Camerona

שם התלמיד: אופק חיימוביץ'
ת.ז: 325616324
בית ספר: כפר הנוער ויצו ניר העמק
שם המוקד ומספר הכיתה: מוקד עמק יזרעאל י"ב 14
שם הרש"צ: אילן פליוב
שם המנטור: אייל עינב
תאריך הגשה:

עבודת גמר

Machine Learning\Deep Learning

למידת מכונה

שנת הלימודים תשפ"א

2.בנה התיק - פרקים

3. מבוא

בעולם של היום, משתולל וירוס מסוכן בשם וירוס הקורונה (Covid-19), אחת הדרכים להאט את התפשטות הוירוס ולמנוע מקרי מוות רבים היא עטיית מסכה בכל מקום ציבורי. חלק מן האנשים לא מקשיבים להנחייה זו ובגלל זה נוצרה בעיה, אנשים אלו מסכנים את הציבור הרחב בגלל שאינם עוטים מסכה, כוחות המשטרה אוכפים את ההנחייה על ידי חלוקת דוחות לאנשים שנמצאים לא עוטים מסכה, אך הדבר לא מספק הרתעה מספיקה בגלל שאין משטרה בכל מקום, כך נוצר מצב בו במקומות בהם יש אכיפה גבוהה אנשים מקפידים יותר על עטיית מסכות ובמקומות בהם האכיפה פחות גבוהה אנשים פחות מקפידים על עטיית מסכה, כאן הפרויקט שלנו נכנס לתמונה, פרויקט Camerona נוצר בשביל לעזור לכוחות המשטרתיים לאכוף את הנחיית עטיית המסכות, ללא צורך בגיוס כוח משטרתי נוסף.

בתחילת העבודה היה עלינו לחקור האם קיים כלי כנו שלנו, מצאנו מספר כלים שהיו די דומים לרעיון שלנו, הכלים היו בעצם תוכנה שרצה על מחשב עם מצלמה, מצלמת את החלל ומציגה על מסך אנשים אשר לא עוטים מסכה, הפרויקט שלנו בעצם לוקח את הרעיון הבסיסי הזה ומוסיף לו פיצ'רים רבים. בנוסף לבדיקה האם אדם עוטה מסכה או לא, אנחנו בודקים את מינו של האדם (לפי הפנים שלו), האם לאדם יש זקן והאם הוא לובש משקפיים, לאחר שהוצאנו את כל הפיצ'רים על כל האנשים אשר נמצאו לא עוטים מסכה, הכלי פונה לאנשים באמצעות רמקולים ואומר להם לעטות מסכה באופן פרטי, הוא מתאר את האדם ומבקש ממנו לעטות מסכה, לאחר שהוא מבקש מאדם לעטות מסכה הוא שומר תמונה של פניו ביחד עם התאריך והשעה שהתמונה נלקחה. במידה ויש יותר מדי אנשים בטווח התמונה, הכלי ינגן הקלטה שמבקשת מהאנשים להתפזר מהאזור. אפשר להבין שהכלי שלנו מביא חידושים רבים ביחס לכלים הקיימים.

הבעיה עמה מתמודד הפרויקט שלנו היא הבעיה של אכיפה לא מספקת על עטיית מסכות, כפי שציינתי מקודם, בגלל שאין מספיק כוחות משטרתיים נוצר מצב בו יש מקומות בהם האכיפה על עטיית מסכות לא מספיק גבוהה. בחרנו בנושא זה מכיוון שהוא כולל טכנולוגיה שעניינה אותנו ורצינו לעבוד איתה, בנוסף לכך, הנושא עניין אותנו מכיוון שזוהי בעיה בעולם האמיתי ולמצוא לה פתרון היה רעיון נחמד ומעניין, המוטיבציה שלנו לעשות את הפרויקט היה בעיקר הטכנולוגיה שלו, שעניינה אותנו מאוד.

הדרישות לפרויקט היו רבות, מעבר לדרישות הרמה של מגשימים הצבנו לעצמנו עוד דרישות בשביל שנעשה פרויקט מעניין ואיכותי, הדרישות בשביל להתחיל את הפרויקט היו ידע מקדים בטכנולוגיית הבינה המלאכותית, הדרישות של הפרויקט עצמו היו רבות, הוא דרש המון זמן, השקעה, מחשבה, חקירה ולמידה, מכיין שאת המודלים שהפרויקט משתמש בהם אמנו וכתבנו בעצמנו היה עלינו לעבוד הרבה עם מידע גולמי, מעבר לחקירה הרבה והלמידה הרבה על נושא הבינה המלאכותית.

הידע התיאורטי הנדרש כרקע בשביל להבין את העבודה והאמצעים לביצועה היה ידע בבינה מלאכותית ותכנות בPython, את העבודה על הבינות החלטנו לבצע בשפת Python כיוון שקיימות ספריות רבות לכך בשפה זו, מעבר הידע על שפת התכנות שהיה עלינו לדעת, היה עלינו לדעת כיצד עובדת בינה מלאכותית, הדבר כולל את הבנייה, תהליך האימון, תהליך השימוש, תהליך הכנת הנתונים ועוד הרבה אחרים, בנוסף לכך היה עלינו לדעת כיצד להשתמש בספריות רבות אחרות בשפת Python בשביל שנוכל לממש את הפרויקט.

4. מסמך ייזום, אפיון, ארכיטקטורה, תוכנית עבודה ועיצובי ספרינטים

יזום

תיאור כללי

כלי אשר מוצב במרחב מסוים במטרה לאתר אנשים אשר לא עוטים מסיכה וקורא להם לעטות מסכה.

מטרת הפרויקט

מטרת הפרויקט היא אכיפה מוגברת של עטיית מסכות במקומות ציבוריים מבלי להשתמש בעוד כוח אדם.
את הרעיון קיבלנו לאחר שפירקנו את רעיון הפרויקט המקורי לפרויקט קטן יותר שהיה זיהוי אנשים אשר אינם עוטים מסיכה ועל גביו הרחבנו את הרעיון ואת הפיצ'רים.

ליבה טכנולוגית

הליבה הטכנולוגית של הפרויקט היא זיהוי אנשים אשר לא עוטים מסיכה בתמונה באמצעות עיבוד תמונה.

טכנולוגיות עיקריות ושיקולים עיקריים

הטכנולוגיות אותן נצטרך ללמוד הן עיבוד תמונה, יהיה עלינו ללמוד את טכנולוגיה זו כיוון שעל זה מתבסס כל הרעיון של זיהוי אנשים אשר אינם עוטים מסכה, לא הייתה לנו התלבטות בדרך, יהיה עלינו ללמוד את הטכנולוגיה עד לנקודה בה נוכל לבצע להבין בצורה יחסית מעמיקה מדוע מה שאנחנו עושים עובד בדרך בה הוא עובד ואם אינו עובד שנוכל לדעת מדוע.

אתגרים טכנולוגיים ומקורות

האתגרים בהם אנו עלולים להיתקל הם מציאת datasets נקיים וטובים שאיתם נוכל לעבוד ולמידת הנושא של בינה מלאכותית, את העזרה אנו נוכל לקבל דרך חיפוש מעמיק באינטרנט.
דוגמאות לאתרים שכנראה נשתמש בהם:

stackoverflow
geeksforgeeks
github
youtube

סוגי משתמשים / קהל היעד

הכלי נועד לגורמים אוכפי החוק כדי להקל עליהם על אכיפת עטיית מסכות בקרב תושבי המדינה.
לא נדרש ידע מקדים כלשהו אצל המשתמש.
למערכת אין מספר סוגי משתמשים יש סוג אחד.

דרישות חומרה

לפרויקט יש דרישות חומרה, הדרישות הבסיסיות ביותר הן מצלמה וכוח עיבוד מספיק בשביל להריץ את המודלים בזמן אמת (המחשב הנוכחי שלנו אמור להספיק), מעבר לכך, כרטיס מסך איתו ניתן לאמן מודלים יכול לעזור מאוד.

פתרונות קיימים

למיטב ידיעתינו אין מוצרים אשר עוזרים באכיפת עטיית מסכות, אולם יש מערכות אשר מזהות עטיית מסכה בדומה למערכת אותה אנו מתכננים ליצור.
להלן סרטונים המתארים פרויקט דומה:

https://www.youtube.com/watch?v=FagQhPkrrws
https://www.youtube.com/watch?v=8XM-r8ChTXM

אפיון

פיצ'רים ותהליכים עיקריים

צילום תמונה - צילום תמונה במצלמה המחוברת למכשיר.

מציאת פרצופים בתמונה - המערכת תצטרך לעבד כל כמה שניות את התמונה המתקבלת מהמצלמה במטרה למצוא פרצופים בתמונה. זאת ניתן לבצע בקלות באמצעות הספריה OpenCV שבה קיים מודל מאומן מראש, שניתן להשתמש בו לביצוע זיהוי פרצופים בתמונה. בתחילת הפרויקט נרצה להשתמש במודל מאומן אבל במהלך הפרויקט נרצה לכתוב ולאמן מודל בעצמנו שיעשה זאת.
לאחר מציאת פרצופים בתמונה נפצל את התמונה לפרצופים בודדים ואותם נעביר למודלים הבאים בשביל לחלץ פיצ'רים לגבי הפרצופים שנמצאו.

זיהוי עטיית מסכה על פרצוף שזוהה- עבור כל פרצוף שנמצא בתמונה על המערכת לבדוק האם הוא עוטה מסכה או לא. זאת נעשה באמצעות מודל שנכתוב ונאמן בעצמנו, יהיה עלינו להשיג בסיס נתונים המכיל פרצופים עם מסכה ופרצופים בלי מסכה.
את הבינה המלאכותית שתבצעה את הבדיקה האם אדם עוטה מסכה נכתוב ונאמן באמצעות ספריית TensorFlow.

זיהוי מגדרו של פרצוף שזוהה - עבור כל פרצוף שנמצא בתמונה על המערכת לבדוק את מין האדם. זאת נעשה באמצעות מודל שנכתוב ונאמן בעצמנו, יהיה עלינו להשיג בסיס נתונים המכיל פרצופים של נשים וגברים.
את הבינה המלאכותית שתבצעה את הבדיקה האם אדם הוא בן או בת נכתוב ונאמן באמצעות ספריית TensorFlow.

בדיקה האם לפרצוף שנמצא יש זקן - עבור כל פרצוף שנמצא בתמונה על המערכת לבדוק האם יש לו זקן. זאת נעשה באמצעות מודל שנכתוב ונאמן בעצמנו, יהיה עלינו להשיג בסיס נתונים המכיל פרצופים עם זקן ובלי זקן.
את הבינה המלאכותית שתבצעה את הבדיקה האם לאדם יש זקן נכתוב ונאמן באמצעות ספריית TensorFlow.

בדיקת משקפיים עבור פרצוף - עבור כל פרצוף שנמצא בתמונה על המערכת לבדוק האם יש לו משקפי ראיה, משקפי שמש או שאינו לובש משקפיים כלל. זאת נעשה באמצעות מודל שנכתוב ונאמן בעצמנו, יהיה עלינו להשיג בסיס נתונים המכיל פרצופים בלי משקפיים, עם משקפי ראיה ומשקפי שמש.
את הבינה המלאכותית שתבצעה את הבדיקה נכתוב ונאמן באמצעות ספריית TensorFlow.

שמירת פרצוף אשר לא עטה מסכה - שמירת התמונה של הפרצוף אשר נמצא לא עוטה מסכה על המחשב.

פנייה לאדם אשר לא עטה מסכה- פנייה לאדם שנמצא לא עוטה מסכה לפי מינו וצבע חולצתו בבקשה לעטות מסכה. זאת ניתן לעשות באמצעות ספרית OpenCV: כדי לזהות צבעים בתמונות, הדבר הראשון שעליך לעשות הוא להגדיר את הגבולות העליונים והתחתונים לערכי הפיקסלים שלך.

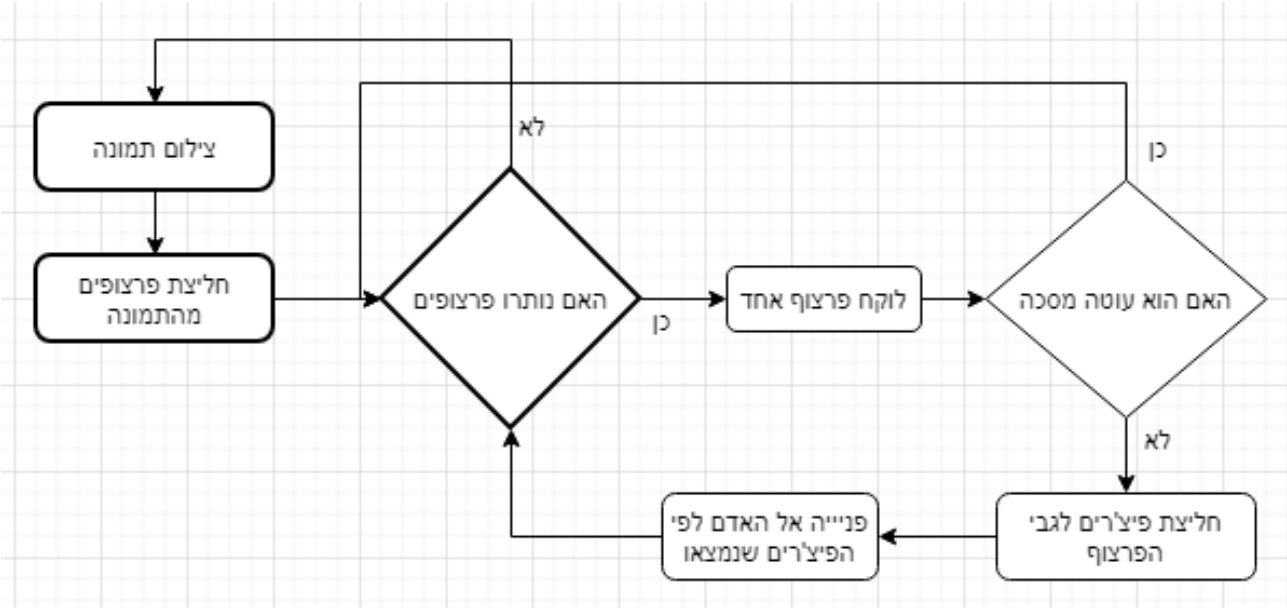
<u>ספירת כמות הנוכחים בחדר</u> - ספירה של כמות האנשים שנמצאו בפריים שצולם, המטרה של הפיצ'ר היא לבדוק שאין מעבר על הגבלת הנוכחים בסביבה מסוימת, במידה ויש חריגה, הקוד ישמור את התמונה הנוכחית.

טכנולוגיות

פיצ'ר/תהליך	טכנולוגיות ושפות תכנות	משאבים נדרשים ושירותים חיצוניים
צילום תמונה	נכתב בשפת python, נשתמש בספריה OPENCV עבור קבלת התמונה מהמצלמה	מצלמה
מציאת פרצופים בתמונה	נכתב בשפת python, נשתמש בספריה OPENCV למציאת הפרצופים בתמונה, בהמשך יעבור להיות מודל שנכתוב ונאמן לבד	מצלמה, בהמשך יהיה עלינו להשיג dataset מתאים לאימון מודל מסוג detection
זיהוי עטיית מסכה של פרצוף שזוהה	נכתב בשפת python, נשתמש במערכת נוירונים בשביל לקבוע האם האדם עוטה מסכה	מצלמה, dataset של אנשים אשר עוטים מסכה ואנשים שלא עוטים מסכה
זיהוי מגדרו של פרצוף שזוהה	נכתב בשפת python, נשתמש במערכת נוירונים בשביל לקבוע את מגדרו של האדם	מצלמה, dataset של פרצופים של נשים וגברים בשביל לאמן את רשת הנוירונים
בדיקה האם לפרצוף שנמצא יש זקן	נכתב בשפת python, נשתמש במערכת נוירונים בשביל לקבוע האם לאדם יש זקן	מצלמה, dataset של פרצופים עם זקן ופרצופים בלי זקן בשביל לאמן את רשת הנוירונים
בדיקת משקפיים עבור פרצוף	נכתב בשפת python, נשתמש במערכת נוירונים בשביל לקבוע את סוג המשקפיים (ראיה\שמש\בלי)	מצלמה, dataset של פרצופים בלי משקפיים כלל, פרצופים עם משקפי ראיה ופרצופים עם משקפי שמש בשביל לאמן את רשת הנוירונים
שמירת פרצוף אשר לא עטה מסכה	נכתב בשפת python, התמונות ישמרו על המכשיר עליו המערכת תרוץ	זיכרון נגיש
פנייה לאדם אשר לא עטה מסכה	נכתב בשפת python, פעולה זו תשמיע מספר הקלטות שייצרו משפט אשר פונה לאדם שאינו עוטה מסכה	רמקול, מספר הקלטות בשילובים שונים אשר יצרו משפט הגיוני ונכון לסיטואציה
ספירת כמות הנוכחים בחדר	נכתב בשפת python, יבצע	מצלמה

	ספירה של מספר הפרצופים שנמצאו בפריים	

תרשים זרימה



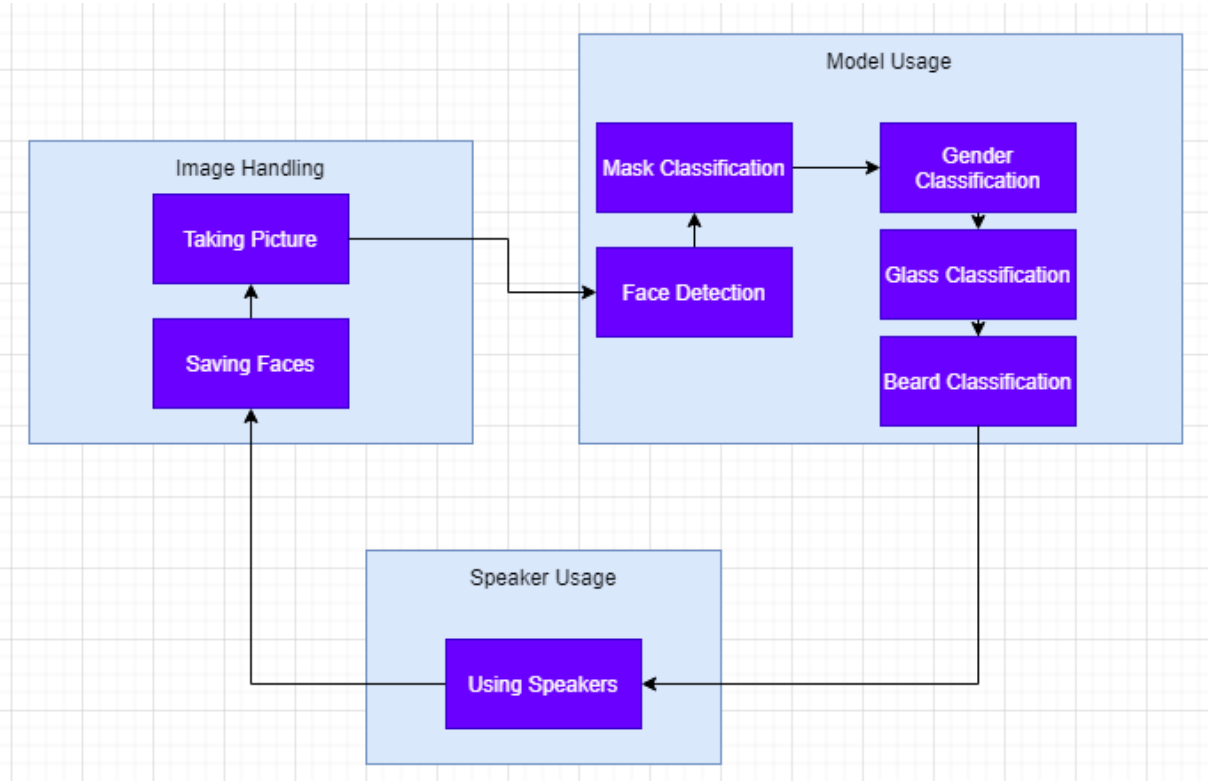
מבנה בסיס נתונים

בפרויקט יהיו חמישה מאגרי מידע, ארבעה מתוך החמישה מתפרקים למאגרים בסיסיים יותר:

- מאגר ראשון - מסכה, המאגר יכיל שתי קבוצות של תמונות, תמונות של פרצופים עם מסכה ותמונות של פרצופים בלי מסכה.
- מאגר שני - מגדר, המאגר יכיל שתי קבוצות של תמונות, תמונות של פרצופים של גברים ותמונות של פרצופים של נשים.
- מאגר שלישי - זקן, המאגר יכיל שתי קבוצות של תמונות, תמונות של פרצופים עם זקן ותמונות של פרצופים בלי זקן.
- מאגר רביעי - משקפיים, המאגר יכיל שלוש קבוצות של תמונות, תמונות של פרצופים עם משקפי ראיה, תמונות של פרצופים עם משקפי שמש ותמונות של פרצופים ללא משקפיים.
- מאגר חמישי - זיהוי פרצופים, המאגר יכיל תמונות עם מיקומים של פרצופים בתוך כל תמונה.

ארכיטקטורה

מבט על



- Image Handling -
  - הרכיב יהיה אחראי על שמירת התמונות שימוש במצלמה.
  - Saving Faces - החלק האחראי על שמירת התמונות.
  - Taking Picture - החלק שלוקח שלוקח תמונה.

- Model Usage -
  - הרכיב אחראי על השימוש במודלים השונים בהם נשתמש בפרויקט.
  - Face Detection - ימצא אנשים בתוך תמונה ויחלק את התמונה לאנשים שונים.
  - Mask Classification - יבדוק מבין האנשים שנמצאו בתמונה מי אינו עוטה מסכה.
  - Gender Classification - יזהה מתוך האנשים אשר אינם עוטה מסכה את מינם באמצעות בינה.
  - Glass Classification - יבדוק סוג משקפיים עבור אנשים שלא עוטים מסכה.
  - Beard Classification - יבדוק האם לאדם יש זקן עבור כל אדם שנמצא לא עוטה מסכה.

- Speakers Usage -

- הרכיב יקבל מידע על האנשים אשר אינם עוטים מסכה וישתמש ברמקולים בשביל לבקש מהם לעטות מסכה.
- Using Speakers - יקבל את המידע על האדם אשר לא עטה מסכה וינגן הקלטה בהתאם למידע שנמצא עליו.

עיצוב נתונים ויישויות מידע

בפרויקט זה יבוצע שימוש בישויות מידע שנכתוב בעצמנו, אנחנו צריכים ישות שתייצג אדם, היא תכיל מידע לגביו, מיקום תחילת פניו בתמונה המקורית, מיקום סיום פניו בתמונה המקורית, מה מינו, האם יש לו משקפי שמש, משקפיים רגילים או כלל אין לו משקפיים והאם יש לו זקן, ליישות זו נקרא Person. כעת עלינו להוסיף עוד ישות שתייצג את כל האנשים שנמצאו בתמונה, היא תצטרך להכיל מערך שמכיל את פניהם של כל מי שנמצא בתמונה ומערך של מידע לגבי כל אחד מהאנשים הללו, ליישות זו קראנו People.

טכנולוגיות עיקריות

כל הקוד יכתב בשפת Python בגלל במגוון הרחב של ספריות ונוחות התכנות בשפה.
בפרויקט יהיה עלינו להשתמש בטכנולוגיות והכלים הבאים:

- יהיה עלינו להשתמש בטכנולוגיית הבינה המלאכותית, אם לדייק נשתמש בטכנולוגיית רשת הנוירונים, נשתמש בטכנולוגיה זאת לכתיבת ואימון כל המודלים שנממש בעצמנו, נשתמש בספרייה הקיימת Tensorflow בשביל לממש את השימוש. בחרנו להשתמש בספרייה זו מכיוון שיש לה הרבה מדריכים באינטרנט ויהיה לנו יחסית קל לקבל עזרה במידה ונתקל בבעיה, בנוסף לכך, אחרי המחקר הראשוני היה נראה כי זוהי הספרייה בה משתמשים לרוב בשוק.
- יהיה עלינו להשתמש במספר כלים שממומשים בספריה Opencv, יהיה עלינו ללמוד איך להשתמש במודל הקיים שלהם למציאת פנים בתמונה, כיצד לצלם תמונה וכיצד לשנות מימדים של תמונה.
- בשביל לאמן את המודלים יש דרישות חומרה, העיקרית היא גישה לכרטיס מסך שאפשר לאמן איתו בינות, כיוון שנכון לעכשיו אין לנו אחד כזה, יהיה עלינו להשתמש בשירותי google collaboratory, בחרנו בכלי זה בגלל שהוא של גוגל, בדיוק כמו ספריית Tensorflow ככה שאנחנו מאמינים שיהיה יותר פשוט להשתמש בהם ביחד מאשר אם נשתמש בשירות אחר לאימון המודלים.

התאמה לאפיון

פיצ'ר	רכיבים רלוונטים
צילום תמונה	Image Handling
מציאת פרצופים בתמונה	Model Usage
זיהוי עטיית מסכה על פרצוף שזוהה	Model Usage
זיהוי מגדרו של פרצוף שזוהה	Model Usage
בדיקה האם לפרצוף שנמצא יש זקן	Model Usage
בדיקת משקפיים עבור פרצוף	Model Usage
שמירת פרצוף אשר לא עטה מסכה	Image Handling
פנייה לאדם אשר לא עטה מסכה	Speaker Usage
ספירת כמות הנוכחים בחדר	Model Usage

תוכנית עבודה

1. צילום תמונה
2. מציאת פרצופים בתמונה
3. זיהוי עטיית מסכה על פרצוף שזוהה
4. זיהוי מגדרו של פרצוף שזוהה
5. בדיקה האם לפרצוף שנמצא יש זקן
6. בדיקת משקפיים עבור פרצוף
7. שמירת פרצוף אשר לא עטה מסכה
8. פנייה לאדם אשר לא עטה מסכה
9. ספירת כמות הנוכחים בחדר

מס' פיצ'ר	משימה	תלויות תשתית (האם המשימה תלויה במשימה אחרת)	נימוק והערות	רכיב רלוונטי
1	מימוש בקוד	אין	מימוש צילום התמונה בקוד	Image Handling
2	העלאת קבצי המודל לקוד	אין	העלאה של קבצי המשקל וקבצי המודל לקוד לקראת שימוש במודל	Model Usage
2	מציאת אדם בתמונה מתוך שימוש במודל	יש תלות במשימה הראשונה של הפיצ'ר	שימוש במודל שהעלנו למציאת אנשים בתמונה	Model Usage
3	השגת dataset	אין	מציאת dataset באינטרנט של אנשים אשר עוטים מסכה ושל אנשים אשר אינם עוטים מסכה	Model Usage
3	כתיבת ואימון המודל	ישנה תלות במשימת השגת ה-dataset	כתיבה ואימון של המודל והכנתו לשימוש בקוד	Model Usage
3	השמת המודל בקוד	ישנה תלות במשימות הקודמות של הפיצ'ר	שימוש במודל שבנינו בקוד	Model Usage
4	השגת dataset	אין	מציאת dataset באינטרנט של תמונות נשים וגברים	Model Usage
4	כתיבת ואימון המודל	ישנה תלות במשימת השגת ה-dataset	שימוש במודל שבנינו בקוד	Model Usage
4	השמת המודל בקוד	ישנה תלות במשימות הקודמות של הפיצ'ר	שימוש במודל שבנינו בקוד	Model Usage
5	השגת dataset	אין	מציאת dataset באינטרנט של פרצופים עם ובלי זקן	Model Usage

5	כתיבת ואימון המודל	ישנה תלות במשימת השגת ה-dataset	שימוש במודל שבנינו בקוד	Model Usage
5	השמת המודל בקוד	ישנה תלות במשימות הקודמות של הפיצ'ר	שימוש במודל שבנינו בקוד	Model Usage
6	השגת dataset	אין	מציאת dataset באינטרנט של פרצופים עם משקפי ראייה, משקפי שמש ובלי משקפיים	Model Usage
6	כתיבת ואימון המודל	ישנה תלות במשימת השגת ה-dataset	שימוש במודל שבנינו בקוד	Model Usage
6	השמת המודל בקוד	ישנה תלות במשימות הקודמות של הפיצ'ר	שימוש במודל שבנינו בקוד	Model Usage
7	מימוש בקוד	אין	מימוש של שמירת פרצוף בקוד	Image Handling
8	בניית משפט לפנייה אל הבן אדם ופנייה אל האדם	קיימת תלות בתוצאות המודלים הקודמים	בניית משפט שהמחשב יגיד כדי שאנשים יעטו מסכה	Speaker Usage
9	ספירת הנוכחים בחדר	קיימת תלות במודל שמוצא אנשים בתמונה	ספירת מספר האנשים השוהים בחדר על מנת לוודא שלא נמצאת חריגה מההתרות למספר האנשים בחדר	Image processing
3	העלאת ה-Accuracy	ישנה תלות במודל הקיים	שיפור המודל הקיים ואף כתיבה ואימונים מחודשים של המודל	Model Usage
4	העלאת ה-Accuracy	ישנה תלות במודל הקיים	שיפור המודל הקיים ואף כתיבה ואימונים מחודשים של המודל	Model Usage
2	השגת dataset	אין	מציאת dataset, איננו בטוחים עדיין בעזרת איזה מידע עלינו לאמן את המודל	Model Usage
2	למידה נוספת על סוג המודל	אין	ללמוד יותר לעומק על סוג המודל שעלינו ליצור	Model Usage
2	כתיבת ואימון המודל	ישנה תלות במשימת השגת ה-dataset	כתיבה ואימון של המודל שמוצא אנשים בתמונה	Model Usage
2	השמת המודל בקוד	ישנה תלות במשימת הכתיבה והאימון של המודל	השמת המודל שיצרנו בקוד הכתוב במקום המודל שהשתמשנו בו עד עכשיו	Model Usage

חלוקה לאיטרציות (ספרינטים)

<u>איטרציה 1</u>

- מימוש בקוד (פיצ'ר 1)
- העלאת קבצי מודל קיים לקוד (פיצ'ר 2)
- מציאת אדם בתמונה מתוך שימוש במודל (פיצ'ר 2)
- השגת dataset (פיצ'ר 3)
- כתיבת ואימון המודל (פיצ'ר 3)
- השמת המודל בקוד (פיצ'ר 3)
- השגת dataset (פיצ'ר 4)
- כתיבת ואימון המודל (פיצ'ר 4)
- השמת המודל בקוד (פיצ'ר 4)

<u>איטרציה 2</u>

- למידה נוספת על סוג המודל (פיצ'ר 2)
- השגת dataset (פיצ'ר 2)
- בניית המשפט לפנייה אל הבן אדם ופנייה אל האדם (פיצ'ר 8)
- ספירת הנוכחים בחדר (פיצ'ר 9)

<u>איטרציה 3</u>

- כתיבת ואימון המודל (פיצ'ר 2)
- השמת המודל בקוד (פיצ'ר 2)
- העלאת ה-Accuracy (פיצ'ר 3)
- העלאת ה-Accuracy (פיצ'ר 4)

<u>איטרציה 4</u>

- השגת dataset (פיצ'ר 6)
- כתיבת ואימון המודל (פיצ'ר 6)
- השמת המודל בקוד (פיצ'ר 6)
- השגת dataset (פיצ'ר 5)
- כתיבת ואימון המודל (פיצ'ר 5)
- השמת המודל בקוד (פיצ'ר 5)

<u>איטרציה 5</u>

- השלמות, במידת צורך
- הוספת פיצ'רים חדשים (במידה ויש)

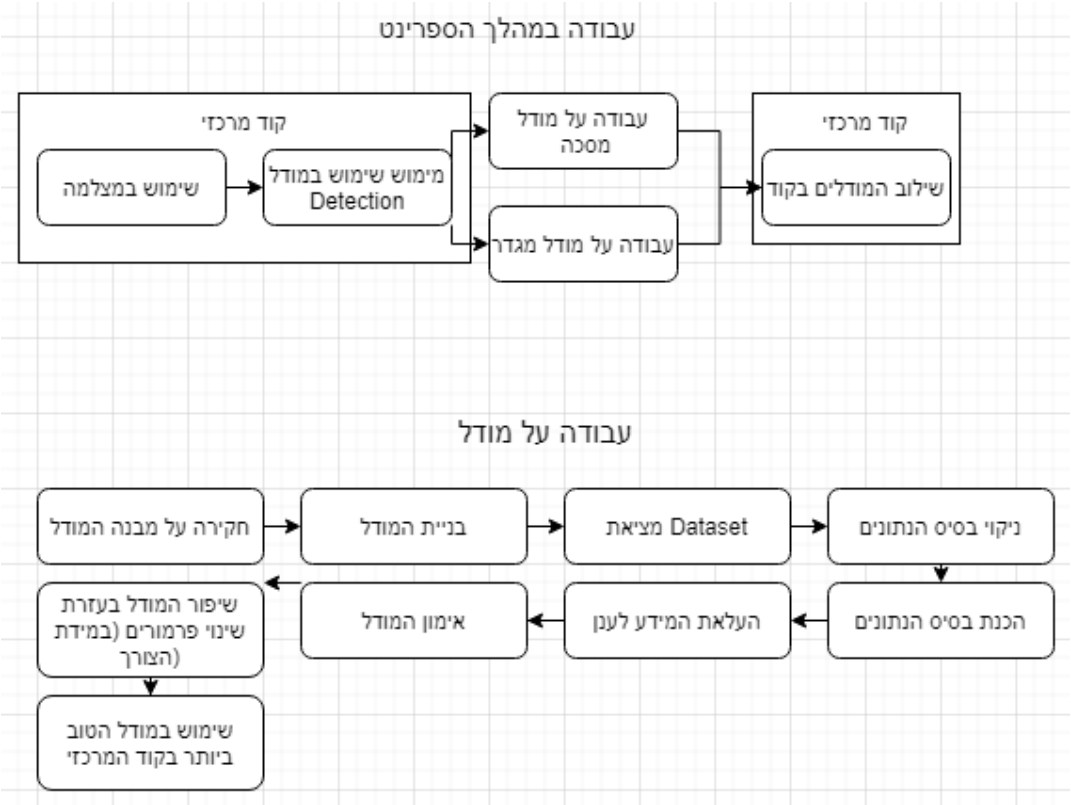
מסמכי עיצוב ספרינט

איטרציה 1 - כתיבת קוד שמשתמש בזיהוי פנים, זיהוי עטיית מסכה וזיהוי מין האדם

החזון שלנו לסיום הספרינט

בסיום הספרינט הנ"ל, ירוץ קוד שיציג על מסך סרטון שצולם בלייב, בסרטון יוקפו האנשים אשר לא עוטים מסכה עם טקסט המציג את מינם ליד ראשם, בנוסף נרצה להוסיף את האחוזים בהם הבינות משערות את הניחוש שלהן ליד ראשם של האנשים שנמצאו לא עוטים מסכה.

תרשים זרימה המתאר את תהליך העבודה בספרינט:



תקציר, הסבר ותוצר

הספרינט הנ"ל הוא הספרינט העמוס ביותר מבין כל הספרינטים עקב הרצון להשאיר זמן להוספת פיצ'רים נוספים במידה ונרצה ו-debugging.

את הספרינט חילקנו לשני שלבים כאשר בשלב הראשון תהיה עבודה במקביל של שנינו, כל אחד על בינה אחרת. את העבודה במקביל בשלב הראשון אנו נרצה לסיים בערך באותו הזמן בשביל שנוכל לעבור לשלב השני בו יהיה על שנינו לעבוד יחד כדי לשלב את התוצרים של כל אחד מהתוצרים של השלב הראשון. את הבינות נכתוב בעזרת שימוש בספרייה Tensorflow.

תחילה, יבוצע צילום של תמונה. לאחר מכן, את התמונה נעביר לבינה אשר מוצאת אנשים בתמונה. את האנשים נעביר לבינה אשר בודקת האם הם עוטים מסכה. את האנשים אשר נמצאו לא עוטים

מסכה נעביר לבינה אשר בודקת את מינם. לבסוף, נעביר את כל המידע לפונקציה אשר תקיף את האנשים האלו ותכתוב את מינם.

התוצר יציג מסך כאשר במסך נראה את אשר המצלמה מצלמת ובמסך תוצג תמונה ובתוכה מסגרות של אנשים אשר לא עוטים מסכה וליד המסגרת יהיה כתוב מינם.

הצגה ויזואלית של מטרת התוצר הסופי לקראת סיום הספרינט:

<u>תכולות טכנולוגיות</u>

בספרינט זה נשתמש בטכנולוגיות בינה מלאכותית, ועיבוד תמונה, נעשה שימוש בספריית TensorFlow ו-opencv.

את המודלים נכתוב ונאמן בעזרת ספריית TensorFlow.

את המודל הקיים של זיהוי אנשים בתמונה נממש באמצעות opencv אשר מכיל בתוכו מודל זה.

איטרציה 2 - כתיבת קוד אשר בונה משפט, סופר נוכחים בחדר ומחקר על face detection

<u>החזון שלנו לסיום הספרינט:</u>

בסיום הספרינט הנ"ל, אנו רוצים להגיע למצב בו הקוד בונה משפט בסיסי, באמצעותו נפנה לאנשים אשר לא עוטים מסכה בתמונה, נמצא dataset נקי (אם נצטרך ננקה) בשביל הכתיבה של המודל של פיצ'ר 2.

<u>תקציר, הסבר ותוצר:</u>

ספרינט זה מתמקד פחות באימונים וכתיבה של מודלים נוספים. הספרינט מתמקד בשימוש הפיצ'רים שנמצאו במודלים הקודמים.

במהלך הספרינט נעבוד שנינו במקביל, כל אחד על המשימות שלו. החלק של אופק יעסוק בעיקר בקוד המרכזי של הפרויקט בעוד החלק של בן יעסוק במחקר והכנות להמשך הפרויקט. במהלך העבודה בן יעזור לאופק במשימות שהוטלו עליו במידה ויהיה צורך בכך.

תחילה, נחלץ מידע על האנשים שנמצאו לא עוטים מסכה (רק מין כרגע), לאחר מכן נשתמש במידע שמצאנו על האדם ונפנה אליו באמצעות השמעת קובץ קול.

התוצר יהיה התוצר הסופי של הספרינט הקודם בתוספת בדיקת כמות הנוכחים בחדר ופנייה לאנשים שנמצאו לא עוטים מסכה.

<u>תכולות טכנולוגיות:</u>

בספרינט זה נחקור עוד סוגים של מודלים בתחום הבינה המלאכותית, נלמד שימוש בספריות קול בפייתון.

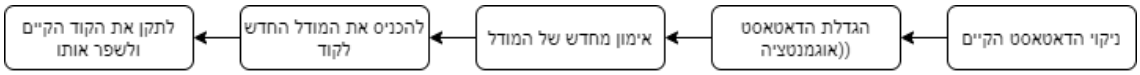
איטרציה 3 - בניית מודל למציאת אנשים ושיפור מודלים קיימים

<u>החזון שלנו לסיום הספרינט:</u>
מודל עובד ומיושם בקוד איתו נמצא אנשים בתמונה.
שיפור המודלים הקודמים שיהיו יותר טובים.

<u>תקציר, הסבר ותוצר:</u>
ספרינט זה נתמקד בעיקר בשיפור הדאטהסטים הקיימים בהם אנחנו משתמשים, בספרינט זה בעיקר ננקה ונרחיב את מאגרי המידע בהם אנו משתמשים ונאמן יותר את המודלים הקיימים בשביל להגיע לרמות דיוק גבוהות יותר, בנוסף לזה, במהלך הספרינט בשביל לשפר את הדיוק של המודלים נתנסה עם ארכיטקטורות שונות של מודלים.

במהלך הספרינט נעבוד במקביל, כל אחד ינקה דאטה סט שונה, כל אחד לאחר מכן יאמן מודל אחר.

התוצר הסופי יהיה די זהה לזה של הספרינט הקודם, רק עם מודלים טובים יותר.



<u>תכולות טכנולוגיות:</u>
בספרינט זה נשתמש בידע שיש לנו כבר, לא יהיה שימוש רב במידע חדש, נשתמש בספריות Tensorflow ו-numpy.
בספרינט זה יתבצע לראשונה שימוש בגוגל קולב, נשתמש בזה כאן בפעם הראשונה בשביל אימון המודל של זיהוי אנשים בתמונה כיוון שעל מחשב ביתי פשוט התהליך יקח מספר רב של ימים לעומת שעות בודדות על שרתי גוגל.

איטרציה 4 - בניית מודלים נוספים

<u>החזון שלנו לסיום הספרינט:</u>

החזון שלנו לספרינט זה הוא הוספת מודלים נוספים למציאת עוד פיצ'רים עבור כל אדם, נכון לעכשיו החלטנו לוותר על מודל ה face detection שלנו ונשתמש במודל שעד עכשיו השתמשנו בו.

<u>תקציר, הסבר ותוצר:</u>

בספרינט זה המיקוד יהיה יצירת ושימוש בפיצ'רים חדשים, בניגוד לספרינט הקודם, שעסק בעיקר בשיפור פיצ'רים קיימים. במהלך הספרינט נשיג datasets חדשים, ניצור נאמן ונתעד מודלים חדשים אותם נאמן בשביל הפיצ'רים שתוכננו.

במהלך הספרינט נעבוד במקביל, כל אחד על פיצ'ר אחר, כל אחד יהיה אחראי על העבודה על מודל אחר וכל מה שנלווה לו.

התוצר הסופי יהיה פיצ'רים נוספים לתוצר הקודם, יבוצע שימוש בפיצ'רים הנוספים כדי למצוא עוד מידע על אנשים אשר נמצאו לא עוטים מסכה.

<u>תכולות טכנולוגיות:</u>

בספרינט זה נשתמש בידע שיש לנו כבר, יהיה עוד מחקר על ארכיטקטורות שונות של מודלים, נשתמש בספריות Tensorflow ו-numpy.
בספרינט זה יתבצע שימוש בגוגל קולב, נשתמש בזה כיון שעל מחשב ביתי פשוט התהליך של אימון המודלים יקח מספר רב של ימים לעומת שעות בודדות על שרתי גוגל. בספרינט הקודם נתקלנו במספר רב של בעיות בעת השימוש בגוגל קולב, למרות אותן תקלות, הגענו למסקנה שהמשך השימוש יהיה הצעד הנבון ביותר.

איטרציה 5 - ייעול קוד ושיפור זמן ריצה

<u>החזון שלנו לסיום הספרינט:</u>

החזון שלנו לספרינט זה הוא להנמיך את זמני החישוב המתבצעים בריצה, להעלות את מהירות החישוב של המודלים והפיכת התוכנה למהירה יותר, אנחנו רוצים שיהיו שלושה ענפים, אחד בו הקוד המקורי, אחד בו השתמשנו ב TensorRT לייעול המודלים ואחד בו השתמשנו בTensorflow Lite.

<u>תקציר, הסבר ותוצר:</u>

בספרינט זה נעבוד על שיפור הקוד הקיים ושיפור זמני החישוב של התוכנית, בניגוד לספרינטים הקודמים, בהם עבדנו על מודלים נוספים וכתיבת קודים המשתמשים בהם הפעם נביא את המודלים לצורה מוגמרת שירוצו בצורה מהירה בעזרת כלים.

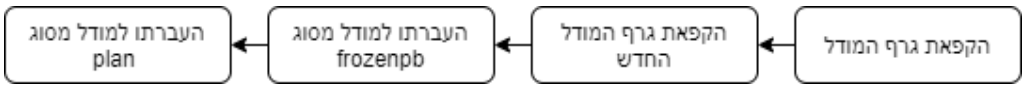
במהלך הספרינט נעבוד ביחד, כל אחד יחקור על טכנולוגיה אחת וישתף את השני במידע שהשיג, לאחר ששנינו נחקור נעבוד ביחד להבאת המודלים לצורת המוגמרת.

התוצר הסופי יהיה זהה לתוצר הסופי של הספרינט הוקדם רק שהוא יעבוד מהר יותר וביעילות רבה יותר, יהיו מספר תוצרים כפי שכבר ציינתי.

<u>תכולות טכנולוגיות:</u>

בספרינט זה יהיה עלינו לחקור על טכנולוגיות חדשות, הטכנולוגיות יהיו TensorRT ו-Tensorflow Lite נצטרך ללמוד ולהבין איך להשתמש בטכנולוגיות (אם בכלל) כדי להפוך את המודלים למהירים יותר.

להלן תרשים המתאר בפשטות את תהליך הטרנספורמציה של המודלים לאופטימיזציה מסוג TensorRT:



5. הוראות שימוש – התקנה והפעלה

1. הוראות שימוש

בפרויקט זה יש שני קבצים ראשיים, אחד עם מסך אשר מציג מה שהמצלמה מצלמת ומידע בנוגע לאנשים שנמצאו בתמונה, השני הוא ללא מסך, השני משתמש ברמקולים בשביל לפנות לאנשים שנמצאו לא עוטים מסכה.
בשביל להתחיל את הפרויקט על המשתמש לפתוח את חלון ה cmd בתיקיית הפרויקט, לאחר מכן עליו לבחור איזה מן הקבצים הוא רוצה להריץ, בשביל להריץ את הקוד עם החלון עליו לכתוב:

cd tf-lite

בשביל להריץ את הקובץ שמשתמש ברמקולים עליו לכתוב:

cd final

לאחר מכן (לא משנה איזה מן הקבצים הוא רוצה להריץ) עליו לכתוב:

python main.py

קישור להורדת קבצי התוצר הסופי
קישור לפרויקט ב-GitLab

2. בסיס נתונים

בפרויקט לא קיים בסיס נתונים סטנדרטי. הקובץ הראשי המשתמש ברמקולים (זה שנמצא בתיקייה speakers) הוא היחיד ששומר מידע מחוץ לקוד. אנחנו שומרים תמונות וקבצי קול בפרויקט אבל לא שומרים אותם בבסיס נתונים. את התמונות אנחנו שומרים בצורה כזאת שהתמונות של אנשים שנמצאו ללא מסכה נשמרים בתיקייה מסוימת כאשר שם כל תמונה מציין את התאריך והזמן שבו היא נשמרה, תמונות של מקומות שיש בהם יותר מדי אנשים נשמרים בתיקייה נפרדת באותה צורה, השם מייצג את התאריך והזמן בו התמונה נשמרה. את קבצי הקול אנחנו שומרים בתיקייה נוספת עם שמות קבועים כיוון שהקבצים האלו לא משתנים בזמן ריצה.
תיקייה בה אנו שומרים פרצופים של אנשים שלא עטו מסכה:

`speakers\no_mask`

תיקייה בה אנחנו שומרים תמונות שיש בהם יותר אנשים ממה שהוגדר כחוקי:

`speakers\over_limit`

תיקייה בה נשמרו קבצי הקול:

`speakers\voices`

3. מדריך התקנה

תחילה על המשתמש להתקין python 3.8.8 מכאן, בתחתית העמוד נמצאים אפשרויות ההתקנה, על המשתמש לבחור את ההתקנה המתאימה למחשב שלו.
לאחר שהתקין python 3.8.8 עליו להתקין pip, ניתן לעשות זאת על ידי פתיחת חלון ה cmd ולכתוב: `python get-pip.py`
לאחר ההתקנה של שני התוכנות האלו על המשתמש להתקין את הספריות שבהן הפרויקט משתמש. כעת על משתמש לפתוח את תיקיית הפרויקט, לפתוח בתוך התיקייה את חלון ה cmd ולכתוב: `pip install -r requirements.txt` כל הספריות יותקנו והמשתמש יוכל להריץ את קוד הפרויקט.

4. התאמת סביבת העבודה להרצה - קונפיגורציה

אין הכנות חובה שעל המשתמש לבצע לפני הרצת התוכנית, יש אפשרות לשנות את מספר האנשים שחוקי שיהיו בפריים. במידה והמשתמש רוצה לשנות את פרמטר זה עליו לפתוח על הקובץ speakers\main.py עם עורך טקסט ולשנות את המספר המופיע בשורה 8 למספר ההגבלה הרצוי.

5. מדריך האפליקציה למנהל

בפרויקט זה לא קיים מנהל, כל המשתמשים הם בעלי אותה הרשאה.

6. מדריך למפתח

סוגי קבצים עיקריים:

- `.py` קובץ המכיל קוד פייתון.
- `.ipynb` קובץ 'מחברת' פייתון, קובץ זה מכיל קוד פייתון בפורמט של מחברת, ניתן לפתוח את הקבצים האלו בעזרת jupyter או google colab.
- `.h5` קבצים המכילים מודלים של Tensorflow.
- `.tflite` קבצים המכילים מודלים של Tensorflow Lite.

הסבר על כל קובץ:

בפרויקט יש קבצים שנמצאים מספר פעמים במיקומים שונים, אתחיל מהסברים על קבצים אלו.

`uti.py`

הקובץ מכיל מחלקות אשר משומשות לארגון מידע לגבי אנשים שנמצאו בתמונה.

מיקומים:

- Camerona/keras/uti.py
- Camerona/speakers/uti.py
- Camerona/tf-lite/uti.py

תוכן:

```
from tensorflow.keras import models
import tensorflow as tf
import numpy as np
import cv2
from time import time, sleep
import numpy as np

class People():
      """
      This class represents the People found, contains faces and other data
     """
      def __init__(self):
            self.faces = []
            self.people = []
```

```
class Person():
    """
    This class represents a single person found, contains data about it
    """
    def __init__(self, startPoint, endPoint):
        self.start_pnt = startPoint #start point of his face in the picture
        self.end_pnt = endPoint #end point of his face in the picture
        self.gender = False
        self.glass = False
        self.sunglass = False
        self.beard = False
```

הסבר:

שם מחלקה: Person

תפקיד: ארגון יעיל ונוח של מידע לגבי אדם.

משתנים:

- start_pnt שיעורי נקודת ההתחלה של האדם בתמונה.
- end_pnt שיעורי נקודת ההתחלה של האדם בתמונה.
- gender האם גבר או אישה, אמת זה גבר שקר זה אישה.
- כל השאר זה פשוט בדיוק מה שהשם אומר.

מתודות:

- קיימת רק מתודת בנייה.

שם מחלקה: People

תפקיד: ארגון יעיל ונוח של כל המידע לגבי כל הפרצופים שנמצאו בתמונה.

משתנים:

- faces מערך המכיל את הפרצופים של האנשים בפורמט numpy.
- people מערך המכיל משתנים מסוג Person.

מתודות:

- קיימת רק מתודת בנייה.

`Coffe.py`

הקובץ מכיל מחלקה המשמשת למציאת פרצופים בתמונה.

מיקומים:

- Camerona/keras/Coffe.py
- Camerona/speakers/Coffe.py
- Camerona/tf-lite/Coffe.py
- Camerona/preperations/model training/keras/face detection/opencv/Coffe.py

- Camerona/preperations/model training/keras/face detection/training/dataset preperation/Coffe.py

תוכן:

```
from uti import *

class Coffe():
      """
      This class uses the opencv model to detect faces in a photo
      """
      def __init__(self, con_th = 0.8):
            """
            This method is the C'tor
            input: minimum confidence to determine if something is a face
            """
            tstamp = time()
            prototxtPath = "models/face-detection-weights.prototxt"
            weightsPath = "models/face-detection-model.caffemodel"
            self.net = cv2.dnn.readNet(prototxtPath, weightsPath)
            self.conf_th = con_th

      def detectFaces(self, img):
            """
            This method uses the model to detect the faces in a picture
            """
            (h, w) = img.shape[:2]

            blob = cv2.dnn.blobFromImage(img, 1.0, (300, 300), (104.0, 177.0, 123.0))

            self.net.setInput(blob)
            detections = self.net.forward()[0][0]

            toRet = People()
            for i in range(0, detections.shape[0]):#iterating through faces found in the
picture
                  confidence = detections[i][2]
                  if confidence > self.conf_th:#checking if the face exceeds the minimum
confidence
                        box = detections[i, 3:7] % np.array([1,1,1,1]) * np.array([w, h,
w, h])
                        (startX, startY, endX, endY) = box.astype("int")
                        (startX, startY) = (max(0, startX), max(0, startY))
```

```
                        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                        start_point = (int(startX),int(startY))
                        end_point = (int(endX),int(endY))
                        if end_point[0] <= start_point[0]:
                              end_point = (w, end_point[1])
                        if end_point[1] <= start_point[1]:
                              end_point = (end_point[0], h)
                        face = img[start_point[1]:end_point[1],
start_point[0]:end_point[0]]
                        face = cv2.resize(face, (64,64))/255
                        face = np.array([face], dtype=np.float32)#changing the data to
float32
                        #adding the person found to the people object
                        toRet.people.append(Person(start_point, end_point))
                        toRet.faces.append(face)
            return toRet
```

הסבר:

שם מחלקה: Coffe

תפקיד: משומש בשביל למצוא פרצופים בתמונה.

משתנים עיקריים:

- net משתנה המכיל את הבינה באמצעותה מתבצעת המציאה.
- conf_th משתנה המכיל את המינימום בשבילו מחשיבים מישהו כפרצוף.

מתודות:

- בנאי, טוען את המודל לתוכנה.
- detectFaces המתודה מקבלת תמונה ומחזירה מערך של פרצופים שנמצאו בתוך התמונה.

`convert.py`

הקובץ מכיל קובץ שממיר מודלים מTensorflow לTensorflow lite.

מיקומים:

- Camerona/preperations/model training/tf-lite/convert.py

תוכן:

```
import tensorflow as tf

#this code is used to convert .h5 file to .tflite
```

```
saved_at = "mask.h5"
save_at = "mask.tflite"

#loading the model
model = tf.keras.models.load_model(saved_at)
#converting keras to tflite model
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]#setting optimizations
tflite_model = converter.convert()
open(save_at, "wb").write(tflite_model)#saving the tflite model
```

הסבר:
משתמש בפונקציות ומחלקות קיימות בשביל להמיר את המודלים.

`classifiers.py`
הקובץ מכיל מחלקות שמשתמשות במודלים מסוג tensorflow lite בשביל לחלץ מידע מפנים של אדם.
מיקומים:

- Camerona/speakers/classifiers.py
- Camerona/tf-lite/classifiers.py
- קיים אחד נוסף אבל התוכן שלו שונה, הוא יופיע עוד מעט בנפרד

תוכן:

```
from uti import *

class ModelData():
    """
    This class uses a tf-lite model to make predictions
    """
    def __init__(self, model_path):
        """
        This method is the C'tor
        input: model path
        """
        self.model = tf.lite.Interpreter(model_path=model_path)#loading model
        self.input_tensor_index = self.model.get_input_details()[0]['index']
        self.output_tensor_index = self.model.get_output_details()[0]['index']
        self.model.allocate_tensors()#allocating memory

    def modelPredict(self, face):
```

```
        """
        This method uses the model to make predictions
        input: data to make predictions about
        output: the prediction
        """
        self.model.set_tensor(self.input_tensor_index, face)
        self.model.invoke()
        prediction = self.model.get_tensor(self.output_tensor_index)
        return prediction

class metaData():
    """
    This class uses the models to make prediction and extract data
    about the people found in the picture
    """
    def __init__(self,
        mask_path="models/mask.tflite", mask_con_th = 0.8,
        glass_path = "models/glass.tflite",
        gender_path = "models/gender.tflite",
        beard_path = "models/beard.tflite"):
        """
        This method is the C'tor, loads all the models
        input: models paths and minimum threshold for the mask classifier
        """
        #index 0 - sunglass
        #index 1 - glass
        #index 2 - nothing
        self.glass = ModelData(glass_path)
        #0 - female
        #1 - male
        self.gender = ModelData(gender_path)
        #0 - no beard
        #1 - beard
        self.beard = ModelData(beard_path)
        #0 - no mask
        #1 - mask
        self.mask = ModelData(mask_path)
        self.mask_th = mask_con_th

    def getMetaData(self, people):
        """
        This method uses the models to extract data
```

```
        input: People object, contains faces and Person object which contains data
        output: updated People object contains updated data
        """
        to_ret = People()
        j = 0
        for i in range(0, len(people.faces)):#iterating through all the faces
            face = people.faces[i]
            #checking mask
            prediction = self.mask.modelPredict(face)[0][0]#making prediction for
face
            if 1 - prediction > self.mask_th:#checking if not wearing a mask
                to_ret.faces.append(people.faces[i])
                to_ret.people.append(people.people[i])
                #checking beard
                prediction = self.beard.modelPredict(face)[0][0]#checking for a
beard
                to_ret.people[j].beard = prediction > 0.5
                #checking gender
                prediction = self.gender.modelPredict(face)[0][0]#checking for
gender
                to_ret.people[j].gender = prediction > 0.5
                #checking glass
                prediction = self.glass.modelPredict(face)[0]#checking glass type
                index = np.argmax(prediction)
                to_ret.people[j].sunglass = index == 0
                to_ret.people[j].glass = index == 1

                j += 1
        return to_ret
```

הסבר:

שם מחלקה: ModelData

תפקיד: לאפשר שימוש נוח במודלים מסוג tensorflow lite.

משתנים עיקריים:

- model המודל בעזרתו יבוצעו ניחושים.

מתודות:

- בנאי, טוען את המודל מהמיקום שניתן לו ושומר מקום בזיכרון.
- modelPredict מתודה משתמשת במודל בשביל לבצע חיזויים.

שם מחלקה: metaData

תפקיד: שימוש במודלים בשביל לחץ מידע.

משתנים עיקריים:

- glass אובייקט מסוג ModelData כאשר המודל הוא של המשקפיים.
- gender אובייקט מסוג ModelData כאשר המודל הוא של המגדר.
- beard אובייקט מסוג ModelData כאשר המודל הוא של הזקן.
- mask אובייקט מסוג ModelData כאשר המודל הוא של המסכה.
- mask_th השערה מינימלית עבורה אדם יחשב עוטה מסכה.

מתודות:

- בנאי, יוצר את האובייקטים העיקריים.
- getMetaData מקבל אובייקט של כל האנשים, עבור כל אדם מוציא עליו מידע ומחזיר מידע מעודכן לגבי כל האנשים שנמצאו.

`Camerona/keras/classifiers.py`

הקובץ מכיל שימוש במודלים מסוג tensorflow בשביל לחלץ מידע על אנשים שנמצאו בתמונה.

תוכן:

```
from uti import *

class maskClassifier():
      """
      This class uses the mask classifier
      """
      #mask - 1
      #no mask - 0
      def __init__(self, con_th = 0.8, model_path = "models/mask.h5"):
            """
            this method is the C'tor
            input: min confidence to tell if a person is wearing a mask,
            the model path
            """
            self.model = models.load_model(model_path)
            self.conf_th = con_th

      def checkMask(self, people):
            """
            This method checks if given people wear a mask
            input: the people to check if
            output: new People object that contains only the people without a mask
            """
            #taking only the faces
            check = np.array(people.faces)
            #predicting
            predictions = self.model.predict(check)
            toRet = People()
```

```
        for i in range(0, len(predictions)):
            if 1 - predictions[i] > self.conf_th:#checking if the prediction
exceeds minimum requirements
                toRet.faces.append(people.faces[i])
                toRet.people.append(people.people[i])
        return toRet

class metaData():
    """
    This class uses all the other model to extract data about the faces found
    """
    def __init__(self, glass_path = "models/glass.h5", gender_path =
"models/gender.h5", beard_path = "models/beard.h5"):
        """
        This method is the C'tor
        input: models paths
        """
        #index 0 - sunglass
        #index 1 - glass
        #index 2 - nothing
        self.glass_model = models.load_model(glass_path)
        #0 - female
        #1 - male
        self.gender_model = models.load_model(gender_path)
        #0 - no beard
        #1 - beard
        self.beard_model = models.load_model(beard_path)

    def getMetaData(self, people):
        """
        This method uses the models to extract data
        input: people to extract data about
        output: the input object with updated data
        """
        check = np.array(people.faces)
        #predicting
        glass_prediction = self.glass_model.predict(check)
        gender_prediction = self.gender_model.predict(check)
        beard_prediction = self.beard_model.predict(check)
        for i in range(0, len(check)):#iterating through all the faces
            people.people[i].gender = gender_prediction[i] > 0.5#updating gender
data
```

```
        people.people[i].beard = beard_prediction[i] > 0.5#updating beard data
        #updating glass data
        index = np.argmax(glass_prediction[i])
        people.people[i].sunglass = index == 0
        people.people[i].glass = index == 1
    return people
```

הסבר:

שם מחלקה: maskClassifier

תפקיד: מאפשר שימוש נוח ויעיל במודל של המסכה מסוג tensorflow.

משתנים עיקריים:

- model המודל.
- conf_th מינימום בשביל להחשיב אדם כעוטה מסכה.

מתודות:

- בנאי, טוען את המודל מהדיסק ומגדיר מינימום.
- checkMask מקבל את כל האנשים שנמצאו, בודק עבור כולם האם הם עוטים מסכה או לא, מחזיר אובייקט מסוג People המכיל מידע רק על מי שנמצא לא עוטה מסכה.

שם מחלקה: metaData

תפקיד: משתמש במודלים מסוג tensorflow בשביל לחלץ מידע.

משתנים עיקריים:

- glass_model המודל של המשקפיים.
- gender_model המודל של המגדר.
- beard_model המודל של הזקן.

מתודות:

- בנאי, טוען את המודלים.
- getMetaData מחלץ מידע לגבי כל אדם שנמצא.

`buildVoice.py`

הקובץ מכיל קוד שממיר טקסט לקובץ קול.

מיקומים:

- Camerona/preperations/audio/buildVoice.py

תוכן:

```
from gtts import gTTS

#this code is used to create the voice file

text = "the"
```

```
save_at = "voices/the.mp3"

lan = 'en'

speech = gTTS(text = text, lang = lan, slow = False)

speech.save(save_at)
```

הסבר:
מכניסים טקסט רצוי ומיקום בו הוא ישמר, הקוד משתמש בספרייה קיימת בשביל לייצר הקלטה של הטקסט ושומר אותו במיקום המבוקש.

`transfer.py`
הקובץ מכיל קוד שמעביר את כל התמונות מתיקייה אחת לאחרת.
מיקומים:

- Camerona/preperations/model training/dataset tools/transfer.py

תוכן:

```
import os
import sys

"""

This code transfers images from one directory to another

"""

"""
excpects:
transfer.py input_folder output_folder
"""
def main():
      data = sys.argv[1:]
      print(data)
      input_path = data[0]
      dst_path = data[1]
      i = len(os.listdir(dst_path)) + 1
      images = os.listdir(input_path)
      print(len(images))
      for image in images:#iterating through images and transferring them
```

```
            full_input_path = os.path.join(input_path, image)
            output_path = os.path.join(dst_path, str(i)+'.jpg')
            os.rename(full_input_path, output_path)
            i += 1

if __name__ == '__main__':
    main()
```

הסבר:
בעת הקריאה להרצת הקוד יתקבלו בתור פרמטרים מיקום בו נמצאות התמונות ומיקום אליו הן יעברו, הקוד מחלץ את הפרמטרים שהתקבלו ולפיהם מעביר את המידע.

`cnt.py`
הקובץ מכיל קוד שסופר את מספר הקבצים בתיקייה.
מיקומים:

- Camerona/preperations/model training/dataset tools/cnt.py

תוכן:

```
import os
import sys


#this code counts all the files in a directory

"""
excpects:
cnt.py all.txt folder_photos
"""
print(len(os.listdir(sys.argv[1])))
```

הסבר:
הקוד מקבל כפרמטר תיקייה בה עליו לספור את הקבצים, הקוד יספור את הקבצים וידפיס את מספר הקבצים שיש בתוך התיקייה.

`split.py`
הקובץ מכיל קוד המפצל קובץ מרכזי של מידע לשלושה קבצים קטנים יותר.
מיקומים:

- Camerona/preperations/model training/dataset tools/split.py

תוכן:

```python
import os
import sys
"""

This code splits a file that contains all the data to
3 files

"""
def write(path, data):
    """
    This function writes data to file
    input: file path, data to write
    output: non
    """
    f = open(path, "w+")
    data = '\n'.join(data)
    f.write(data)
    f.close()

"""
expects:
split.py all_data.txt train.txt percentage_in_float test.txt percentage_in_float
val.txt
"""
def main():
    #extracting data given in call
    data = sys.argv[1:]
    data_file = data[0]
    train = data[1]
    train_am = float(data[2])
    test = data[3]
    test_am = float(data[4])
    val = data[5]
    #taking the data from the full file
    f = open(data_file, 'r')
    data = f.read().split('\n')
    f.close()
    l = len(data)
    #splitting the data to different arrays
    test_data = data[:int(l*test_am)]
    train_data = data[int(l*test_am):int(l*(test_am+train_am))]
```

```
    val_data = data[int(l*(test_am+train_am)):]
    #writing the splitted data into different files
    write(val, val_data)
    write(test, test_data)
    write(train, train_data)

if __name__ == '__main__':
    main()
```

הסבר:

הקוד מקבל בתור פרמטרים מספר דברים, מיקום של קובץ מרכזי, מיקום של שלושה קבצים אליהם יישמר המידע, שני מספרים בין 0-1 שמייצגים את החלק מהמידע המלא שיכנס לכל אחד משני הקבצים הראשונים, השארית נכנסת לקובץ השלישי, בגדול המטרה היא לפצל דאטה סט שלם לשלושה חלקים, אימון, אימות ומבחן.

`rotate.py`

הקובץ מכיל קוד שיוצר העתקים מסובבים של כל התמונות בתיקייה מסויימת.

מיקומים:

- Camerona/preperations/model training/dataset tools/rotate.py

תוכן:

```
import cv2
import imutils
import os
import sys
import numpy as np

"""

this code rotates all the photos in a given directory

"""

"""
expect:
rotate.py input_folder output_folder
"""
def main():
    #extracting input and output directories
    data = sys.argv[1:]
```

```
    input_path = data[0]
    dst_path = data[1]
    images = os.listdir(input_path)
    i = len(os.listdir(dst_path)) + 1
    j = 1
    for image in images:#iterating through images
        if j%100 == 0:
            print("{0}/{1}".format(j, len(images)))
        j += 1
        full_input = os.path.join(input_path, image)
        im = cv2.imread(full_input)#loading base image
        full_output = os.path.join(dst_path, str(i)+'.jpg')
        os.rename(full_input, full_output)
        i += 1
        for angle in np.arange(10, 30, 5):#rotating the image
            rotated = imutils.rotate(im, angle)
            full_output = os.path.join(dst_path, str(i)+'.jpg')
            cv2.imwrite(full_output, rotated)#saving rotated image
            i += 1
        for angle in np.arange(330, 360, 5):#rotating the image
            rotated = imutils.rotate(im, angle)
            full_output = os.path.join(dst_path, str(i)+'.jpg')
            cv2.imwrite(full_output, rotated)#saving rotated image
            i += 1

if __name__ == '__main__':
    main()
```

הסבר:

הקוד מקבל שני פרמטרים, תיקיית מקור ותיקיית יעד, הקוד עובר על כל התמונות בתיקיית המקור, עבור כל תמונה הוא יוצר לה העתקים מסובבים ושומר אותם בתיקיית היעד, הקוד נועד בשביל לבצע אוגמנטציה על המידע.

`cutVid.py`

הקובץ מכיל קוד שלוקח כל פריים חמישי מסרטון ושומר אותה בתיקייה מסויימת.

מיקומים:

- Camerona/preperations/model training/dataset tools/cutVid.py

תוכן:

```
import imutils
```

```python
import sys
import os
import cv2

"""

This code takes a video and cuts it into frames

"""

"""
expects:
cutVid.py input_file output_folder
"""
def main():
    #extracting the input video path and output directory
    data = sys.argv[1:]
    print(data)
    input_file = data[0]
    dst_path = data[1]
    i = len(os.listdir(dst_path)) + 1
    cap = cv2.VideoCapture(input_file)
    while(cap.isOpened()):#iterating through frames of the video
        ret, frame = cap.read()
        if ret == False:
            break
        for _ in range(0,5):#skipping 5 frames
            _,_ = cap.read()
        #frame = imutils.rotate(frame, 90)
        output_path = os.path.join(dst_path, str(i)+'.jpg')
        cv2.imwrite(output_path, frame)#writing image
        i+=1
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
     main()
```

הסבר:

הקוד מקבל כפרמטרים מיקום סרטון ומיקום תיקיית יעד, הקוד רץ על הסרטון ושומר כל פריים חמישי בתיקיית היעד.

`cutFaces.py`

הקובץ מכיל קוד שחותך את הפרצופים מכל התמונות בתיקייה מסויימת ושומר אותם בתיקייה אחרת.

מיקומים:

- Camerona/preperations/model training/dataset tools/cutFaces.py

תוכן:

```
import sys
import os
import cv2
from time import time
import numpy as np

"""

This code take a directory of images, cut the faces
 in each image and saves the faces

"""

class Coffe():
     def __init__(self, con_th = 0.8):
          tstamp = time()
          prototxtPath = "models/face-detection-weights.prototxt"
          weightsPath = "models/face-detection-model.caffemodel"
          self.net = cv2.dnn.readNet(prototxtPath, weightsPath)
          self.conf_th = con_th

     def getFaces(self, img):
          (h, w) = img.shape[:2]

          blob = cv2.dnn.blobFromImage(img, 1.0, (300, 300), (104.0, 177.0, 123.0))

          self.net.setInput(blob)
          detections = self.net.forward()[0][0]

          toRet = []
          for i in range(0, detections.shape[0]):
               confidence = detections[i][2]
```

```
                if confidence > self.conf_th:
                      box = detections[i, 3:7] % np.array([1,1,1,1]) * np.array([w, h,
w, h])
                      (startX, startY, endX, endY) = box.astype("int")
                      (startX, startY) = (max(0, startX), max(0, startY))
                      (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

                      start_point = (int(startX),int(startY))
                      end_point = (int(endX),int(endY))
                      if end_point[0] <= start_point[0]:
                            end_point = (w, end_point[1])
                      if end_point[1] <= start_point[1]:
                            end_point = (end_point[0], h)
                      face = img[start_point[1]:end_point[1],
start_point[0]:end_point[0]]
                      face = cv2.resize(face, (64,64))
                      toRet.append(face)
           return toRet

"""
expects:
cutFaces.py input_folder output_folder
"""
def main():
      data = sys.argv[1:]
      input_path = data[0]
      dst_path = data[1]
      i = len(os.listdir(dst_path)) + 1
      images = os.listdir(input_path)
      print(len(images))
      detector = Coffe()
      for image in images:#iterating through the images in a given directory
            full_input_path = os.path.join(input_path, image)
            img = cv2.imread(full_input_path)
            im = img
            faces = detector.getFaces(im)
            for face in faces:#iterating through the faces found in image
                  output_path = os.path.join(dst_path, str(i)+'.jpg')
                  cv2.imwrite(output_path, face)
                  i += 1

if __name__ == '__main__':
```

```
    main()
```

הסבר:
הקוד מקבל כפרמטרים תיקיית מקור המכילה תמונות של אנשים ותיקיית יעד, הקוד יעבור על כל התמונות בתיקיית המקור, יחתוך משם את כל הפרצופים וישמור אותם בתיקיית היעד.

`buildData.py`
הקובץ מכיל קוד שבונה את קובץ המידע המרכזי.
מיקומים:

- Camerona/preperations/model training/dataset tools/buildData.py

תוכן:

```
import os
import random
import sys

"""

This code builds the data files

"""

"""
expects:
buildData.py all.txt folder_photos class_number folder_photos class_number
folder_photos class_number ...
"""
def main():
      data = sys.argv[1:]
      all_file = data[0]
      data = data[1:]
      i = 1
      folder_val = {}
      while i < len(data):#extracting the directory and directory value
            folder_val[data[i-1]] = data[i]
            i+=2
      data = []
      for key in folder_val:#iterating through directories
            images = os.listdir(key)
            for img in images:#iterating through images in directories
```

```
                data.append(key+img+' '+folder_val[key])
    random.shuffle(data)
    with open(all_file, 'w+') as f:#writing all the data into a file
        f.write('\n'.join(data))

if __name__ == '__main__':
    main()
```

הסבר:

הקוד מקבל כפרמטרים מיקום של קובץ טקסט בו נשמר המידע, תיקיות ומספר עבור כל תיקייה, הקוד ישמור את כל המידע בקובץ מרכזי, עבור כל הקבצים בתיקייה מסויימת הקוד ישמור את שם הקובץ יחד עם המספר שהקוד קיבל עבור התיקייה, לדוגמה, עבור הקלט:

all.txt some_folder 0 some_folder1 1

בקובץ all.txt יהיו שורות רבות כאשר כל שורה מכילה כתובת מלאה של קובץ אחריו רווח ואז הערך שניתן לתיקייה, נגיד שבתיקייה some_folder יש את הקובץ 1.jpg אז בקובץ הטקסט יהיה:

some_folder/1.jpg 0

`beard_classification.py`

הקובץ מכיל מחלקה שמשתמשת במודל של הזקן בשביל לחלץ מידע לגבי אנשים.

מיקומים:

- Camerona/preperations/model training/keras/beard classification/beard_classification.py

תוכן:

```
from uti import *

class beardClassifier():
    """
    This class uses the beard classifier
    """
    def __init__(self, model_path = "models/beard.h5"):
        """
        This is the C'tor
        input: model path
        """
        self.model = models.load_model(model_path)

    def checkBeard(self, people):
        """
        This method uses the model to predict beards
```

```
        input: People object that contains the faces
        """
        check = np.array(people.faces)
        #predicting
        predictions = self.model.predict(check)
        for i in range(0, len(predictions)):#iterating through predictions
              people.people[i].beard = predictions[i] < 0.5
        return people
```

הסבר:

שם מחלקה: beardClassifier

תפקיד: משתמשת במודל של הזקן בשביל לחלץ מידע

משתנים עיקריים:

- model מכיל אובייקט של מודל tensorflow.

מתודות:

- בנאי, טוען את המודל מהמיקום שניתן.
- checkBeard עבור כל אדם בתוך האובייקט שניתן מסוג People בודק האם יש לו זקן ומעדכן את שדה המידע הזה אצל אותו אדם.

`gender_classification.py`

הקובץ מכיל מחלקה שמשתמשת במודל של המגדר בשביל לחלץ מידע לגבי אנשים.

מיקומים:

- Camerona/preperations/model training/keras/gender classification/gender_classification.py

תוכן:

```
from uti import *

class genderClassi():
      """
      This class is the class that uses the gender classifier
      """
      def __init__(self, model_path = "models/gender_best_model.h5"):
            self.model = models.load_model(model_path)

      def checkGenders(self, faces):
            #getting only the faces
            check = []
            for face in faces:
                  check.append(face.face)
```

```
        check = np.array(check)
        predictions = self.model.predict(check)
        for i in range(0, len(predictions)):
            faces[i].gender = (predictions[i] < 0.5)
        return faces
```

הסבר:

שם מחלקה: genderClassi

תפקיד: שימוש במודל המגדר במטרה לחלץ מידע לגבי פרצופים.

משתנים עיקריים:

- model מכיל אובייקט של מודל מסוג tensorflow.

מתודות:

- בנאי, טוען את המודל מהמיקום הנתון.
- checkGenders עבור כל אדם בתוך האובייקט שניתן מסוג People בודק את מגדרו ומעדכן את שדה המידע הזה אצל אותו אדם.

glass_classification.py

הקובץ מכיל מחלקה שמשתמשת במודל של המשקפיים בשביל לחלץ מידע לגבי אנשים.

מיקומים:

- Camerona/preperations/model training/keras/glass classification/glass_classification.py

תוכן:

```
from tensorflow.keras import models
import numpy as np

class glassClassifier():
    """
    This class uses the glass classifier
    """
    #index 0 - sunglass
    #index 1 - glass
    #index 2 - nothing
    def __init__(self, model_path = "models/glass-model.h5"):
        """
        This is the C'tor
        input: model path
        """
        self.model = models.load_model(model_path)

    def checkGlassType(self, people):
```

```
        """
        This method uses the model to predict glass type
        input: People object that contains the faces
        """
        check = np.array(people.faces)
        #predicting
        predictions = self.model.predict(check)
        for i in range(0, len(predictions)):#iterating through predictions
              index = np.argmax(predictions[i])
              people.people[i].sunglass = index == 0
              people.people[i].glass = index == 1
        return people
```

הסבר:

שם מחלקה: glassClassifier

תפקיד: שימוש במודל המשקפיים בשביל לחלץ מידע לגבי פרצופים.

משתנים עיקריים:

- model מכיל אובייקט של מודל מסוג tensorflow.

מתודות:

- בנאי, טוען את המודל מהמיקום הנתון.
- checkGlassType עבור כל אדם בתוך האובייקט שניתן מסוג People בודק את סוג המשקפיים ומעדכן את שדה המידע הזה אצל אותו אדם.

`mask_classification.py`

הקובץ מכיל מחלקה שמשתמשת במודל של המסכה בשביל לחלץ מידע לגבי אנשים.

מיקומים:

- Camerona/preperations/model training/keras/mask classification/mask_classification.py

תוכן:

```
from uti import *

class maskClassifier():
      """
      This class uses the mask classifier
      """
      #mask - 1
      #no mask - 0
      def __init__(self, con_th = 0.8, model_path = "models/mask.h5"):
```

```
        """
        This is the C'tor
        input: minimum confidence, model path
        """
        self.model = models.load_model(model_path)
        self.conf_th = con_th

    def checkMask(self, people):
        """
        This method uses the model to predict weather the person wears a mask or not
        input: People object contains faces to check
        """
        check = np.array(people.faces)
        #predicting
        predictions = self.model.predict(check)
        toRet = People()
        for i in range(0, len(predictions)):#iterating through predictions
            if 1 - predictions[i] > self.conf_th:
                toRet.faces.append(people.faces[i])
                toRet.people.append(people.people[i])
        return toRet
```

הסבר:

שם מחלקה: maskClassifier

תפקיד: שימוש במודל המסכה למציאת אנשים שאינם עוטים מסכה.

משתנים עיקריים:

- model מכיל אובייקט של מודל מסוג tensorflow.

מתודות:

- בנאי, טוען את המודל מהמיקום הנתון.
- checkMask עבור כל האנשים שנמצאו בתמונה בודק האם הם עוטים מסכה, במידה ולא הוא מוסיף אותם לאובייקט חדש ולאחר שהמתודה בודקת את כולם מחזירה את האובייקט החדש שמכיל רק אנשים שלא עטו מסכה.

Camerona\preperations\model training\keras\beard classification\training\beard_classification.ipynb
Camerona\preperations\model training\keras\mask classification\training\mask_classification.ipynb
Camerona\preperations\model training\keras\gender classification\training\gender_classification.ipynb

הקבצים הם קבצי האימון של המודלים, הם זהים כמעט לחלוטין, ההבדל היחידי הוא המידע עליו כל מודל מאומן, משמע ההבדל היחידי הוא כמה משתנים קבועים וחוץ מזה הקוד הוא בדיוק אותו הדבר.

תוכן:

```
# -*- coding: utf-8 -*-
"""mask_classi.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1eKZ4od0097dTxfBf600NMgdof90D4TLc
"""

# Commented out IPython magic to ensure Python compatibility.
# #downloading weights and biases
# %%capture
# !pip install wandb -qqq

import tensorflow.keras as keras
from google.colab import drive
import numpy as np
import cv2
import wandb
from wandb.keras import WandbCallback

#constant things I will use in the program
MODELS_PATH = "/content/gdrive/My Drive/magshimim/camerona/models/"
BASE = "mask_base_model.h5"
BEST = "mask_best_model.h5"
DATASET_PATH_DRIVE = "/content/gdrive/My Drive/magshimim/camerona/dataset/mask/"
DATASET_PATH_COLAB = "/content/mask/"
VAL_TEXT = DATASET_PATH_COLAB+"val.txt"
TRAIN_TEXT = DATASET_PATH_COLAB+"train.txt"
TEST_TEXT = DATASET_PATH_COLAB+"test.txt"

#mounting google drive to access files
drive.mount('/content/gdrive/')

#doing this because in this way the data loads faster
!ln -s "/content/gdrive/My Drive/magshimim/camerona/dataset/mask/" "/content/"
```

```
#logging into the weights and biases account
!wandb login

def buildModel(lr = 0.001):
  """
  this function build the model
  input: lr - learning rate of the model
  output: the model
  """
  input_layer = keras.layers.Input(shape=[64,64,3])

  #block1
  layer = keras.layers.Conv2D(32, (3,3), strides=(1,1), activation='relu',
padding='same', name='block1_conv1')(input_layer)
  layer = keras.layers.MaxPool2D(2, name='block1_max1')(layer)

  #block2
  layer = keras.layers.Conv2D(16, (3,3), strides=(1,1), activation='relu',
padding='same', name='block2_conv1')(layer)
  layer = keras.layers.MaxPool2D(2, name='block2_max1')(layer)

  #output block
  layer = keras.layers.Flatten()(layer)
  layer = keras.layers.Dense(16, activation="relu")(layer)
  layer = keras.layers.Dense(1, activation="sigmoid", name='output_layer')(layer)

  #building and compiling model
  model = keras.Model(inputs=input_layer, outputs=layer)
  opt = keras.optimizers.Adam(learning_rate=lr)
  model.compile(optimizer = opt, loss = ['binary_crossentropy'],
metrics=[keras.metrics.BinaryAccuracy()])
  print(model.summary())

  return model

def getData(file_path):
  """
  This function build the data from a file
  input: file path
  output: the x and y of the data
  """
  #opening the data file
```

```
  f = open(file_path, 'r')
  data = f.read().split('\n')
  f.close()
  x = []
  y = []
  all = len(data)
  cnt = 1
  #iterating through the file lines
  for line in data:
    if cnt%250 == 0:
      print("{0}/{1}".format(cnt, all))
    cnt += 1
    line_data = line.split(" ")
    line_data[0] = line_data[0].replace('\\', '/')
    try:
      #loading the data image
      img = cv2.imread(DATASET_PATH_COLAB+line_data[0])
      img = cv2.resize(img, (64,64))/255
    except:
      continue
    #adding the x and y data
    x.append(img)
    y.append(int(line_data[1]))
  #converting to numpy array
  x = np.array(x)
  y = np.array(y)
  return x,y

#building and saving base model
model = buildModel()
model.save(MODELS_PATH+BASE)

#initiating wandb project
wandb.init(project="mask-classification")

#callbacks
best_callback = keras.callbacks.ModelCheckpoint(MODELS_PATH+BEST, save_best_only=True)
wandb_callback = WandbCallback()

#loading data
print("loading validation data")
val_x, val_y = getData(VAL_TEXT)
```

```
print("loading train data")
train_x, train_y = getData(TRAIN_TEXT)

model.fit(train_x,
          train_y,
          epochs=24,
          validation_data=(val_x, val_y),
          callbacks=[best_callback, wandb_callback])

#loading the best model, we used the tf callback to save it
model = keras.models.load_model(MODELS_PATH+BEST)

#loading test x and y data
test_x, test_y = getData(TEST_TEXT)

#evaluating the model
loss, acc = model.evaluate(x = test_x, y = test_y)
```

הסבר:

משתנים עיקריים:

- model משתנה שמחזיק את אובייקט המודל.
- best_callback פרמטר שאחראי על שמירת המודל הטוב ביותר שנוצר במהלך האימונים.
- wandb_callback פרמטר שאחראי על שליחת מידע האימונים לאתר weights and biases בשביל לעקוב אחרי האימונים בצורה גרפית ונוחה.

פונקציות:

- buildModel פונקציה שבה נבנה המודל, בתוך הפונקציה נבנה המודל ואז הוא מוחזר בצורתו הסופית, לא היה חייב לעשות זאת בפונקציה פשוט זה יותר נוח.
- getData, הפונקציה אחראית על חילוץ המידע מגוגל דרייב וטעינתו לקוד בזמן ריצה בשביל לאמן את המודל, הפונקציה עוברת על הקובץ המבוקש וממנו מוציאה את כל המידע הרלוונטי תוך טעינת התמונות מגוגל דרייב.

`maskLiteAcc.ipynb`
`genderLiteAcc.ipynb`
`beardLiteAcc.ipynb`

הקבצים מכילים קוד שאחראי על בדיקת הדיוק של המודלים.

מיקומים:

- Camerona\preperations\model training\tf-lite\check accuracy\maskLiteAcc.ipynb
- Camerona\preperations\model training\tf-lite\check accuracy\genderLiteAcc.ipynb

- Camerona\preperations\model training\tf-lite\check accuracy\beardLiteAcc.ipynb

תוכן:

```
# -*- coding: utf-8 -*-
"""maskLiteAcc.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1kk8SjbzdBCy7LrXTuISydtNiAzLkbe75
"""

import tensorflow.keras as keras
import tensorflow as tf
from google.colab import drive
import numpy as np
import cv2

#constant stuff
MODEL = "/content/models/mask.tflite"
DATASET_PATH_COLAB = "/content/mask/"
TEST_TEXT = DATASET_PATH_COLAB+"test.txt"

#mounting to drive
drive.mount('/content/gdrive/')

!ln -s "/content/gdrive/My Drive/magshimim/camerona/dataset/mask/" "/content/"
!ln -s "/content/gdrive/My Drive/magshimim/camerona/models/" "/content/"

def getData(file_path):
  """
  This function loads the data from a file
  input: the text file path
  output: x and y of the data
  """
  #opening the file
  f = open(file_path, 'r')
  data = f.read().split('\n')
  f.close()
  x = []
  y = []
```

```
  all = len(data)
  cnt = 1
  for line in data:
    #iterating through the file
    if cnt%250 == 0:
      print("{0}/{1}".format(cnt, all))
    cnt += 1
    line_data = line.split(" ")
    line_data[0] = line_data[0].replace('\\', '/')
    try:
      #loading the image
      img = cv2.imread(DATASET_PATH_COLAB+line_data[0])
      img = cv2.resize(img, (64,64))/255
    except:
      continue
    #adding data
    x.append(img)
    y.append(int(line_data[1]))
  #converting to numpy array
  x = np.array(x)
  y = np.array(y)
  return x,y

class ModelData():
  """
  class that uses the tflite models, it's easier to do everything like this
  """
     def __init__(self, model_path):
    """
    This method is the C'tor
    input: model path
    """
           self.model = tf.lite.Interpreter(model_path=model_path)
           self.input_tensor_index = self.model.get_input_details()[0]['index']
           self.output_tensor_index = self.model.get_output_details()[0]['index']
           self.model.allocate_tensors()

     def modelPredict(self, face):
    """
    This method is responsible to predict
    input: data to predict its output
    """
```

```
            self.model.set_tensor(self.input_tensor_index, face)
            self.model.invoke()
            prediction = self.model.get_tensor(self.output_tensor_index)
            return prediction

model = ModelData(MODEL)

test_x, test_y = getData(TEST_TEXT)
predictions = []

#getting predictions
for img in test_x:
  img = np.array([img], dtype=np.float32)
  predictions.append(model.modelPredict(img))

#processing the data
p = []
y = []
size = test_y.shape[0]
i = 0
while i < size:
  y.append([test_y[i]])
  p.append(predictions[i][0])
  i+=1
predictions = p

#checking accuracy
m = tf.keras.metrics.BinaryAccuracy()
m.update_state(test_y, predictions)
print(m.result().numpy())
```

הסבר:

שם מחלקה: ModelData

תפקיד: מממש דרך נוחה לשימוש במודלים מסוג Tensorflow lite.

משתנים עיקריים:

- model מחזיק את אובייקט המודל.

מתודות:

- בנאי, טוען את המודל מהכתובת המבוקשת, שומר מקום בזיכרון לקראת שימוש במודל.
- modelPredict משתמש במודל בשביל לבצע חיזויים בנוגע למידע.

משתנים עיקריים:

- model משתנה שמחזיק את אובייקט המודל.

פונקציות:

- getData, הפונקציה אחראית על חילוץ המידע מגוגל דרייב וטעינתו לקוד בזמן ריצה בשביל לאמן את המודל, הפונקציה עוברת על הקובץ המבוקש וממנו מוציאה את כל המידע הרלוונטי תוך טעינת התמונות מגוגל דרייב.

`glass_classification.ipynb`

הקובץ מכיל את הקוד עבור אימון המודל של המשקפיים, ברובו הוא זהה לקודים האחרים האחראים על אימון, ההבדל היחידי נמצא בפונקצית בניית המודל וטעינת המידע כיוון שהמודל שונה והמידע נשמר בפורמט שונה.

מיקומים:

- Camerona\preperations\model training\keras\glass classification\training\glass_classification.ipynb

תוכן:

```
# -*- coding: utf-8 -*-
"""glass_classi.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1VOcd8ti3AtHFL-SmzkWdTKItA6pwe5Vj

**Glass classification model**
"""

# Commented out IPython magic to ensure Python compatibility.
# #downloading weights and biases
# %%capture
# !pip install wandb -qqq

import tensorflow.keras as keras
from google.colab import drive
import numpy as np
import cv2
import wandb
from wandb.keras import WandbCallback

#constant things I will use in the program
MODELS_PATH = "/content/gdrive/My Drive/magshimim/camerona/models/"
BASE = "glass_base_model.h5"
BEST = "glass_best_model.h5"
```

```
DATASET_PATH_DRIVE = "/content/gdrive/My Drive/magshimim/camerona/dataset/glass/"
DATASET_PATH_COLAB = "/content/glass/"

VAL_TEXT = DATASET_PATH_COLAB+"val.txt"
TRAIN_TEXT = DATASET_PATH_COLAB+"train.txt"
TEST_TEXT = DATASET_PATH_COLAB+"test.txt"
#mounting google drive to access files
drive.mount('/content/gdrive/')

#doing this because in this way the data loads faster
!ln -s "/content/gdrive/My Drive/magshimim/camerona/dataset/glass/" "/content/"

#logging into the weights and biases account
!wandb login

def buildModel(lr = 0.001):
  """
  this function build the model
  input: lr - learning rate of the model
  output: the model
  """
  input_layer = keras.layers.Input(shape=[64,64,3])

  #block1
  layer = keras.layers.Conv2D(32, 2, strides=1, activation='relu', padding='same',
name='block1_conv1')(input_layer)
  layer = keras.layers.MaxPool2D(2, name='block1_max1')(layer)

  #block2
  layer = keras.layers.Conv2D(32, 2, strides=1, activation='relu', padding='same',
name='block2_conv1')(layer)
  layer = keras.layers.MaxPool2D(2, name='block2_max1')(layer)

  #output block
  layer = keras.layers.Flatten()(layer)
  layer = keras.layers.Dense(16, activation="relu")(layer)
  layer = keras.layers.Dense(3, activation="softmax", name='output_layer')(layer)

  #building and compiling model
  model = keras.Model(inputs=input_layer, outputs=layer)
  opt = keras.optimizers.Adam(learning_rate=lr)
  model.compile(optimizer = opt, loss = ["categorical_crossentropy"],
```

```
metrics=["categorical_accuracy"])
  print(model.summary())

  return model

def getData(file_path):
  """
  This function build the data from a file
  input: file path
  output: the x and y of the data
  """
  #opening the data file
  f = open(file_path, 'r')
  data = f.read().split('\n')
  f.close()
  x = []
  y = []
  all = len(data)
  cnt = 1
  #iterating through the file lines
  for line in data:
    if cnt%250 == 0:
      print("{0}/{1}".format(cnt, all))
    cnt += 1
    line_data = line.split(" ")
    try:
      #loading the data image
      img = cv2.imread(DATASET_PATH_COLAB+line_data[0])
      img = cv2.resize(img, (64,64))/255
    except:
      continue
    #adding the x and y data
    _y = np.zeros(3)
    _y[int(line_data[1])] = 1
    x.append(img)
    y.append(_y)
  #converting to numpy array
  x = np.array(x)
  y = np.array(y)
  return x,y

wandb.init(project="glass-classification")
```

```
#callbacks
best_callback = keras.callbacks.ModelCheckpoint(MODELS_PATH+BEST, save_best_only=True)
wandb_callback = WandbCallback()

#loading data
print("loading validation data")
val_x, val_y = getData(VAL_TEXT)
print("loading train data")
train_x, train_y = getData(TRAIN_TEXT)

#building and saving base model - first time training
model = buildModel()
model.save(MODELS_PATH+BASE)

model.fit(train_x,
          train_y,
          epochs=24,
          validation_data=(val_x, val_y),
          callbacks=[best_callback, wandb_callback])

#loading the best model, we used the tf callback to save it
model = keras.models.load_model(MODELS_PATH+BEST)

#loading test x and y data
test_x, test_y = getData(TEST_TEXT)

#evaluating the model
loss, acc = model.evaluate(x = test_x, y = test_y)
```

הסבר:

משתנים עיקריים:

- model משתנה שמחזיק את אובייקט המודל.
- best_callback פרמטר שאחראי על שמירת המודל הטוב ביותר שנוצר במהלך האימונים.
- wandb_callback פרמטר שאחראי על שליחת מידע האימונים לאתר weights and biases בשביל לעקוב אחרי האימונים בצורה גרפית ונוחה.

פונקציות:

- buildModel הפונקציה אחראית על בניית המודל, המודל הוא מולטי קלאס קלסיפייר.
- getData הפונקציה אחראית על טעינת מידע מקובץ מסויים, הקוד קורא את המידע שנמצא בקובץ, טוען את התמונות ומצרף אליהן את המידע הרלוונטי.

glassLiteAcc.ipynb

הקובץ מכיל קוד שאחראי על בדיקת הדיוק של המודל משקפיים ב Tensorflow lite.

מיקומים:

- Camerona\preperations\model training\tf-lite\check accuracy\glassLiteAcc.ipynb

תוכן:

```
# -*- coding: utf-8 -*-
"""glassLiteAcc.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1ppc2FDy9mj1PCBVKk-cSrPha6doL7T6M
"""

import tensorflow.keras as keras
import tensorflow as tf
from google.colab import drive
import numpy as np
import cv2

#constant stuff
MODEL = "/content/models/glass.tflite"
DATASET_PATH_COLAB = "/content/glass/"
TEST_TEXT = DATASET_PATH_COLAB+"test.txt"

#mounting to drive
drive.mount('/content/gdrive/')

!ln -s "/content/gdrive/My Drive/magshimim/camerona/dataset/glass/" "/content/"
!ln -s "/content/gdrive/My Drive/magshimim/camerona/models/" "/content/"

def getData(file_path):
  """
  This function loads the data from a file
  input: the text file path
  output: x and y of the data
  """
  #opening the file
  f = open(file_path, 'r')
  data = f.read().split('\n')
```

```
  f.close()
  x = []
  y = []
  all = len(data)
  cnt = 1
  for line in data:
    #iterating through the file
    if cnt%250 == 0:
      print("{0}/{1}".format(cnt, all))
    cnt += 1
    line_data = line.split(" ")
    try:
      #loading the image
      img = cv2.imread(DATASET_PATH_COLAB+line_data[0])
      img = cv2.resize(img, (64,64))/255
    except:
      continue
    #adding data
    _y = np.zeros(3)
    _y[int(line_data[1])] = 1
    x.append(img)
    y.append(_y)
  #converting to numpy array
  x = np.array(x)
  y = np.array(y)
  return x,y

class ModelData():
  """
  class that uses the tflite models, it's easier to do everything like this
  """
     def __init__(self, model_path):
    """
    This method is the C'tor
    input: model path
    """
           self.model = tf.lite.Interpreter(model_path=model_path)
           self.input_tensor_index = self.model.get_input_details()[0]['index']
           self.output_tensor_index = self.model.get_output_details()[0]['index']
           self.model.allocate_tensors()

     def modelPredict(self, face):
```

```
            self.model.set_tensor(self.input_tensor_index, face)
            self.model.invoke()
            prediction = self.model.get_tensor(self.output_tensor_index)
            return prediction

model = ModelData(MODEL)

test_x, test_y = getData(TEST_TEXT)
test_prediction = []

#getting predictions
for img in test_x:
  img = np.array([img], dtype=np.float32)
  test_prediction.append(model.modelPredict(img))

#processing the data
predictions = test_prediction
p = []
y = []
size = test_y.shape[0]
i = 0
while i < size:
  p.append(predictions[i][0])
  i+=1
predictions = np.array(p)

#checking accuracy
m = tf.keras.metrics.CategoricalAccuracy()
m.update_state(test_y, predictions)
m.result().numpy()
```

הסבר:

שם מחלקה: ModelData

תפקיד: מממש דרך נוחה לשימוש במודלים מסוג Tensorflow lite.

משתנים עיקריים:

- model מחזיק את אובייקט המודל.

מתודות:

- בנאי, טוען את המודל מהכתובת המבוקשת, שומר מקום בזיכרון לקראת שימוש במודל.
- modelPredict משתמש במודל בשביל לבצע חיזויים בנוגע למידע.

משתנים עיקריים:

- model משתנה שמחזיק את אובייקט המודל.

פונקציות:

- getData, הפונקציה אחראית על חילוץ המידע מגוגל דרייב וטעינתו לקוד בזמן ריצה בשביל לאמן את המודל, הפונקציה עוברת על הקובץ המבוקש וממנו מוציאה את כל המידע הרלוונטי תוך טעינת התמונות מגוגל דרייב.

SSD.ipynb

הקובץ מכיל קוד איתו אני בונה ומאמן את המודל שאחראי על מציאת פרצופים בתמונה.

מיקומים:

- Camerona\preperations\model training\keras\face detection\training\SSD.ipynb

תוכן:

```
# -*- coding: utf-8 -*-
"""SSD.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1bpPAXMLZ86nLeueTdpYv2lTzem7IuZPW
"""

# Commented out IPython magic to ensure Python compatibility.
# %%capture
# !pip install wandb -qqq

import tensorflow as tf
import tensorflow.keras as keras
from google.colab import drive
import numpy as np
import cv2
import keras.backend as K

MODELS_PATH = "/content/gdrive/My Drive/magshimim/models/"
SSD_BASE = "ssd_base_model.h5"
SSD_BEST = "ssd_best_model.h5"
DATASET_PATH_DRIVE = "/content/gdrive/My Drive/magshimim/dataset/"
DATASET_PATH_COLAB = "/content/dataset/"
DATASET_IMG = DATASET_PATH_COLAB + "images/"
VAL_TEXT = DATASET_PATH_COLAB+"val.txt"
TRAIN_TEXT = DATASET_PATH_COLAB+"train.txt"
TEST_TEXT = DATASET_PATH_COLAB+"test.txt"
```

```
drive.mount('/content/gdrive/')

!ln -s "/content/gdrive/My Drive/magshimim/dataset/" "/content/"

import wandb
from wandb.keras import WandbCallback
!wandb login

#input shape: (batch_size,20,2,2)
def computeArea(a):
  x1 = a[:,:,0,0]
  x2 = a[:,:,1,0]
  y1 = a[:,:,0,1]
  y2 = a[:,:,1,1]
  x_len = tf.math.subtract(x2,x1)
  y_len = tf.math.subtract(y2,y1)
  area = tf.math.multiply(x_len, y_len)
  return area

#input shape: (batch_size,20,2,2), (batch_size,20,2,2)
def IoU(pred_bbox, ground_truth):
  area_pred = computeArea(pred_bbox)
  area_gt = computeArea(ground_truth)

  inter_x1 = tf.math.maximum(pred_bbox[:,:,0,0], ground_truth[:,:,0,0])
  inter_x2 = tf.math.minimum(pred_bbox[:,:,1,0], ground_truth[:,:,1,0])
  inter_y1 = tf.math.maximum(pred_bbox[:,:,0,1], ground_truth[:,:,0,1])
  inter_y2 = tf.math.minimum(pred_bbox[:,:,1,1], ground_truth[:,:,1,1])

  y = tf.math.subtract(inter_y2,inter_y1)
  x = tf.math.subtract(inter_x2,inter_x1)

  area_inter = tf.math.multiply(x,y)

  area_union = tf.math.add(area_pred,area_gt)
  area_union = tf.math.subtract(area_union,area_inter)

  x_tf = tf.where(x<=0.0, True, False)
  y_tf = tf.where(y<=0.0, True, False)
  is_inter = tf.math.logical_or(x_tf,y_tf)
  acc = tf.where(is_inter, 0.0, tf.divide(area_inter, area_union))
```

```
  area_gt_tf = tf.where(area_gt==0.0, True, False)
  area_pred_tf = tf.where(area_pred==0.0, True, False)
  is_one_area = tf.math.logical_and(area_gt_tf,area_pred_tf)
  acc = tf.where(is_one_area, 1.0, acc)

  return acc

def boxLoss(pred_bbox, ground_truth):
  loss = tf.math.subtract(pred_bbox, ground_truth)
  loss = tf.math.abs(loss)
  loss = tf.math.sqrt(tf.math.sqrt(loss))
  loss = tf.math.multiply(10.0,loss)
  return loss

def boxLoss1(pred_bbox, ground_truth):
  acc = tf.math.pow(IoU(pred_bbox, ground_truth), 2)
  return tf.subtract(1.0, acc)*10

def loadVGG16():
  model = keras.applications.VGG16(
    include_top = False,
    weights ="imagenet",
    input_tensor = None,
    input_shape = (256,256,3),
    pooling=None)
  return model

def buildModel():
  vgg16_backbone = loadVGG16()

     #making the backbone untrainable
  for layer in vgg16_backbone.layers:
    layer.trainable = True

  ####SSD layers
  ssd = keras.layers.Conv2D(1024, 3, dilation_rate=6, activation='relu',
padding='same', name = "ssd_block1_conv1")(vgg16_backbone.output)

  ssd = keras.layers.Conv2D(1024, 1, activation='relu', padding='same', name =
"ssd_block2_conv1")(ssd)
```

```
  ssd = keras.layers.Conv2D(256, 1, activation='relu', padding='same', name =
"ssd_block3_conv1")(ssd)
  ssd = keras.layers.Conv2D(512, 3, strides=2, activation='relu', padding='same', name
= "ssd_block3_conv2")(ssd)

  ssd = keras.layers.Conv2D(128, 1, activation='relu', padding='same', name =
"ssd_block4_conv1")(ssd)
  ssd = keras.layers.Conv2D(256, 3, strides=2, activation='relu', padding='same', name
= "ssd_block4_conv2")(ssd)

  ssd = keras.layers.Conv2D(128, 1, activation='relu', padding='same', name =
"ssd_block5_conv1")(ssd)
  ssd = keras.layers.Conv2D(256, 3, activation='relu', padding='same', name =
"ssd_block5_conv2")(ssd)

  ssd = keras.layers.Conv2D(128, 1, activation='relu', padding='same', name =
"ssd_block6_conv1")(ssd)
  ssd = keras.layers.Conv2D(256, 3, activation='relu', padding='same', name =
"ssd_block6_conv2")(ssd)
  ####

     ####output layers
  #class layers
  flt = keras.layers.Flatten()(ssd)
  class_output = keras.layers.Dense(20, activation='sigmoid', name='output_class')(flt)

  #location layers
  ssd = keras.layers.Conv2D(20, 3, activation='relu', padding='same', name =
"output_loc_b4_reshape")(ssd)
  loc_output = keras.layers.Reshape((20,2,2), name='output_loc')(ssd)
  ####

  #model and optimizer
  model = keras.Model(inputs=vgg16_backbone.input, outputs=[class_output, loc_output])
  opt = keras.optimizers.SGD(learning_rate=0.075)

  #custom loss and metrics
  metrics = {'output_class':keras.metrics.BinaryAccuracy(), 'output_loc':IoU}

  losses = {'output_class':"binary_crossentropy", 'output_loc':boxLoss}

  model.compile(optimizer = opt, loss=losses, metrics=metrics)
```

```
  print(model.summary())

  return model

def buildData(data, show_every=250):
  x = []
  y_guess = []
  y_loc = []
  print(len(data))
  cnt1 = 1
  for line in data:
    if cnt1%show_every == 0:
      print(cnt1)
    cnt1 += 1
    l = line.split(' ')
    try:
      #some data was deleted and finding it is a pain, this is easier
      img = cv2.imread(DATASET_IMG+l[0])
      img = cv2.resize(img, (256,256))/255.0
    except:
      continue
    y_class = np.zeros(20)
    y_location = np.zeros((20,2,2))#[[[x1, y1],[x2, y2]]...20]
    if len(l) > 1:
      l = l[1:]
      cnt = 0
      try:
        for points in l:
          points = points.split(',')
          y_location[cnt][0][0] = tf.cast(float(points[0]), tf.float32)
          y_location[cnt][0][1] = tf.cast(float(points[1]), tf.float32)
          y_location[cnt][1][0] = tf.cast(float(points[2]), tf.float32)
          y_location[cnt][1][1] = tf.cast(float(points[3]), tf.float32)
          y_class[cnt] = 1
          cnt += 1
      except:
        continue
    x.append(img)
    y_guess.append(y_class)
    y_loc.append(y_location)
  x, y_guess, y_loc = np.array(x), np.array(y_guess), np.array(y_loc)
  print(x.shape, y_guess.shape, y_loc.shape)
```

```
  return x, y_guess, y_loc

def dataFromFile(path):
      f = open(path, 'r')
      ret = f.read().split('\n')
      f.close()
      return ret

def splitArrayToArrays(arr, am):
  l = len(arr)
  to_ret = []
  i = 0
  cnt = 0
  while i < l:
    cnt += 1
    a = []
    while i < am*cnt and i < l:
      a.append(arr[i])
      i+=1
    to_ret.append(a)
  return to_ret

"""Building model and saving it"""

model = buildModel()
model.save(MODELS_PATH+SSD_BASE)

"""Callbacks"""

wandb.init(project="SSD - face detector")

best_callback = keras.callbacks.ModelCheckpoint(MODELS_PATH+SSD_BEST,
save_best_only=True)
wandb_callback = WandbCallback()

"""**checking the default outputs**"""

val_data = [dataFromFile(VAL_TEXT)[0]]
val_x, val_y_guess, val_y_loc = buildData(val_data)
y1, y2 = model.predict(val_x)

print(y2)
```

```
#getting validation data
val_data = dataFromFile(VAL_TEXT)

#getting test pre-data
train_data = splitArrayToArrays(dataFromFile(TRAIN_TEXT), 700)

val_x, val_y_guess, val_y_loc = buildData(val_data, 100)

cnt = 1
l = len(train_data)
for data in train_data:
  print("\n\niteration {0}/{1}".format(cnt, l))
  train_x, train_y_guess, train_y_loc = buildData(data, 100)

  history = model.fit(train_x,
          [train_y_guess, train_y_loc],
          epochs=16,
          validation_data=(val_x, [val_y_guess, val_y_loc]),
          callbacks=[best_callback,wandb_callback],
          batch_size=4)
  to_save = keras.models.load_model(MODELS_PATH+SSD_BEST, compile=False)
  to_save.save(MODELS_PATH+"ssd_" + str(cnt) + ".h5")
  if cnt>=4:
    if input() == 'y':
      break
  cnt += 1

"""**checking outputs after training**"""

x, y1, y1 = buildData([val_data[0]], 100)
y1, y2 = model.predict(val_x)
print(y2)

best_models = []
for i in range(15, 19):
  best_models.append(keras.models.load_model(MODELS_PATH+"ssd_" + str(i) + ".h5"))

#loading test data
test_data = splitArrayToArrays(dataFromFile(TEST_TEXT), 1500)
test_x, test_y_guess, test_y_loc = buildData(test_data[0])
```

```
#testing all the models
best_acc = 0
best_model = None
cnt = 15
print(best_models[0].metrics_names)
for model in best_models:
  print(cnt)
  print(model.evaluate(test_x, [test_y_guess, test_y_loc]),"\n")
  cnt += 1

model = buildModel()

data = dataFromFile(TRAIN_TEXT)[:100]
test_x, test_y_guess, test_y_loc = buildData(data, 10)

model.fit(test_x,
          [test_y_guess, test_y_loc],
          epochs=24)
```

הסבר:

משתנים עיקריים:

- model משתנה שמחזיק את אובייקט המודל.
- best_callback פרמטר שאחראי על שמירת המודל הטוב ביותר שנוצר במהלך האימונים.
- wandb_callback פרמטר שאחראי על שליחת מידע האימונים לאתר weights and biases בשביל לעקוב אחרי האימונים בצורה גרפית ונוחה.

פונקציות:

- computeArea פונקציה שאחראית לחישוב שטח של הריבועים שהמודל מצא.
- IoU פונקציה לחישוב הדיוק של המודל.
- boxLoss פונקציה לחישוב הטעות של המודל בשביל תהליך האימון.
- boxLoss1 פונקציה לחישוב הטעות של המודל בשביל תהליך האימון.
- loadVGG16 פונקציה לטעינת מודל קיים ומאומן בתור התחלה למודל שלי.
- buildModel פונקציה שאחראית על בניית המודל המלא.
- buildData פונקציה שאחראית על בניית וטעינת המידע מהדרייב.
- dataFromFile פונקציה שמקלה על קריאת המידע מקובץ מסוים.
- splitArrayToArrays פונקציה המשמשת לפיצול מערך למספר מערכים.

`use_speakers.py`

הקובץ מכיל קוד שאחראי להשתמש ברמקולים.

מיקומים:

- Camerona\speakers\use_speakers.py

תוכן:

```python
from playsound import playsound

def play(person):
    """
    This function is used to play a sound by given metadata
    """
    if person.gender or person.beard:
        playsound("voices/man.mp3")
    else:
        playsound("voices/woman.mp3")
    if person.glass or person.sunglass or person.beard:
        playsound("voices/with.mp3")
        playsound("voices/the.mp3")
    if person.glass:
        playsound("voices/glasses.mp3")
    elif person.sunglass:
        playsound("voices/sunglass.mp3")
    if person.beard:
        playsound("voices/beard.mp3")
    playsound("voices/pwam.mp3")
```

הסבר:

משתנים עיקריים:

- person משתנה ששומר אובייקט מסוג Person ששומר בתוכו מידע בנוגע לפרצוף מסוים.

פונקציות:

- play מקבל אובייקט מסוג Person ולפי המידע מנגן הקלטות בהתאם.

`main.py`

הקובץ מכיל קוד שמשתמש בכל הכלים יחד ומשלב אותם, זהו הקוד המרכזי, הקוד קיים במספר מקומות, מטרתו ורוב המימוש זהים בין כולם ולכן אני אסביר באופן כללי במקום ממוקד על כל קובץ בנפרד.

מיקומים:

- Camerona\keras\main.py
- Camerona\tf-lite\main.py
- Camerona\speakers\main.py

הסבר:

משתנים עיקריים:

- face_detector משתנה שמכיל בתוכה את האובייקט באמצעותו מתבצעת איתור פרצופים בתמונה.

- metadata_models משתנה המכיל בתוכו אובייקט שבאמצעותו בקלות ניתן להוציא מידע גבי פרצופים.
- mask_classifier משתנה המכיל אובייקט באמצעותו ניתן להשתמש במודל המסכה בקלות.

פונקציות:

- main פונקציה האחראית על כל מהלך הריצה, בתוכה יש את כל המשתנים העיקריים ובתוכה מתנהל כל הניהול של הכלים.
- addressPeople פונקציה שרצה על כל אדם שנמצא לא עוטה מסכה ומשתמשת ברמקולים בשביל לפנות אליו.
- drawNoMasks פונקציה שרצה על כל אדם שנמצא לא עוטה מסכה ומוסיפה למסך מידע לגביו.

`convert_using_frozenpb.ipynb`

הקובץ מכיל .

מיקומים:

- Camerona\preperations\model training\temsorRT\using frozenpb\convert_using_frozenpb.ipynb

תוכן:

```
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.python.framework import graph_io


def keras_to_frozen_pb(model_in_path,
                       model_out_path,
                       custom_object_dict=None,
                       tensor_out_name=None,
                       tensorboard_dir=None):
    """
    Converter that transforms keras model to frozen pb model

    Args:
        model_in_path (str): Input model path (.h5)
        model_out_path (str): Output model path (dir)
        tensor_out_name (str, optional): Specified name of output tensor.
                                         If None, it will get default tensor name from
keras model.
                                         Defaults to None.
        tensorboard_dir (str, optional): Output tensorboard dir path for inspecting
output model graph.
                                         If None, it doesn't generate.
                                         Defaults to None.
```

```
    """

    graph = tf.Graph()
    with graph.as_default():
        sess = tf.compat.v1.Session()
        K.set_session(sess)
        K.set_learning_phase(0)

        # load the model to graph and sess
        model = tf.keras.models.load_model(model_in_path,
custom_objects=custom_object_dict)

        # get the tensor_out_name
        if tensor_out_name is None:
            if len(model.outputs) > 1:
                raise NameError("the model has multiple output tensors. Need to specify
output tensor name.")
            else:
                tensor_out_name = model.outputs[0].name.split(":")[0]

        # freeze the graph
        graphdef = tf.compat.v1.graph_util.convert_variables_to_constants(sess,
graph.as_graph_def(), [tensor_out_name])
        graphdef = tf.compat.v1.graph_util.remove_training_nodes(graphdef)
        graph_io.write_graph(graphdef, './', model_out_path, as_text=False)

      # output tensorboard graph
    if not tensorboard_dir is None:
        tf.compat.v1.summary.FileWriter(logdir=tensorboard_dir, graph_def=graphdef)

    return tensor_out_name
input_keras_model = "../input/mask.h5"
output_pb_model = "../output/mask.pb"


if __name__ == "__main__":
    node_out_name = keras_to_frozen_pb(input_keras_model, output_pb_model)
    print("the output node name is:", node_out_name)
import tensorflow as tf
import uff
import tensorrt as trt
```

```python
def frozen_pb_to_plan(model_path,
                      output_path,
                      tensor_in_name,
                      tensor_out_name,
                      input_size,
                      data_type=trt.float32,
                      max_batch_size=1,
                      max_workspace=1<<30,
                      tensorboard_dir=None):

    # infer with pb model
    graph_def = tf.GraphDef()
    with tf.io.gfile.GFile(model_path, "rb") as f:
        graph_def.ParseFromString(f.read())

    # convert TF frozen graph to uff model
    uff_model = uff.from_tensorflow_frozen_model(model_path, [tensor_out_name])

    # create uff parser
    parser = trt.UffParser()
    parser.register_input(tensor_in_name, input_size)
    parser.register_output(tensor_out_name)

    # create trt logger and builder
    trt_logger = trt.Logger(trt.Logger.INFO)
    builder = trt.Builder(trt_logger)
    builder.max_batch_size = max_batch_size
    builder.max_workspace_size = max_workspace
    builder.fp16_mode = (data_type == trt.float16)

    # parse the uff model to trt builder
    network = builder.create_network()
    parser.parse_buffer(uff_model, network)

    # build optimized inference engine
    engine = builder.build_cuda_engine(network)

    # save inference engine
    with open(output_path, "wb") as f:
        f.write(engine.serialize())
```

```python
import tensorflow as tf
import tensorrt as trt
import frozen_pb_to_plan

BATCH_SIZE = 1
H, W, C = 299, 299, 3

if __name__ == "__main__":
    '''
    generate the inference engine
    '''
    pb_model_path = "../output/mask.pb"
    plan_model_path = "../output/mask_float32.plan"
    input_node_name = "input_1"
    output_node_name = "predictions/Softmax"

    frozen_pb_to_plan(pb_model_path,
                      plan_model_path,
                      input_node_name,
                      output_node_name,
                      [C, H, W],
                      data_type=trt.float32, # change this for different TRT precision
                      max_batch_size=1,
                      max_workspace=1<<30)
```

הסבר:

משתנים עיקריים:

- model_in_path - מחרוזת נתיב מודל הקלט מסוג h5
- model_out_path - מחרוזת נתיב מודל הפלט מסוג pb
- graph - גרף המודל המקורי המוקפא שנוצר באמצעות מתודה של Tensorflow
- model - המודל המקורי שעולה אל הגרף המוקפא
- graph_def - גרף המודל המשני המוקפא שנוצר באמצעות מתודה של Tensorflow

פונקציות:

- keras_to_frozen_pb - מקבל את נתיב המודל מסוג h5, את נתיב השמירה של מודל היצוא pb, שם למודל הטנסור(אופציונלי) ונתיב לייצוא גרף המודל(אופציונלי).
- frozen_pb_to_plan - מקבל את נתיה המודל מסוג pb, את נתיב השמירה של מודל היצוא plan, שם החוליה של הקלט, שם החוליה של הפלט, גודל הקלט, סוג הדאטא(אופציונלי), גודל ה-batch המקסימלי(אופציונלי), מרחב העבודה המקסימלי(אופציונלי) ונתיב לייצוא גרף המודל(אופציונלי).

- main - הפעלת הפונקציות השונות לפי סדר הטרנספורמציה (קודם לסוג frozen_pb ואז לסוג plan)

convert_Keras_model_to_Tensorflow.ipynb1

הקובץ מכיל .

מיקומים:

- Camerona\preperations\model training\temsorRT\using frozenpb\1_convert_Keras_model_to_Tensorflow.ipynb

תוכן:

```
# import the needed libraries
import tensorflow as tf
tf.keras.backend.set_learning_phase(0) #use this if we have batch norm layer in our network
from tensorflow.keras.models import load_model

# path we wanna save our converted TF-model
#MODEL_PATH = "./model/tensorflow/big/model1"
MODEL_PATH = "./model/tensorflow/small/model_small"

# load the Keras model
#model = load_model('./model/modelLeNet5.h5')
model = load_model('./model/modelLeNet5_small.h5')

# save the model to Tensorflow model
saver = tf.train.Saver()
sess = tf.keras.backend.get_session()
save_path = saver.save(sess, MODEL_PATH)

print("Keras model is successfully converted to TF model in "+MODEL_PATH)
```

הסבר:

משתנים עיקריים:

- MODEL_PATH - מחרוזת נתיב מודל הפלט מסוג trt
- model - המודל המקורי מסוג h5
- save_path - מפעיל את מתודת השמירה המובנית של Tensorflow שכוללת את נתיב השמירה ואת הסשן שנפתח בעזרת keras

פונקציות:

- main - הפעלת המתודות האחריות על שמירת המודל בתור trt.

2_convert_TF_to_TRT.ipynb

הקובץ מכיל .

מיקומים:

- Camerona\preperations\model training\temsorRT\using tensorTRT\2_convert_TF_to_TRT.ipynb

תוכן:

```
#section 1
# import the needed libraries
import tensorflow as tf
import tensorflow.contrib.tensorrt as trt
from tensorflow.python.platform import gfile


# has to be use this setting to make a session for TensorRT optimization
with
tf.Session(config=tf.ConfigProto(gpu_options=tf.GPUOptions(per_process_gpu_memory_fract
ion=0.50))) as sess:
    # import the meta graph of the tensorflow model
    #saver = tf.train.import_meta_graph("./model/tensorflow/big/model1.meta")
    saver = tf.train.import_meta_graph("./model/tensorflow/small/model_small.meta")
    # then, restore the weights to the meta graph
    #saver.restore(sess, "./model/tensorflow/big/model1")
    saver.restore(sess, "./model/tensorflow/small/model_small")

    # specify which tensor output you want to obtain
    # (correspond to prediction result)
    your_outputs = ["output_tensor/Softmax"]

    # convert to frozen model
    frozen_graph = tf.graph_util.convert_variables_to_constants(
        sess, # session
        tf.get_default_graph().as_graph_def(),# graph+weight from the session
        output_node_names=your_outputs)
    #write the TensorRT model to be used later for inference
    with gfile.FastGFile("./model/frozen_model.pb", 'wb') as f:
        f.write(frozen_graph.SerializeToString())
    print("Frozen model is successfully stored!")

#section 2

# convert (optimize) frozen model to TensorRT model
```

```
trt_graph = trt.create_inference_graph(
input_graph_def=frozen_graph,# frozen model
outputs=your_outputs,
max_batch_size=2,# specify your max batch size
max_workspace_size_bytes=2*(10**9),# specify the max workspace precision_mode="FP32") #
precision, can be "FP32" (32 floating point precision) or "FP16"

#write the TensorRT model to be used later for inference with
gfile.FastGFile("./model/TensorRT_model.pb", 'wb') as f:
f.write(trt_graph.SerializeToString())
print("TensorRT model is successfully stored!")
```

הסבר:

משתנים עיקריים:

- frozen_graph - הגרף המוקפא של המודל המקורי.
- saver - נועד לשמירת פרטי ה- meta של הגרף ששייך למודל המקורי.
- trt_graph - גרף המודל מסוג trt אליו מיוצא גרף המודל המקורי.

פונקציות:

- main - הפעלת המתודות האחריות על שמירת המודל בתור trt.

6. סיכום אישי \ רפלקציה

העבודה על הפרויקט הייתה קשה וארוכה אך ממלאת מאוד מבחינה נפשית, אישית, אני מאוד אוהב אתגרים בתחומים שמעניינים אותי, בינה מלאכותית הוא נושא שמעניין אותי מאוד ככה שגם עם כל האתגרים הקשים, הזמן וההשקעה שהפרויקט גזל ממני, עדיין נהניתי מהעבודה בגלל שזה מילא אותי בנפש, עשיתי במהלך כל הזמן הזה משהו שאני נהנה ממנו ומסקרן אותי.

מהעבודה על הפרויקט קיבלתי כלים וידע רב, מרבית הידע קשור בבינה מלאכותית, כיצד היא עובדת, איזה סוגי למידה יש, כיצד משתמשים בהם, מתי משתמשים בהם, חלק מן המתמטיקה מאחורי רשתות נוירונים ועוד, בנוסף לידע הרב שצברתי והשגתי במהלך העבודה, למדתי לעבוד עם מספר כלים כאשר העיקרי מביניהם הוא עבודה עם עיבוד בענן. עיבוד באמצעות ענן הוא שימוש בשרתים חיצוניים של חברה מסויימת לצורך הרצת קוד, לוקחים קוד פייתון בפורמט מסויים ומעלים לשרת, ברגע שמריצים השרת מתחיל תהליכי עיבוד לפי מה שמבקשים. במהלך שימוש בשרת לצורך הרצת הקוד ניתן גם לבחור רכיבים פיזיים שיהיו לשרת כמו מעבד גרפי, הדבר הועיל מאוד בתהליך האימון של המודלים כיוון שבעזרת שימוש בכרטיסי עיבוד גרפי האימון מתבצע מהר יותר.

הכלי העיקרי שאותו אני לוקח איתי הוא הכלי של שימוש בשרת חיצוני לעיבוד קוד, אני מאמין שיהיו לי שימושים רבים בכלי זה בגלל היתרונות הרבים שלו.

העבודה על הפרויקט טמנה בחובה אתגרים רבים, חלקם היו קשורים בניהול הזמן והעבודה וחלקם היו קשורים בלמידה וכתיבת קוד. היה עליי לנהל את הזמן שלי בצורה טובה בשביל שאוכל להספיק את כל מה שאני רוצה לעשות, בנוסף לכך, היה עליי לחקור וללמוד הרבה בשביל להבין כיצד לעשות דברים בפרויקט, היה עליי להבין יותר טוב כיצד לממש ולהשתמש ברשתות נוירונים, איך לעבוד עם מידע ועוד המון דברים קטנים שביחד הצטברו לאתגרים גדולים ורציניים שדרשו ממני הרבה.

ברצוני לציין שאני בהחלט אמשיך לעבוד ולהשתמש בטכנולוגיית הבינה המלאכותית ושעבודה על פרויקט המערב בינה מלאכותית הייתה החלטה נכונה מאוד. למדתי, חקרתי וגיליתי דברים רבים ומעניינים שאני בהחלט ישתמש בהם בעתיד.

חייב להבהיר שאינני חושב שהייתי משנה דבר בתהליך, העבודה על הפרויקט הייתה מאוד יעילה ונכונה, הסיבה העיקרית שלשמה בחרתי בפרויקט הייתה סיבה מאוד מוצלחת וטובה, כל העיצוב, החלוקה והעבודה היו גם הם מאוד מוצלחים ובאופן כללי מאוד נהניתי מהעבודה על הפרויקט ככה שאני לא חושב שיש משהו שהייתי עושה שונה.

התוצר הסופי הוא אכן מה שתכננתי להשיג מהעבודה על הפרויקט, אמנם קיים פער קל בין התכנון המקורי לתוצר הסופי, בהתחלה תכננו שיהיה לנו מודל שמבצע face detection שנכתוב ונאמן לבד, בסוף לא הצלחנו לעשות את זה בגלל שלא היה לנו זמן וידע מספיק, למרות זאת, הספקנו לעשות משהו אחר שלא היה בתכנון המקורי, בספרינט האחרון הוספנו ייעול של הקוד ושל המודלים, השתמשנו בספרייה Tensorflow Lite אשר הופכת את המודלים לקלים ומהירים יותר, לדבר הייתה השפעה עצומה על התוצר הסופי כיוון שזמן העיבוד והמהירות שבה הקוד רץ הרבה יותר טובים.

לסיכום, העבודה על הפרויקט הייתה מאתגרת ומהנה בו בעת, למדתי הרבה והשגתי הרבה כלים, אני מאמין שהרווחתי הרבה מן העבודה על הפרויקט הזה ושתהיה לו השפעה עליי ועל עתידי כמתכנת בעתיד.

7. משוב

במהלך הפרויקט היה עליי ללמוד ולהתגבר על אתגרים רבים, הן בלימודים והן בכל הקשור לתכנון זמן. היה עליי לתכנן את הזמן שלי בצורה מיטבית וללמוד בצורה אינטנסיבית ורצינית. הידע והעניין שהפרויקט הוסיף לי היו רבים, תקופה ארוכה שהתעניינתי בטכנולוגיית הבינה המלאכותית והפרויקט נתן לי הזדמנות נהדרת לחקור, להתנסות ולהשתמש בטכנולוגיה זו. המטרה שלי לפרויקט זה היה להתנסות בטכנולוגיית הבינה המלאכותית בצורה עצמאית ומלאה, לאורך כל הפרויקט זה מה שעשיתי, היה עליי לחקור וללמוד רבות מעבר לידע שהיה לי ולהשתמש בידע הזה בשביל ליצור בינות מלאכותיות ולהשתמש בהן, אני מאמין שהשגתי את מטרותיי. מעבר לכך אני מאמין שהפרויקט הפך אותי לתכנת יותר טוב בגלל שבמהלך הפרויקט נתקלתי בבעיות רבות והצלחתי להתגבר עליהן, הדבר העשיר וחיזק אותי בתור מתכנת מכיוון שהפתרון דרש ממני מחקר, חשיבה יצירתית והבנה מעמיקה, שאלו, לדעתי, תכונות עיקריות אצל מתכנת טוב.

8. קוד הפרויקט

כל תיקייה הנקראת models מכילה אך ורק קבצי מודלים.

תיקייה: Camerona\keras
התיקייה מכילה את החלק של הפרויקט שבו יש חלון שמראה את ניחושי המודלים כאשר המודלים הם מודלים מסוג tensorflow ללא אופטימיזציה, כל הקבצים והתיקיות בתוכה קשורים למטרה זו.

תיקייה: Camerona\tf-lite
התיקייה מכילה את החלק של הפרויקט שבו יש חלון שמראה את ניחושי המודלים כאשר המודלים הם מודלים מסוג tensorflow lite, כל הקבצים והתיקיות בתוכה קשורים למטרה זו.

תיקייה: Camerona\speakers
התיקייה מכילה את החלק הסופי של הפרויקט, הקוד משתמש במודלים מסוג tensorflow lite בשביל לחלץ מידע בנוגע לפרצופים שנמצאו בתמונה ואז פונה אליהם באמצעות הקלטות ורמקולים.

תיקייה: Camerona\preperations
התיקייה מכילה את כל הקבצים והתיקיות הקשורות לכל מה שהיה עלינו לבצע בשביל להגיע לתוצר הסופי של הפרויקט (קבצי אימונים, עיבוד מידע, אופטימיזציה וכדומה).

תיקייה: Camerona\preperation\audio
התיקייה מכילה את כל הקבצים הקשורים לרמקולים, הדבר כולל את קבצי השמע ואת הקובץ בו השתמשתי בשביל ליצור אותם.

תיקייה: Camerona\preperation\model training
התיקייה מכילה את כל מה שקשור למודלים והכנתם (אימונים, אופטימיזציה כלים של ה database).

תיקייה: Camerona\preperation\model training\dataset tools
התיקייה מכילה את כל הקודים שיצרתי בשביל להקל על יצירת dataset.

תיקייה: Camerona\preperation\model training\tf-lite
התיקייה מכילה את כל הקבצים הקשורים לשינוי של המודלים מ tensorflow ל tensorflow lite.

תיקייה: Camerona\preperation\model training\tf-lite\check accuracy
התיקייה מכילה את הקבצים בהם השתמשנו בשביל לבדוק את מידת הדיוק של המודלים שעברו אופטימיזציה מ tensorflow ל tensorflow lite.

תיקייה: Camerona\preperation\model training\tensorRT
התיקייה מכילה את כל הקבצים הקשורים לאופטימיזציה של TensorRT.

תיקייה: Camerona\preperation\model training\tensorRT\using frozenpb
התיקייה מכילה קבצים הקשורים לצורה מסוימת של אופטימיזציה בעזרת TensorRT.

תיקייה: Camerona\preperation\model training\tensorRT\using tensorTRT
התיקייה מכילה קבצים הקשורים לצורה מסוימת של אופטימיזציה בעזרת TensorRT.

תיקייה: Camerona\preperation\model training\keras\beard classification
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון וההכנה של מודל הזקן.

תיקייה: Camerona\preperation\model training\keras\beard classification\training
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון של מודל הזקן.

תיקייה: Camerona\preperation\model training\keras\beard classification\training\training results
התיקייה מכילה את כל הקבצים הקשורים לתיעוד האימון של המודל ותוצאותיו.

תיקייה: Camerona\preperation\model training\keras\gender classification
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון וההכנה של מודל הזקן.

תיקייה: Camerona\preperation\model training\keras\gender classification\training
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון של מודל הזקן.

תיקייה: Camerona\preperation\model training\keras\gender classification\training\training results
התיקייה מכילה את כל הקבצים הקשורים לתיעוד האימון של המודל ותוצאותיו.

תיקייה: Camerona\preperation\model training\keras\glass classification
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון וההכנה של מודל הזקן.

תיקייה: Camerona\preperation\model training\keras\glass classification\training
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון של מודל הזקן.

תיקייה: Camerona\preperation\model training\keras\glass classification\training\training results
התיקייה מכילה את כל הקבצים הקשורים לתיעוד האימון של המודל ותוצאותיו.

תיקייה: Camerona\preperation\model training\keras\mask classification
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון וההכנה של מודל הזקן.

תיקייה: Camerona\preperation\model training\keras\mask classification\training
התיקייה מכילה את כל הקבצים הקשורים לתהליך האימון של מודל הזקן.

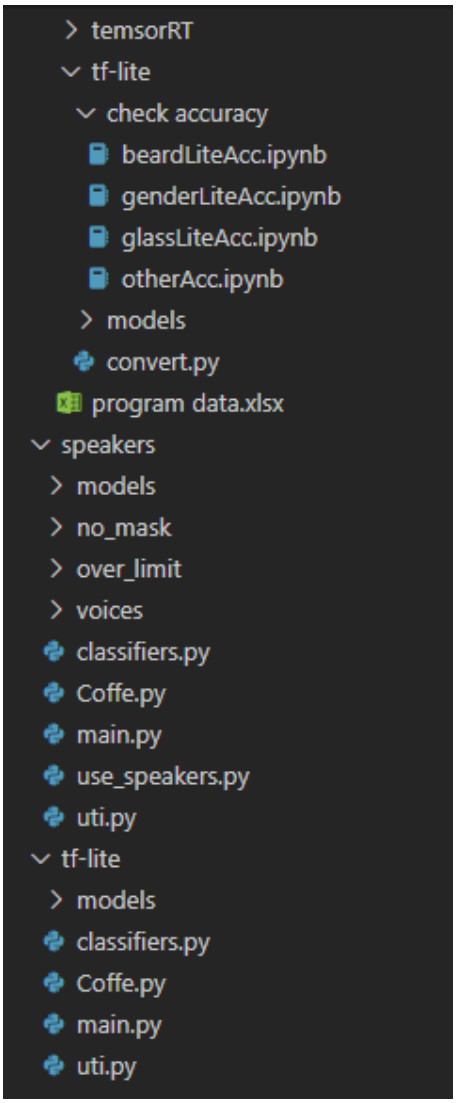
תיקייה: Camerona\preperation\model training\keras\mask classification\training\training results
התיקייה מכילה את כל הקבצים הקשורים לתיעוד האימון של המודל ותוצאותיו.

תיקייה: Camerona\preperation\model training\keras\face detection
התיקייה מכילה את כל הקבצים הקשורים למודל שמוצא פנים בתמונה.

תיקייה: Camerona\preperation\model training\keras\face detection\opencv
התיקייה מכילה את כל הקבצים הקשורים למודל שמוצא פנים בתמונה הממומש על ידי opencv.

תיקייה: Camerona\preperation\model training\keras\face detection\training
התיקייה מכילה את כל הקבצים הקשורים למודל שמוצא פנים בתמונה שניסינו לאמן ולבנות לבד.


keras
models
classifiers.py
Coffe.py
main.py
uti.py
preperations
audio
voices
buildVoice.py
model training
dataset tools
keras
beard classification
models
training
training results
beard_classification.ipynb
beard_classification.py
beardModel.png
face detection
opencv
models
Coffe.py
training
SSD.ipynb
gender classification
gender classification
models
training
training results
gender_detection.ipynb
gender_classification.py
genderModel.png
glass classification
models
training
after training
amounts.txt
glass_classi.ipynb
glass_classification.py
glassModel.png
mask classification
models
training
training-resaults
mask_classi.ipynb
mask_classification.py
maskModel.png
temsorRT
tf-lite
check accuracy
beardLiteAcc.ipynb
genderLiteAcc.ipynb


temsorRT
tf-lite
check accuracy
beardLiteAcc.ipynb
genderLiteAcc.ipynb
glassLiteAcc.ipynb
otherAcc.ipynb
models
convert.py
program data.xlsx
speakers
models
no_mask
over_limit
voices
classifiers.py
Coffe.py
main.py
use_speakers.py
uti.py
tf-lite
models
classifiers.py
Coffe.py
main.py
uti.py

9. רשימת מקורות (ביבליוגרפיה)

ספר שממנו למדתי את רוב נושא הבינה המלאכותית.

https://youtu.be/5fHngyN8Qhw
https://youtu.be/bNntsCOdFxg
https://youtu.be/tPYj3fFJGjk
https://youtu.be/aircAruvnKk
https://youtu.be/ukzFI9rgwfU
https://youtu.be/5fHngyN8Qhw
https://youtu.be/tPYj3fFJGjk
https://youtu.be/bNntsCOdFxg
https://towardsdatascience.com/covid-19-face-mask-detection-using-tensorflow-and-opencv-702dd833515b
https://developer.nvidia.com/blog/speeding-up-deep-learning-inference-using-tensorflow-onnx-and-tensorrt/
https://github.com/onnx/onnx-tensorrt
https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html#working_tf
https://github.com/onnx/tensorflow-onnx
https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html