מבוא לראייה ממוחשבת (22928) – פרויקט גמר

מגיש: דותן אסלמן 301372975

1. הקדמה

בפרוייקט זה התבקשנו לממש מערכת לסיווג פרצופי פנים ולבחון אותה על dataset מוגדר מראש. מבנה האתגר בנוי בצור שהinput של המערכת לתהליך הלמידה הוא רצף פרצופים פרונטליים(mug shot) של 28 אנשים שונים עם תאורה משתנה בלבד. ובשלב הtest על המערכת לזהות תמונות פרצופים שצולמו בשינוי pose ותאורה.

ניתן לראות לשם ההמחשה את הבדלי התאורה בdataset

Figure 1- אדם מהdataset בתאורות משתנות

כמו כן את הבדלי הpose (בתאורה קבועה):

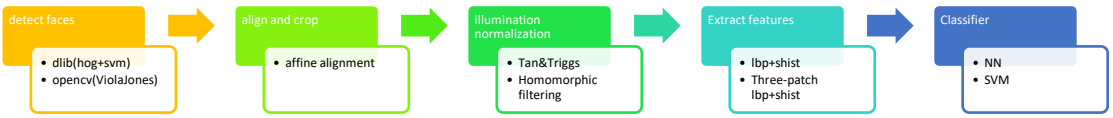
Figure 2 - אדם מהdataset ב poses שונים

תאורה משתנה poseו משתנה הן 2 האתגרים העיקריים היום למערכות סיווג פנים בתרחישי מציאות ולכן מבנה האתגר תואם תרחיש מציאותי בו יש לנו במאגר רק תמונות פרונטליות אך אין לנו תמונות בposes שונים אותן נצלם בתרחיש אמת.

על מנת להתמודד עם אתגרים אלו פיתחתי מערכת בעלת 5 שלבים :

- שלב ראשון : מציאת אזור הפנים על מנת להפריד אותו משאר הרקע ה'מפריע'
- שלב שני: התמודדות עם הpose המשתה ע"י ביצוע טרנספורמציה ליישור הפנים והעברתם לתצורה קנונית
- שלב שלישי : התמודדות עם התאורות המשתנות ע"י ביצוע filter שמצמצם הבדלי תאורה בתמונות
- שלב רביעי: חילוץ פיצ'רים מפרידים(discriminative) מהתמונות
- שלב חמישי: לימוד מודל עבור המסווג

להלן סכמה המתארת את הFlow המלא של המערכת



המסמך מאורגן באופן הבא: בפרק 2 אפרט על שיטות הPreprocess שמימשתי ובדקתי במסגרת הפרויקט (עד לשלב של features extraction), הדיסקריפטורים בהם השתמשתי יתוארו בפרק 3, בפרק 4 אפרט על אלגוריתמי הסיווג שהשתמשתי בהם ובפרק 5 אסכם ואציג תוצאות אמפיריות של ההרצות ואת ה flow הסופי המוצע, פרק 6 הוא בבילוגרפיה של המאמרים עליהם הסתמכתי במימוש המערכת.

2. שלב ה-Pre-Process

המטרה של שלב הpreprocess היא להכין את התמונות לקראת שלב חילוץ הפיצ'רים. שלב זה מחולק לתתי-שלבים בהם : איתור הפנים, יישור התמונה, גזירת אזור הפנים, ונירמול התאורה המשתנה בתמונה.

2.1 'גזירת' אזור הפנים (cropping)

על מנת להתמקד באזור המעניין בתמונות ולבטל את ההפרעות של הרקע רצוי לגזור את אזור הפנים בתמונה בלבד.
את הגזירה ביצעתי ב3 שלבים : איתור מיקום הפנים, ביצוע יישור לייצוג קנוני ולאחר מכן גזירת אזור עניין בתמונה. השלבים מפורטים להלן:

2.1.1 איתור מיקום הפנים

את האיתור ביצעתי באמצעות שילוב של ספריית dlib ו opencv, קודם ניסיתי את הdetector של dlib ובמידה ולא עבד ניסיתי להריץ אותו לאחר ביצוע שיפורי תמונה : שיווי היסטוגרמה ושיווי היסטוגרמה לוקאלי(CALHE) שניהם מתוך opencv, במידה ועדיין לא נמצאו פנים עברתי לשימוש בdetector של Opencv אשר מבוסס על האלגוריתם של ויולה וג'ונס (haar features + adaboost) ובמידה וגם זה לא מצא כלום (הרוב המכריע כן נמצא) לקחתי את אזור מרכז התמונה בתור ניחוש.

2.1.2 יישור הפנים face alignment

המטרה של יישור הפנים היא להביא למצב שכל הפנים בdataset (וכן פנים עתידיות שיצטרכו 'להיבדק') מיושרות באותו אופן. כלומר, העיניים והסנטר נמצאים באותו המקום בתמונה בכל התמונות.

על מנת לבצע את היישור מימשתי 2 שיטות, הראשונה בסיסית ביותר (נקראת בקוד noramlize_alignment_rotate) שהמטרה שלה היא להביא למצב שהקו בין 2 העיניים הוא ישר לאורך התמונה (כלומר – העיניים באותו הגובה) ואת הcrop מבצעים סביב נק' עיגון זו.

עבור השיטה השניה (noramlize_alignment_affine) השתמשתי גם בעיניים וגם בשפה התחתונה על מנת לחשב טרנספורמציה אפינית (הזזה, סיבוב וscale) מהתמונה אל המצב ה'קנוני' שבו העיניים והסנטר נמצאים במקום אחיד. את התצורה הקנונית שמרתי במטריצת קבועים שנקראת TEMPLATE בקוד שלי והיא הושאלה מפרוייקט openface.

פעולת יישור הפנים יכולה תיאורטית לצמצם את ההשפעה של שינויי pose שמאופיינים בdataset הנוכחי ובכך להעלות את הביצועים (בפועל זה לא היה המצב במקרה שלי).

2.1.3 גזירת אזור עניין

לאחר ביצוע 2 שלבים אלו חתכתי מהתמונה את אזור הפנים שזוהו בתוספת שוליים של 25% בכל ציר ולאחר מכן ביצעתי resize לרזולוציה הנדרשת.

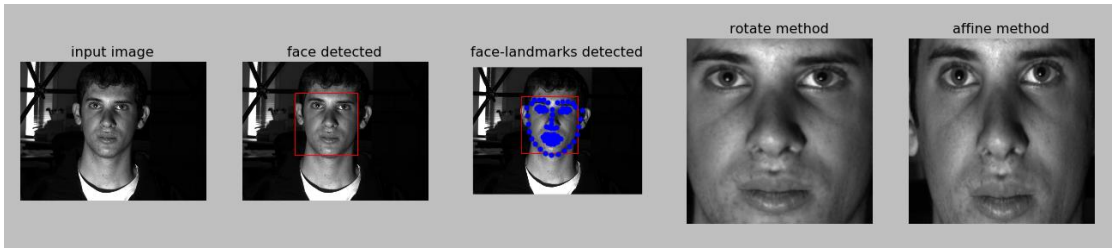


Figure 3 - משמאל לימין - שלבי תהליך הalignment כאשר הימני ביותר הוא יישור affine ומשמאלו rotate

בדקתי גם אפשרויות של עם/ללא שלב היישור וכן ללא crop כלל, מה שבסופו של דבר נתן את התוצאה הטובה ביותר.

2.2 נירמול גווני אפור – Illumination Normalization

המוטיבציה של נרמול גווני האפור היא לבטל (או לצמצם במידת האפשר) את ההשפעה של התאורה המשתנה על ההשוואה בין 2 תמונות, כך ש2 תמונות של אותו אדם בתאורות שונות יהיו דומות ככל האפשר. לשם כך בדקתי 3 שיטות : שיווי היסטוגרמה, homomorphic filtering שהוצג ב[1] וTan&Triggs method שהוצג ב[2] ונקרא ע"ש כותבי המאמר, וכן בדקתי אפשרות לא לבצע נימרול כלל .

להלן פירוט על 2 השיטות הלא מוכרות:

2.2.1 Homomorphic Filtering

כפי שמתואר ב[2] , הרעיון מאוחרי homomorphic filtering מבוסס על Lambertian-reflecance model בו ניתן לתאר תמונה I בתור:

$$I(x,y) = R(x,y)L(x,y)$$

כאשר R – ההחזר מהתאורה ו L עוצמת ההארה בנקודה x,y יחד הם מרכיבים את I שהוא הpixel intensity (כלומר, התמונה)

ניתן להחשיב את R בתור הטקסטורה של התמונה מה שמעיד שכנראה מכיל תדרים גבוהים יותר לעומת L שכן תאורה בתמונה כמעט לא משתנה מאזור לאזור ואם כן אז בהשתנות איטית

מכאן – על מנת להוריד את ה'אפקט' של התאורה עלינו ננסה לצמצם את ההשפעה של מטריצה L ולחזק את ההשפעה של מטריצה R על התוצאה. בשביל לעשות זאת נצטרך להפריד את I ל R וL אך לא ניתן להפריד בניהם באופן ישיר. לשם כך נבצע ln לI וכך נוכל להפוך כפל לחיבור.

$$\ln\big(I(x,y)\big) = \ln\big(R(x,y)\big) + \ln(L(x,y))$$

עכשיו אם נבצע fourier לln של I נקבל

$$F(\ln\big(I(x,y)\big)) = F(\ln\big(R(x,y)\big)) + \mathrm{F}\big(\ln\big(L(x,y)\big)\big) \overset{def}{\Longleftrightarrow} F_R + F_L$$

נגדיר את הפילטר H כך :

$$H(u,v) = (high - low)highbandpass(u,v,c,cutoff) + low$$

כיוון ש*L* בנוי בעיקר מתדרים נמוכים ו*R* מגבוהים, הפילטר *highpass* יכול להפריד בניהם במישור התדר וכך הפילטר *H* מחזק את *R* ומצמצם את *L*. *High* ו *low* הינם קבועים כאשר *low* קטן מ1 ו*high* גדול מ1

נפעיל את *H* ונקבל :

$$S(u,v) = H(u,v)F_R(u,v) + H(u,v)F_L(u,v)$$

כאשר *S* התמונה המסוננת לפני החזרה למישור התמונה (וביטול ה*ln*). נבצע התמרת פורייה הפוכה ו*exponent* ונחזור למצב המקורי לאחר הסינון.

2.2.2 Tan & Triggs

כפי שמתואר ב[1] אלגוריתם של Tan&Triggs לנרמול התאורה בתמונה מחולק לשלבים הבאים :

- Gamma correction – כאשר $\gamma > 1$ תיקון הגאמה מגדיל את התחום הדינאמי של האזורים החשוכים יותר בתמונה(על פני המוארים)

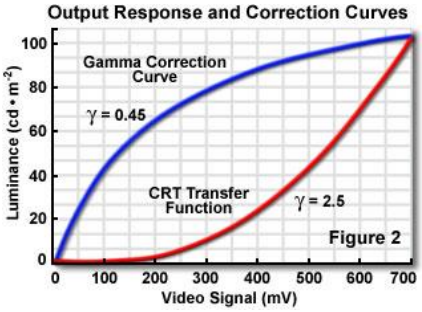


Figure 4 - תיאור של מיפוי רמות אפור ע"י gamma correction

- DoG Filtering – בשלב השני מתבצע מעין high-bandpass filtering באמצעות הטכניקה המקורבת Diffrence of gaussians בו מחושבים 2 מריחות גאוסיאניות אחת בסטיית תקן נמוכה ואחת בגבוהה ומחסרים בין התוצאות, מה שמתקבל הינם התדרים שאבדו בין המריחה הנמוכה לגבוהה וכך נאבד את השפעת התאורה (שנמצאת בתדרים הנמוכים) בלי לקחת יותר מדי רעש(שנמצא בתדרים הגבוהים יותר מהמריחה הנמוכה)
- Contrast Equalization – בשלב האחרון מתבצע נירמול לרמות האפור בתמונה, כיוון שהתמונה עדיין בעלת ערכי קיצון כתוצאה משוליים ו'רעש' שנוסף בשלבים הקודמים הנירמול מתבצע ע"י אקספוננט ש'מחליש' את ההשפעה של ערכים גבוהים. הנירמול מתבצע ב2 השלבים הבאים :

$$I(x,y) \leftarrow \frac{I(x,y)}{(\mathrm{mean}(|I(x',y')|^a))^{1/a}}$$
$$I(x,y) \leftarrow \frac{I(x,y)}{(\mathrm{mean}(\min(\tau, |I(x',y')|)^a))^{1/a}}$$

כאשר a דוחס באופן משמעותי את התחום הדינאמי(בדיפולט =0.1) ו T הוא threshold שמשמש לקטוע ערכים גבוהים מדי שיצאו מהשלב הראשון. (בדרך כלל T=10)

להלן תוצאות של הרצת 2 סוגי הפילטרים על תמונות מהdataset :

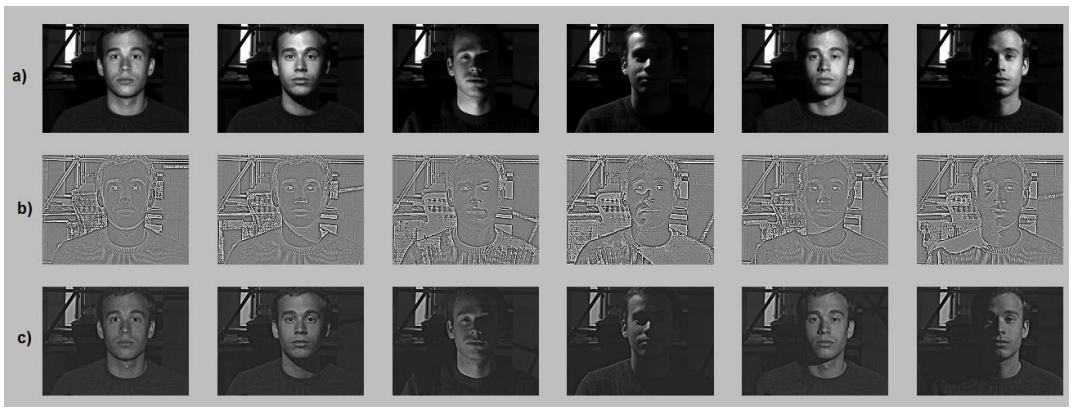


Figure 5 : a התמונות המקוריות. b: התמונות לאחר tan&triggs filter c: התמונות לאחר homomorphic filtering

3 שלב ה-Features Extraction

לאחר ביצוע שלבי הpreprocess עלינו לחלץ פיצ'רים מפרידים מהתמונה ולייצר לכל פרצוף את ה signature שלו, נעשה זאת ב2 תתי-שלבים : הפעלת אופרטור על התמונה (בחנתי מספר ויריאציות של LBP, יפורט בהמשך) ולאחר מכן ביצוע spatial histogram על התמונה וקבלת וקטור תיאור (descriptor) לקראת לימוד ה classifier

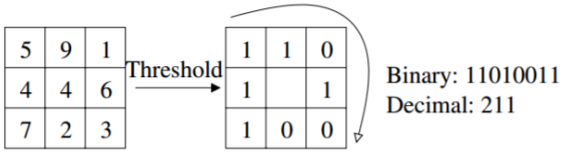
3.1 אופרטור

3.1.1 LBP [3]

LBP – או בשמו המלא Local Binary Pattern הינו descriptor לייצוג של טקסטורות מהיר ומאד פשוט למימוש, כאשר ב[3] הראו שהוא גם יכול להיות מאד שימושי לסיווג פנים.

כדי לייצר LBP יש לעבור על כל פיקסל בתמונה ועל כל פיקסל יש להסתכל על הסביבה שלו ברדיוס r (לצורך הדוגמה נניח r =1) ולחשב את ההפרש בין הפיקסל המרכזי לסביבת השכנים שלו, באמצעות השוואה בין הפיקסל המרכזי לשכניו נקבע רצף בינאי בעל ביט אחד לכל שכן. אם הפיקסל במרכז גדול מהשכן [הביט עבור הצמד יהיה 0 ואם קטן מהשכן הביט יהיה 1

לאחר מכן יש לאגד את הביטים מכל השכנים (במקרה שלנו 8) ביחד לבית אחד וזהו יהיה הערך של הפיקסל בתצורת הLBP שלו,. סדר הביטים הוא **clockwise.** ניתן לראות את התהליך המתואר בתמונה הבאה:



כמובן, במקרה של r=2 יילקחו השכנים במרחק המתאים, ובמידה והם בין 2 פיקסלים תתבצע אינטרפולציה

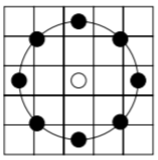


Figure 6 - שכני LBP במקרה שr=2

*את LBP לא מימשתי אלא השתמשתי במימוש קיים בספריית skimage

3.1.2 [4] Rotation invariant LBP

תבנית LBP יכולה לייצר S(מספר השכנים) ערכים שונים בהתאם לסדר הקריאה של הביטים (כל פעם התחלה בביט אחר S פעמים) כאשר כל ערך בהתחלה שונה תואם לזווית שבה אנו מנתחים את הטקסטורה. כלומר – טקסטורות זהות בשינויי סיבובי יתפסו כערכים אחרים בLBP רק בגלל המיקום של התחלת קריאת הביטים.

על מנת להפוך את LBP לrotation invariant יש צורך לקרוא את הביטים בצורה שמבטלת את השפעת הזווית של הטקסטורה.

השיטה, כפי שמתוארת ב[4] היא לבדוק את כל S הוריאציות ולקחת תמיד את זו עם הערך הנמוך ביותר, כך למעשה ייקבע שכל הערכים יתחילו ברצף של 0ים ופעולה זו תבצע את הנירמול לrotation. כלומר :

$$LBP_{P,R}^{ri} = min\{ROR(LBP_{P,R}, i) \quad | \quad i = 0,1,\ldots,P-1\}$$

כאשר ROR הינו circular shift של הביטים i צעדים.

*גם את וריאציה זו של LBP לא מימשתי אלא השתמשתי במימוש קיים בספריית skimage

3.1.3 [5] Three-Patch LBP

עוד וריאציה של LBP הינה הThree-Patch LBP. הרעיון הוא שעבור כל פיקסל בתמונה נלקח patch שהוא נמצא במרכזו (בד"כ בגודל 3 על 3) ובהתאם לרדיוס נבחרים S (במקרה הקלאסי 8) שכנים סביב ה patch המרכזי שלוקחים גם את הpatches סביבם (ניתן לראות באיור מטה), ערך פיקסל במסיכה ייבנה ע"י השוואה של ההפרש בין המרחק(או סימילריות) של patch שכן אחד מהpatch המרכזי לpatch שכן אחר כאשר המרחק בין 2 הpatchים בהשוואה הוא הפרמטר α

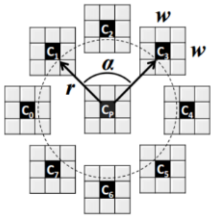


Figure 7 - חישוב פיקסל בtplbp

וכך מתבצע באופן סיבובי על כל השכנים עד לקבלת ערך הפיקסל המרכזי במסיכת התוצאה.

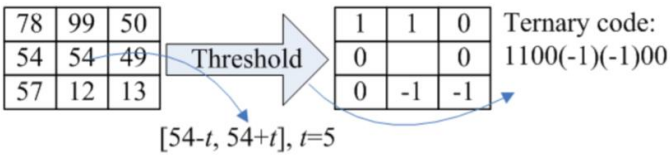
3.1.4 [1] LTP

LTP- Local Trinary Pattens משדרג את LBP באמצעות שימוש ב3 ערכים 1,0 ו -1.

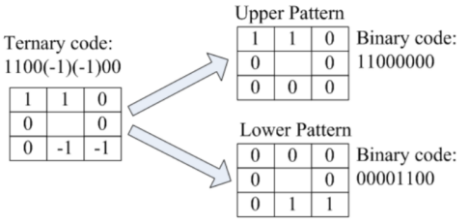
תצורת העבודה שלו דומה לLBP אך הוא מכיל גם פרמטר t וכאשר ההפרש בין פיקסל לשכן שלו קטן מt הביט שיילקח יהיה 0, אם יהיה שלילי הערך יהיה -1 ואם חיובי 1, סף זה תורם לרובסטיות של המסיכה לרעשים קטעים בתמונה.

$$s'(u, i_c, t) = \begin{cases} 1, & u \geq i_c + t \\ 0, & |u - i_c| < t \\ -1, & u \leq i_c - t \end{cases}$$

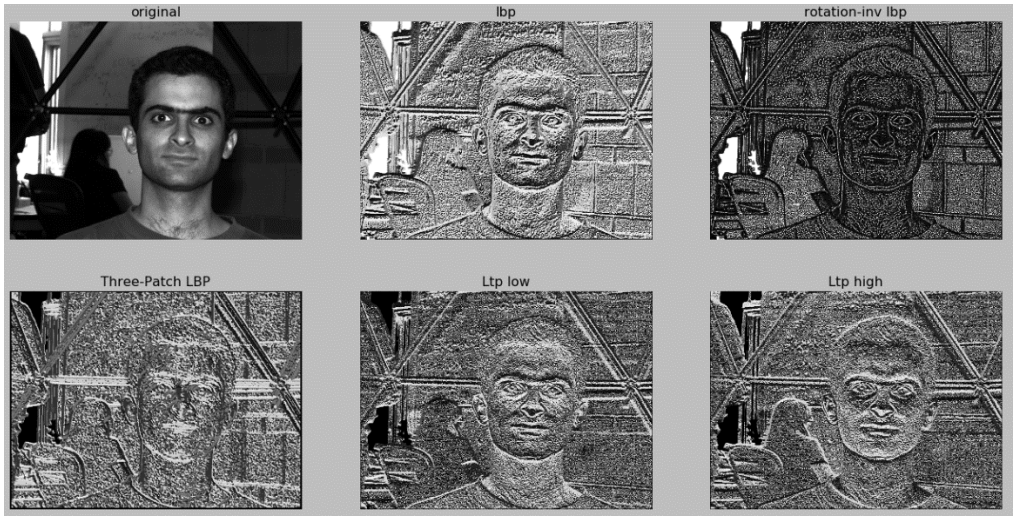
לדוגמה:



ולאחר מכן על מנת לחלץ ערך סופי למסיכה אנו מפצלים את הפיקסל ל2 מסיכות נפרדות low וhigh כאשר הlow מחזיק את ה1- וה 0 (כאשר ה1- הופך ל1 וכל מקום שהיה 1 הופך ל0) והhigh לוקח רק את החיוביים (כך שכל ערך שהיה 1- הופך ל0) ובכך נוצרות 2 מסיכות לכל תמונה. להלן תיאור סכמטי של התהליך:



להלן תוצאות כל אחד מהאופרטורים(יש לשים לב לכותרות)



3.2 Spatial histogram

היסטוגרמת רמות אפור היא כלי מצויין לתאר תמונה בצורה תמציתית אך ייצוג ההיסטוגרמה מאבד לחלוטין את נתונים ה'גאומטריים' (spatial info) של התמונה ולכן "איפה נמצא כל דבר" הולך לאיבוד ונשארת רק ההתפלגות הכללית של רמות אפור. מעין פשרה באמצע של ייצוג חסכוני של התמונה עם היסטוגרמה תוך שימור הנתונים הspatialים באופן גס נקרא spatial histogram. הרעיון הוא לפצל את התמונה לNxM אזורים נפרדים ולחשב היסטוגרמה על כל אזור. את כל ההיסטוגרמות נשרשר יחד לוקטור אחד ארוך והוא יהיה הדסקריפטור של התמונה. נראה לוקטור זה ה-**face signature** של התמונה.

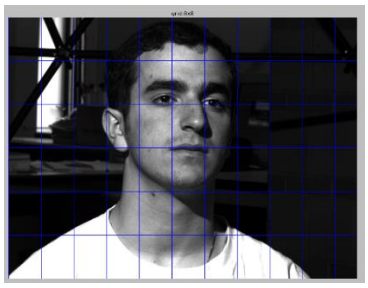


Figure 8 - חלוקת התמונה לאזורים ללא חפיפה וחישוב היסטוגרמה על כל אזור בנפרד

לאחר חישוב כל היסטוגרמה מתבצע נירמול שלה על מנת שתהיה השוואתית ולא תלויה בכמות הפיקסלים בregion. בחנתי מספר שיטות נירמול שונות, hellinger kernel, שיטה ללא שם שמצאתי במאמר [5] ונימרול סטנדרטי לunit vector

3.2.1 Hellinger

נירמול באמצעות Hellinger kernel הומלץ ב[6], הרעיון הוא לבצע נירמול שיאפשר השוואה פשוטה בL2 ושתהייה שקולה להשוואת χ^2 במקרה של נירמול רגיל. הסיבה לכך היא שהשוואה ב χ^2 מתאימה יותר להשוואת היסטוגרמות לעומת L2 שרגיש יותר להבדלים גדולים בשדות ספציפיים

3.2.2 Hassner

מימשתי נרמול קצת 'לא מוגדר' שראיתי ב[5], בנירמול זה יש אכיפה לערך מקסימלי של 0.2 לשדה בהיסטוגרמה ונירמול מחדש במידה ויש כזה. לא היה נימוק על הסיבות אך זה כן שיפר את הביצועים במקרה של tplbp ולכן הוא חלק מהמימוש.

ניתן לראות באמצעות t-SNE שלאחר יצירת ה face-signature לכל תמונה ההפרדה משמעותית ברורה יותר בין האנשים השונים

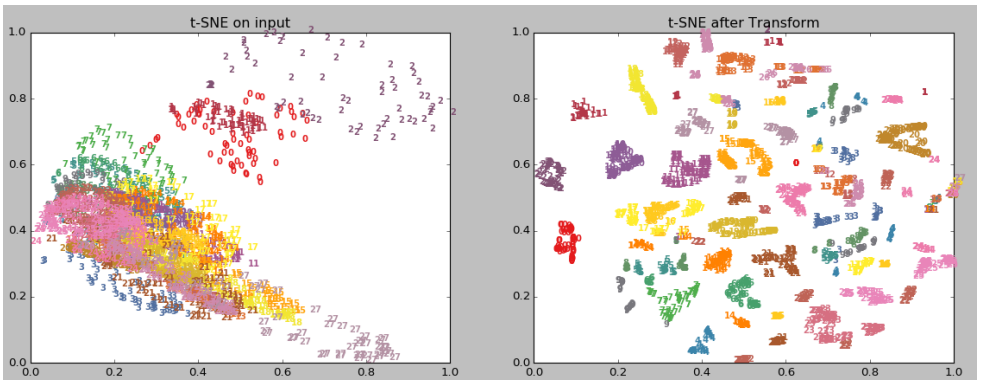


Figure 9 - שמאל : t-SNE על תמונות המקור. ימין : t-SNE על ה faces signatures

4 שלב ה-Classification

4.1 סיווג ראשוני

את הסיווג שעשיתי ב2 שיטות סטנדרטיות :

- **1-NN :** כלומר מציאת הface signature הקרוב ביותר בtrain לתמונה אותה אנו רוצים לבדוק. כיוון שהsignature שלנו הוא וקטור של היסטוגרמות בחנתי בנוסף לפונק' המרחק הדיפולטית של מרחק אוקלידי (L2) גם מרחק χ^2 אשר נותן תוצאות טובות יותר להשוואה של היסטוגרמות כפי שניתן לראות ב[1]
  <u>הערה :</u> כיוון ש1NN הוא לא מבוסס threshold ובעייתי לבצע עליו ROC Curve מימשתי עבורו predict_proba ייחודי מבוסס על המרחק מהשכן הקרוב ביותר בתור מדד של ודאות. כדי שמרחק קרוב ייחשב ודאות גבוהה(כלומר מספר נמוך) הודאות מחושבת בתור 1/distance .
- **SVM :** בדקתי גרעין לינארי ו rbf וכמו כן כיוון שזה multiclass בחנתי בניית multiclass של OneVsRest – שיוצר מסווג בינארי לכל class בנפרד וכן OneVsOne שמייצר מסווג לכל צמד labels ונותן ביצועים טובים יותר אך על פני זמן ריצה גבוה יותר (גם בלמידה וגם בשלב ה test)

4.2 Fine tuning

לאחר ביצוע השלב הסיווג הראשון ביצעתי "fine tune" על צמדים ספציפיים שהיה בלבול נפוץ בניהם, על צמדים אלו הפעלתי svm ליניארי להפריד בין שניהם בלבד.

הצמדים הם שביצעתי להם fine-tune הם (yaleB19-> yaleB17) ו (yaleB34-> yaleB37) ואת הfine-tune ביצעתי רק על הקונפיגורציה שהגיעה לתוצאות הטובות ביותר לפניו.

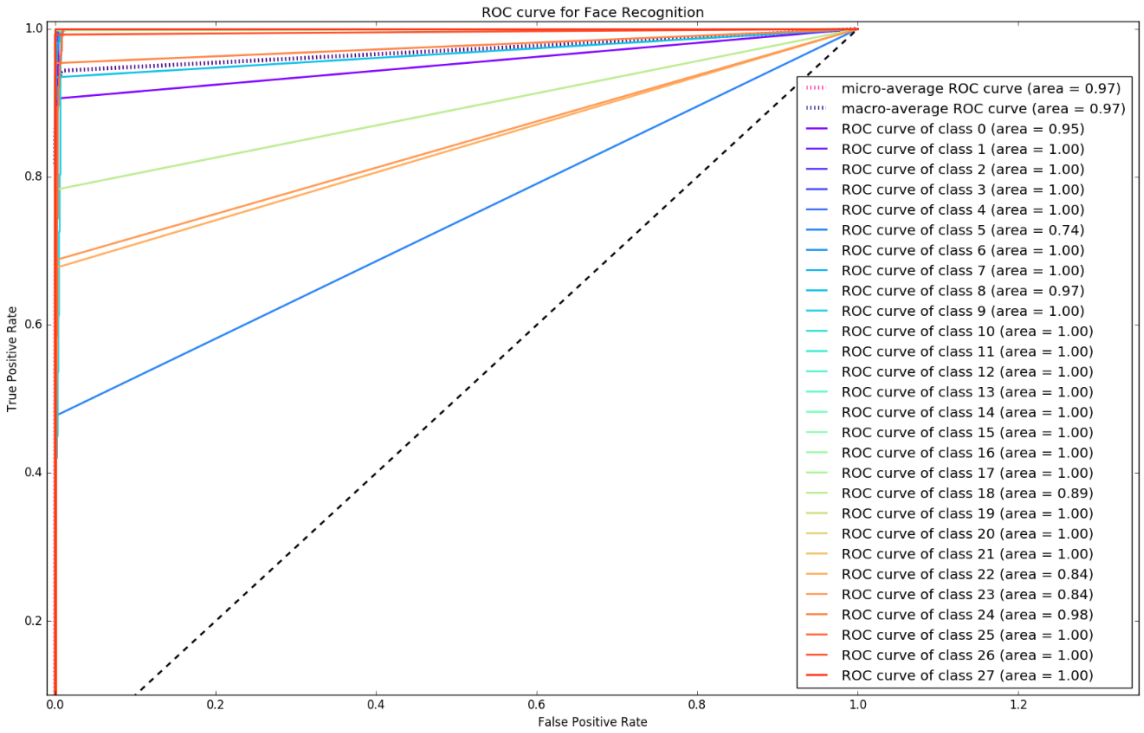
5 תוצאות וסיכום

להלן סיכום של תוצאות ההרצות על מנת למצוא flow אופטימלי, התוצאות הינן חלקיות בלבד וצמצמתי הרצות שהיו 'לא רלוונטיות'

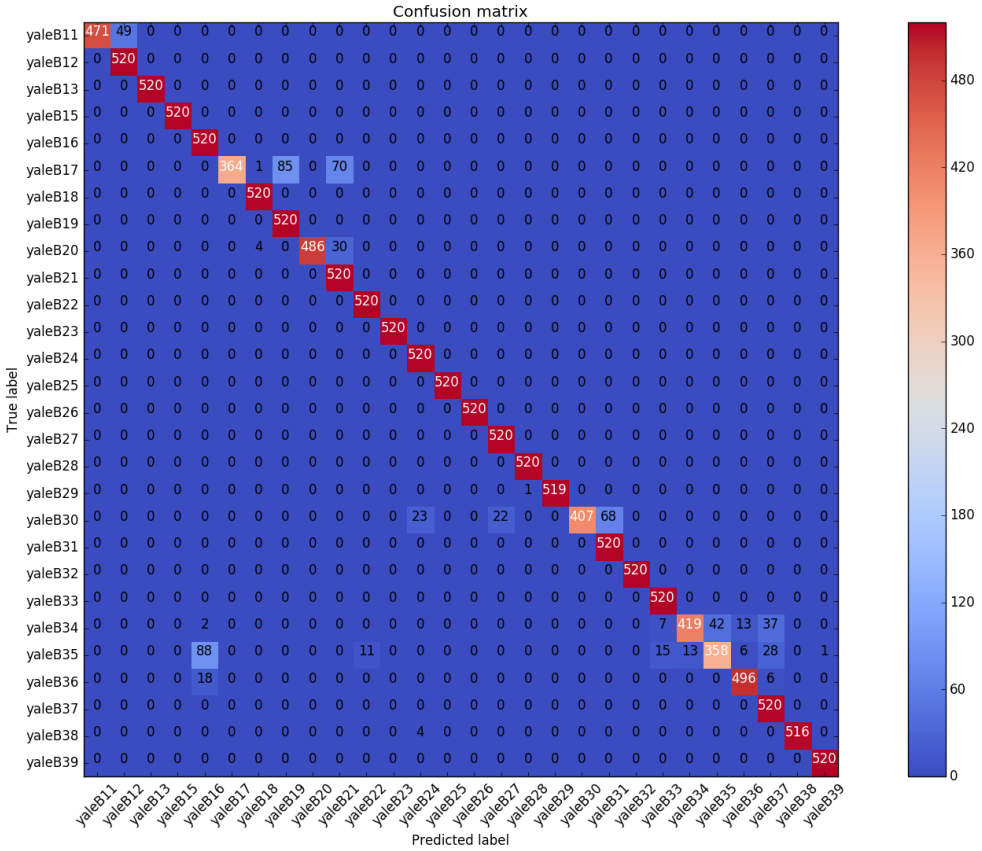
Crop	Align	Illumination normalization	Features	Classifier	Fine-tune	Accuracy score
None	None	None	Lbp 8x8 hist	Linearsvm	No	**0.9088**
None	None	None	Lbp r=2 8x8 hist	1-NN chi2 dist	No	**0.9193**
150x150	Affine	None	Lbp r=1 2x1 hist	1-NN chi2 dist	No	**0.2324**
None	None	None	Tplbp 7x8	1-NN chi2 dist	No	**0.9196**
180x180	None	None	Tplbp 13x11	1-NN chi2 dist	No	**0.7170**
120x120	Rotate	None	Lbp_ror r=2 7x8	1-NN chi2 dist	No	**0.755**
60x60	Rotate	None	Lbp_ror r=2 7x8	1-NN chi2 dist	No	**0.7745**
60x60	Rotate	Tantriggs	Ltp 7x8	1-NN chi2 dist	No	**0.6987**
None	None	None	Lbp-ror r=2 7x8 hist	Linearsvm	No	**0.9111**
None	None	histeq	Lbp-ror r=2 7x8 hist	1-NN chi2 dist	No	**0.9515**
200x200	None	None	Lbp-ror r=2 14x15 hist	1-NN, l2 dist	No	**0.5037**
200x200	None	None	Tplbp 14x14	1-NN, l2 dist	No	**0.6932**
100x100	None	None	Lbp-ror r=2 4x3 hist	1-NN, l2 dist	No	**0.4706**
None	None	None	Lbp-ror r=2 7x8 hist	Rbfsvm C=200	No	**0.8993**
None	None	None	Lbp-ror r=2 7x8 hist	1-NN chi2 dist	No	**0.9432**
None	None	None	Lbp-ror r=2 7x8 hist	1-NN chi2 dist	Yes	**0.9557**

לסיכום בסופו של דבר על אף המאמצים בשלבי הpreprocess כל דבר שנעשה בשלב הpreprocess רק הזיק מה שהשאיר אותי עם תצורה יחסית בסיסית, גם ה svm בכל הקונפיגורציות שניסיתי לא התעלה על 1-NN פשוט. הfinetune אכן עזר ודחף את התוצאות למעלה במעל 1% וסך הכל תוצאת accuracy סופית של **0.9557**

להלן ROC של ההרצה הטובה ביותר (ללא שלב הfine tune), למרות השימוש במרחק מהשכן הקרוב ביותר על מנת לדמות סף עדיין הגרף לא 'רציף' :



וה CONFUSION MATRIX (כולל fine tune):



confusion matrix -10 Figure

בבילוגרפיה:

1. Enhanced Local Texture Feature Sets for Face Recognition Under Difficult Lighting Conditions
2. Face recognition under varying illumination based on adaptive homomorphic eight local directional patterns
3. Face Recognition with Local Binary Patterns
4. Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns Methods
5. Descriptor Based Methods in the Wild
6. Three things everyone should know to improve object retrieval

Appendix – הערות לגבי הרצת הקוד:

- classה שבו מימוש כל האלגוריתמים נמצא בקובץ faceRecognition.py ודוגמת ההרצה נמצאת בdemo.py
- כל הdependencies רשומים בקובץ depends.txt
- כל פונקציה על פרמטרייה מתועדת באופן פרטני בקוד כדי לצמצם עומס על הדו"ח
- על מנת להריץ את demo.py צריך להזין לו בתור basedir את התיקייה בה נמצאת תיקיית ExtendedYaleB ובה כל התמונות
- המודל השמור הינו הקובץ model ויש לשים אותו גם בתיקיה הראשית. בכל אופן0 תהליך הלימוד כולו לוקח דקות בודדות.
- לינק לקבצי המודלים בdropbox :
  https://www.dropbox.com/s/4gk7vzujc0m2gyz/model.zip?dl=0
  קובץ 7z שכולל את המודלים לface detection כמו גם המודל של הface recognition (קובץ בשם model) **חשוב: יש לפרוש את הקובץ ולשים את כל הקבצים בתיקיית ההרצה הראשית**.
- כל הגרפים שמוצגים בדו"ח מוצגים כחלק מ demo.py (למעט t-SNE שההרצה שלו איטית אז השארתי אותו בהערה)
- הרצה עובדת נבדקה רק ב Ubuntu14.04 x64