



המחלקה למתמטיקה
Department of Mathematics

פרויקט מסכם לתואר בוגר במדעים (B.Sc) במתמטיקה שימושית



אפליקציית אנדרואיד לסיווג כלבים לגזעים עפ"י תמונה
באמצעות למידת מטריקת מרחק

תמי מאירוביץ

**Dog Breed Classification Android System using
Distance metric learning**

Tammi Meirovich



**פרויקט מסכם לתואר בוגר במדעים (B.Sc)
במתמטיקה שימושית**



**אפליקציית אנדרואיד לסיווג כלבים לגזעים עפ"י תמונה
באמצעות למידת מטריקת מרחק**

תמי מאירוביץ

**Dog Breed Classification Android System using
Distance metric learning**

Tammi Meirovich

Advisor:

Prof. Vollkovich Zeev

מנחה:

פרופ' וולקוביץ זאב

Karmiel

2015

כרמיאל

תוכן עניינים:

1	הקדמה	1.
2	חלק א' – רקע תאורטי	2.
2	עבודת תמונה	2.1
2	פיקסל	2.2
3	רזולוציה	2.3
3	אלגוריתם SIFT	2.4
3	ייצוג תמונה	2.5
4	אלגוריתם "השכן הקרוב" (K-NEAREST NEIGHBORS)	2.6
5	חלק ב' – בניית המודל המתמטי	3.
5	מהי מטריקה	3.1
5	הגדרה פורמלית	3.2
6	מהי למידת מטריקת מרחק	3.3
6	הגדרה פורמלית	3.4
8	בעיות שקולות	3.5
9	דוגמאות לפסאודו מטריקות	3.6
10	מציאת מטריצה אופטימלית	3.7
11	ניסוח הבעיה המתמטית	3.8
12	הוכחת טענה	3.9
15	מציאת המטריקה	4.
15	בניית המטריקה	4.1
16	האלגוריתם	4.2
18	פסאודו-קוד של שלב ה"למידה" וה"סיווג"	4.3
20	תרשימי הנדסת תוכנה	5.
20	תרשימי זרימה – FLOW CHARTS	5.1
23	תרשים רצף – SEQUENCE DIAGRAM	5.2
24	ממשק משתמש	5.3
28	תוצאות ומסקנות	6.
28	תהליך פיתוח המערכת	6.1
30	תוצאות	6.2
33	מסקנות	6.3
34	ביבליוגרפיה	7.
35	נספחים	8.
35	קטעי קוד	8.1
35	יצירת מאגר מידע	8.1.1
37	סיווג תמונה	8.1.2
38	שרת	8.1.3
40	לקוח	8.1.4

1. הקדמה

בשלבם המתקדמים של התואר חשבתי רבות כיצד אצליח לשלב את הידע שלי בתחום המתמטיקה והנדסת תוכנה עם האהבה הגדולה של חיי, הכלבים. היה לי חשוב, שפרויקט הגמר שלי יהיה משהו מקורי, מעניין, מלהיב ועם זאת פרקטי.

במשך זמן רב לא מצאתי דרך שבה אוכל לאחד בין שלושת העולמות (מתמטיקה, תוכנה והכלבים). וכך, ממש לפני שנדרשתי לבחור נושא לפרויקט, אני וחברי ישבנו בחצר ביתי וניסינו לפענח לאילו גזעים הכלבה המעורבת שלי שייכת. בזמן שניסיתי ללמד אותו על גזעי כלבים פחות מוכרים וסימני ה"היכר" שלהם, פתאום עלה במוחי הרעיון – אכתוב תוכנה שתשייך כלבים לגזעים הדומיננטיים שלהם מבחינה חזותית (מראה חיצוני).

התוכנה תתבסס על תמונות של גזעי כלבים ותשייך את תמונת הכלב לגזעים להם הוא דומה ביותר.

תחום עיבוד תמונה תמיד סיקרן אותי והיה לי מוכר ברמה התאורטית מקורסים שלמדתי כגון עיבוד תמונות ספרתי וממוחשב, תהליכים אקראיים, טורי פורייה והתמרות אינטגרליות. לעומת זאת, מעולם לא הזדמן לי לבחון את הידע בצורה מעשית.

בשבילי, מטרת הפרויקט היא להשתמש בידע ובכלים שרכשתי במהלך התואר, לנתח את התמונה בצורה מתמטית ולתכנן אלגוריתם שיפתור את בעיית הסיווג לגזעי כלבים בצורה מיטבית.

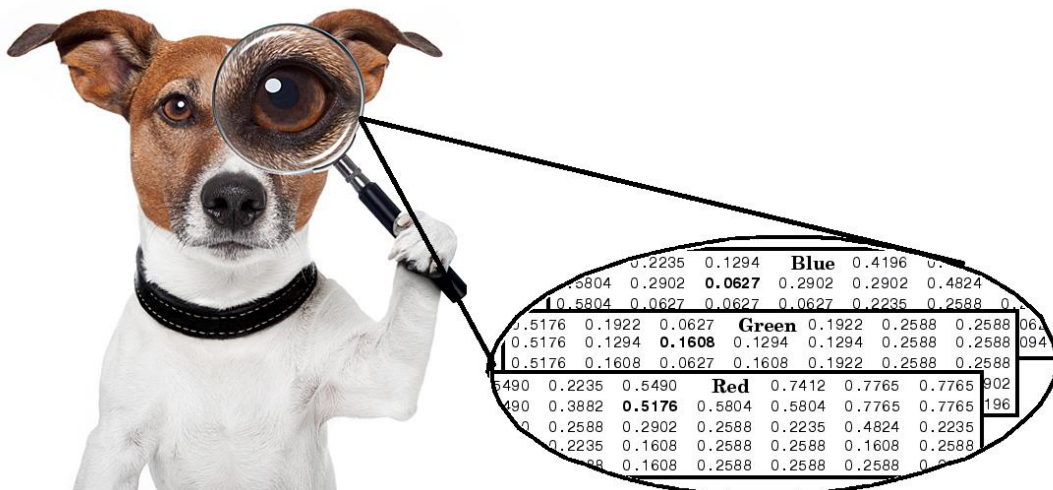
2. חלק א' – רקע תאורטי

2.1. עיבוד תמונה

עיבוד תמונה (Image Processing) הוא תחום טכנולוגי-הנדסי, העוסק בעיבוד של אותות המייצגים תמונה – בין אם היא תמונה בודדת (stills) או תמונה מסרטון וידאו (frame). תחום עיבוד התמונה כולל אלגוריתמים לשיפור תמונות, חילוץ מידע מתוך תמונות, התאמת תמונה, גילוי שינויים, זיהוי תנועה בוידאו, ראייה ממוחשבת ועוד.

2.2. פיקסל

פיקסל (קיצור של Picture Element) היא יחידת מידע גרפית בסיסית, המתארת נקודה בתמונה. החלק הקטן ביותר של הקווים, התווים, והצורות שעל צג המחשב או בתמונה המודפסת. כל תמונה בנויה ממאות אלפי עד עשרות מיליוני נקודות קטנות, ונקודה זו היא יחידת תמונה.



איור 1: ייצוג של תמונה צבעונית במטריצה תלת-ממדית

קביעת העוצמות של מרכיבי הצבע השונים נעשית על ידי השמת המספרים השלמים בין 0 ל 255, כאשר 0 פירושו לא להאיר בכלל את הפיקסל, ו-255 מסמל הארה במלוא העוצמה (לעתים מסמנים עוצמה יחסית על ידי חלוקה ב-255, ואז הערכים הם בין 0 ל-1). התמונה מופיעה בזיכרון המחשב בצורת מטריצה דו ממדית $p[x][y]$ כאשר x, y מציינים את מיקום הנקודה בתמונה דו ממדית וכל איבר במטריצה מייצג פיקסל. כל פיקסל מכיל אינפורמציה על רמת האור שהפיקסל הזה נחשף לה בזמן הצילום. במקרה של טיפול בתמונה צבעונית משתמשים לרוב ב־RGB (Red Green Blue) ואז ישנן שלוש מטריצות:

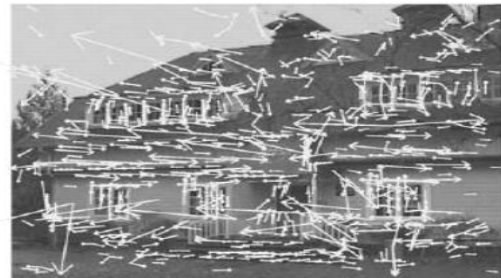
$$p_red[x][y], p_green[x][y], p_blue[x][y]$$

2.3. רזולוציה

עבור ממדי תמונה קבועים, ככל שהתמונה מורכבת מיותר פיקסלים, צפיפות הפיקסלים שלה גדולה יותר. כתוצאה מכך, רמת ההבחנה שלה תהיה גבוהה יותר, ויאמר עליה שהרזולוציה שלה גבוהה יותר. התמונה תכיל יותר פרטים ותהיה חדה, טובה וקריאה יותר.

2.4. אלגוריתם SIFT

Scale-invariant feature transform (או SIFT) הוא אלגוריתם שמזהה ומתאר תכונות ייחודיות ובלתי משתנות בתמונה. האלגוריתם פורסם ע"י David Lowe בשנת 1999. בשיטה זו ניתן לצמצם את ממדי התמונה ע"י ייצוג של התמונה כולה כאוסף "וקטורי תכונות", כאשר כל אחד מהם הוא בלתי משתנה (invariant) לשינוי גודל, כיוון, זווית ראייה וסיבוב, אך משתנה חלקית עבור רעש והצללה.



איור 2: שימוש של SIFT בתמונה שחור-לבן. שימו לב לאופן שבו מיוצגות התבניות בתמונה

כאשר כל התמונות במאגר מידע הן דומות, אין כלל קושי להשוות בין התבניות. אך כאשר התמונות משתנות בגודלן, בסיבוב, בתאורה ובנקודת המבט שלהן ניתן להשתמש ב-SIFT כדי לתאר תכונות חשובות במדויק. תכונות אלו הן ייחודיות מאוד, לכן ההסתברות שניתן יהיה להתאים בצורה נכונה תכונה אחת מול מסד נתונים גדול של תכונות היא גבוהה.

2.5. ייצוג תמונה

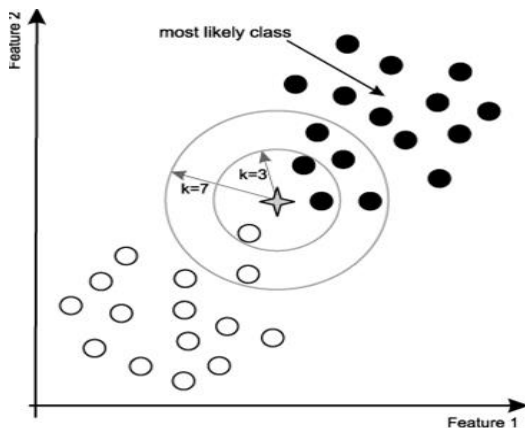
בפרויקט זה התמונות מיוצגות כ"וקטורי תכונות" במרחב אוקלידי רב-ממדי. באלגוריתם הסיווג, תמונות דו-ממדיות מומרות ל"וקטורי תכונות" של d ממדים, אשר לאחר מכן מועתקים למרחב חדש בעזרת טרנספורמציה ליניארית מיוחדת שתואר בהמשך בהרחבה. מכאן ואילך נתייחס אליו כאל "מרחב הגזעים".

ייצוג תמונות כאוסף של "וקטורי תכונות" מסייע להדגיש את המאפיינים החשובים של התמונה. בחירת מספר תכונות ייחודיות מצומצם יעזרו להקטין את ממדי הבעיה משמעותית. צמצום הממדים יפשט את החישובים ובכך יקל על ההשוואה בין שתי תמונות.

2.6. אלגוריתם "השכן הקרוב" (K-Nearest Neighbors)

בהינתן קלט של דוגמה חדשה, תפקידו של האלגוריתם הוא לשייך את הקלט למחלקה הנפוצה ביותר בקרב k השכנים הקרובים (כאשר k מוגדר כמספר חיובי שלם, בדרך כלל מספר קטן). אם $k=1$ האובייקט משויך למחלקה של השכן הבודד הקרוב ביותר.

לפני החלת אלגוריתם K-NN היה עלינו למצוא דרך למצמצם את ממדי התמונה כדי למנוע את ההשפעות של קללת הממד "Curse of dimensionality". קללת הממד אינה בעיה של נתונים בעלי ממד גבוה, זו בעיה משותפת לנתונים ולא אלגוריתם המיושם. הבעיה מתעוררת כאשר נדרשת כמות של זמן או מקום בזיכרון שגדלים באופן מעריכי לכל ממד. פתרון טוב להתמודדות מול "קללת הממד" הוא, בחירת אלגוריתם אחר או pre-processing (עיבוד מוקדם) של המידע לממד נמוך יותר.



איור 3: המחשה של K-NN

3. חלק ב' – בניית המודל המתמטי

3.1. מהי מטריקה

חישוב מרחק "רגיל" בין שתי נקודות, שניתן אף למדוד עם סרגל, הינו למעשה תוצאה של פונקציה המתאימה לכל זוג נקודות במרחב מספר אי-שלילי, ומקיימת כמה תנאים פשוטים. במקרה של מרחק "רגיל" אנו משתמשים בפונקציית מרחק אוקלידית, שמוגדרת באופן הבא: יהיו $x, y \in \mathbb{R}^n$

$$(1) \quad d(x, y) = d_{\text{Euclidean}}(x, y) = \|x - y\|_{\text{Euclidean}} = \sqrt{(x - y)^T (x - y)} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

מטריקה או פונקציית הנה הכללה של פונקציית מרחק אוקלידית, באמצעות המטריקה נוכל להגדיר מרחק בין אלמנטים, במרחב כלשהו. ברגע שהוגדרה מטריקה מסוימת, קנה המידה לא ישתנה.

3.2. הגדרה פורמלית

תהא S קבוצה כלשהי, לא ריקה.
(פסאודו) מטריקה על S היא פונקציה $d : S \times S \rightarrow \mathbb{R}$ המקיימת את התכונות הבאות:

1. אקסיומה 1 – אי-שליליות:

$$d(x, y) \geq 0 \quad \text{לכל } x, y \in S$$

כלומר: מרחק בין שתי נקודות הוא אי-שלילי.

2. אקסיומה 2 – סימטריות:

$$d(x, y) = d(y, x) \quad \text{לכל } x, y \in S$$

כלומר: המרחק הוא סימטרי - המרחק מ- x ל- y הוא כמו המרחק מ- y ל- x .

3. אקסיומה 3 – אי שוויון המשולש:

$$d(x, z) \leq d(x, y) + d(y, z) \quad \text{לכל } x, y, z \in S$$

כלומר: המרחק בין x ל- y הוא הקצר ביותר האפשרי. אם ננסה ללכת ב"דרך עקיפה", דרך איזו נקודת ביניים z , סכום המרחקים "ש"נלך" יהיה גדול יותר.

הקבוצה S יחד עם הפונקציה d נקראת מרחב (פסאודו) מטרי.

במידה ומתקיים $d(x, y) = 0$ אם ורק אם $x = y$ (כלומר במקרה, שבו שתי הנקודות זהות אז המרחק ביניהן הוא תמיד אפס) אזי פונקציה d מגדירה מטריקה והקבוצה S יחד עם הפונקציה d נקראת מרחב מטרי.

3.3. מהי למידת מטריקת מרחק

אנו מתייחסים לסוגיה כיצד נוכל להגדיר את מושג ה"מרחק" $d(x, y)$ בין נקודות x ו y אשר יסתמך על הידע הקודם שהגדירה קבוצת הנקודות ה"דומות" וקבוצת הנקודות ש"אינן דומות", או במקרה שלנו כיצד נוכל לקבוע האם תמונה של כלב מסוים דומה לגזע רוטוויילר, פודל או אולי פיקינו.

לשם כך, יש לבנות את מטריצה A בהתאם לידע הנלמד, כך שניצור בכל פעם מטריקת מרחק רלוונטית למאגר המידע. פונקציית מרחק זו נקראת למידת מטריקת מרחק או באנגלית Distance metric learning.

3.4. הגדרה פורמלית

יהי מרחב S וקטורי מעל שדה $\mathbb{R}^n$ ויהיו x, y וקטורים כך ש $x, y \in S$.
 תהי A מטריצה סימטרית $A \in \mathbb{R}^{n \times n}$, אי-שלילית (כלומר $x^T A x \geq 0$ לכל $x \in \mathbb{R}^n$).
הערה: ידוע כי עבור מטריצה סימטרית כל הערכים העצמיים (ע"ע) ממשיים ו A מוגדרת אי שלילית אם ורק אם כל הע"ע גדולים או שווים לאפס. כמו כן, ידוע כי קיימים n וקטורים עצמיים (ו"ע) אורתוגנליים עבור מטריצה סימטרית.

טענה: הביטוי

$$(2) \quad d(x, y) = d_A(x, y) = \|x - y\|_A = \sqrt{(x - y)^T A (x - y)} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n a_{ij} (x_i - y_i)(x_j - y_j)}$$

מהווה מכפלה פנימית ב- $\mathbb{R}^n$

הוכחה:

$d_A(x, y)$ תקרא מכפלה פנימית על מרחב S אם היא מקיימת שלוש תכונות

1. אקסיומה 2 – סימטריות:

$$\begin{aligned} d_A(y, x) &= \|y - x\|_A = \sqrt{(y - x)^T A (y - x)} = \\ &= \sqrt{(-(x - y))^T A (-(x - y))} = \sqrt{(x - y)^T A (x - y)} = d_A(x, y) \end{aligned}$$

2. אקסיומה 2 – לינאריות: קל לראות שמתקיימת לינאריות.

3. אקסיומה 3 – אי-שליליות:

נתון כי $A \succeq 0$ ומכאן שמתקיים $x^T A x \geq 0$ נגדיר וקטור חדש $w \in \mathbb{R}^n$ כך ש

$w = x - y$ ולכן מתקיים $w^T A w \geq 0$, כעת קל לראות שהאקסיומה מתקיימת.

$$d_A(x, y) = \|x - y\|_A = \sqrt{(x - y)^T A (x - y)} = \sqrt{w^T A w} \geq 0$$

הערה: בדרך כלל דורשים שמטריצה A תהייה חיובית לחלוטין, אבל אי-שוויון קושי שוורץ מתקיים גם עבור A אי-שלילית ומכאן נובע אי שוויון המשולש.

3.5. בעיות שקולות

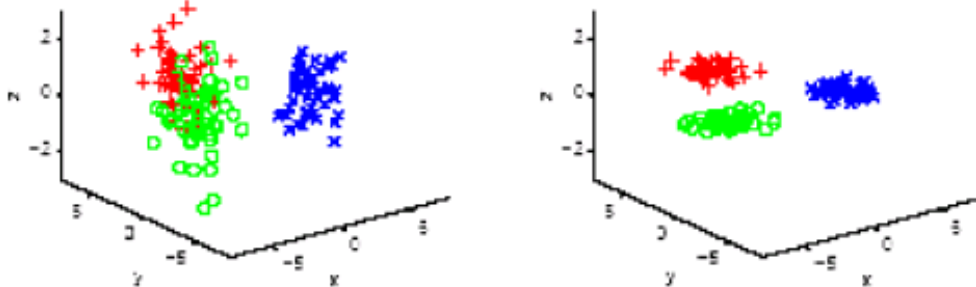
ניתן להראות כי, ע"י ביצוע טרנספורמציה לינארית

$$(3) \quad x' = A^{1/2}x$$

אנו מעתיקים את וקטור x מהמרחב האוקלידי למרחב חדש, שבו הווקטור x' מייצג את ההקשר הנלמד ע"י מטריצה A (כפי שנמחיש בהמשך). במרחב החדש נקודות דומות "יתקרבו" זו לזו בזמן שנקודות שונות "יתרחקו" מהן). לאחר ביצוע ההטלה למרחב החדש נשתמש בפונקציית המרחק האוקלידית הסטנדרטית הידועה לנו כדי למדוד "מרחק". התוצאה שתקבל בדרך זו תהייה זהה לתוצאה בשימוש במטריקת המרחק שהגדרנו לעיל. או במילים אחרות:

$$d_{\text{Euclidean}}(x', y') = d_A(x, y) \quad \text{כאשר } x' = A^{1/2}x \text{ ו- } y' = A^{1/2}y$$

$$\begin{aligned} d_{\text{Euclidean}}(x', y') &= \sqrt{(x' - y')^T (x' - y')} = \sqrt{(A^{1/2}x - A^{1/2}y)^T (A^{1/2}x - A^{1/2}y)} \\ &= \sqrt{(x - y)^T A^{1/2} (A^{1/2}x - A^{1/2}y)} = \sqrt{(x - y)^T A^{1/2} A^{1/2} (x - y)} = \\ &= \sqrt{(x - y)^T A (x - y)} = d_A(x, y) \end{aligned}$$



איור 4: המחשה של ביצוע ההעתקה לינארית $x' = A^{1/2}x$ על וקטורים בעלי שלושה ממדים. בצד שמאל הנקודות מפוזרות במרחב לפני ההטלה (לפני למידת תוכנות מזהות ומבדילות על הקלט) ובצד ימין ניתן לראות כיצד נקודות "דומות" מתקרבות זו לזו ובנוסף מתרחקות מנקודות מקבוצה שונה.

3.6. דוגמאות לפסאודו מטריקות

נבחן מספר דוגמאות בהן מטריצה A היא מטריצה אי-שלילית, כלומר $A \succeq 0$, אך קיימים מצבים בהם מתקיים $d_A(x, y) = 0$ כאשר $x \neq y$.

נגדיר את מטריצה A במישור להיות המטריצה הבאה: $A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$

הערכים העצמאיים של מטריצה A הם $\lambda_1 = 1$ $\lambda_2 = 0$

כל הערכים העצמאיים הינם אי-שליליים ולכן מטריצה A היא אי-שלילית.

נגדיר $x, y \in \mathbb{R}^2$ כלשהם כך ש- $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ ולכן $x - y = \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \end{pmatrix}$

$$\begin{aligned} d(x, y) &= d_A(x, y) = \|x - y\|_A = \sqrt{(x - y)^T A (x - y)} = \\ &= \sqrt{\begin{pmatrix} x_1 - y_1 & x_2 - y_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \end{pmatrix}} = \\ &= \sqrt{\begin{pmatrix} x_1 - y_1 & 0 \end{pmatrix} \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \end{pmatrix}} = \sqrt{(x_1 - y_1)^2} = |x_1 - y_1| \end{aligned}$$

ניתן לראות כי כאשר נרצה לחשב מרחק בין שתי נקודות $x, y \in \mathbb{R}^2$ שנמצאות על אותו ישר האופקי לציר ה- x ($x_1 = y_1$) תמיד יתקיים $d_A(x, y) = 0$, כלומר במקרה של מטריצה A הנוכחית לא ניתן לחשב "מרחק" כאשר שתי נקודות נמצאות על הישר $x_1 = y_1$.

נבחן דוגמא נוספת, נגדיר במרחב מטריצה A למטריצה הבאה: $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

הערכים העצמאיים של מטריצה A הם $\lambda_1 = 1$ $\lambda_2 = 1$ $\lambda_3 = 0$

כל הערכים העצמאיים הינם אי-שליליים ולכן מטריצה A היא אי-שלילית.

נגדיר $x, y \in \mathbb{R}^3$ כלשהם כך ש- $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ ולכן $x - y = \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \\ x_3 - y_3 \end{pmatrix}$

$$\begin{aligned}
 d(x, y) &= d_A(x, y) = \|x - y\|_A = \sqrt{(x - y)^T A (x - y)} = \\
 &= \sqrt{\begin{pmatrix} x_1 - y_1 & x_2 - y_2 & x_3 - y_3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \\ x_3 - y_3 \end{pmatrix}} = \\
 &= \sqrt{\begin{pmatrix} x_1 - y_1 & 0 & x_3 - y_3 \end{pmatrix} \begin{pmatrix} x_1 - y_1 \\ x_2 - y_2 \\ x_3 - y_3 \end{pmatrix}} = \sqrt{(x_1 - y_1)^2 + (x_3 - y_3)^2}
 \end{aligned}$$

ניתן לראות כי כאשר ננסה למדוד מרחק בין שתי נקודות $x, y \in \mathbb{R}^3$ על אותו ישר האופקי לציר

$$d_A(x, y) = 0 \quad (x_1 = y_1, x_3 = y_3) \quad x_2$$

3.7. מציאת מטריצה אופטימלית

נזכר שחישוב ה"מרחק" מתבצע בהתבסס על ידע קודם ולכן המטריקה שלנו היא זו שצריכה לשמר את ההקשר הנלמד. כפי שהראנו, מטריצה A אינה יכולה להיבחר בצורה שרירותית. בחירת מטריצה זו שקולה למציאת מרחב חדש שבאמצעותה מתבצעת ההטלה למרחב חדש, במקרה שלנו קראנו למרחב זה "מרחב הגזעים", שבו נשתמש בפונקציית המרחק האוקלידית הסטנדרטית הידועה לנו כדי למדוד "מרחק".

בפרויקט אנו דנים במטריצה סימטרית A , כיוון שמדובר בתבניות ריבועיות. כדי למצוא את המטריקה הטובה ביותר בהתבסס על הידע הקודם, עלינו לפתור את בעיית האופטימיזציה הבאה. עלינו למצוא מטריצה A , כך שסכום כל המרחקים מהנקודות המוגדרות "דומות" הוא מינימלי בהינתן תנאי התחלה שבו מטריצה A היא מטריצה אי-שלילית ($A \succeq 0$), כלומר לכל וקטור x מתקיים $x^T A x \geq 0$ וסכום המרחקים בין הנקודות המוגדרות ש"אינן דומות" גדול מקבוע כלשהו c , לשם הנוחיות נקבע כי $c = 1$

3.8. ניסוח הבעיה המתמטית

בהינתן סט של m נקודות במרחב האוקלידי $\mathbb{R}^n$, $\{X_i\}_{i=1}^m \subseteq \mathbb{R}^n$, ובהינתן ידע קודם על זוגות של נקודות שקראות "דומות" נגדיר את קבוצה S באופן הבא:

$$S: (x_i, x_j) \in S \text{ כך ש } x_i, x_j \text{ הן נקודות דומות.}$$

על נקודות שקראות "שונות" נגדיר את קבוצה D באופן הבא:

$$D: (x_i, x_j) \in D \text{ כך ש } x_i, x_j \text{ הן נקודות שאינן דומות.}$$

כעת נחפש מטריצה A כך שהמרחק בין הנקודות ששייכות ל- S יהיה מינימלי, כלומר

$$\min_A \sum_{(x_i, x_j) \in S} \|x_i - x_j\|_A^2$$

בכדי לא לקבל את הפתרון הטריטוריאלי $A = 0$, נדרוש שהמרחק בין שאר הנקודות יהיה גדול מקבוע מסוים, כלומר

$$\sum_{(x_i, x_j) \in D} \|x_i - x_j\|_A \geq C$$

נקבע ש $C = 1$, אזי קיבלנו את בעיית האופטימיזציה

$$\begin{cases} \min_A \sum_{(x_i, x_j) \in S} \|x_i - x_j\|_A^2 \\ \text{S.T.} \sum_{(x_i, x_j) \in D} \|x_i - x_j\|_A \geq 1 \\ A \succeq 0 \end{cases}$$

זוהי בעיית אופטימיזציה קמורה לכן קיים לה פתרון.

הערה:

ניתן לחשוב על בעיה פשוטה יותר

$$(4) \begin{cases} \min_A \sum_{(x_i, x_j) \in S} \|x_i - x_j\|_A^2 \\ \text{S.T.} \sum_{(x_i, x_j) \in D} \|x_i - x_j\|_A^2 \geq 1 \\ A \succeq 0 \end{cases}$$

אך מתברר שלבעיה זו קיים פתרון שאינו יעיל.

כיוון שהמינימום המתקבל הוא מטריצה מדרגה אחת. מטריצה מדרגה אחת מעתיקה את $\mathbb{R}^n$ לישר במרחב $\mathbb{R}^n$ וזהו עיוות גס של התמונה.

3.9. הוכחת טענה

נוכיח את הטענה כי הפתרון של בעיה (4) הוא מטריצה בעלת דרגה אחת, כלומר, $A = aa^T$,

כאשר $a \in \mathbb{R}^n$.

לשם כך נדרש הרקע הבא:

- יהיו $a, b \in \mathbb{R}^n$ אזי $trace(ab^T) = a^T b = a \cdot b$. כאן A^T זה מטריצה משוחלפת ו $a \cdot b$ מכפלה סקלרית.

- יהיו $\{x_1, \dots, x_N\}$ וקטורים ב $\mathbb{R}^n$, ונציב $M = \sum_i x_i x_i^T$

אז M מטריצה סימטרית ואי-שלילית ומתקיים:

$$\sum_i \|x_i\|_A^2 = \sum_i x_i^T A x_i = \sum_i (A x_i)^T x_i = \sum_i trace(A x_i x_i^T) = trace(A \sum_i x_i x_i^T) = trace(AM)$$

- זה מסביר את הזהות

$$\frac{\sum_{(x_i, x_j) \in D} \|x_i - x_j\|_A^2}{\sum_{(x_i, x_j) \in S} \|x_i - x_j\|_A^2} = \frac{trace(AM_D)}{trace(AM_S)}$$

$$M_D = \sum_{(x_i, x_j) \in D} (x_i - x_j)(x_i - x_j)^T \text{ ובדומה } M_S = \sum_{(x_i, x_j) \in S} (x_i - x_j)(x_i - x_j)^T$$

- אם A סימטרית, אז על סמך המשפט הספקטרלי

$$A = \sum_{i=1}^n \lambda_i a_i a_i^T$$

כאשר a_i וקטורים עצמיים אורתונורמליים ו λ_i ערכים עצמיים.

- לפיכך מכיוון ש M סימטרית, אז

$$trace AM = \sum_{i=1}^n \lambda_i trace(a_i a_i^T M) = \sum_{i=1}^n \lambda_i trace(a_i (M a_i)^T) = \sum_{i=1}^n \lambda_i a_i^T M a_i$$

- אם A סימטרית, אז הערך העצמי הקטן ביותר המתקבל על ידי המינימום של

Rayleigh-quotient: $\min_{\|x\|=1} x^T A x$, והמינימום מתקבל בווקטור העצמי השייך

לערך העצמי הקטן ביותר. במיוחד, אם כל הערכים העצמיים אי-שליליים, אז הערך העצמי הקטן ביותר הוא

$$\min_{\|x\|=1} \frac{x^T A x}{\|x\|^2} = \min_{\|x\|=1} x^T A x = \min_{\|x\| \geq 1} x^T A x = \lambda_1$$

- וקטורים וערכים עצמיים מוכללים: $Av = \lambda Mv$, כאשר M חיובית לחלוטין, או אי-

שלילית. קיימת מטריצה סימטרית Q כך ש $M = Q^2$. אם M חיובית לחלוטין, אז Q

הפיכה והצבת $u = Qv$ נותנת ש $Av = AQ^{-1}u = \lambda Qu$ או $Q^{-1}AQ^{-1}u = \lambda u$.

- גם במקרה הזה הערך העצמי הקטן ביותר מתקבל על ידי

$$\begin{aligned}\lambda_1 &= \min \frac{y^T Q^{-1} A Q^{-1} y}{\|y\|^2} = \min \frac{(Q^{-1} y)^T A (Q^{-1} y)}{\|y\|^2} = \min \frac{x^T A x}{\|Qx\|^2} = \\ &= \min \frac{x^T A x}{(Qx)^T (Qx)} = \min \frac{x^T A x}{(x^T Q^2 x)} = \min \frac{x^T A x}{(x^T M x)}\end{aligned}$$

הוכחת הטענה: הבעיה (1) שקולה ל

$$\begin{aligned}& \min_{\left\{ \sum_{(x_i, x_j) \in D} \|x_i - x_j\|_A^2 \geq 1, A \succeq 0 \right\}} \sum_{(x_i, x_j) \in S} \|x_i - x_j\|_A^2 = \\ &= \min_A \frac{\sum_{(x_i, x_j) \in S} \|x_i - x_j\|_A^2}{\sum_{(x_i, x_j) \in D} \|x_i - x_j\|_A^2} = \min_A \frac{\text{trace}(A M_S)}{\text{trace}(A M_D)} = \min_A \frac{\sum_{j=1}^n a_j^T M_S a_j}{\sum_{j=1}^n a_j^T M_D a_j}\end{aligned}$$

קשה לעבוד עם הביטוי האחרון מכיוון שווקטורים עצמיים מוכללים אינם אורתונורמליים.

לפיכך, נעשה את המעבר הבא. תהי Q מטריצה סימטרית כך ש $Q^2 = M_D$.

המטריצה Q קיימת מכיוון ש M_D אי-שלילית. אבל כעת נצטרך ש Q תהיה הפיכה ולכן חייב

להתקיים שהמטריצה M_D חיובית לחלוטין.

נציב $b_j = Q a_j$ אז $a_j^T M_S a_j = b_j^T Q^{-1} M_S Q^{-1} b_j$ ו- $\|b_j\|^2 = (Q a_j)^T (Q a_j) = a_j^T M_D a_j$

כעת $Q^{-1} M_S Q^{-1}$ מטריצה סימטרית ואי-שלילית, לכן יש בסיס אורתונורמלי $\{\hat{v}_1, \dots, \hat{v}_n\}$ שמורכב

מווקטורים עצמיים של $Q^{-1} M_S Q^{-1}$. נסדר את הערכים העצמיים בסדר עולה

$0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$, ונציג כל b_j בבסיס של וקטורים עצמיים,

$$b_j = \sum_{i=1}^n \beta_{ij} \hat{v}_i$$

אז מכיוון שהבסיס הוא אורתונורמלי,

$$\sum_{j=1}^n \|b_j\|^2 = \sum_{j=1}^n \sum_{i=1}^n \beta_{ij}^2 = \sum_{i=1}^n \left(\sum_{j=1}^n \beta_{ij}^2 \right)$$

ו

$$\begin{aligned}\sum_{j=1}^n b_j^T Q^{-1} M_S Q^{-1} b_j &= \sum_{j=1}^n \left(\sum_{i=1}^n (\beta_{ij} \hat{v}_i)^T Q^{-1} M_S Q^{-1} (\beta_{ij} \hat{v}_i) \right) = \\ &= \sum_{j=1}^n \left(\sum_{i=1}^n (\lambda_i \beta_{ij}^2) \right) = \sum_{i=1}^n \lambda_i \left(\sum_{j=1}^n \beta_{ij}^2 \right)\end{aligned}$$

לפיכך

$$\frac{\sum_{j=1}^n a_j^T M_S a_j}{\sum_{j=1}^n a_j^T M_D a_j} = \frac{\sum_{i=1}^n \lambda_i (\sum_{j=1}^n \beta_{ij}^2)}{\sum_{i=1}^n (\sum_{j=1}^n \beta_{ij}^2)} \geq \lambda_1 \frac{\sum_{i=1}^n (\sum_{j=1}^n \beta_{ij}^2)}{\sum_{i=1}^n (\sum_{j=1}^n \beta_{ij}^2)} = \lambda_1$$

והמינימום יתקבל כאשר $\beta_{ij} = 0$ עבור $i = 2, \dots, n$

כלומר $b = \beta \hat{v}_1$ וללא הגבלת הכלליות ניתן לבחור $\beta = 1$. נציב $a = Q^{-1} \hat{v}_1$ אז המטריצה שתיתן את המינימום היא

$$A = aa^T = (Q^{-1} \hat{v}_1)(Q^{-1} \hat{v}_1)^T = Q^{-1}(\hat{v}_1 \hat{v}_1^T) Q^{-1}$$

שזוהי למעשה מטריצה מדרגה אחת.

4. מציאת המטריקה

עד כה דנו בפתרון בבעיית אופטימיזציה למציאת מטריצה A כך שיתקבל סכום מינימלי עבור כל המרחקים מהנקודות ה"דומות" וסכום כל המרחקים מהנקודות ה"שונות" יהיה גדול מקבוע כלשהו. כפי שהראנו לבעיה קיים פתרון, אך הוא מצריך חישובים נומריים רבים ומסורבלים וכאשר מדובר על תמונות, שהן בעלות ממד גבוה, הפתרון נהייה מסובך עוד יותר. לכן בחרתי להתמקד בשיטה מתקדמת למציאת מטריקה על ידי שימוש במאמרם של Shijun ו Rong Jin.

בפרויקט גמר שביצעתי בהנדסת תוכנה, חקרתי כיצד ניתן לבנות מטריקה אשר תעזור בסיווג תמונות של כלבים לפי גזעים. בשיטה זו, המטריקה נבנית בצורה דינמית כך שתמונות ממאגר של כלבים נסרקות אחת אחרי השנייה וממופות ל"מרחב גזעים". תמונות מאותו הגזע יקובצו באותו אזור במרחב ובמקביל תמונות מגזעים שונים יורחקו זה מזה.

4.1. בניית המטריקה

Rong Jin ו Shijun Wang מתארים מטריקה A אשר מייצגת את הקשר בין הווקטורים (התמונות) בהתבסס על נתונים סטטיסטיים (ממוצע ומטריצת שונות משותפת – covariance) של וקטורים מאותה המחלקה. הרעיון המרכזי הוא למקם את הווקטורים במרחב חדש, בצורה כזו שבה וקטורים דומים יתקרבו זה לזה והווקטורים השונים יורחקו מהם. נזכיר כי מציאת מטריצת מרחק A אשר באמצעותה ניתן לבצע טרנספורמציה לינארית שתעתיק את הווקטורים הדומים לאותו אזור במרחב היא שווה ערך למציאת מציאת מרחב חדש - מרחב הגזעים.

לכל קבוצה של וקטורים השייכים לאותה המחלקה (מספר תמונות של כלבים מאותו הגזע), נחשב את הווקטור הממוצע ומטריצת השונות המשותפת, אשר מייצגת שונות ודמיון בין וקטורים מאותה המחלקה (כלבים שונים מאותו הגזע בלבד). לאחר קבלת תמונה כקלט אשר יש לסווג, בשלב הראשון נבצע את ההעתקה הלינארית (1). ההעתקה זו, תמקם את הווקטור במרחב שבו, נוכל למדוד "מרחקים" באמצעות פונקציית המרחק האוקלידית. בשלב הבא נמדוד את המרחקים בין תמונת הקלט לבין יתר התמונות במאגר המידע. במקרה הפשוט, התמונה תסווג לגזע אשר המרחק אליו הוא מינימלי. בהמשך נתאר סיווגים למספר מחלקות ע"י שימוש באלגוריתם השכן הקרוב או K-NN.

יהי $X = (x_1, x_2, \dots, x_n)$ אשר מייצג את אוסף התמונות במאגר מידע. כל תמונה במאגר מידע היא $x_i \in \mathbb{R}^m$ ולכן X הוא מטריצה מגודל $m \times n$. יהי $C \in \mathbb{N} > 0$ מספר המחלקות של גזעי הכלבים ו- $Y = (y_1, y_2, \dots, y_n)$ מצינת לאיזה מחלקה משתייך $x_i \in \mathbb{R}^m$. כל y_i הוא וקטור בינארי של C אלמנטים אשר מסמלים האם התמונה משתייכת למחלקה זו או לא, כלומר $y_i = (y_i^1, \dots, y_i^C) \in \{0, 1\}^C$. במאגר המידע החלטתי שכל תמונה תשתייך למחלקה אחת בלבד (הכלבים במאגר הם גזעיים ולכן מתאימים לגזע אחד בלבד). יהי z_k השורה ה- k במטריצה Y , שורה זו מציגה את השיוך (או אי השיוך) של n התמונות במאגר מידע למחלקה ה- k . יהיה s_k מספר התמונות עבור מחלקה k , כלומר $s_k = |z_k|$. יהיו $\bar{x}_k$ ו- $\sum_k$ וקטור הממוצע ומטריצת השונות המשותפת של תמונה הקלט במחלקה ה- k ית, כאשר $k = 1, \dots, C$. $\lambda > 0$ הוא פרמטר ההחלקה.

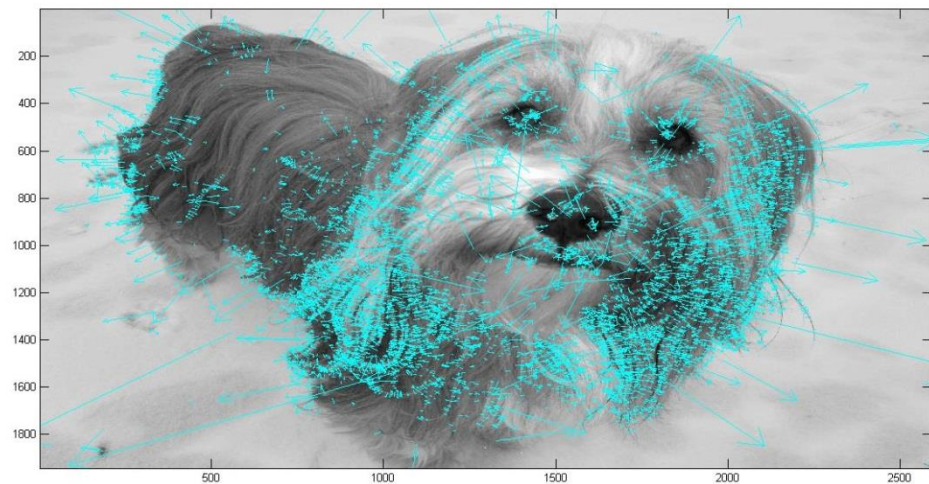
$$\begin{aligned} \bar{x}_k &= \frac{1}{s_k} X_{z_k} \\ \sum_k &= \frac{1}{s_k} \sum_{i=1}^n y_i^k [x_i - \bar{x}_k] [x_i - \bar{x}_k]^T \\ A &= \lambda \left(\sum_{k=1}^C s_k \left[\sum_k + \frac{\lambda \bar{x}_k \bar{x}_k^T}{\lambda + s_k} \right] \right)^{-1} \end{aligned}$$

4.2. האלגוריתם

כדי לסווג תמונות של כלבים לגזעים תכננתי אפליקציית אנדרואיד בעלת ממשק ידידותי למשתמש, המאפשרת למשתמש להעלות תמונה מגלריות תמונות או לצלם תמונה ברגע אמת. לאחר מכן, המשתמש נדרש לבחור מספר "נקודות מפתח" שהוגדרו מראש (העיניים, האוזניים והאף). נקודות אלו נבחרות ע"י שימוש במסך המגע. התמונה והמידע יועבר לשרת לעיבוד ע"י סקריפטים של MATLAB. לאחר שיבוצעו החישובים, רשימה של גזעים תשלח בחזרה אל המשתמש. הרשימה תכלול את שלושת הגזעים שהכי דומים לגזע הכלב המופיע בתמונת הקלט שהתקבלה מהמשתמש.

החלק המורכב והקשה ביותר בתהליך המתואר לעיל הינו מציאת נקודות תואמות בין תמונת הקלט לכלל התמונות במאגר המידע. במקרה זה רצוי להפעיל אלגוריתמים אוטומטיים של "ראיה ממוחשבת" למציאת נקודות עניין, ואחת מהשיטות בקטגוריה זו היא להשתמש באלגוריתם SIFT (scale invariant feature transform). אלגוריתם זה עובד בשיטה של "טשטוש" התמונה בסקלות שונות ומציאת סקלת הטשטוש והמיקום בתמונה בהם העצם עובר טשטוש לכדי כתם יחיד עם עוצמה ההולכת ודועכת ממרכזו. לדוגמה, במבט מקרוב על כדור

הארץ ניתן להבחין בפרטים כמו יבשות, הרים וימים, אך אם נטשטש את התמונה בשלבים לבסוף נקבל כתמים חומים באזורי היבשה וכתמים כחולים באזורי הים. מרכזו של כל כתם חום יהיה מרכז של יבשת (שהוא בעצם מרכז נקודת העניין) ואילו המעגל התוחם את הכתם יגדיר סקלה של גודל הכתם (תחום העניין). כדי לתאר את "אופיה" של כל נקודת עניין מחלקים את תחום העניין לרשת של 4×4 ומחשבים היסטוגרמות של גרדיאנטים עבור כל נקודה ברשת, כך שעבור כל נקודת עניין מתקבל תאור הכולל את מיקומה, תחום העניין שסביבה (סקלה) ותיאור (descriptor) בן 128 נתונים המתאר את פילוג הגרדיאנטים בתוך תחום העניין. נקרא ל-descriptor "וקטור תכונות".



איור 5: הרצת אלגוריתם SIFT על תמונה של כלב

בשלב הבא נשתמש באלגוריתם שנקרא IGML. באמצעות אלגוריתם זה נוכל למצוא דרך להפוך את וקטורי התכונות שבמאגר מידע לוקטורי תכונות חדשים במרחב הגזעים. זוהי שיטה מגאומטריית האינפורמציה (Information Geometry) ללמידת מטריקת מרחק אשר מתארת בצורה מיטבית את היחסים בין פיקסלים בווקטורים והמחלקה המתאימה להם. בעזרת מטריקה טובה, יש סיכוי טוב ששני וקטורים דומים (מאותה מחלקה) ימוקמו קרוב אחד לשני במרחב הגזעים. המטרה של המטריקה היא בחישוב המרחק בין שתי תמונות במרחב הגזעים.

בשלב הראשון של האלגוריתם, שלב הלמידה, בונים מאגר של תמונות כלבים המחולקים לפי גזעים. הופכים את כל התמונות במאגר לווקטורי תכונות ולאחר מכן מחושבת מטריקת המרחק – מטריצה A ע"י אלגוריתם IGML. בעזרת A ממקמים את כל וקטורי התכונות שקיבלנו במרחב הגזעים החדש.

בשלב השני של האלגוריתם, שלב הסיווג, המשתמש שולח תמונה של כלב למערכת. בנוסף, המשתמש מסמן בעזרת האפליקציה נקודות עניין בתמונה כגון עיניים, אוזניים ואף. הופכים את התמונה לווקטור תכונות ובעזרת A (שחושבה מראש) ממקמים את הווקטור במרחב הגזעים. בהנחה שווקטור זה יהיה קרוב לווקטורים דומים מאותו גזע במאגר, מחשבים את המרחקים בין

הווקטור שלנו לכל שאר הווקטורים במאגר. K-NN הוא אלגוריתם לסיווג ע"י הצבעת רוב שמתבצעת ע"י כל השכנים כאשר שכנים נקבעים ע"י המרחק האוקלידי ביניהם במרחב הגזעים.

4.3. פסאודו-קוד של שלב ה"למידה" וה"סיווג"

בכל תמונה במאגר המידע נבחרו בצורה ידנית 5 "נקודות עניין" שהוגדרו מראש. נקודות אלו מאוכסנות במערך בגודל שבו עמודה מייצגת תמונה. יהי $x'_j \in \mathbb{R}^m$, $j \in (1,5)$, המייצג 5 וקטורי תכונות לוקאליים שמתארים 5 "חלונות", אחד עבור כל נקודת עניין שנבחרה בצורה ידנית. לאחר שנשרשר את חמש וקטורי התכונות הלוקאליים נגדיר את $x_i \in \mathbb{R}^{5m}$, $i \in (1,n)$ להיות וקטור התכונות שמייצג את התמונה ה- i -ית במאגר המידע. יהי $X = (x_1, x_2, \dots, x_n)$ אוסף וקטורי התכונות של n תמונות. כל $x_i \in \mathbb{R}^{5m}$ ולכן X הוא מטריצה בגודל $m \times n$. יהי C מספר הגזעים (מחלקות) ויהי $Y = (y_1, y_2, \dots, y_n)$ כאשר כל y_i הוא וקטור בינארי של C אלמנטים אשר מסמלים האם התמונה משתייכת למחלקה זו או לא. נגדיר את p כווקטור התכונות של תמונת הקלט.

Learning stage

1. Set $i = 1$
2. For each dog breed $K \in (1, C)$ in the database do:
 - 2.1 For each image in the breed group do:
 - 2.1.1 Extract face image from database
 - 2.1.2 Convert to 2D greyscale matrix
 - 2.1.3 Turn matrix into feature vector x_i :
 - 2.1.3.1 Initialize feature vector x_i
 - 2.1.3.2 Extract 5 windows from image according to keypoints
 - 2.1.3.3 For each window $j \in (1,5)$ do:
 - 2.1.3.3.1 Calculate SIFT local feature vector x'_j
 - 2.1.3.3.2 Concatenate local feature vector x'_j to the main feature vector x_i .
 - 2.1.4 Set feature vector as column i in matrix X .
 - 2.1.5 Set the label in vector y_i to match the breed k .
 - 2.1.6 Increment i
 - 2.2 Calculate the mean matrix for breed k : $\bar{x}_k = \frac{1}{s_k} X_{z_k}$
 - 2.3 Calculate the covariance matrix for breed k : $\Sigma_k = \frac{1}{s_k} \sum_{i=1}^n y_i^k [x_i - \bar{x}_k][x_i - \bar{x}_k]^T$
 - 2.4 Smooth covariance matrix using closest correlation matrix method
3. Calculate the distance matrix $A = \lambda \left(\sum_{k=1}^c \left[\Sigma_k + \frac{\lambda \bar{x}_k \bar{x}_k^T}{\lambda + s_k} \right] \right)^{-1}$
4. Store matrix A for later use.
5. Calculate $X_{transformed} = A^{1/2} \cdot X$ to position each feature vector column in X to its new location in the Breed Space.
6. Store $X_{transformed}$ matrix for later use.

Classification stage

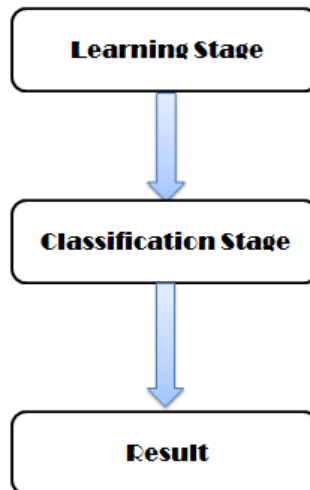
1. Get input image and keypoints
2. Convert to 2D greyscale matrix
3. Turn matrix into feature vector p :
 - 3.1 Initialize feature vector p
 - 3.2 Extract 5 windows from image according to keypoints
 - 3.3 For each window $j \in (1,5)$ do:
 - 3.3.1 Calculate SIFT local feature vector p_j
 - 3.3.2 Concatenate local feature vector p_j to the main feature vector p .
4. Calculate transformation $p \rightarrow A^{1/2}p$ to position the feature vector to its new location in the Breed space.
5. Use K-NN where K=6 to determine closest neighbors.
 - 5.1 Compute the Euclidean distance $d(p, x_i)$ between p and all $x_i \in X$
 - 5.2 Sort all points x_i according to the distance $d(p, x_i)$
 - 5.3 Select the first k points from the sorted list and put them in group G
 - 5.4 classify by a majority vote of the selected k points
6. Return the 3 highest ranked breeds list to user.

5. תרשימי הנדסת תוכנה

5.1. תרשימי זרימה – Flow charts

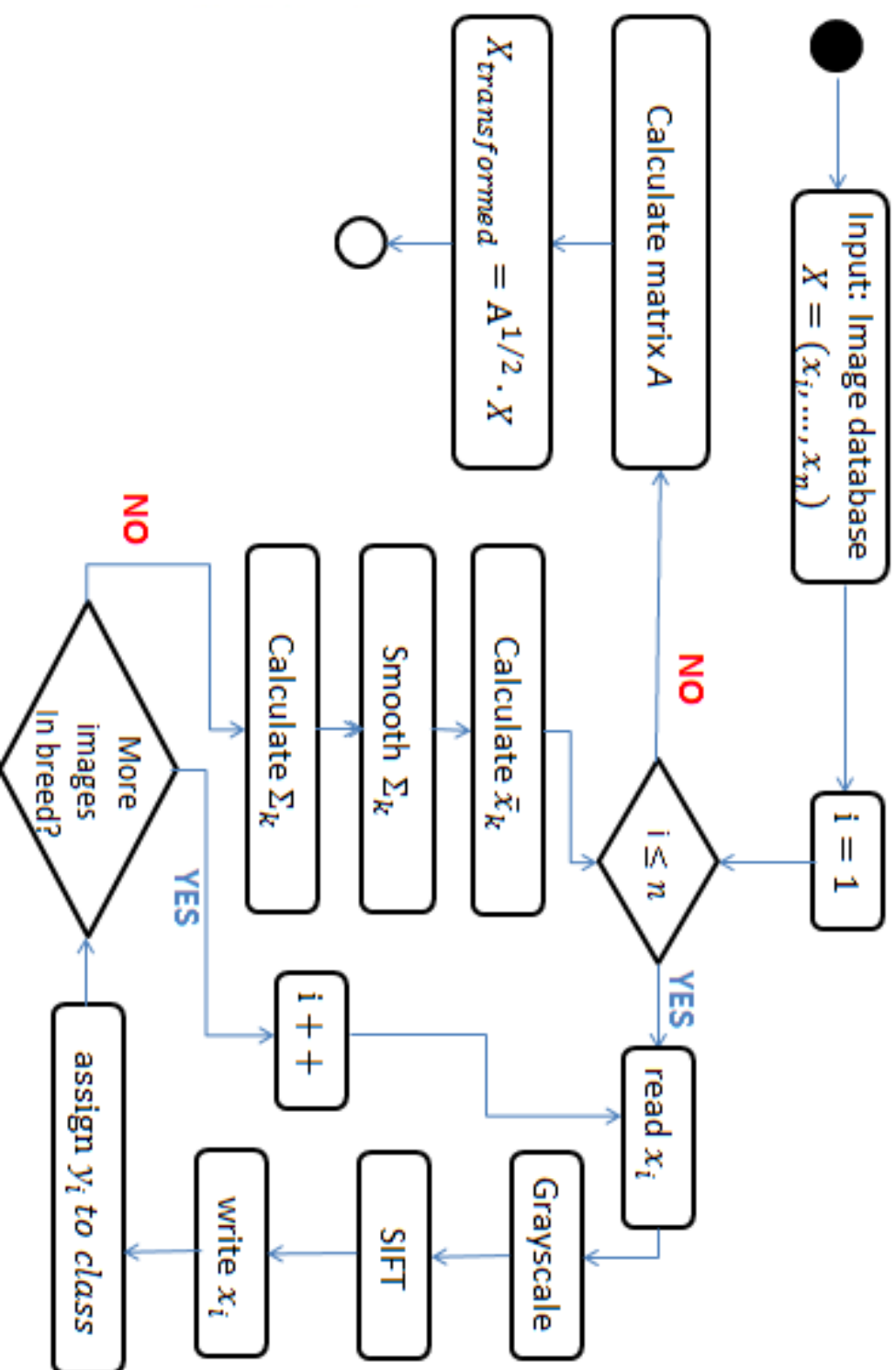
תרשימי הזרימה עוזרים לתאר את סדר פעולות המערכת ובכך מפשטות את התכנון שלה.

מצורפים מספר תרשימי זרימה מעולם הנדסת התוכנה.



איור 8: תרשים זרימה של האלגוריתם הכללי

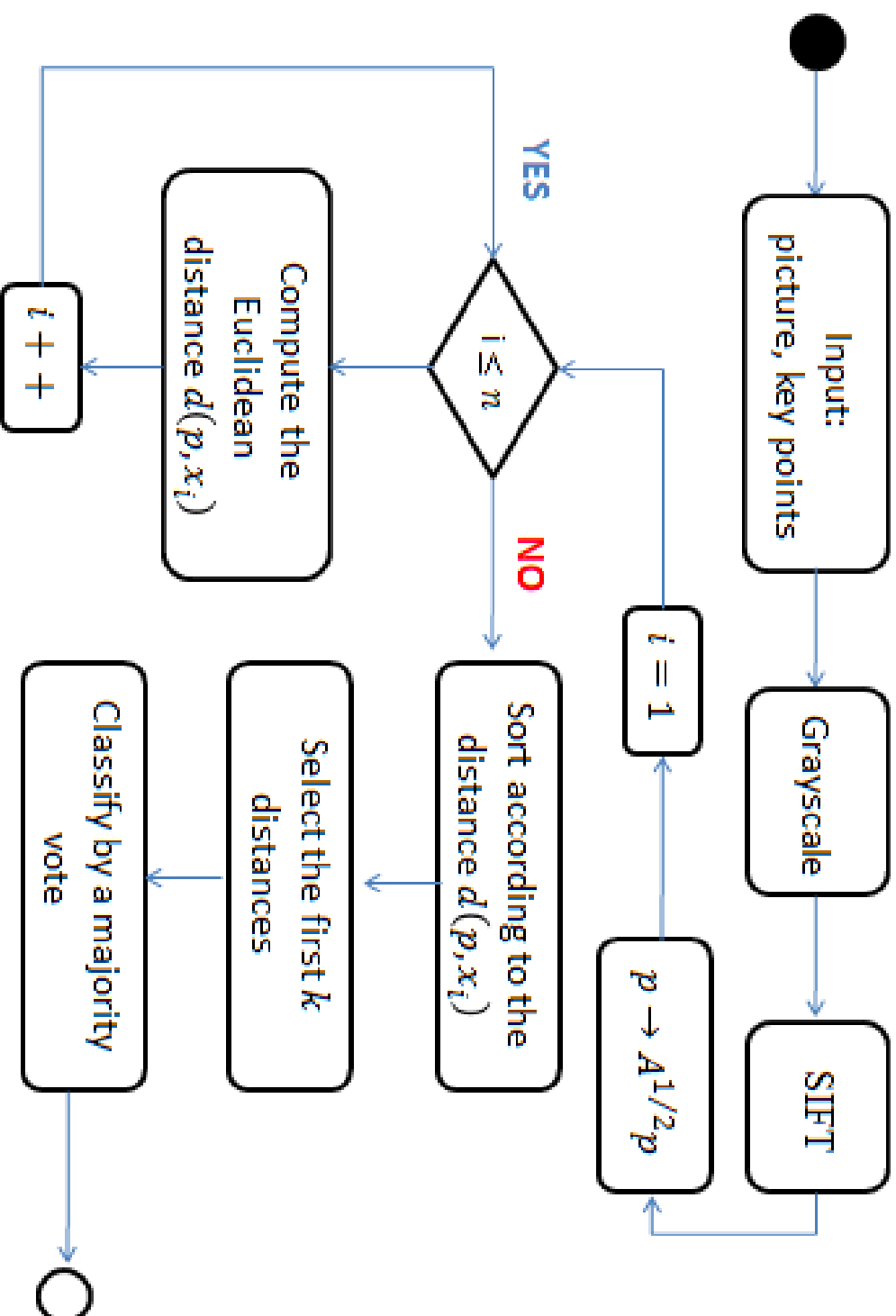
Learning Stage



איור 6: תרשים זרימה של שלב ה"למידה" – מתבצע "offline" באמצעות מאגר המידע

בלבד

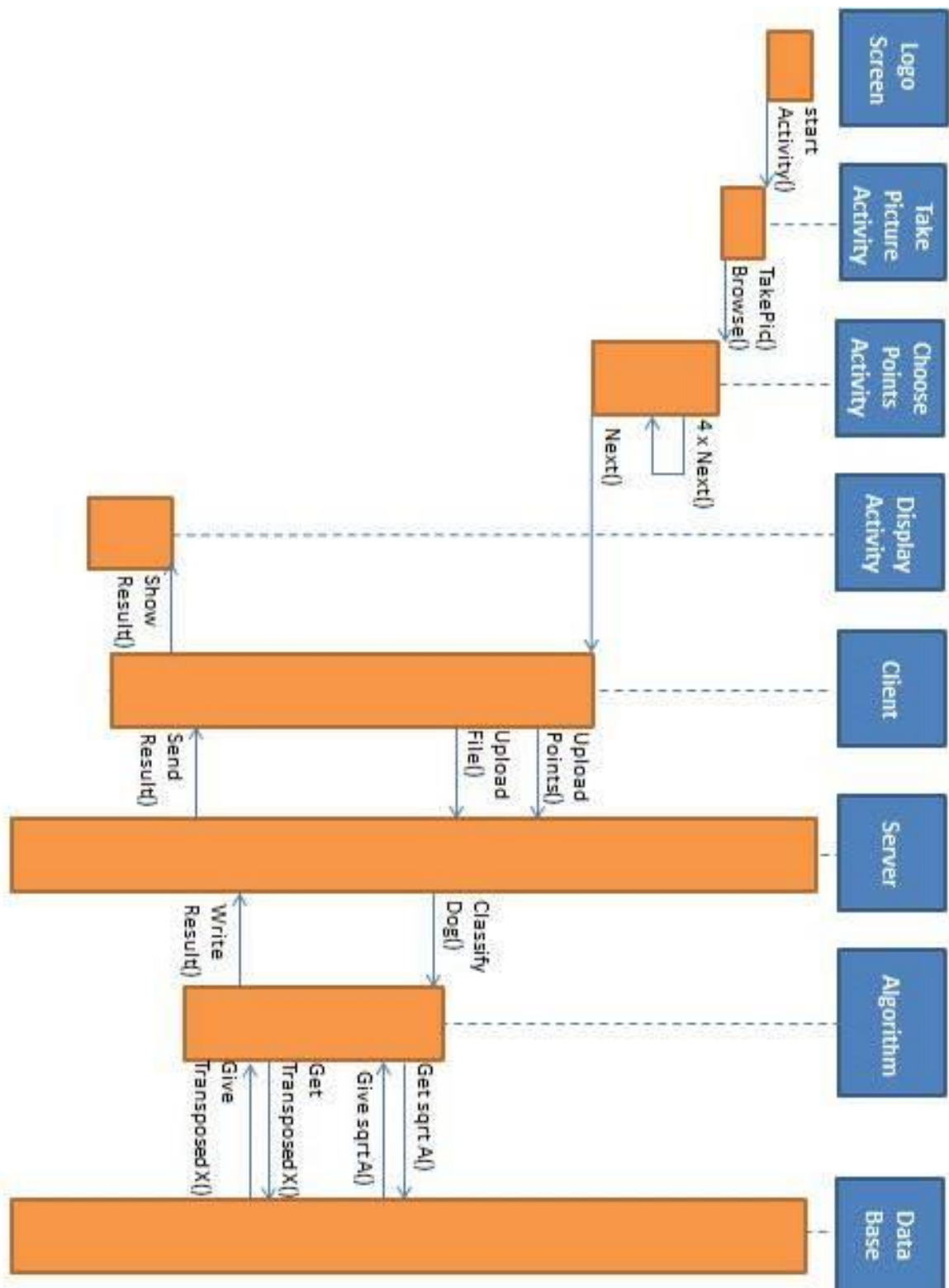
Classification Stage



איור 7: תרשים זרימה של שלב ה"סיווג" – מתבצע "online" ומשייך את תמונת הקלט

למספר גזעים להם הוא דומה מהמאגר מידע.

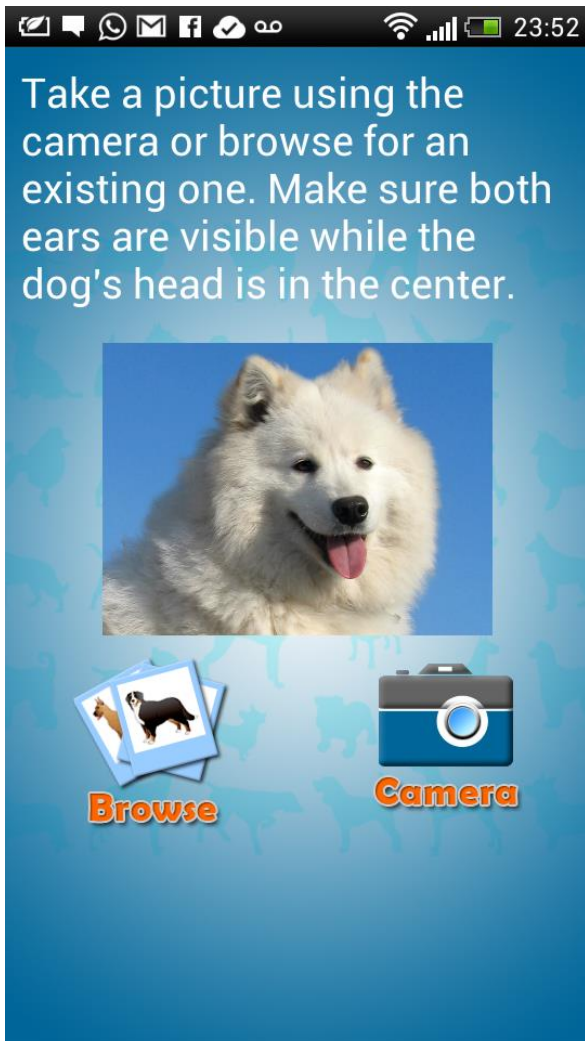
5.2. תרשים רצף – Sequence diagram



איור 9: תרשים רצף – Sequence diagram – ממשיך את פעולות המערכת "מאחורי

הקלעים"

5.3. ממשק משתמש

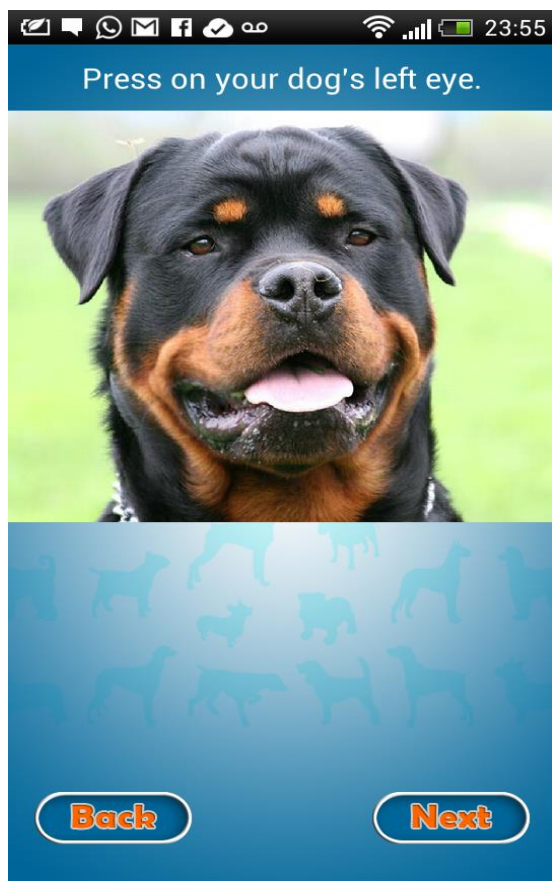


איור 11: מסך מרכזי

- Browse – אפשרות לבחירת תמונה מגלריה..
- Camera – אפשרות לצילום תמונה באמצעות המצלמה.



איור 10: מסך כניסה



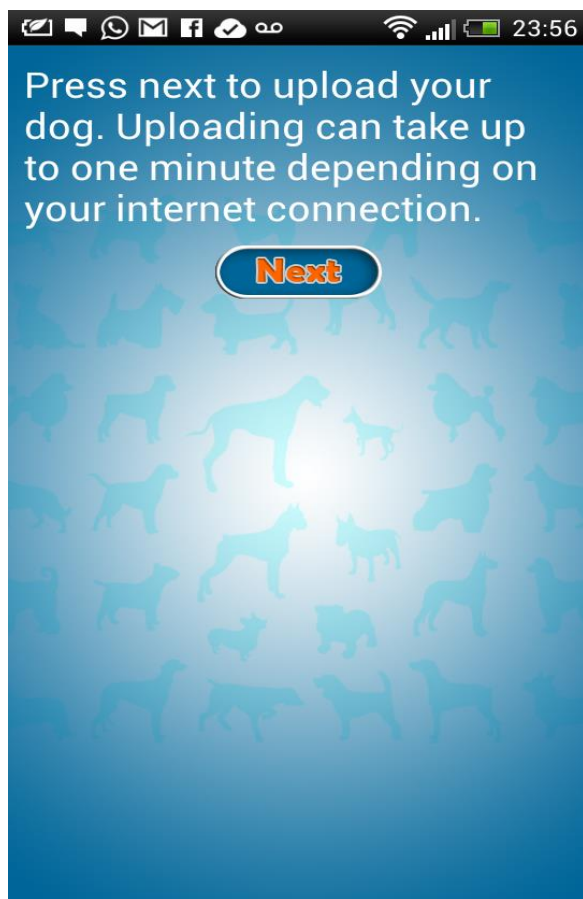
איור 13: מסך בחירת "נקודות עניין"

סדרה של מסכים עם הוראות משתנות לבחירת אזורים על פניו של הכלב

- Back - חזור לבחירה קודמת
- Next - המשך למסך הבא

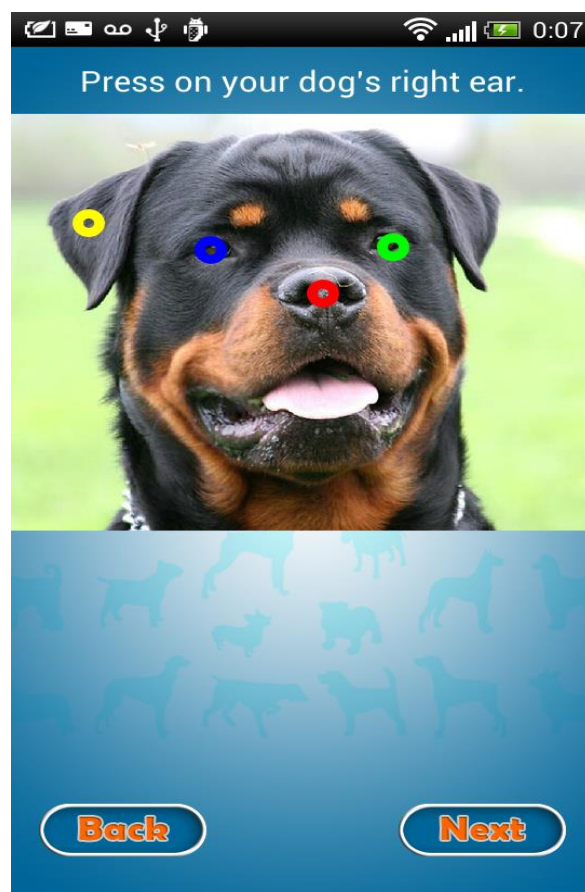


איור 12: בחירת תמונה מגלריית תמונות

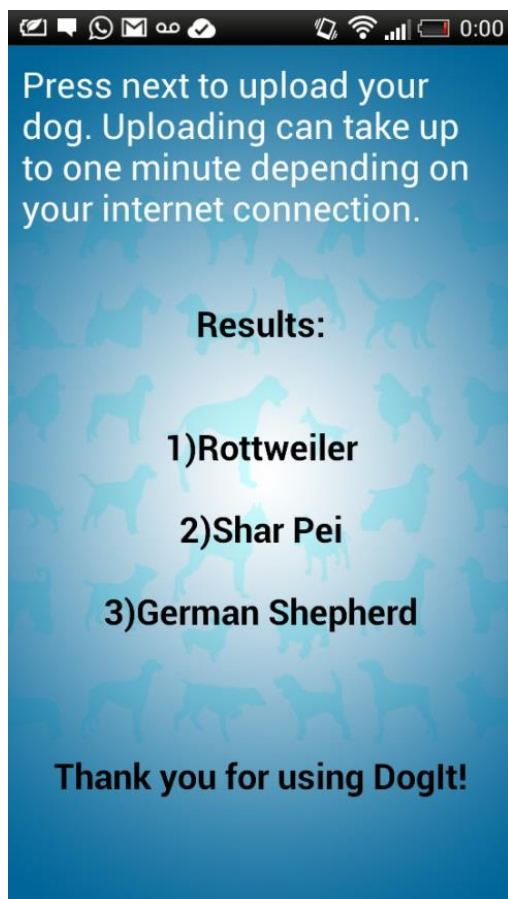


איור 15: שליחת מידע לשרת

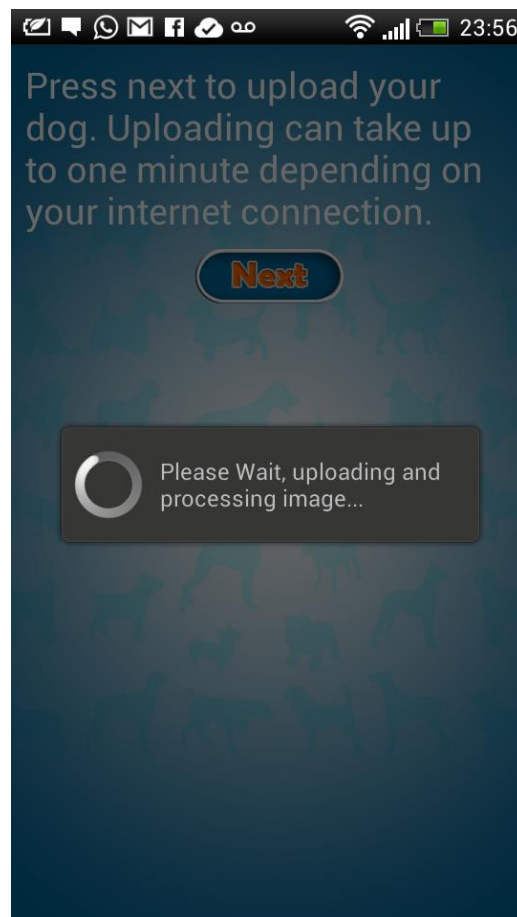
- שליחת התמונה עם החלקים שנבחרו לשרת.
- Next - שליחה לשרת



איור 14: סיום בחירת "נקודות עניין"



איור 17: מסך תוצאות



איור 16: עיבוד תמונה בשרת

6. תוצאות ומסקנות

6.1. תהליך פיתוח המערכת

האלגוריתם שתואר לעיל הינו אלגוריתם סדרתי. תחילה מוצאים נקודות עניין ע"י SIFT ומרכיבים מהם את וקטור התכונות שמייצג את התמונה. לאחר מכן ממשיכים לשלב למידת המטריקה ומחשבים את מטריצה A. רק לאחר חישוב המטריצה נוכל להתקדם ולהפוך את מסד הנתונים של תמונות הכלבים למסד נתונים של וקטורי תכונות מתאימים. לבסוף, נבצע סיווג לתמונה ספציפית מול המאגר ע"י K-NN.

מכיוון שהאלגוריתם סדרתי, כל החלטה שמתקבלת לגבי המימוש, תשפיע על המשך ביצוע האלגוריתם - לטובה ולרעה. אביא לדוגמא שתי החלטות שקיבלתי אשר עזרו להצלחת האלגוריתם.

1. בניית וקטור התכונות – בתחילת תהליך המימוש התנסיתי בווקטורי תכונות בגדלים שונים. למשל, בניתי וקטור תכונות בגודל 12,800 – מה שאומר שהרבה מידע מהתמונה המקורית נשמר. לצערי, זה יצר בעיות ביצועים ותכנת MATLAB קרסה מדי פעם בגלל מחסור בזיכרון. פתרתי בעיה זו ע"י הקטנת וקטור התכונות לממד 2560. המשמעות היא שמכל תמונה ניתן לבחור רק 20 נקודות SIFT. לכן הוספתי את האפשרות באפליקציה לבחור נקודות עניין בצורה ידנית (אזניים, עיניים, אף), דבר זה הבטיח ש-20 נקודות SIFT יבחרו מתוך האזורים הכי חשובים בתמונה.
2. קביעת רזולוציית התמונה - אם הרזולוציה של התמונה שנשלחת באפליקציה נמוכה מדי, יהיה קשה למצוא מספיק נקודות SIFT. מצד שני, אם הרזולוציה גבוהה מדי, לוקח הרבה זמן לשלוח את התמונה לשרת לעיבוד. לבסוף אחרי הרבה ניסויים החלטתי על רזולוציה 1024x768 או יחס של 3:4.
3. בניית מסד הנתונים – משימה זו הייתה קשה במיוחד מכיוון שרוב התמונות של כלבים באיכות טובה הן בתשלום. כאשר חיפשתי באינטרנט אחר תמונות למאגר שלי, נתקלתי בבעיות כגון שוני מהותי בין התמונות בגודל, זווית, איכות ותאורה. היה עלי לערוך את התמונות ידנית. לדעתי, לשוני במאגר התמונות שבניתי היה השפעה רעה על תוצאות האלגוריתם.
4. בחירת למבדה – בתהליך הבנייה של מטריצה A יש שימוש בפרמטר החלקה למבדה. במאמר IGML, הוסבר בקצרה הרעיון מאחורי הפרמטר אך לא ניתן מידע כיצד ניתן לחשב או להעריך פרמטר זה. לאחר שביצעתי מספר ניסויים שלא צלחו, החלטתי לשנות כיוון ולבחור בגישה אחרת שתתואר בהמשך.

בנוסף, נתקלתי במספר מכשולים במהלך הפיתוח וחלק מהבעיות דרשו עצירה ותכנון מחדש. שתי הדוגמאות להלן ממחישות זאת בצורה טובה.

1. כשניגשתי לבניית אפליקציית האנדרואיד, הסתמכתי על ידע מוקדם שהיה לי ב-Java. וחשבתי שאוכל לבנות מערכת לקוח-שרת בסיסית ב-Java. מסתבר שבאנדרואיד שפת ה-Java קצת שונה ומוגבלת. באנדרואיד קיימת מכונה וירטואלית שונה בשם "Dalvik" אשר מכילה הגבלות על זיכרון ומעבד, כיאה לסמארטפונים. חלק ניכר מהאפשרויות שקיימות בשפת ה-Java המקורית אינן קיימות באנדרואיד ולכן הייתי חייבת לשנות את שיטת שליחת התמונה שתכננתי. במקום להעביר אובייקטים ע"י זרם (stream) מימשי את שליחת התמונה ב-HTTP.
2. בבניית מטריצה A נתקלתי בבעיה רצינית. ב-SIFT, כאשר מחשבים את מטריצת השונות המשותפת, חייבים לקבל מטריצה סימטרית חיובית לחלוטין בעלת ערכים עצמיים חיוביים בלבד. במהלך בניית וקטור התכונות, לעיתים התקבלו מטריצות "כמעט חיוביות לחלוטין" אשר הכילו מספר ערכים עצמיים שליליים מאוד קרובים לאפס. זה יצר בעיה מכיוון שאלו מטריצות סינגולריות שלא ניתנות להפיכה – חלק הכרחי באלגוריתם שלנו. ניסיתי לפתור זאת ע"י פרמטר ההחלקה שתואר לעיל אך ללא הצלחה. הפתרון שנבחר היה להשתמש באלגוריתם NCM (Nearest Correlation Matrix) אשר הופעל על מטריצות אלו. זה עזר לפתור את הבעיה ע"י הוספת קצת רעש ונתן לי אפשרות לקבל מטריצה A שאיתה אוכל להתחיל לעבוד.

6.2. תוצאות

בדיקת ביצועי האלגוריתם התבצעה עם שלושה מאגרים שונים כדי להמחיש את השפעת גודל המאגר על התוצאות. עבור כל אחד מהמאגרים ביצעתי את תהליך הלמידה כולו ולכן לכל מאגר מטריקה משלו ומטריצה A משלו.

1. מאגר עם 4 גזעים, 2 תמונות לכל גזע ובסה"כ 8 תמונות.
2. מאגר עם 4 גזעים, 5 תמונות לכל גזע ובסה"כ 20 תמונות.
3. מאגר עם 4 גזעים, 10 תמונות לכל גזע ובסה"כ 40 תמונות.

בדקנו בסה"כ 40 תמונות של כלבים (10 תמונות לכל גזע) מול המאגרים השונים. הבדיקה התבצעה בצורה ידנית בעזרת האפליקציה כך שכל תמונה עוברת בדיוק את אותו התהליך של התאמת רזולוציה ובחירת אף, אוזניים ועיניים. כל תמונה אשר נשלחת לשרת מעובדת ע"י האלגוריתם ונשלחת תשובה לאפליקציה – 3 הגזעים שהכי דומים לתמונה שנשלחה.

להלן טבלת התוצאות. השורות מייצגות את תמונות המבחן והעמודות מייצגות את הגזעים השונים במאגר. התוכן של כל תא מייצג את הציון (מיקום יחסי) שקיבל הגזע עבור התמונה הספציפית.

למשל, כשהוכנסה תמונה מספר 4 של רועה גרמני לאפליקציה, התשובה שהתקבלה הייתה (1 שארפיי (2 רועה גרמני (3 רוטוויילר (4 סמוייד.

רועה גרמני	רוטוויילר	סמוייד	שארפיי
1	2	4	3
2	2	4	3
3	3	1	4
4	3	4	1
5	2	4	3
6	3	4	1
7	2	3	4
8	4	2	1
9	1	3	4
10	4	1	3

טבלה 1: רועה גרמני - תוצאות בשלב 3

מקום ראשון	מקום שני	מקום שלישי	מקום רביעי	
40%	60%	10%	0%	רועה גרמני

טבלה 2: רועה גרמני – סטטיסטיקה

	רועה גרמני	רוטוויילר	סמוייד	שארפיי
רוטוויילר 1	2	3	4	1
רוטוויילר 2	2	1	3	4
רוטוויילר 3	2	1	4	3
רוטוויילר 4	2	1	3	4
רוטוויילר 5	2	1	4	3
רוטוויילר 6	3	1	4	2
רוטוויילר 7	3	2	4	1
רוטוויילר 8	1	3	4	2
רוטוויילר 9	4	1	3	2
רוטוויילר 10	2	1	4	3

טבלה 3: רוטוויילר - תוצאות בשלב 3

	מקום ראשון	מקום שני	מקום שלישי	מקום רביעי
רועה גרמני	70%	10%	20%	0%

טבלה 4: רוטוויילר –סטטיסטיקה

	רועה גרמני	רוטוויילר	סמוייד	שארפיי
סמוייד 1	2	3	1	4
סמוייד 2	3	4	1	2
סמוייד 3	3	4	1	2
סמוייד 4	4	3	1	2
סמוייד 5	1	3	4	2
סמוייד 6	4	3	1	2
סמוייד 7	4	3	1	2
סמוייד 8	3	4	1	2
סמוייד 9	2	3	4	1
סמוייד 10	4	3	1	2

טבלה 5: סמוייד - תוצאות בשלב 3

	מקום ראשון	מקום שני	מקום שלישי	מקום רביעי
סמוייד	80%	0%	0%	20%

טבלה 6: סמוייד –סטטיסטיקה

שארפיי	סמוייד	רוטוויילר	רועה גרמני	
2	1	4	3	שארפיי 1
2	4	3	1	שארפיי 2
1	3	2	4	שארפיי 3
1	2	3	4	שארפיי 4
2	4	3	1	שארפיי 5
2	1	3	4	שארפיי 6
3	4	1	2	שארפיי 7
2	4	1	3	שארפיי 8
3	2	1	4	שארפיי 9
2	4	1	3	שארפיי 10

טבלה 7: שארפיי - תוצאות בשלב 3

מקום ראשון	מקום שני	מקום שלישי	מקום רביעי	
20%	60%	10%	10%	שארפיי

טבלה 8: שארפיי –סטטיסטיקה

להלן סיכום התוצאות עבור שלב 3 :

מקום ראשון	מקום שני	מקום שלישי	מקום רביעי	
40%	50%	10%	0%	רועה גרמני
70%	10%	20%	0%	רוטוויילר
80%	0%	0%	20%	סמוייד
20%	60%	10%	10%	שארפיי
52.5%	30%	10%	7.5%	סה"כ

טבלה 9: סיכום - תוצאות סטטיסטיות בשלב 3

6.3. מסקנות

המסקנה המיידית מהתוצאות היא, שלגודלו של מסד הנתונים יש השפעה על דיוק הסיווג. כאשר מגדילים מסד הנתונים מ-8 ל-20 תמונות, יש עלייה של 5% ברמת דיוקו של הסיווג. רמת הדיוק עולה עוד יותר, עד ל-17.5%, כאשר מגדילים את מסד הנתונים מ-20 ל-40 תמונות. אני משערת, שהתוצאות יכולות להשתפר אם אמשיך ואגדיל את בסיס הנתונים.

כאשר אנו מסתכלים על התוצאות של שלב 3 אנו יכולים להבחין בהבדל משמעותי ברמת דיוק הסיווג בגזעים מסוימים. לדוגמא, הזן "שארפיי" מראה תוצאות גרועות באופן משמעותי מה"סמוייד". אני מעריכה, שזה קשור לדרך שבו אלגוריתם SIFT לוכד תבניות. קיימת אפשרות ש"נקודות העניין" מתמונה של "שארפיי" אחד אינו דומה ל"נקודות עניין" מתמונה של "שארפיי" אחר. אם זה המקרה, זה גורם לבעיות בתהליך הבנייה של המטריקה.

אם וקטורי תכונה של זן מסוים אינם דומים זה לזה, הם ימוקמו באופן לא מדויק במרחב הכלבים החדש ולא יהיו מקובצים יחד. הווקטור הממוצע של הגזע יהיה חסר משמעות, ומטריצת השונות המשותפת של הגזע תהיה בעלת ערכים גבוהים מאוד (השוני הוא גבוה מדי). פתרון אפשרי הוא לשלב בין שני טכניקות לחילוץ מידע (שאחת מהן היא SIFT), כך שתיצור וקטורי תכונה טובים יותר. אולי כמה טכניקות יכולות ללכוד "נקודות עניין" טובות בגזע ה"שארפיי".

גזע ה"סמוייד" לעומת זאת, מציג תוצאות יוצאות דופן. כמעט בכל תמונה של "סמוייד" ישנו דיוק גבוה ברמת הסיווג. שבנוסף מבדיל אותו מגזעים אחרים. כאשר מתבצע סיווג לא נכון של גזעים אחרים, בדרך כלל הם לא מתבלבלים עם "סמוייד". לגזע ה"סמוייד" יש רק 23% טעויות בסיווג, כלומר שבו הסמוייד נבחר במקום הראשון לתמונה של כלב מגזע אחר. ל"שארפיי" יש 46% טעויות בסיווג, "רועה גרמני" 30% ו"רוטוויילר" 31%.

התוצאות הכוללות הן די טובות. ב-52.5% מהבדיקות הייתי יכולה לקבוע את הגזע בצורה נכונה וב-82.5% מהבדיקות, הגזע הנכון היה ב-2 התוצאות הראשונות.

7. ביבליוגרפיה

1. Xing, E. P.; Ng, A. Y.; Jordan, M. I.; Russell, S. (2002). "Distance Metric Learning, with Application to Clustering with Side-information". Advances in Neural Information Processing Systems 15 (MIT Press): 505–512.
2. Wang, S; Jin, R. (2009) "An Information Geometry Approach for Distance Metric Learning". Proceedings of the 12th International Conference on Artificial Intelligence and Statistics: 591-598.
3. D. G. Lowe; (2004) "SIFT: Distinctive image features from scale-invariant keypoints". IJCV 2(60): 91-110.
4. Vries, A. P; de, Mamoulis; N., Nes; N., Kersten, M. (2002). "Efficient k-NN Search on Vertically Decomposed Data". Proc of ACM SIGMOD, pp. 322-333
5. Bellet, A.; Habrard, A.; Sebban, M. (2013). "A Survey on Metric Learning for Feature Vectors and Structured Data". arXiv: 1306.6709 [cs.LG].
6. Kulis, B. (2012). "Metric Learning: A Survey". Foundations and Trends in Machine Learning.
7. Chechik, G.; Sharma, V.; Shalit, U.; Bengio, S. (2010). "Large Scale Online Learning of Image Similarity Through Ranking" Journal of Machine Learning research 11: 1109–1135.
8. Weinberger, K. Q.; Blitzer, J. C.; Saul, L. K. (2006). "Distance Metric Learning for Large Margin Nearest Neighbor Classification". Advances in Neural Information Processing Systems 18 (NIPS): 1473–1480.
9. Davis, J. V.; Kulis, B.; Jain, P.; Sra, S.; Dhillon, I. S. (2007). "Information-theoretic metric learning". International conference in machine learning (ICML): 209–216.
10. Guillaumin, M.; Verbeek, J.; Schmid, C. (2009). "Is that you? Metric learning approaches for face identification". IEEE International Conference on Computer Vision (ICCV).
11. Mignon, A.; Jurie, F. (2012). "PCCA: A new approach for distance learning from sparse pairwise constraints". IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

8. נספחים

8.1. קטעי קוד

8.1.1 יצירת מאגר מידע

ConstructMatrixA.m

```
% variable initialization
n=40; %number of images
indexes = 1;
numWindows = 5;
noKeyPointPerWindow = 4;
keyPointDescriptorLength = 128;
m = keyPointDescriptorLength*noKeyPointPerWindow*numWindows;

% initialize vectors and arrays
for i=2:noKeyPointPerWindow
    indexes = [indexes i];
end
X=zeros(m,n);
Y = zeros(C, n);
i = 0;

%parse folders for images
for k = 1 : C
    % Get the file pattern for dir().
    filePattern = [listOfFolderNames{k}, '\*.png'];
    % Find out names of PNG files in this folder.
    pngFiles = dir(filePattern);
    numOfImagesInBreed = length(pngFiles);
    if ~isempty(pngFiles)
        % for each image in breed
        for imageNumber = 1 :numOfImagesInBreed/numWindows

            imageFeatureVector = [];
            for windowNumber = 1 : numWindows
                fullFileName = fullfile(listOfFolderNames{k},
                    pngFiles((imageNumber-1)*numWindows+windowNumber).name);
                imageArray = imread(fullFileName);
                GrayImg = single(rgb2gray(imageArray)) ;
                %calculate SIFT descriptors for window
                [descriptors, locs] = sift(GrayImg);
                [rows cols] = size(descriptors);
                if rows < noKeyPointPerWindow
                    %if there's not enough keypoints in the window,
                    %generate random vector of the same scale as a
                    %feature vector.
                    if rows == 0 & windowNumber ~= 1
                        descriptor = transpose(descriptor);
                        descriptors = descriptor([2 4 1 3], :);
                    else
                        for s=1:(noKeyPointPerWindow-rows)
                            newRow = descriptors(end, : );
                            newRow = awgn(newRow,10,'measured');
                            descriptors = [descriptors newRow];
                        end
                    end
                end
            end
        end

        d = transpose(descriptors);
```

```

        descriptor = d(:,indexes);
        imageFeatureVector = [ imageFeatureVector ;
                                descriptor(:)];
    end
    %storing feature vector in breed matrix x which is the
size
    %of the number of images in breed (5)
    i = i + 1
    X(:,i)=imageFeatureVector;
    yi = zeros(C,1);
    %the current breed gets a label of 1, all others get 0
    yi(k) = 1;
    %the current image is assigned to breed k
    Y(:,i) = yi;
end
end
end
zzzStage = 1
YPath = ['C:\Users\Iftah\DogDatabase\Y.dat'];
csvwrite(YPath, Y);

%initialize matrix A
tempA = zeros(m,m);
lambda = 0.14;
for k = 1 : C
    Zk = Y(k,:);
    %ZkT is a nx1 binary column vector indicating for
    %each image 1 if it belongs to breed k and 0 otherwise
    ZkT = transpose(Zk);
    %Sk is the number of images in breed k
    Sk = sum(Zk);
    %calculate mean vector
    xkMean = (1/Sk)* ( X * ZkT) ;
    %Calculate covariance matrix
    sumCov = zeros(m,m);
    for i = 1 : n
        sumCov = sumCov + (Y(k,i) * (X(:,i) - xkMean) *
transpose(X(:,i) - xkMean)) ;
    end
    SigmakCov = (1/Sk) * sumCov;
    %apply NCM method on covariance matrix
    [SmoothSigmakCov,iter] = nearcorr(SigmakCov);
    tempA = tempA + Sk * (SmoothSigmakCov + ((lambda/(lambda + Sk)) *
(xkMean * transpose(xkMean)) ));
end
%finish calculating matrix A
A = lambda * inv(tempA);
%calculate A^(1/2) for future use
srA = sqrtm(A);
%Transform all vectors in X using the new metric
for i=1:n
    imageFeatureVector = X(:,i);
    newImageFeatureVector = srA * imageFeatureVector;
    transformedX(:,i) = newImageFeatureVector;
end

srAPath = ['C:\Users\Iftah\DogDatabase\matrixA4x5x128.dat'];
csvwrite(srAPath, srA);
zzzStage = 5
normalXPath = ['C:\Users\Iftah\DogDatabase\normalX.dat'];

```

```

csvwrite(normalXPath, X);
transformedXPath = ['C:\Users\Iftah\DogDatabase\transformedX.dat'];
csvwrite(transformedXPath, transformedX);

```

8.1.2 סיווג תמונה

ClassifyDog.m

```

function ClassifyDog(input_img_path, output_img_path)
clc;
n=40;
C=4;
numWindows = 5;
noKeyPointPerWindow = 4;
indexes = 1;
fv = [];
for i=2:noKeyPointPerWindow
    indexes = [indexes i];
end
%cut the windows from the image
cutPicture(input_img_path, input_points_path, input_folder_path);
%for each window create sift descriptor and concatenate to feature
vector
for i = 1 : numWindows
    switch i
        case 1
            fullFileName = [input_folder_path 'rightEye.jpeg'];
        case 2
            fullFileName = [input_folder_path 'leftEye.jpeg'];
        case 3
            fullFileName = [input_folder_path 'nose.jpeg'];
        case 4
            fullFileName = [input_folder_path 'rightEar.jpeg'];
        case 5
            fullFileName = [input_folder_path 'leftEar.jpeg'];
    end
    imageArray = imread(fullFileName);
    GrayImg = single(imageArray) ;
    [descriptors, locs] = sift(GrayImg);
    d = transpose(descriptors);
    %extract only most significant keypoints
    descriptor = d(:,indexes);
    fv = [fv; descriptor(:)];
end
%Load matrices A, Y and transformedX for k-nn
APath = ['C:\Users\Iftah\DogDatabase\matrixA4x5x128.dat'];
srA = csvread(APath);
YPath = ['C:\Users\Iftah\DogDatabase\Y.dat'];
Y = csvread(YPath);
transformedXPath = ['C:\Users\Iftah\DogDatabase\transformedX.dat'];
transformedX = csvread(transformedXPath);
%position feature vector in its new
%location in the Breed space
transformedFv= srA*fv;
%measure distance between input feature vector
%to all feature vectors in the database
distances = zeros(1,n);
for i=1:n
    d = (norm(transformedFv - transformedX(:,i)));

```



```

        distances(i) = d;
    end
    %start knn
    knn_k = 6;
    %sort distances
    [newd indexes2] = sort(distances, 'ascend');
    %determine best k candidates
    if numel(indexes2) > knn_k
        selected = indexes2(1:knn_k);
    else
        selected = indexes2;
    end
    %extract breed from matrix Y for each of the
    %candidtes and do a majority vote
    breedVotes = zeros(1,C);
    for j=1:numel(selected)
        indexOfCandidateInY = selected(j);
        colVector = Y(:, indexOfCandidateInY);
        breedIndexes = find(colVector);
        breedVotes(breedIndexes(1)) = breedVotes(breedIndexes(1)) + 1;
    end
    %determining the most popular breed
    [temp winnerIndexes] = sort(breedVotes, 'descend');
    %extract list of breed names
    namesPath = 'C:\Users\Iftah\DogDatabase\breedNames.txt';
    fid = fopen(namesPath, 'r');
    breedNames = textscan(fid, '%q');
    breedNames = transpose(breedNames{1});
    fclose(fid);
    winners = breedNames(winnerIndexes);
    %print the 3 most popular dogs of knn to a string
    answer = sprintf( '%s#%s#%s', winners{1}, winners{2}, winners{3});
    %output the answer back to the server
    Log(answer);
end

```

8.1.3 שרת

UploadPoints.php

```

<?php
    if (isset($_POST['points_string']) && $_POST['points_string'] !=
    '') {
        $points_string = $_POST['points_string'];
        $file = './uploads/points.txt';
        file_put_contents($file, $points_string);
        echo 'received the string: ' . $points_string;
    } else {
        echo 'fail';
    }
}
?>

```

UploadToServer.php

```

<?php
    $input_file_path = "uploads/";
    $input_file_path = $input_file_path . basename(
    $_FILES['uploaded_file']['name']);

    $output_input_file_path = "output/processed/";

```

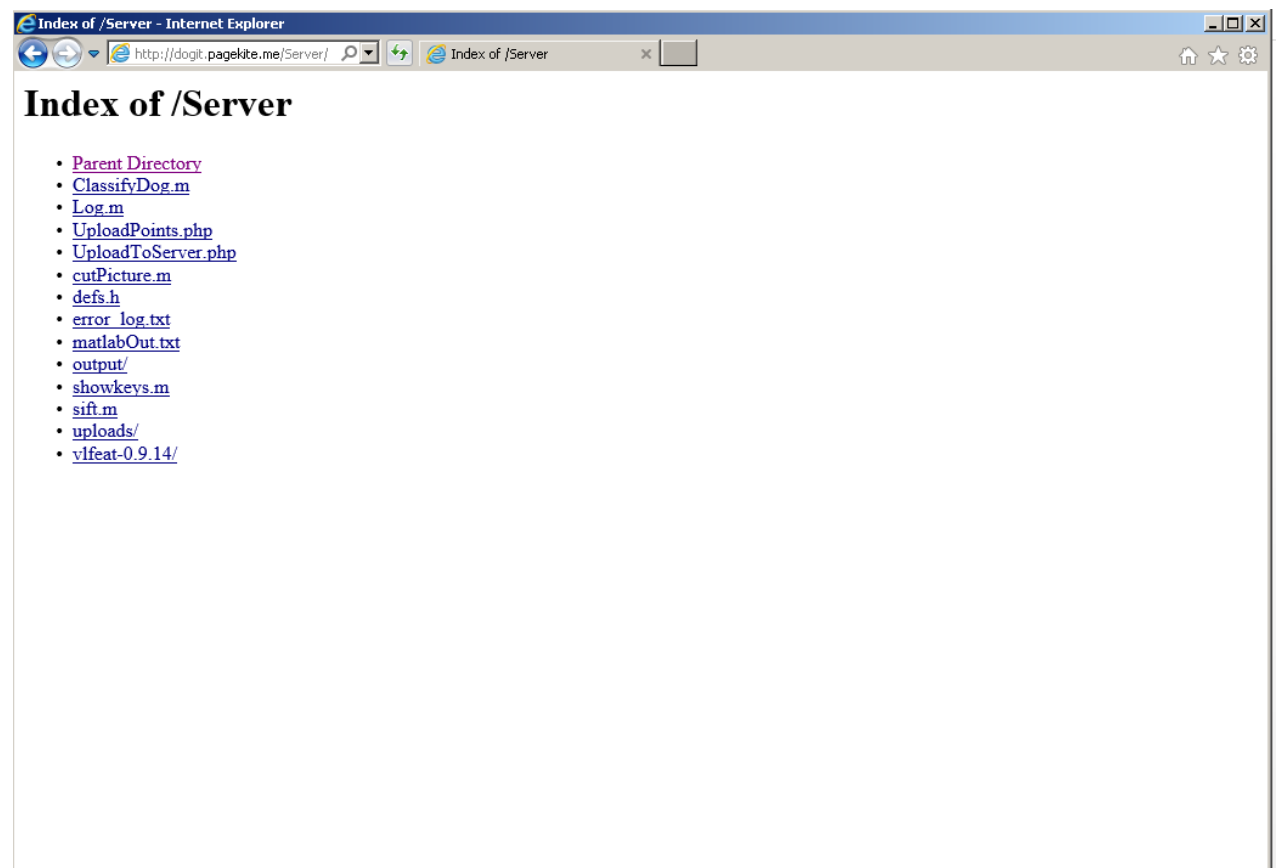
```

$output_input_file_path = $output_input_file_path. basename(
$_FILES['uploadedfile']['name']);

ini_set('upload_max_filesize', '20M');
ini_set('post_max_size', '20M');
ini_set('max_input_time', 300);
ini_set('max_execution_time', 300);

if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'],
$input_file_path)) {
    $command = "matlab -nojvm -nodesktop -nodisplay -wait -r
\"ClassifyDog('$input_file_path','$output_input_file_path');exit\"";
    exec($command);
    $file = './uploads/matlabLog.txt';
    $answer = file_get_contents($file);
    echo "success#$answer";
} else{
    echo "fail";
}
?>

```



running server :18 איור

8.1.4 לקוח

UploadToServer.java – uploadFile() function

```
public int uploadFile(String sourceFileUri) {
    // image saved by earlier activity
    String fileName = sourceFileUri;
    HttpURLConnection conn = null;
    DataOutputStream dos = null;
    String lineEnd = "\r\n";
    String twoHyphens = "--";
    String boundary = "*****";
    int bytesRead, bytesAvailable, bufferSize;
    byte[] buffer;
    int maxBufferSize = 1 * 1024 * 1024;
    File sourceFile = new File(sourceFileUri);

    if (!sourceFile.isFile()) {
        dialog.dismiss();
        Log.e("uploadFile", "Source File not exist :" +
            pathOfFileToSend);
        runOnUiThread(new Runnable() {
            public void run() {
                messageText.setText("Source File not exist :" +
                    pathOfFileToSend);
            }
        });
        return 0;
    } else {
        try {
            FileInputStream fileInputStream = new FileInputStream(
                sourceFileUri);
            URL url = new URL(uploadServerUrl);
            // Open an HTTP connection to the URL
            conn = (HttpURLConnection) url.openConnection();
            conn.setDoInput(true); // Allow Inputs
            conn.setDoOutput(true); // Allow Outputs
            conn.setUseCaches(false); // Don't use a Cached Copy
            conn.setRequestMethod("POST");
            conn.setRequestProperty("Connection", "Keep-Alive");
            conn.setRequestProperty("ENCTYPE", "multipart/form-
data");

            conn.setRequestProperty("Content-Type",
                "multipart/form-data;boundary=" + boundary);
            conn.setRequestProperty("uploaded_file", fileName);

            // Output stream is used to send the data
            dos = new DataOutputStream(conn.getOutputStream());
            dos.writeBytes(twoHyphens + boundary + lineEnd);
            dos.writeBytes("Content-Disposition: form-data;
name=\"uploaded_file\";filename=\"\"
                + fileName + "\"" + lineEnd);
            dos.writeBytes(lineEnd);

            // create a buffer of maximum size
            bytesAvailable = fileInputStream.available();
            bufferSize = Math.min(bytesAvailable, maxBufferSize);
            buffer = new byte[bufferSize];
```

```

        // read buffer sized data from the file
        bytesRead = fileInputStream.read(buffer, 0, bufferSize);

        // transfer buffer to server and repeat until no data is
left to send
        while (bytesRead > 0) {
            dos.write(buffer, 0, bufferSize);
            bytesAvailable = fileInputStream.available();
            bufferSize = Math.min(bytesAvailable,
maxBufferSize);
            bytesRead = fileInputStream.read(buffer, 0,
bufferSize);
        }

        dos.writeBytes(lineEnd);
        dos.writeBytes(twoHyphens + boundary + twoHyphens +
lineEnd);

        // wait for result
        InputStream stream = conn.getInputStream();
        InputStreamReader isReader = new
InputStreamReader(stream);

        // put output stream into a string
        BufferedReader br = new BufferedReader(isReader);

        String s = null;
        boolean uploadSuccess = false;
        while ((s = br.readLine()) != null) {
            // messageText.setText(s);
            Log.i("uploadFile", "HTTP Response (stream) is : "
+ s);

            tempAnswer = s.split("#");
            if (tempAnswer[0].equalsIgnoreCase("success")) {
                uploadSuccess = true;
                break;
            } else if (tempAnswer[0].equalsIgnoreCase("fail"))
            {
                uploadSuccess = false;
                break;
            }
        }
    }
}
}

```