פרויקט גמר 5 יחידות לימוד

התמחות- תכנון ותכנות מערכות Deep Learning



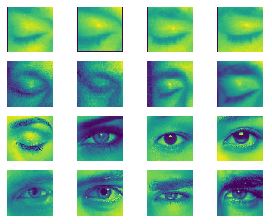
**בית הספר:** מקיף י"א ראשונים

**שם התלמיד:** יניב מובשוביץ

**ת"ז התלמיד:** 325638013

**שם המנחה:** דינה קראוס

**תאריך הבחינה:** 5.7.2022



**תוכן עניינים**

[מבוא 3](#_Toc106456458)

[מבנה / ארכיטקטורה 4](#_Toc106456459)

[איסוף, הכנה, וניתוח הנתונים 4](#_Toc106456460)

[שלב בניית המודל 6](#_Toc106456461)

[הסבר על סוגי השכבות השונות ברשת 6](#_Toc106456462)

[שלב היישום (Software deployment) 17](#_Toc106456463)

[תיאור והסבר כיצד היישום משתמש במודל 17](#_Toc106456464)

[תרשים UML של המודלים 17](#_Toc106456465)

[תיאור הטכנולוגיה שעל פיה מומש ממשק המשתמש 18](#_Toc106456466)

[תיאור קוד הקולט את ה- DATA שעליו יבוצע החיזוי והתאמתו למבנה נתונים המתאים לחיזוי 18](#_Toc106456467)

[מדריך למפתח 20](#_Toc106456468)

[קובץ ה- test\_creation.py: 21](#_Toc106456469)

[קובץ ה- deleting\_images.py: 22](#_Toc106456470)

[קובץ ה- model\_learning.py: 23](#_Toc106456471)

[קובץ ה- model\_testing.py: 28](#_Toc106456472)

[קובץ ה- eyes\_app.py: 31](#_Toc106456473)

[קובץ ה- main\_form.py: 32](#_Toc106456474)

[מדריך למשתמש 33](#_Toc106456475)

[הסברים על החלונות בממשק המשתמש 34](#_Toc106456476)

[תפקידי הכפתורים 38](#_Toc106456477)

[הודעות למשתמש (alert למיניהם) 39](#_Toc106456478)

[רפלקציה 41](#_Toc106456479)

[ביבליוגרפיה 42](#_Toc106456480)

# מבוא

השנה במסגרת המגמה פרויקט הגמר שלנו צריך להיות קשור לנושא "למידה עמוקה".

(Deep Learning). תחום זה עוסק ביכולת של המחשב לזהות ולהבין דברים בדרך חשיבה זהה לחשיבתו של האדם. מוח האדם בנוי מכמות רבה של נוירונים אשר כל אחד עושה עבודה פשוטה ומינימלית, אך כאשר יש כמות רבה של נוירונים הנקראות שכבות נוירונים העובדות במקביל ובתיאום, המוח יכול לבצע פעולות מורכבות. כך גם תחום ה"למידה העמוקה" עובד, וזה בא לידי ביטוי בפרויקט שלי. רשת נוירונים של "למידה עמוקה" עובדת באופן שבו היא מורכבת ממשקלים ונוירונים שכל אחד מהם מבצע פעולה מתמטית פשוטה, ובסופו של דבר התוכנה מסוגלת ללמוד באופן דומה למוח האנושי.

בפרויקט זה החלטתי לכתוב תכנה המזהה אם העין של הבן אדם היא פתוחה או סגורה. דבר זה מתבצע לאחר אימונים רבים של המודל. בחרתי בנושא זה כי אני מאמין שהוא יכול לפתור המון בעיות, כמו לבדוק האם אנשים ישנים בשיעור (רואים בתמונות בזמנים סמוכים שהעיניים שלהם עצומות הרבה פעמים, לכן התלמיד נרדם או אינו מרוכז בשיעור), או לראות האם הנהגים מרוכזים בנהיגה.

פרויקט זה יכול לעבוד בצורה נהדרת בזמן אמת (real time), הוא יכול לעבוד בשיתוף עם מצלמות אבטחה, ופרויקטים נוספים שאחד מזהה אנשים מתמונה (תמונות שנלקחות ממצלמת האבטחה) ושני שמזהה עיניים מתמונות של אנשים, ולאחר מכן מגיע הפרויקט שלי, שמזהה האם העין סגורה או פתוחה

רעיון לפרויקט זה הגיע כאשר ראיתי סרטון ב- YouTube על אנשים שמדברים על כך שלפעמים הם מתקשים להתרכז בנהיגה עקב עייפות, דבר זה מסוכן מאוד גם להם וגם לעוברי הדרך, ואיזו דרך טובה יותר יש בלזהות האם האנשים מרוכזים בנהיגתם מלזהות את מצב עיניים לאורך זמן?  
אם העיניים שלהם עצומות יותר זמן מהרגיל, זאת אומרת שהם אינם מרוכזים בנהיגתם, ויש לעשות משהו בנידון (כמו קבלת הודעה מהאוטו שיש לעצור ולהתרענן, קבלת אס-אמ-אס, או במקרים חמורים יותר שדבר שכזה חוזר על עצמו, חלוקת דו"חות.

מטרת פרויקט זה היא לעזור לאנשים, בין אם בדרכים כמו שציינתי מקודם, או בדרכים אחרות ויצירתיות כמו לזהות האם מישהו לא מרוכז בשיעור של מרצה.

למעשה, בחקר שוק שעשיתי גיליתי כי החברה Panasonic עוסקת בנושא זה, מניעת הירדמות אנשים בנהיגה. והם אכן משתמשים בטכנולוגיה אותה אני רוצה לפתח בפרויקט זה, הם מזהים את מצב הערנות של הנהג בדרכים מתוחכמות ביותר כמו: קצב מצמוציו (בדומה לפרויקט שלי, זיהוי מצב העין), הבעות הפנים שלו, תנועות ההגה, מהירות הרכב, טמפרטורות גוף הנהג, וממיינים את מצב ערנותו של הנהג ל-5 מצבים: עירני, עירני פחות, מנומנם, מנומנם מאוד ומנומנם בצורה חמורה.

תחום ה- deep learning הינו תחום חדשני מאוד, והינו תחום שנמצא בפיתוח תמידי – כלומר יכולים להיות לי קשיים במהלך עבודתי על פרויקט זה.

בתחום זה צריך כמות גדולה מאוד של תמונות כדי שהמחשב יוכל ללמוד לפיהן לסווג תמונות מסוימות לדברים מסוימים (במקרה שלי, להחליט לפי תמונה של עין האם העין שבתמונה פקוחה או עצומה).

לכן אני חושש שאתקע בשלב מציאת מאגר התמונות שיתאים לפרויקט שלי בצורה הטובה ביותר, ואבזבז זמן רב על שלב זה.

# מבנה / ארכיטקטורה

## איסוף, הכנה, וניתוח הנתונים

את מאגר התמונות מצאתי באתר Kaggle, באתר זה מצאתי dataset של תמונות שמתאימות למטרת הפרויקט שלי. במאגר תמונות זה היה שלל תמונות של עיניים סגורות ועיניים פתוחות,   
כ- 360,000 תמונות! [https://www.kaggle.com/datasets/mukhtarkassar/eyes-images?select=eyesDataset](https://www.kaggle.com/datasets/mukhtarkassar/eyes-images?select=eyesDataset" \t "_blank" \o "https://www.kaggle.com/datasets/mukhtarkassar/eyes-images?select=eyesDataset)

חשוב לציין שרוב התמונות במאגר מידע זה הינן בשחור לבן (grayscale)

לשם התמונה קיימת משמעות ותבנית, תמונות של עין פתוחה מתחילות באות O, ותמונות של עין סגורה מתחילות באות – C. כל תמונה ממוספרת כדי להבדיל בין התמונות השונות של העיניים.

בנוסף אם העין של התמונה היא עין ימין, שם התמונה יכיל R. במידה והעין היא עין שמאל, שם התמונה יכיל L.

לצד העין אין משמעות בפרויקט זה, אך שם התמונות הגיע בפורמט זה ב- dataset עצמו ולא שיניתי את שמות התמונות, לכן גם המספרים של התמונות יכולים להגיע עד כ- 350,000 (ככמות התמונות ב- dataset המקורי)

להלן דוגמות לשמות של התמונות:

O-L-37900.jpeg (כלומר, תמונה של עין שמאלית פתוחה)

O-R-225002.jpeg (כלומר, תמונה של עין ימנית פתוחה)

C-L-1338.jpeg (כלומר, תמונה של עין שמאלית סגורה)

C-R-23893.jpeg (כלומר, תמונה של עין ימנית סגורה)

מאגר התמונות לאימון המודל שלי כולל תמונות מסוג jpeg ומחולק לתיקיות, כאשר כל תיקייה מייצגת קבוצת עיניים, תיקייה אחת נקראת closed look והתיקייה השנייה נקראת open look. זאת על מנת שהמחשב ידע איזה עין מוצגת לפניו וכך יבצע את תהליך הלמידה (למעשה ה- Label של כל תמונה ודרך האימון מתבצע בעזרת החלוקה לתיקיות). בנוסף, באמצעות החלוקה לתיקיות, נחסך זמן ריצה יקר משום שאין צורך בכל הרצה מחדש למיין את התמונות.

בנוסף, יש לי מאגר תמונות נוסף אשר מטרתו היא לבחון את המודל, עליו יבוצע שלב החיזוי, לפי מה שהמחשב למד הוא יכול לחזות איזו תמונה של עין הוא קיבל, סגורה או פתוחה. המחשב מביע את דעתו על כל תמונה ואומר האם היא סגורה או פתוחה.

במהלך עבודתי על מאגר המידע הבחנתי כי ישנן תמונות רבות אשר אינן מיועדות לאימון (תמונות קטנות מאוד ברזולוציה קטנה), בכדי להימנע מאימון על תמונות כאלו אשר יכולות להזיק למודל עצמו, כתבתי קטע קוד אשר ימחק את כל התמונות הקטנות. המודל שלי מתאמן על תמונות ברזולוציה של 105x105, בכדי למחוק תמונות בעלות רזולוציה נמוכה מידי אשר אי אפשר לשנות להן את הרזולוציה (הבדלים גדולים מידי), כתבתי קטע קוד (הפעולה deleting\_small\_photos אשר עליה ארחיב בהמשך בפרק – מדריך למפתח) אשר ימחק את כל התמונות שגודלן קטן מ-1800 KB.

לאחר המחיקה הבחנתי כי במאגר התמונות שלי קיימות תמונות רבות אשר דומות כמעט לחלוטין אחת לשנייה, דבר שיכול לגרום ל- overfitting, לכן במטרה להעלים את ה – overfitting מחקתי בצורה ידנית(!), כ-10 אלף תמונות! הדבר שיפר את ביצועי המודל, אך ה- overfitting עדיין היה קיים ולא היה שיפור משמעותי, כלומר לא מחקתי מספיק תמונות.   
שיטה זו גזלה לי זמן יקר רב, והבנתי כי אם אני לא מוצא שיטה יעילה יותר, אני לא אספיק לשפר את מצב המודל שלי. חיפשתי אפשרויות למחיקת תמונות דומות באינטרנט, ומצאתי כי באמת אפשר למחוק תמונות באמצעות קטע קוד ולא לבזבז זמן רב במחיקת תמונות באופן ידני.



כתוצאה מכך כתבתי קטע קוד (הפעולות delete\_similar\_images ו – similarity\_check) אשר ימחק את התמונות אם הן דומות במידה מסוימת ומעלה (לפי ה- hash שלהן).

לאחר עבודות אלו על מאגר התמונות נותרתי עם 4,894 תמונות! (לעומת 360,000 במאגר המידע המקורי)

הקפדתי שלכל מצב שעין נמצאת בו (כלומר סגור או פותח) תהיה את אותה כמות תמונות ( 2447 לכל סוג)

את מאגר המידע שלי הכנתי לאימון בצורה הבאה:

1. תחילה העברתי חלק מה- dataset ל- test בהתאם ל- test split שהוחלט קודם לכן (בדרך כלל 20%)
2. לאחר מכן באמצעות הפעולה של הספרייה tensorflow ו- keras tf.keras.utils.image\_dataset\_from\_directory, מיינתי את ה- dataset שלא הועבר ל -test ל- train ול- validation. באמצעות החלוקה לתיקיות נוצרו תמונות עם label שמתאים לתמונה עצמה.
3. לאחר מכן, עם בניית המודל מגיע שלב נרמול התמונות. אני מבצע את הנרמול באמצעות שכבה במודל.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטינרמול למעשה הינו העברה של מספרים גדולים בעלי ערכים שונים אחד מהשני לטווח מספרים קטן יותר, מ0 ועד 1 (כמובן מספרים לא שלמים ולא רק 0 ו-1), הנרמול מתבצע באופן הבא:

כך למעשה הערך הקטן ביותר (במקרה שלנו הפיקסל הכהה ביותר) יהיה 0, והערך הגדול ביותר (במקרה שלנו הפיקסל הבהיר יותר) יהיה 1.

## שלב בניית המודל

### הסבר על סוגי השכבות השונות ברשת

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

במודל שלי קיימות 4 שכבות נוירונים, אך לכל שכבה יש כמות שונה של נוירונים.

ה- activation function שבחרתי ל- Convolutional layer הוא relu, פונקציית הפעלה זו היא הפונקציה שהמודל למד איתה בצורה הטובה ביותר.

ה- activation function שבחרתי ל- dense layer הוא softmax, בעבר ניסיתי את פונקציית ההפעלה sigmoid, אך הביצועים של המודל טובים יותר כאשר אני משתמש ב-softmax.

שכבה 1 – Convolutional layer

|  |  |
| --- | --- |
| 32 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| relu | Activation function |
| 0.2 | Dropout |

שכבה 2 – Convolutional layer

|  |  |
| --- | --- |
| 128 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| relu | Activation function |
| 0 | Dropout |

שכבה 3 – Convolutional layer

|  |  |
| --- | --- |
| 32 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| relu | Activation function |
| 0.2 | Dropout |

שכבה 4 – Dense layer

|  |  |
| --- | --- |
| 16 | Neurons amount |
| softmax | Activation function |

**נתונים נוספים בנוגע למודל הנוכחי עליהם אסביר בהמשך:**

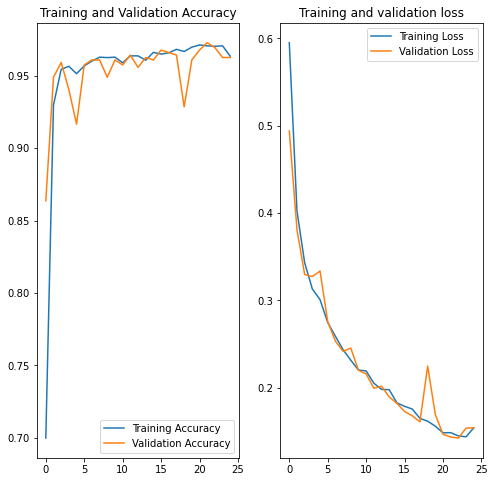
batch\_size = 64 # the size of each images batch

image\_height = 105 # images height

image\_width = 105 # images width

NB\_EPOCH = 25 # amount of times the model will train each image

להלן גרפים המתארים את התקדמות לימוד המודל בכל epoch



**משתנה בלתי תלוי בשני הגרפים: epoch**

**משתנה תלוי בגרף הימני: loss**

**משתנה תלוי בגרף השמאלי: accuracy**

הגרף הימני מתאר את ה- loss של ה- train וה- validation. ה- validation הינו הגרף הכתום וה- train הינו הגרף הכחול.

הגרף השמאלי מתאר את ה- accuracy של ה- train וה- validation. ה- validation הינו הגרף הכתום וה- train הינו הגרף הכחול.

הבחנתי שכאשר אני מריץ יותר פעמים מ- 25 ביצועי המודל מפסיקים להשתפר בקצב רציני ולכן עצרתי ב- epoch ה- 25.

במהלך עבודתי בשנה זו שיניתי רבות את מאפייני השכבות וכמותן, עד שהגעתי לתוצאות שאני מאמין שהן מספיק טובות ומרצות אותי, עכשיו אפרט ואציג שניים מבין הניסיונות הרבים שהיו לי במהלך השנה.

**ההרצה הראשונה שלי:** תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

שכבה 1 – Convolutional layer

|  |  |
| --- | --- |
| 16 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| relu | Activation function |
| 0.2 | Dropout |

שכבה 2 – Convolutional layer

|  |  |
| --- | --- |
| 32 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| relu | Activation function |
| 0 | Dropout |

שכבה 3 – Convolutional layer

|  |  |
| --- | --- |
| 16 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| relu | Activation function |
| 0.2 | Dropout |

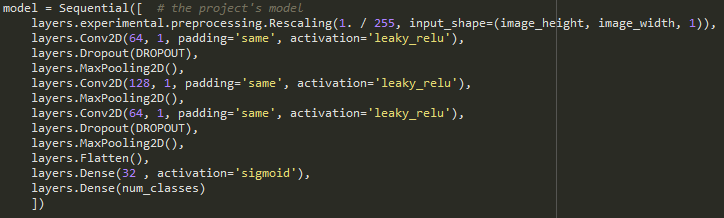
שכבה 4 – Dense layer

|  |  |
| --- | --- |
| 1 | Neurons amount |
| sigmoid | Activation function |

ביצועי מודל זה היו מזעזעים, וכמעט ולא הייתה התקדמות ככל שה- epoch התקדמו, ידעתי שאני מוכרח לבצע שינויים.



**הניסיון השני שבחרתי להציג בספר הפרויקט:**



**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

.

שכבה 1 – Convolutional layer

|  |  |
| --- | --- |
| 64 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| Leaky relu | Activation function |
| 0.2 | Dropout |

שכבה 2 – Convolutional layer

|  |  |
| --- | --- |
| 128 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| Leaky relu | Activation function |
| 0 | Dropout |

שכבה 3 – Convolutional layer

|  |  |
| --- | --- |
| 64 | Neurons amount |
| 1 | Strides |
| Same | Padding |
| Leaky relu | Activation function |
| 0.2 | Dropout |

שכבה 4 – Dense layer

|  |  |
| --- | --- |
| 32 | Neurons amount |
| sigmoid | Activation function |

להלן גרפים המתארים את התקדמות לימוד המודל בכל epoch

****

ניתן לראות שאפילו שה- loss ב- train קטן עם הזמן וה- accuracy גדל עם הזמן (כלומר המודל משתפר) ה- validation לא משתפר כלל, הוא עולה ויורד כל הזמן. ה- loss וה- accuracy לא יציבים כלל, לכן הבנתי שאני צריך לבצע שינוי נוסף אף על פי שתוצאות מודל זה טובות בהרבה מהמודל הקודם שהצגתי.

**דוח הכולל ריכוז כל ה- Hyper Parameters במודל הסופי**

|  |  |
| --- | --- |
| **פרמטרים שקשורים לאימון** | **פרמטרים שקשורים לצורה של רשת הנוירונים** |
| Learning rate – 0.001 | Number of hidden layers - 4 |
| Epoch – 25 | Dropout – 0.2 |
| Iterations – 53 | Activation function –relu, softmax |
| Batch Size – 64 | Weights initialization - glorot\_uniform |
| Optimizer Algorithm- Adam |  |
| Momentum – 0.01 |  |

**דוח הכולל ריכוז כל ה- Hyper Parameters במודל הראשון**

|  |  |
| --- | --- |
| **פרמטרים שקשורים לאימון** | **פרמטרים שקשורים לצורה של רשת הנוירונים** |
| Learning rate – 0.001 | Number of hidden layers - 4 |
| Epoch – 25 | Dropout – 0.2 |
| Iterations – 53 | Activation function –relu, sigmoid |
| Batch Size – 64 | Weights initialization - glorot\_uniform |
| Optimizer Algorithm- Adam |  |
| Momentum – 0.01 |  |

**דוח הכולל ריכוז כל ה- Hyper Parameters במודל השני**

|  |  |
| --- | --- |
| **פרמטרים שקשורים לאימון** | **פרמטרים שקשורים לצורה של רשת הנוירונים** |
| Learning rate – 0.001 | Number of hidden layers - 4 |
| Epoch – 25 | Dropout – 0.2 |
| Iterations – 111 | Activation function –leaky relu, sigmoid |
| Batch Size – 32 | Weights initialization - glorot\_uniform |
| Optimizer Algorithm- Adam |  |
| Momentum – 0.01 |  |

**תיעוד והסבר של פונקציית השגיאה**



פונקציית השגיאה מודדת את המרחק בין התשובה האמיתית לתשובה של המודל. ככל שערך הפונקציה קטן יותר, מרחק התוצאות קטן יותר וכך גם דיוק המודל.

בפרויקט שלי אני משתמש בפונקציית שגיאה שמיועדת למיון של יותר מ-2 סוגים שונים של תמונות. (במקרה שלי יש 2 סוגים, עין עצומה ועין פתוחה).

פונקציית שגיאה זו מקבלת שני ערכים.

y\_true – רשימה בגודל ה- batch\_size, יכיל את ה- label האמיתי של התמונה במספר שלם, כלומר 0, או 1 (עין סגורה או פתוחה)

y\_pred – רשימה דו ממדית אשר גודלה הוא [batch\_size, num\_classes]

דוגמה לצורת המשתנים כאשר ה- batch\_size הוא 2 וה- num\_classes הוא 3.



תמונה אחת היא מסוג 1, והתמונה השנייה היא מסוג 2.

בתמונה הראשונה, המודל חושב שיש 5% שהתמונה היא מסוג 0 , 95% שהתמונה היא מסוג 1 ו- 0% שהתמונה היא מסוג 2.

בתמונה השנייה, המודל חושב שיש 10% שהתמונה היא מסוג 0, 80% שהתמונה היא מסוג 1, ו- 10% שהתמונה היא מסוג 2.

כלומר, המודל כמעט בטוח בתשובתו על התמונה הראשונה, ותשובתו אכן נכונה, אך במקרה השני המודל טועה, הוא כמעט בטוח (80%) שהתמונה היא מסוג 1 גם כן בעוד היא 2. מה שמעלה משמעותית את ערך ה- loss function במקרה זה (1.1769392)

במידה ונשנה את ה- label של התמונה השנייה למה שהמודל חושב שהיא, ערך ה- loss יקטן משמעותית מאחר והמודל כמעט בטוח בשתי תשובותיו, ושתיהן נכונות.

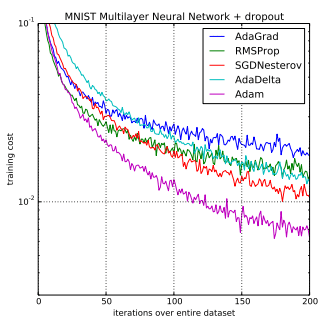


(ערכים אלו מורידים את ה- loss ל- 0.137)

לכן ככל שהמודל יותר בטוח בתשובותיו, השינוי ב- loss יהיה חד יותר, כלומר אם המודל בטוח בתשובתו והיא אכן נכונה, ה- loss לא יגדל, אם המודל בטוח בתשובתו אך תשובתו אינה נכונה, ה loss יעלה משמעותית, ואם המודל לא בטוח על האם התמונה היא א' או ב' (לדוגמה 50% על כל אחד) ה- loss עדיין יגדל אפילו אם רוב האחוזים הולכים לעבר התשובה הנכונה.

כך בעצם יכול להיווצר מצב שה- accuracy של המודל גבוה מאוד, אך גם ה- loss יצא גבוה מהרגיל ( המודל בוחר בתשובה הנכונה כמעט בכל פעם, אך אינו בטוח בתשובותיו).

**תיעוד והסבר של ייעול ההתכנסות (Optimization)**

ה- optimizer בו אני משתמש בפרויקט שלי הוא Adam (Adaptive Moment Estimation) , הוא יעיל מאוד (מתאים לבעיות עם שיפוע מתון מאוד, כלומר משנה את קצב הלימוד בהתאם לצורך), לא לוקח הרבה  
זיכרון והוא מותאם נהדר לעבודות עם מידע ופרמטרים רבים.

ה- optimizer בכללי הוא אלגוריתם שמשתמשים בו בכדי לשנות ערכים ברשת הנוירונים, כמו משקלים וקצב הלימוד (ה- learning rate) במטרה להוריד את ה- loss של המודל ולשפר את ביצועיו.

בתמונה ניתן לראות כי מבין כל ה- optimizers המופיעים בגרף, ה- optimizer שעובד בצורה הטובה ביותר הינו Adam.

## שלב היישום (Software deployment)

### תיאור והסבר כיצד היישום משתמש במודל

כאשר מתחילים את העבודה עם היישום, נשאל המשתמש האם הוא רוצה לאמן מודל חדש או להשתמש במודל ישן ולבחון אותו.

במידה והמשתמש בוחר לאמן מודל חדש המשתמש בוחר split rate ולאחר מכן המודל מתאמן.

היישום משתמש במודל כאשר מגיע שלב הבחינה. המשתמש בוחר מודל בקובץ מסוג h5 והמודל באמצעות הפעולות model\_score, model\_predict אשר נמצאות בקובץ model\_testing.py אשר עליו ארחיב בהמשך.

למשתמש ניתנת האפשרות להוסיף תמונות לבחינת המודל, עליו לשים במקומות המתאימים (כלומר עיניים פתוחות במיקום העיניים הפתוחות ב- test ועיניים סגורות במיקום העיניים הסגורות ב- test), וכמות זהה של תמונות מכל סוג, כך למעשה יכול המשתמש לבחון את המודל על התמונות שרלוונטיות אליו.

### תרשים UML של המודלים

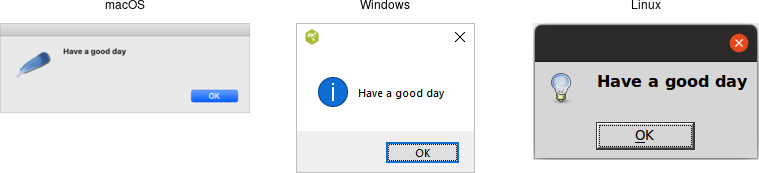
### תיאור הטכנולוגיה שעל פיה מומש ממשק המשתמש

במטרה לממש את ממשק המשתמש בפרויקט זה השתמשתי ב- tkinter

Tkinter הינה ספריית GUI (ממשק משתמש) בסיסית בפייתון. פייתון בשילוב עם tkinter מביא דרך קלה ומהירה ליצירת אפליקציות למשתמש בצורה נוחה.

אין צורך להתקין ספרייה זו מאחר שהיא אמורה להגיע עם ההתקנה של הפייתון עצמו (ספרייה סטנדרטית בפייתון), כל שעלינו לעשות היא לזמן אותה בסקריפט שכתבנו ואפשר להתחיל לעבוד!





### תיאור קוד הקולט את ה- DATA שעליו יבוצע החיזוי והתאמתו למבנה נתונים המתאים לחיזוי

כאשר מגיעים לשלב החיזוי (כלומר לאחר שהמשתמש בחר באפשרות של בחינת המודל, ובחר מודל הוא נשאל האם הוא רוצה לאמן על תמונות שלו, או על התמנות אותן המודל אשר בחר לא אימן.

במידה והמשתמש בוחר לחזות תמונות משלו, עליו לשים במקומות המתאימים (כלומר עיניים פתוחות במיקום העיניים הפתוחות ב- test ועיניים סגורות במיקום העיניים הסגורות ב- test), וכמות זהה של תמונות מכל סוג, כך למעשה יכול המשתמש לבחון את המודל על התמונות שרלוונטיות אליו.

לאחר והמשתמש שם במקומות הרלוונטיים את התמונות, פעולות החיזוי מהקובץ model\_testing.py ייקראו.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיבשלב ה- score, כלומר evaluate ה- DATA מותאם לחיזוי בצורה הבאה:

באמצעות הפעולה utils.image\_dataset\_from\_directory של keras, התמונות מהתיקיות יותאמו לחיזוי.

התמונות יהיו בגודל אשר עליהן המודל התאמן (105x105), בשחור לבן, ומפוצלות לשתי קבוצות (עיניים פתוחות וסגורות.

בשלב ה- predict, המצב פשוט יותר, כאשר אנחנו רוצים לחזות תמונה מה- DATA, נשתמש בפעולה הבאה:

גם כאן, התמונה תותאם לגודל בו התמונות אומנו במודל (105x105), ויהיו במצב של שחור לבן.

# מדריך למפתח

פרק זה מכיל את כל תוכן הקוד עם הסברים על כל משתנה ועל כל פעולה עם חשיבות לכלל התוכנית. את הפרויקט שלי חילקתי למספר קבצי קוד אשר לכל אחד מהם יש תחום אחריות שונה. צורה זו הקלה על עבודתי בפרויקט, ועזרה לי לאתר בעיות בצורה נוחה יותר – כאשר רציתי לשנות משהו, ישר ידעתי לאן אני צריך לגשת והיכן לעבוד.

בפרויקט שלי קיימים כשמונה קבצים סך הכל. לכל קובץ תחום אחריות שונה.

|  |  |
| --- | --- |
| **שם הקובץ** | **תפקידו** |
| test\_creation.py | תפקיד קובץ זה הינו לעבוד על ה- dataset.  במידה ולא קיימת תיקייה של test במיקום ה- dataset תיקייה כזו תיווצר.  בקובץ זה מתבצע העברת התמונות ל- test על פי ה- train\_validation\_test split שנקבע למודל.  קובץ זה יכול גם להחזיר בחזרה את התמונות ל- dataset עצמו לאחר העבודה במידה ורוצים. |
| deleting\_images.py | תפקיד קובץ זה הינו לסדר את ה- dataset המקורי. כלומר למחוק תמונות שיכולות לפגום באימון המודל ( כלומר תמונות קטנות מידי ולא ברורות, או מחיקת תמונות רבות אשר דומות אחת לשנייה, דבר אשר יכול לגרום ל- overfitting\*. |
| model\_learning.py | קובץ זה הינו הקובץ העיקרי בפרויקט. בו מתבצע אימון המודל ויצירתו! |
| model\_testing.py | תפקיד קובץ זה הינו לבחון את המודל. בקובץ זה יש שתי אפשרויות בחינה, הראשונה היא predict בה המודל אומר על כל תמונה (שלא למד עליה בעבר, כלומר נתקל בה בפעם הראשונה) מה הוא חושב עליה (כלומר, האם העין בתמונה שמוצגת בפניו פתוחה או סגורה).  או evaluate בה יוצג למשתמש את איכות הניחושים של המודל (כלומר, ה- accuracy וה- loss שלו על ה- test). |
| eyes\_app.py | קובץ זה הינו ממשק המשתמש, כולו עוסק בעבודה עם ממשק המשתמש. |
| main\_from.py | קובץ זה מחבר בין הקבצים הנדרשים ומקל על עבודתי עם הקובץ של ממשק המשתמש  (eyes\_app.py) |

## קובץ ה- test\_creation.py:

תפקיד קובץ זה הינו לעבוד על ה- dataset.

במידה ולא קיימת תיקייה של test במיקום ה- dataset תיקייה כזו תיווצר.

בקובץ זה מתבצע העברת התמונות ל- test על פי ה- train\_validation\_test split שנקבע למודל.

קובץ זה יכול גם להחזיר בחזרה את התמונות ל- dataset עצמו לאחר העבודה במידה ורוצים.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**הפעולה make\_test\_dir()**

פעולה זו יוצרת תיקיה של test במידה וה- dataset שקיבלה לא מחולק ל- dataset עצמו ול- test.

תמונה שמכילה טקסט, שעון

התיאור נוצר באופן אוטומטי

**הפעולה test\_maker(image\_count, test\_split = 0.2)**

פעולה זו מקבלת את כמות התמונות הכוללת ב- dataset, ואת הtest\_split שהוחלט. במידה ולא מקבלת את ה- test\_split, הוא נקבע אוטומטית ל- 0.2.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**הפעולה back\_from\_test(data\_dir)**

פעולה זו מקבלת את כתובת מיקום ה- test במחשב, והיא מחזירה את התמונות שקיימות ב- test אל ה- dataset המקורי.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

## קובץ ה- deleting\_images.py:

תפקיד קובץ זה הינו לסדר את ה- dataset המקורי.

כלומר למחוק תמונות שיכולות לפגום באימון המודל ( כלומר תמונות קטנות מידי ולא ברורות, או מחיקת תמונות רבות אשר דומות אחת לשנייה, דבר אשר יכול לגרום ל- overfitting\*).

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**הפעולה deleting\_small\_photos(data\_dir)**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיפעולה זו מקבלת את המיקום עליו רוצה המשתמש למחוק את התמונות הקטנות, אשר לא יהיו מיעודות לאימון תקין של המודל. פעולה זו מוחקת כל תמונה אשר גודלה לא עולה על 1,800 K/B, תמונות כאלו הן ברזולוציה קטנה למדיי, כלומר יש צורך למחוק אותן.

**הפעולה similarity\_check(photo1, photo2)**

פעולה זו מקבלת שתי תמונות ובודקת לפי ה- hash\* שלהן, כמה הן דומות – במידה והתמונות דומות למדיי, הפונקציה מחזירה True, אחרת False. תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**הפעולה delete\_similar\_images(data\_dir)**

הפעולה הזו מקבלת את הכתובת אשר עליו רוצה לעבור, ולמחוק האם קיימות בו תמונות זהות.

פעולה זו עובדת עם הפעולה similarity\_check(photo1, photo2), היא שולחת לפעולה הזו שתי תמונות מהכתובת אשר קיבלה, ובמידה ומקבלת True, הפעולה הזו תמחק את התמונה ממקור המידה (ה- dataset אשר קיבלה) תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

## קובץ ה- model\_learning.py:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיקובץ זה הינו הקובץ העיקרי בפרויקט. בו מתבצע אימון המודל ויצירתו!

**הפעולה creating\_test\_dataset(image\_count, test\_split)**

פעולה זו מקבלת את כמות התמונות הכוללת ב- dataset, ואת הtest\_split שהוחלט. היא קוראת לפעולה test\_maker(image\_count, test\_split = 0.2) מהקובץ test\_creation.py.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**הפעולה model\_training(train, val, test)**

פעולה זו היא הפעולה הראשית בכל הפרויקט, בה מתבצע לימוד המודל.

תמונה שמכילה טקסט, מסך, צילום מסך

התיאור נוצר באופן אוטומטיהפעולה מקבלת את ה- train, validation, test split. כל אחד מהם מגיע בכמה אחוזים כל חלק יקבל, כלומר יש לחלק אותם פי 100 בכדי לעבוד איתם.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**batch\_size = 64**

ה- dataset יחולק לקבוצות במהלך האימון, גודל כל קבוצה של תמונות הוא 64.

**DROPOUT = 0.2**

המודל ישתמש ב- Dropout של 0.2 במהלך אימון המודל.

**image\_height = 105**

גובה כל תמונה במהלך האימון יהיה 105 פיקסלים.

**image\_width = 105**

רוחב כל תמונה במהלך האימון יהיה 105 פיקסלים.

**NB\_EPOCH = 25**

המודל יעבור על כל תמונה ותמונה 25 פעמים.

**test\_split = test/100**

כפי שנאמר קודם לכן, הפעולה מקבלת את ה- split ratio באחוזים, ובכדי לעבוד איתם יש לחלק פי 100 (מכפילים את כמות התמונות ב- dataset באחוזים חלקיי 100)

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

החילוק ל- train, validation מתבצע על ידי keras.preprocessing.image\_dataset\_from\_directory. פעולה זו מקבלת את ה- dataset המלא (לאחר שהועברו תמונות לתיקיית test על ידי הפעולה creating\_test\_dataset) ומחלקת אותו ל- train ול- validation. היא מחלקת את שניהם למקבצים קטנים (batches) לפי המשתנה batch\_size, והופכת כל תמונה לגודל זהה (לפי המשתנים image\_height, image\_width)

ב- train\_ds יהיו התמונות אשר עליהן המודל יתאמן (train)

ב- validation\_ds יהיו התמונות אשר עליהן המודל יוודא את אמינותו (validation)



**history**

משתנה זה שומר את כל ההיסטוריה של כל epoch במהלך אימון המודל. באמצעותו ניתן להדפיס גרף אשר יציג את התקדמות המודל לאורך שלב ה- train וה- validation.

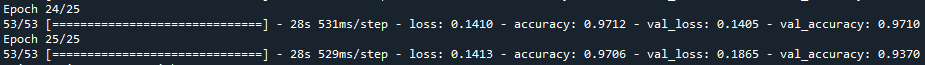
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

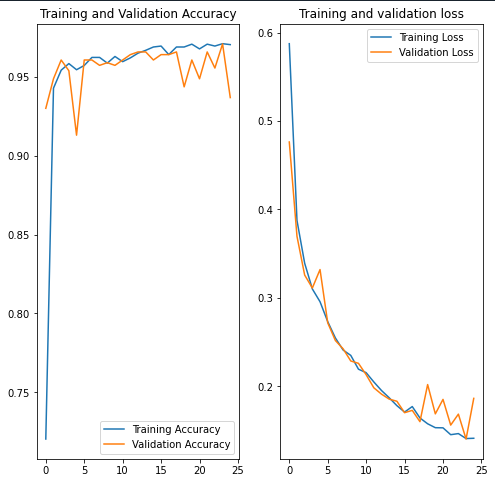
**תדפיס הפעולה**

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

****

## קובץ ה- model\_testing.py:

תפקיד קובץ זה הינו לבחון את המודל.

בקובץ זה יש שתי אפשרויות בחינה, הראשונה היא predict בה המודל אומר על כל תמונה (שלא למד עליה בעבר, כלומר נתקל בה בפעם הראשונה) מה הוא חושב עליה (כלומר, האם העין בתמונה שמוצגת בפניו פתוחה או סגורה).

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיאו evaluate בה יוצג למשתמש את איכות הניחושים של המודל (כלומר, ה- accuracy וה- loss שלו על ה- test).

**הפעולה model\_score(model\_path)**

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיפעולה זו מקבלת את כתובת המודל אותו אנחנו רוצים לבחון, פעולה זו בוחנת את המודל ומחזירה את התוצאה שלו, כלומר את ה- accuracy וה- loss.

כמו בשלב האימון, גם בפעולה זו אני מחלק את ה- test לשני מקבצים שונים, באמצעות

Keras.utils.image\_dataset\_from\_directory

באמצעות פעולה זו ה- test יחולק לי ל- x\_test ול- y\_test, כלומר לתמונות עם ה- Label שלהן (מצב העין).

ה- y\_test (label) ידוע למחשב באמצעות החלוקה לשתי תיקיות נפרדות בתיקיית ה- test

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תוצאות המודל מחושבות על ידי evaluate בצורה הבאה:

**תדפיס הפעולה על שתי התמונות שנמצאות ב- test**

****

**הפעולה model\_predict(model\_path)**

פעולה זו מקבלת את כתובת המודל אותו אנחנו רוצים לבחון. פעולה זו אומרת על כל תמונה הנמצאת ב- test מה המודל אומר שהיא, האם היא סגורה או פתוחה.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

התמונה נקראת באמצעות keras.preprocessing.image.load\_img ונשמרת במשתנה eye\_img



חיזוי המודל מתבצע באמצעות הפעולה predict, והוא נשמר במשתנה predictions



כך באמצעות לולאת for שפועלת ככמות התמונות שנמצאות ב- test, עבור כל תמונה המודל אומר מה הוא חושב ומדפיס את חיזויו לכל תמונה ותמונה.

**תדפיס הפעולה על שתי התמונות שנמצאות ב- test**

תמונה שמכילה ריבוע

התיאור נוצר באופן אוטומטי

## קובץ ה- eyes\_app.py:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטיקובץ זה הינו ממשק המשתמש, כולו עוסק בעבודה עם ממשק המשתמש, באמצעות tkinter.

בממשק המשתמש שבניתי (אשר עליו ארחיב בהמשך בפרק מדריך למשתמש) ניתן לבחור האם לאמן מודל חדש או להשתמש במודל קיים שכבר אומן ומוכן לבחינה. על המשתמש לעקוב אחר ההוראות שרשומות לו במדריך למשתמש בכדי להימנע מתקלות מיותרות.

**הפעולה beginning()**

כאשר רוצים להפעיל את ממשק המשתמש, יש לקרוא לפעולה זו, בה מתבצע הפעלת הממשק, וממנה הכל מתחיל – בפעולה זו למעשה המשתמש נשאל האם הוא רוצה לאמן את המודל או להשתמש במודל קיים, ומשם מנתבת את המשתמש לפעולה הרלוונטית אליו.

**הפעולה split\_getter()**

פעולה זו מופעלת כאשר המשתמש בוחר לאמן מודל חדש. ניתנת למשתמש בפונקציה זו לבחור את ה- train, validation, test split איתו רוצה לאמן את המודל החדש.

עם סיום הפעולה, הפעולה beginning() נקראת מחדש.

**הפעולה model\_selection()**

פעולה זו מופעלת כאשר המשתמש בוחר להשתמש במודל קיים (כמובן שפעולה זו יכולה להיקרא כאשר המשתמש מאמן מודל מחדש, ולאחר מכן בוחר להשתמש במודל אותו אימן)

פעולה זו נותנת למשתמש לבחור מודל – במידה והמודל אינו תקין (אינו קובץ h5) יתבקש המשתמש לשים מודל אמיתי ותקין.

לאחר שהמשתמש בחר מודל, הוא נשאל האם הוא רוצה לאמן על תמונות ספציפיות שהוא רוצה – במידה והוא עונה שהוא רוצה, נפתחת בפניו התיקייה שעליה המודל יבחן (כלומר, תיקיית ה- test) על המשתמש לשים את התמונות של העיניים הפתוחות בתיקייה open\_look ואת התמונות של העיניים הסגורות בתיקייה closed\_look. על המשתמש להקפיד שהוא שם כמות זהה של תמונות בכל אחד מהתיקיות. כאשר המשתמש סיים עליו לסגור את התיקייה.

במידה והמשתמש לא רוצה לאמן על תמונות ספציפיות, יבחן המודל על התמונות שכבר קיימות בתיקיית ה- test.

לאחר מכן הפעולה model\_testing(model\_path) מהקובץ main\_form.py תיקרא, והמודל ייבחן.

ומיד לאחר מכן גם הפונקציה predicting(model\_path) מהקובץ main\_form.py.

**מבנה ממשק המשתמש והמשך הסברים יופיעו בפרק – מדריך למשתמש**

## קובץ ה- main\_form.py:

קובץ זה מחבר בין הקבצים הנדרשים ומקל על עבודתי עם הקובץ של ממשק המשתמש

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי(eyes\_app.py)

למעשה כאשר רוצים להריץ את הפרויקט, יש להריץ את הקובץ הזה, הוא מגשר בין כל הקבצים הרלוונטיים לאימון ובחינת המודל.

**הפעולה הראשית main()**

כאשר מריצים את הקובץ הזה, הפעולה main() מופעלת, והיא קוראת לפעולה beginning() שנמצאת בקובץ eyes\_app.py.

**הפעולה predicting(model\_path)**

פעולה זו קוראת לפעולה model\_predict(model\_path) שנמצאת בקובץ model\_testing.py

**הפעולה model\_testing(model\_path)**

פעולה זו קוראת לפעולה model\_score(model\_path) שנמצאת בקובץ model\_testing.py

**הפעולה learning\_phase(train, val, test)**

פעולה זו קוראת לפעולה model\_training(train, val, test) שנמצאת בקובץ model\_learning.py

למעשה קובץ ה main\_form.py קורא לפעולות בקבצים השונים בפרויקט עצמו, הוא המגשר בין הקבצים.

# מדריך למשתמש

הסבר עבור המעבר בין המסכים ואיך מתנהל ממשק המשתמש:

Main Menu

Train

Test

Model Predict

Model Evaluate

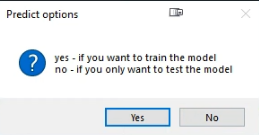
Changing Test Folder

No New Photos

New Photos

Select Split Rates

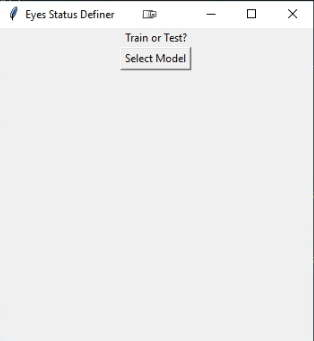
### הסברים על החלונות בממשק המשתמש

החלון הראשון בממשק  
(Main Menu):

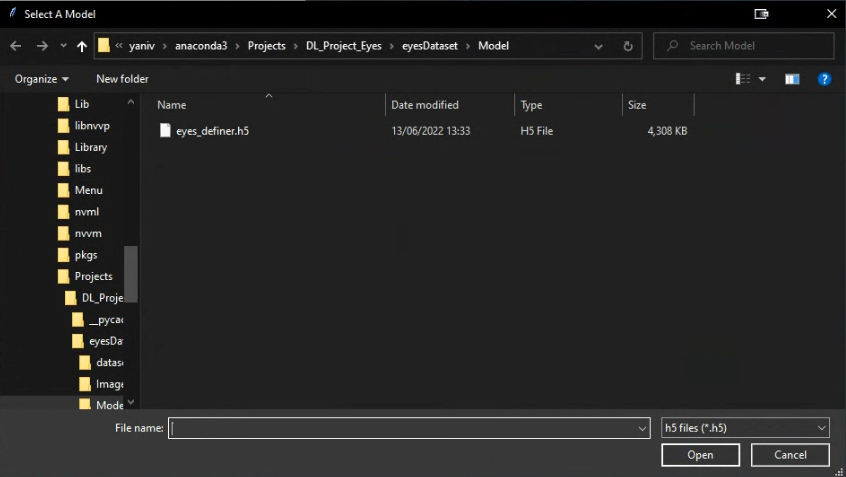
כאשר ממשק המשתמש מופעל, למשתמש קיימת האופציה להחליט האם הוא רוצה לאמן את המודל מחדש וליצור מודל חדש (לבחור "כן") או להשתמש במודל שכבר קיים ולקפוץ ישר לבחינת המודל (לבחור "לא")

**Test**

במידה והמשתמש החליט לבחור מהמודלים הקיימים ולא לאמן את המודל מחדש וליצור מודל חדש, במסך הממשק תופיע לו האופציה לבחור מודל מסוג h5:

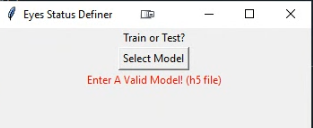


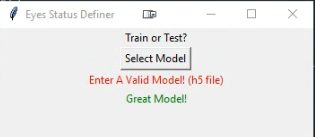
כאשר המשתמש לוחץ על הכפתור נפתח לו החלון בו כל המודלים נמצאים, והוא צריך לבחור מודל

****

במידה והמשתמש בוחר בקובץ לא נכון, או לא בוחר קובץ בכלל, הוא יקבל את ההודעה הבאה:

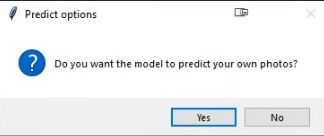
והוא יצטרך לבחור מודל אמיתי (קובץ h5)

****

לאחר שהמשתמש בוחר מודל, מוצגת לו הודעה שהמודל תקין, ולאחר מכן תוצג לו השאלה הבאה:

תמונה שמכילה ריבוע

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**Model Predict**

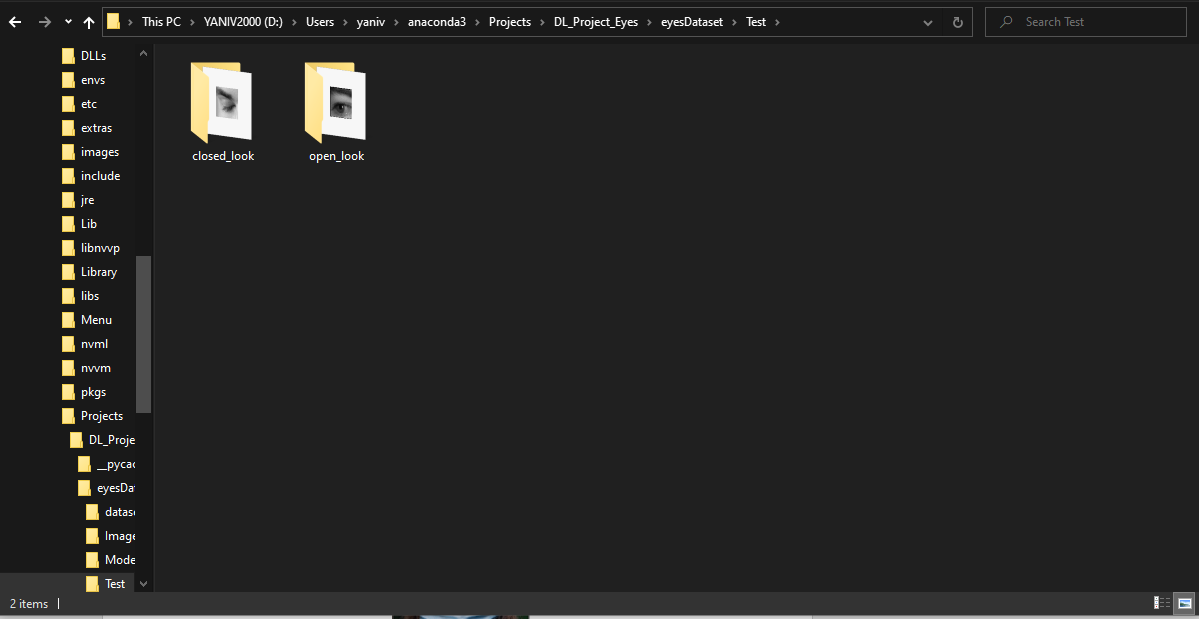
לאחר שנסגור את החלון שנפתח, יוצג ב- kernel את החיזויים של המודל על התמונות שנמצאות ב- test, במקרה זה של ההרצה יש רק שתי תמונות, אחת מכל סוג.

**No New Photos + Model Evaluate**

במידה והמשתמש לא רוצה לבדוק את המודל על תמונות שלו, המודל ייבחן לפי התמונות שנמצאות כרגע ב- test, לפי train/validation/test split בו אומן המודל הקיים (65/15/20)

ולאחר מכן יוצגו תוצאות הבדיקה על ה- test.

האם המשתמש רוצה לבחון תמונות שכבר קיימות בתיקייה של ה- test, או על תמונות שהוא בוחר ורוצה לבחון את מצב העיניים



**New Photos**

במידה והמשתמש רוצה לבדוק את המודל על תמונות שלו, תיפתח התיקייה בה נמצא התמונות עליהן המודל יבחן – המשתמש יצטרך להעביר לתיקייה את התמונות אשר איתן רוצה לבחון את המודל – עליו להקפיד שתהיה את אותה כמות התמונות בשני מצבי העיניים (לדוגמה 7 תמונות של עיניים פקוחות, לכן צריכות להיות גם 7 תמונות של עיניים סגורות), כאשר המשתמש סוגר את התיקייה, המודל יבחן על התמונות האלו ויציג את תוצאות המודל (loss ו- accuracy) ואת חיזוייו. (ראה 2 תמונות מעלה)

מבנה התיקייה Test

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**Train**

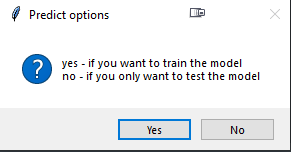
במידה והמשתמש בוחר באופציה של לאמן מודל נפתח בפניו כפתור חדש, כפתור בו הוא בוחר את הsplit rate של הdataset, כמה מה- dataset ילך ל- train, validation, test.

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

**Select Split Rates**

כאשר המשתמש לוחץ על הכפתור, הוא יכול לבחור בין שתי אפשרויות: 65/15/20 או 70/10/20. שתי אפשרויות החילוק המומלצות ביותר. ולאחר מכן יתחיל שלב האימון.



לאחר שהאימון מסתיים, נפתח בחזרה למשתמש החלון אשר הוצג לו בהתחלה, החלון ששואל אותו האם הוא רוצה לאמן מודל חדש, או לבחון מודל קיים. לאחר האימון נוצר מודל חדש, ולכן המשתמש יכול לבחור באופציה של בחינת המודל.

### תפקידי הכפתורים

1.



****

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

תפקיד כפתור זה (Select Mode) הוא לאפשר למשתמש לבחור את המודל איתו רוצה לעבוד. כפתור זה קורה לפעולה model\_selection שבתחילתה מתבצע שלב בחירת המודל.

2.



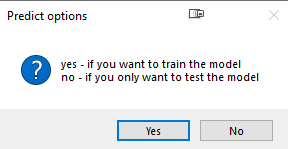
**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

תפקיד כפתור זה (Enter Split Rates) הוא לתת למשתמש לבחור את ה- train, validation, test rate איתו רוצה המשתמש לאמן את המודל. (המשתמש יכול לבחור בשתי אפשרויות)

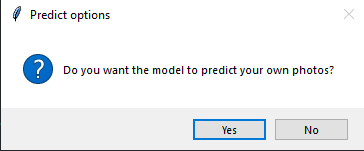
### הודעות למשתמש (alert למיניהם)

1. עם פתיחת האפליקציה מגיעה ההודעה הראשונה בה המשתמש בוחר האם הוא רוצה לאמן את המודל או לעבוד עם מודל קיים



תמונה שמכילה טקסט, פגיעה

התיאור נוצר באופן אוטומטי

1. כאשר המשתמש רוצה לבחון את המודל איתו עובד, הוא מקבל הודעה ששואלת אותו האם הוא רוצה לבחון את המודל עם התמונות שכבר קיימות ב- test לפי ה- split rate שנקבע במודל, או עם תמונות משלו.



1. כאשר בחינת המודל הסתיימה, מוצגת הודעה שמציגה את ה- score של המודל. היא מציגה את ה- test accuracy וה- test\_loss. (כאשר רק 2 תמונות של עיניים (אחת מכל סוג) נמצאות ב- test, אלו הם הנתונים שנמצאים בהודעה.



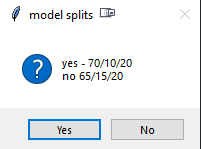
תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

1. כאשר המשתמש בוחר לאמן את המודל, הוא צריך לבחור split rate ל- train, test, validation.

לכן מוצגת למשתמש הודעה שהוא צריך לבחור בין שתי אפשרויות (האפשרויות האופטימליות ביותר). תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

****

# רפלקציה

במהלך עשיית פרויקט הגמר שלי בהתמחות זו למדתי המון דברים בנושא למידת מכונה, ובשפה Python בפרט. לפני עשיית הפרויקט הידע שלי בשפת התכנות הזו היה בסיסי מאוד, אך כיום אני מרגיש שאני יכול לכתוב פעולות יותר מורכבות משיכולתי בעבר, בזכות הפרויקט הבנתי כיצד עובדים עם ספריות, ועם קבצים שלא קשורים בהכרח לשפת התכנות עצמה, בנוסף למדתי כיצד לחפש בצורה יעילה מאוד באינטרנט כאשר אני נתקל בבעיה שאין לי פתרון אליה.

במהלך עבודתי על פרויקט זה התמודדתי עם קשיים רבים בדרך (כמו תמונות שלא מתאימות לאימון, תמונות זהות רבות, ואפילו תוצאות לא טובות אשר לקח זמן רב לסדר) – כתוצאה מכך לוקח אני איתי להמשך דרכי לעולם לא לוותר ולפרוש גם כשקשה, ככל שמתמודדים עם יותר קשיים כך ההצלחה בסוף מתוקה יותר. בנוסף, למדתי על עצמי שכשאני רוצה משהו והוא חשוב לי מספיק , אני אעשה אותו עד שאני אצליח, לא משנה כמה קשה יהיה לי וכמה זמן זה ייקח לי.

במהלך עבודתי על פרויקט זה הכרתי יותר ויותר את תחום ה"למידת מכונה" והבנתי כי לתחום זה יש פוטנציאל עצום, יש שלל אפשרויות לדברים שאפשר לפתח בתחום הזה שיעזרו בתחומים רבים – כמו הפחתת פקקים, מכוניות אוטונומיות, ומזהה וירוסים אוטומטיים.

לו הייתי עושה פרויקט זה מחדש הייתי מתכנן את הזמן שלי בצורה נכונה יותר, הייתי עובד בזמנים מסודרים יותר – לא דוחה את רוב העבודה לסוף ופורס את זמן העבודה שלי לכל אורך השנה. כאשר נתקלתי בבעיה, לא היה לי הרבה זמן לתקן, והכל היה תחת לחץ מרובה.

לו הייתי עובד בצורה יעילה יותר לחץ רב וכאבי ראש היו נחסכים ממני – זמן רב היה נחסך עקב עבודה לא יעילה מפאת לחץ רב מחוסר עמידתי בזמנים.

לסיכום, רוצה אני לומר תודה לבית הספר שלי שבחר לאפשר לנו ללמוד נושא מעניין וחושב זה ולבצע את ההתמחות שלנו בתחום זה. אני מאמין שתחום זה הינו תחום בעל עתיד רב ולכן לעסוק בו כבר בבית הספר זו זכות גדולה בשבילי, אני מאמין שידע זה כוח, וכאשר אני מבין יותר ויותר בנושא חשוב זה – אני מאמין שזה חשוב מאוד.

# ביבליוגרפיה

1. Isik, S & Anagun, Y. (2021). *A DEEP LEARNING BASED SLEEPNESS AND WAKEFULNESS DETECTION FOR DRIVERS*. Retrieved from:

<https://dergipark.org.tr/en/download/article-file/1618184>

1. Kassar, M. (2021). Eyes Images. Retrieved from: <https://www.kaggle.com/datasets/mukhtarkassar/eyes-images?select=eyesDataset>
2. Brownlee, J(2019). *Deep Learning for Computer Vision*. Retrieved from:

<https://books.google.co.il/books?hl=iw&lr=&id=DOamDwAAQBAJ&oi=fnd&pg=PP1&dq=Image+Classification,+Object+Detection++and+Face+Recognition+in+Python&ots=3rxugJJAAP&sig=qhL3lMQi3d6G3poMhjPDNORRYQs&redir_esc=y#v=onepage&q=Image%20Classification%2C%20Object%20Detection%20%20and%20Face%20Recognition%20in%20Python&f=false>

1. freeCodeCamp.org (2019). Tkinter Course - Create Graphic User Interfaces in Python Tutorial. Retrieved from:

<https://www.youtube.com/watch?v=YXPyB4XeYLA>

# Course - Create Graphic User Interfaces in Python Tutorial