



EMOTIONLENS

SEEING BEYOND THE SURFACE

מצגת מלווה - פרויקט למידת מכונה

נחמי ולדנברג

המטרה שלנו

בעידן הדיגיטלי, הבנת חווית המשתמש חיונית להצלחת כל אתר אינטרנט. באופן מסורתי, בעלי אתרים הסתמכו על סקרי שביעות רצון כדי לאמוד את רגשות המשתמש ושביעות הרצון. עם זאת, שיטות אלה אינן יעילות, מכיוון שהן עלולות לגזול זמן, נוטות להטיות ואינן לוכדות את רגשות המשתמש בזמן אמת.

כדי להתמודד עם האתגר הזה, פיתחתי את EmotionLens - פלטפורמה מתקדמת שנועדה לספק לבעלי אתרים דרך מקיפה, מפורטת וקלה להבנה של חוויות המשתמשים שלהם. EmotionLens עושה שימוש במודל בינה מלאכותית לזיהוי רגשות, כדי לנתח את הבעות הפנים של המשתמשים בזמן אמת כאשר הם גולשים באתר.

EmotionLens מספקת פתרון אופטימלי שמשנה את הדרך שבה בעלי אתרים מקבלים מידע על אינטראקציות המשתמשים עם האתר שלהם. על ידי מתן ניתוחים מבוססי רגשות בזמן אמת, היא מעניקה לעסקים את הכוח לקבל החלטות מבוססות נתונים לשיפור שביעות הרצון של המשתמשים, לייעל את ביצועי האתר ולהוביל להצלחה משמעותית.

ארכיטקטורת הפרויקט

בכמה מילים...

Emotionlens היא פלטפורמה שמטרתה לספק תובנות על חוויות משתמש עבור בעלי אתרים.

כל אתר (לקוח) שנרשם לשירות מקבל [רכיב lens](#) להטמעה באתר.

הרכיב דואג לצילום תמונות המשתמשים + פיענוח הניתוב באתר בו צולמו, בצד השרת - הפלטפורמה מנתחת הבעות פנים מתמונות המשתמשים על ידי מודל בינה מלאכותית לסיווג רגשות ומאחסנת את המידע.

דרך האזור האישי בEmotionLens ניתן לצפות במידע מפורט וברור על חוויות המשתמשים באתר הלקוח.

מילון מונחים

:EmotionLens



פלטפורמה שמספקת מידע על חווית משתמש.

:Client - לקוח:



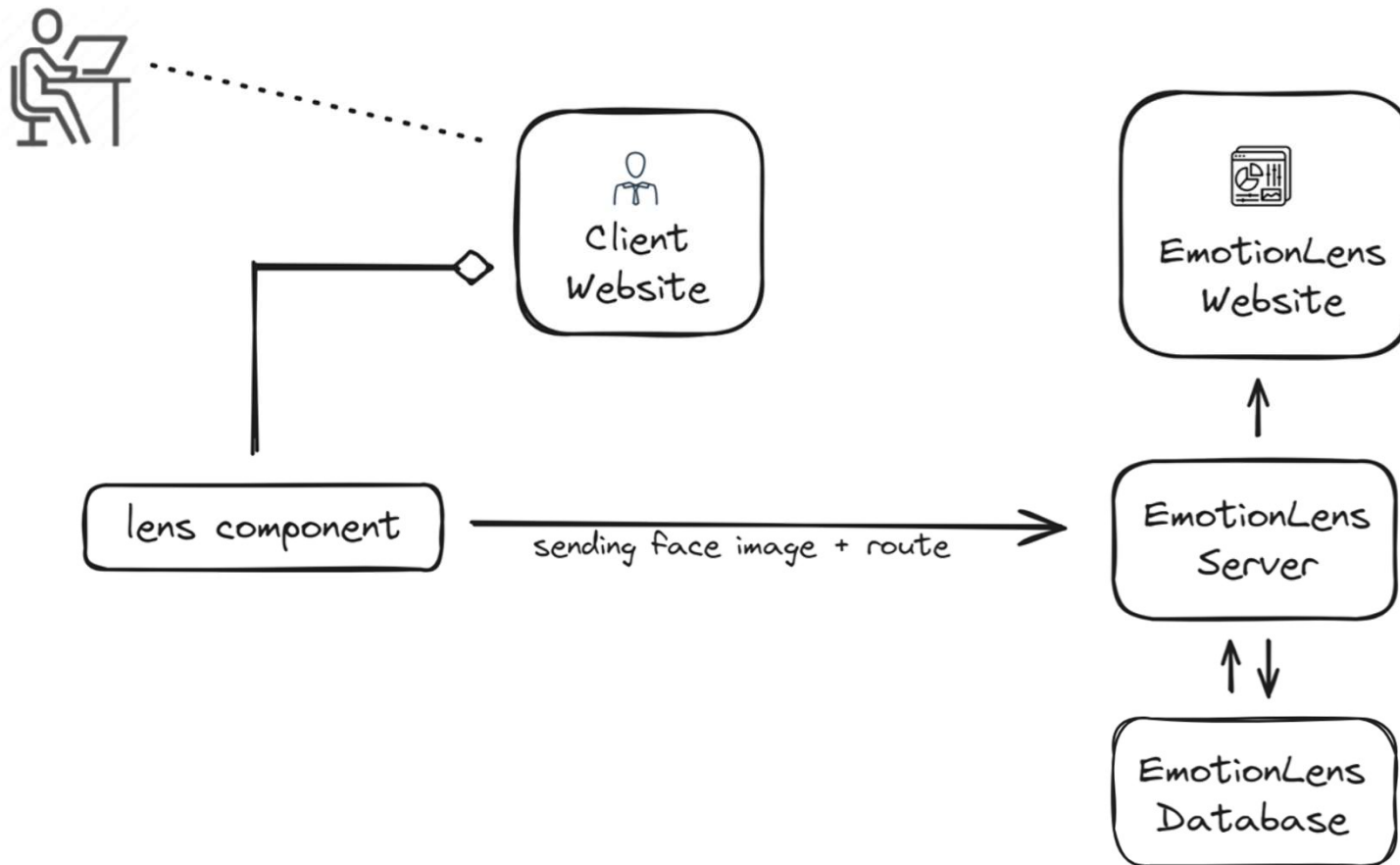
בעל אתר שקנה את השירות של EmotionLens. הקומפוננטה Lens מותקנת באתר שלו.

: User



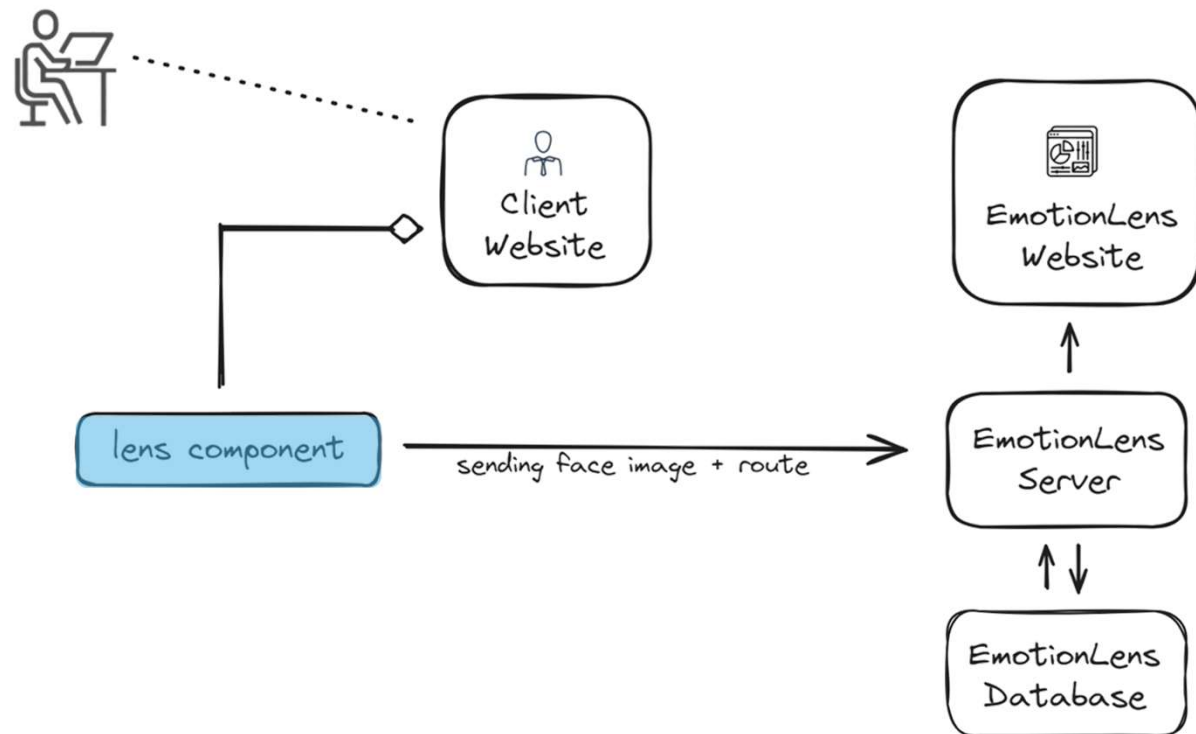
גולש באתר הקליינט.

ארכיטקטורה



פירוט

רכיב lens



האתגר

האתגר שלי היה שכל אתר לקוח יוכל להטמיע אצלו בקלות, ללא שינויים דרמטיים בקוד, את האפשרות לצלם את משתמשי האתר ולקלוט את הניתוב בו צולמה התמונה. התמונה והניתוב צריכים להישלח בצורה מאובטחת לשרת שבו יבוצעו ניתוח הנתונים.

לאחר בחינה מעמיקה של מספר טכנולוגיות פוטנציאליות, הגעתי למסקנה שהפתרון האופטימלי הוא יצירת "קופסה סגורה" – רכיב בעל פונקציונליות מובנית לקליטת תמונות וניתוב ושליחתם לשרת. הפתרון הזה פותח כקומפוננטה הניתנת להעלאה והתקנה בקלות דרך npm, מה שמאפשר אינטגרציה מהירה ופשוטה באתרי הלקוחות, ללא צורך בשינויים משמעותיים בקוד הקיים.

לצורך העלאת הרכיב לnpm נדרשה למידה עצמית. נעזרתי [במאמר הבא](#).

יישום

אופן השימוש ברכיב:

בעל האתר מתקין את רכיב ה-lens באתר שלו
הוא מוסיף את קומפוננטת lens בדפים בהם הוא מעונין בניתוח חווית המשתמש
בנוסף, הוא יכול להתאים את תדירות השימוש בקומפוננטה על ידי הגדרת טווח הזמן בשניות.

סקירת הפונקציונליות:

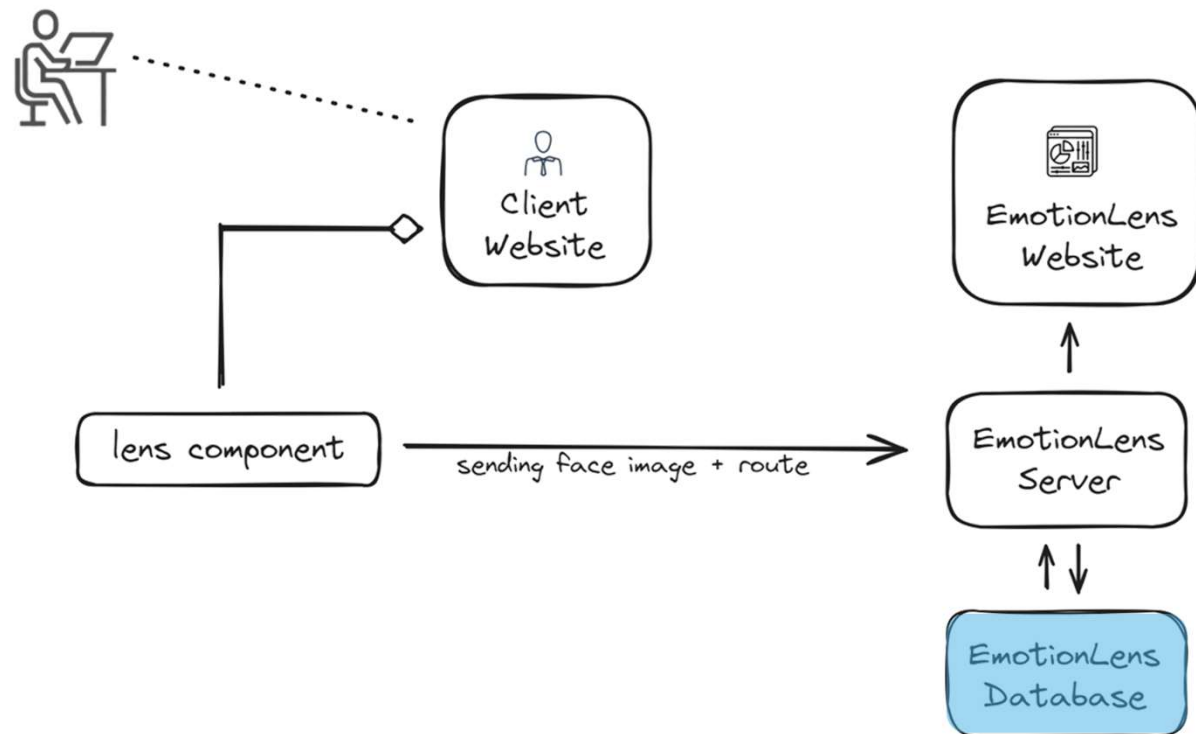
התממשקות למצלמת המשתמש

הקומפוננטה מצלמת תמונות מהמצלמה המובנית של המשתמש בדפדפן.
כל מספר השניות שהוגדרו על ידי הלקוח מופעלת פונקציה אסינכרונית שמצלמת את הוידאו ומשתמשת בקנבס
לצילום התמונה. לאחר מכן היא ממירה אותה ל- dataURL ושומרת במצב פנימי. לבסוף, היא שולחת את התמונה
בפורמט Blob לשרת באמצעות בקשת HTTP POST באמצעות axios.

קליטת הניתוב שבו צולמה התמונה

הקומפוננטה עוקבת אחר כתובת ה- URL הנוכחית של הדף באמצעות שימוש באובייקט window. הניתוב נשלח לשרת
באמצעות HTTP POST.

DataBase



טבלת לקוחות:

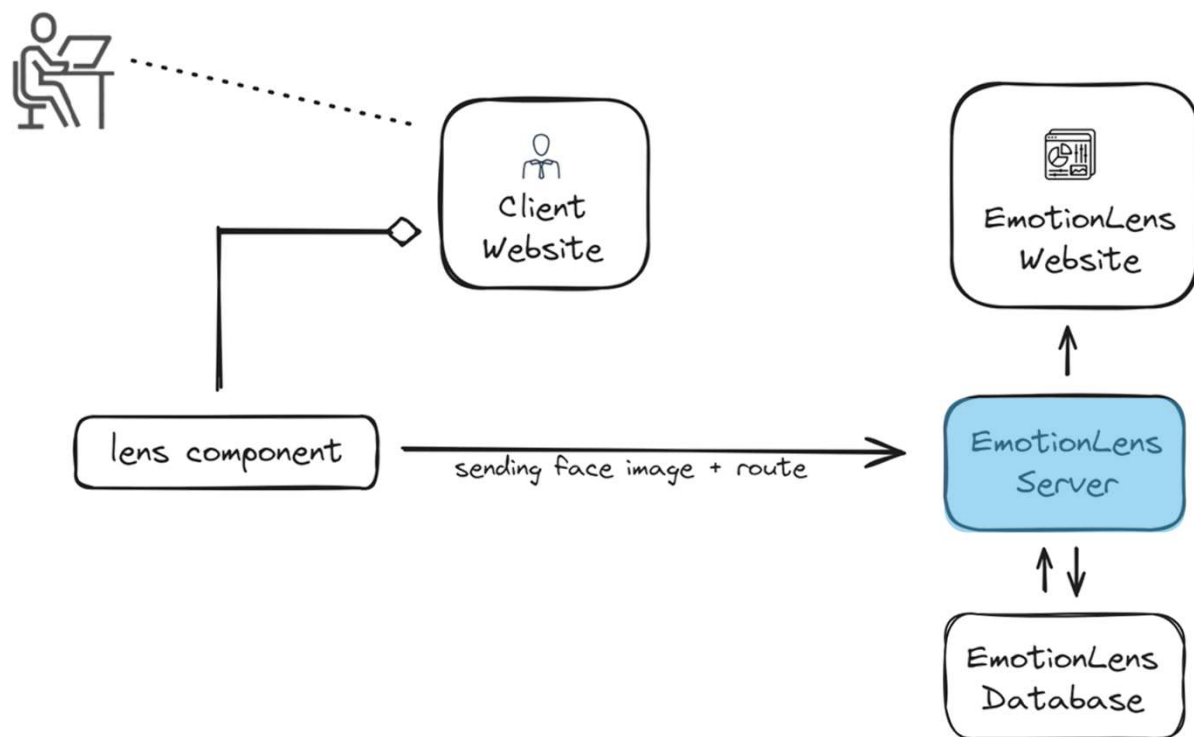
בטבלה זו נשמרים עבור כל לקוח שנרשם פרטים אישיים והכתובת הבסיסית של האתר בו הוא מעוניין במעקב.

טבלת רגשות:

כל רגש שפוענח מקושר למספר הלקוח שלו, וכולל את סוג הרגש, את התאריך שבו נרשם הרגש ואת המיקום באתר בו זוהה.

באמצעות תיעוד הרגשות ניתן לספק לכל לקוח מידע על רגשות המשתמשים באתר שלו.

צד שרת



צד השרת צריך לתמוך בשני דברים מרכזיים:

1. קבלת הנתונים מהרכיב ותהליך פיענוח הרגש:

- קבלת תמונות ונתיבים מהרכיב המוטמע באתר הלקוח.
- שליחת התמונות למודל הבינה המלאכותית לצורך פענוח רגשות, ניתוח נתונים ושמירתם.

2. טיפול באתר EmotionLens:

- ניהול המשתמשים בשירות.
- שליפת ושליחת נתונים ל-database בצורה מאובטחת ויעילה.
- תמיכה בצד הלקוח לצורך הצגת המידע ונתוני הניתוח בצורה נוחה ואינטראקטיבית.

קבלת הנתונים מהרכיב ותהליך פיענוח הרגש:

התהליך המלא נזקק לשני הנתונים בזמן אמת: תמונה + נתיב, אך השרת מקבל אותם בבקשה נפרדת. לצורך כך, פונקציה פועלת ברקע ורק כששתי הנתונים מתקבלים בהצלחה הם נשלחים להמשך טיפול.

קבלת התמונה מהרכיב, שליחתה למודל וסיווג הרגש:

התמונה מתקבלת כתמונה צבעונית גדולה ומכילה פריטים רבים. המודל מקבל תמונה של פנים בלבד בגווי האפור ובגודל 48×48 לשם כך נדרש לעשות מניפולציות על התמונה. לצורך המניפולציות נעשה שימוש בספריית OpenCV – ספריית קוד פתוח לעיבוד תמונה וראיה ממוחשבת, המספקת כלים ומודלים מגוונים לזיהוי ועיבוד אובייקטים בתמונות ווידאו:

- שינוי לגווי האפור
כדי להמיר תמונה מצבעונית (BGR) לתמונה בגווי אפור (Gray) השתמשתי בפונקציה cvtColor מספריית OpenCV.
- זיהוי הפנים מתוך התמונה המלאה:
השתמשתי בכלי לזיהוי פנים באמצעות האלגוריתם 'Haar Cascade' המאפשר זיהוי אזורי פנים בתמונות בצורה מדויקת ויעילה. אלגוריתם זה עובר על התמונה ומחפש מסננים המתאימים לזיהוי תבניות חזותיות המאפיינות פנים. כאשר הוא מוצא תבנית שמזוהה כפנים, הוא מחזיר את האזור בתמונה שמתאים.
- שינוי הגודל
כדי לשנות את גודל התמונה השתמשתי בפונקציה: cv2.resize

תמונת הפנים הערוכה נשלחת למודל שמסווג את הרגש ומחזיר אותו.

קבלת הניתוב:

מהרכיב התקבלו תמונה וניתוב המתאר היכן צולמה התמונה. על מנת לשמור את הרגש בדטהייס נדרש לדעת לאיזה לקוח שייך הניתוב שהתקבל (מאיזה אתר הגיעה התמונה). את הניתוב שהתקבל ניתחתי באמצעות ביטוי רגולרי לחילוף כתובת הבסיס של האתר. לאחר שהייתה כתובת הבסיס, יכולתי לשלוף את הלקוח המתאים מהDB.

לאחר שהתמונה והניתוב עברו את התהליך הנ"ל, יש בידינו את כל הפרמטרים לעדכון הרגש החדש בDB.

טיפול באתר EnotionLens

בתפקיד הצד שרת של EnotionLens מתבצעות מספר פעולות מרכזיות המיועדות לניהול ולאבטחת הנתונים באופן יעיל ומאובטח:

ניהול הכניסה וההתחברות של הלקוחות

הצד שרת של EnotionLens מספק תשתית לניהול הכניסה וההתחברות של הלקוחות למערכת. זה כולל אימות ואימות גישה על פי מידע אישי וסודי כמו שם משתמש וסיסמה. פעולות אלו מבטיחות שרק משתמשים מורשים יכולים לגשת למידע ולבצע פעולות במערכת.

ניהול ושמירה ב-session storage

עם כניסה של המשתמשים למערכת, המידע הרלוונטי נשמר ב-session storage. זה מאפשר לשמור את מצב ההתחברות הנוכחי של הלקוח ולהשתמש בו במהלך השימוש הנוכחי באתר, בצורה מאובטחת ובלתי נגישה לגורמים חיצוניים.

ניהול וגישה לבסיס הנתונים

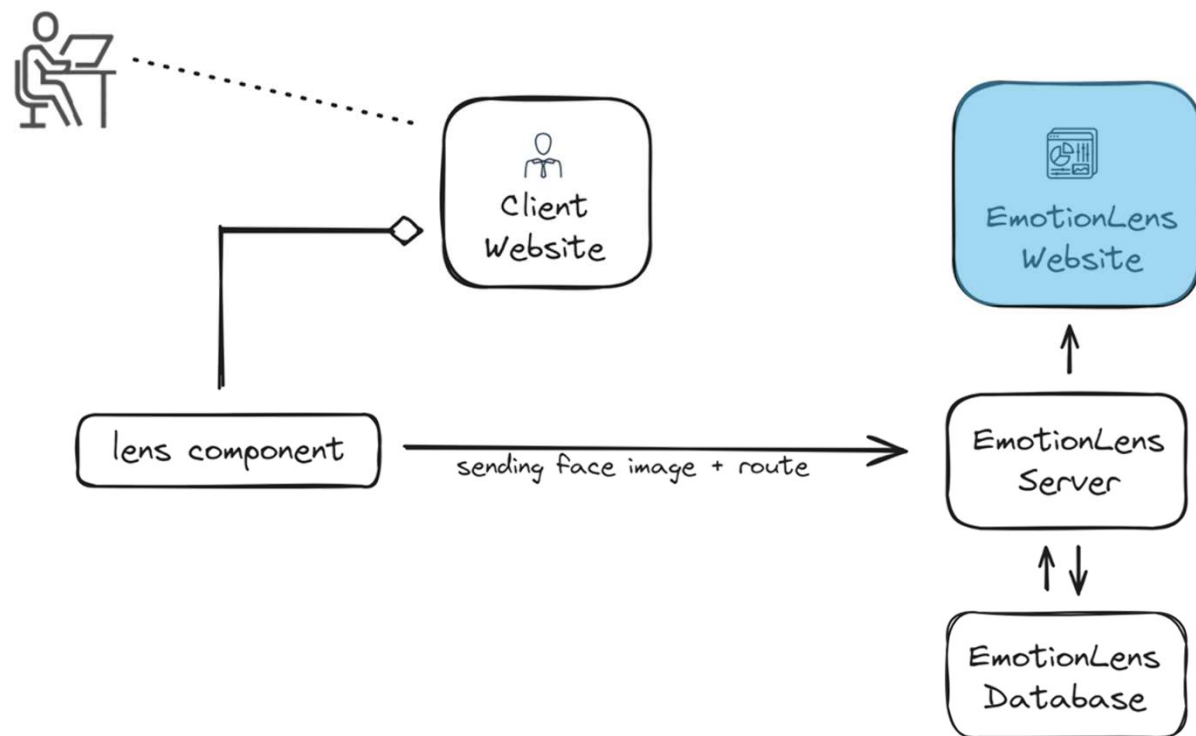
הצד שרת מספק גישה מאובטחת וניהולית לבסיס הנתונים של המערכת. זה כולל פעולות כמו שליפה, עדכון ומחיקת נתונים באופן מאובטח, על מנת להבטיח את השמירה והטיפול בנתונים בצורה יעילה ומוגנת.

אבטחת פרטיות:

על מנת לשמור על פרטיות ואבטחת הגולשים, התמונה שצולמה לא נשמרת בשום מקום. היא מועברת בפרמטר באופן ישיר למודל ולא נעשה בה שימוש נוסף.

באמצעות תשתית הצד שרת המוכנה והמאובטחת של EnotionLens מובטחת הגנה וניהול יעיל של המידע הרגיש שנשמר במערכת, מאפשרת ללקוחות חוויה בטוחה ומותאמת אישית במערכת.

צד לקוח



הרשמה:

אתר שמעוניין בשירות נרשם ל-EmotionLens על ידי מילוי טופס הרשמה.

ללקוח רשום ניתנות האפשרויות הבאות:

הורדת הרכיב:

הלקוח יכול להוריד את רכיב ה-EmotionLens ולהטמיע אותו באתר באמצעות הוראות ברורות ופשוטות. קישורים ל- npm זמינים להורדה מהירה וסיוע טכני זמין במקרה של שאלות או בעיות.

אזור אישי:

האזור האישי ב-EmotionLens מספק גישה מאובטחת ונוחה למידע מפורט על תוצאות הניתוח של חווית המשתמשים. בעלי האתר יכולים לצפות במידע על רגשות המשתמשים בצורה ברורה ומפורטת, באמצעות גרפים וטבלאות שניתנות לסינון אישי. זה מאפשר להם לראות את המידע הרלוונטי ביותר בצורה מהירה ויעילה.

אבטחה ופרטיות:

כל המידע המוצג באזור האישי מנוהל באופן מאובטח ומוגן, מבטיח שהמשתמשים יכולים להיות בטוחים בשמירה על פרטיותם. האבטחה המתקדמת מבטיחה את השמירה על המידע בכל עת.

מודל בינה מלאכותית

המודל מקבל תמונה, מחלץ ממנה את הפנים ומסווג לאחד מחמשת הרגשות.

תכנון עבודה

1. איסוף data
2. הכנת הנתונים למודל
3. בניית המודל
4. שמירת המודל
5. שימוש במודל
6. הפיכת הקוד ממודל לפרויקט

מקור הנתונים

מאגר הנתונים ההתחלתי

במקרה זה היה צורך למקור נתונים גדול, שמתמקד ברגשות ספציפיים. מתוך שבעת הרגשות האוניברסליים שקבע פול אקמן, פסיכולוג אמריקאי, (גועל, כעס, פחד, עצב, אושר, בוז והפתעה) סיננתי את הרגשות הרלוונטיים: כעס, עצב, אושר, הפתעה ורגש ניטרלי.

כמקור ראשוני פניתי למאגר הנתונים Kaggle, שנתן אלף תמונות בממוצע עבור כל רגש.

[מאגר הנתונים ההתחלתי](#)

ניקוי רעשים
























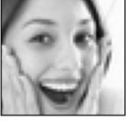

אמנם במבט ראשוני וברפרוף מהיר נראה היה ש**המאגר** מעולה, אך בבדיקה מעמיקה התברר כי סיווג הרגשות כפי שהם בתיקיות לא ברור גם לעין אנושית. קודם כל סיננתי ידנית 130 תמונות לכל רגש וניסיתי להפעיל עליהם את המודל, הוא נתן תוצאות משופרות, אך לא אופטימליות עקב הנתונים המעטים. לשם כך היה צורך להגדיל את מאגר הנתונים.

לא היה צורך לבצע מניפולציות על התמונות –
הם הגיעו בגודל אופטימלי (48*48) ובגווני שחור לבן (בעלות 2 ממדים בלבד)

עיבוי נתונים

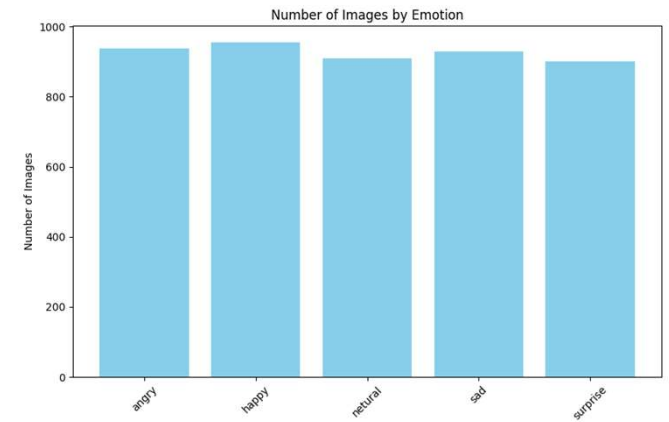
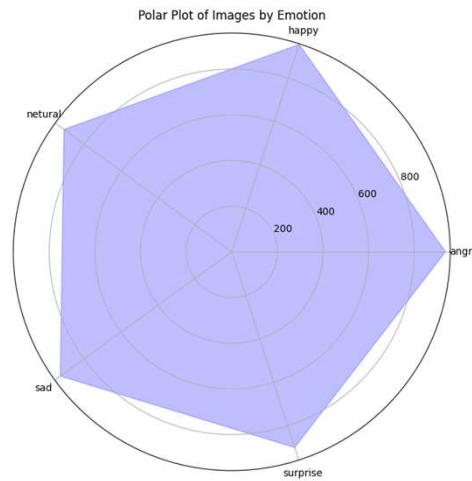
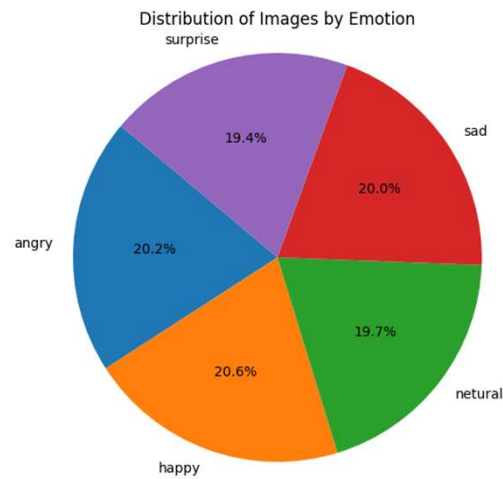
את ההגדלה עשיתי באמצעות גישה של augmentation –
טכניקה המשמשת בלמידת מכונה, כדי להגדיל את המגוון של מערך האימון מבלי
לאסוף נתונים חדשים בפועל.
הגיוון בא לידי ביטוי בשינויים ויזואליים דוגמת סיבוב, צביעה, בהירות, חדות ותוספת
רעש.

הצגת הנתונים

angry					
happy					
netural					
sad					
surprise					

תרשימים

התרשימים הראו שכמות הנתונים פחות או יותר שווה בין הרגשות השונים, כך שלא יתכן מצב בו רגש אחד יהיה מאומן יותר מהשני.



הכנת הנתונים למודל

הכנת הנתונים למודל

את הנתונים חילקתי ל 3 תיקיות: train, validation, test
ביחס של 70% 20% 10%
החלוקה התבצעה באמצעות קוד בפייתון.

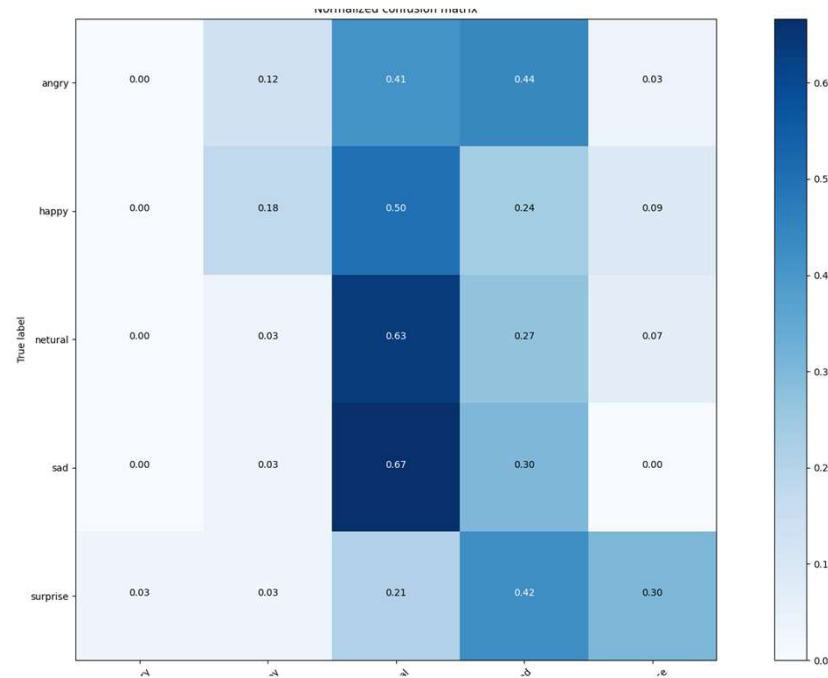
בניית המודל

סוג למידה - CNN

סוג הלמידה הוא Convolutional Neural Network - CNN
מודלים של CNN יעילים במיוחד עבור סיווג תמונות, זיהוי אובייקטים ומשימות ראייה
ממוחשבת אחרות, ולכן בחרתי להשתמש בו בפרויקט.

תהליך בניית המודל

הגישה הראשונית שלי לסיווג תמונה כללה שימוש במודל קלאסי של CNN. למרות ההשקעה, ביצועי הדגם היו לא אופטימליים. ניסינו שיפורים שונים, כמו הוספת שכבות נוספות וכוונן היפרפרמטרים, אך התוצאות עדיין לא משביעות רצון. זה גרם לי לחקור ארכיטקטורות מתקדמות יותר שהוכחו כיעילות.



תהליך בניית המודל

על מנת להגיע לתוצאות אפקטיביות, חקרתי את צורת הפעולה של מודל VGG-16, מודל סיווג תמונות מבוסס ויעיל במיוחד. VGG-16 ידוע בארכיטקטורה העמוקה שלו, המורכבת מ-16 שכבות משוקללות, שהוכחה כמספקת ביצועים מעולים במשימות שונות של סיווג תמונות. בניית המודל נעשתה לפי ארכיטקטורת השכבות בVgg-16, התאמתי את המודל כך שיתאים יותר לצרכים הספציפיים שלי ולמאפיינים הייחודיים של הנתונים.

A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

```

model = Sequential()
model.add(Input(shape=input_shape))

# Convolutional layers
model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=128, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(Conv2D(filters=128, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(Conv2D(filters=256, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

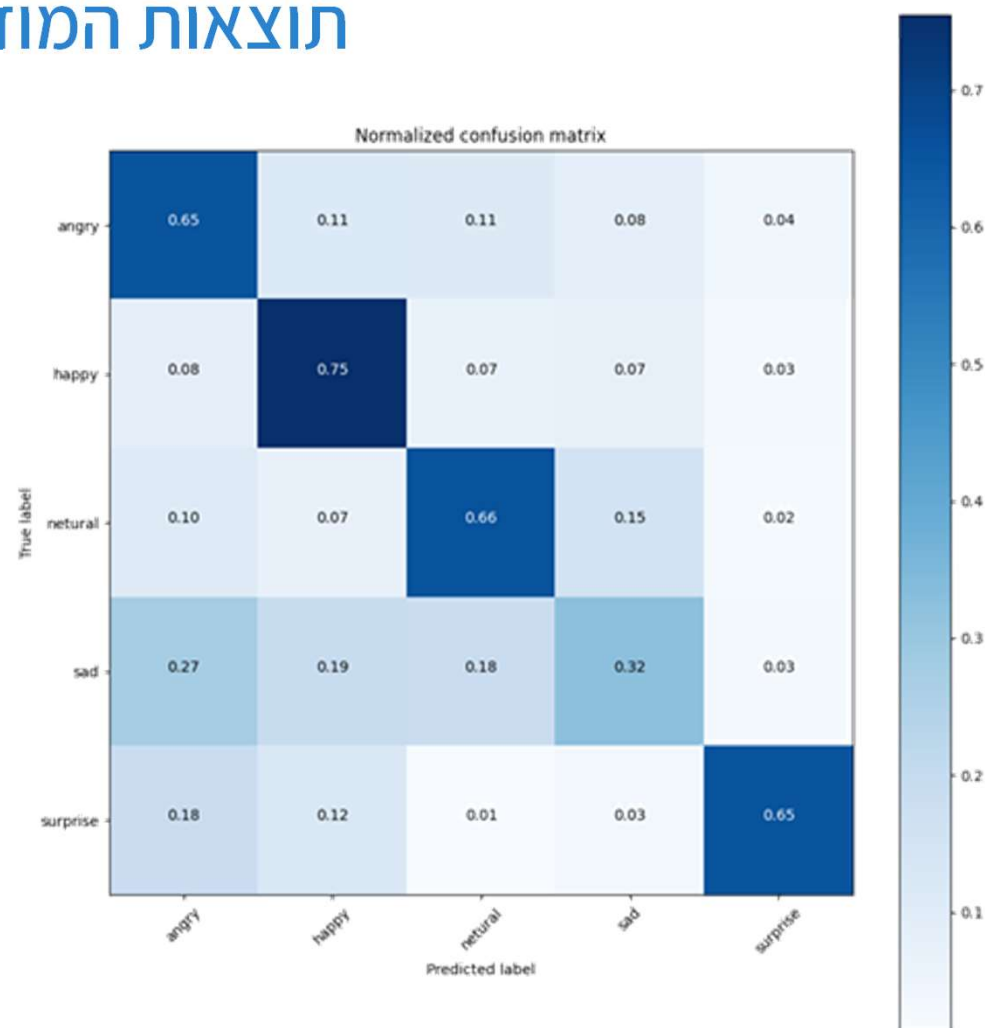
model.add(Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(Conv2D(filters=512, kernel_size=(3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Fully connected layers
model.add(Flatten())
model.add(Dense(units=4096, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(units=4096, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(units=4096, activation='relu'))
model.add(Dropout(0.25))

```


תוצאות המודל

המודל השיג דיוק של 79.11%. ניתוח התוצאות מגלה שהסיווגים השגויים התרחשו בעיקר ברגשות שזיהויים מאתגר אפילו את העין האנושית. כתוצאה מכך, אחוז הדיוק נחשב משביע רצון.



ארכיטקטורת המודל

המודל הוא רשת עצבית Convolutional Neural (CNN) שמבוססת על הארכיטקטורה של VGG16. המודל מיועד לסיווג תמונות לחמש קטגוריות רגשות.

המודל נבנה באמצעות ה-API 'Sequential' מבית Keras עם השכבות הבאות:

- 1. Input Layer**
צורת הקלט היא (1, 48, 48) עבור תמונות בגוויי אפור.
- 2. Convolutional Layers**
נעשה שימוש בארבע קבוצות של שכבות קונבולוציוניות, כל אחת עם גדלי מסננים הולכים וגדלים (64, 128, 256, 512) ופונקציית אקטיבציה Relu. שכבות MaxPooling משמשות לאחר כל קבוצה של שכבות קונבולוציוניות כדי להפחית ממדים מרחביים.
- 3. Fully Connected Layers**
שלוש Fully Connected Layers, כל אחת עם 4096 יחידות, ולאחריהן הפעלת Relu ו-Dropout להסדרה.
- 4. Output Layer**
שכבה צפופה אחרונה - dense עם 5 יחידות (אחת לכל רגש) והפעלת softmax ל multi-class classification

קומפילציה ואימון המודל

Optimizer: Adam optimizer עם קצב למידה 0.0001

Loss Function: מכיוון שזו בעיית סיווג נעשה שימוש ב `categorical_crossentropy`

ModelCheckpoint: שומר את הדגם הטוב ביותר בהתבסס על `validation accuracy`.

EarlyStopping: מפסיק את האימון אם `validation loss` אינו משתפר במשך 7 epochs.



EMOTIONLENS

SEEING BEYOND THE SURFACE

תודה שהייתם איתנו:)

נדלקתם על הרעיון? חושבים שיש לו עתיד גדול? מוכנים להשקיע?
כתבו לי ל emotionlens.invest@gmail.com