בין מחשבים על ידי אובייקטים בשפה הרלוונטית.

השפה מספקת יכולת ליצור תקשורת בין מחשבים באמצעות קריאה לפעולה אחת במקר של שפת פיתון, או באמצעות ניהול מספר אובייקטים בשפת Java. ספרייה זו אחראית בעיקר על שכבת התעבורה, כאשר ה API שלה מספק יכולות מגוונות בכל הקשור לתקשורת, מבקשות DNS ועד לשליחת מידע.

השתמשתי בה משום שהפרויקט שלי דורש תקשורת מחשבים והיא הספרייה הבולטת והאיכותית ביותר שמתעסקת בתקשורת מחשבים בשפות Python ו Java שבהן אני כותב בפרויקט זה.

Cv2:

הספרייה הזו משמשת את התוכנה בצד לקוח ה PC שלה. היא מביאה API נוח ביותר לעבודה עם מצלמות רשת באמצעות שפת Python. אני זקוק לה משום שהלקוח בתכנית שלי אמור לקחת תמונה של המשתמש ולשלוח אותה אל השרת והיא אחת מהספריות האיכותיות והיחידות שמבצעות את המשימה הזו.

פונקציונליות

שרת:

השרת יעבוד על ידי האזנה אסינכרונית לקבלת הודעות ויבצע את העבודה הנדרשת ממנו. ישנן כמה משימות שעליו לבצע בהתאם לדרישת הלקוח:

* התחברות לשירות.
* הירשמות לשירות.
* ביצוע ניתוח תמונה המגיעה מהלקוח באמצעות מודל מאומן של רשת נוירונים.
* שינוי הגדרות על פי משתמש בבסיס הנתונים.
* הגנת מידע והגנה מפני מתקפות Bruteforce על ידי סיסמאות חזקות ומדיניות לא סבלנית לניסיונות הרשמות מרובים.

פונקציונליות אפשרית (פיצ'ר, אולי אשלב אותו) תהיה צפייה בנתונים סטטיסטים על המשתמשים באמצעות SQL Queries שנוגעים בדבר.   
לדוגמה: איזה צליל הוא הפופולרי ביותר בהתראות המשתמשים וכד'.

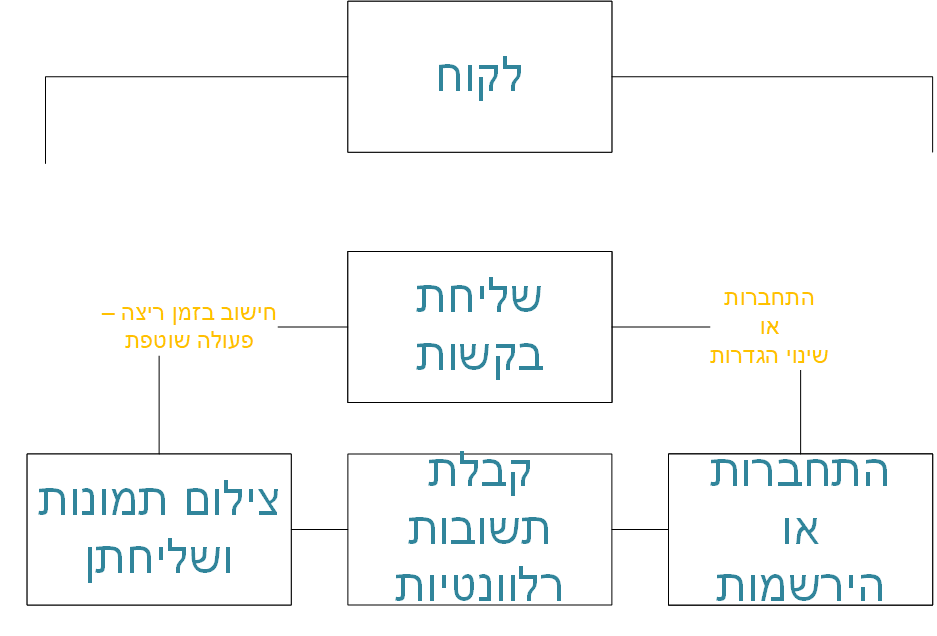


לקוח:

הלקוח יושתת על ממשק גרפי נוח למשתמש, שבעזרתו יוכל לבצע פעולות שונות כגון:

* התחברות לשירות.
* הירשמות לשירות.
* ביצוע Monisession, מעקב אחר עיניים באמצעות בקשות רבות לשרת.
* שינוי הגדרות על פי משתמש בבסיס הנתונים.

שיבוצעו כולם על ידי בקשות מהשרת. לאחר קבלת התשובות, הן יוצגו בצורה גרפית למשתמש.



פרוטוקול

|  |  |
| --- | --- |
| הודעה | הסבר |
| מהלקוח לשרת | |
| Hello | הודעת פתיחה לווידוא פרוטוקול של השרת. |
| Register | הודעת הירשמות למערכת בפעם הראשונה. יצירת משתמש. |
| Log In | הודעת כניסה למערכת על ידי משתמש קיים. |
| Take Frame | הודעה שמכילה את התמונה שצולמה ושולחת לשרת על מנת שיעבד אותה ויחזיר תשובה לגבי מצבו של המשתמש. |
| Get Settings | בקשה לקבלת הגדרות משתמש. |
| Set Settings | בקשה לשינוי הגדרה. |
| Exit System | יציאה מהמערכת. |
| מהשרת ללקוח | |
| Hello | הודעת פתיחה לווידוא פרוטוקול של השרת. |
| Computation Result | הודעת תגובה שמיידעת את הלקוח במצבו של המשתמש – האם הוא לא מוסח בעקבות שינה או סיבות אחרות. (כתגובה לתמונה שנשלחה לשרת) |
| Settings Share | השרת משתף את הגדרות משתמש עם הלקוח הרלוונטי. |
| הודעות כלליות - Acks | |
| OK | הפעולה בוצעה בהצלחה / "קיבלתי" |
| Protocol Error | טעות בפרוטוקול ההודעה הקודמת. |
| Connection Error | בעיה טכנית בחיבור עם הלקוח. |
| Wrong Password | סיסמא שגויה בניסיון להיכנס. |
| Uniqueness Error | קיים שם משתמש כזה כבר. |
| Other | טעות לא צפויה בריצת התוכנה. |
| הודעות שקשורות להצפנה | |
| Diffie <Level> | העברת מספרים (והשלב שבו העברת המפתחות נמצאת) ומפתחות בצורה מאובטחת בין השרת והלקוח. |

הסבר לצבעים בטבלה:

אדום = הודעה שגוררת ניתוק קשר עם הלקוח. תגובה לביצוע מעשה חמור שדורש ניתוק קשר עמו.

צהוב = בעקבות מספר מסוים של פעמים שהודעה שתגובתה צהובה, הקישור ייסגר, על מנת למנוע מתקפות Brute Force.

כחול = להודעה המיוחסת שאין הגבלה למספר הפעמים ששולחים אותה. עם זאת, היא חוסמת את המשתמש מלהמשיך בהליכים שלו. במקרה זה – להירשם במערכת.

ירוק = ההודעה האופטימלית והאידאלית לכל פעולה – אישור של המערכת.

מבנה הודעה:

וידוא "סיסמא חזקה" יימצא בצד הלקוח ולכן אין הודעה שמתארת תקשורת שקשורה לכך.

הגבלות

לפרויקט שלי יש כמה סייגים והגבלות – נושאים שבהם לא יעסוק.

שלמות פעולת רשת הנוירונים[[20]](#footnote-20):

כתיבת רשת נוירונים היא משימה קשה. כתיבת רשת נוירונים באחוזי טעות מזעריים ובאופן כללי מספיק כך שהמוצר יהיה מוגמר לגמרי היא משימה קשה אף יותר. הפרויקט שלי מציע כוח חישוב של רשת נוירונים שצודקת ברוב מוחלט של הפעמים ברוב התנאים. למרות זאת, ישנם מצבים בהם תפקודה של הרשת לא יהיה אופטימלי. למשל, במקרה של משתמש עם משקפיים, הרשת מזהה פחות טוב עיניים פקוחות מאשר אנשים שלא מרכיבים משקפיים. זאת שבל בעיות מינוריות ב haarcascade של cv2 ובמידע האימון של רשת הנוירונים. זאת ועוד, משום שכתיבת הקוד אינה הסתיימה, המגבלה יכולה להצטמצם עם שיפורים ברשת הנוירונים שאבצע על להגשת הפרויקט.

רציפות העברת המידע בבדיקת העיניים:

משום ש Android היא מערכת שמוגבלת משאבים באופן בולט, שליחת תמונות בקצב גדול מדי יכול לגרום ללקוח שלי להפסיק לעבוד. לכן, קצב העברת התמונות מהלקוח יחקרו במהלך כתיבת הקוד, עד להגעה לקצב אופטימלי של תמונות. באופן פרקטי, בתקופת העבודה על תכנון הפרויקט, חשבתי לבצע אותו ללא תקשורת בבדיקת העיניים של המשתמש. משום שזהו פרוייקט הגמר, ואחת מהדרישות הבסיסיות ביותר היא בניית מערכת תקשורת בין מחשבים וגם משום שהעברת מודל שאומן במחשב למצב שבו יוכל לרוץ על מערכת Android העדפתי לעשות את הפרויקט בדרך הנוכחית.

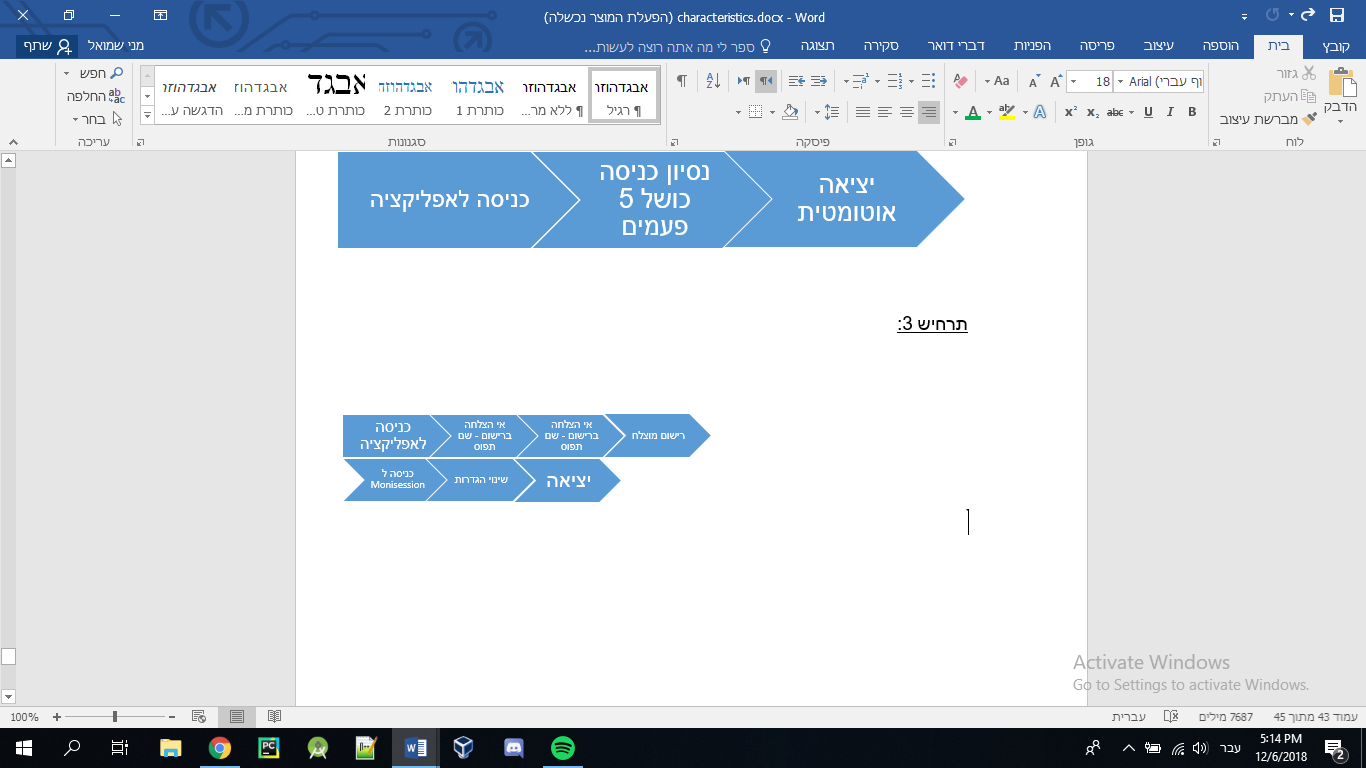
במידה ואחליט להמשיך ולהתקדם עם הפרוייקט גם לאחר הלימודים, אשלב את המודל המאומן בתוך פלטפורמת ה Android והתקשורת תיעשה לצרכי ניהול חשבונות בלבד.

תרחישים אפשריים

תרחיש 1:

תרחיש 2:

תרחיש 3:



קישורים חיצוניים

<https://he.wikipedia.org/wiki/%D7%A8%D7%A9%D7%AA_%D7%A2%D7%A6%D7%91%D7%99%D7%AA_%D7%9E%D7%9C%D7%90%D7%9B%D7%95%D7%AA%D7%99%D7%AA> – רשת נוירונים ויקיפדיה

<https://www.youtube.com/watch?v=mDLkZGxuA-8> – שיחה עם אריקה

<https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi> – סדרת סרטונים על רשת נוירונים Convolutional.

<https://medium.com/@curiousily/tensorflow-for-hackers-part-iii-convolutional-neural-networks-c077618e590b> - כתיבת קוד למזהה חתולים וכלבים – עזר לי בכתיבת הקוד שלי.

<https://www.youtube.com/watch?v=--tnZMuoK3E> – Salted hash.

<https://pythonprogramming.net/tensorflow-neural-network-session-machine-learning-tutorial/> - סדרת סרטונים שמסבירים איך לכתוב רשת נוירונים בפייתון.

<https://developer.android.com/> - אתר אנדרואיד סטודיו ממנו אפשר למצוא תשובה לכל שאלה.

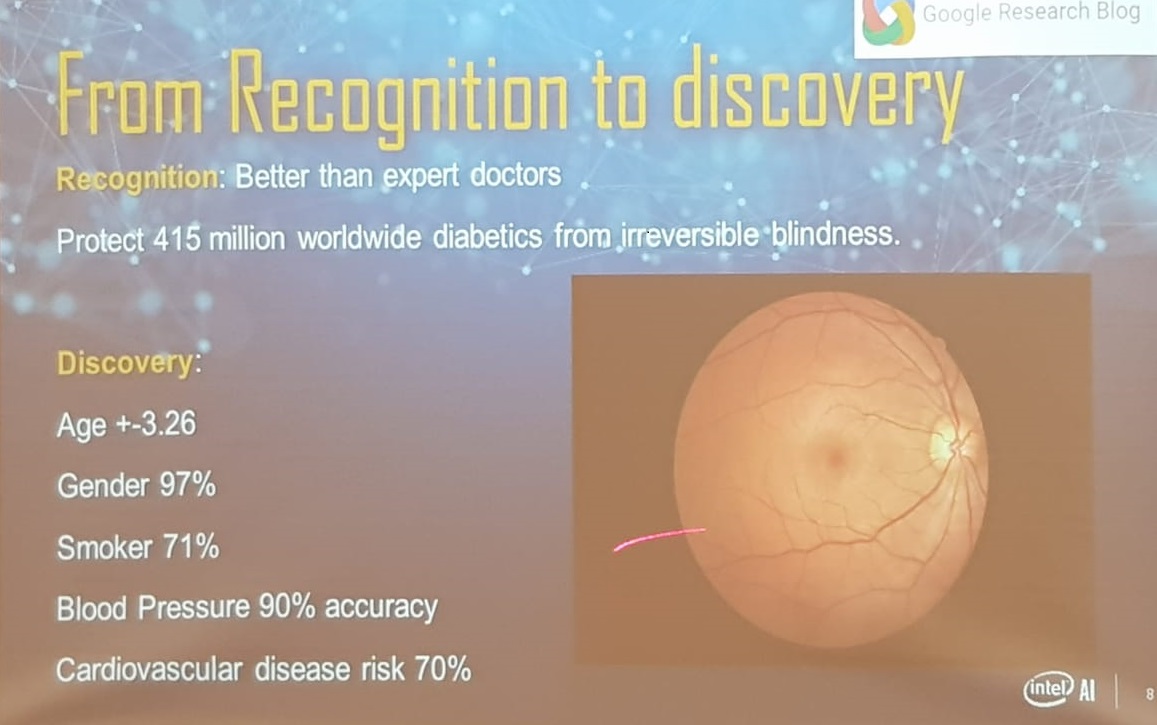
<https://www.youtube.com/watch?v=FmpDIaiMIeA> – איך CNN עובד?

<https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/> - מאמר שמסביר שורה אחר שורה את הקוד של רוזברוק.

<https://www.youtube.com/watch?v=0w8FaiKP5h8> – סרטון שמראה איך הקוד של רוזברוק עובד.

<https://cloud.google.com/vision/?utm_source=google&utm_medium=cpc&utm_campaign=emea-il-all-iw-dr-bkws-all-all-trial-b-gcp-1003963&utm_content=text-ad-none-any-DEV_c-CRE_170514772533-ADGP_Hybrid+%7C+AW+SEM+%7C+BKWS+~+BMM_1:1_IL_IW_ML_Vision+API_google+vision-KWID_43700016973744999-kwd-84320397668-userloc_1007982&utm_term=KW_%2Bgoogle%20%2Bvision-ST_%2Bgoogle+%2Bvision&ds_rl=1242853&ds_rl=1245734&ds_rl=1245734&gclid=Cj0KCQiA3IPgBRCAARIsABb-iGIiyee3CHuRt8OJMxJ-SktvmjXVzND8pbSD21X2DxbO6o9-T8gCJYsaAnFXEALw_wcB> – הסבר על הפונקציונליות של הממשק הויזואלי של גוגל וניסוי הממשק שלהם.

נספח 1 – שקופית מתוך מצגת של Intel.



1. Monitoring + session = Monisession [↑](#footnote-ref-1)
2. ראה עמוד קישורים. [↑](#footnote-ref-2)
3. עם זאת, לא נקדים את המאוחר. בפרק המעשי, נבין מה זאת אומרת "ערכים בתוך המכונה". [↑](#footnote-ref-3)
4. השכבה החבויה מכילה בתוכה שכבות רבות. [↑](#footnote-ref-4)
5. תהליך זה הוא סוג של "נרמול". [↑](#footnote-ref-5)
6. יש כמה פונקציות אקטיבציה שימושיות במיוחד. בפרויקט שלי השתמשתי בפונקציית 'ReLU'. [↑](#footnote-ref-6)
7. עם זאת, יש להדגיש ש"נרמול" המידע בצורה תכנותית-רגילה לפני ההרצה ברשת עוזר מאוד לקבל תשובות הגיוניות בזמן קצר יחסית של אימון. [↑](#footnote-ref-7)
8. <https://keisan.casio.com/exec/system/15362815974714> - מחשבון ReLU. [↑](#footnote-ref-8)
9. תהליך שינוי הערכים בתוך הרשת לטובת שיפורה. [↑](#footnote-ref-9)
10. במידה ונרצה להקטין את ערך נוירון הפלט, נקטין את הקשרים שמגדילים אותו (הפוך ממה שכתוב בסעיפים הנ"ל) [↑](#footnote-ref-10)
11. תמונה של תקרת הקפלה הסיסטנית במדינת הוותיקן. [↑](#footnote-ref-11)
12. המחשב לא יודע לקרוא תמונה כתמונה כמו מבקר תמונות, אלא קיימת המרה של המידע למערך של בתים, כאשר כל אחד מייצג פיקסל על פי הבהירות שלו. לפיכך, הוא "מעביר את החלון" על המערך על ידי חישובים מתמטיים. [↑](#footnote-ref-12)
13. הערכים של המטריצה הנ"ל משתפרים בכל ריצת מכונה על ידי אלגוריתם Gradient Decent כפי שתואר בפרק הקודם. [↑](#footnote-ref-13)
14. יש להדגיש שבמידה ו n קטן מדי, יהיה עומס גדול על הרשת, ובמידה ו n גדול מדי, הרשת לא תתפקד כמו שצריך. לכן עדיף לשאול שאלות על כך בכמה שיותר פורומים שונים עד שמוצאים את התשובה, או לקיים ניסוי וטעייה בעניין. [↑](#footnote-ref-14)
15. כמובן שמכונה מזהה הרבה מאוד תכונות שמאפיינות אובייקט מסוים בתמונה, ולמשל בתחום Unsupervised Learning תפקיד המכונה הוא לציין תכונות יוצאות דופן (שסביר להניח שאדם לא יבחין בהם) בתמונה מסוימת. [↑](#footnote-ref-15)
16. ראה נספח 1. [↑](#footnote-ref-16)
17. קיימים כמה תיעודים לשיחה עם הרובוט הנ"ל, ביניהם בעמוד הקישורים. [↑](#footnote-ref-17)
18. התהליך הזה הוא המלצה שלי למי שרוצה לעשות מספר מינימלי של טעויות בבניית פרוייקט שנוגע ברשתות נוירונים. עם זאת, אני ממליץ בפרויקט הראשון שמערב רשת נוירונים לקיים את חוק 4 בששת הדיברות שבעמוד הבא. [↑](#footnote-ref-18)
19. חשוב להדגיש, שעל מנת ליצור רשת נוירונים ב Tensorflow, אין צורך בידע מעמיק ברשתות נוירונים, אך משום שלקחתי על עצמי את הנושא כנושא מחקר בכל מצב, התנסיתי בו גם בכלים אחרים, וחקרתי אותו לעומק. [↑](#footnote-ref-19)
20. כל זאת נכון לזמן כתיבת האפיון – דצמבר 2018. כל זאת יכול להשתפר עד להגשת הפרויקט. [↑](#footnote-ref-20)