

פרויקט גמר 5 יחידות לימוד

התמחות- תכנון ותכנות מערכות

Deep Learning



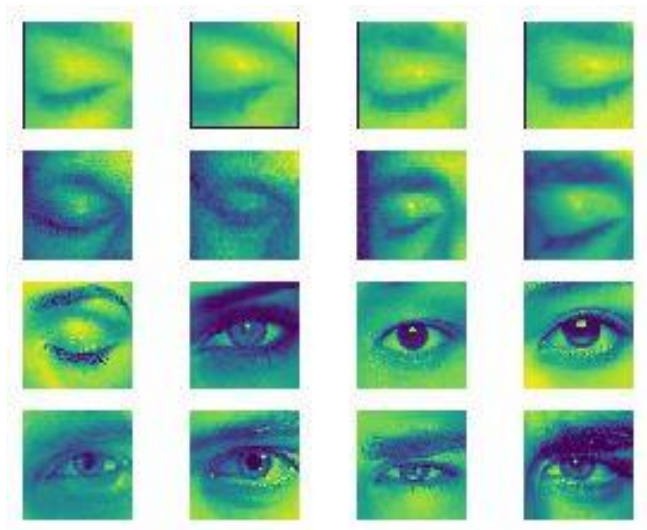
בית הספר: מקיף י"א ראשונים

שם התלמיד: יניב מובשוביץ

ת"ז התלמיד: 325638013

שם המנחה: דינה קראוס

תאריך הבחינה: 5.7.2022



תוכן עניינים

3	מבוא
4	מבנה / ארכיטקטורה
4	איסוף, הכנה, וניתוח הנתונים
6	שלב בניית המודל
6	הסבר על סוגי השכבות השונות ברשת
17	שלב היישום (Software deployment)
17	תיאור והסבר כיצד היישום משתמש במודל
17	תרשים UML של המודלים
18	תיאור הטכנולוגיה שעל פיה מומש ממשק המשתמש
18	תיאור קוד הקולט את ה-DATA שעליו יבוצע החיזוי והתאמתו למבנה נתונים המתאים לחיזוי
20	מדריך למפתח
21	קובץ ה-test_creation.py:
22	קובץ ה-deleting_images.py:
23	קובץ ה-model_learning.py:
28	קובץ ה-model_testing.py:
31	קובץ ה-eyes_app.py:
32	קובץ ה-main_form.py:
33	מדריך למשתמש
34	הסברים על החלונות בממשק המשתמש
38	תפקידי הכפתורים
39	הודעות למשתמש (alert למיניהם)
41	רפלקציה
42	ביבליוגרפיה

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 שצינתי מקודם, או בדרכים אחרות ויצירתיות כמו לזהות האם מישו 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>

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)

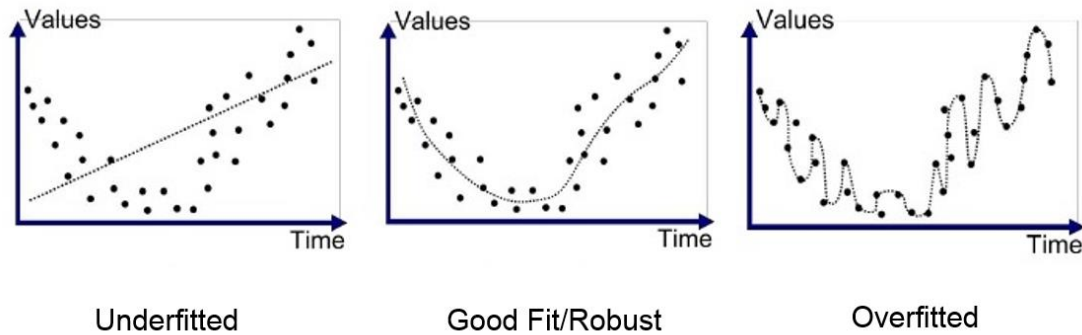
C-R-23893.jpeg (כלומר, תמונה של עין ימנית סגורה)

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.

```
layers.experimental.preprocessing.Rescaling(1. / 255, input_shape=(image_height, image_width, 1))
```

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

Normalization Formula

$$X_{\text{normalized}} = \frac{(X - X_{\text{minimum}})}{(X_{\text{maximum}} - X_{\text{minimum}})}$$

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

שלב בניית המודל

הסבר על סוגי השכבות השונות ברשת

```

model = Sequential([ # the project's model
    layers.experimental.preprocessing.Rescaling(1. / 255, input_shape=(image_height, image_width, 1)),
    layers.Conv2D(32, 1, padding='same', activation='relu'),
    layers.Dropout(DROPOUT),
    layers.MaxPooling2D(),
    layers.Conv2D(128, 1, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 1, padding='same', activation='relu'),
    layers.Dropout(DROPOUT),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(16, activation='softmax'),
    layers.Dense(num_classes)
])

```

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 105, 105, 1)	0
conv2d_3 (Conv2D)	(None, 105, 105, 32)	64
dropout_2 (Dropout)	(None, 105, 105, 32)	0
max_pooling2d_3 (MaxPooling 2D)	(None, 52, 52, 32)	0
conv2d_4 (Conv2D)	(None, 52, 52, 128)	4224
max_pooling2d_4 (MaxPooling 2D)	(None, 26, 26, 128)	0
conv2d_5 (Conv2D)	(None, 26, 26, 32)	4128
dropout_3 (Dropout)	(None, 26, 26, 32)	0
max_pooling2d_5 (MaxPooling 2D)	(None, 13, 13, 32)	0
flatten_1 (Flatten)	(None, 5408)	0
dense_2 (Dense)	(None, 16)	86544
dense_3 (Dense)	(None, 2)	34
=====		
Total params: 94,994		
Trainable params: 94,994		
Non-trainable params: 0		

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

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

ה- activation function שבחיתי 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

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

שכבה 2 – Convolutional layer

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

שכבה 3 – Convolutional layer

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

שכבה 4 – Dense layer

Neurons amount	16
Activation function	softmax

נתונים נוספים בנוגע למודל הנוכחי עליהם אסביר בהמשך:

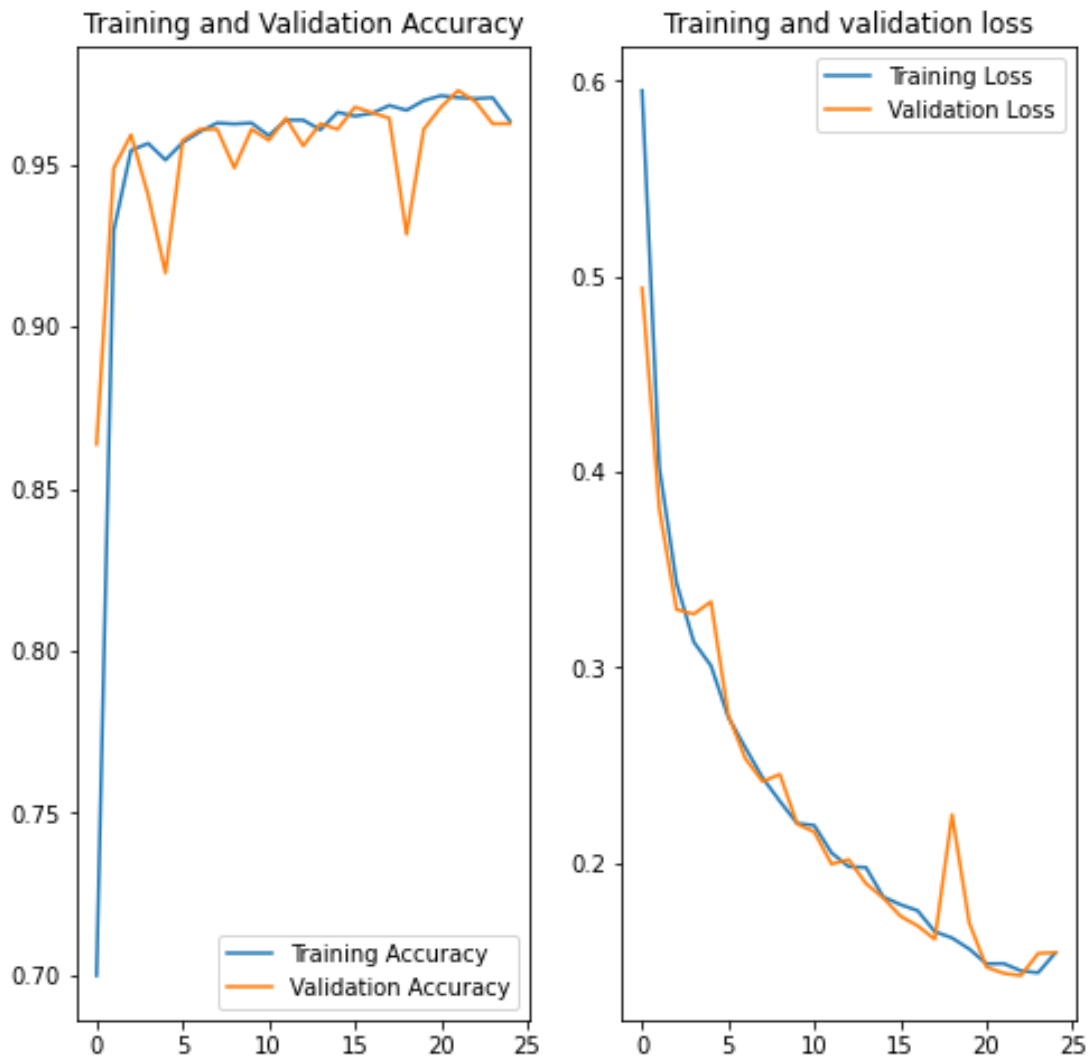
batch_size = 64 # the size of each images batch

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

loss: משתנה תלוי בגרף הימני

accuracy: משתנה תלוי בגרף השמאלי

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 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:

```
model = Sequential([ # the project's model
    layers.experimental.preprocessing.Rescaling(1. / 255, input_shape=(image_height, image_width, 1)),
    layers.Conv2D(16, 1, padding='same', activation='relu'),
    layers.Dropout(DROPOUT),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 1, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(16, 1, padding='same', activation='relu'),
    layers.Dropout(DROPOUT),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(1, activation='sigmoid'),
    layers.Dense(num_classes)
])
```

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 105, 105, 1)	0
conv2d (Conv2D)	(None, 105, 105, 16)	32
dropout (Dropout)	(None, 105, 105, 16)	0
max_pooling2d (MaxPooling2D)	(None, 52, 52, 16)	0
conv2d_1 (Conv2D)	(None, 52, 52, 32)	544
max_pooling2d_1 (MaxPooling2D)	(None, 26, 26, 32)	0
conv2d_2 (Conv2D)	(None, 26, 26, 16)	528
dropout_1 (Dropout)	(None, 26, 26, 16)	0
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 16)	0
flatten (Flatten)	(None, 2704)	0
dense (Dense)	(None, 1)	2705
dense_1 (Dense)	(None, 2)	4
=====		
Total params: 3,813		
Trainable params: 3,813		
Non-trainable params: 0		

שכבה 1 – Convolutional layer

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

שכבה 2 – Convolutional layer

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

שכבה 3 – Convolutional layer

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

שכבה 4 – Dense layer

Neurons amount	1
Activation function	sigmoid

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



הניסיון השני שבחרתי להציג בספר הפרויקט:

```

model = Sequential([ # the project's model
    layers.experimental.preprocessing.Rescaling(1. / 255, input_shape=(image_height, image_width, 1)),
    layers.Conv2D(64, 1, padding='same', activation='leaky_relu'),
    layers.Dropout(DROPOUT),
    layers.MaxPooling2D(),
    layers.Conv2D(128, 1, padding='same', activation='leaky_relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 1, padding='same', activation='leaky_relu'),
    layers.Dropout(DROPOUT),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(32, activation='sigmoid'),
    layers.Dense(num_classes)
])

```

Model: "sequential"

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 105, 105, 1)	0
conv2d (Conv2D)	(None, 105, 105, 64)	128
dropout (Dropout)	(None, 105, 105, 64)	0
max_pooling2d (MaxPooling2D)	(None, 52, 52, 64)	0
conv2d_1 (Conv2D)	(None, 52, 52, 128)	8320
max_pooling2d_1 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_2 (Conv2D)	(None, 26, 26, 64)	8256
dropout_1 (Dropout)	(None, 26, 26, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 64)	0
flatten (Flatten)	(None, 10816)	0
dense (Dense)	(None, 32)	346144
dense_1 (Dense)	(None, 2)	66
=====		
Total params: 362,914		
Trainable params: 362,914		
Non-trainable params: 0		

שכבה 1 – Convolutional layer

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

שכבה 2 – Convolutional layer

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

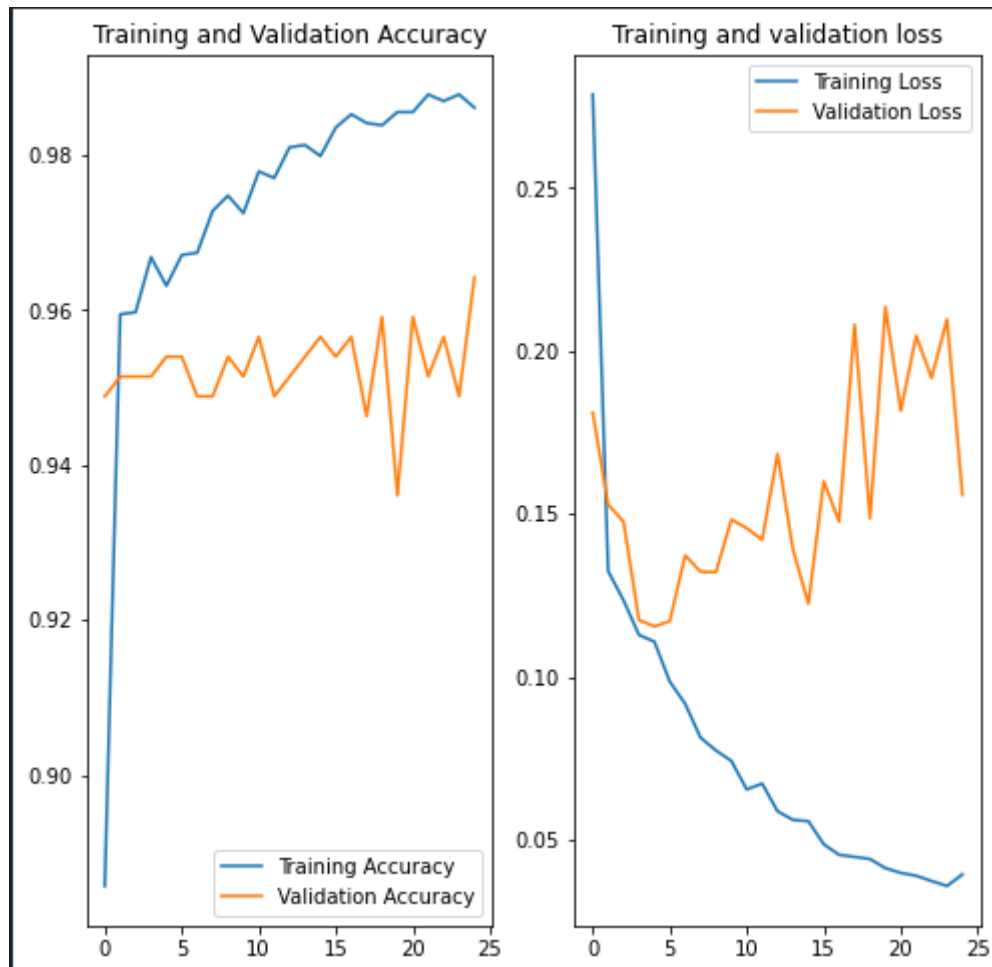
שכבה 3 – Convolutional layer

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

שכבה 4 – Dense layer

Neurons amount	32
Activation function	sigmoid

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

```
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

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, יכול 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.

```
y_true = [1, 2]
y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
```

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, ושתייהן נכונות.

```
y_true = [1, 1]
y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
```

(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).

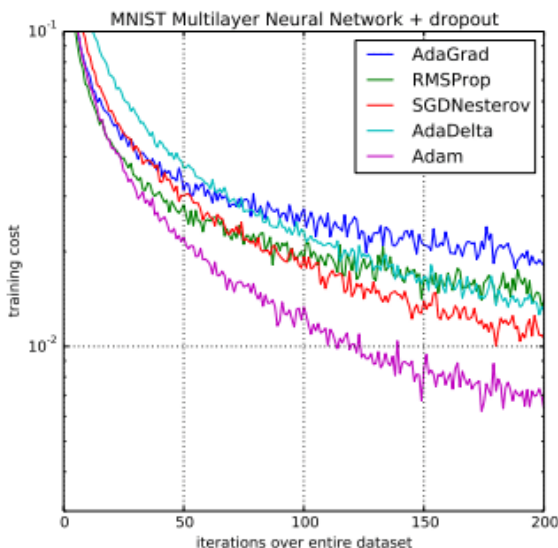
תיעוד והסבר של ייעול ההתכנסות (Optimization)

```
optimizer='Adam'
```

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

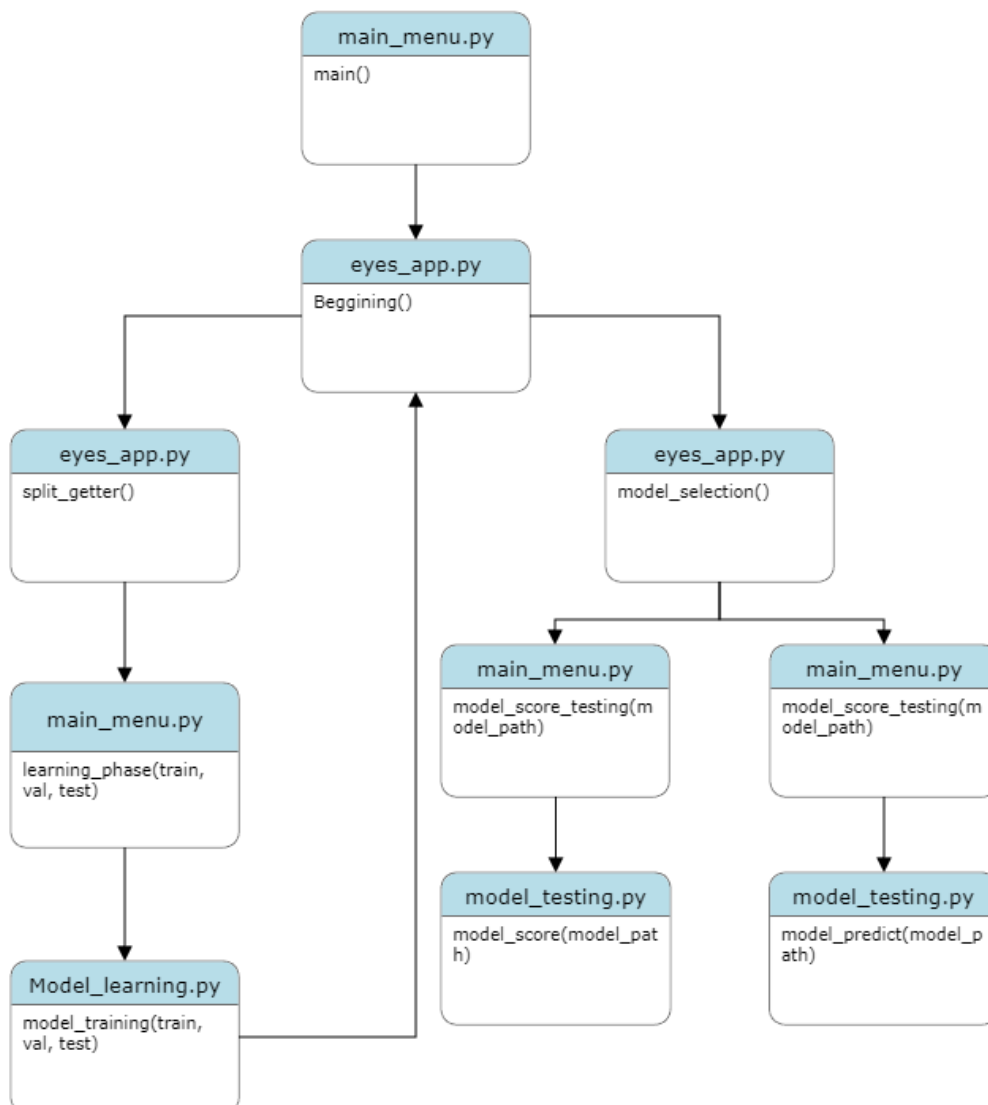
כאשר מתחילים את העבודה עם היישום, נשאל המשתמש האם הוא רוצה לאמן מודל חדש או להשתמש במודל ישן ולבחון אותו.

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_predict, model_score 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



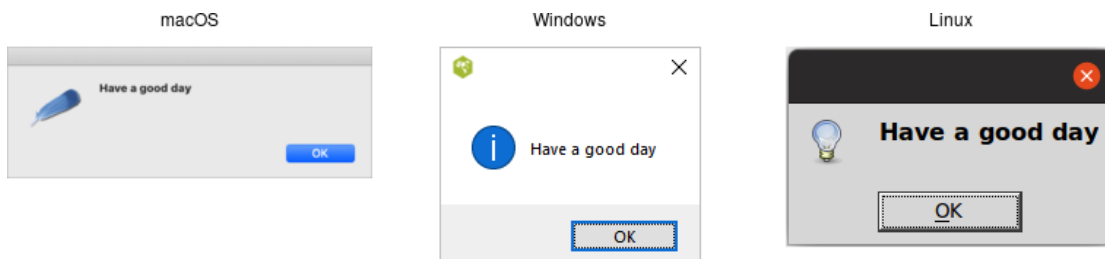
תיאור הטכנולוגיה שעל פיה מומש ממשק המשתמש

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!

```
from tkinter import *
```



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 התמונות 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.

בשלב ה- score, כלומר evaluate ה- DATA מותאם לחיזוי בצורה הבאה:

```
test_ds = tf.keras.utils.image_dataset_from_directory(
    test_dir,
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='grayscale',
    batch_size=489,
    image_size=(image_height, image_width),
    shuffle=True,
    seed=2,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

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).

פרויקט למידת מכונה / זיהוי עין פקוחה או עצומה, יניב מובשוביץ

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

```
eye_img = keras.preprocessing.image.load_img(eyes_path, target_size=(105, 105), color_mode='grayscale')
```

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.

<u>שם הקובץ</u>	<u>תפקידו</u>
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-split train_validation_test 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.

```
import pathlib
import os
import random
import shutil
```

הפעולה make_test_dir()

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

```
def make_test_folder():
    """
    This function makes a test folder if there isn't one already.
    """
```

הפעולה test_maker(image_count, test_split = 0.2)

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

```
def test_maker(image_count, test_split=0.2):
    """
    This function moves photos from the whole images directory into the test directory using the test split.
    :param image_count: whole dataset images ammount.
    :param test_split: the train test split.
    :return:
    """
```

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.

```
def back_from_test(data_dir):
    """
    This function brings back the photos from the test directory to the images directory
    :param data_dir: the dataset you want to move back to test
    """
```

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).

```
import imagehash
import os
import pathlib
from PIL import Image
```

הפעולה deleting_small_photos(data_dir)

פעולה 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.

```
def deleting_small_photos(data_dir):
    """
    This function deletes photos if their size is too small
    :param data_dir: the directory you want to delete photos from.
    """
```

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.

```
def similarity_check(photo1, photo2):
    """
    This function checks between two photos how similar their hashes are.
    if the are similar enough, True would be returned.
    I made this function in order to prevent overfitting.
    :param photo1: first photo
    :param photo2: second photo
    :return: True if the photos are similar, False if they aren't
    """
```

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)

```
def delete_similar_images(data_dir):
    """
    This function deletes a photo if there are another photo similar to it.
    """
```

קובץ ה-model learning.py

קובץ זה הינו הקובץ העיקרי בפרויקט. בו מתבצע אימון המודל ויצירתו!

```
import matplotlib.pyplot as plt
import numpy as np
import pathlib
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import test_creation
import os
```

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.

```
def creating_test_dataset(image_count, test_split):
    """
    this function calls the function that moves photos into the test directory
    Parameters
    -----
    :param image_count: whole dataset images ammount.
    :param test_split: the train test split.

    Returns
    -----
    None.

    """
```

הפעולה model_training(train, val, test)

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

```
def model_training(train, val, test):
    """
    this function trains the model, and saves it.
    Parameters
    -----
    :param train: train split
    :param val: validation split
    :param test: test split

    Returns
    -----
    None.

    """
```

```
batch_size = 64 # the size of each images batch
DROPOUT = 0.2 # model dropout
image_height = 105 # images height
image_width = 105 # images width
NB_EPOCH = 25 # amount of times the model will train each image
test_split = test/100 # how much TRAIN & VALIDATION is reserved for TEST
```

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 חלקי 100)

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    label_mode='binary',
    color_mode='grayscale',
    validation_split=val/100,
    subset="training",
    seed=2,
    image_size=(image_height, image_width),
    batch_size=batch_size
)

validation_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    label_mode='binary',
    color_mode='grayscale',
    validation_split=val/100,
    subset="validation",
    seed=2,
    image_size=(image_height, image_width),
    batch_size=batch_size
)
```

החילוק ל-train, validation מתבצע על ידי

dataset.keras.preprocessing.image_dataset_from_directory. פעולה זו מקבלת את ה-creating_test_dataset (ומחלקת המלא (לאחר שהועברו תמונות לתיקיית test על ידי הפעולה

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 יודא את אמינותו (validation)

```
history = model.fit(train_ds, validation_data=validation_ds, epochs=NB_EPOCH)
```

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.

```
accuracy = history.history['accuracy'] # each epoch train accuracy
val_acc = history.history['val_accuracy'] # each epoch validation accuracy

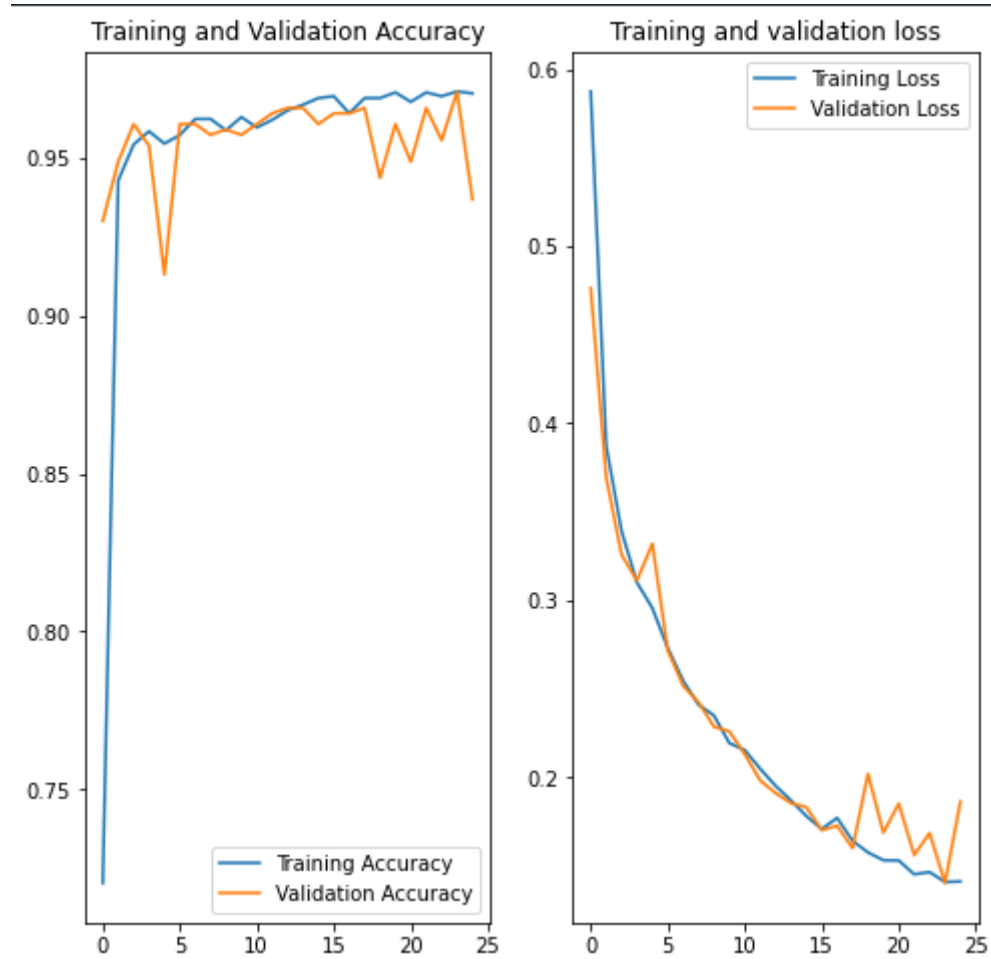
loss = history.history['loss'] # each epoch train loss
val_loss = history.history['val_loss'] # each epoch validation loss
```

תדפיס הפעולה

Model: "sequential"		
Layer (type)	Output Shape	Param #
=====		
rescaling (Rescaling)	(None, 105, 105, 1)	0
conv2d (Conv2D)	(None, 105, 105, 64)	128
dropout (Dropout)	(None, 105, 105, 64)	0
max_pooling2d (MaxPooling2D)	(None, 52, 52, 64)	0
conv2d_1 (Conv2D)	(None, 52, 52, 128)	8320
max_pooling2d_1 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_2 (Conv2D)	(None, 26, 26, 64)	8256
dropout_1 (Dropout)	(None, 26, 26, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 13, 13, 64)	0
flatten (Flatten)	(None, 10816)	0
dense (Dense)	(None, 32)	346144
dense_1 (Dense)	(None, 2)	66
=====		
Total params: 362,914		
Trainable params: 362,914		
Non-trainable params: 0		

פרויקט למידת מכונה / זיהוי עין פקוחה או עצומה, יניב מובשוביץ

```
Epoch 1/25
53/53 [=====] - 31s 578ms/step - loss: 0.5874 - accuracy: 0.7203 - val_loss: 0.4763 - val_accuracy: 0.9302
Epoch 2/25
53/53 [=====] - 29s 547ms/step - loss: 0.3876 - accuracy: 0.9429 - val_loss: 0.3697 - val_accuracy: 0.9489
Epoch 3/25
53/53 [=====] - 30s 557ms/step - loss: 0.3392 - accuracy: 0.9543 - val_loss: 0.3258 - val_accuracy: 0.9608
Epoch 4/25
53/53 [=====] - 32s 606ms/step - loss: 0.3101 - accuracy: 0.9585 - val_loss: 0.3112 - val_accuracy: 0.9540
Epoch 5/25
53/53 [=====] - 30s 572ms/step - loss: 0.2957 - accuracy: 0.9546 - val_loss: 0.3321 - val_accuracy: 0.9131
Epoch 6/25
53/53 [=====] - 28s 520ms/step - loss: 0.2732 - accuracy: 0.9573 - val_loss: 0.2719 - val_accuracy: 0.9608
Epoch 7/25
53/53 [=====] - 29s 537ms/step - loss: 0.2547 - accuracy: 0.9625 - val_loss: 0.2518 - val_accuracy: 0.9608
Epoch 8/25
53/53 [=====] - 28s 520ms/step - loss: 0.2411 - accuracy: 0.9625 - val_loss: 0.2425 - val_accuracy: 0.9574
Epoch 9/25
53/53 [=====] - 28s 526ms/step - loss: 0.2350 - accuracy: 0.9588 - val_loss: 0.2287 - val_accuracy: 0.9591
Epoch 10/25
53/53 [=====] - 28s 520ms/step - loss: 0.2194 - accuracy: 0.9631 - val_loss: 0.2261 - val_accuracy: 0.9574
Epoch 11/25
53/53 [=====] - 29s 545ms/step - loss: 0.2155 - accuracy: 0.9597 - val_loss: 0.2132 - val_accuracy: 0.9608
Epoch 12/25
53/53 [=====] - 29s 536ms/step - loss: 0.2050 - accuracy: 0.9622 - val_loss: 0.1984 - val_accuracy: 0.9642
Epoch 13/25
53/53 [=====] - 29s 550ms/step - loss: 0.1953 - accuracy: 0.9652 - val_loss: 0.1912 - val_accuracy: 0.9659
Epoch 14/25
53/53 [=====] - 29s 548ms/step - loss: 0.1871 - accuracy: 0.9670 - val_loss: 0.1856 - val_accuracy: 0.9659
Epoch 15/25
53/53 [=====] - 30s 565ms/step - loss: 0.1782 - accuracy: 0.9691 - val_loss: 0.1832 - val_accuracy: 0.9608
Epoch 16/25
53/53 [=====] - 28s 525ms/step - loss: 0.1710 - accuracy: 0.9697 - val_loss: 0.1703 - val_accuracy: 0.9642
Epoch 17/25
53/53 [=====] - 30s 563ms/step - loss: 0.1772 - accuracy: 0.9643 - val_loss: 0.1729 - val_accuracy: 0.9642
Epoch 18/25
53/53 [=====] - 30s 563ms/step - loss: 0.1643 - accuracy: 0.9691 - val_loss: 0.1601 - val_accuracy: 0.9659
Epoch 19/25
53/53 [=====] - 30s 556ms/step - loss: 0.1577 - accuracy: 0.9691 - val_loss: 0.2020 - val_accuracy: 0.9438
Epoch 20/25
53/53 [=====] - 29s 550ms/step - loss: 0.1533 - accuracy: 0.9709 - val_loss: 0.1689 - val_accuracy: 0.9608
Epoch 21/25
53/53 [=====] - 30s 569ms/step - loss: 0.1531 - accuracy: 0.9679 - val_loss: 0.1853 - val_accuracy: 0.9489
Epoch 22/25
53/53 [=====] - 29s 537ms/step - loss: 0.1454 - accuracy: 0.9709 - val_loss: 0.1563 - val_accuracy: 0.9659
Epoch 23/25
53/53 [=====] - 28s 519ms/step - loss: 0.1466 - accuracy: 0.9697 - val_loss: 0.1686 - val_accuracy: 0.9557
Epoch 24/25
53/53 [=====] - 28s 531ms/step - loss: 0.1410 - accuracy: 0.9712 - val_loss: 0.1405 - val_accuracy: 0.9710
Epoch 25/25
53/53 [=====] - 28s 529ms/step - loss: 0.1413 - accuracy: 0.9706 - val_loss: 0.1865 - val_accuracy: 0.9370
```



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).

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

```
import matplotlib.pyplot as plt
import numpy as np

import pathlib
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import test_creation
import os
```

הפעולה(model_path)model_score

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

```
def model_score(model_path):
    """
    this function returns the model's score on the test dataset.

    Parameters
    -----
    :param model_path: the path of the model we want to test

    Returns
    -----
    score: test accuracy and loss

    """
```

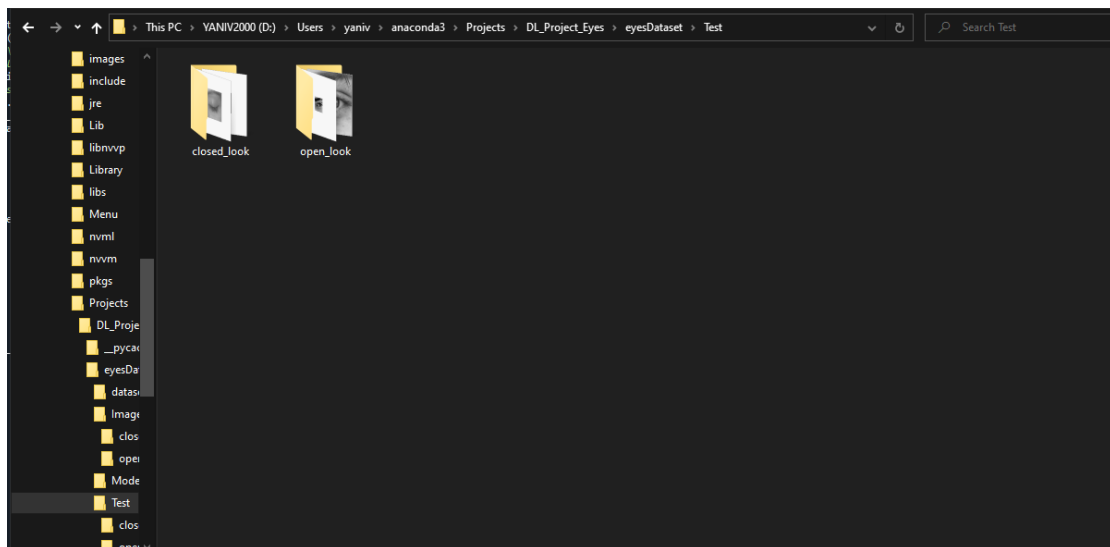
כמו בשלב האימון, גם בפעולה זו אני מחלק את ה-test לשני מקבצים שונים, באמצעות

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

```
test_ds = tf.keras.utils.image_dataset_from_directory(
    test_dir,
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='grayscale',
    batch_size=batch_size,
    image_size=(image_height, image_width),
    shuffle=True,
    seed=2,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False
)
(x_test, y_test) = test_ds # test dataset with labels
test_dataset = test_ds.from_tensor_slices((x_test, y_test))
```



```
score = model.evaluate(
    x=test_ds,
    batch_size=None,
    verbose="auto",
    sample_weight=None,
    steps=None,
    callbacks=None,
    max_queue_size=10,
    workers=1,
    use_multiprocessing=False,
    return_dict=False,
)
```

תוצאות המודל מחושבות על ידי evaluate בצורה הבאה:

תדפיס הפעולה על שתי התמונות שנמצאות ב- test

```
[0.014607364311814308, 1.0]
Test loss: 0.014607364311814308
Test accuracy: 1.0
```

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.

```
def model_predict(model_path):  
    """  
    this function makes the model predict each image whether it is opened or closed  
  
    Parameters  
    -----  
    :param model_path: the path of the model we want to test  
  
    Returns  
    -----  
    None.  
  
    """
```

התמונה נקראת באמצעות keras.preprocessing.image.load_img ומשמרת במשתנה eye_img

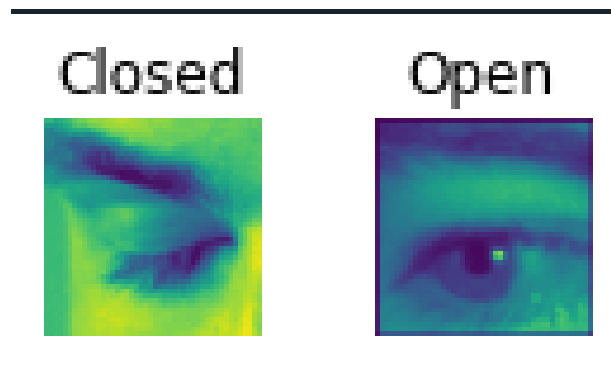
```
eye_img = keras.preprocessing.image.load_img(eyes_path, target_size=(105, 105), color_mode='grayscale')
```

חיזוי המודל מתבצע באמצעות הפעולה predict, והוא נשמר במשתנה predictions

```
predictions = model.predict(img_arr) # model prediction to a certain image in the dataset
```

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

תדפיס הפעולה על שתי התמונות שנמצאות ב- test



קובץ ה- eyes app.py :

קובץ זה הינו ממשק המשתמש, כולו עוסק בעבודה עם ממשק המשתמש, באמצעות tkinter.

```
from tkinter import *
from tkinter import filedialog, messagebox
import pathlib
import time
from PIL import ImageTk, Image
from main_form import learning_phase, predicting, model_score
```

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 התיקיה שעליה המודל יבחן (כלומר, תיקיית ה-test) על המשתמש לשים את התמונות של העיניים הפתוחות בתיקיה open_look ואת התמונות של העיניים הסגורות בתיקיה closed_look. על המשתמש להקפיד שהוא שם כמות זהה של תמונות בכל אחד מהתיקיות. כאשר המשתמש סיים עליו לסגור את התיקיה.

במידה והמשתמש לא רוצה לאמן על תמונות ספציפיות, יבחן המודל על התמונות שכבר קיימות בתיקית 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

```
import model_testing  
import model_learning  
import eyes_app
```

(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

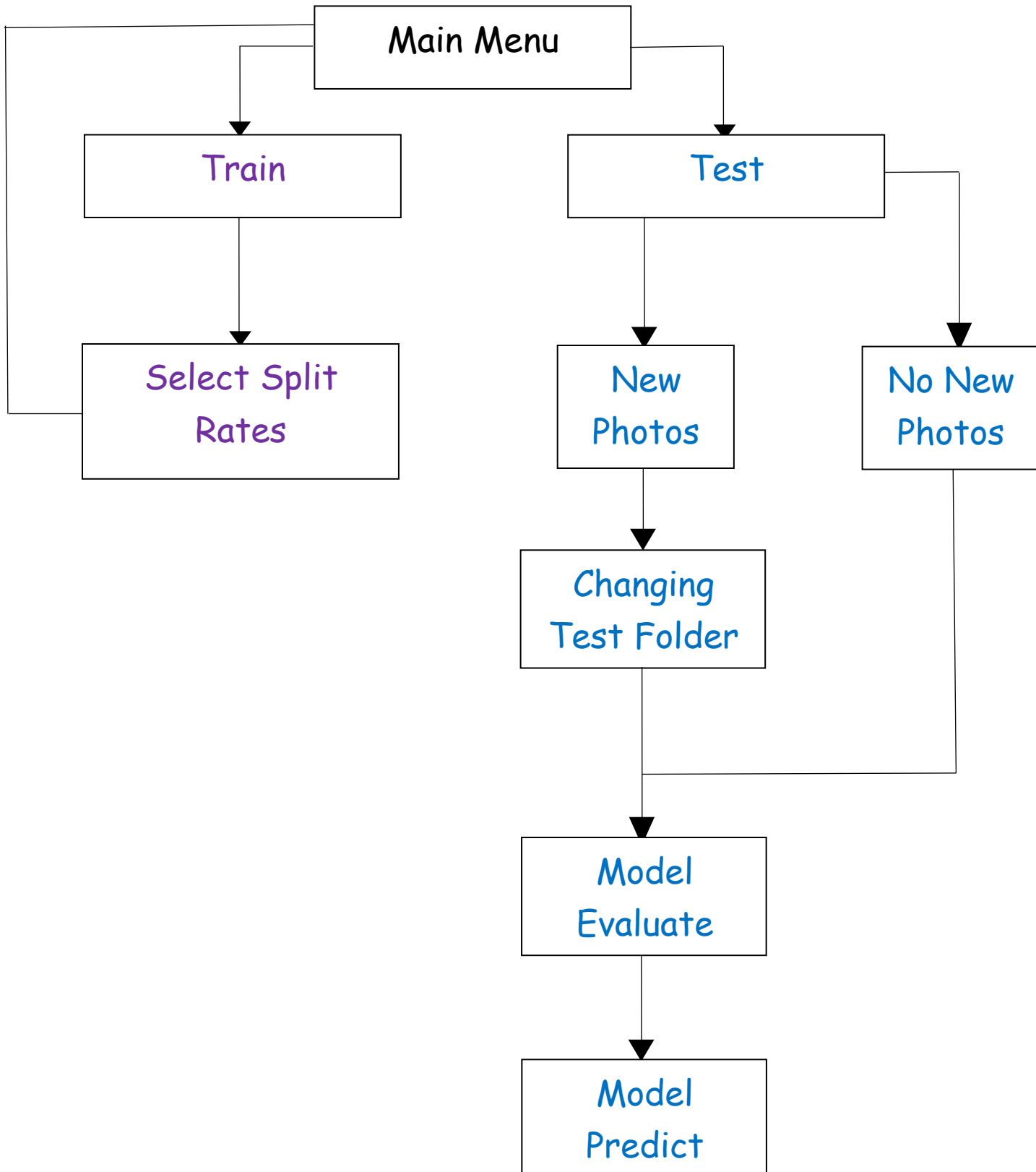
הפעולה learning_phase(train, val, test)

פעולה זו קוראת לפעולה model_training(train, val, test) שנמצאת בקובץ model_learning.py

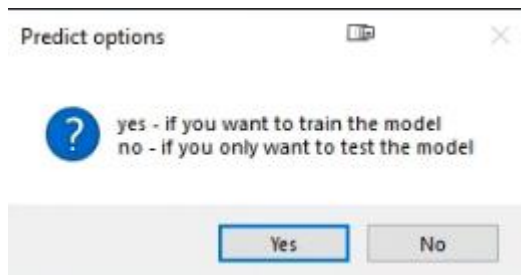
למעשה קובץ ה main_form.py קורא לפעולות בקבצים השונים בפרויקט עצמו, הוא המגשר בין הקבצים.

מדריך למשתמש

הסבר עבור המעבר בין המסכים ואיך מתנהל ממשק המשתמש :



הסברים על החלונות בממשק המשתמש

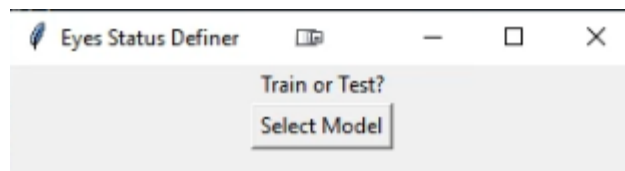


החלון הראשון בממשק : (Main Menu)

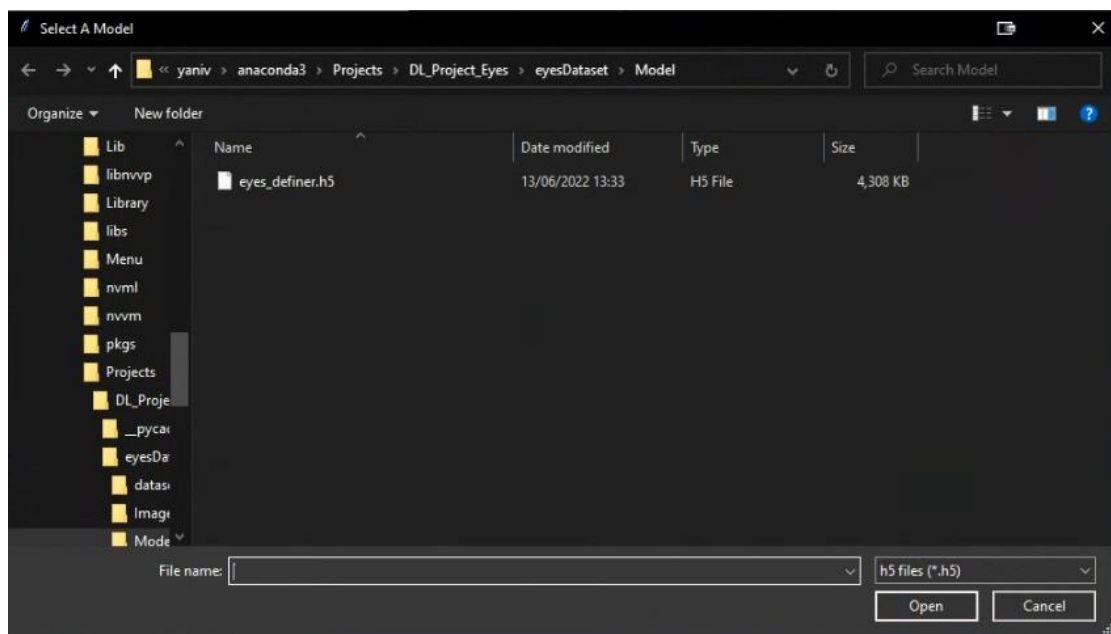
כאשר ממשק המשתמש מופעל, למשתמש קיימת האופציה להחליט האם הוא רוצה לאמן את המודל מחדש וליצור מודל חדש (לבחור "כן") או להשתמש במודל שכבר קיים ולקפוץ ישר לבחינת המודל (לבחור "לא")

Test

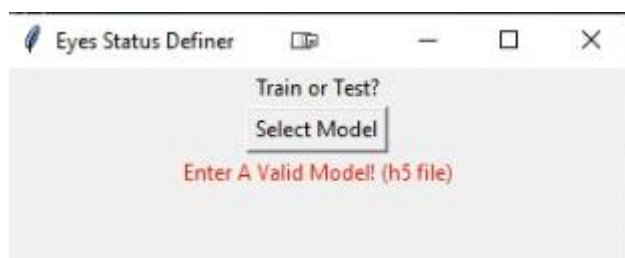
במידה והמשתמש החליט לבחור מהמודלים הקיימים ולא לאמן את המודל מחדש וליצור מודל חדש, במסך הממשק תופיע לו האופציה לבחור מודל מסוג h5 :

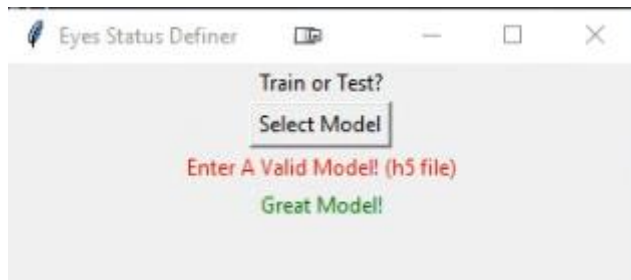


כאשר המשתמש לוחץ על הכפתור נפתח לו החלון בו כל המודלים נמצאים, והוא צריך לבחור מודל

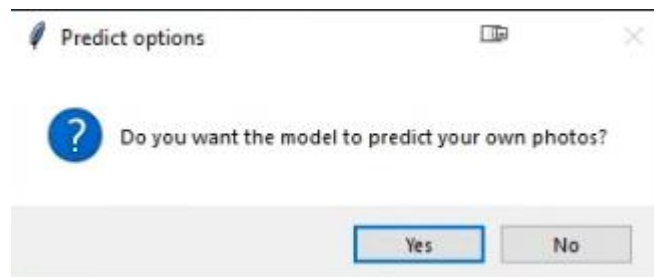


במידה והמשתמש בוחר בקובץ לא נכון, או לא בוחר קובץ בכלל, הוא יקבל את ההודעה הבאה :
והוא יצטרך לבחור מודל אמיתי (קובץ h5)





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

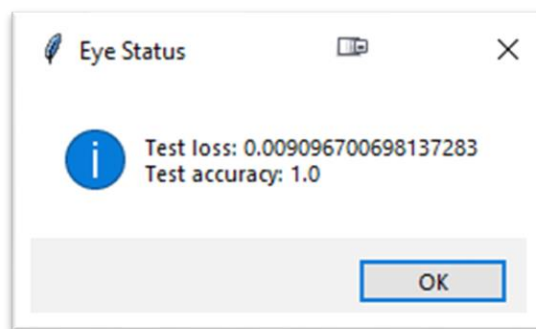


האם המשתמש רוצה לבחון תמונות שכבר קיימות בתיקייה של ה-test, או על תמונות שהוא בוחר ורוצה לבחון את מצב העיניים

No New Photos + Model Evaluate

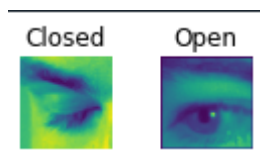
במידה והמשתמש לא רוצה לבדוק את המודל על תמונות שלו, המודל ייבחן לפי התמונות שנמצאות כרגע ב-test, לפי train/validation/test split בו אומן המודל הקיים (65/15/20)

ולאחר מכן יוצגו תוצאות הבדיקה על ה-test.



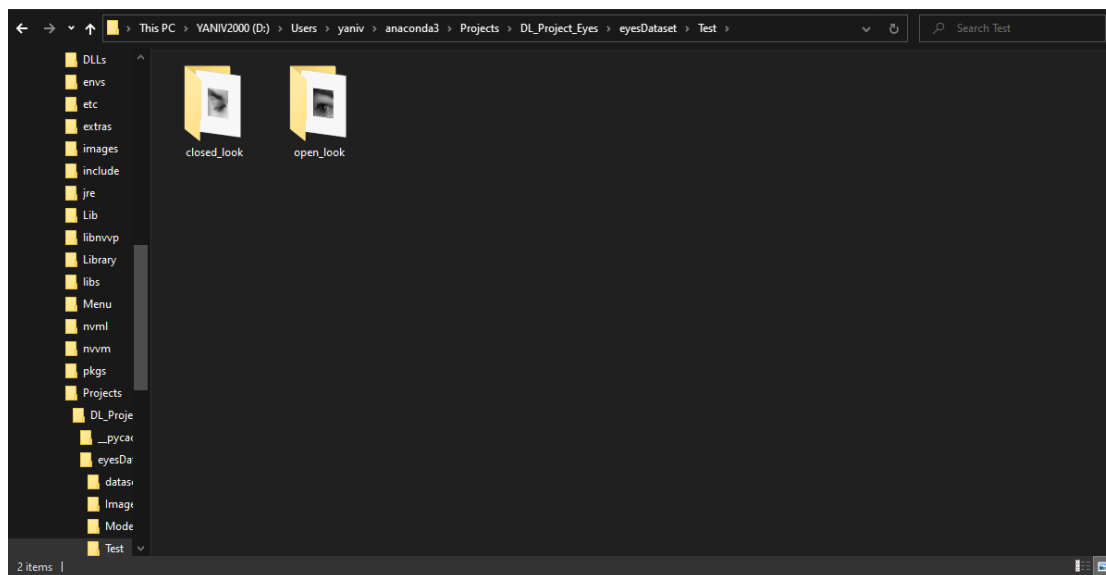
Model Predict

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.

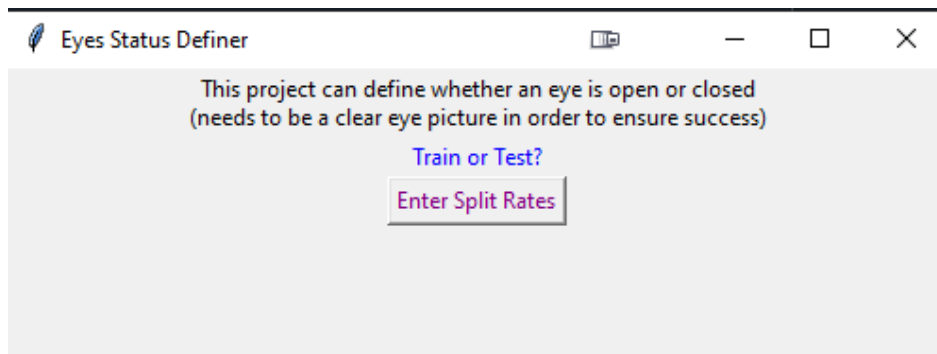


New Photos

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 ויצג את תוצאות המודל (loss ו- accuracy) ואת חיזוייו. (ראה 2 תמונות מעלה)

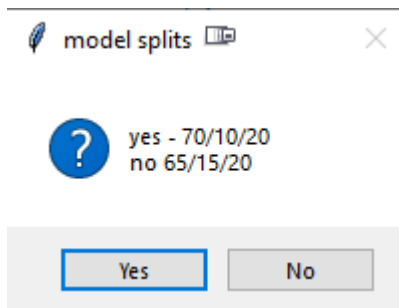


מבנה התיקייה Test



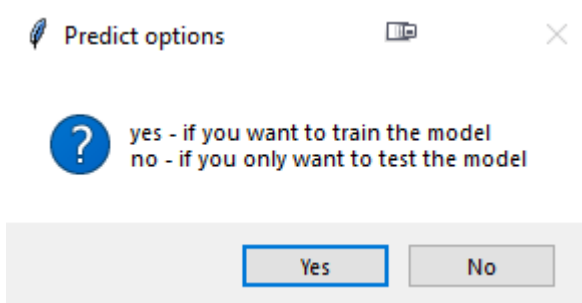
Train

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



Select Split Rates

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

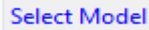


לאחר שהאימון מסתיים, נפתח בחזרה למשתמש החלון אשר הוצג לו בהתחלה, החלון ששואל אותו האם הוא רוצה לאמן מודל חדש, או לבחון מודל קיים. לאחר האימון נוצר מודל חדש, ולכן המשתמש יכול לבחור באופציה של בחינת המודל.

תפקידי הכפתורים

1.

```
model = Button(root, text="Select Model", command=model_selection, fg="blue")
```



```
def model_selection():
    models_dir = pathlib.Path(r"D:\Users\yaniv\anaconda3\Projects\DL_Project_Eyes\eyesDataset\Model")
    root.filename = filedialog.askopenfilename(initialdir=models_dir,
                                              title="Select A Model",
                                              filetypes=(("h5 files", "*.h5"), ("all files", "*.*")))
```

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.

```
train = Button(root, text="Enter Split Rates", command=split_getter, fg="purple")
```

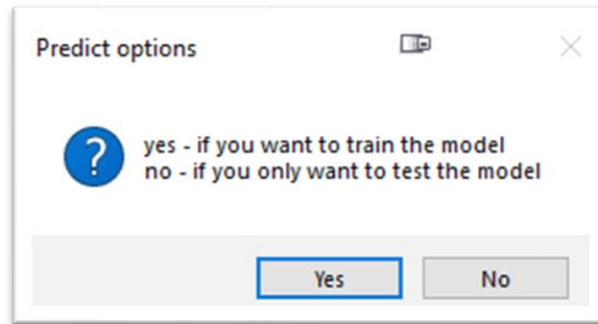


```
def split_getter():
    split = messagebox.askquestion("model splits", "yes - 70/10/20 \nno 65/15/20")
    if train_or_test == "no":
        learning_phase(65,15,20)
    elif train_or_test == "yes":
        learning_phase(70,10,20)
    beggining()
```

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)

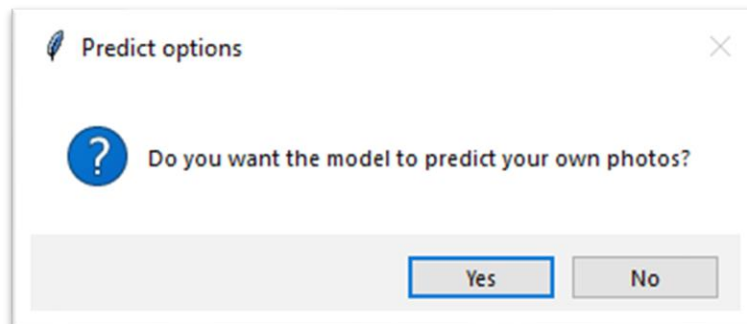
1. עם פתיחת האפליקציה מגיעה ההודעה הראשונה בה המשתמש בוחר האם הוא רוצה לאמן את המודל או לעבוד עם מודל קיים



```
train_or_test = messagebox.askquestion("Predict options",  
                                       "yes - if you want to train the model \\\nno - if you only want to test the model")
```

```
if train_or_test == "no":  
    model = Button(root, text="Select Model", command=model_selection, fg="blue")  
    model.pack()  
elif train_or_test == "yes":  
    train = Button(root, text="Enter Split Rates", command=split_getter, fg="purple")  
    train.pack()
```

2. 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.

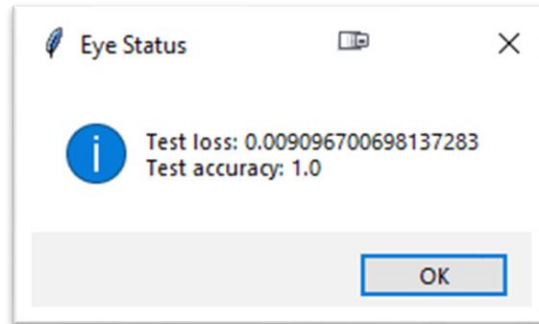


```
local_test = messagebox.askquestion("Predict options", "Do you want the model to predict your own photos?")
```

```
if local_test == "yes":  
    label = Label(root, text="move the photos you want to test to the test directory with the right label")  
    root.filename = filedialog.askopenfilename(initialdir=r"D:\\Users\\yaniv\\anaconda3\\Projects\\DL_Project_Eyes\\eyesDataset\\Test")
```

3. 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).

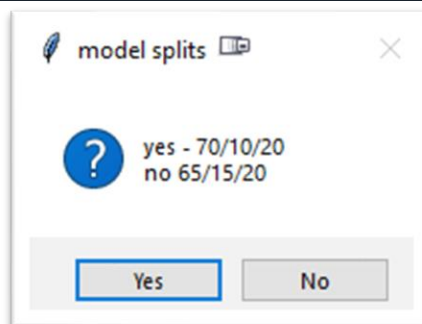
```
messagebox.showinfo("Eye Status", loss_accuracy)
```



4. 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).

```
split = messagebox.askquestion("model splits", "yes - 70/10/20 \nno 65/15/20")
if train_or_test == "no":
    learning_phase(65,15,20)
elif train_or_test == "yes":
    learning_phase(70,10,20)
```



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, מכונות 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 וחשוב 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>
2. Kassir, M. (2021). Eyes Images. Retrieved from:
<https://www.kaggle.com/datasets/mukhtarkassar/eyes-images?select=eyesDataset>
3. 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=qhL3lMOi3d6G3poMhjPDNORRYQs&redir_esc=y#v=onepage&q=Image%20Classification%2C%20Object%20Detection%20%20and%20Face%20Recognition%20in%20Python&f=false
4. freeCodeCamp.org (2019). Tkinter Course - Create Graphic User Interfaces in Python Tutorial. Retrieved from:
<https://www.youtube.com/watch?v=YXPyB4XeYLA>