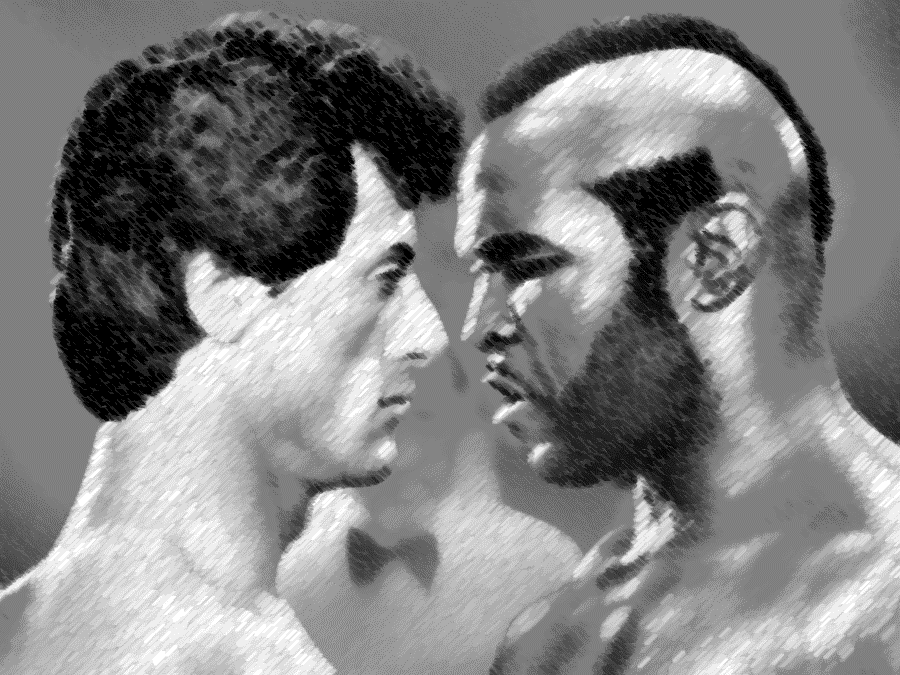
זיהוי תנועת עיניים

זיהוי מתאר עין אנושית בעזרת טכנולוגיית למידת מכונה/למידה עמוקה

עבודת סיום בקורס יסודות הבינה המלאכותית והאנושית

מוגש לפרופסור עודד מימון



|  |  |
| --- | --- |
| פליקס קרסניצקי | 306012865 |
| צבי ששון | 022114730 |

תוכן

[1. הקדמה 3](#_Toc61866796)

[2. תהליך הKDD 4](#_Toc61866797)

[שלב 1 הבעיה 5](#_Toc61866798)

[שלב 2 איסוף הנתונים 5](#_Toc61866799)

[שלב 3 טיוב הנתונים 7](#_Toc61866800)

[שלבים 5,6,7 Data Mining 7](#_Toc61866801)

[שלב 8 בנית הפיטרון התכנותי והרצתו 11](#_Toc61866802)

[שלב 9 הרצה והצגת הידע 18](#_Toc61866803)

[3. מקורות מידע: 19](#_Toc61866804)

# הקדמה

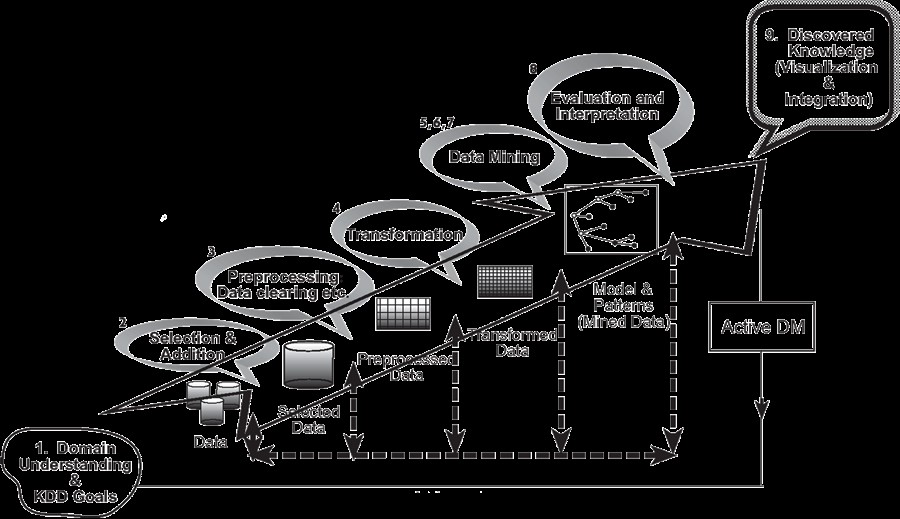
ישנם מספר רב של סיבות רפואיות בהם חולה מאבד יכולת שליטה על השרירים בגוף ושומר רק על יכולות להזיז את שרירי העיניים. במקרים אלו יכולת התקשורת עם החולה לתשובות פשוטות כגון כן \ לא או כל תשובה בעלת מספר אפשרויות מוגבל יכולה להתקיים ע"י האפשרות לעצום את העיניים, למצמץ. שיש שאלה ששואלים את החולה בצורה אקטיבית יראה השואל את התשובה. אבל קיימת הבעיה כיצד ימשוך החולה צומת לב במידה שאין איש לידו? במקרים רבים אין אפשרות לצור קשר עין לעין עם החולה 24/7 במשך זמן ממושך גם עם נמצא אדם בקרבת החולה ובמקרים רבים לא ניתן למצא אדם כזה בגלל בעיות כ"א בבית החולים או אפשרויות המשפחה.

בכדי לפתור בעיה זו ניתן לשים מצלמות ייעודיות המצלמות את פני החולה 24/7 ומודיעות לצוות הרפואי במידה והחולה מבקש ליצור קשר.

בעבודה זו ננסה להראות את פתרון אפשרי בעזרת שימוש בלמידה עמוקה לזיהוי המצמוץ. מכוון שמדובר בהוכחת יכולת אנחנו מגבילים את עצמנו מראש לזיהוי עיין עצומה בתמונת סטילס.

# תהליך הKDD

העבודה תעקוב אחרי המבנה והתהליך של KDD כפי שנלמד בקורס



תהליך מורכב מ- 9 שלבים עיקריים:

1. הבנת התחום – מה השאלה ומה הדרישות, האם יש מגבלות ומה הנתונים הקיימים
2. איסוף הנתונים
3. ארגון הנתונים חיבור ביחד לבסיס נתונים אחד בעל אחידות ביחידות , הורדת נתונים שגויים, השלמת חורים ...
4. טרנספורמציה של הנתונים
5. 6,7 בצוע תהליך הdata mining
6. פיענוח התוצאות
7. הצגת הידע הנרכש

## שלב 1 הבעיה

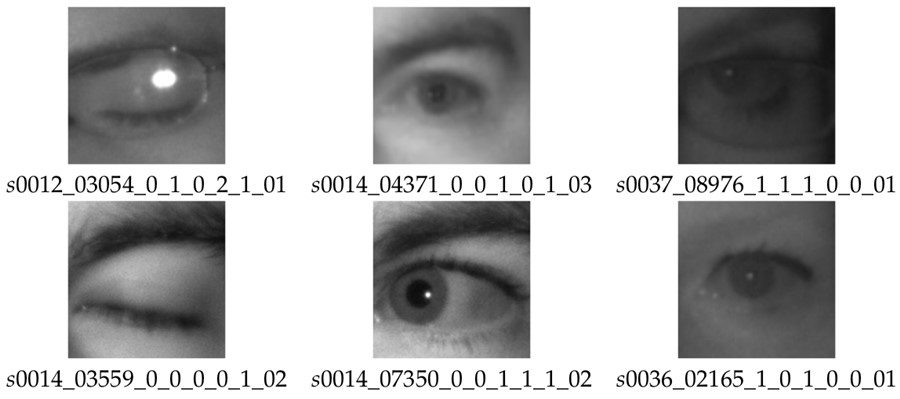
כפי שהוצגה בהקדמה הבעיה היא זיהוי עצימת עיניים של חולה הדרישה שהצבנו לעצמנו היא זיהוי מצב עין סגור \ פתוח בתמונה. קיימים ברשת מספר מאגרי מידה בהם מצולמות עיניים ובהם נשתמש

## שלב 2 איסוף הנתונים

בסיס הנתונים שבחרנו הוא בסיס נתונים MRL Eye Dataset הוא מכיל כ 85000 תמונות של עיניים בפורמט PNG.

התמונות שצולמו הן של 37 אנשים כאשר 43000 תמונות הן של עיניים פתוחות ו42000 הן של עיניים סגורות, בסיס הנתונים מאוזן.

התמונות הבאות מדגימות את תוכן בסיס הנתונים



לכל תמונה בבסיס הנתונים קיים תיוג המופיע בשמה של התמונה לפי הפירוט הבא:

* subject ID: number
* image number: number
* Gender: Nominal
  + 0 – male
  + 1 – female
* glasses: Nominal
  + 0 – no
  + 1 – yes
* eye state: Nominal
  + 0 – close
  + 1 – open
* reflections: Nominal
  + 0 – none
  + 1 – low
  + 2 – high
* lighting conditions/image quality:
  + 0 – bad
  + 1 – good
* sensor type:
  + 01 - RealSense SR300 640x480
  + 02 - IDS Imaging, 1280x1024
  + 03 - Aptina Imagin 752x480

לדוגמה:  
הפירוש של התיוג לתמונה s001\_00123\_0\_0\_0\_0\_0\_01.png

תמונה 123 של נבדק מס' 1. גבר, ללא משקפיים, עיניים סגורות, ללא החזרות, תנאי תאורה לא טובים, תמונה ברזולוציה 640x480 של חיישן RealSense SR300

## שלב 3 טיוב הנתונים

בכדי להקל על העבודה הנחנו כי המטופלים יהיו ללא משקפיים לכן ניתן לבתר על התמונות הכוללות עיניים עם משקפיים

בנוסף הורדנו תמונות שתויגו כתמונות בתנאי תאורה גרועה או עם השתקפויות

נותרו לאחר הסינון 20482 תמונות מהן 14818 של גברים ו5664 של נשים

קובץ הXLS המלא שימש אותנו לסינון התמונות נימצא ב.......

## שלבים 5,6,7 Data Mining

ביצענו סקר ספרות על מנת ללמוד מה הוא האלגוריתם שעדיף להשתמש בו לזיהוי מצב העיין בפרט ומאפייני פני ככלל.

רוב הדעות באתרי שיתוף הידע כגון IEEE וKaggle המליצו על שימוש ברשתות קונבולוציה לצורך זיהוי מתאר הפנים ומצב העיין

נמצאה עבודה אחת שהמליצה על שימוש בשיטת עצי החלטה.

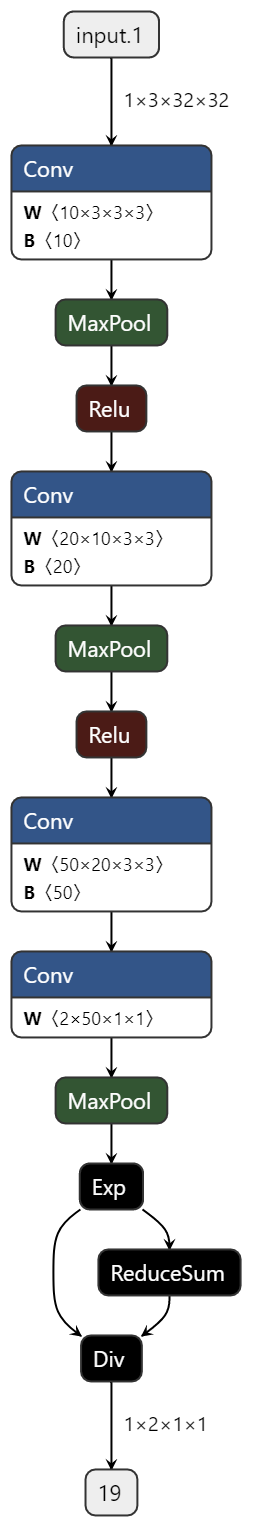
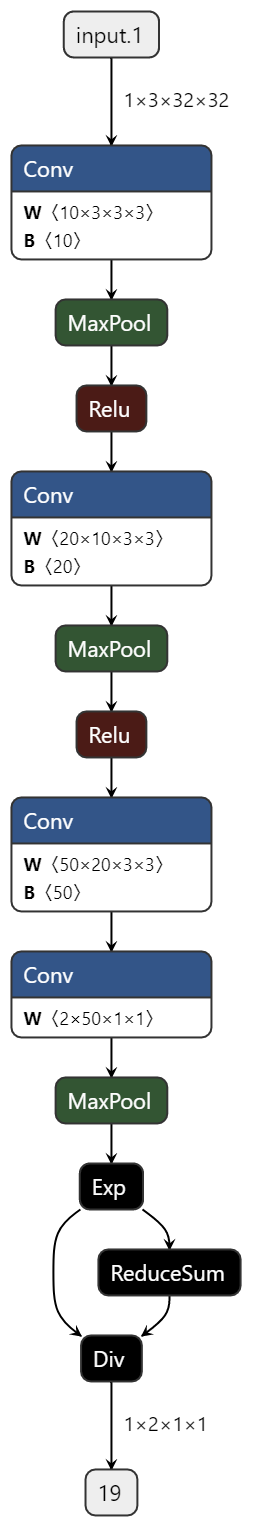
בגלל שמירב ההמלצות והספרות היו על רשתות קונבולוציה ובגלל חוסר המידע לגבי שימוש בעצי החלטה לצורך זיהוי תמונה החלתנו לעבוד בשיטה הראשונה.

ע"פ מספר מקורות, ביניהם Journal of Theoretical and Applied Information Technology מומלצת רשת פשוטה עם מספר שכבות convolution, Relu, MaxPooling



הסבר מפורט על רשתות קונבולוציה נמצא במצגת שהוצגה במהלך הקורס והיא נמצאת ב....

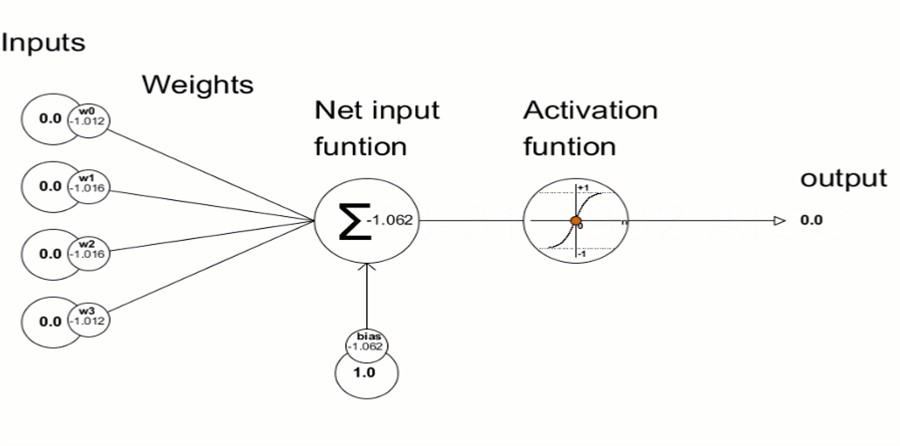
לפי המלצת המאמר[‎2]  בנינו את רשת הקונבולוציה הבאה:



סדר הפעולות ברשת

* קונבולוציה ראשונה 3X3
* צמצום מימד 1<-2
* פונקצית אקטיבציה מסוג RelU
* קונבולוציה שנייה 3X3
* צמצום מימד 1<-2
* קונבולוציה שלישית 3X3
* קונבולוציה רביעית 1X1
* צמצום מימד 1<-2
* SoftMax
* הסתברות

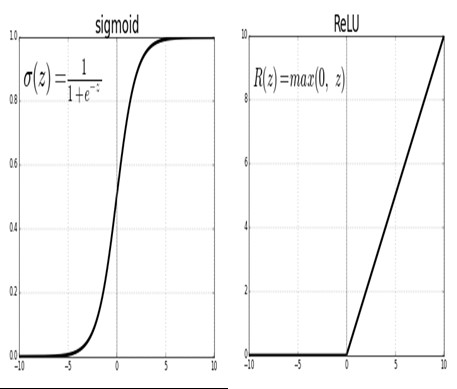
פונקציית אקטיבציה הינה פונקציית תמסורת Transfer function, הפונקציה ממירה את תוצאת שכבת חישוב במקרה שלנו pool למרחב שונה



במקרה שלנו השתמשנו בפונקציה אקטיבציה מסוג RelU Rectified Linear Unit מהסיבות הבאות:

* יותר קלה לחישוב
* מגיבה טוב לשינויים (בתחום החיובי)
* מנוונת נוירונים ומרדדת את הרשת
* התכנסות טובה יותר בתהליך האימון (מלבד בעיית ה"פיצוץ")

הפונקציה ניראת כך



## שלב 8 בנית הפתרון התכנותי והרצתו

את הפתרון התכנותי כתבנו בשפת פייטון והשתמשנו בספריית PyTorch למימוש הרשת

ספריית PyTorch היא סיפרייה מתמטית עם רישיון קוד פתוח הכוללת מספר רב של מודולים בניהם כאלו למימוש לימוד מכונה. הסיפריה בעלת ממשק פייטון וכוללת יכולת להאצה חומרתית את הסיפריה ניתן להוריד מאתר pytorch.org בו גם קיימת דוקומנטציה ודוגמאות

השתמשנו בסיפרייה זאת למימוש אלג לבנית הרשת, דוגמה למימוש אפשרי מופשט לצורך הבנה

import torch  
import torchlayers as tl  
  
# torch.nn and torchlayers can be mixed easily  
model = torch.nn.Sequential(  
 tl.Conv(10),  
 tl.pooling.MaxPool, # specify ONLY out\_channels  
 torch.nn.ReLU(), # use torch.nn wherever you wish  
 tl.Conv(20),  
 tl.pooling.MaxPool,  
 tl.ReLU(),  
 tl.Conv(50),  
 tl.Conv(2),  
 tl.pooling.MaxPool,  
 torch.nn.Softmax(),  
 tl.GlobalMaxPool(),   
 tl.Linear(2), # Output for 2 classes  
)

קוד האימון נכתב לפי הפירוט הבא

אתחול האימון:

def main():  
 # variables -------------  
 batch\_size = 64  
 lr = 0.001  
 epochs = 40  
 # ------------------------  
  
 factory = DataSetFactory()  
 training\_loader = DataLoader(factory.training, batch\_size=batch\_size, shuffle=True, num\_workers=1)  
 validation\_loader = DataLoader(factory.validation, batch\_size=batch\_size, shuffle=True, num\_workers=1)  
 network = model.Model(num\_classes=2).to(device)  
 if not torch.cuda.is\_available():  
 summary(network, (1, shape[0], shape[1]))  
  
 optimizer = torch.optim.Adam(network.parameters(), lr=lr)  
 criterion = nn.CrossEntropyLoss()  
  
 min\_validation\_loss = 10000

שלושת השורות הראשונות הגדרת מבנה וכמות איטרציות של תהליך האימון

*batch\_size = 64  
 lr = 0.001  
 epochs = 40*

בשורות הבאות הגדרת בסיס נתונים לאימון וולידציה, הגדרת רשת

*factory = DataSetFactory()  
 training\_loader = DataLoader(factory.training, batch\_size=batch\_size, shuffle=True, num\_workers=1)  
 validation\_loader = DataLoader(factory.validation, batch\_size=batch\_size, shuffle=True, num\_workers=1)  
 network = model.Model(num\_classes=2).to(device)*

בסוף הגדרת אופטימייזר וקריטריון התכנסות

*optimizer = torch.optim.Adam(network.parameters(), lr=lr)  
 criterion = nn.CrossEntropyLoss()  
  
 min\_validation\_loss = 10000*

תהליך האימון עצמו וחישוב דיוק החיזוי

for epoch in range(epochs):  
 network.train()  
 total = 0  
 correct = 0  
 total\_train\_loss = 0  
  
 for i, (x\_train, y\_train) in enumerate(training\_loader):  
 optimizer.zero\_grad()  
 x\_train = x\_train.to(device)  
 y\_train = y\_train.to(device)  
 y\_predicted = network(x\_train)  
 loss = criterion(y\_predicted, y\_train)  
 loss.backward()  
 optimizer.step()  
 \_, predicted = torch.max(y\_predicted.data, 1)  
 total\_train\_loss += loss.data  
 total += y\_train.size(0)  
 correct += predicted.eq(y\_train.data).sum()  
 accuracy = 100. \* float(correct) / total  
 print('Epoch [%d/%d] Training Loss: %.4f, Accuracy: %.4f' % (  
 epoch + 1, epochs, total\_train\_loss / (i + 1), accuracy))

האימון

*optimizer.zero\_grad()  
 x\_train = x\_train.to(device)  
 y\_train = y\_train.to(device)  
 y\_predicted = network(x\_train)  
 loss = criterion(y\_predicted, y\_train)  
 loss.backward()  
 optimizer.step()  
 \_, predicted = torch.max(y\_predicted.data, 1)*

חישוב דיוק החיזוי

*total\_train\_loss += loss.data  
 total += y\_train.size(0)  
 correct += predicted.eq(y\_train.data).sum()  
 accuracy = 100. \* float(correct) / total  
 print('Epoch [%d/%d] Training Loss: %.4f, Accuracy: %.4f' % (  
 epoch + 1, epochs, total\_train\_loss / (i + 1), accuracy))*

ובסוף ולידציה חישוב דיוק הולידציה ושמרת המודל

with torch.no\_grad():  
 total = 0  
 correct = 0  
 total\_validation\_loss = 0  
 for j, (x\_val, y\_val) in enumerate(validation\_loader):  
 x\_val = x\_val.to(device)  
 y\_val = y\_val.to(device)  
 y\_val\_predicted = network(x\_val)  
 val\_loss = criterion(y\_val\_predicted, y\_val)  
 \_, predicted = torch.max(y\_val\_predicted.data, 1)  
 total\_validation\_loss += val\_loss.data  
 total += y\_val.size(0)  
 correct += predicted.eq(y\_val.data).sum()  
 accuracy = 100. \* float(correct) / total  
 if total\_validation\_loss <= min\_validation\_loss:  
 if epoch >= 3:  
 print('saving new model')  
 state = {'net': network.state\_dict()}  
 torch.save(state, '../trained/model\_%d\_%d\_%.4f.t7' % (epoch + 1, accuracy, total\_validation\_loss / (j + 1)))  
 min\_validation\_loss = total\_validation\_loss  
 print('Epoch [%d/%d] validation Loss: %.4f, Accuracy: %.4f' % (  
 epoch + 1, epochs, total\_validation\_loss / (j + 1), accuracy))

הולידציה

*x\_val = x\_val.to(device)  
 y\_val = y\_val.to(device)  
 y\_val\_predicted = network(x\_val)  
 val\_loss = criterion(y\_val\_predicted, y\_val)  
 \_, predicted = torch.max(y\_val\_predicted.data, 1)*

חישוב הדיוק הוולידציה

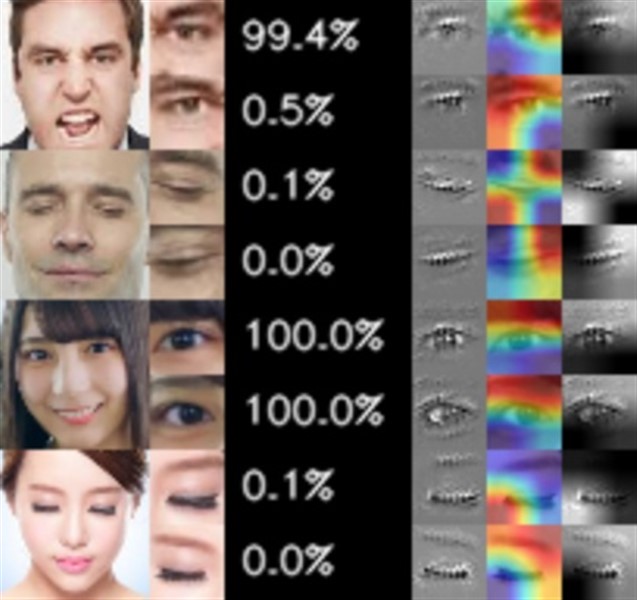
*total\_validation\_loss += val\_loss.data  
 total += y\_val.size(0)  
 correct += predicted.eq(y\_val.data).sum()  
 accuracy = 100. \* float(correct) / total*

שמירת המודל

*print('saving new model')  
 state = {'net': network.state\_dict()}  
 torch.save(state, '../trained/model\_%d\_%d\_%.4f.t7' % (epoch + 1, accuracy, total\_validation\_loss / (j + 1)))*

## שלב 9 הרצה והצגת הידע

לסיום בוצעה הרצה על תוצאות ידועות



ניתן ליראת זיהוי ברמת סמך גבוהה של עיין סגורה או פתוחה ב7 מ8 המקרים

עדיין לא מובן לנו מדוע מקרה אחד לא הצליח בזיהוי ונמשיך לחקור

# מקורות מידע:

1. Kim, K., Hong, H., Nam, G.P., & Park, K. (2017). A Study of Deep CNN-Based Classification of Open and Closed Eyes Using a Visible Light Camera Sensor. Sensors (Basel, Switzerland), 17.
2. Al-Shakarchy, N.D., & Ali, I. (2019). Open and Closed Eyes Classification in Different Lighting Conditions Using New Convolutional Neural Networks Architecture.
3. Fusek, R.: MRL eye dataset. <http://mrl.cs.vsb.cz/eyedataset> (Jan 2018)