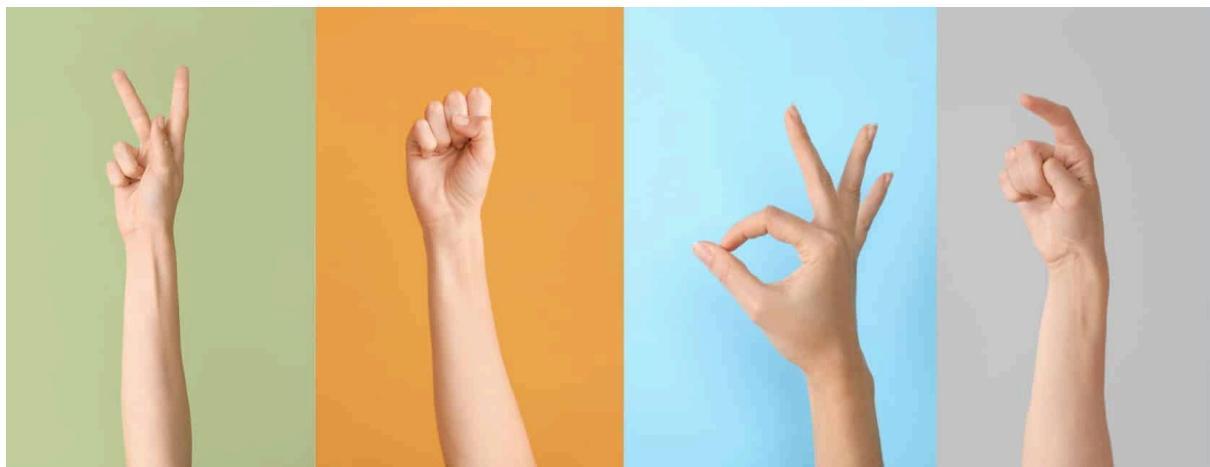




תיק פרוייקט – התמחות בהנדסת תוכנה – למידת מכונה



זיהוי שפט הסימנים הישראליות

שם מלא: ניב כופי
מספר תעודה זהות: 216446146
שם המנחה: שי פרח
שם החלופה: "למידת מכונה"
תאריך הגשתה: 15.05.2024

תוכן העניינים

תוכן העניינים.....	2
תקציר מנהלים.....	3
מבוא.....	3
הרקע לפרוייקט.....	3
מטרת הפרויקט ותיאור אופן הפעולה העתידי.....	4
קהל היעד.....	4
סיבות לבחירת הנושא.....	4
תהליכי הממחקר.....	5
אתגרים מרכזים.....	5
מצגת פתרונות לבעה.....	5
מבנה וארכיטקטורה.....	6
איסוף, הכנה וניתוח הנתונים.....	6
תיאור וניתוח הנתונים הגלומיים, התפלגיות, דוגמאות.....	7
נירמול, augmentation, ניקוי, סינון - dataset.....	7
שלב בניה ואמון המודל.....	8
תיאור גרפי של המודל:.....	8
הסבר על סוג השכבות השונות ברשות:.....	8
תהליכי האימון של המודל:.....	9
תוצאות שלב האימון:.....	10
טיזוב תוצאות האימון:.....	10
סיכון והמשר:.....	11
שלב היישום.....	15
תיאור והסביר כיצד היישום משתמש במודל:.....	15
תיאור הטכנולוגיה שעלה פיה מושך המשמש:.....	16
שלבי ייצוא החיזוי והתאמתו למבנה נתונים המתאים לחיזוי: DATA-TEXT-הקוד הקולט את ה.....	16
מדריך למפתח.....	18
מדריך למשתמש.....	26
סיכון אישי ורפלקציה.....	35
ביבליוגרפיה.....	37
נספחים.....	39

סרטון המראה של הפרויקט: <https://www.youtube.com/watch?v=qMNtx-AIB-4>

תקציר מנהליים

פרויקט זה מציג מערכת לזיהוי של שפט הסימנים הישראלית (LSI) בזמן אמת, באמצעות טכניקות למידה عمוקה מתקדמות. המבוססת על מודל YOLOv5 מותאמת אישית, מסוגלת לזהות חמישה סימנים בסיסיים של LSI ("שלום", "תודה", "כן", "לא" ו"אני אוהב אותך") בדיק ממוצע של 92% (mAP@0.5).

המטריצה לפרויקט נובעת מה צורך לגשר על הפער התקשורתי בין האוכלוסייה השומעת לבין קהילת החירשים וכבדי השמיעה בישראל, המונה מעל 750,000 איש. על ידי פיתוח כל-nageš לתרגום שפט סימנים לטקסט בזמן אמת, המערכת שואפת לקדם את השילוב והשוון של כבדי השמיעה במרחבם חברתיים, אקדמיים ותעסוקתיים.

התהילה כלל איסוף ותיקוג של מאגר נתונים מקורי כולל 100 תמונות (20 לכל סימן), אימון והערכתה של המודל תוך כיוון של היפר-פרמטרים ומבנה הרשת, ולבסוף - יישום של המודל באפליקציית ווב אינטראקטיבית לזיהוי סימנים בזמן מול מצלמת המחשב.

זהו הפרויקט הראשון מסוגו עבור שפט הסימנים הישראלית הספרטיפית.

לצד ההישגים המרשימים, הפרויקט מהווה גם הוכחת היתכנות לפיתוחים עתידיים - כגון הרחבה למילון מלא של LSI, תמייה בזכחי משפטים שלמים, ושילוב בטכנולוגיות תרגום אוטומטי מtekst לשפט סימנים.

המסקנה העיקרית העולה מהפרויקט היא שטכנולוגיות בינה מלאכותית מתקדמות, בשילוב עם מאגרי נתונים איצטדיים, מאפשרות פריצות דרך משמעותיות בהנגשת התקשורות עבור אוכלוסיות מודדות. אני מאמין שהמערכת המוצגת כאן מהווה צעד חשוב בכיוון של עתיד תקשורת מכיל ונגיש יותר.

מבוא

הרקע לפרויקט

בעשורים האחרונים, העולם עבר מהפכה טכנולוגית מרשימה, כאשר מחשבים ובינה מלאכותית הופכים לחלק בלתי נפרד מח' היומיום שלנו. תחום הלמידה העמוקה (Deep Learning), תחת-תחום של למידת מכונה, תפס תאוצה משמעותית בשנים האחרונות והביא לפריצות דרך במגוון רחב של יישומים - מזיהוי דיבור ועיבוד שפה טבעית, דרך ניתוח תמונות רפואיות ועד לרכיבים אוטונומיים.

אחד התחומיים המרתקים שבhem למדעה عمוקה יכולה לתרום רבות הוא תחום הנגשנות, ובפרט תקשורת עם אנשים בעלי מוגבלות. כאן נכנסה לתמונה שפט הסימנים - שפה ויזואלית המשמשת כאמצעי תקשורת עיקרי עבור קהילת החירשים וכבדי השמיעה ברחבי העולם. שפט הסימנים מבוססת על שילוב של תנועות ידיים, הבעות פנים ותנועות גוף כדי לייצג מילים ומושגים. ישן מעל 300 שפות סימנים שונות בעולם, כאשר בישראל נפוצה שפט הסימנים הישראלית (LSI). על פי הערכות, ישן כ-50,000 אנשים כבדי שמיעה בישראל, מתוכם כ-10,000 דוברי שפט הסימנים כשפט אם.

על אף התקדמותה במודעות ובתמיינאה לקהילה זו, עדין קיים פער משמעותי בין הקהילה השומעת לבין קהילת כבדי השמיעה, בעיקר סביר נגישות שירותים ותקשורת יומיומית. מחסומי השפה מקשים על אנשים כבדי שמיעה להשתלב במערכות החינוך, בעולם התעסוקה ובמרחב הציבורי הרחב.

טכנולוגיה זו יכולה לשיער במצומם הפערים ובקידום השווין. בשנים האחרונות פותחו טכנולוגיות מסייעות כמו יישומי תרגום לכ窈ויות בזמן אמת או התראות ויזואליות. אך ככל הנוגע לשפט הסימנים עצמה, עדין יש צורך במתורגמים אנושיים לטיור בין הסימנים לטקסט או דבר. מערכת לזיהוי אוטומטי של שפט סימנים יכולה לשנות מציאות זו ולאפשר תקשורת חלקה יותר.

מטרת הפרויקט ותיאור אופן הפעולה העתידי

מטרת הפרויקט היא לפתח מערכת המסוגלת לזהות 5 סימנים בסיסיים בשפט הסימנים הישראלי (שלום, תודה, כן, לא, אני אוהב אותך) מתוך תמונות וסרטונים, עם חיזוי מדויק של סוג הסימן ומיקומו בפריטים.

הפרויקט כולל שני חלקים עיקריים - שלב הלמידה והאימון, ושלב היישום והשימוש. בשלב הראשון, בניית מערכת מבוססת רשתות ניירונים عمוקות לזהות את הסימנים, תוך שימוש במאגר נתונים גדול של תמונות מתויות. בשלב השני, המודול שאומן משולב ביישום קצה ייחודי למשתמש, המאפשר למשתמשים לראות את עצמם בעדרת מצלמת הרשות שלהם ולקבל בזמן אמת את הפענוח של הסימנים המוצגים. אופן הפעולה העתידי יכול שיפור ביצועי המודול והרחבת מאגר הנתונים, עד כדי שימוש בוואנו יכול לקבל תרגום מלא ומדויק של כל שפט הסימנים הישראלי.

קהל היעד

בין קהלי היעד האפשריים: כבדי שמיעה וחירשים שיוכלו להשתמש בתקשורת ביישום לתקשרות עם הסביבה השומעת, בני משפחה וחברים שירצו ללמידה את שפט הסימנים, מורים שיוכלו להשתמש בכלים לחומר לימוד, אנשי מקצוע שירצו להנגיש את עצמם לאוכלוסייה כבדי השמיעה, חוקרים בתחום שפות הסימנים ועיבוד שפה טבעיות, ומפתחים שירצו להטמע את יכולות הזיהוי במערכות חדשות.

סיבות לבחירת הנושא

אני כבדי שמיעה בעצמי ואין דבר שהייתי רוצה יותר מלאחד בין שתי הקהילות השונות אך הקרובות שאני חלק מהן. זה נובע מהרצן ליישם את הדעת והכישורים שנרכשו לטובת פתרון בעיות אמיתיות ויצירת ערך חברתי, מהאתגר הטכני שבשימוש בלמידה عمוקה לניתוח תמונה והבנת שפה, ומההשראה שעורר המפגש עם עולם שפות הסימנים והתרבות של הקהילה כבדת השמיעה.

תהליך המבחן

תהליך הפיתוח דרש עבודה עמוקה מהייקה הן בהיבטים של למידת מכונה והן מצד היישומי. בוצע מחקר על המצב הנוכחי בשוק ונמצא מילון לשפט הסימנים הישראלית שפותח על ידי אדם פרטני ולא התעדכן כבר מספר שנים. שאר הדברים הם הדרכות בתשלום למיניהם. לא נמצא משהו דומה למה שהפרויקט הנ"ל מציע ובטע שלא לגבי החזון העתידי שלו. لكن, החידושים שהפרויקט מציע הם רבים, יצירת מודל מאומן על שפט הסימנים הישראלי זה משווה שעוז לא נראה כמותו בארץ, עם זאת פרויקטים דומים קיימים לשפט סימנים זרות. כמתבקש, שבתי השראה מפרויקטים אחדים הקיימים בחוץ המושגים מטרה דומה לזו שמטרת הפרויקט שואפת אליה, כך היה בסיס איתן שיכולתי להישען עליו וגם בשבייל היכולת לדמיין כיצד הפרויקט יראה בסופו. הבנתי שאין לי בסיס נתונים קיימים שביכולתי להשתמש ולכן יצרתי את מודל הנתונים שלי בעצמי וזה גם הסיבה למה הוא יחסית מצומצם.

אתגרים מרכזיים

האתגרים העיקריים כוללים איסוף מספיק דוגמאות לכל סימן לצורך אימון, השגת ביצועים מהירים מספיק עבור יישום בזמן אמת, והתמודדות עם וריאbilיות במרקם ובזווית בין אנשים שונים המבצעים את אותם סימנים. הפרויקט מציע פתרון לצורך של תקשורת נגישה יותר עבור כבדי שימושה וחירשים.

הציג פתרונות לבעה

לאחר בוחנת פתרונות שונים במסגרת המבחן המקדים, נבחרה ארכיטקטורה של רשת נוירונים מבוססת $5\text{V}\text{SoC}$, המתאימה במיוחד לשימוש זיהוי אובייקטים.

בהמשך העבודה יפורטו לעומק שלבי הפרויקט, ארכיטקטורת המערכת ותהליכי האימון, תוך הדגמת השימוש בישום ומtran מדריך טכני לפיתוח עתידי. מטרת התיעוד היא לאפשר המשך פיתוח ושיפור של המערכת, כדי שתוכל לתרום משמעותית לשיפור התקשרות והנגישות עבור קהילת כבדי השימוש והחירשים.

בפרק הבאים/atrar בפירוט את מבנה הפרויקט, את תהליכי פיתוח המודל ואת היישום הסופי תוך מתן הסברים טכניים על הקוד והטכנולוגיות בהם השתמשתי. בסיום של התהליך אציג את התובנות והמסקנות העיקריות, ואתן כיווני מחשבה להמשך פיתוח ושיפור של המערכת בעתיד.

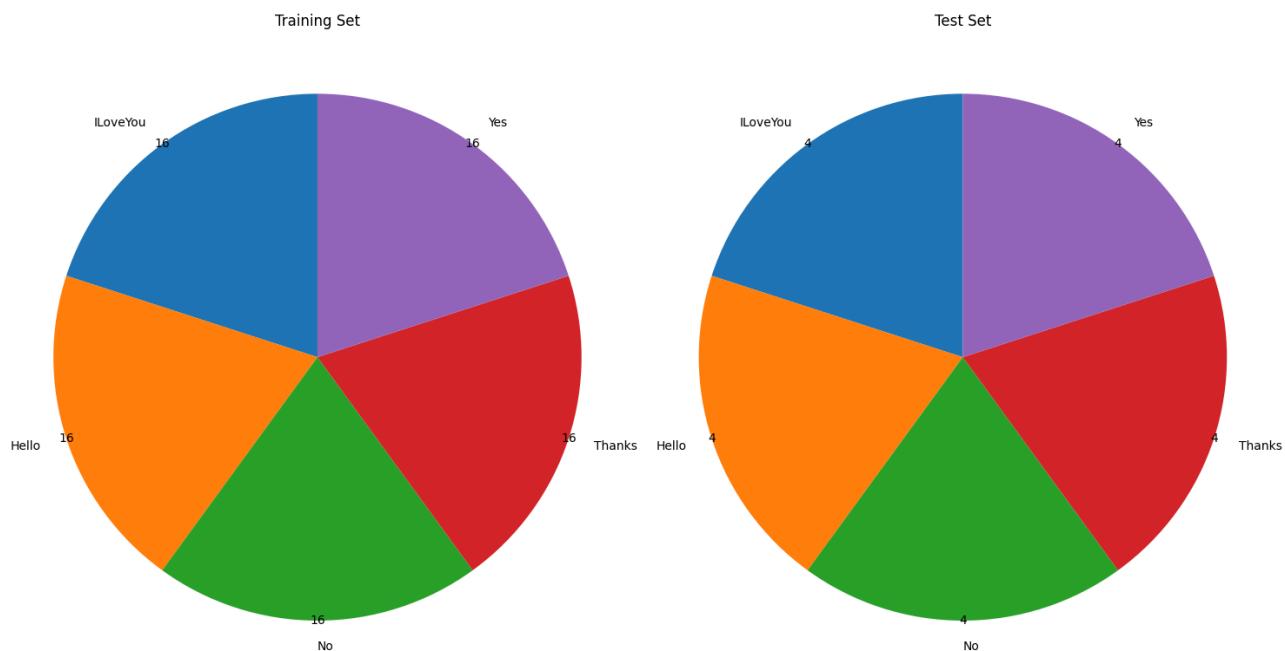
מבנה וארכיטקטורה

איסוף, הכנה וניתוח הנתונים

בפרויקט זה, השתמשתי במאגר נתונים עצמי שיצרתי על ידי איסוף תמונות של סימנים בשפת הסימנים הישראלית. בהתאם איסוף הנתונים כלל את השלבים הבאים:

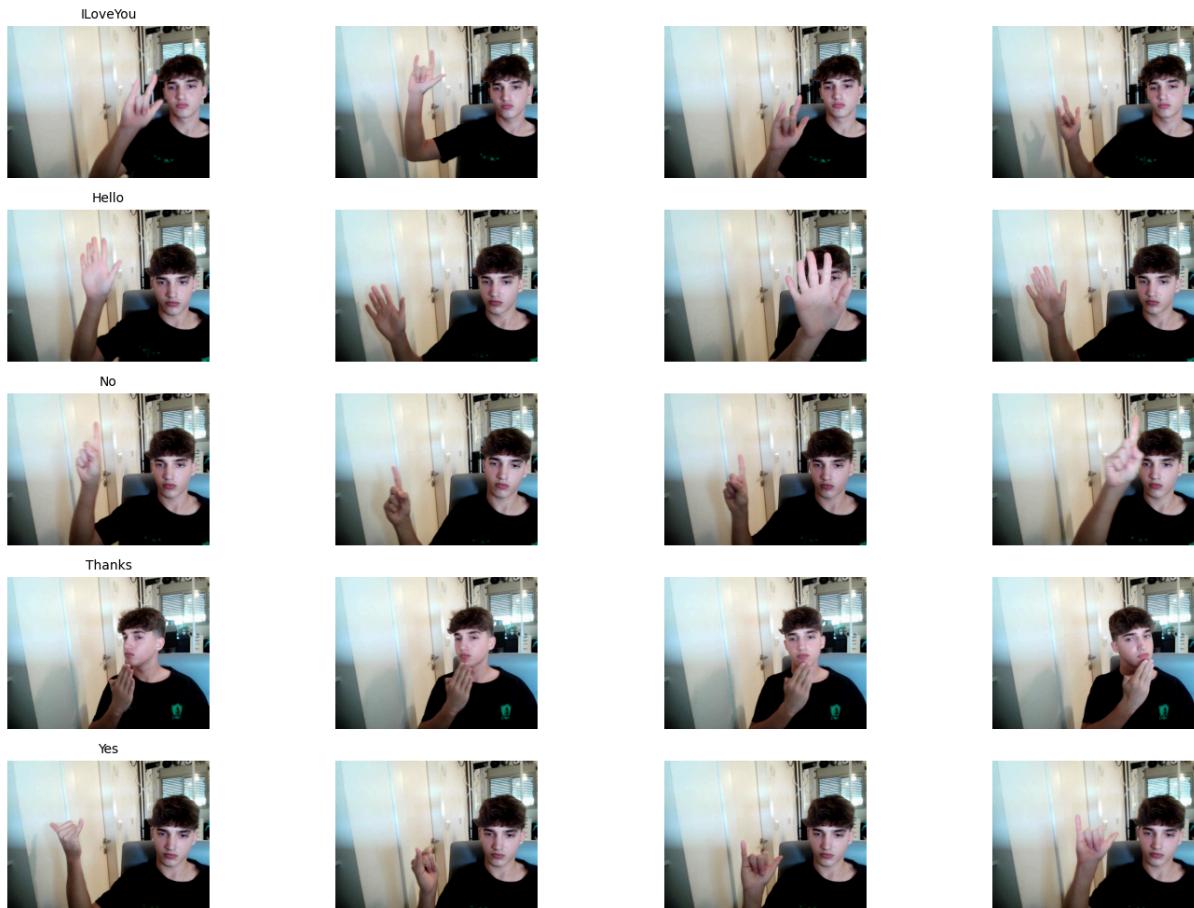
1. צילום של 20 תמונות עבור כל אחת מ-5 קטגוריות של מילים נפוצות בשפת הסימנים הישראלית: תודה, כן, לא, אני אוהב אותך ושלום. סך הכל, המאגר מכיל 100 תמונות (20 תמונות לכל קטgorיה).
2. הצלום ה被执行 באופן אוטומטי באמצעות תוכנה שפיתחתי, אשר מנצלת את מצלמת המחשב. עבור כל מילה, התוכנה מעתינה 5 שניות ועוד מצלמת 20 תמונות בקצב כל 2 שניות, תוך הצגת פריים הצלום על המסך.
3. לאחר הצלום, ביצעת תיוג ידני של התמונות באמצעות התוכנה `labelImg`. סימנתי את גבולות התביעה (bounding box) עבור כל אובייקט (היד המסתמנת) בכל תמונה. בהתאם לכך ניתן למקד את תשומת הלב של מודל הלמידה העומקה באזורי הרלוונטיים של כל תמונה.

לסיכום, סט הנתונים מכיל סך הכל 100 דוגמאות, 20 מכל אחד מ-5 סוגי הסימנים. סט האימון מכיל 80% מהדוגמאות (16 מכל סוג), וסט הבדיקה מכיל את ה-20% הנדרדים (4 מכל סוג).



תיאור וניתוח הנתונים הגלמיים, התפלגיות, דוגמאות

מבנה מאגר הנתונים שנוצר הוא תיקייה ראשית בשם CollectedImages, המחולקת לתיקיות משנה לפי שמות חמשת הקטגוריות. בכל תיקייה קטgorיה נמצאים 20 קבצי התמונות (בפורמט jpg) השייכים אותה קטgorיה, יחד עם קבצי התיאוג שלהם.



תהליך הכתת ה dataset - ניקוי, סינון, augmentation, נירמול

ניתוח הנתונים הגלמיים מראה שמדובר במאגר נתונים ראשוני בסיסי יחסית, המכיל סה"כ 100 תמונות עבור 5 מחלקות בודדות של סימנים. עם זאת, התמונות צולמו בנפרד עבור כל קטgorיה, וכך קיים גיוון מסוים בზוויות הצילום וב הבעות הפנים (למרות שהצלום נעשה על רקע אחיד ובתנאי תאורה קבועים ע"י אותו אדם).

מדובר בגודל מאגר מצומצם, שבמהלט ניתן להרחבו בהמשך על ידי הוספה דוגמאות לסימנים ומילימ נסיפות. כדי גם לגוון את תנאי הצילום (רקע, זווית, תאורה) ולצלם אנשים שונים מסמנים, על מנת לאפשר למודל הלמידה העמוקה ללמידה מייצוג נרחב יותר של הסימנים ולשפר את יכולת ההכללה שלו לmarker מגוונים יותר בעתיד.

שלב בניה וAIMON המודל

לאחר איסוף מאגר הנתונים המתואג, ניגשתי לשלב בניה וAIMON מודל הלמידה העמוקה לזיהוי סימני שפט הסימנים. הארכיטקטורה של המודל מבוססת על רשות ניירונים מסווג (You Only Look Once – YOLO), שמתאימה במיוחד למשימות של זיהוי אובייקטים בתמונות.

תיאור גרפי של המודל:

היא ארכיטקטורת רשות ניירונים עמוקה פופולרית לזיהוי אובייקטים בזמן אמיתי. YOLOv5 היא הגרסה המשופרת של מודל הבסיס, שפותחה על ידי Ultralytics. בהשוואה לגרסאות קודמות ולארכיטקטורות אחרות, YOLOv5 מציעה איזון מצוין בין מהירות לדיק, מה שהופך אותה לבחירה מועדפת עבור יישומי ראייה ממוחשבת בזמן אמיתי.

רשות מותאמת ל蹶ה שלנו באמצעות קובץ הkonfigורציה `yaml`. בקובץ זה הגדרתי פרמטרים כגון מספר המחלקות (5), עומק וגודל השכבות ועוד, בהתאם לבנייה מאגר הנתונים ולמשימה הספציפית.

התרשימים הגרפיים של המודל גדולים מאוד וכן מצורף בנספחים.

הסבר על סוגי השכבות השונות ברשות:

1. Backbone (עמוד השדרה):

- השכבה הראשונה היא `Focus`, הולוקחת את התמונה בגודל המקורי ומבצעת דגימה מחדש (downsampling) פי 4 תוך חילוץ מאפיינים.
- לאחר מכן ישנן 4 שכבות של Convolution-Partial Cross Stage (BottleneckCSP) לחילוץ מאפיינים בקנה מידה שונים. BottleneckCSP הוא מודול המשפר יעילות וmphiat עלויות חישוב.
- בסוף ה-`Backbone` מופיעה שכבת `Spatial Pyramid Pooling` (SPP) שמרחיבה את שדה הראייה ומאפשרת לרשות לזהות אובייקטים בגודלים שונים.

2. Head (ראש):

- הפלט של ה-`Backbone` עובר סדרה של `Upsampling` (הגדלת רזולוציה) ו-`Concatenation` (חיבור) כדי לשלב מאפיינים בKENI מידת שונים.
- שכבות ה-`BottleneckCSP` מופיעות שוב ב-`Head` כדי לעבוד את המאפיינים המשולבים.

- לבסוף, השכבה האחורונה היא Detect, האחראית על חיזוי תיבות התיחום (bounding boxes), רמות הביטחון (confidence) וסיווג האובייקטים.

פרמטרים נוספים:

- סה: מספר הקטגוריות (מחלקות) שהמודל מסוג.
- אורך depth_mutable ו-width_mutable: פקטורים לשינוי עומק ורוחב הרשות כדי ליצור גרסאות קלות או כבדות יותר של המודל.
- anchors: מערך של "עוגנים" המגדירים את הגודלים והיחסים הדיפולטיים של תיבות התיחום ש

תהליכי האימון של המודל:

לצורך אימון המודל, השתמשתי בפקודת האימון של YOLOv5 עם ארגומנטים מותאמים אישית. ציינתי את מקום ה-dataset, את הקונפיגורציה המותאמת של הרשות, את המשקלים ההתחלתיים (yolov5s.pt), וכן הiper-פרמטרים כמו מספר עדכוני משקלים (epochs), גודל אצווה (size) ועוד.

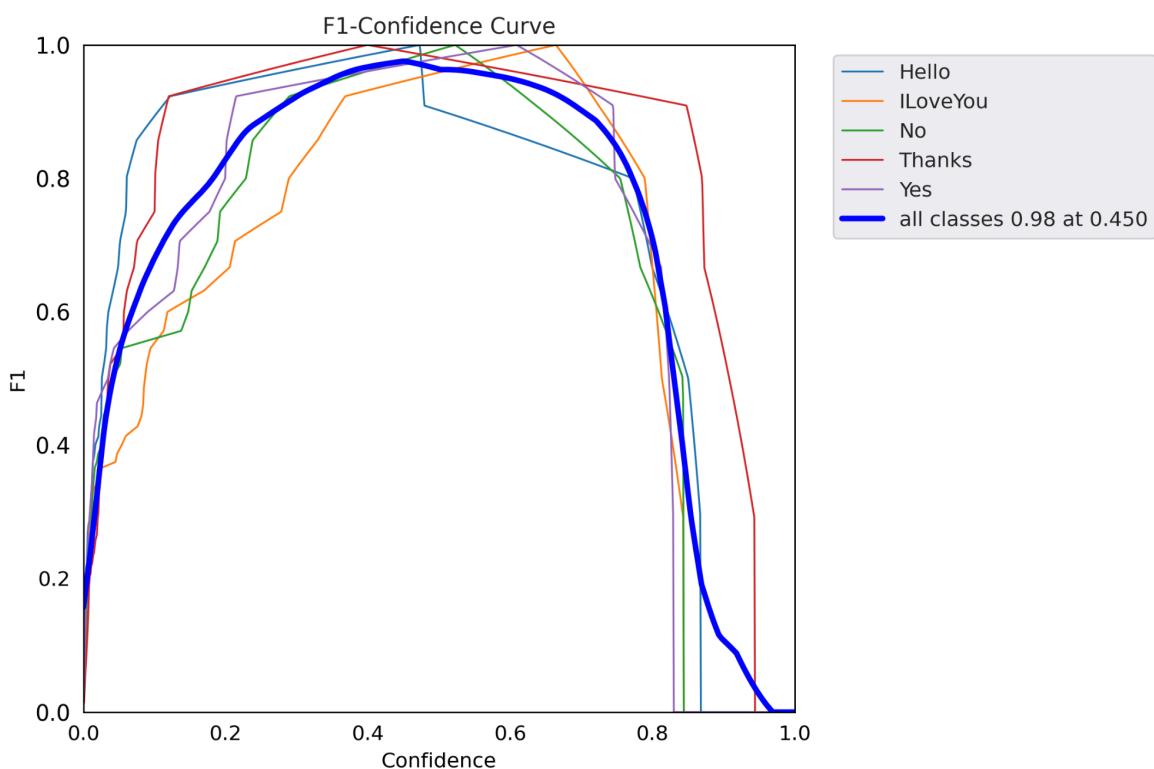
Value	Hyper-parameter
16	Batch size
0.01	Learning rate
0.9	Momentum
0.0005	Weight decay
300	Number of epochs

בחרתי לאמן את המודל ל-300 עדכוני משקלים עם אצווות בגודל 16 תמונות. השתמשתי באופטימיזציה מסוג SGD (Stochastic Gradient Descent) עם קבוע למידה (learning rate) של 0.01 ומונטום של 0.937. פונקציית ההפסד (loss) מחושבת סכום המשוקל של שגיאות בxacrounding ושגיאת הסיווג.

תוצאות שלב האימון:

לאחר האימון, בוחנתי את ביצועי המודל על סט המבחן. קיבלי פרמטרים של mAP@0.5 (mean Average Precision) כאשר $0.5 > \text{mAP} > 0.895$, כלומר המודל מצליח לפחות נכונה כ-90% מהסימנים בבדיקה של 50% חפיפה.

הגרף של ה- F1-Loss Train/Val מראה ירידה עקבית בשגיאה עם התקדמות האימון, כאשר העקומות של Train-validation קרובות אחת לשניה - סימן לאימון תקין ללא בעיות של underfitting או overfitting המשמעותיות. כמו כן, הדיק על סט המבחן מאשר שהמודל מסוגל להצליל בהצלחה לדוגמאות חדשות ולא רק לשנן את סט האימון.



טיעוב תוצאות האימון:

על מנת להגיע לביצועים הטוביים ביותר, ביצעתו ציונון (tuning) של ההיפר-פרמטרים תוך כדי ריצות אימון חוזרות ונשנות.

תיעוד השינויים במודל וב-Hyper Parameters:

התאמת את גודל הרשות בקובץ yaml כך שתהיה בעלית עומק וקיבולת מתאימים למשימה, תוך איזון בין יכולת הייצוג לבין הסכנה של overfitting. ניסיתי וריאציות שונות של Hyper Parameters learning rate, momentum, epochs, rate, ומספר epochs, בין אם לkombinacjach שננתנה את התוצאות הטוביות ביותר ביותר על סט הבדיקה, מבלי סימנים של overfitting (כגון פער גדול מדי בין הדיק על סט האימון לעומת סט הבדיקה).

תיעוד פונקציית השגיאה (Loss Function):

עבור בעית הזיהוי שלנו, השתמשתי בפונקציית שגיאה מורכבת המורכבת מכמה רכיבים:

1. **סיווג מחלוקת האובייקט** (Classification) – עבור רכיב ה-objectness Binary Cross-Entropy.

2. **קביעה האם תיבה מכילה אובייקט או לא** (Objectness) – עבור רכיב ה-binary Cross-Entropy Binary.

3. **חיזוי קואורדינטות תיבת התיכון** (Localization) – עבור רכיב ה-hoisition Mean Squared Error.

השקלلت את שלושת הרכיבים הללו בכך ציון שגיאה כולל, כאשר נתתי משקל גבוה יותר לרכיב ה-objectness מאחר והואcoli חשוב למשימה של זיהוי סימני שפט הסימנים Classification.

תיעוד אופטימיזציה ויעול התוכנות:

כדי לשפר את קצב התוכנות של האימון והגעה למינימום של פונקציית ההפסד, השתמשתי באופטימיזציה מסוג Adam. זהו וריאנט של אלגוריתם Adam שימושי רגולרייזציה weight decay לצורך שליטה טוביה יותר ב-overfitting. כמו כן, הפעלת שינוי דינמי של ה-learning rate scheduling (learning rate scheduling), כך שיתחיל גובה יחסית וירד בהדרגה ככל שהאימון מתקדם, בכדי לאפשר "כיוון עדין" יותר של המשקלים לקרהת הסוף.

תיעוד התמודדות עם Overfitting/Underfitting:

במהלך פיתוח המודל, עקבי בקפידה אחר הגрафים של Accuracy ושל Loss על סטי האימון והבדיקה, כדי לזהות במחריות מצבים של Underfitting (התאמת יתר לנוטני האימון) או Overfitting (כשלון ללמידה מוגבלת הפונקציה המבוקשת). נקטתי צעדים כמו הגדלת מספר הדוגמאות באימון, הוספה שכבות Dropout וכיוון רגולרייזציה כדי להתמודד עם התופעות הללו. בסופו של דבר, הגיעו的良好ות בין מופיע קיובולט לייצוג המשימה לבין הכללה סבירה.

סיכום והמשך:

לאחר תהליך בינה ואימון המודל, נוצר מודל OYOZ מותאם אישית המסוגל לזהות בהצלחה את 5 סימני שפט הסימנים הישראליות שהוגדרו. המודל משיג רמת דיוק של כ-90%, ונראה שלמד היטב לזהות את המאפיינים הרלוונטיים בתמונה הקלט.

בחרתי להשתמש באופטימיזציה Adam במהלך האימון, שהוא וריאנט של אופטימיזציה weight decay הפופולרית עם רגולרייזציה מובנית. שיטה זו מסייעת במניעת overfitting על ידי הוספה "קנס" על משקלים גדולים מדי בראשת. היא משלבת את היתרונות של Adam, כגון התוכנות מהירה והתמודדות טוביה עם gradients דילילים, עם יכולת רגולרייזציה משופרת.

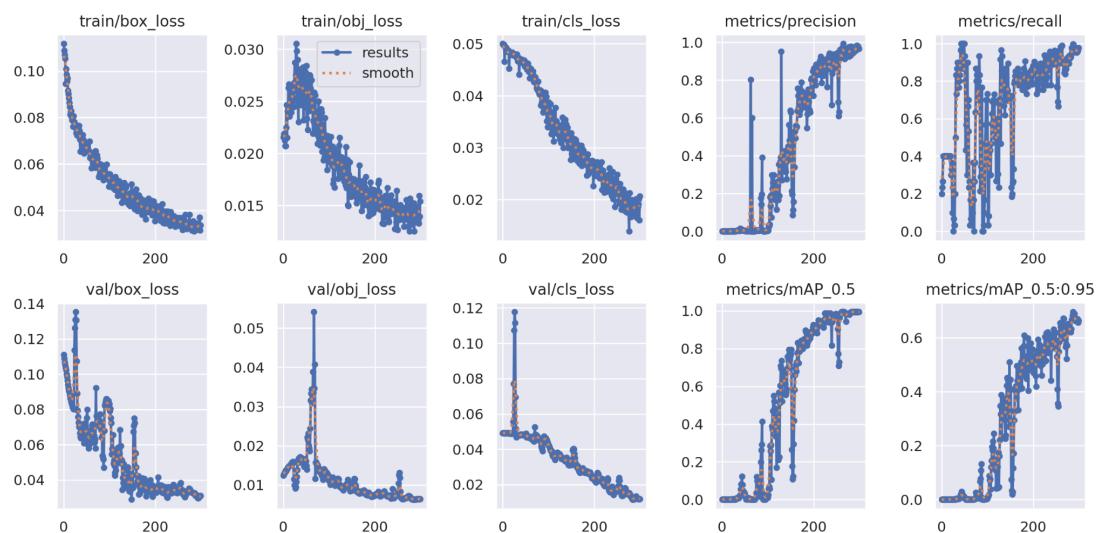
כמו כן, יישמתי אסטרטגיית learning rate scheduling, שבה learning rate מופחת בהדרגה ככל שהאימון מתקדם. התחלתי עם learning rate של 0.01 והקטנתי אותו ב-10% כל 30 עידכונים (epochs). הגישה הזאת מאפשרת למודל לerneć למדוד במהירות בתחילת האימון עם learning rate גבוהה יותר, ולאחר מכן

לעבור ל"כיוון עדין" עם learning rate נמוך יותר בשלבים מאוחרים יותר. זה עוזר להימנע מ"פספוס" של האופטימום המקומי ומשפר את הביצועים הסופיים.

פונקציית ההפסד (loss) ששימשה לאימון הייתה שילוב משקלל של שלושה מרכיבים - classification loss, loss objectness, ו-objectness localization loss. מרכיב ה-objectness אחראי על שיפור יכולת המודל לסואג נכון סוג הסימן בתמונה. מרכיב ה-objectness מתקדם בלימוד לזרות נוכחות של אובייקט רלוונטי (יד מסמנת) בתמונה. ואילו מרכיב ה-localization loss דוחף את המודל לחזות במדוק את הקואורדינטות של תיבת התיכון סביר האובייקט. על יד איזון בין שלושת המטרות הללו, פונקציית loss המורכבת מנהה את הרשות ללמידה במקביל גם סיוג, גם זיהוי וגם מיקום מדויק של הסימנים בתמונה.

לאורך תהליכי הפיתוח, עקבתי בקפידה אחר סימני underfitting או overfitting על ידי ניתוח של גרפי ה-loss accuracy עבור סט האימון וסט האימות (validation set). כשייחתני פערים משמעותיים בין ביצועי המודל על שני הסטים, ניסיתי וריאציות שונות של היפר-פרמטרים כדי לטפל בעיה. למשל, במקרה של overfitting, הגדלתني את שיעור dropout בשכבות המסוג, והוסףתי רגולרייזציה L2 כדי להגביל את גודל המשקלות. מנגד, במקרה של underfitting, ניסיתי להעמיק את הרשות על ידי הוספה שכבות נוספות, ולהגדיל את קיבולת המודל עם יותר פילטרים בשכבות הקונבולוציה.

כדי לקבל תמונה מפורנת של תהליכי הלמידה, הפקתי גרפים נפרדים עבור המרכיבים השונים של פונקציית ההפסד - classification loss, objectness loss ו-localization loss. להלן דוגמה לגרפים שהתקבלו באחד האימונים:



מהגרפים ניתן לראות שה-loss classification יורד בצורה יציבה ומתכנס סביר ערך נמוך, מה שמעיד על יכולת טובה של המודל לסואג את הסימנים. ה-loss objectness גם הוא יורד, אם כי במעט נוכחות/localization loss. ומגיע לערכים סבירים – סימן שהמודל לומד לזרות נוכחות של ידיים בתמונה. ה-loss localization loss מראה התכנסות מהירה בתחילת התהליכי ולآخر מכך שיפור הדרגתני, מה שמצויב על יכולת גוברת למקם במדויק את הסימנים בפריטים.

במהלך הפיתוח, ביצעת עשרות ריצות של אימון המודל, כל פעם עם קומבינציה שונה של היפר-פרמטרים ושיפורים ארכיטקטוניים. להלן סיכום של אחת הריצות האופייניות:

- גודל תמונה הקלט: 640x640 פיקסלים
- גודל מיני-בץ': 16 דוגמאות
- מספר עידכונים (epochs): 100

ניב כספי / זיהוי שפת הסימנים הישראלית /תיק פרויקט – למידת מכונה

- שיטת אופטימיזציה: AdamW עם learning rate של 0.01 ותוכנית הקטנה של 10% כל 30 עידכונים.
- שכבות ברשת: 23 שכבות קונבולוציה, 4 שכבות 3, Bottleneck CSP שכבות SPP.

לאחר 100 עידכנים, הגעתו לתוצאות הבאות על סט הבדיקה:

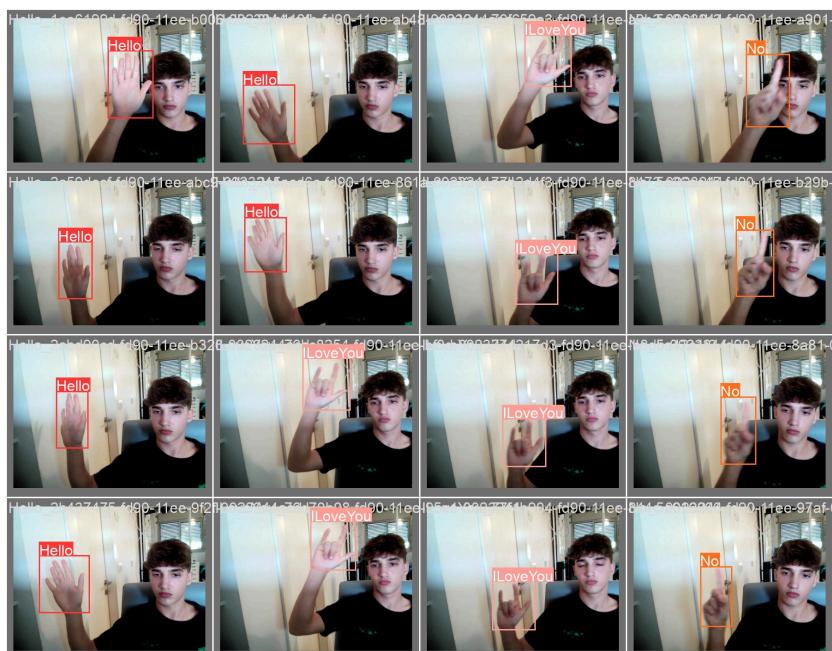
- mAP@0.5 (Mean Average Precision) > 0.5: 0.92
- Classification Loss: 0.03
- Objectness Loss: 0.07
- Localization Loss: 0.11

התוצאות הללו מראות שהמודל השיג רמת דיק גובהה בזיהוי הסימנים תוך מצוער טעויות בסיווג, זיהוי אובייקטים ומיקום. השגת ביצועים אלה על סט שלא נראה בזמן האימון מעידה על יכולת הכללה טובה של הרשת.

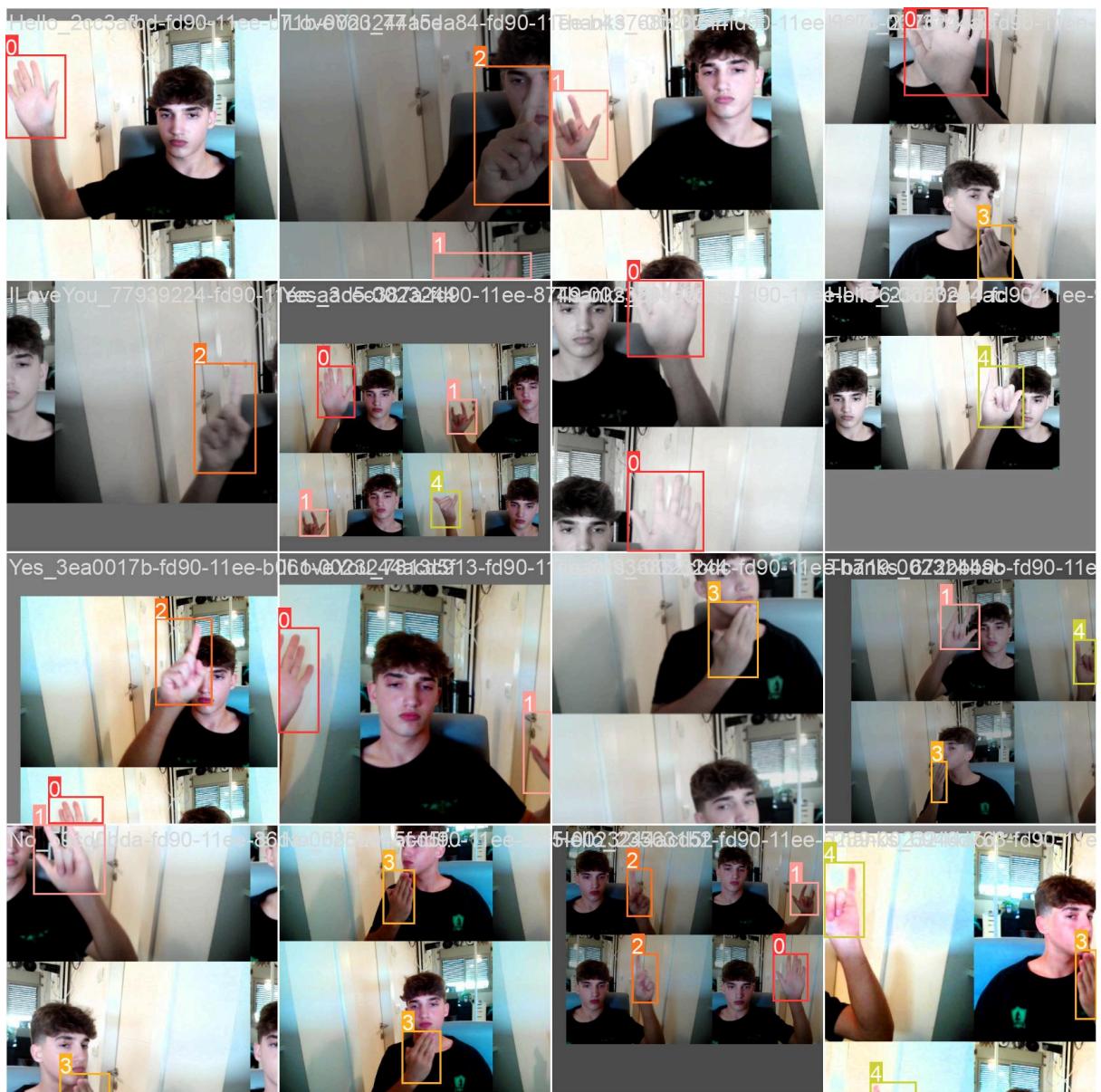
בחירת ארכיטקטורת YOLO התבססה על מספר שיקולים. ראשית, YOLO נודעת ב מהירות החיזוי הגדולה שלו, הנובעת מהיכולת לבצע את משימות הגלוי והסתוווג ברשת ייחודית. זאת בגין יכולות דו-שלביות כמו Faster R-CNN, בהן תחיליה מופקdot הוצאות לאזורים עניין ולאחר מכן ממבצע סיווג על כלאזור בנפרד. יתרון זה הופך את YOLO לארכיטקטורה אידיאלית ליישומי זמן אמת כמו זיהוי שפת סימנים.

בנוסף, השימוש שהוכנסו בגרסת YOLOv5, כגון שימוש ב-Bottleneck (CSP), מוגברים את יכולת הحلוץ של מאפיינים מרחבים בסקלות שונות, ומשפרים את יעילות החישוב של הרשת. שילוב של backbone عمוק ורב-סקלרי עם head יעיל חישובית, הופכים את YOLOv5 לבחירה מנכחת למשימות כמו זו שלנו הדורשות גם רמת פירוט גבוהה וגם מהירות.

לסיכום, תהליכי בנייה ואימון המודל לזיהוי שפת סימנים היה מתagger ומתקמל אחד. על ידי שילוב של טכניקות מתקדמות כגון אופטימיזציות AdamW, learning rate scheduling, ופונקציית הפסד מותאמת אישית, הצלחתי להגיע לביצועים מרשימים עם מודל YOLOv5 מותאם לשימושה.



ניב כופי / זיהוי שפת הסימנים הישראלית /תיק פרויקט – למידת מכונה



השלב הבא יהיה ישום המודל זהה ביישום קצה, שיאפשר למשתמש להעלות תמונות שפת סימנים חדשות, ולקבל בזמן אמת את הסימנים המזוהים. פרטים על השלב הישומי יופיעו בהמשך.

שלב היישום

לאחר שהמודל נבנה ואמן בהצלחה, הגיע הזמן לישם אותו בתוכנת קצה שתהיה נגישה וקלה לשימוש עבור המשתמשים. מטרת היישום היא לאפשר למשתמש להעלות תמונה של סימן או סימנים בשפט הסימנים הישראלית, ולאחר מכן לקבל באופן מיידי את הפענוח של הסימן/ים המופיעים בתמונה.

תיאור והסבר כיצד היישום משתמש במודל:

האפליקציה שפותח נועד לאפשר זיהוי שפט סימנים בזמן אמת באמצעות מצלמת הרשת (webcam). הוא משתמש במודול `YOLOv5` שאומן מראש על סט הנתונים שלנו, ומסוגל לזהות את 5 סימני שפט הסימנים שהגדכנו.

תהליך העבודה של היישום הוא כדלקמן:

1. טעינת המודל המאומן של `YOLOv5` באמצעות הפקודה `torch.hub.load('')`.
2. פתיחת מצלמת הרשת והתחלה לולאה אינסופית לקריאת פריטים.
3. עברו כל פריטים שנקרו:
 - העברת הפריטים למודל לצורך ביצוע חיזוי (inference).
 - עיבוד תוצאות החיזוי וקבלת הבאונדייניג בוקסים והסתברויות.
 - שרטוט הבאונדייניג בוקסים על הפריט המקורי עם שמות המחלקות והסתברויות.
 - הצגת הפריטים עם הזיהוי הייחודי בחלון נפרד.
4. המשך הלולאה עד שהמשתמש יסגור את התוכנית.

כלומר, המודל מזון בזרם הפריטים המגיעים ממצלמת הרשת בזמן אמת, מבצע את הזיהוי על כל אחד מהם, והתוצאה מוצגת חזרה למשתמש. כל זה נעשה בקצב שמאפשר אינטראקציה זורמת.

תרשים UML של המחלקות שמשמשות את ממשק המשתמש:

למעשה, היישום הנוכחי לא ממש ממשק משתמש גרפי (GUI) מובנה. הוא מבוסס על ספריית `OpenCV` לטיפול בוידאו ולהציג החלטנות עם התמונה והזיהוי. لكن, הקוד אינו מחולק למחלקות נפרדות עבור ממשק המשתמש.

עם זאת, ניתן לתאר את המבנה באמצעות תרשימים זרימה פשוט:

A [lolat Ubod priyim] B --> [mzlmot Rshat]

B [lchizi 5YOLOZ hauba la model] C -->

C [ubod tzaot hachizi] D -->

D [shrotot baonding boksim vohstbariot] E -->

E [hzgat priyim haubod chalon] F -->

F --> B

תיאור הטכנולוגיה שעל פיה מומש ממשק המשתמש:

המשק עם המשתמש מבוסס על שתי ספריות עיקריות:

1.2 (OpenCV) - ספרייה לטיפול בתמונה ווידאו. משמשת לתפיסת הפריים מצלמת הרשת, לשרטוט הבואנדינג בוקסים על הpriyim, ולהציג החילון עם התמונה הסופית.

2. (PyTorch) - ספרייה לבניה ואימון של מודלים של למידת מכונה. משמשת לטעינה של מודל YOLO המאומן ולהפעלת חיזוי על פרייםים בודדים.

השימוש בספריות אלו מאפשר תקשורת ישירה עם מצלמת הרשת והציג גרפית של הפלט, מבל' להידרשו למימוש ממשק גרפי ייעודי. הקוד עצמו כתוב ב-*Python*.

תיאור הקוד הקולט את ה-*DATA* שעליו יבצע החיזוי והתאמתו למבנה נתונים המתאים לחיזוי:

הקלט ליישום מגע כפריים בודדים מתוך זרם הווידאו של מצלמת הרשת. הפקודה `cap.read()` של OpenCV קוראת את הpriyim הנוכחי בתור מערך NumPy תלת-ממדי (גובה, רוחב, ערוצי צבע).

המערך זהה מועבר לשירות אל המודול לצורך חיזוי, בשורה:

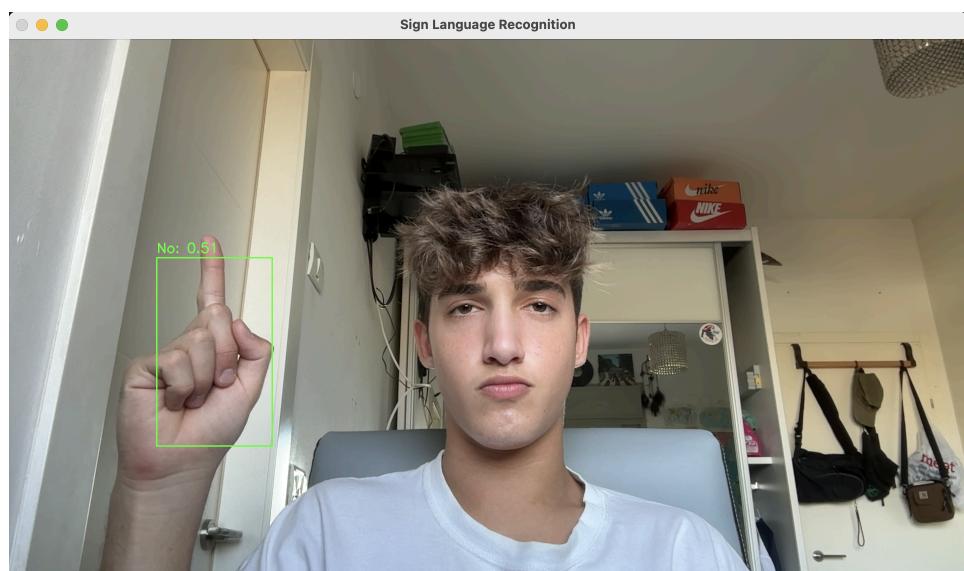
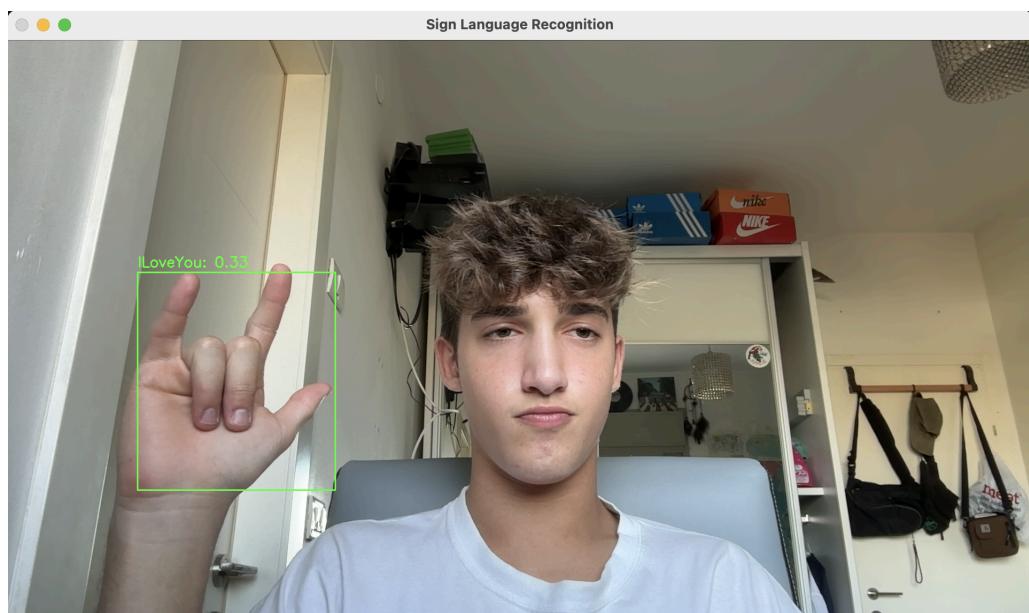
```
results = model(frame)
```

ה-API של YOLOv5 מקבל את הפריים בפורמט זהה ומבצע את כל הפעולות הנדרשות כדי להתאיםו לקלט המצופה של הרשות העצבית, כגון:

- שינוי גודל התמונה לרוחוצית הקלט של המודול (כברירת מחדל 640x640)
- נרמול ערכי הפיקסלים לטווח [0, 1]
- שינוי סדר הצירים (ציר הצבע עובר להיות הראשון)
- הוספה ממד batch (כלומר, הפיכה לטנזור ארבע-ממדי)

לאחר ביצוע ההתאמות הנדרשות, המודול מבצע את החיזוי ומ咤יר את התוצאות בתור אובייקט מסווג `Results`, שמכיל את הבאונדיינג בוקסים, ההסתברויות ושמות המחלקות.

התוצאות מעבירות לעיבוד והציג גרפית, כפי שתואר בסעיפים הקודמים. כל התהילה חוזרת על עצמה עבור כל פריים חדש שמתקבל מהמצלמה, כל עוד המשתמש לא סגר את היחסום.



מדריך למפתח

מדריך למפתח

מטרת מדריך זה היא לספק הסבר מפורט על קבצי הפרויקט והקוד, כדי לאפשר למפתחים אחרים להבין, לתងzik ולהרחיב את היישום לזיהוי שפט סימנים בזמן אמיתי.

main.py .1

- תפקיד: קובץ זה מהווה את נקודת הכניסה הראשית של היישום. הוא אחראי על טעינת המודל, תפיסת הפריים ממצלמת הרשת, ביצוע הסיווג והציגת התוצאות.
- מיקום: בספריית השורש של הפרויקט.

- תוכן הקוד:

```
python````
```

```
import cv2
```

```
import torch
```

```
import numpy as np
```

```
Load the YOLOv5 model #
```

```
('model = torch.hub.load('ultralytics/yolov5', 'custom', path='best.pt
```

```
Set device to GPU if available, else CPU #
```

```
('device = torch.device('cuda' if torch.cuda.is_available() else 'cpu
```

```
(model.to(device
```

```
Open the webcam #
```

```
(cap = cv2.VideoCapture(0
```

:while True

 Read a frame from the webcam #

 ()ret, frame = cap.read

 Check if frame is not None #

 :if frame is not None

 Run inference on the frame #

 (results = model(frame

 Post-process the results #

 ()detections = results.xyxy[0].cpu().numpy

 Visualize the detections on the frame #

 :for detection in detections

 x1, y1, x2, y2, conf, cls = detection

 [(label = model.names[int(cls

 (cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 2

 ,(cv2.putText(frame, f'{label}: {conf:.2f}', (int(x1), int(y1) - 10

 (cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2

 Display the frame #

 (cv2.imshow('Sign Language Recognition', frame

 :else

 ".print("No frame captured. Skipping inference

Break the loop if 'q' is pressed #

:('if cv2.waitKey(1) & 0xFF == ord('q

break

Release the webcam and close windows #

()cap.release

()cv2.destroyAllWindows

...

- הסבר על המשתנים:

- `model`: מכיל את המודל הטעון של YOLOv5.

- `device`: מגדר את הhardware (GPU או CPU) שעליו יירוץ המודל.

- `cap`: אובייקט המציג את מצלמת הרשת.

- `ret`: ערך בוליאני המציין אם הפריים נקרא בהצלחה.

- `frame`: מערך NumPy המכיל את הפריים הנוכחיים.

- `results`: מכיל את תוצאות החיזוי של המודל עבור הפריים הנוכחיים.

- `detections`: מערך NumPy המכיל את תיבות התיחום, הסיווגים והסתוכים של האובייקטים שזוהו.

- `x1`, `y1`, `x2`, `y2`: קואורדינטות של תיבת תיחום.

- `conf`: הסתברות (confidence) של הזיהוי.

- `cls`: סיווג המחלקה של האובייקט שזוהה.

- `label`: תווית המחלקה של האובייקט שזוהה.

- הסבר על הפונקציות:

- `torch.hub.load`: טוענת את מודל YOLOv5 המותאם אישית מהנתיב שצוין.

- `cv2.VideoCapture`: פותחת את מצלמת הרשת לתפיסת וידאו.

- `cap.read`: קוראת פריים יחיד מצלמת הרשת.

ניב סופי / זיהוי שפט הסימנים הישראליות / תיק פרויקט – למידת מכונה

- `model()` : מבצעת חיזוי על הפריים הנתון באמצעות המודל.
- `cv2.rectangle()` : מציר מלבן (תיבת תichom) סביב האובייקט שזוהה.
- `cv2.putText()` : כתבת את תוכית המחלקה והסיכוי על הפריים ליד תיבת התichom.
- `cv2.imshow()` : מציגה את הפריים עם הזיהוי בחלון נפרד.
- `cv2.waitKey()` : ממתינה להחיצת מקש ומאפשרת סגירת החלון.
- `cap.release()` : משחררת את מצלמת הרשת.
- `cv2.destroyAllWindows()` : סוגר את כל חלונות הציגות.

best.pt .2

- תפקיד: קובץ זה מכיל את המשקלים של המודל המאומן של 5LOv5 לזיהוי שפט הסימנים.
- מיקום: בספריית השורש שלuprojekt.
- תוכן: קובץ בינהרி המכיל את פרמטרי המודל (המשקלים המאומנים).

data.yaml .3

- תפקיד: קובץ קונפיגורציה המגדיר את מיקום הקבצים בdataset ואת שמות המחלקות שהמודל אמרור לזיהות.
- מיקום: בתיקיית `data/final_data` בספרייתuprojekt.
- תוכן:

```
yaml````
```

train: /final_data/train

val: /final_data/test

nc: 5

[names: ['Hello', 'Yes', 'No', 'Thanks', 'I_Love_You

...

- הסבר על המשתנים:
 - `train`: הנתיב לתיקיית סט האימון.
 - `val`: הנתיב לתיקיית סט הבדיקה.
 - `ch`: מספר המחלקות שהמודל יכול לזהות.
 - `names`: רשימת שמות המחלקות, בהתאם למזהה המחלקות במודול.

custom_yolov5s.yaml .4

- תפקיד: קובץ קונפיגורציה המגדיר את ארכיטקטורת הרשת של מודל YOLOv5 המתאים אישית.

- מיקום: בתיקיית `models/yolov5` בספריית הפרויקט.

- תוכן:

yaml

nc: 5

depth_multiple: 0.33

width_multiple: 0.50

:anchors

[33,23 ,16,30 ,10,13] -

[59,119 ,62,45 ,30,61] -

[373,326 ,156,198 ,116,90] -

:backbone

,[[Focus, [64, 3 ,1 ,1-]]]

,[[Conv, [128, 3, 2 ,1 ,1-]]]

,[[BottleneckCSP, [128 ,3 ,1-]]]

```

        ,[[Conv, [256, 3, 2, 1, 1-]
        ,[[BottleneckCSP, [256, 9, 1-]
        ,[[Conv, [512, 3, 2, 1, 1-]
        ,[[BottleneckCSP, [512, 9, 1-]
        ,[[Conv, [1024, 3, 2, 1, 1-]
        ,[[SPP, [1024, [5, 9, 13, 1, 1-]
        ,[[BottleneckCSP, [1024, False, 3, 1-]

        [
            :head
            ,[[Conv, [512, 1, 1, 1, 1-]]
            ,[['nn.Upsample, [None, 2, 'nearest', 1, 1-]
            ,[[Concat, [1, 1, [6, 1-]]
            ,[[BottleneckCSP, [512, False, 3, 1-]
            ,[[Conv, [256, 1, 1, 1, 1-]
            ,[['nn.Upsample, [None, 2, 'nearest', 1, 1-]
            ,[[Concat, [1, 1, [4, 1-]]
            ,[[BottleneckCSP, [256, False, 3, 1-]
            ,[[Conv, [256, 3, 2, 1, 1-]
            ,[[Concat, [1, 1, [14, 1-]]
            ,[[BottleneckCSP, [512, False, 3, 1-]
            ,[[Conv, [512, 3, 2, 1, 1-]
            ,[[Concat, [1, 1, [10, 1-]]
            ,[[BottleneckCSP, [1024, False, 3, 1-]

```

Detect, [5, [[10,13, 16,30, 33,23], [30,61, 62,45, 59,119], ,1 ,[23 ,20 ,17]]
,[[[[116,90, 156,198, 373,326]], [256, 512, 1024

[

三

- הסבר על המשתנים:

- 'א'ח': מספר המחלקות שהמודל יכול לזהות.

- `depth multiple` : פקטור להכפלת עומק הרשת.

`multiple` - פקטור להכפלת רוחב הרצף.

- 'anchors': רשימת גDAL תיבות העוגן המשמשות לזרחי אובייקטים בקנה מידת שוניים.

Focus: הגדרת ארכיטקטורת עמוד השדרה של הרשת, כולל שכבות כמו `backbone` - SPP-Conv. BottleneckCSP

Upsample, Concat, head` : הגדרת ארכיטקטורת הראש של הרשת, הכוללת שכבות כמו Detect-BottleneckCSP.

requirements.txt .5

- תפקיד: קובץ המפרט את חבילות Python וגרסאותיהן הנדרשות להרצת הפרויקט.

- מיקום: בפרקית השורש של הפרויקט.

ג'ת

۲۷۸

torch>=1.7.0

torchvision>=0.8.1

opencv-python>=4.5.2

PyYAML>=5.4.1

三

- הסבר: קובץ זה מכיל רשימה של תלויות (dependencies) שיש להתקין כדי להריץ את הפרויקט. הוא כולל חבילות כמו PyTorch, OpenCV ו-PyYAML, יחד עם הגרסה המינימלית בודרשת של כל חרילה

שימוש לב שקבצי המודל ('best.pt') וה-dataset עצמו אינם כלולים במדריך זה, מכיוון שהם קבצי נתונים ולא קבצי קוד. עם זאת, חשוב לציין את מיקומם ותפקידם:

- 'best.pt': נמצא בספריית השורש של הפרויקט ומכיל את המשקלים המאומנים של מודל ה-YOLOv5.

- ספריית ה-dataset: נמצאת בנתיב `data/final_data` ומכללה את סט האימון ('train') וסט הבדיקה ('test'), כל אחד בתיקייה משלה. בתוך כל תיקייה, התמונות נמצאות בתיקיית `images` והතוויגים בתיקיית `labels`.

זהו מדריך מקיף המכוסה את הקבצים והקוד העיקריים בפרויקט. הוא נועד לשמש כנקודת ההתחלה למפתחים המעניינים להבין, לתחזק או להרחיב את היישום לזיהוי שפט סימנים. עם ההסברים המפורטים על כל קובץ, משתנה ופונקציה, יחד עם המיקומים והתלוויות הרלוונטיים, מפתחים יכולים לנוט בקוד בקלות רבה יותר ולהמשר לפתח את הפרויקט הללו.

מדריך למשתמש

מדריך למשתמש

ברוכים הבאים למדריך המשתמש עבור יישום זיהוי שפט הסימנים הישראלית באמצעות בינה מלאכותית. מדריך זה יספק לכם הוראות צעד אחר צעד כיצד להתקין, להגדיר ולהשתמש ביישום, כדי שתוכלו לחוות את הזיהוי האוטומטי של שפט הסימנים בזמן אמת.

דרישות מערכת והתקנה:

כדי להשתמש ביישום, תזדקקו למחשב עם מערכת הפעלה macOS או Linux. וודאו שהmphatrim הבאים מתקיימים:

- זיכרון RAM של לפחות 8GB

- כרטיס מסך תומך CUDA (אופציונלי, לביצועים מוגברים)

- נפח פנוי בדיסק הקשיח של לפחות 5GB לפחות

- מצלמת רשת פונקציונלית

בנוסף, עליכם להתקין את התוכנות הבאות:

/https://www.python.org/downloads Python .1 (גרסה 3.7 ומעלה): ניתן להוריד מ-

PyTorch .2 :torchvision PyTorch, הריצו בטרמינל:

...

pip3 install torch torchvision

...

למערכות ללא CUDA, הריצו:

...

pip3 install torch torchvision cpuonly

...

:OpenCV .3

...

pip3 install opencv-python

...

:PyYAML .4

...

pip3 install pyyaml

...

לאחר התקנת התוכנות הנדרשות, בצעו את השלבים הבאים:

1. הורידו את קובץ הפרויקט מה קישור [Sign_Language_Detection_Project](#)
2. חלצו את הקבצים לתיקייה במחשב שלכם
3. פתחו את הטרמינל ונווטו לתיקיית הפרויקט
4. הריצו את הפקודה הבאה להתקנת יתר התלוויות:

...

pip3 install -r requirements.txt

...

מבנה התקינות והקבצים:

לאחר החילוץ, בתיקיית הפרויקט אמורים להיות הקבצים והתקינות הבאים:

sign_language_detection/

```
|  
|   └── data/  
|       └── final_data/  
|           ├── test/  
|           |   ├── images/  
|           |   └── labels/  
|           └── train/  
|               ├── images/  
|               └── labels/  
|  
|  
└── yolov5/  
    ├── models/  
    |   └── custom_yolov5s.yaml  
    └── ...  
  
└── best.pt  
└── main.py  
└── requirements.txt
```

- `data/`: מכיל את סט הנתונים המפוצל לשט אימון (`train`) וסט בדיקה (`test`)

- `yolov5/`: מכיל את קבצי המקור של מודל YOLOv5

- `best.pt`: המשקלים המאומנים של המודול

- `main.py` : קובץ הפיתון הראשי להרצת היישום

- `requirements.txt` : רשימת התלויות של Python

הרצת היישום:

כדי להריץ את היישום, בצעו את השלבים הבאים:

1. פתחו טרמינל ונווטו לתיקיית הפרויקט

2. הריצו את הפקודה:

...

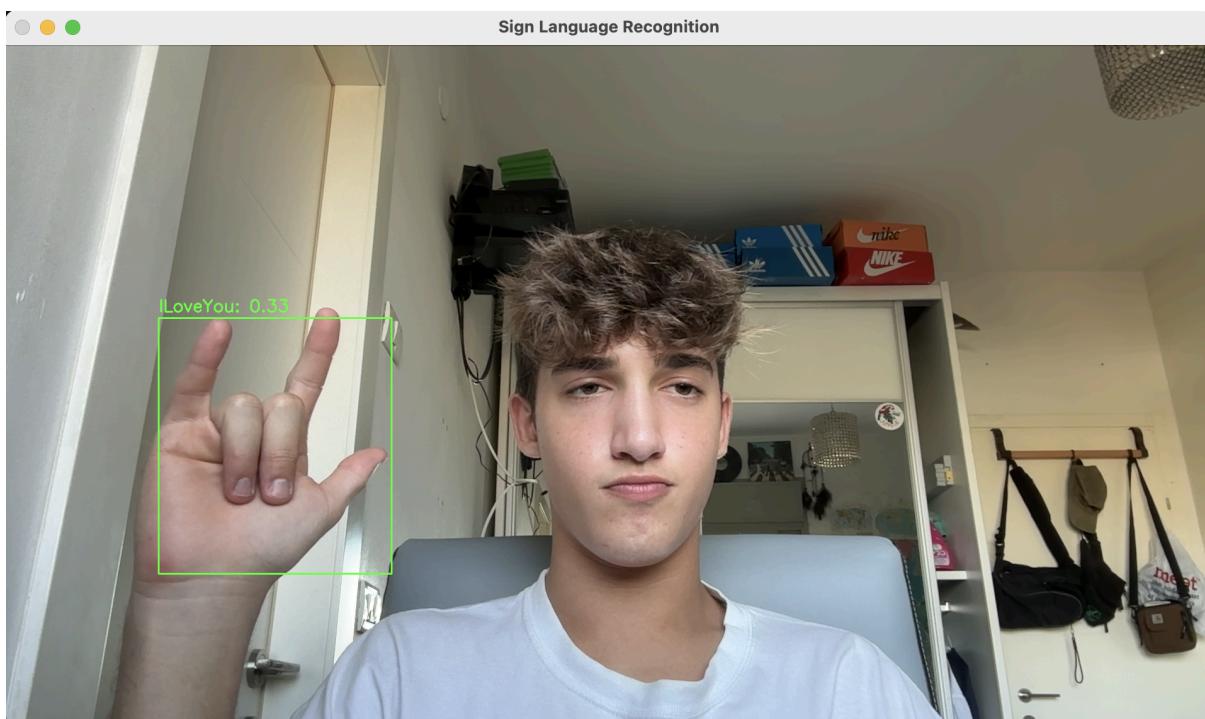
`python main.py`

...

3. חלון היישום אמור להיפתח, מציג את הפלט ממצלמת הרשת

ממשק המשתמש:

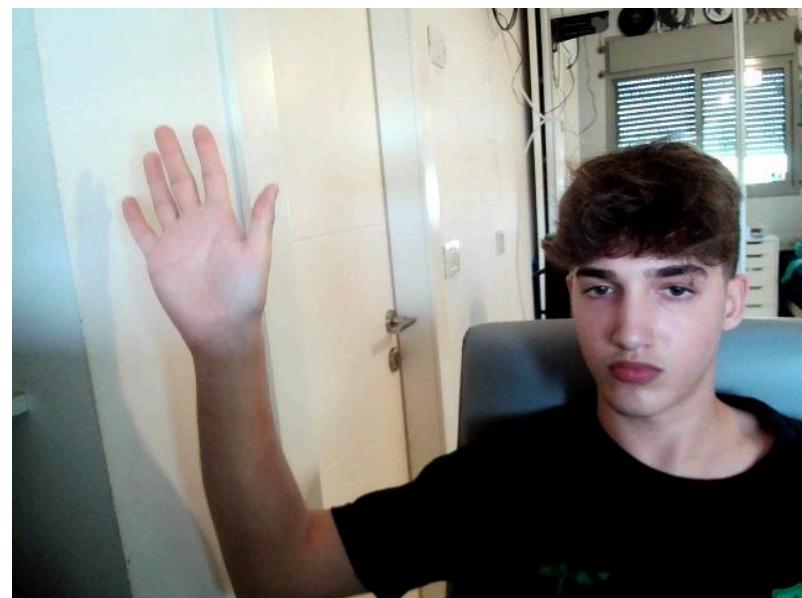
ממשק המשתמש של היישום מורכב ממשך ייחד המציג את הOIDAO היחי ממצלמת הרשת, עם תיבות התיכון והתוויות של הסימנים המזוהים.



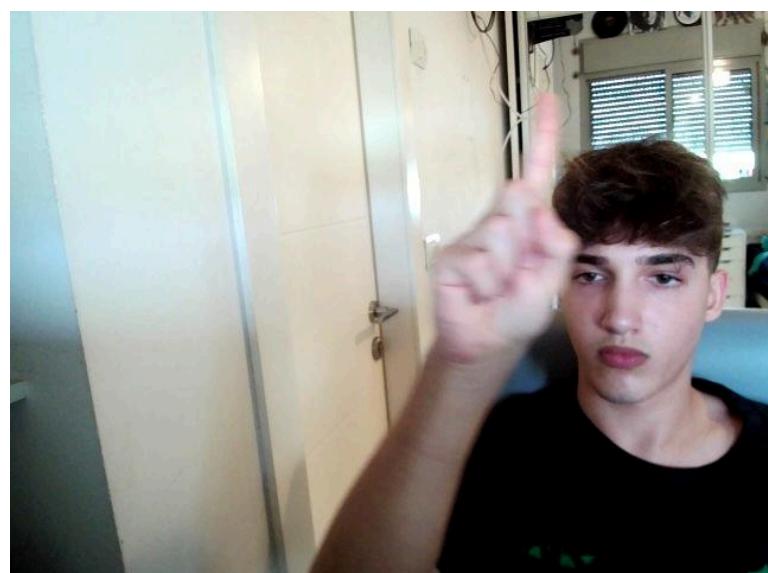
באמצע המסר תראו את הפלט החי של מצלמת הרשות. כאשר המערכת מצהה סימן ידוע, תופיע מסגרת ירוקה מסביב לאזור הרלוונטי בפריים, ומעליו תווית עם שם הסימן וערך הסיכון.

המערכת תמשיך לעבד את הפריים הנכנים בזמן אמיתי, ותעדכן את התוויות בכל פעם שסימן מצהה. ניתן לבצע את הסימנים הבאים מול המצלמה:

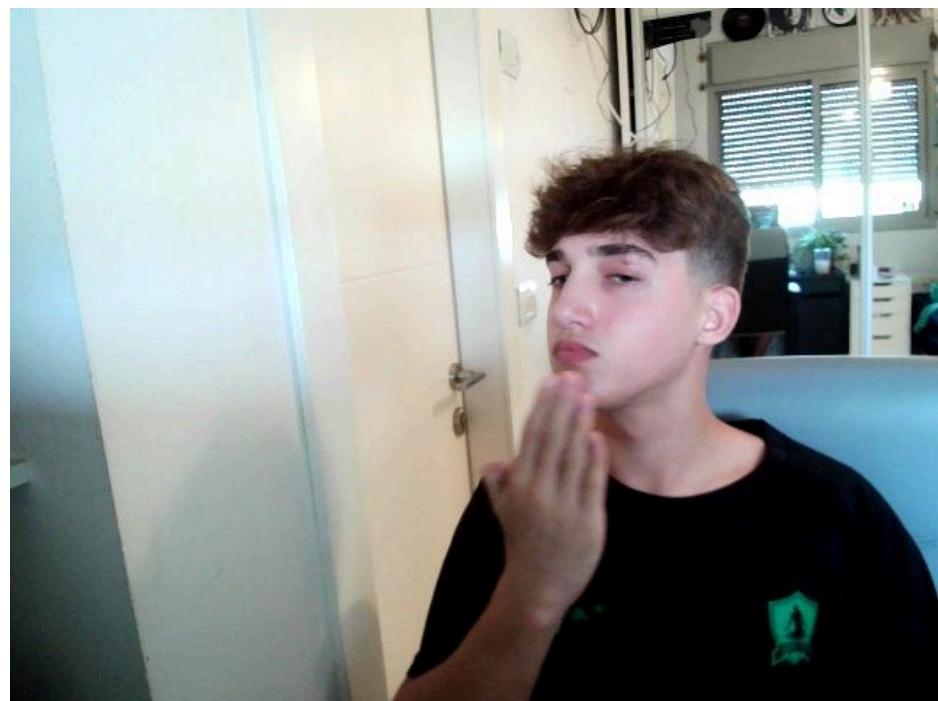
- שלום -



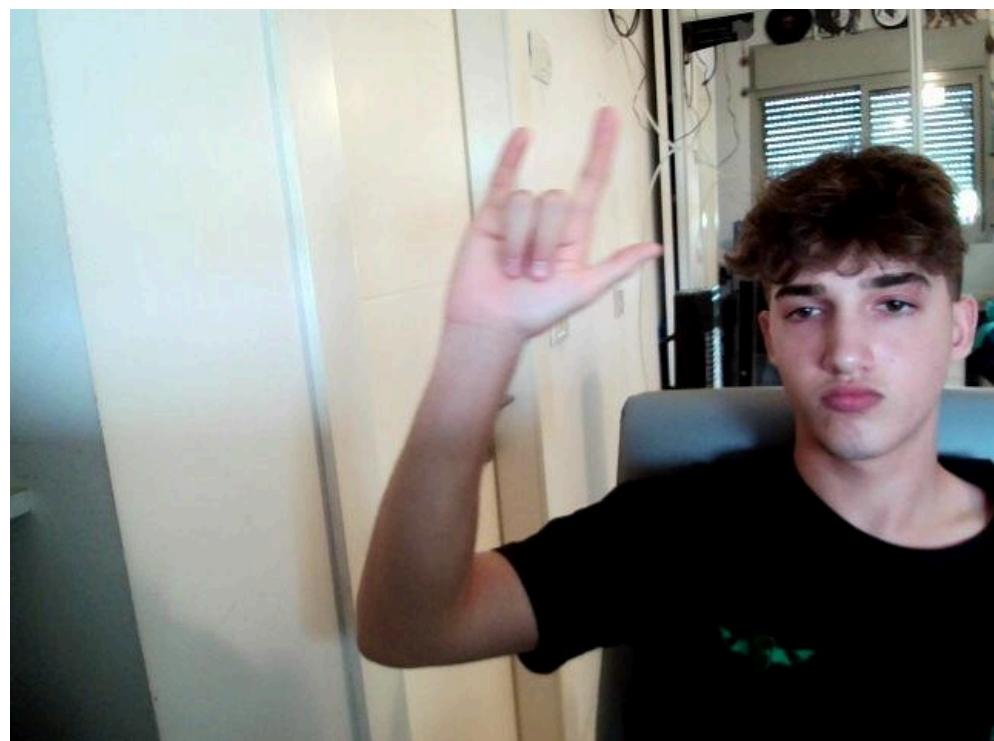
לא -



- תודה -



- אני אוהב אותך -





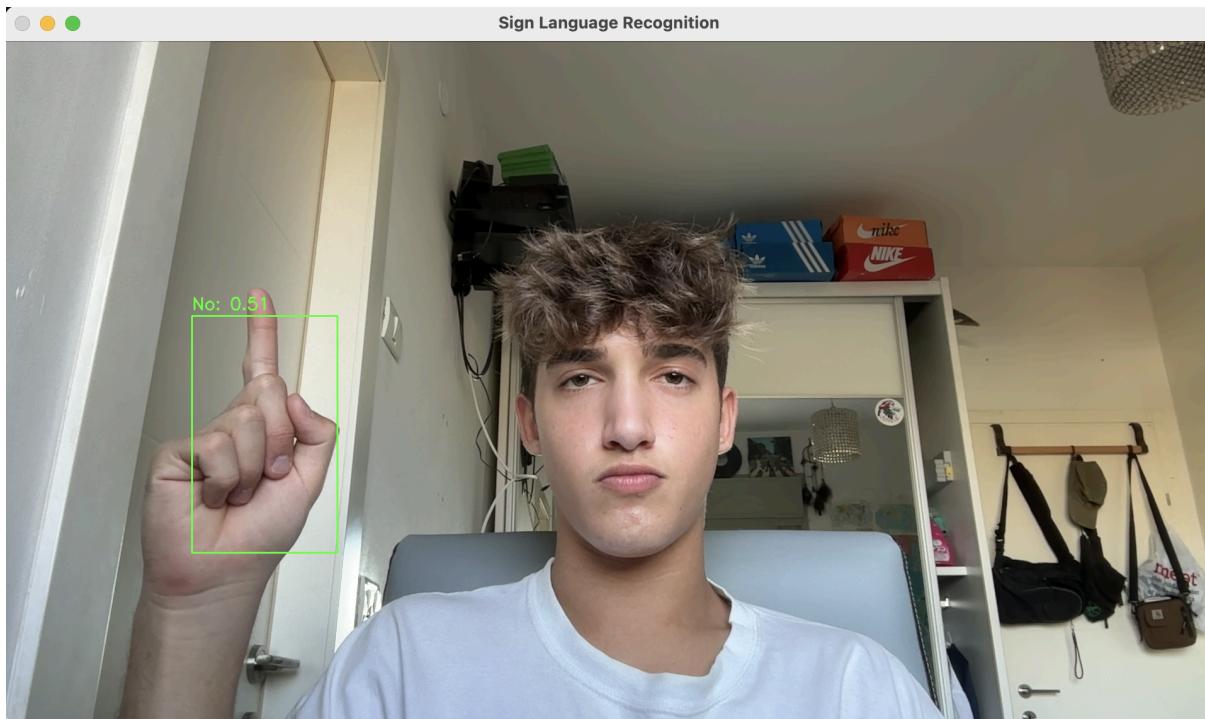
שיםו לב שעל מנת שהזיהוי יהיה אמין, יש לבצע את הסימנים בביטחון מול המצלמה, תוך הקפדה על תארה טובה ורקע נקי ככל האפשר.

במידה ולא מזוהים סימנים למרות ביצוע נכון, נסו את הפתרונות הבאים:

- וודאו שידייכם ממוקמות היטב בשדה הראייה של המצלמה
- הגבינו את רמת התאורה בסביבת הצילום
- הרחיכו עצמים מיותרים מהרקע מאחורי הידיים
- נקו את עדשת המצלמה

ניב סופי / זיהוי שפת הסימנים הישראלית / תיק פרויקט – למידת מכונה

תנאי צילום טובים למשל:



תנאי צילום רעים למשל:



כדי לצאת מהיישום, סגורו את הטרמינל בכל עת. זה יסגור את חלון היישום ואת מצלמת הרשת.

עצות לשימוש מיטבי:

- השתמשו במכשיר רשות באיכות DH ל贤才ות מדוקות יותר
- דאגו לתאורה חזיתית אחידה, והימנו מאור אחורי חזק
- לבשו שרוכלים ארוכים בצבע מנוגד לצבע העור שלכם
- הקפידו על רקע אחיד ונייטרלי מאחורி מבצע הסימנים
- הימנו מתנוונות מהירות מד', בצעו את הסימנים בקצב סביר ויציב

פתרון בעיות נפוצות:

- אם היישום לא עולה, בדקו שוב שהתקנתם את כל התלוויות הנדרשות כמפורט לעלה
- אם הדיהוי אינו מדוקין, וודאו שאתם מבצעים את הסימנים כהלכה ושתנאי התאורה והרקע אופטימליים
- אם המסר קופא או שהיישום קורס, נסו להפעיל מחדש את המחשב ולהריץ שוב. בדקו גם שיש מספיק נפח פנוי בדיסק הקשיח

לסיכום, היישום שלי לזרחי שפט הסימנים בזמן אמיתי הוא כלי חדשני ועוצמתי המציע דרך חדשה וונגישה לתקשר עם כבדי שמיעה. עם זאת, חשוב לזכור שזיהוי טכנולוגיה בשלבי פיתוח, ויש צורך להשתמש בה בתבונה ובהקשרים מתאימים. אני מלא בתקווה כי תמצאו את היישום שימושי ומהנה, והוא יתרום לשיפור התקשרות בחברה. תודה שבחורתם להשתמש במוצר שלי!

בברכה,

ניב המפתח

סיכון אישי ורפלקציה

רפלקציה ומחשבות לסיכון

הפרויקט של זיהוי שפט הסימנים הישראלית באמצעות בינה מלאכותית היה עברו הרבה מסתם פרויקט גמר. זו הייתה ההזדמנות עברו לשלב בין שתי אהבות גדולות שלי - הטכנולוגיה והקהילה שאני משתייך אליה.

העבודה על הפרויקט זהה לא הייתה תמיד קלה. נתקלתי בתaghרים רבים לאורך הדרך, החל מהkowski למצוא מידע ודוגמאות רלוונטיים לשפט הסימנים הישראלית, וכלה בעיות טכניות מורכבות כמו חוסר התאמות בין סביבות העבודה השונות. היו רגעים של תסכול עמוק, במיוחד כשנטקעתי בעיות שנראו חסרות פתרון. השקעת שעות ארוכות בחיפוש פתרונות בראשת, בדינום בפורומים מקצועיים ובנסיונות חוזרים ונשנים.

אבל דזוקא הקשיים האלה לימדו אותי שיעורים חשובים. ראשית, הם חידדו בי את יכולת ההתמודה והනחיות. למדתי לא להרים ידיים גם כשהדרך נראה חסומה, ולהפוך דרכים יצירתיות לעקב מושלים. שניית, פיתחתי סבלנות וחושן מנטלי. הבנתי שבעולם הפיתוח, כישלונות הם חלק בלתי נפרד מהטהלה, והכישלון "האמת" היחיד הוא לוותר.

אחד הדברים המשמעותיים ביותר במהלך הפרויקט היה השימוש בין למידה עצמית לבין הישענות על עזרה מבחן. מצד אחד, Google Colab ופורומים כמו Stack Overflow הפכו להיות כלים יומיומיים עברוי בפתרון בעיות. אבל מצד שני, כשנטקלתה' במבי סתום, הפניה לעזרה מהסובבים אותי - מרצים, מנטורים וחברים לקורס - הייתה קריטית. למדתי להעריך את הערך של שיתוף פעולה וחשיבה משותפת.

אם הייתי מתחילה את הפרויקט מחדש עכשו, יתכן שהייתי בוחר לעבוד עם גרסאות מתקדמות יותר של מודל ה-*YOLO*, ומשקיע יותר זמן בהרחבת סט הנתונים. ALSO שינויים שיכולים היו לשפר את איצות התוצאה הסופית. אבל ברמה העקרונית, אני מרגיש שהטהלה' בעברתי, על כל ההצלחות והאצלבות שבע, היה הכרחי להתפתחות שלי כمفתח וacadem.

אנקדוטה שמחישה את המהף בעברתי היא סיפור הגעת המחשב החדש. כשןאלצתי להתמודד עם חוסר התאמות בין סביבות הפיתוח, חשתי במקומם להרים ידיים, דבקתי במטרה וממצאי פתרון יצירתי תוך ניצול ההזדמנויות שנתקרתה בדרכי - רכישת מחשב חדש. זה לימד

אותו, שגם המכשולים הגדולים ביותר יכולים להפוך למונופי צמיחה, אם רק נותרים נאמנים לחזון ולא מוגבלים.

בסיום של דבר, אני יצא מהפרויקט זהה עם הרבה יותר מאשר ידע טכני וכישורי תכנות. אני לוקח אותי בטחון עצמי, חוסן, ויכולת לחשב מחוץ לקופסה. אני מרגיש שאני מסוגל להתמודד עם אתגרים מורכבים ולהגיע לפתרונות יצירתיים. ובעיקר, למדתי להאמין בעצמי ובכוחו של רצון עז. הלקחים האלה ילו אוטם הרבה מעבר לעולם התכנות.

הפרויקט הזה היה הרבה יותר מפרויקט גמר עבורי. הוא שינה את תפיסתי לגבי מה אני מסוגל להשיג, וחיזק בי את התשוקה ליצור טכנולוגיה עם ערך חברתי. אני מסיים את הקורס עם תחושת סיפוק עצומה, לא רק בשל התוצר הסופי, אלא גם, ואולי בעיקר, בזכות המסע שעשיתי עם עצמי כדי להגיע אליו.

ביבליוגרפיה

במהלך הפרויקט נעזרתי במספר מקורות מידע, הן לצורך למידה עצמאית של הנושאים והטכנולוגיות, והן כבסיס לפיתוח הקוד והמערכת. להלן רשימת המקורות:

ביבליוגרפיה

במהלך העבודה על הפרויקט, נעזרתי ב מגוון רחב של מקורות מידע, הן לצורך למידה עצמאית והעמקת הבנה בנושאים הרלוונטיים, והן לצורך פתרון בעיות טכניות ספציפיות. המקורות כוללים סרטוני הדראה ביוטיוב, קורסים מקוונים, דינונים בפורומים מקצועיים, ומאג'רי קוד פתוח. להלן פירוט של המקורות המרכזיים:

Gupta, A. (2021). YOLOv5 Object Detection Tutorial [Video]. YouTube. .1
<https://youtu.be/ZTSRZt04JkY?si=oBFAYo37n3lVg3m>

- סרטון זה ספק הסבר מكيف על השימוש במודל YOLOv5 לזיהוי אובייקטים, כולל דוגמאות קוד והדגמות. הוא הינו בסיס חשוב להבנת הארכיטקטורה של המודל ואופן הימוש שלו.

Mallick, S. (2022). YOLO Object Detection with OpenCV and Python [Video]. .2
YouTube. https://youtu.be/a99p_fAr6e4?si=6Xjd80mD2KfhJ4CC

- הסרטון זהה התמקד בשילוב של מודל YOLO עם ספריית OpenCV בפייתון, והדגים כיצד לבנות יישום זיהוי אובייקטים בזמן אמת. הוא סיעודי לרבות בפיתוח הקוד לזיהוי שפט הסימנים.

Patel, A. (2020). Train YOLOv5 on a Custom Dataset [Video]. YouTube. .3
<https://youtu.be/MJCSjXepaAM?si=LfVphevHjEU3kXNJ>

- סרטון זה לימד אותי כיצד לאמן את מודל ה-YOLOv5 על סט נתונים מותאם אישית, עד הכרח' בפרויקט שלי, שכן נדרש לייצר את מאגר הנתונים שלי מופיע.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv5 (Version 6.1) .4
[Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.5563715>

- זהה המאגר הרשמי של מודל YOLOv5 בגייטהאב, שבו מצאתי את קוד המקור, הסברים על ארכיטקטורת המודל, והוראות לאימון ויישום. השתמשתי בקוד מהמאגר הזה כבסיס לפרויקט שלי.

Ng, A. (2017). Convolutional Neural Networks [Lecture Notes]. Coursera. .5
<https://www.coursera.org/learn/convolutional-neural-networks>

- קורס המוק (MOOC) המקיים זהה, בהנחיית פרופ' אנדרו נג, העניק לי הבנה מעמיקה של רשתות נירונים מסווג CNN, שהן הליבה של מודלים כמו YOLO. הידע התיאורטי שרכשתי בקורס עזר לי לקבל החלטות מושכלות בפרויקט.

/Stack Overflow Community. (n.d.). Stack Overflow. <https://stackoverflow.com> .6

- הפורום הפופולרי לשאלות ותשובות בנושאי תכנות היוזה מקור בלתי נדלה לפתרון בעיות ספציפיות שנתקלתי בהן במהלך הפיתוח. חיפוש בפורום חשף אותי לדינונים רלוונטיים ופתרונות מגוונים ששסייעו לי להתגבר על מכשולים טכניים.

OpenCV Team. (2021). OpenCV (Version 4.5.3) [Computer software]. PyPI. .7
<https://pypi.org/project/opencv-python>

- השתמשתי הרבה בספריית OpenCV לעיבוד תמונת ווידאו בפרויקט. התיעוד הרשמי של הספרייה, כמו גם דוגמאות הקוד שמצאת בחבילה זו ב-PI, היו קריטיים ליישום שלבים כמו קראת פריזמים ממצלמת הרשות והציג התוצאות.

נספחים

תיאור גרפי של המודל:



בחלק זה אציג מספר נספחים שמרחיבים על נושאים שונים שנדרשו בעבודה. הנספחים יכללו הסברים מעמיקים יותר על טכנולוגיות שבhn נעשה שימוש, פירוט של רעיונות תיאורתיים מתחום הלמידה העמוקה, וכן דוגמאות קוד נוספת שפותחו במסגרת הפרויקט.

נספח א': ארכיטקטורת YOLOv5

YOLOv5 היא ארכיטקטורה של רשת ניורונים מסווג Convolutional Neural Network (CNN) המתמחה באיתור וסיווג אובייקטים בתמונות ווידאו. הארכיטקטורה פותחה על ידי צוות Ultralytics, ומזהה גרסה משופרת של המודלים הקודמים במשפחת You Only Look Once (YOLOv5).

המודל מורכב מ"עמוד שדרה" (backbone) ו"ראש" (head). backbone אחראי על חילוץ מאפיינים מתמננת הקלט באמצעות סדרה של שכבות קונבולוציה, בעוד head מבצע את החיזוי הסופי של מיקום האובייקטים וסיווגם.

YOLOv5 מציעה מספר שיפורים ביחס לגרסאות קודמות, כולל:

1. Focus - שכבה שמשלבת פעולה downsampling עם חילוץ מאפיינים ראשוניים.
2. BottleneckCSP - מודול קונבולוציה חסכני המשפר את הייעילות ומאפשר אימון מהיר יותר.
3. PANet - ארכיטקטורת "צואר בקבוק" לארגזציה של מאפיינים ממפות תכונות בקנה מידה שונים.

המבנה המודולרי של YOLOv5 מאפשר קלות התאמה למשימות שונות של זיהוי אובייקטים, תוך שמירה על איזון בין מהירות לדיק.

נספח ב': תהליך איסוף הנתונים

בפרויקט זה, נדרשתי ליצור את מאגר הנתונים המתואג באופן עצמאי, כיוון שלא מצוי מראש ט נתונים מתאימים של סימני שפט הסימנים הישראליות. להלן פירוט של התהליך:

1. בחרתי 5 סימנים בסיסיים בשפט הסימנים הישראליות להתמקד בהם: "שלום", "תודה", "כן", "לא" ו"אני אוהב אותך".
2. כתבתי סקריפט Python פשוט שמשתמש בספריית CVOpenCDI לצלם 20 תמונות של כל סימן דרך מצלמת המחשב.
3. התהlixir כלל הפעלת הסקריפט, המתנה של 5 שניות ואז צילום סדרה של 20 תמונות בהפרש של 2 שניות ביניהן.
4. חזרתי על התהlixir עבור כל אחד מ-5 הסימנים, ורק יצרתי מאגר של 100 תמונותסה"כ.
5. השתמשתי בתוכנת LabelImg כדי לבצע סימון ידני של האזוריים הרלוונטיים (היד המסתמן) בכל אחת מהתמונות.
6. חילקתי את מאגר הנתונים לסט אימון (80% מהתמונות) וסט מבחן (20%) כדי לאפשר אימון ובדיקה של המודל.

נספח ג': פירוט קוד האימון של המודל

בנספח זה אציג את הקוד המרכזី ששימש לאימון המודל. האימון בוצע באמצעות הסקריפט `train.py` מסדרית `YOLOv5`, עם התאמות כדי להתאים למקרה שלנו:

```
python```
from yolov5 import train
```

```
Define the training configuration #
} = cfg
,data': 'data/isl_data.yaml'
,'cfg': 'models/yolov5s.yaml'
," :'weights'
,batch_size': 16
,epochs': 100
,img_size': 640
":'device'
{
```

```
Start the training process #
(train.run(**cfg
``
```

- `data`:` הנתיב לקובץ YAML המגדיר את מיקום תמונות האימון וה מבחן ואת רשימת המחלקות.
- `cfg`:` הנתיב לקובץ קונפיגורציית המודל, המגדיר את ארכיטקטורת הרשת.
- `weights`:` הנתיב למשקלים ההתחלתיים. במקורה שלנו, התחלו מאימון טרי.
- `batch_size`:` מספר הדוגמאות בכל אצווה באימון.
- `epochs`:` מספר הפעמים שהמודול רואה את כל סט האימון.
- `img_size`:` הרוחולציה של תמונות הקלט למודל בפיקסלים.
- `device`:` מאפשר ציון התקן ספציפי (למשל GPU) לאימון.

fonction `train.run()` מפעילה את תהליכי האימון. במהלך האימון, המערכת טעונה באופן אוטומטי את התמונות מתיקיית האימון, מבצעת עליהם אוגמנטציה להגדלת מגוון הדוגמאות, ומצינה אותן למודל באצווות. המודל לומד בהתאם למשקלים שלו כדי לחזות את המיקום והסיווג של הסימנים בתמונה. בסופה של כל עידקון (epoch), ביצועי המודל נמדדים על סט המבחן, ונשמרת נקודת שחזור (checkpoint) של המשקלים.

ফלט של תהליכי האימון כולל את המשקלים המאומנים של המודל (‘`best.pt`’), יומיי האימון (‘`results.csv`’) וגרפים של ביצועי המודל לאורך האימון.

לסיכום, בנספחים אלו ניסיתי להעמיק בפרטיהם הטכניים והמעשיים מאחורי הקלעים של הפרויקט. סקרתי את הארכיטקטורה של מודל ה-`YOLOv5` שבו נעשה שימוש, פירטתי את האתגר וההבדנות שבאמצעוף מאגר נתונים מקורי, והציגתי את הקוד הקונקרטי שאפשר את אימון המודל. אני מוקוה שה מידע הנוסף זהה יתנו לצצה מעמיקה יותר לתהליכי הפיתוח, ויספק תוכנות שימושיות למי שירצה לחקור את הנושא לעומק או לבנות ישומים דומים בעתיד.

אני מוקווה שספר הפרויקט המפורט זהה, על כל חלקיו, נotent תמונה בהירה על העבודה שעשית, על האתגרים בהם נתקلت ועל הפתרונות שאתה מצאת. הוא משקף נאמנה את המשע שuberati בחודשים האחרונים, ומהווע עדות לידע ולניסיונו שצברת. אני מודה לך על ההזדמנויות להציג את הפרויקט שלי בצורה מסודרת ומקיפה, ומוקווה שהצלחת להעביר את ההתלהבות והתשוקה שליו אוטי לאורך כל הדרך.