# ספר פרויקט:

בית ספר: מקיף י"א ראשונים

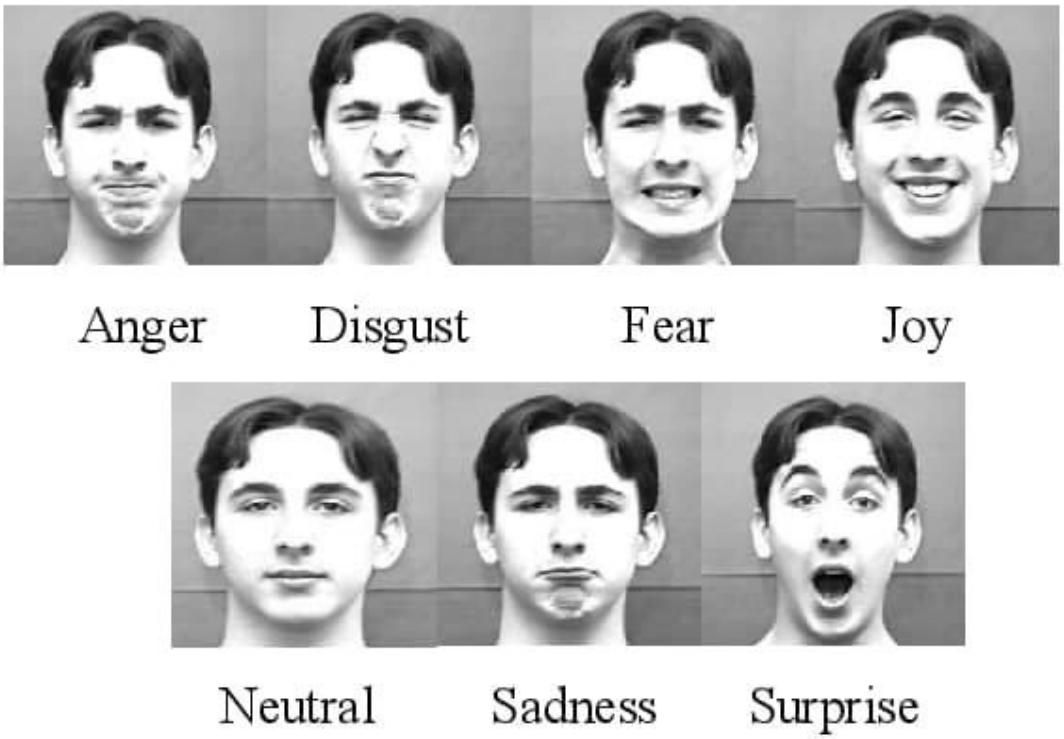
שם עבודה:facial expression recognition model

שם: אודליה לוין

תז: 327577599

שם המנחה: דינה קראוס

תאריך הגשה:19.6.22



# תוכן עניינים



# מבוא-

## הרקע-

בפרויקט שלי רציתי לשים לעצמי לקהל יעד אנשים עיוורים, המטרה של הפרויקט היא בעצם לזהות הבעות פנים, הקוד שכתבתי מזהה את ההבעות הבאות: כעס, גועל, פחד, שמחה, נטרלי, עצב, והפתעה. בעזרת הקוד העיוורים יוכלו לזהות את ההבעות של האנשים שאיתם הם מדברים וכך יוכלו להבין אותם טוב יותר, ותקשורת בין אנשים עיוורים לאנשים בלי המוגבלות תעשה באופן חלק יותר. בעתיד האופן האידיאלי שאני רואה את האלגוריתם הזה פועל בו הוא באפליקציה נוחה בטלפון לעיוורים שבה בעזרת המצלמה הטלפון יזהה את ההבעה של האדם כל כמה זמן רלוונטי ויעביר אותה לאוזניה של המשתמש. בחרתי בנושא הזה משום שהבעות פנים מעידות לעיטים על רגשות, ולרגשות יש חשיבות עצומה בחיינו ,ביחסים בין אישיים ובכלל. הנושא עניין אותי ויותר מזה ראיתי מה ניתן לעשות עם דבר כזה בשוק , בעזרת הקוד ופיתוח שלו ניתן להגיע למגוון רחב של תחומים שיהיה ניתן להשתמש בו, למשל שיפור שירות של חברות לכל אדם באופן אינדיבידואלי לפי איך שהוא הרגיש בזן צריכת המוצר שלהם, הדבר יכול להיות אפקטיבי אוד בעולם הווירטואלי ואף ניתן יהיה לזהות רגשות של אנשים חשובים( בעולם הפוליטי ועוד) אחד כלפי השני.

# ארכיטקטורת הפרויקט-

## איסוף הכנה וניתוח הנתונים-

הdataset שלי נלקח מהאתר Kaggle הנה הקישור-
https://www.kaggle.com/datasets/aadityasinghal/facial-expression-dataset

הוא בנוי מקבצים של תמונות של הבעות פנים, יש קבצים של train וקבצים של test , התמונות בגודל של 48*48 פיקסלים

**הנה תמונה לדוגמה**

כפי שרואים היא בשחור לבן (gray scale) בגודל של 48*48 פיקסלים.

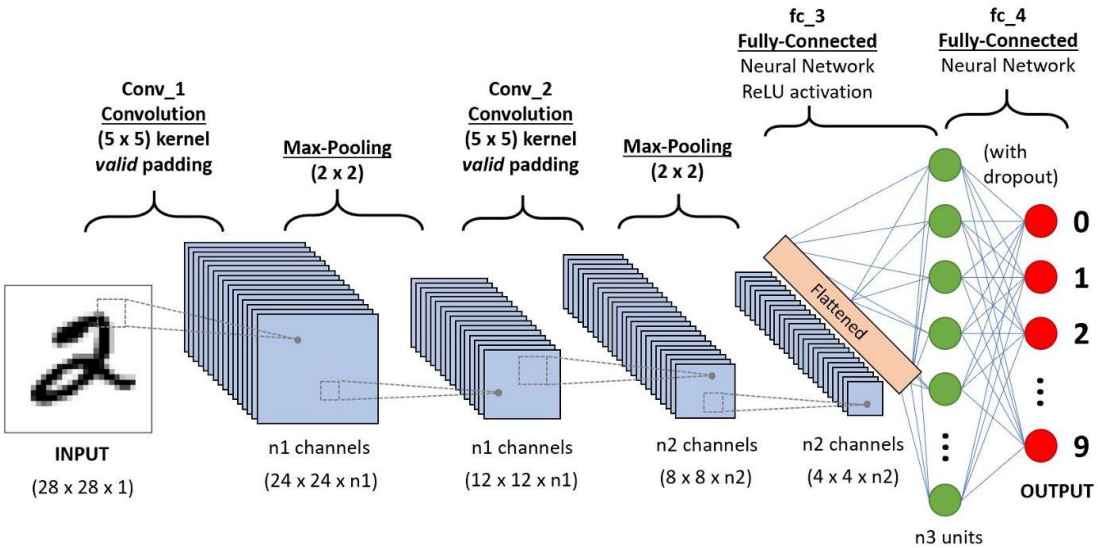
לא התעסקתי עם הקבצים יותר מידי כי הם באו מוכנים זאת אומרת שהם כבר היו מחולקים במבנה הקבצים שלהם לפי ההבעות פנים שאיתם התעסקתי. השתמשתי בimage data generator של keras בכדי להגדיל באופן מלאכותי את כמות הפריטים בdataset בכך שהוא מייצר תמונות חדשות , כי הוא הופך אותם בציר האנכי. בנוסף השתמשתי בflow from directory שלוק את מיקום תיקיית הdataset ומחלק את המידע לbatches באופן אוטומטי לפי מבנה התיקייה. נרמלתי את הנתונים בעזרת color mode gray scale של הפונקציה flow from directory של keras שהופך כל פיקסל לערך בין 1-0 כאשר 0 זה שחור ו 1 זה לבן וביניהם זה גוונים שונים של אפור.

## בנייה ואימון המודל-

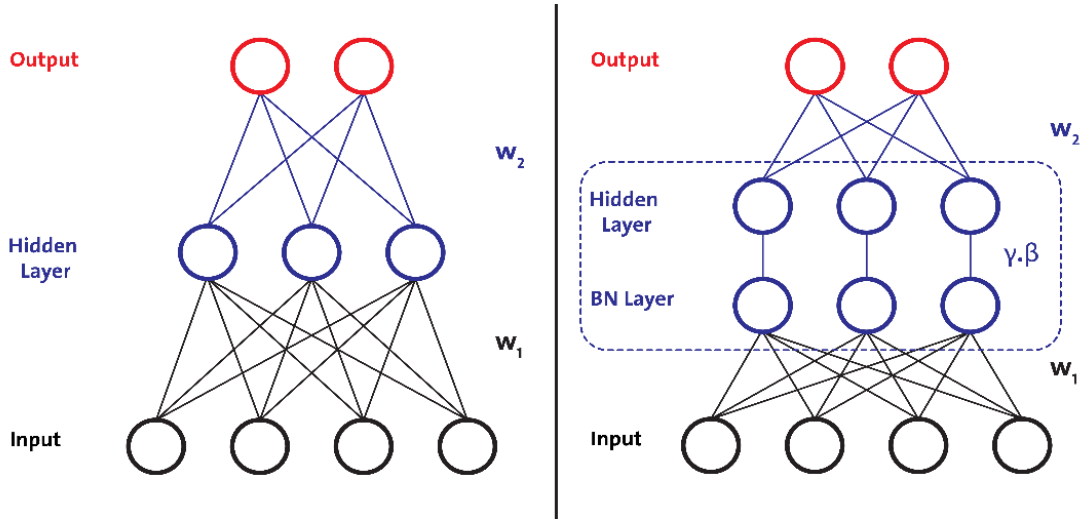
תיאור גרפי של המודל עליו בוצע האימון נמצא בנספחים.

סוגי שכבות שונות:

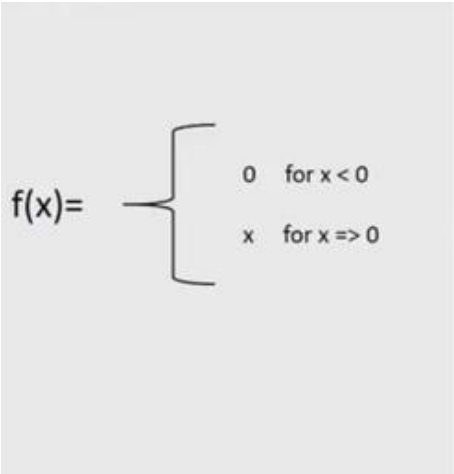
בConvolutional Layer – cnn סוג של רשת נוירונים שמשתמשת בפעולת הקונבלוציה במקום בכפל כללי בלפות את השכבות. הסוג הזה של רשת הנוירונים משמש בעיקר לעיבוד תמונה ועוד כל מיני שימושים.

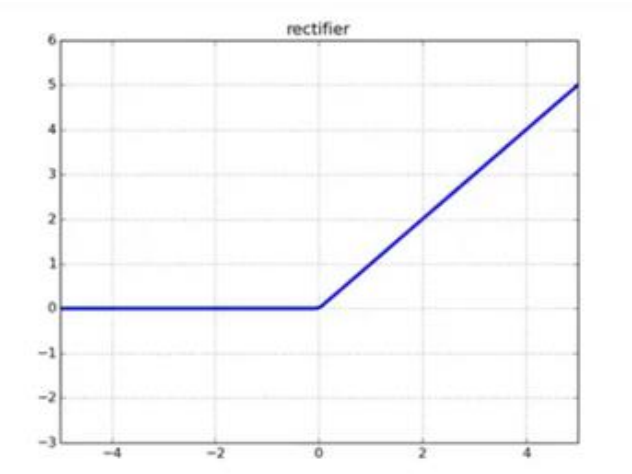



בbatch normalization – היא שיטה המשמשת להפיכת רשתות נוירונים מלאכותיות למהירות ויציבות יותר באמצעות נורמליזציה של הinputs של השכבות על ידי מרכוז מחדש ושינוי קנה המידה.

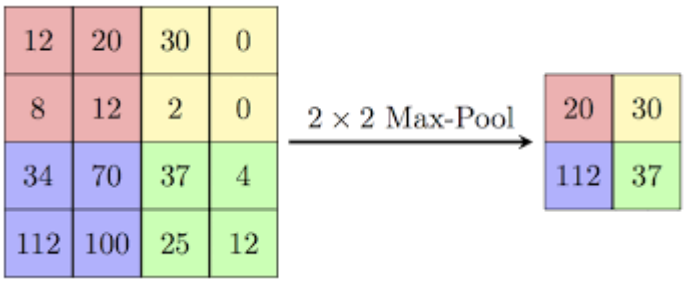



בrelu פונקציית הפעלה שמשמאל לאפס היא אפס ומימין לאפס היא מחזירה את הערך עצמו.
משתמשים בזה בשביל להאיץ את מהירות האימון של רשתות נוירונים .

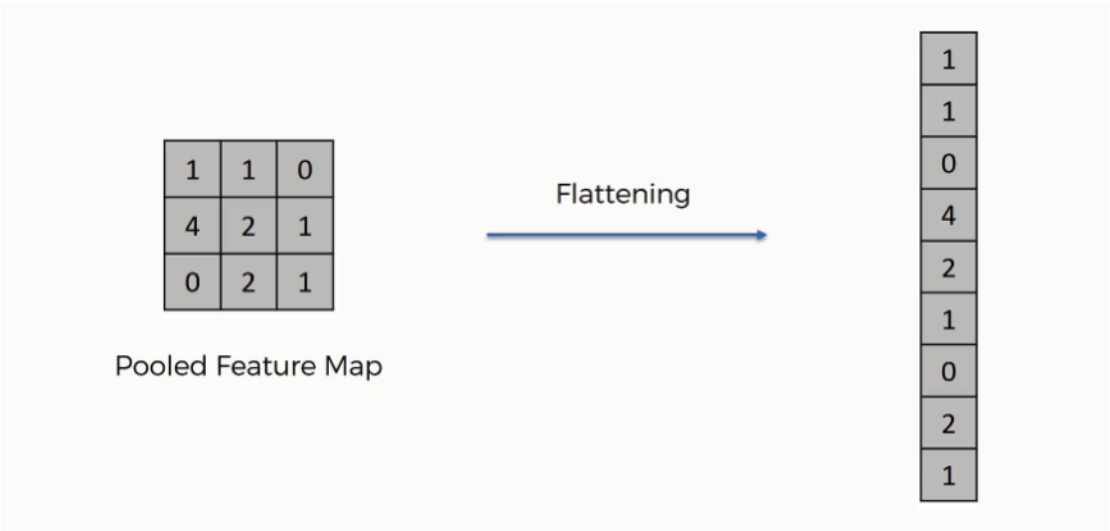


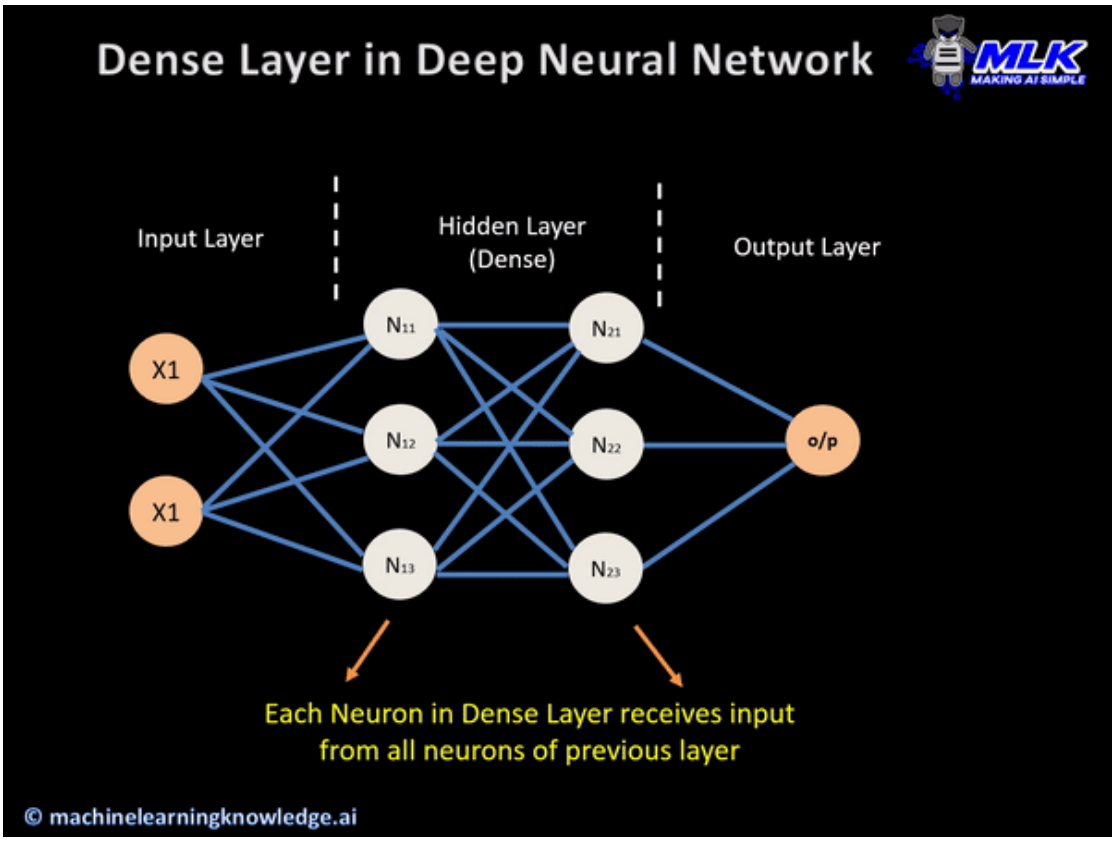

בmaxpooling2d עבור נתונים מרחביים דו ממדיים, מוריד את הרזולוציה של הקלט לאורך הממדים המרחביים שלו(גובה ורוחב) על ידי לקיחת הערך המקסימלי בחלון הקלט שבחרנו. החלון מוזז בצעדים לאורך כל מימד. בכך אנחנו מבטיחים תוצאות כלליות יותר, אשר יתאימו יותר גם לtest .



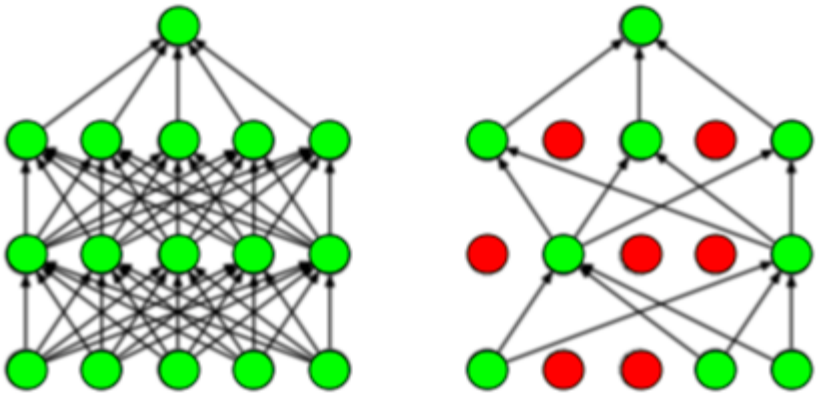

בflatten לוקח מטריצה ומשטח אותה, הופך אותה לווקטור בעל אותו מספר ערכים.




בdense layer בכל רשת נוירונים , שכבה צפופה (dense layer) היא שכבה המחוברת באופן עמוק עם השכבה הקודמת שלה, כלומר הנוירונים של השכבה מחוברים לכל נוירון מהשכבה הקודמת שלה.

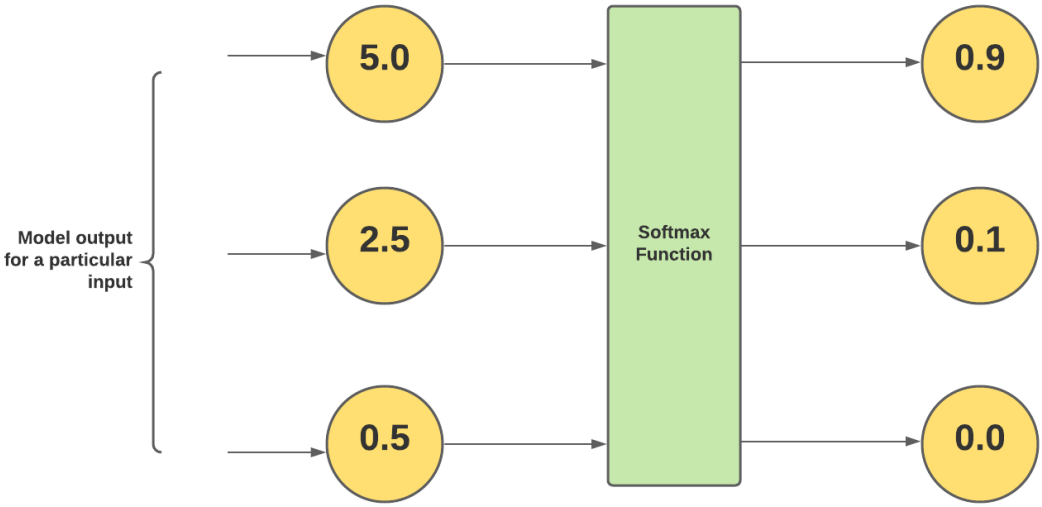



בdropout layer , שכבה המגדירה באופן אקראי יחידות קלט ל0 עם תדירות מסוימת בכל שלב בזמן האימון , מה שעוזר במניעת overfitting , מבטיח תוצאות כלליות יותר.

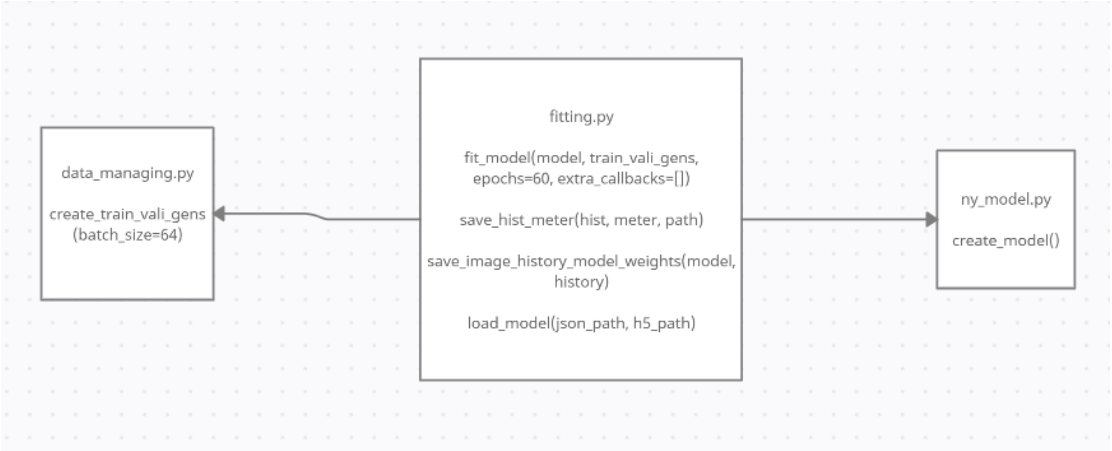


(a) Standard Neural Net (b) After applying dropout.

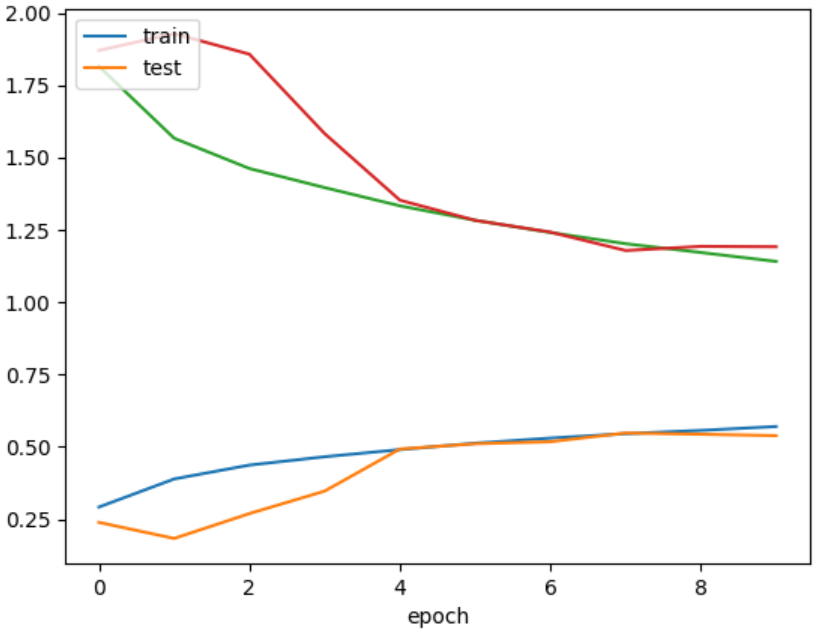
בsoftmax function , משמשת כפונקציית הפעלה בשכבת הפלט של מודלים של רשתות נוירונים המנבאות התפלגות של הסתברויות. הוא מחלק את ההסתברויות של הפלטים כך שהסכום שלהם יהיה 1.




תיאור UML של המודולים בשלב בניית המודל.



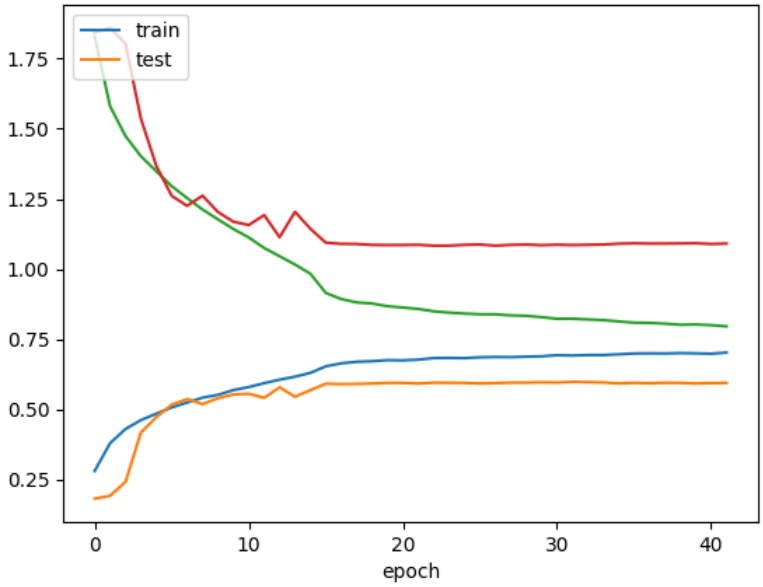
תוצאות של האימון בפעם הראשונה (ניסוי ולכן כמות epochs קטנה, 9)




כאשר הקו הכחול מייצג את הדיוק בtrain והקו הכתום מייצג את הדיוק בtest.

הקו הירוק מייצג את השגיאה בtrain והקו האדום את השגיאה בtest.

תוצאות של האימון בפעם השנייה (כמות epochs גבוהה יותר)




ניתן להבחין בהשפעה של ReduceLROnPlateau (callback שבא עם tensorflow שהעברנו לתהליך האימון), בערך באזור epoch מספר 14, שם ה accuracy וה loss בשלב האימון קפצו לערכים טובים יותר.

| ערך סופי | הייפר-פארמטר |
|---|---|
| 10 | EarlyStopping patience |
| 0.00001 | Minimum lr |
| 0.0002 | Lr התחלתי |
| 0.6 | Dropout rate |
| 64 | Batch_size |

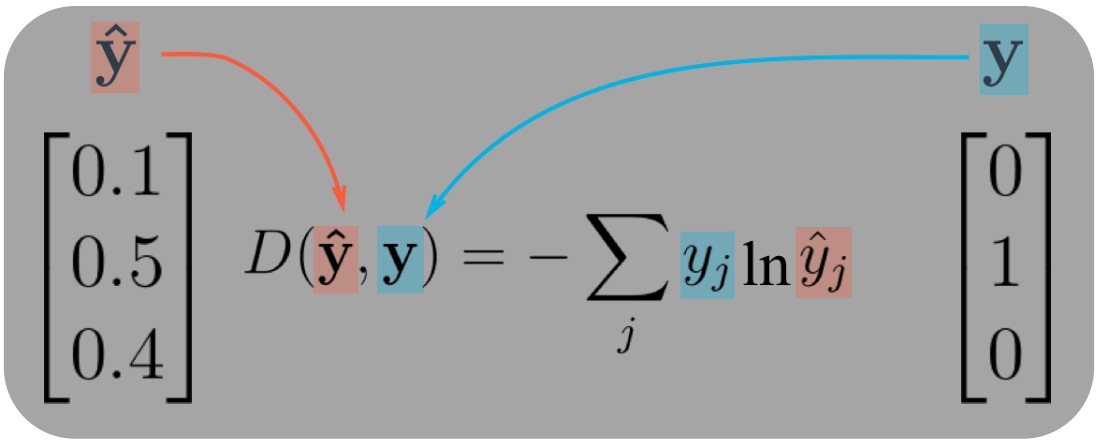
| | |
|---|---|
| Convolutional layer kernel size | 3,3 |
| ארכיטקטורת המודל | מצורף בנספחים |

Loss function פונקציית הפסד זה בעצם פונקציה שאנו מגדירים בכדי לדעת עד כמה טוב הקוד שלנו. ערכה של הפונקציה עולה כאשר המודל מתפקד באופן לא טוב, ויורד כאשר הוא מתפקד באופן מדויק יותר.

אני השתמשתי בפונקציית loss שקוראים לה cross entropy שבעצם בנויה כמו פונקציה של לוגריתם הפוך שהזיזו אותו קצת שמאלה: h(x) = -log(P(x)), בכדיי לא לקבל בח"מ באפס.

בהינתן $n$ מחלקות אפשריות, המסווג מחזיר כפלט עבור כל דוגמת דאטה שהוא רואה וקטור בעל $n$ מספרים, אחד לכל אחת מ-$n$ המחלקות, כאשר המחלקה בעל הציון הגבוה ביותר היא הנבחרת.

ניתן להפוך "ציונים" אלו להסתברויות, וכך להיות מסוגלים לחשב באמצעות מדד ה-Cross Entropy כמה ההסתברויות שקיבלנו מהמסווג קרובות להסתברותיות האמיתיות.




אופטימיזציה הינה עבודה לשיפור ביצועים של מודלים קיימים (אפילו אם יוצרו כרגע, שכן כל הפרמטרים מאותחלים עם ערך התחלתי כלשהו ביצירת מודל חדש).

בשביל להביא למינימום ערך של פונקציית הloss בעלת מספר רב של פרמטרים(פרמטרים של המודל) השתמשתי בפונקציית adam. אלגוריתם משולב, קומבינציה של Mometum ו-RMSProp, אשר מתאים שיעור למידה שונה לכל משתנה ומעדכן אותו לפי הקצב שלו. Adam גם מצליח להתגבר על בעיית ההתכנסות המהירה וגם מאפשר טיפול נכון בתנודתיות המתרחשת באזור המינימום המקומי בעזרת השימוש ב-Momentum. בנוסף, בשלב השני לפני עדכון המשקולת, Adam מבצע Bias-correction עקב הטיה בסביבת 0 המתקבלת בזמן האיטרציות הראשונות עקב קבלת ממוצע נע התחלתי באזור ה-0. התיקון הוא עבור שלב האיטרציות הראשונות עד שהפרמטרים מתכנסים ל-0 ואין צורך עוד בתיקון. בזכות התיקון שמבוצע כאן, Adam מצליח ברוב המקרים לתת תוצאות טובות יותר מ- RMSProp. Adam הוא אחד מהאלגוריתמים החזקים והשימושיים ביותר בתחום בניית רשתות, עקב היכולת שלו להתאים את עצמו לדאטה ושמור על יציבות במהלך האימון.

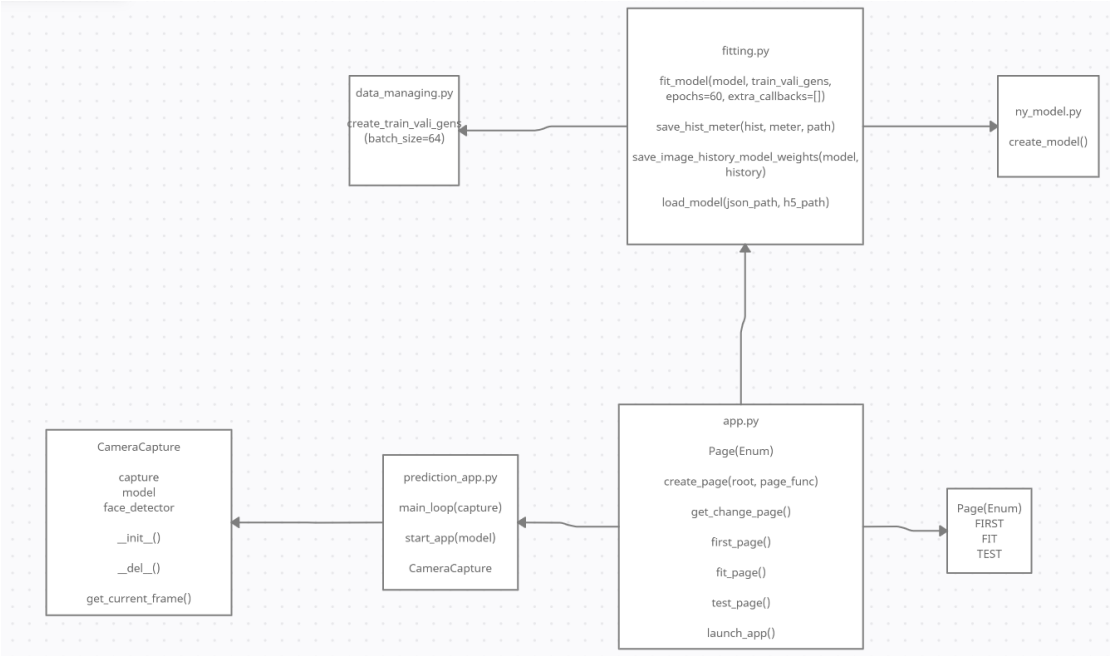
Classic GD זהו האלגוריתם הקלאסי ממשפחת gd אלגוריתם זה מייצג את המבנה הבסיסי לכל אלו הבאים אחריו. בעזרת גזירת פונקציית Loss אותה הגדרנו במודל, לפי כל אחד ממקדמי המשתנים מסומנים כ θ המסבירים בדאטה, ניתן למצוא את משקל הפרמטרים האופטימלי במודל GD. מתקדם בכל איטרציה בכיוון ה Gradient-כפול $\alpha$ עד שהוא מצליח להגיע לנקודה האופטימלית ביותר בהינתן הנתונים ושיעור הלמידה שהכתבנו מראש.

Momentum – ל-SGD יש קושי לנווט בסביבת מינימום מקומי עקב המהירות אליה הוא מגיע, הוא תנודתי מאוד ופחות מכוון. שימוש בשיטת המומנטום עוזר ל-SGD להאיץ בכיוון הנכון ומקטין את התנודתיות באמצעות הוספת פרמטר חיכוך, . בכל איטרציה הוא מחשב ה-Gradient צועד לפיו ואז מתקן באמצעות . בפועל Momentum דואג לעדכן ולתת משקל גדול יותר למשתנים בעלי אותו כיוון כמו כיוון הכללי ולהקטין את משקלם של אלו הנמצאים בכיוון ההפוך. על הפרמטר להיות קטן מ-1.

RMSprop שיטה זו מתגברת על בעיית ההתכנסות המהירה באמצעות חישוב ממוצע משוקלל של ה-Gradient בריבוע, והענקת משקל גדול יותר ל-Gradient האחרון שחושב, ז"א גדול מחצי.

### שלב היישום-

היישום מאמן ועושה testing למודל. ניתן לבחור במודל שאומן מראש, אך לא יהיה לו את האופציה לעשות testing. היישום בנוסף מתחיל את האפליקציה הראשית שחוזה את הבאות הפנים בזמן אמת לפי המצלמה של המכשיר עליו היא רצה, המודל הזה ומודל שאומן מראש שיודע לזהות פרצופים ולתת את מיקומם (על מנת להביא תמונות בפורמט שהמודל שלנו יודע לזהות).




ממשק המשתמש מומש ב tkinter שזוהי ספרייה בפייתון שנותנת לנו ממשק עם ספריית Tk לGUI.

נחשבת כספריית ה GUI הסטנדרטית של פייתון. כך ניתן לעשות GUI במהירות ב python, בלי לדעת את השפה שאיתה מתמשקים מול Tk.

בנוסף, השתמשתי ב OpenCV על מנת לגשת למצלמה ולקבל את המיקום של הפרצופים שלהם נזהה את הרגשות.

למעשה לכל frame בוידיאו ש OpenCV נותן לנו גישה אליו, אנחנו מזהים את הפרצופים (בעזרת מודל שאומן מראש על לזהות פרצופים) ומעבירים את התמונות הספציפיות של הפרצופים למודל שלנו שמזהה הבאות הפנים.

## המדריך למפתח-

| תיאור ותפקיד הפעולה/מחלקה | חתימת הפעולה/מחלקה | מיקום הפעולה/מחלקה (מיקום הקובץ) |
|---|---|---|

| | | |
|---|---|---|
| prediction_app.py | CameraCapture(object)<br>self.capture<br>self.model<br>self.face_detector | מחלקה הזו מייצגת את המצלמה שלנו שעושה predictions.<br>יש לה תכונות.<br>נותן למחלקה גישה - capture לווידאו (כל פעם שצריך לקחת פריים).<br>המודל שלנו שמזהה - model הבעות פנים.<br>מודל מאומן - face_detector מראש שבא עם open_cv.<br>תפקידו לזהות את הפרצופים בתמונה ולתת את מיקומם.<br>__del__(self)<br>פעולה שנקראת כאשר נגמרו האזכורים למקרה מסוים של המחלקה.<br>משחררת את ה capture (ובכך מסיימת את הקלטת הווידאו)<br><br>get_current_frame(self)<br>פעולה פנימית שמחזירה פריים "מתוקן", עם החיזויים. את הפריים הזה ניתן להציג בהמשך למשתמש.<br>היא משתמשת ב capture על מנת להשיג את הפריים הנוכחי מהמצלמה, מעבירה את הפריים לgrayscale, ומשתמשת בface_detector על מנת לזהות את מיקומם ואת גודלם של הפרצופים בפריים.<br><br>הפעולה עוברת על כל פרצוף ברשימה שface_detector החזיר, (שמכילים את גודלם ומיקום של הפרצופים), כדי להוציא מהפריים את החלק של הפרצוף. מתרגם את זה לפורמט שהמודל שלנו יוכל לעשות predictions עליו ( numpy array 1x48x48x1).<br>הפעולה משנה את הפריים כאשר היא מדגישה באמצעות ריבוע את מיקום הפנים, ושמה מעל בטקסט את הprediction.<br><br>אחרי שעברה על כל הפרצופים היא מחזירה את הפריים הערוך, על מנת שנוכל להציג אותו. |
| prediction_app.py | main_loop(capture) | פעולה שמקבלת אובייקט מסוג CameraCapture, ועושה לופ אינסופי (עד שלוחצים על q או |

| | | |
|---|---|---|
| | | escape) בו היא כל הזמן מבקשת מהcapture שלנו לעשות predictions על הווידאו שלה כדי להראות את הפריימים הערוכים.<br>אם הפעולה מזהה בלופ שנלחץ על q או escape (שקוד האסקי שלו 27) הלופ נשבר ולפני שהיא מחזריה, היא מורה לopen cv לסגור את כל החלונות. |
| prediction_app.py | start_app(model) | פעולה שמתחילה את האפליקציה עם המודל שהיא מקבלת. היא מגדירה את הface_detector, ומגדירה מקרה של CameraCapture וקוראת לmain_loop. |
| app.py | create_page(root, page_func) | פעולה זו מקבלת את ה root, של האפליקציה בtkinter, ואת פונקציית העמוד שאותו רוצים ליצור.<br>הפעולה יוצרת עמוד בסיסי ואחרי זה קוראת לפונקצית העמוד שנתנו לה, שהיא תיצור כמובן עמוד יותר מותאם למה שצריך (fit לדוגמה) |
| app.py | get_change_page(root, page_to : Page) | פעולה זו מקבלת את ה root של האפליקציה ב tkinter, ואת העמוד אליו רוצים לעבור. מחזירה פונקציה שעוברת לעמוד הזה. קוראים לה לפני ההרצה על מנת שתייצר פונקציית מעבר בין עמודים לכל כפתור באופן אישי. |
| app.py | changepage() | פעולה זו היא הפונקציה אותה הפונקציה הקודמת מחזירה. היא פונקציית מעבר בין עמודים, היא הורסת את כל ה widgets בעמוד הנוכחי, וקוראת ל create_page עם העמוד החדש ועם הroot שנתנו לפונקציה שיצרה אותה. |
| app.py | first_page(root, page) | פעולה זו היא פעולת העמוד הראשון. היא מקבלת את root האפליקציה של tkinter, ואת העמוד הבסיסי שנוצר כבר. מוסיפה לעמוד הזה את הwidgets שצריך בעמוד הראשון. טקסט שמסביר את מהלך פעולת האפליקציה |

| | | |
|---|---|---|
| | | למשתמש. כפתור שמעביר לעמוד הבא (fitting). כפתור שנותן אופציה להתחיל את האפליקציה עם מודל שאומן מראש. |
| app.py | fit_page(root, page) | פעולה זו היא פעולת העמוד של fitting, מקבלת את שורש האפליקציה של tkinter ואת העמוד הבסיסי שנוצר כבר. מוסיפה לעמוד הזה את הwidgets שצריך בעמוד הזה. טקסט שמסביר את מהלך האפליקציה בעמוד זה. כפתור שמתחיל את התהליך (קורא לפעולה הבאה). כפתור שנותן אופציה להתחיל את האפליקציה עם מודל שאומן מראש. |
| app.py | fit() | פעולה פנימית של הפעולה הקודמת, אחראית על תהליך הfitting ועדכון הui בהתאם. |
| app.py | test_page(root, page) | פעולה זו היא פעולת העמוד של testing, מקבלת את שורש האפליקציה של tkinter ואת העמוד הבסיסי שנוצר כבר. מוסיפה לעמוד הזה את הwidgets שצריך בעמוד הזה. טקסט שמסביר את מהלך האפליקציה בעמוד זה. כפתור שמתחיל את התהליך (קורא לפעולה הבאה). כפתור שנותן אופציה להתחיל את האפליקציה עם מודל שאומן מראש. |
| app.py | test() | פעולה פנימית של הפעולה הקודמת, אחראית על תהליך הtesting ועדכון הui בהתאם. |
| app.py | launch_app() | מתחיל את האפליקציה הראשית של ה predictions עם המצלמה. משתמש במודל שאומן עד עכשיו ואם הוא לא קיים אז לוקח כברירת מחדל מודל מוכן מהקבצים. קורא ל start_app של הקובץ הקודם. |
| model_development\ my_model.py | create_model() | יוצר את המודל עם הארכיטקטורה שבחרנו. |

| | | |
|---|---|---|
| model_development\ fitting.py | fit_model(model : Sequential, train_vali_gens , epochs=60, extra_callbacks=[]) | מתחיל את תהליך הfitting. מקבל מודל, גנרטורים של ה data, מספר epochs, ו callbacks נוספים לתהליך האימון. הפעולה מגדירה את ה callbacks הרגילים שלה, ReduceLROnPlateau (שמקטין את ה learning rate אחרי הרבה זמן ש val_loss אינו משתנה). ו EarlyStopping שעוצר את תהליך האימון אם תוך 10 epochs המדד של val_accuracy אינו משתנה. בנוסף, הגדרנו שיעצור את התהליך וישמור רק את סט הפרמטרים שנתנו את התוצאה הטובה ביותר. |
| model_development\ fitting.py | save_image_history_model_ weights(model, history) | מקבל ושומר את המודל ואת ההיסטוריה של תהליך האימון לקבצים. מחליט לאן לשמור את כל הקבצים לפי משתנים קבועים שהוגדרו בתחילת הקובץ. משתמש בpyplot של matplotlib, על מנת לייצר גרפים של ההיסטוריה ולשמור אותם. משתמש בפונקציות של מודלים בkeras כמו model.to_json() או model.save_weights(). |
| model_development\ fitting.py | save_hist_meter(hist, meter, path) | שומר מדד מסוים מתוך ההיסטוריה של האימון ל path שבוחרים. (לדוגמה accuracy). משתמש ב pyplot של matplotlib. |
| model_development\ data_managing.py | create_train_vali_gens(batch _size=64) | מקבל batch_size ומגדיר אותו להיות הbatch_size שהמודל משתמש בו. משתמש ב ImageDataGenerator של keras על מנת לייצר באופן מלאכותי עוד תמונות בכך שהוא הופך את כולם אופקית ובכך מכפיל את כמות התמונות. מגדיר img_size שזה כמובן גודל תמונה (אצלי x4848). משתמש ב flow_from_directory של ImageDataGenerator של keras, על מנת שיזהה באופן |

| | | |
|---|---|---|
| אוטומטי את מבנה הקבצים וייצר labels שיתאימו לכל מקרה במבנה הנתונים על פי מיקומו בקבצים (לדוגמה אם תמונה מסוימת נמצאת בhappy, הוא ייתן לה label שמתאימה ל happy שזה 3).<br>מגדיר כך גם את הגנרטור של test data וגם של train_data.<br>מחזיר את שניהם בtuple. | | |

## המדריך למשתמש-

יש להוריד ולהתקין את כל החבילות המצוינות כעת:

(לא צריך להתקין tkinter כי היא ספריית ה GUI הסטנדרטית של python שבאה עם כל התקנה של python).

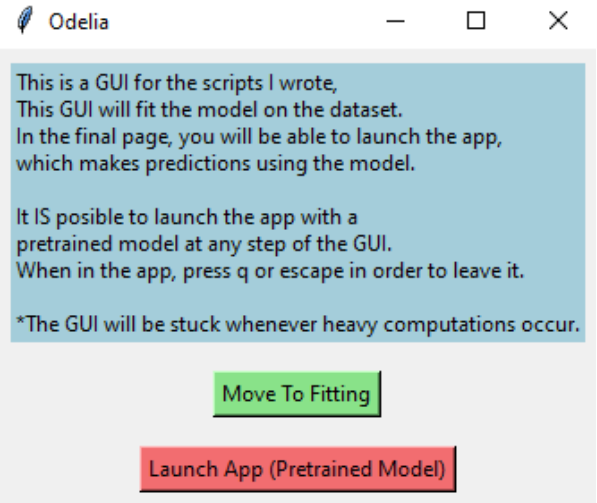
python 3.9

tensorflow 2.9.1

opencv-python 4.5.5.64

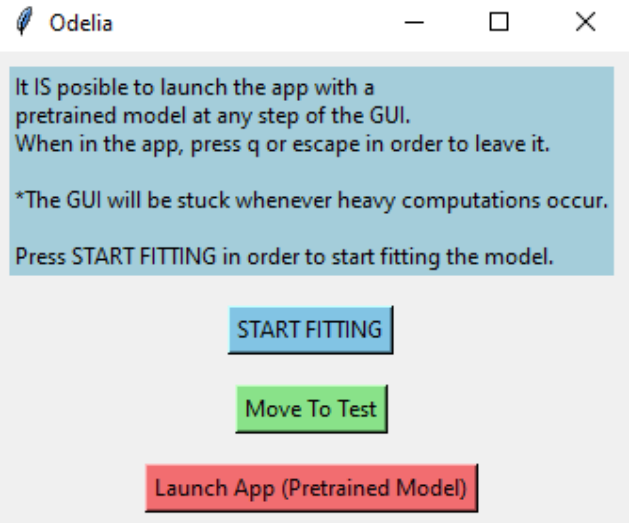
יש להוריד את הקוד של הפרויקט, ולהריץ את קובץ app.py.

זהו המסך הראשון שהמשתמש רואה (בכל שלב ניתן להתחיל את האפליקציה עם מודל מוכן מראש):

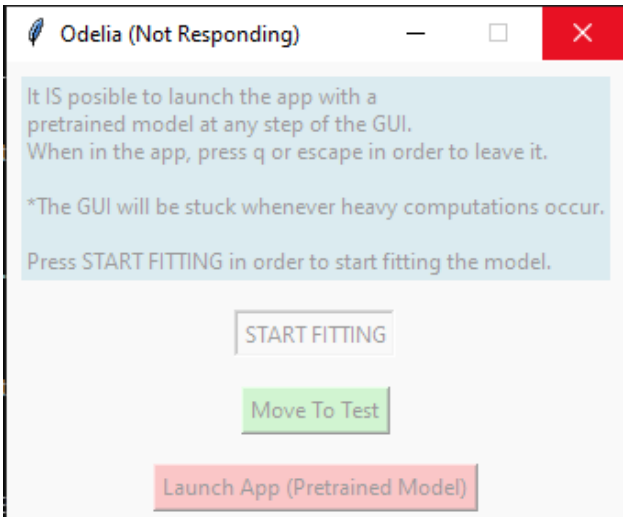



יש לו שתי אופציות, לעבור לתהליך האימון (fitting), או להתחיל את אפליקציית הזיהוי עם מודל שאומן מראש.

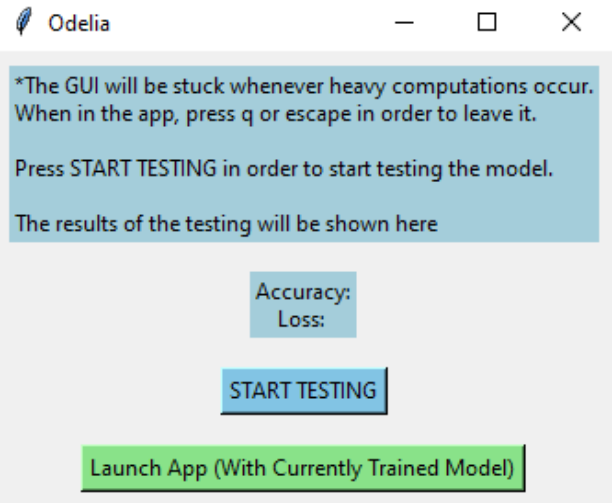
כאשר המשתמש עובר לתהליך האימון, החלון נראה כך:




וכאשר הוא לוחץ על START FITTING, המסך נתקע אך ברקע המודל מתאמן (אפשר לראות את זה ב anaconda prompt)

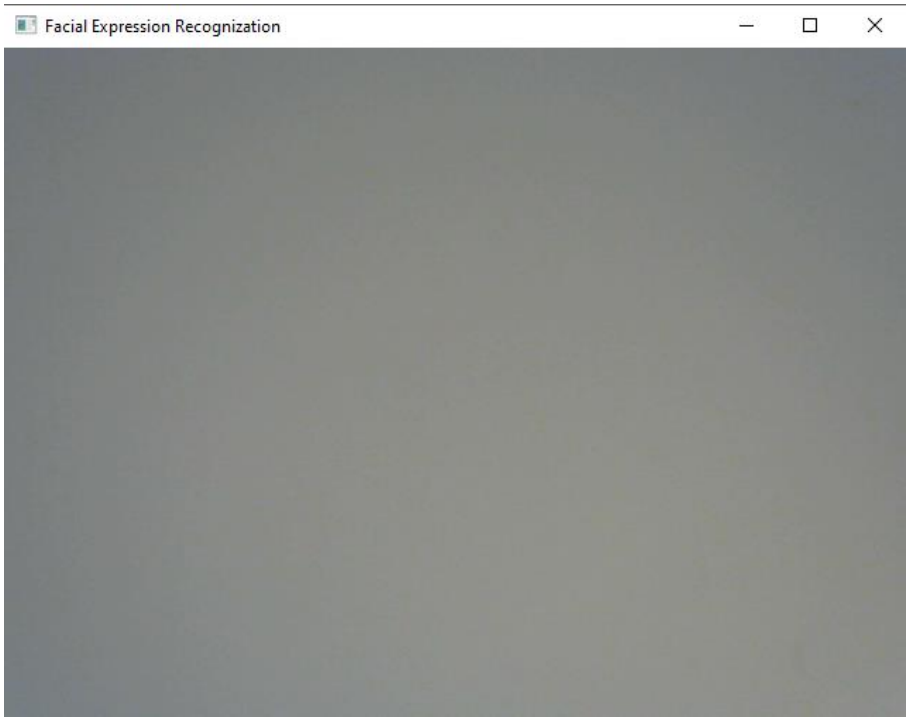



כאשר המשתמש מסיים עם הרצת האימון, הוא יכול ללחוץ על Move To Test, שמעביר אותו לחלון הבדיקה.




לאחר שסיימנו לבדוק את המודל, המדדים של הדיוק והשגיאה יופיעו לנו על המסך (Accuracy, loss).

כרגע אפשר להתחיל את האפליקציה הראשית (של זיהוי הבאות הפנים) עם המודל שאימנו עכשיו (ולא שאומן מראש) וככה היא נראית: (המצלמה שלי מכוסה, לכן אין שום פרצוף בתמונה)




## רפלקציה-

העבודה על הפרויקט הייתה מעניינת ומאתגרת מאוד משום שזה קונספט שלא הכרתי ולא התעסקתי אתו אף פעם. למדתי על ספריות שונות והצלתי לקלוט את השפה (python) בצורה טובה. קיבלתי מהעבודה על הפרויקט ידע נרחב על tensorflow וkeras , ועל כיצד הם עובדים. העבודה לימדה אותי שגם כאשר אני לא יודעת משהו אני יכולה תמיד ללמוד אותו בעצמי גם בלי שילמדו אותי , פשוט מהאינטרנט. במשך הפרויקט למדתי לחפש תכנים ולסנן אותם בצורה טובה יותר. אתגר שהיה לי בעבודה היה שבאיזשהו שלב הגעתי לloss גבוהה מאוד ולא הצלחתי להוריד אותו ולשפר את תוצאות המודל שלי וזה הוריד לי מאוד את המורל והביטחון בפרויקט אך המשכתי לחקור באינטרנט ומצאתי פתרון. בשביל להצליח לעשות את הפרויקט ברמה במבוקשת היה עליי ללמוד המון חומר תאורתי ולרכוש ניסיון מספיק.

מהעבודה והחקר שערכתי אני לוקחת איתי את כל הידע שצברתי ובנוסף את הסקרנות שפיתחתי לתחום החדשני כל כך הזה. תחום הדיפ לרנינג הוא עוד ענק והייתי רוצה ללוד אותו לעומק ולחקור אותו מעבר. אני מאמינה שהתחום הזה תרם רבות לעולם הטכנולוגי ואני מאוד שמחה שיכולתי להתנסות בו בשנה הזאת עם הפרויקט.

לסיכום במהלך הפרויקט בכל פעם למדתי משהו חדש שיוכל לשפר את המודל שלי ובכל פעם התקרבתי לתוצר הסופי שקיוויתי לו, ואני נורא גאה בה שיצרתי בשנה הזאת.

## ביבליוגרפיה-

https://stackoverflow.com/questions/58292617/how-to-have-multiple-pages-in-tkinter-gui-without-opening-new-windows-using-fu

https://en.wikipedia.org/wiki/Tkinter

https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-/networks

/https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks

/https://keras.io/api/layers/reshaping_layers/flatten

/https://machinelearningknowledge.ai

https://www.openbookproject.net/py4fun/gui/tkPhone.html

/https://pythonexamples.org/python-tkinter-button-background-color

## נספחים-

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
batch_normalization (BatchNo (None, 46, 46, 32)        128
_________________________________________________________________
activation (Activation)      (None, 46, 46, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 46, 46, 64)        256
_________________________________________________________________
activation_1 (Activation)    (None, 46, 46, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 64)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 21, 21, 64)        256
_________________________________________________________________
activation_2 (Activation)    (None, 21, 21, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 21, 21, 128)       512
_________________________________________________________________
activation_3 (Activation)    (None, 21, 21, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 128)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 200)               409800
_________________________________________________________________
dropout (Dropout)            (None, 200)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 1407
=================================================================
Total params: 690,055
Trainable params: 689,223
Non-trainable params: 832
```