**תיכון עירוני ד׳, תל אביב - יפו**

**אפליקציית אייפון לסיווג ולוקליזציה של כלבים**

****

**מגישה: <שם המגישה>, <מס. ת.ז>**

**מנחה: שי פרח**

**שם החלופה: למידת מכונה**

**תאריך הגשה:**

**תוכן עניינים**

[**מבוא 3**](#_ybv3rtqbdtt2)

[רקע מקדים 3](#_1f45kb4826yo)

[אופן הפעולה של האפליקציה 3](#_w36si7105wn3)

[סיבת הבחירה בנושא 3](#_sucxt6akwyw3)

[תהליך המחקר ואתגרים מרכזיים 4](#_vq1rd4stjd0y)

[**מבנה הפרויקט 5**](#_6ys5852s7ip3)

[**איסוף, הכנה וניתוח הנתונים 5**](#_vm260hq2n0ic)

[ניתוח מסד הנתונים - Stanford dogs 5](#_u1bco6xn7uf)

[נרמול הנתונים והכנתם למודל 5](#_q12i29imhnjh)

[אוגמנטציית נתונים 6](#_oxc7dvmtcmrj)

[**בניית ואימון המודל 7**](#_7ud6da6ggdgu)

[תיאור גרפי והסבר על שכבות המודל 7](#_enmmt123s8bj)

[ניתוח הביצועים בשלב האימון והמבחן 10](#_va4nlxhrduzw)

[תיאור השינויים והניסויים שנעשו במהלך העבודה 12](#_ooxq65bt09us)

[טבלה מסכמת של כל הניסויים 19](#_6ii981qhsqwi)

[היפר פרמטרים במודל הנוכחי 20](#_pacq0ojuv4am)

[פונקציית השגיאה 20](#_1wqqd3886wul)

[מדדי הביצוע 22](#_x0mobpaju65s)

[**יישום 23**](#_jj0tckxzy6t0)

[הסבר על היישום ואיך הוא משתמש במודל 23](#_uq30ury8x5fs)

[מחלקות האפליקציה 23](#_jjits3od786m)

[תרשים UML 25](#_745iciwgtxvt)

[הטכנולוגיה בה השתמשתי - SwiftUI 26](#_gm148gbrhf1z)

[התאמת נתוני החיזוי למודל 27](#_goj1drhicqlq)

[**מדריך למפתח 28**](#_n73s0469g2ki)

[המודל 28](#_637tyamafrbo)

[היישום 29](#_81ghsojfzfai)

[**מדריך למשתמש 30**](#_v29c087wpkjr)

[התקנת האפליקציה: 30](#_v07dn5cb9qtu)

[תרשים של מסכי האפליקציה 31](#_11vbyoig1ofj)

[הסבר על כל מסך והכפתורים שבו 31](#_fl0jwmaufh2x)

[**רפלקציה אישית 34**](#_r6g6qy7qcc6r)

[**ביבליוגרפיה 35**](#_b3r4hvi0aem)

[**נספחים 36**](#_lx56505hlq6b)

# מבוא

## רקע מקדים

הפרויקט עוסק בסיווג ולוקליזציה של 120 גזעי כלבים באמצעות למידת מכונה. החשיבות של יכולת זו היא ביצירת בסיס להערכה של גזעי כלבים מעורבים, דבר שיכול לסייע להבנה טובה יותר של בעלי כלבים את כלביהם, ולבחירת הכלב המתאים. הגישה הייתה שימוש בשיטת transfer learning על בסיס הרשת העמוקה Xception שאומנה על מסד הנתונים ImageNet. התוצאות הראו דיוק של 88.9% וביצועי IoU של 64.15% בסיווג ולוקליזציה של הכלבים מ-120 הגזעים.

### אופן הפעולה של האפליקציה

ראשית, המשתמש מתבקש לצלם או לבחור תמונה של כלב מתוך ספריית התמונות בטלפון שלו. לאחר מכן, התמונה מעובדת לגודל 299 על 299 פיקסלים, על מנת להתאים למודל שבחרתי. התמונה מועבר במודל, והאפליקציה מציגה למשתמש את הזן שחזה המודל, את הביטחון שלו בחיזוי, ומציירת מלבן אדום שמייצג את ה-bounding box שהמודל חזה.

**קישור לסרטון של האפליקציה בפעולה:** [**https://youtu.be/CCF38XGFVyI**](https://youtu.be/CCF38XGFVyI)

### סיבת הבחירה בנושא

בחרתי בנושא כי יש לי כלבה, אמנם מעורבת, אבל חשבתי שזה יכול להיות מעניין לראות האם המודל שלי יצליח לזהות מה הזן העיקרי שלה או כמה זנים עיקריים. בנוסף, זאת הייתה דרך נהדרת בשבילי ללמוד על לוקליזציה של אובייקטים בנוסף לסיווג, וגם להתנסות בפיתוח אפליקציה לאייפון, שהוא נושא שאני רוצה ללמוד כבר הרבה זמן ועוד לא הייתה לי הזדמנות.

הפרוייקט שלי בא לעזור לאנשים שנמצאים ממש באותו המצב כמוני- שהיו רוצים לדעת את זן הכלב שלהם, או אפילו להכיר זנים נוספים, ואין להם איך.

## 

## תהליך המחקר ואתגרים מרכזיים

בפרוייקט שלי בחרתי להתמקד בפיתוח האפליקציה, כלומר ממשק המשתמש. לכן, עיקר האתגרים הגיעו בשלב המרת המודל לפורמט שאוכל לעבוד איתו בתוך האפליקציה שלי, הלמידה מאפס לחלוטין של שפת התכנות Swift וה-debugging של קוד שהייתי צריכה ללמוד להבין. בנוסף, בחרתי בבעיה יחסית מסובכת עבור מודלים פשוטים, על כן נאלצתי להשתמש במודלים חזקים יותר במהלך הניסויים שלי. מרבית הזמן המודלים החזקים לקחו זמני הרצה ארוכים משמעותית מהרגיל, וכל שינוי שבחרתי לעשות הצריך זמן אימון מחודש, מה שהקשה עליי לבצע כמויות גדולות של ניסויים.

בעיה נוספת שנתקלתי בה במהלך העבודה היא בעיית overfit של המודל שבחרתי. השתמשתי בtransfer learning משני מודלים חזקים שאומנו על מסד הנתונים imageNet, והשוויתי בין התוצאות של כל אחד על הבעיה הנתונה. בגלל כמות הנתונים הקטנה שיש לכל מחלקה של הנתונים (כ-170 תמונות לכל זן של כלב) והקרבה הויזואלית בין זנים רבים, המודלים החזקים שאומנו על מיליוני תמונות שונות הגיעו ל-overfit מאוד במהירות. לכן, בחרתי להשתמש באוגמנטציית נתונים על מנת להתמודד עם הבעיה, מה שלצערי יצר overfit קל לנתוני הולידציה. שיניתי את רמת האוגמנטציה ואת סוגיה על מנת להתקרב כמה שניתן לוריאציות אמיתיות שראיתי בתמונות והצלחתי להקטין באופן משמעותי את הפער בין נתוני האימון לנתוני הולידציה.

## גישת הפתרון המוצעת לבעיה

לפתרון בעיית הסיווג והלוקליזציה של דמויות כלבים, בחרתי בגישת transfer learning באמצעות רשת עמוקה שאומנה מראש על מסד הנתונים ImageNet. גישה זו נבחרה מתוך הנחה כי ארכיטקטורה עמוקה ומורכבת יותר תספק ביצועים טובים באופן משמעותי לעומת אימון מודל מאופס, במגבלות כוח החישוב שעמד לרשותי.

לצורך הערכת הביצועים, הסיווג נבחן לפי מדדי הדיוק ו-loss, ואילו לוקליזציית הדמויות נבחנה בעזרת מדד החפיפה (IoU - Intersection Over Union), המודד את החלק המשותף בין תיבת הזיהוי המוצעת לבין תיבת הזיהוי האמיתית, ביחס לשטח הכולל של שתיהן.

במהלך העבודה, השוויתי בין שתי ארכיטקטורות רשת שונות - MobileNetV2 ו-Xception. השוואה זו בוצעה תוך אימון שני המודלים באותם תנאים - אוגמנטציית נתונים זהה, נתוני אימון ובחינה זהים, והיפרפרמטרים דומים. ההשוואה התבססה על מדדי הדיוק, ה-IoU והפסדים לסיווג וללוקליזציה. יצוין כי ההשוואה בין שתי הארכיטקטורות בוצעה רק לאחר טיוב ראשוני של MobileNetV2, מכיוון שזמני האימון של Xception היו גבוהים פי 5 ויותר, מה שהיה מגביל את יכולת הטיוב השיטתי.

לאחר השוואה מדוקדקת, הארכיטקטורה שנבחרה כגישת הפתרון הסופית היא Xception, בשל הביצועים המשופרים שלה ורמת ה-Overfit הנמוכה מזו של ה-MobileNetV2.

# מבנה הפרויקט

## איסוף, הכנה וניתוח הנתונים

### ניתוח מסד הנתונים - Stanford dogs

מאגר הנתונים שהשתמשתי בו הוא Stanford Dogs, הכולל 20,580 תמונות של 120 גזעי כלבים שונים. יש 150-200 תמונות של כל גזע בנתונים. כל תמונה מכילה בין כלב אחד ועד 5 כלבים מאותו הגזע. מסד הנתונים מתועד בשתי דרכים:

1. תגית (label) המציינת את גזע הכלב/ים בתמונה.
2. תיבת זיהוי (bounding box) סביב כל כלב בתמונה, מיוצגת על-ידי ערכי ה-y,x של הפינה השמאלית העליונה והימנית התחתונה.

אנוטציות אלו אפשרו לי לבצע סיווג של גזעי הכלבים ולוקליזציה של מיקומם בתמונה, כמו גם גילוי מרובה של דמויות כלבים בהמשך.

בפרוייקט הנוכחי, סיננתי את הנתונים כך שאשתמש רק בתמונות עם כלב אחד בלבד. כדי להמחיש את המאפיינים החזותיים של הנתונים צירפתי 5 דוגמאות רנדומליות עם אנוטציות מלאות.



### נרמול הנתונים והכנתם למודל

כיוון שהמודל שבחרתי להסתמך עליו, Xception, מצפה לתמונות בגודל מסויים שהפיקסלים שלהן מנורמלים לטווח [1,1-], נרמול הנתונים היה פשוט. מצורפות אותן 5 דוגמאות רנדומליות לאחר התאמת גודל התמונות (נרמול ערכי הפיקסלים מתבצע בתוך המודל).



### אוגמנטציית נתונים

על מנת להתמודד עם בעיית האוברפיט במודל, עליה אסביר יותר בהמשך, בחרתי, בין השאר, להשתמש באוגמנטציית נתונים על מנת להגדיל את הוריאציה ולדמות מסד עם יותר נתונים לכל זן של כלב.

על מנת לבחור בסוגי האוגמנטציה הסתמכתי על המאמר ״Dog Breed Classification and Visualisation״ (Candido, 2020), שגם הוא השתמש בXception על אותו מסד נתונים, ובחרתי בשתי אוגמנטציות שהרגישו לי הכי דומות לוריאציות שניתן יהיה לראות בתמונות אמיתיות שאני או אחרים יצלמו.

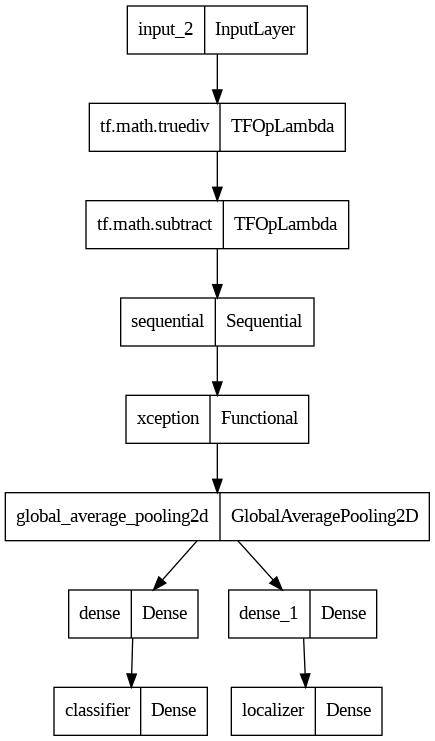
1. היפוך תמונות אקראי (אופקי)
2. סיבוב התמונה בטווח של 20% באופן רנדומלי

בחרתי בשתי האוגמנטציות הללו בגלל שהן מדמות באופן יחסית טוב וריאציות אמיתיות בתמונות - כלב שעומד או מצולם בכיוון ההפוך יראה כמו תמונה שהפכו בכיוון ההופכי, ואנשים שונים נוטים להחזיק את מצלמתם בזוויות שונות, כך שסיבוב התמונה הוא וריאציה הגיוני לחלוטין.

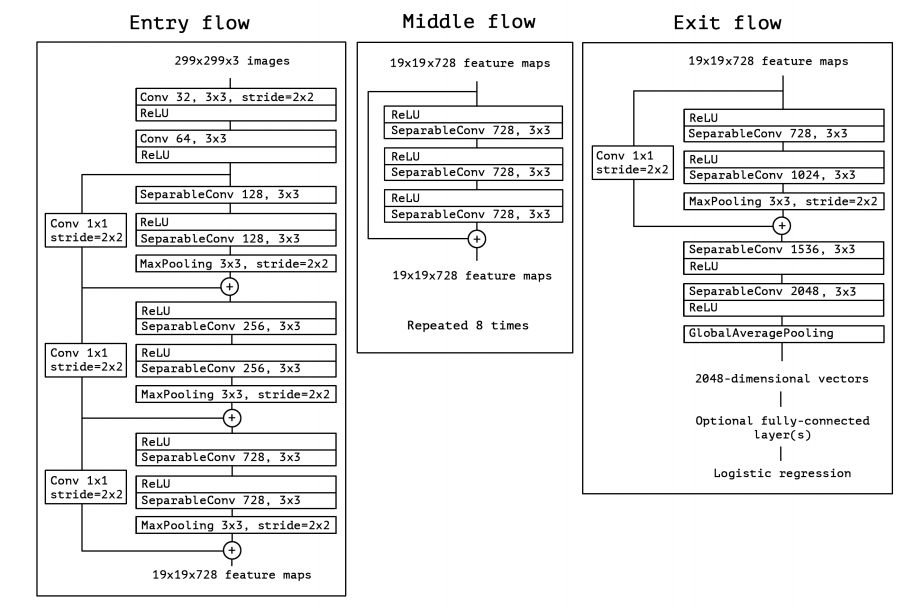
## 

## בניית ואימון המודל

### תיאור גרפי והסבר על שכבות המודל

**הצגה גרפית של המודל (רק ראש המודל, ללא המודל המלא של Xception)**

תרשים זה מציג רק את ״ראש״ המודל, בו פיצלתי את פעולת המודל לשתיים, במקום לבצע רק סיווג הוא מבצע גם לוקליזציה, ועל כן בחרתי לא להשתמש בכל רשת ה-Xception אלא להתעלם מכמה השכבות האחרונות.

**מבנה המודל השלם**

המבנה של Xception מבוסס על הInception שבאה לפני, אבל לוקח אותה צעד קדימה בעזרת שימוש בקונבולוציות מופרדות לעומק (depthwise separable convolutions). הארכיטקטורה של Xception מחולקת לשלושה חלקים עיקריים: זרימת כניסה (Entry flow), זרימת אמצע (Middle flow) וזרימת יציאה (Exit flow). כל חלק מתמקד בשלב שונה בתהליך עיבוד התמונה.

**Entry Flow**

החלק הראשון של הארכיטקטורה מתמקד בהפחתת הממדים של התמונה המקורית (299x299x3) ואיסוף התכונות הבסיסיות:

1. Conv 32, 3x3, stride=2x2: קונבולוציה ראשונית עם 32 פילטרים, גודל פילטר 3x3, וצעד 2x2.
2. ReLU: פונקציית אקטיבציה ReLU.
3. Conv 64, 3x3: קונבולוציה שנייה עם 64 פילטרים.
4. ReLU: פונקציית הפעלה ReLU.
5. SeparableConv 128, 3x3: קונבולוציה מופרדת לעומק עם 128 פילטרים, גודל פילטר 3x3.
6. MaxPooling 3x3, stride=2x2: שכבת מקס פולינג עם פילטר בגודל 3x3 וצעד 2x2.

**Middle Flow**

החלק האמצעי של הארכיטקטורה חוזר על עצמו שמונה פעמים ומתמקד באיסוף תכונות מורכבות יותר מהתמונה:

1. SeparableConv 728, 3x3: קונבולוציה מופרדת לעומק עם 728 פילטרים, גודל פילטר 3x3.
2. ReLU: פונקציית אקטיבציה ReLU.
3. זרימת האמצע חוזרת על עצמה 8 פעמים.

**Exit Flow**

החלק האחרון של הארכיטקטורה מסיים את תהליך העיבוד ומפיק את הווקטור הסופי לתהליך הסיווג:

1. SeparableConv 728, 3x3: קונבולוציה מופרדת לעומק עם 728 פילטרים, גודל פילטר 3x3.
2. ReLU: פונקציית אקטיבציה ReLU.
3. Conv 1x1, stride=2x2: קונבולוציה בגודל 1x1 עם צעד 2x2.
4. MaxPooling 3x3, stride=2x2: שכבת מקס פולינג עם פילטר בגודל 3x3 וצעד 2x2.
5. SeparableConv 1536, 3x3: קונבולוציה מופרדת לעומק עם 1536 פילטרים, גודל פילטר 3x3.
6. ReLU: פונקציית אקטיבציה ReLU.
7. SeparableConv 2048, 3x3: קונבולוציה מופרדת לעומק עם 2048 פילטרים, גודל פילטר 3x3.
8. ReLU: פונקציית אקטיבציה ReLU.

הארכיטקטורה הנ״ל היא זו שאני השתמשתי בה כאשר ביצעתי transfer learning. במצב בו משתמשים ברשת כפי שהיא נבנתה במקור יהיו עוד 3 שכבות בסוף שנועדו לסיווג של ה-ImageNet. בגלל שאני משתמשת ברשת גם לסיווג וגם ללוקליזציה הורדתי את ״ראש״ הרשת והשתמשתי בה ללא השכבות האחרונות.

**שכבת קונבולוציה**

שכבת קונבולוציה היא השכבה העיקרית במודל רשת נוירונים מבוססת קונבולוציה (CNN), המשמשת לזיהוי תבניות בתמונות. השכבה משתמשת בפילטרים קטנים שנעים על פני התמונה ומחפשים תכונות כמו קצוות, זוויות וצורות פשוטות יותר. כל פילטר מייצר מפת תכונות (feature map) שמייצגת את המיקומים בתמונה שבהם הוא זיהה את התבנית שהוא חיפש. השכבות הבאות משמשות לזהות תבניות מורכבות יותר על ידי שילוב של מפות התכונות הקודמות. שכבת הקונבולוציה מאפשרת למודל ללמוד תכונות רלוונטיות אוטומטית, תוך שמירה על יחס המרחב במידע התמונה ובשימוש בכוח מחשוב נמוך משמעותית מרשת נוירונים Fully connected.

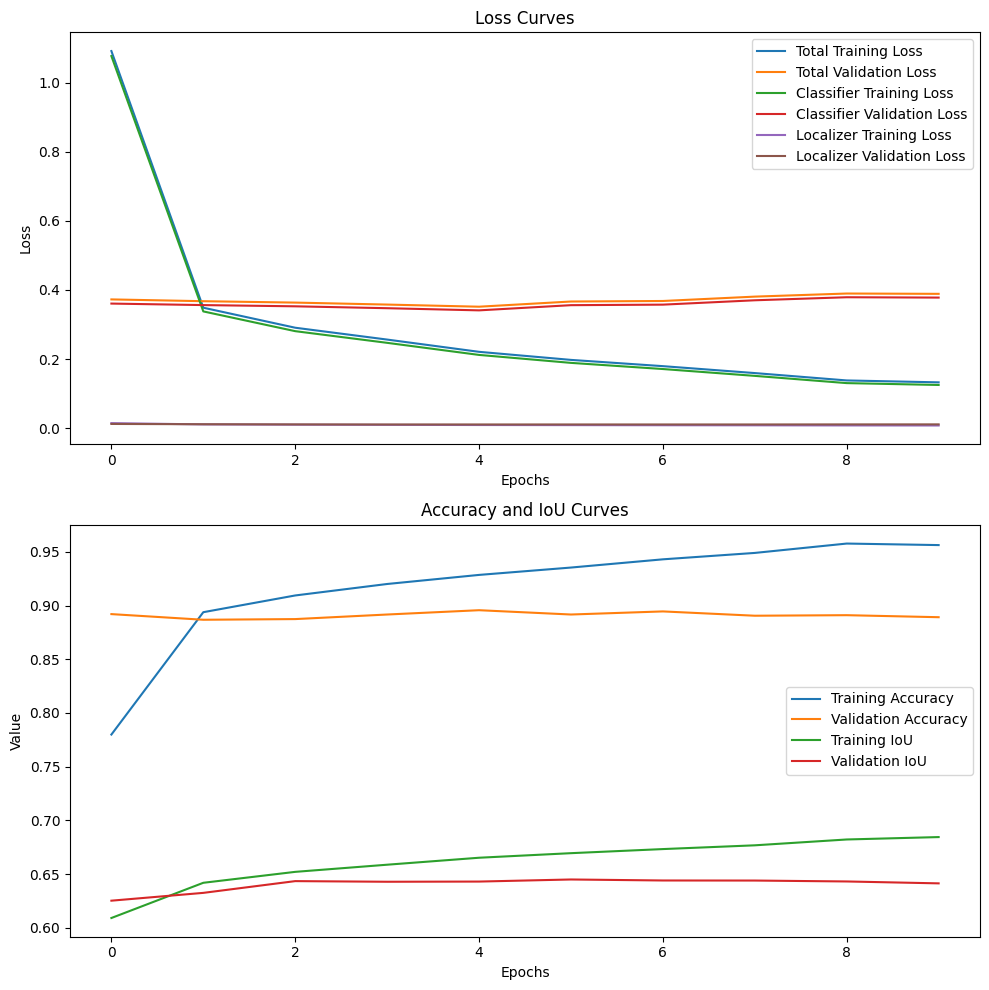
**שכבת Pooling**

שכבת פולינג (Pooling Layer) משמשת להפחתת המימדים של מפות התכונות שנוצרות על ידי שכבות הקונבולוציה, ובכך מקטינה את מספר הפרמטרים והחישובים במודל. התהליך הנפוץ ביותר הוא Max Pooling, שבו מחלקים את מפת התכונות לתאים קטנים, ובכל תא נבחר הערך המקסימלי. פעולה זו שומרת על התכונות החשובות ביותר במפות התכונות תוך הפחתת רעשים ושיפור הכלליות של המודל. כתוצאה מכך, שכבת הפולינג תורמת ליציבות ואפקטיביות האימון על ידי הפחתת הסיכוי לאוברפיט ושיפור היכולת של המודל להתמודד עם שינויים בתמונות הקלט.

**RelU**

ReLU היא פונקציית אקטיבציה המשמשת ברשתות נוירונים, ובמיוחד ברשתות קונבולוציה. היא מגדירה שכל ערך שלילי במפת התכונות יוגדר כ-0, בעוד שכל ערך חיובי יישאר ללא שינוי. פעולה זו מסייעת להכניס א-לינאריות למודל, ומאפשרת לו ללמוד ולייצג תכונות מורכבות יותר. יתרונה המרכזי של ReLU הוא בפשטות שלה וביכולת שלה להאיץ את תהליך האימון על ידי הפחתת בעיית היעלמות הגרדיאנטים (vanishing gradients), שמתרחשת כאשר גרדיאנטים קטנים מדי אינם מאפשרים עדכון אפקטיבי של המשקלים בשכבות הקודמות של הרשת.

### ניתוח הביצועים בשלב האימון והמבחן

זהו תקציר של תוצאות הניסוי האחרון שעשיתי, שבו יצרתי את המודל בו בחרתי להשתמש

loss: 0.3888

classifier\_loss: 0.3780

localizer\_loss: 0.0108

classifier\_accuracy: 0.8892

localizer\_IoU: 0.6414

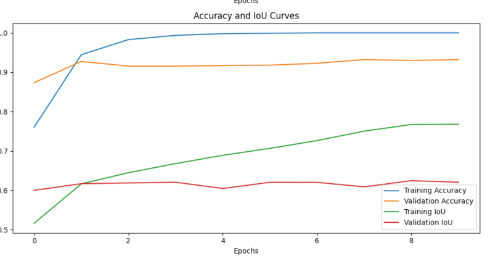
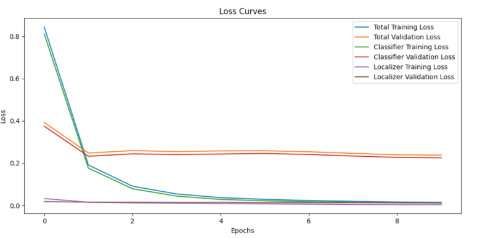
### 

זוהי הצגה ויזואלית של יכולות הסיווג והלוקליזציה של המודל הנבחר:

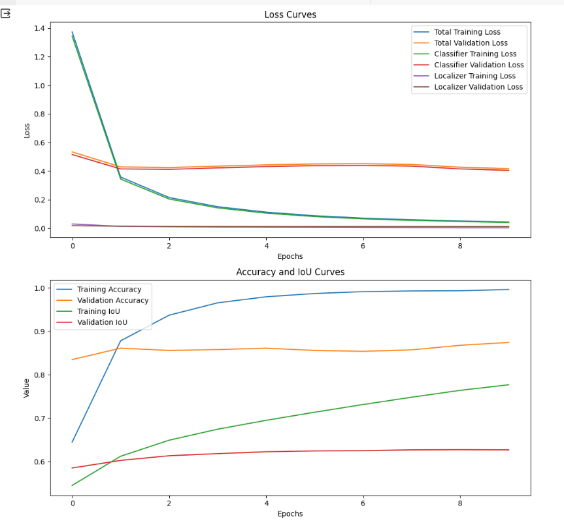
### 

### תיאור השינויים והניסויים שנעשו במהלך העבודה

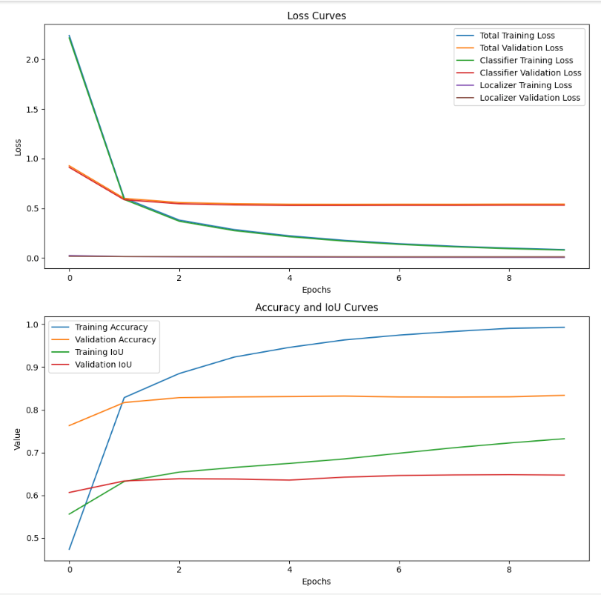
1. בשלבים הראשונים, בחנתי את ביצועי הארכיטקטורה MobileNetV2 על 10 גזעי כלבים בלבד כנקודת התחלה. המודל הראה דיוק ולידציה של 93.20%, IoU של 62.06% ו-loss ולידציה של 0.2377.



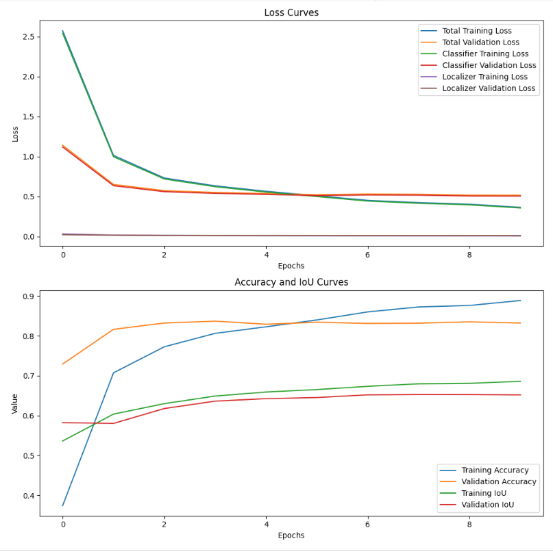
1. לאחר מכן, העליתי את מספר הגזעים ל-20 ללא שינוי היפרפרמטרים, כדי לבחון את השפעת הסיבוכיות הגוברת של הבעיה. חלה ירידה בדיוק הולידציה ל-87.43% אך שיפור קל ב-IoU ל-62.66%, עם loss ולידציה של 0.4173.



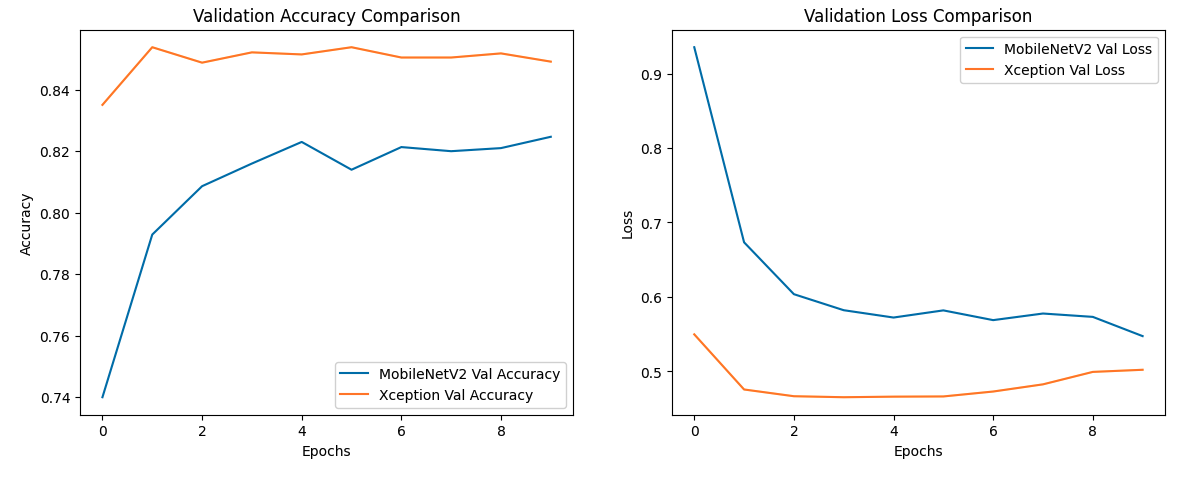
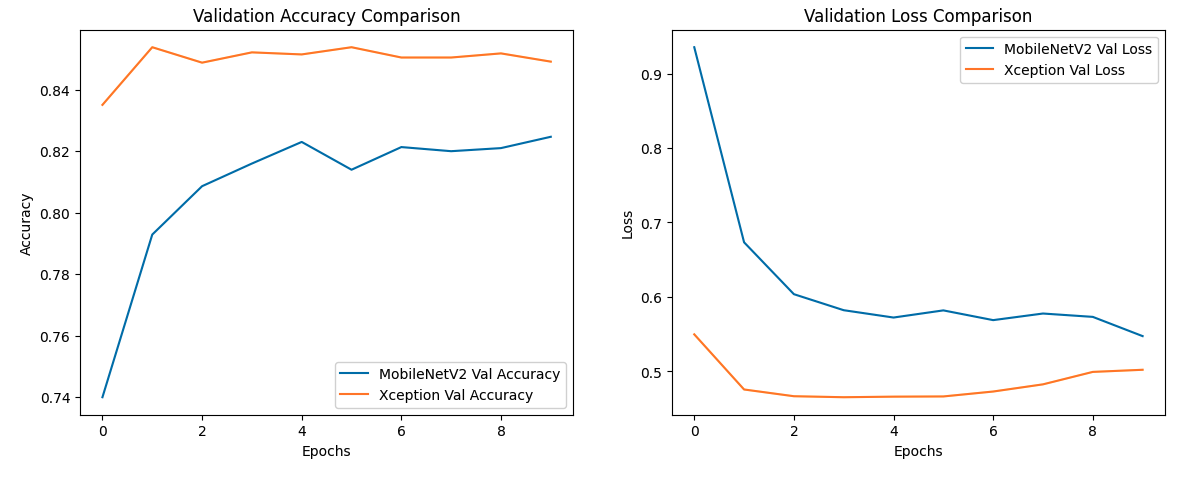
1. בניסוי נוסף, העליתי את מספר הגזעים ל-40 תוך הגדלת גודל ה-batch במקביל כדי לשמור על זמני אימון סבירים. ללא אוגמנטציה, הדיוק הגיע ל-83% והיה ניכר overfit בהשוואה לדיוק האימון של 99%.



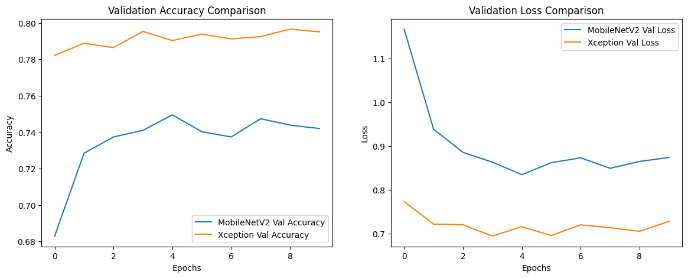
1. לאחר הניסויים הראשונים, כדי לטפל בבעיית ה-overfit הוספתי טכניקת אוגמנטציית נתונים המבוססת על עבודות קודמות בתחום, הכוללת: היפוך תמונות אופקי אקראי וסיבוב בזוויות עד 20%. התוצאות השתפרו עם דיוק ולידציה של 83.20%, IoU של 65.20% ו-loss ולידציה של 0.5151, לעומת דיוק אימון של 88.90%.

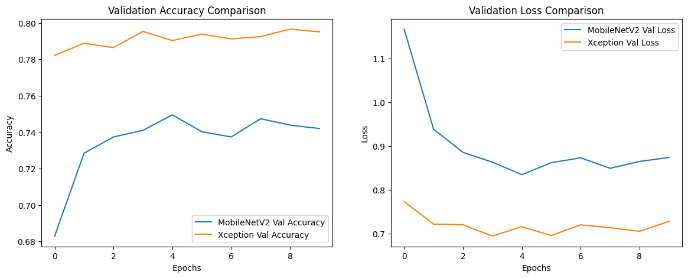


1. לאחר מכן, השוויתי את ביצועי MobileNetV2 לארכיטקטורת Xception. בשלב זה לא הצלחתי להביא את Xception לביצועים טובים יותר באופן משמעותי מה-MobileNetV2, אבל ידעתי שברגע שאתחיל לשלם על כוח חישוב חזק יותר אוכל לשפר את ביצועיה משמעותית, מתוך המאמר שקראתי (״Dog Breed Classification and Visualisation״ (Candido, 2020)), על כן בחרתי להמשיך איתה.

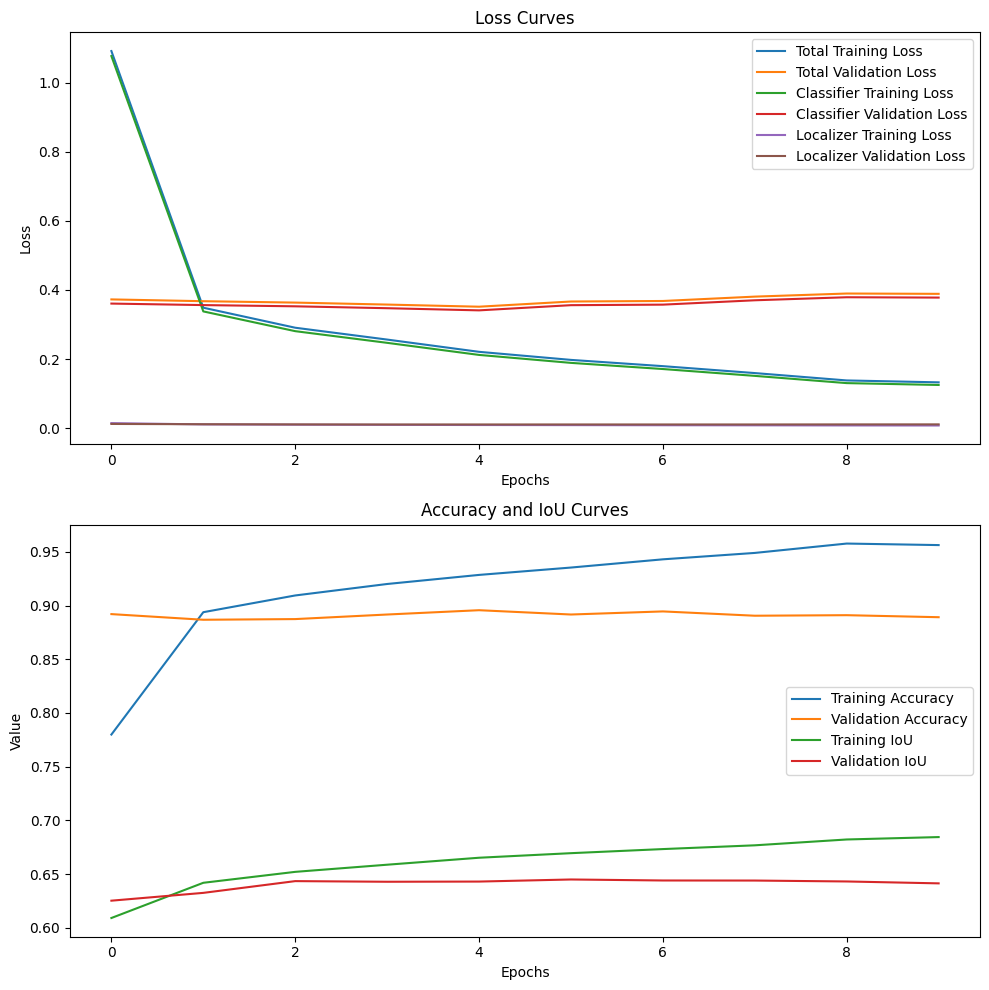


1. ברגע שהגדלתי את כוח החישוב שהיה ברשותי עברתי לסיווג של כל 120 הזנים שנמצאים ב-Stanford Dogs, אבל התוצאות שהגעתי אליהן היו נמוכות בהרבה מהמצופה מ-Xception. הגעתי ל-79% דיוק על נתוני הולידציה, loss ולידציה של 0.72 ו-IoU של 65%. רציתי לוודא שאכן ההשוואה הייתה נכונה, אז השוויתי שוב בין שתי הרשתות:





1. בשלב זה הבנתי שהתוצאות של Xception היו נמוכות משמעותית מהמצופה כיוון שהשתמשתי בנרמול לא נכון של ערכי הפיקסלים, כלומר הסתמכתי על הנרמול שנעשה בעבור המוביילנט, שאינו תואם לאחד הנחוץ בעבור Xception. לאחר שהחלפתי את הנרמול הגעתי לתוצאות גבוהות בהרבה על 120 זנים - דיוק ולידציה של 92%, loss ולידציה של 0.2710 ו-IoU של 64.3%.



### טבלה מסכמת של כל הניסויים

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ניסוי ראשון** |  |  |  |  |
| מספר זנים | **10** | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר אפוקים | 10 | 93.20% | 62.06% | 0.2377 |
| batch size | 32 | דיוק אימון | IoU אימון | loss אימון |
| מודל בסיס | MobileNetV2 | 100.00% | 76.77% | 0.0143 |
| אוגמנטציה? | לא |  |  |  |
| **ניסוי שני** |  |  |  |  |
| מספר זנים | **20** | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר אפוקים | 10 | 87.43% | 62.66% | 0.4173 |
| batch size | 70 | דיוק אימון | IoU אימון | loss אימון |
| מודל בסיס | MobileNetV2 | 99.60% | 77.67% | 0.0437 |
| אוגמנטציה? | לא |  |  |  |
| **ניסוי שלישי** |  |  |  |  |
| מספר זנים | **40** | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר אפוקים | 10 | 83% | 64.80% | 0.5413 |
| batch size | 150 | דיוק אימון | IoU אימון | loss אימון |
| מודל בסיס | MobileNetV2 | 99% | 72.24% | 0.0986 |
| אוגמנטציה? | לא |  |  |  |
| **ניסוי רביעי** |  |  |  |  |
| מספר זנים | 40 | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר אפוקים | 10 | 83.20% | 65.20% | 0.5151 |
| batch size | 150 | דיוק אימון | IoU אימון | loss אימון |
| מודל בסיס | MobileNetV2 | 88.90% | 68.60% | 0.3641 |
| אוגמנטציה? | **כן** |  |  |  |
| **ניסוי חמישי** | **מודל 1** |  |  |  |
| מספר זנים | 40 | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר אפוקים | 10 | 82.47% | 65.90% | 0.5474 |
| batch size | 150 | דיוק אימון | IoU אימון | loss אימון |
| מודל בסיס | **MobileNetV2** | 89.10% | 69.10% | 0.3475 |
| אוגמנטציה? | כן |  |  |  |
| **ניסוי חמישי** | **מודל 2** |  |  |  |
| מספר זנים | 40 | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר אפוקים | 10 | 85.20% | 64.70% | 0.5022 |
| batch size | 150 | דיוק אימון | IoU אימון | loss אימון |
| מודל בסיס | **Xception** | 90.50% | 67.30% | 0.3097 |
| אוגמנטציה? | כן |  |  |  |
| **ניסוי שישי** |  | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר זנים | **120** | 79.5% | 65% | 0.7281 |
| מספר אפוקים | 10 | דיוק אימון | IoU אימון | loss אימון |
| batch size | 150 | 81.5% | 67% | 0.5958 |
| מודל בסיס | Xception |  |  |  |
| אוגמנטציה? | כן |  |  |  |
| טווח נרמול | [0,1] |  |  |  |
| **ניסוי שביעי** |  | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר זנים | 120 | 89.24% | 64.4% | 0.4080 |
| מספר אפוקים | 10 | דיוק אימון | IoU אימון | loss אימון |
| Batch size | 32 | 98.6% | 70.7% | 0.0625 |
| מודל בסיס | Xception |  |  |  |
| אוגמנטציה? | כן |  |  |  |
| טווח נרמול | [1,1-] |  |  |  |

### היפר פרמטרים במודל הנוכחי

כפי שניתן לראות, בחרתי בסוף להשתמש במודל ה-Xception כמודל הסופי של הפרויקט.

### 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **המודל** |  | דיוק ולידציה | IoU ולידציה | loss ולידציה |
| מספר זנים | 120 | 89.24% | 64.4% | 0.4080 |
| מספר אפוקים | 10 | דיוק אימון | IoU אימון | loss אימון |
| Batch size | 32 | 98.6% | 70.7% | 0.0625 |
| מודל בסיס | Xception |  |  |  |
| אוגמנטציה? | כן |  |  |  |
| טווח נרמול | [1,1-] |  |  |  |
| אופטימייזר | nadam |  |  |  |

### פונקציית השגיאה

במודל שלי, כיוון שביצעתי שתי פעולות שונות בעזרת אותו המודל, השתמשתי בשתי פונקציות שגיאה שונות.

**סיווג:**

למטרת הסיווג השתמשתי ב-Sparse Categorical Crossentropy, שהיא פונקציית הפסד הנפוצה ביותר עבור בעיות סיווג מרובות מחלקות (multi-class classification). היא משמשת להערכת הביצועים של מודל למידת מכונה כאשר הפלט הוא הסתברויות שמשתמשות בפונקציית softmax. המטרה היא למזער את פונקציית ההפסד כדי שהמודל ינבא הסתברויות קרובות ככל האפשר לתוויות האמת (הנתונים המסווגים).

הפונקציה מחשבת את ההפרש בין ההסתברויות החזויות על ידי המודל לבין ההסתברויות בפועל (תוויות האמת) באמצעות הנוסחה הבאה:

כאשר:

* N היא מספר המחלקות
* היא התווית האמיתית (0 או 1 בעבור כל מחלקה)
* היא ההסתברות שהמודל חוזה למחלקה i

במילים אחרות, Cross-Entropy Loss משווה בין ההסתברות של החיזוי של המודל לבין התוויות האמיתיות ומחשבת את ההפסד על ידי חישוב הלוגוריתם של ההסתברויות של המודל והכפלתו בערכים האמיתיים.

**לוקליזציה:**

למשימת הלוקליזציה השתמשתי ב-MSE, או בשמה המלא Mean Squared Error, שהיא פונקציית הפסד הנפוצה מאוד בשימוש במודלים של למידת מכונה, במיוחד במשימות של רגרסיה. פונקציה זו מחשבת את ממוצע הריבועים של ההפרשים בין הערכים החזויים לערכי האמת, ובכך מודדת את השגיאה של המודל. המטרה היא למזער את פונקציית ההפסד כדי לשפר את דיוק התחזיות של המודל.

החישוב נעשה על פי הנוסחה הבאה:

כאשר:

* N היא כמות הדוגמאות
* הוא הערך האמיתי של הדוגמה i
* הוא הערך החזוי של הדוגמה i

במילים אחרות, הפונקציה מחשבת את ההפרשים בין ערכי האמת לערכים החזויים, ולאחר מכן מעלה כל הפרש בריבוע, כדי להבטיח שערך השגיאה תמיד יהיה חיובי ולתת יותר משקל לשגיאות גדולות יותר.

### מדדי הביצוע

**סיווג:**

על מנת למדוד את יכולות הסיווג של המודל שלי השתמשתי במדד הדיוק (accuracy), שבודק בכמה אחוזים מתוך המדידות המודל נתן את התשובה הנכונה (כלומר סך הבדיקות בהן המודל צדק מתוך כל סך הבדיקות). בחרתי במדד זה כיוון שמסד הנתונים שלי מאוזן באופן יחסי, כך שאיני מפחדת שהמודל יחזה באופן קבוע את אותה מחלקה (או מספר מחלקות) ובכך יקבל אחוזי דיוק גבוהים ממה שהם אמורים להיות. בנוסף זהו מדד פשוט להבנה ונוח להשתמש בו בתור רפרנס לביצועי המודל.

**לוקליזציה:**

על מנת למדוד את יכולות הלוקליזציה של המודל השתמשתי בפונקציית IoU או Intersection over Union. מדד זה משתמש בשטח הכולל של ה-bounding box הנכון לפי הנתונים המקוריים ומוסיף אותו לזה החזוי עבור ה-Union, ומחשב את ההפרש ביניהם כ-Intersection. כך הוא מבין כמה אחוזים מהסימון האמיתי באמת באים לידי ביטוי בסימון החזוי, ובכך יכול להעריך כמה טוב המודל בלוקליזציה של האובייקט הנתון. אבל, יש לשים לב שזהו מדד מטעה מעט, כיוון שתוצאת IoU של כ-60% לרוב תחשב תוצאה טובה, למרות שישנם מדדים בהם תוצאה נמוכה מ-90% תחשב תוצאה בעייתית במשימה כמו זו.

## יישום

### הסבר על היישום ואיך הוא משתמש במודל

היישום שאני בחרתי לבנות הוא אפליקציה לאייפון (כלומר ל-ios), שתאפשר למשתמש לצלם או להעלות תמונה מתוך ספריית התמונות בטלפון, ולאחר מכן תתאים את גודלה לגודל הקלט המצופה במודל, תסווג אותה ותבצע לוקליזציה, תחזיר אותה לגודלה המקורי ותציג למשתמש את תוצאות הסיווג ואת התמונה המקורית עם האנוטציות החזויות.

לאפליציה כרגע קוראים DogLocalizerSUI ואינה מאפשרת שימוש במכשירים נוספים, והיא משתמשת במודל הסיווג והלוקליזציה שלי.

כל תכנות וריצת האפליקציה מבוססים על שפת Swift ועל ספריית SwiftUI של אפל, שנועדה לתכנות אפליקציות למערכת ההפעלה שלה.

### מחלקות האפליקציה

בניתי את האפליקציה בעזרת מספר מחלקות, מטרת חלקן היא עיצובית בלבד וחלקן נועדו על מנת לבצע את הסיווג והלוקליזציה:

1. DogLocalizerSUIApp - המחלקה הראשית, היא זו שמאפשרת את הפעלת האפליקציה. כוללת בתוכה את מחלקת rootView, כלומר את עמוד הבית של האפליקציה
2. HomePage - המחלקה שמכילה בתוכה את עמוד הבית עצמו, בעיקר מחלקה עיצובית אבל כוללת גם 4 כפתורים - כפתור שפותח את סרגל הניווט, כפתור שפותח את ההסבר על האפליקציה, ושני כפתורים שמאפשרים לצלם או להעלות תמונה על מנת לסווג אותה
3. About - מחלקה עיצובית לחלוטין שנועדה להציג מעין עמוד ReadMe על האפליקציה, שכולל הסבר קצר על הסיבה שיצרתי אותה ואיך היא עובדת
4. SideMenu - סרגל הניווט של האפליקציה, שכרגע מאפשר להכנס לעמוד הבית או להסבר על האפליקציה. בעתיד אני מקווה להשמיש גם את שני הכפתורים הנוספים - כניסה להגדרות האפליקציה ובניית ספריית תמונות מסווגות לכל משתמש
5. ImagePicker - מחלקה שנועדה לאפשר העלאת תמונה מספריית התמונות בטלפון או מתוך המצלמה, ולהשתמש בה לדברים שונים באפליקציה
6. PhotoViewerView - עמוד הכולל 4 כפתורים, ונפתח לאחר שהמשתמש בוחר תמונה מהמצלמה או מהטלפון. שניים מהכפתורים הם כמו בעמוד הבית - פתיחת סרגל הניווט ועמוד ההסבר על האפליקציה. שני הכפתורים הנותרים מאפשרים למשתמש לסווג את התמונה שבחר, ואם אינו מרוצה ממנה, להחליף אותה בתמונה אחרת.
7. NavigationManager - מחלקה שנועדה לטפל בהתנהלות עם מספר עמודים שונים באפליקציה, והמעבר בין אחד לשני. במצב הנוכחי יכולתי להסתדר בלעדיה אבל במידה וארצה להרחיב את האפליקציה היא תקל על העבודה מאוד.
8. ModelManager - מחלקה שמטרתה לעבוד מול המודל באפליקציה. במחלקה פונקציות להתאמת התמונות לגודל הנחוץ למודל, פונקציית סיווג הקוראת למודל עצמו, ופונקציה שנועדה להמיר את תוצאות הסיווג והלוקליזציה לערכים שניתן להציג באופן ויזואלי (למשל המרת מספר המחלקה לשם זן הכלב)
9. RootView - מחלקה זו מסתמכת על ה-NavigationManager, והיא כוללת בתוכה את עמודי האפליקציה על מנת שיהיה קל יותר להחליף ביניהם. בנוסף, בה מתואר המצב הדיפולטיבי של כל עמוד באפליקציה
10. PhotoSwipeView - מחלקה שנועדה לשמור בתוכה 4 תמונות, ביניהן ניתן להחליף על ידי החלקה על המסך. מחלקה זו היא עיצובית בלבד ונועדה על מנת להקל על המשתמשים ללמוד על השימוש באפליקציה בקלות בעמוד הבית

### 

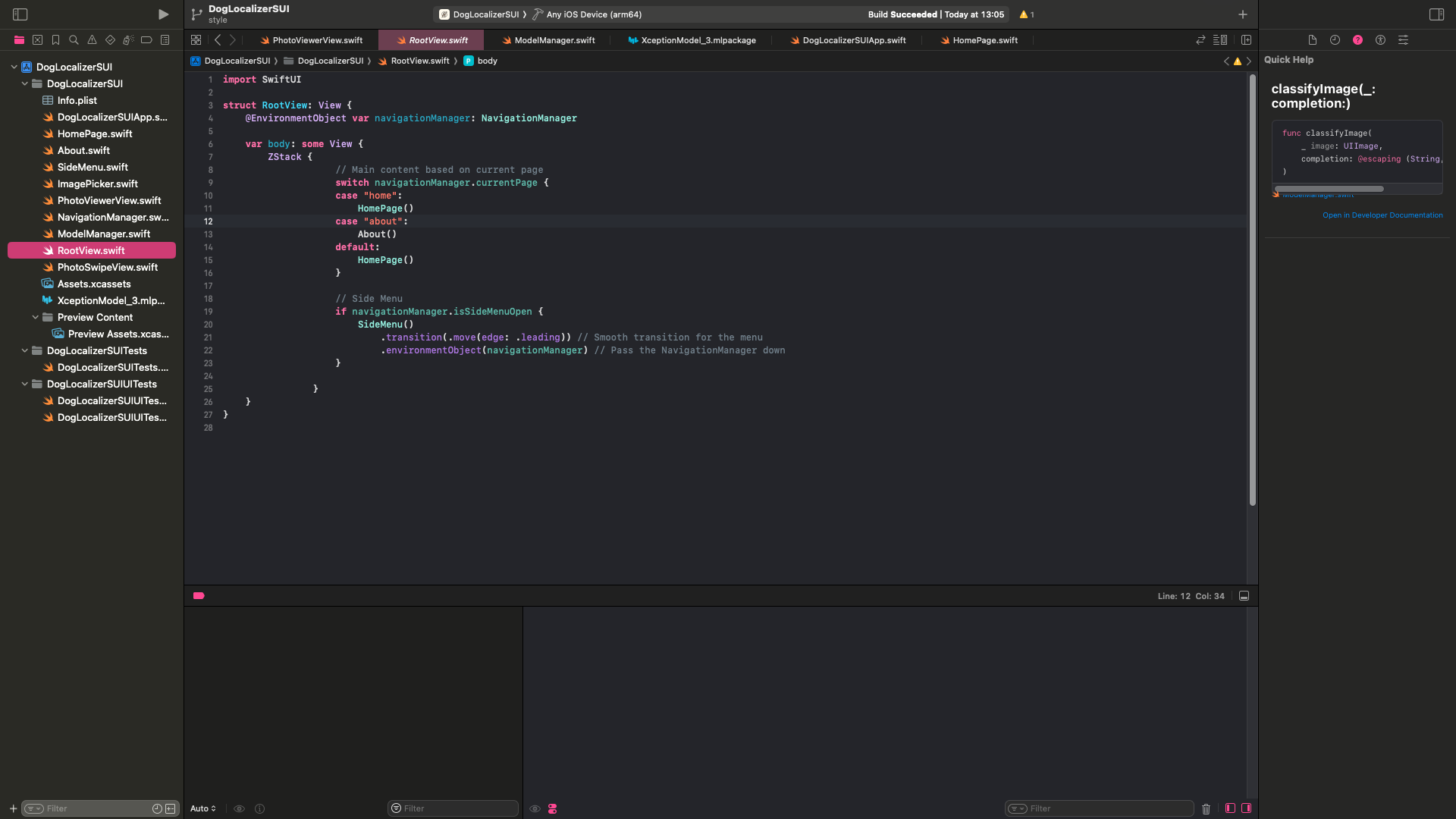
### תרשים UML

### 

### 

### הטכנולוגיה בה השתמשתי - SwiftUI

בשלב היישום, על מנת לבנות אפליקציה פשוטה ונוחה למשתמש ולמפתח, ולאפשר לעצמי את החופש שאני מחפשת בעיצוב האפליקציה שממשקי משתמש פשוטים יותר לא אפשר לי, ובנוסף לאתגר את עצמי ולהרחיב את האופקים שלי, בחרתי להשתמש בפלטפורמת Xcode. ספציפית, בחרתי לכתוב את האפליקציה בשפת Swift, בשימוש ב-SwiftUI של אפל, שבנוי לחלוטין על שפת Swift.

Xcode היא תוכנה למאק בלבד, והיא התוכנה הרשמית היחידה שמאפשרת בניית אפליקציות לאייפון, אייפד, מאק וכו׳. בחרתי להשתמש בה בגלל שבעבר יצא לי להתנסות איתה קצת וזכרתי שהיא מאוד אינטואיטיבית ונוחה לשימוש. למרות זאת, הפעם בחרתי להשתמש בפלטפורמת פיתוח שלא הכרתי (כיוון שהיא גם מאוד חדשה) - SwiftUI. בשונה מהפלטפורמה הקודמת, StoryBoards, שהשתמשה בעיקר בגרירת אובייקטים על המסך ובניית האפליקציה באופן ויזואלי כמעט לחלוטין, SwiftUI בנויה על קבצי קוד בשפת Swift. בחרתי להשתמש בה גם כיוון שהיא מאפשרת הצגה תמידית של מראה האפליקציה בזמן הנתון וכך מקלה מאוד את העיצוב והתכנות, וגם כיוון שזו הרגישה כמו הזדמנות נהדרת ללמוד קצת על שפת Swift.

חלון העבודה בXcode נראה כמו חלון של כל תוכנת פיתוח אפליקציות אחרת, כיוון שכפי שאמרתי היא מתבססת באופן מלא על כתיבת קוד. למרות זאת, התוכנה מאוד אינטראקטיבית ונוחה למשתמשים מתחילים (בהנחה שמנסים לבנות אפליקציה פשוטה), בעיקר בזכות התפריט בפינה הימנית העליונה, שמאפשר הכנסת אלמנטים אל המסך באופן ידני ואולי יותר אינטואיטיבי לחלק מהמשתמשים. בנוסף, שמות המשתנים והפונקציות לעיצוב העמודים הם, ברובם, מאוד אינטואיטיבים, כך שמהר מאוד נהיה נוח יותר להקליד את שמות המשתנים מאשר להכניס אותם באופן ידני אחד אחד.

בנוסף, היתרון העיקרי של SwiftUI לעומת Storyboards הוא היכולות הטובות משמעותית שלה, בעיקר ברמה הגבוהה יותר של יכולות שליטה בכל משתנה, בין אם עיצובי או פונקציונלי.

שמחתי מאוד גם לגלות, שבאופן יחסי לתוכנה שלא נועדה באופן ספציפי לאימון ושימוש במודלי למידת מכונה, היה נוח מאוד לעבוד עם התוכנה והצלחתי להבין את הפונקציות והמשתנים השונים יחסית מהר.

### התאמת נתוני החיזוי למודל

כיוון שרציתי שהאפליקציה תוכל לפעול ללא כל קשר למחשב או לקולאב, הייתי צריכה לייצא את המודל לקובץ שיתקבל ב-xcode. לשם כך השתמשתי בספריית coremltools, שנועדה לכלים של עבודה עם מודלי למידת מכונה, ביניהם היכולת להמיר קובץ מקובץ של tensorflow לקובץ שמאפשר את אינטגרציה של המודל באפליקציה.

לאחר מכן בניתי את מחלקת ה-ModelManager, כיוון שאומנם Xcode יוצר ספרייה אוטומטית לכל מודל אך היא מאוד דומה לזו שקיימת בkeras - היא מאפשרת לחזות באמצעות המודל, אך לא מעבר לכך. היא לא נותנת להמיר את התחזיות לפורמט ברור או לעשות מניפולציה עליהן בצורה מסויימת.

בנוסף, הייתי צריכה לשנות את גודל התמונות כך שיתאימו לגודל המצופה במודל, ממש כפי שעשיתי במהלך האימון והולידציה. לכן, כתבתי פונקציית preprocessing שכוללת את התאמת הגודל של התמונה לגודל הנחוץ.

את נרמול ערכי הפיקסלים המודל עושה באופן עצמאי, כך שלא הייתי צריכה להפעיל צורות נוספות של התאמה של הנתונים.

# מדריך למפתח

## המודל

קישור לתדפיס המלא של מחברת הקוד: [תדפיס הקוד - מודל לפרויקט הגמר.pdf](https://drive.google.com/file/d/1NO_aSqcbSkgb3T11Ko3AulHW9T4ze2Nf/view?usp=drive_link)

**קוד יצירת המודל**

data\_augmentation = tf.keras.Sequential([

layers.experimental.preprocessing.RandomFlip("horizontal"),

layers.experimental.preprocessing.RandomZoom(0.2),

])

from tensorflow.keras.applications import Xception

# Create the Xception base model

xception\_base = Xception(input\_shape=(\*IMAGE\_SHAPE, 3), include\_top=False, weights='imagenet')

xception\_base.trainable = False # Freeze the convolutional base

# Reuse the same input layer

inputs = tf.keras.Input(shape=(None, None, 3)) # Accepts images of any size

x = keras.applications.xception.preprocess\_input(inputs)

# Apply the same data augmentation

x = data\_augmentation(x)

# Use Xception as the base model

x = xception\_base(x, training=False)

x = layers.GlobalAveragePooling2D()(x)

# Classifier head

label\_head\_xception = layers.Dense(128, activation='relu')(x)

output\_label\_xception = layers.Dense(NUM\_CLASSES, activation='softmax', name='classifier')(label\_head\_xception)

# Localizer head

bbox\_head\_xception = layers.Dense(128, activation='relu')(x)

output\_bbox\_xception = layers.Dense(4, activation='sigmoid', name='localizer')(bbox\_head\_xception)

# Build the Xception model

xception\_model = keras.Model(inputs, [output\_label\_xception, output\_bbox\_xception])

# Compile the Xception model

xception\_model.compile(optimizer='nadam',

loss={'classifier': 'sparse\_categorical\_crossentropy',

'localizer': 'mse'},

metrics={'classifier': 'accuracy',

'localizer': IoU})

## היישום

על מנת להקל על העברת הקוד וכיוון שאני משתמשת ב-git וב-github על מנת לשמור על הפרויקט שלי ללא קשר, בחרתי לשים כאן לינק לעמוד ה-github שבו נמצא כל הקוד של האפליקציה שלי, כולל הגרסאות הקודמות בהן ניסיתי דרכים שונות לעשות אימפלמנטציה למודל וכל ניסוי שעשיתי עד שהגעתי לאפליקציה הנוכחית.

להלן הקישור:

<https://github.com/danaohayon/DogLocalizerSUI>

כל קובץ קרוי על שם המחלקה שנכללת בתוכו. בנוסף לקבצי המחלקות יש כמה קבצים חשובים נוספים:

1. Assets - הקובץ הכולל כל קובץ חיצוני שאני רוצה לשתמש בו באפליקציה, תמונות, לוגואים וכו.
2. XceptionModel\_3 - המודל בגרסא המותאמת לXcode
3. Info - קובץ המכיל כמה הרשאות שחובה שיהיו לאפליקציה (במקרה הזה, שימוש במצלמה ובספריית התמונות של הטלפון)

בכל המחלקות ישנן הערות בקוד על מנת להפוך אותו לקריא וברור יותר.

### 

# מדריך למשתמש

### התקנת האפליקציה:

כרגע לא ניתן להתקין את האפליקציה על מכשירים נוספים באופן קבוע, כיוון שהעלאתה לחנות האפליקציות תעלה כסף. על מנת להשתמש בה על מכשיר נוסף ניתן לבצע את השלבים הבאים:

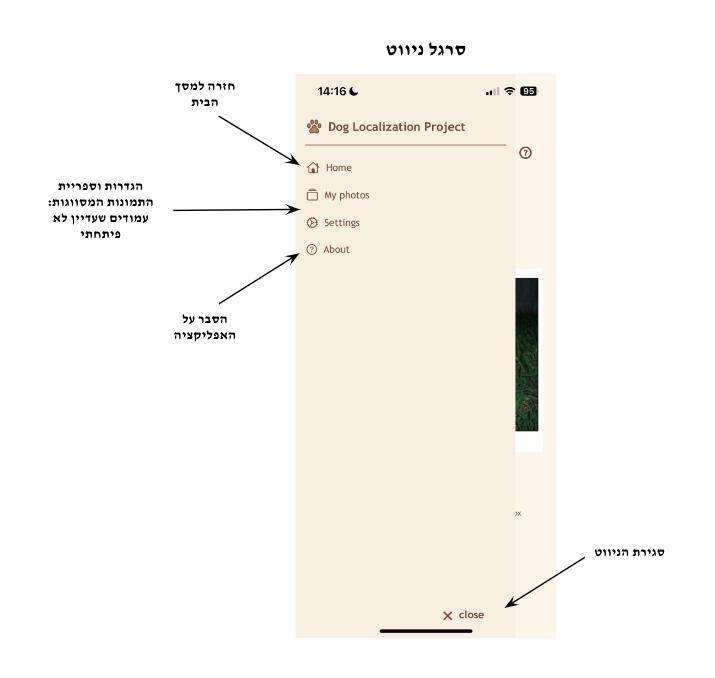
1. להוריד למחשב את קוד האפליקציה וכל הקבצים הנחוצים (כל הקבצים בתיקיית הפרויקט)
2. לחבר את המכשיר באופן חוטי למחשב (חשוב לציין, זוהי אפליקציה שנועדה לאייפון וניתן להריץ אותה רק בשימוש באייפון ובמאקבוק)
3. כאשר מחברים את הטלפון למחשב תופיע התראה שתשאל את המשתמש האם לסמוך על המכשיר, חובה לאשר על מנת שיהיה אפשר להריץ את האפליקציה
4. להכנס לאחד מקבצי האפליקציה
5. לגשת לפאנל העליון בחלון בו מופיע שם האפליקציה (במקרה הזה DogLocalizerSUI) וללחוץ על שם המכשיר שמופיע ליד החץ
6. ברשימת המכשירים למצוא את הטלפון הרצוי וללחוץ עליו

\*במידה והמכשיר לא נמצא יש ללחוץ על Manage run destinations בתחתית הרשימה ולחבר את הטלפון משם

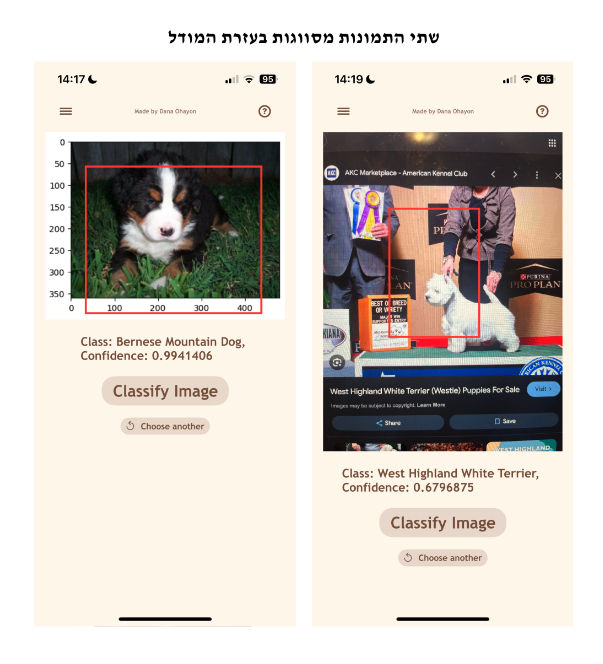
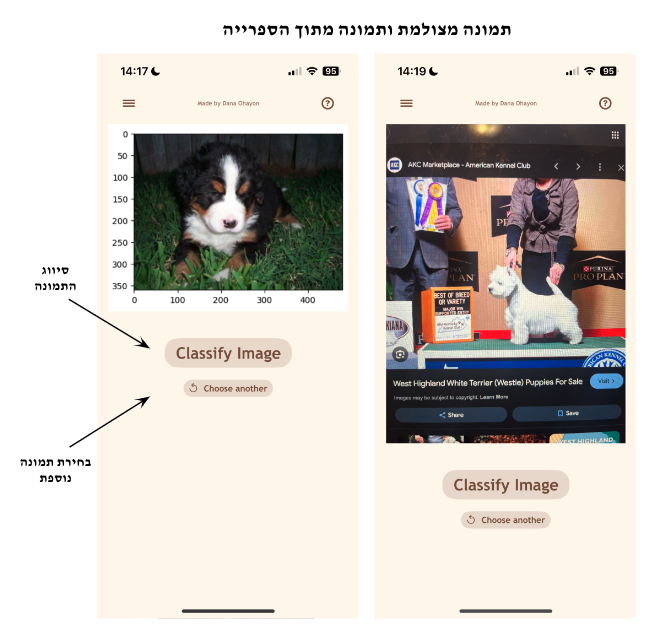
1. לאחר מכן יש ללחוץ על המשולש בצד שמאל של המסך, על מנת להריץ את האפליקציה (לא ניתן לנתק את הכבל ולהמשיך להריץ את האפליקציה במצב הזה על כן יש להשאיר את המכשיר מחובר)

### תרשים של מסכי האפליקציה

### הסבר על כל מסך והכפתורים שבו



# 



# רפלקציה אישית

הפרוייקט הזה הוא אחד מהמהנים והמעניינים ביותר שהייתה לי ההזדמנות לעשות לכל אורך שנותיי בתיכון, ויכול להיות שהוא אפילו המעניין ביותר. מעבר לכך שאני מאוד אוהבת ומתחברת לנושא של למידה עמוקה באופן כללי, הפרוייקט הזה באופן ספציפי נתן לי הזדמנות פז ללמוד דברים חדשים, להתנסות בטכנולוגיות שלא יצא לי להתעסק איתן אף פעם וגם לחזק את כישורי העיצוב הגרפי שלי.

אני חושבת שברמה המקצועית קיבלתי המון ידע ויכולות שלא היו לי קודם - קודם כל, כפי שציינתי, למדתי לתכנת בשפת Swift. בנוסף לכך, במהלך העבודה על הפרוייקט למדתי על מספר אלגוריתמים ונושאים מאוד מעניינים בהקשר של למידת מכונה שלצערי לא הצלחתי להשתמש בהם בסוף. למשל, למדתי על object detection ואלגוריתם YOLO, משהו שבחיים לא הייתי מגיעה אליו ללא הפרויקט, או על הדרך לבנות מפת חום שמבוססת על יכולת החיזוי של המודל שלי.

גם ברמת יכולות העבודה שלי למדתי המון מהפרויקט, בעיקר מהאספקטים שבהם אני חושבת שיכולתי להתנהל בצורה יותר טובה. החלטתי להחליף את נושא הפרוייקט בשלב מאוחר יחסית, מה שהכריח אותי לעמוד בלו״ז הרבה יותר צפוף ולעבוד הרבה יותר קשה, ובנוסף הכריח אותי להפוך את הפרוייקט לפשוט יותר מהמתוכנן. זה היה כנראה האתגר הגדול ביותר בשבילי בעבודה על הפרוייקט - ההבנה שלפעמים אני צריכה להוריד קצת את הציפיות מהפרוייקט שלי על מנת להצליח להוציא אותו לפועל הלוח הזמנים הנתון, והביטחון בעצמי שאם באמת ארגיש שחשוב לי להמשיך ולהתעמק בנושא, אעשה זאת גם ללא לחץ הזמן הקשה של כתיבת הספר והפרוייקט.

אבל, לחץ הזמן גם לימד אותי המון על עצמי, על יכולות העמידה שלי בזמנים ובעיקר על הדברים שמעניינים אותי - אם היו שואלים אותי לפני שנה על מה הייתי בוחרת לעשות את הפרוייקט, בחיים לא הייתי חושבת שאהנה כל כך לשבת וללמוד לבנות אפליקציה, ואז לבנות אחת לגמרי מאפס.

אם הייתי מתחילה היום את הפרוייקט מחדש, אני חושבת שבעיקר הייתי עובדת בצורה מסודרת יותר ומתעדת את הטעויות שאני עושה בצורה טובה יותר, כיוון שיצא לי להתקל באותן הבעיות מספר פעמים בחלקים שונים בפרוייקט, וכיוון שלא תיעדתי כיצד פתרתי את אותה הבעיה בפעם הקודמת שנתקלתי בה, לא ידעתי מה לעשות ונאלצתי לחפש שוב דרכי פתרון.

מעבר לכך, אני לא חושבת שהייתי משנה עוד הרבה - נהניתי מההזדמנות ללמוד על המון נושאים חדשים, למשל OCR, שעליו תכננתי לעשות את הפרוייקט, או object detection כפי שציינתי קודם, גם אם בדיעבד הלמידה הזו הייתה סוג של בזבוז זמן, כי בסופו של דבר לא השתמשתי בנושאים האלו בפרוייקט הזה. אני חושבת שהיתרון האדיר במגמה ובפרוייקטים האלו הוא ההבנה שחשוב ללמוד לשם הלמידה, גם אם לא כל נושא יעזור לי באופן ישיר בפרוייקט או מטלה הנוכחיים, כי אולי זה משהו שאבין שאני רוצה להעמיק בו בעתיד, או אפילו רק לשם הידע הכללי הקצת יותר נרחב שלי.

בסופו של דבר, הפרוייקט הזה רק גרם לי להבין עוד יותר כמה יש לי וכמה אני רוצה עוד ללמוד, ואמנם אני בהחלט חושבת שיכולתי לנהל את הזמן שלי בצורה טובה יותר אבל באופן כללי אני מאמינה שזאת הדרך שהייתי צריכה לעבור, ואני מאוד מאוד מרוצה מהפרוייקט הסופי שלי, אז אני חושבת שזה היה משתלם בסופו של דבר.

# ביבליוגרפיה

1. Khosla, A. Jayadevaprakash, A. Yao, B. Fei-Fei, L. (2011). Novel dataset for fine-grained image categorization. Retrieved from <http://vision.stanford.edu/aditya86/ImageNetDogs/>
2. Candido, J. (2020). Dog breed classification and visualisation. Retrieved from <https://cs230.stanford.edu/projects_spring_2018/reports/8291007.pdf>
3. Chadha, A. (n.d.). Autonomous driving application: Car detection. GitHub. Retrieved May 17, 2024, from <https://github.com/amanchadha/coursera-deep-learning-specialization/blob/master/C4%20-%20Convolutional%20Neural%20Networks/Week%203/Car%20detection%20for%20Autonomous%20Driving/Autonomous_driving_application_Car_detection_v3a.ipynb>
4. Loye, G. (n.d.). Training YOLOv3 on the Stanford Dogs dataset. GitHub. Retrieved May 17, 2024, from <https://github.com/gabrielloye/yolov3-stanford-dogs/blob/master/train.ipynb>
5. ResearchGate. (n.d.). The Xception architecture [Figure]. Retrieved May 17, 2024, from <https://www.researchgate.net/figure/The-Xception-architecture-8_fig4_374740259>