



שם מכללה: בנות - אלישבע | שם סטודנט: יהודית ארואס | ת.ז: 212012017

שם מנהה: גב' שולמית ברלין | תאריך הגשה: 15.06.2021



תוכן עניינים:

| | |
|------------|---|
| 4. | הצעת פרוייקט |
| 4..... | תיאור הפרויקט:..... |
| 4..... | הגדרת הבעיה האלגוריתמית:..... |
| 5..... | רקע תיאורטי בתחום הפרויקט:..... |
| 5..... | תהליכיים עיקריים בפרויקט:..... |
| 5..... | תיאור הטכנולוגיה..... |
| 5..... | בסיס נתונים:..... |
| 5..... | פרוטוקולי תקשורת..... |
| 6..... | לוחות זמנים:..... |
| 7. | תקציר/מבוא |
| 7..... | מבוא לפרויקט:..... |
| 8..... | תהליך המחבר:..... |
| 9..... | סקירה ספורת:..... |
| 10. | מטרות/יעדים |
| 10..... | מטרות הפרויקט:..... |
| 10..... | יעדי הפרויקט:..... |
| 11..... | אתגרים:..... |
| 13. | מדדי האצלהה למערכת: |
| 14. | רקע תיאורטי: |
| 14. | ניתוח חלופות מערכת: |
| 15..... | תיאור החלופה הנבחרת, כולל נימוקים לבחירתה:..... |
| 16. | אפיון המערכת שהוגדרה: |
| 16..... | ניתוח דרישות המערכת..... |
| 16..... | מודול המערכת..... |
| 17..... | אפיון פונקציונלי וביצועים עיקריים..... |
| 18..... | AILOCIM..... |
| 19. | תיאור ארכיטקטורה: |
| 19..... | הארכיטקטורה של הפתרון המוצע בפורמט שלחדר Top-Down level Design..... |
| 19..... | תיאור הרכיבים בפתרון..... |
| 19..... | ארכיטקטורת רשות..... |
| 19..... | תיאור פרוטוקולי התקשרות..... |
| 19..... | תיאור טכנולוגיות השרת והלקוח הנבחרות..... |



| | |
|-----|--|
| 19. | תיאור הצפנות |
| 20. | ניתוח ותרשים UML /Use cases של המערכת |
| 21. | רישימת Use cases: |
| 21. | תיאור ה-Use cases |
| 23. | תיאור מבני הנתונים בפרויקט |
| 23. | ✓ Tuples – לאחסון הפרימיטיבים לאחר הציגו, אחת מתכונות tuple שהיא immutable, תכונה זו מבטיח הגנה על הפרימיטיבים ומיקומם, לאחר העיבוד. |
| 24. | תרשים מודולים: |
| 24. | תרשים מחלקות – Class diagram – |
| 25. | תיאור המחלקות המוצעות – בצד ה - Server : |
| 25. | חלק א' - Algorithm |
| 28. | חלק ב' - API: |
| 28. | תיאור הקומponentות בצד ה – Client |
| 30. | תיאור התוכנה: |
| 31. | אלגוריתמים מרכזיים: |
| 60. | קוד התוכנית |
| 60. | קוד התוכנית בצד ה – Server |
| 60. | חלק א' - Algorithm |
| 70. | חלק ב' – API: |
| 72. | קוד התוכנית בצד ה – Client |
| 82. | מסד נתונים: |
| 85. | תיאור מסcis: |
| 91. | מדריך למשתמש: |
| 91. | בדיקות והערכה: |
| 93. | ניתוח ייעולות: |
| 93. | אבטחת מידע |
| 93. | מסקנות: |
| 94. | פיתוחים עתידיים – רעיונות לשיפור: |
| 94. | ביבליוגרפיה: |



הצעת פרויקט

מספר מוסד: 18908

שם מכללה: בננות אלישבע

שם הסטודנט: יהודית ארואו

ת"ז הסטודנט: 212012017

שם הפרויקט: זיהוי חכם של תמרורים ושלטים

תיאור הפרויקט:

הפרויקט הינו תוכנה לזיהוי תמרורים וסימני – דרך בכביש, המערכת תתריע בפני הנהג על משמעות/zיהוי./zיהוי מתבצע ע"י טכנולוגיית ראייה ממוחשבת – עובוד תמונה ורשותות נירונים תוך שימוש בספריות רלוונטיות.

הגדרת הבעה האלגוריתמית:

זיהוי התמרורים וסימני – הדרך בכביש./zיהוי יעשה באמצעות אימון רשת נירונית שתזהה תמרור ועפ"י התוצאה שתזהה תשלח התראה מתאימה למשתמש על משמעות/zיהוי.

תהליכי האימון יכולות:

* קבלת תמונות של תמרורים וסימני – דרך מסך נתונים.

* עבודה הנתונים המתקבלים: טיפול בرعשים, חילוץ אינדיקטורים, נרמול נתונים.

* צמצום מדדים

* לימוד המודל

* מבחן המודל – בדיקה בכמה אחוזים דיקה הרשות ושיפורה בהתאם עד לקבלת התוצאה האופטימלית.

במהלך בניית הרשות, נשמר תהליכי הלמידה של המחשב כתצוגה בה נוכל לצפות בכמה אחוזים מדויקת רשות שאימנו.



רקע תיאורתי בתחום הפרויקט:

כיום, במאה ה-21, אחד מגורמים המוות העיקריים הינו הקטל בדרכים. התנהגות הנהג בכביש, או כפי שנוהג לכנות "הגולם האנושי", אחראי ליותר מ 90% מהתאונות הדריכים.

אחד הגורמים העיקריים העיקריים להטופה זו הינו חוסר תשומת – לב הנהג לתמרורים וסמי – דרך בכביש בזמן נהיגה, בין אם זה תמרור המורה על חיוב האטה או תמרור המראה על נתיב עם תנוצה דו-סטרית או אסור הנסעה בנסיבות העולה על מספר הקמ"ש הרשות בתמרור ועוד.

התראעה לנוהג הינה טכנולוגית בטיחות מתקדמת היוכלה למנוע תאונות הנגרמות ע"י חוסר שימוש לב.

בנוסף לחברות רבות עוסקות על תחומיים רבים לרכיב האוטונומי והפרויקט הינו ייחידה קטנה במרקם הכללי.

תהליכיים עיקריים בפרויקט :

*התהיליך העיקרי הינו כתיבה ואימון רשת נירונים לזהות תמרורים וסמי דרך בכביש.

*קליטת התמונה ממצלמות המותקנות בצד' הרכיב שמשזהות תמרורים בכביש.

*עבוד הקלט באלגוריתם זיהוי התמונה.

*הציג הפלט - בדמות התראעה לנוהג על משמעות הזיהוי.

תיאור הטכנולוגיה

צד שרת - שפת תכנות מצד השרת: C#, Python

צד לקוח - שפת תכנות מצד הלוקה: Html, Java – Script, React

מסד נתונים:

קובץ Excel בו יאוחסנו תמונות של תמרורים וסמי – דרך ומשמעותו של כל אחד מהם.

פרוטוקול תקשורת:

Http



לוחות זמנים:

- אוקטובר - תכנון הפרויקט
- נובמבר - למוד החומר לצורך האלגוריתם
- דצמבר - כתיבת האלגוריתם
- ינואר - נסוי ובדיקות
- פברואר - בניית פונקציות נוספות וממשק המשתמש (UI)
- מרצ - השלמות ופינישים
- אפריל - ספר פרויקט
- מאי - הציגת הפרויקט

חתימת הסטודנט: יהודית ארואו

חתימת רץ המגמה:

אישור משרד החינוך:



תקציר/מבוא

מבוא לפרויקט:

ישנם פרויקטים מסווגים שונים. ישנו-Calala שהחלק החזק בהם הוא הטכנולוגיה החדשה, הפרקטיקה או העיצוב. שבאתני לבחור נשא היה ברור לי שאני מעוניינת בפרויקט המכיל אלגוריתם מורכב לצד טכנולוגיה חדשה. רציתי שהוא יהיה אף חינמי ו שימושי.

במקביל רציתי להביא לידי ביטוי את הידע והכלים שרכשתי ביגעה הרבה לארוך תקופה הלימודים, עם זאת העמיך את הידע שלו בתחוםים חדשים נוספים שיתרמו לי כסטודנטית וכמהנדסת, בפרט בתחום הבינה המלאכותית (AI) – ענף מרתק שמצאת בו עניין רב.

לאחר שעות רבות של חשיבה בנוגע לבחירה שלו, בסופה של דבר החלטתי לפתח פרויקט בתחום הבינה המלאכותית המזהה תמרורים בזמן אמת בכיביש, ולפי התמרור המזוהה, המערכת תשלח התרעה מתאימה לנוהג על משמעותו.

כיום, במאה ה-21, אחד מגורמי המוות העיקריים הינו הקטל בדרכים.

התנהגות הנהג בכיביש, או כפי שנוהג לכנות "הגורם האנושי", אחראי ליותר מ 90% מהתאונות הדריכים.

אחד הגורמים העיקריים הגורמים לתופעה זו הינו חוסר תשומת – לב הנהג לתמרורים וסמי – דרך בכיביש בזמן נהיגה, בין אם זה תמרור המורה על חיזוב האטה או תמרור המראה על נתיב עם תנוצה דו-סטרית או אסור נסיעה במהירות גבוהה על מספר הקמ"ש הרשום בתמרור וכו'.

פרויקט זה בא לתת מענה לצורך החינוי בשמירה חי' אדם ולסייע לנוהג בניהga בטוחה בהרבה וביתר קלות, התרעה לנוהג היא טכנולוגית בטיחות היוכלה למנוע תאונות הנגרמות ע"י חוסר תשומת לב.

הפרויקט גם נכנס לתחום חשוב ומתאים היום "רכב אוטונומי" הרכיב האוטונומי אמרור להחליף לחלווטין את הנהג. זיהוי התמרורים באופן אוטומטי הוא אבן דרך נוספת ביצירת רכב אוטומטי شهرיה הרכב יזהה את התמרור ויכול לפעול ע"פ משמעותו.

הפרויקט משתמש במלחת רכב ומפעיל אותה באופן שוטף, לאורך כל הנסעה, כאשר תמרור זהה ע"י המערכת נשלחת לנוהג התרעה על משמעות התמרור שזוהה, הזיהוי מתבצע ע"י טכנולוגיית ראייה ממוחשבת – עובוד תמונה ורשתות נירוניות תוך שימוש בספריות רלוונטיות.



תהליכי מחקר:

התהילר המרכז התרץ בשאלת איך לזהות תמרור מיותר הסרטה?

כדי לפטור בעיה זו,

- ישנה אפשרות להשתמש באלגוריתמים שונים של עיבוד תמונה. אלגוריתמים אלה - עובדים על"פ העיקרון הבא - העברת התמונה למטריצה של פיקסלים והפעלת פעולות מתמטיות שונות. וכך- הכפלת כמות הפיקסלים, סיווג אובייקטים, זיהוי צורות, צבע ונקודות או רemarkable features, במקומות מוגדרים ועוד.

אר באופציה זו, ישנה בעיה מहותית והיא הדיווק, נניח שבמהלך הנסעה יהיה תמרור שחצי מוטושטש, על – פי האלגוריתם הנ"ל, לא נקבל זהה תמרור, כיון שהוא ינסה לחתך דגימה של הצורה שאמורה להיות במרכז התמרור, אך הדגימה לא תהיה נכונה כיון שחלק מהצורה מוטושטש ולא ברורה.

2. אפשרות נוספת, השווהה של התמונה המתבקשת לזיהוי עם שאר התמונות במאגר וכך זיהוי התמונה בהתאם.

מזכירות אבדית של לא קיימת במאגר. המרכיב לא תדע לzechot אונטן. שאינה נמצאת במאגר התמונות, הטענץ זיהוי שגוי. לדוגמא: התקבלה תמונת תמרור עם כל התמונות במאגר). שניית, השיטה איננה בטוחה כיון שם התקבלה תמונת תמרור אך אפרשות זו נפסלה, ראשית, כיון שאין שזו סיבוכיות זמן גבואה מאד (השוואה במקרה הגראע

אם כך, עלינו למצוא שיטה שתסייע את התזאה האופטימלית ביותר לחייו תמרורים בדרך.

3. על כן, האפשרות הנוסףת שנבחנה, הייתה זיהוי התמורות באמצעות בניית רשות נזירונים המשמשת בעקרון הבינה המלאכותית. והיא תיתן זיהוי תקין וכן גם במקרה קצה, כדוגמת המקרה שתואר לעיל, וזאת מפני **שאלמד** את הרשות לזהות תמרורים (כפי שיאסביר בהרחבה בהמשך).

תחום זה הינו ענף פורץ דרך במדעי המחשב, ומתבסס על הרעיון הבא: מודל מתמטי חישובי שפותח בהשראת תהליכי מח אדם. הרשות מכילה מספר רב של יחידות מידע מהוועדות



זו לזו. צורת הקישור בין היחידות, המכילה מידע על חזק הקשר, מדמה את אופן חיבור הנירוניים במוח.

תהליך השימוש בראשת כולל תהליך אימון אשר קובע את המשקלות לזיהוי אוטומטי של תבניות בראשת העצבית.

הפתרון הנבחר הוא כאמור, בניית רשת נירונית שתזהה תמרור, ובנוסף תדע לסוג למשמעו הנכונה..

היתרון שבבניה הרשת נירונית על – פni האלגוריתמים הנ"ל, היא יכולה **להקיש** מדוגמא אחת לחברתה, גם אם התמונה לא בדיק מכילה את כל הפרמטרים הנדרסים לזיהוי הנכון,

זאת ועוד האלגוריתמים הנ"ל, לא יודיעים לה קיש מקרה אחד לשנהו, כיוון שלומדים לזהות כל תמונה בפni עצמה או לחילוף לה השוואת התמונה המתקבלת למ Lager נתוני מוגדר, ואם אין את כל הנתונים הנדרשים לזיהוי הספציפי, הוא יזהה עפ"י הפרמטרים החלקיים שמצא, מה שהרבות המקרים ייתן תוצאה שגוייה.

לאחר החלטה על אופן פועלות האלגוריתם, יצרתי מספר רשומות, ובחנתי כל אחת מהן מי הרשת עם יכולת הדיק בזיהוי הגובה ביותר, והיא נבחרה לרשת שתזהה באופן אופטימלי את התמורות.

סקירת ספורות:

- / <https://github.com> - GitHub ✓
- <https://reshetech.co.il/python-tutorials/all-the-tutorials> - Python Tutorials ✓
- <https://www.ai-blog.co.il/category/deep> - AI – Blog ✓
- <https://opencv.org> - OpenCV ✓
- Mosh Hamedani – I learn Python & React ✓
<https://programmingwithmosh.com>
- <https://www.pyimagesearch.com> - Learn Deep Learning & OpenCV ✓
- Learn Machine Learning with Python ✓
- <https://www.murtazahassan.com/courses/machine-learning-with-python> ✓
- <https://reshetech.co.il/python-tutorials/pillow-imaging-library> ✓



- לימוד Python - <https://www.programiz.com/python-programming/methods/built-in/min> ✓
- רשת VGG16 - <https://keras.io/api/applications/vgg/#vgg16-function> ✓
- ספריית Keras - <https://keras.io> ✓
- ספריית TensorFlow - <https://www.tensorflow.org> ✓
- ספרית NumPy - [https://numpy.org/doc/stable/reference/generated/numpy.ndarray.tolist.html](#) ✓
- מדריך React - <https://he.reactjs.org/tutorial/tutorial.html> ✓
- .Wikipedia ✓

מטרות/יעדים

מטרות הפרויקט:

- ⊗ פיתוח תוכנה לחיי תמרורים וסימני – דרך בכיביש.
- ⊗ סיווע בנסיעה בטוחה וביתר קלות.
- ⊗ לימוד והטמונות בתחום ה – AI (בינה מלאכותית).
- ⊗ רכישת מיומנות גבוהה בשפת חסות Python תוך הכרה והתנסות בספריות חדשות ומשמעותיות.
- ⊗ התנסות בלמידת מכונה.
- ⊗ רכישת יכולת גבוהה של מידת עצמית בתחום מגוון.
- ⊗ למידה והכרה של תחום העיבוד תמונה.
- ⊗ ממשק נח, נעים וברור למשתמש, תוך הקפדה על נראות בסטנדרטים גבוהים ומקצועיים.

יעדי הפרויקט:

- ⊗ הכרת אלגוריטמים לעיבוד מוקדים של תמונות, כגון שינוי רזולוציה או התמרה אופינית של התמונה.
- ⊗ הכרת אלגוריטמי סיווג שיקבלו מאפייני תמונה, ויחלוito, במידה וקיים, באיזה תמורה מדובר מתוך קבוצה של תמרורים, אחרת, יחזירו شيئاً תමור.
- ⊗ תכנון המערכת תוך שימוש באלגוריתם יעל, כתיבה נכונה, מאורגנת ומڪוועית של הקוד.
- ⊗ איסוף מספר רב של תמונות ע"י צילום עצמי של תמרורים, וכן איסוף תמונות מתוך האינטרנט.
- ⊗ אימון מספר סוגים לרשות על מאגר התמונות תוך שיפור וייעול כל רשות עד לקבלת התוצאה האופטימלית שהיא יכולה להפיק.
- ⊗ בחירת הרשת הנותנת את אחוז הדיקט המרבי בזיכרון.
- ⊗ יצירת ממשק משתמש נח, ברור ומקצועי תוך שימוש בעיצוב יידידותי ונעים לעין.



- ⊗ שימוש בשפות מתקדמות מעולם הפיתוח.
- ⊗ הצגת ההתרעה לנוהג תוך שמירה על ערכנותו.

אתגרים:

מספראתגרים עמדו בפני בעת כתיבת הפרויקט, בעיקר בתחום לימוד נושא הבינה המלאכותית.

ראשית, למידת נושא הלמידה מכונה – תחום העוסק בפיתוח אלגוריתמים המאפשרים לאפשר למחשב ללמידה מתוך סדרות של נתונים, ופועל במגוון משימות חישוביות בין התכונות הקלאטי אינן אפשרי. למשל, בעית זיהוי שמוcharה אונשי מסוגל לפטור, אך לא מסוגל לכתוב את הכללים לזרחי בצורה מפורשת, או שהם משתנים עם הזמן ולא ניתנים לכתיבה מראש.

מטרת הלמידה יכולה להיות חיזוי או גילוי של עבודות. לדוגמה: עבור זיהוי תמרורים ניתן להשתמש בלמידה מכונה כדי לגלוות מהו התמരור המופיע.

הלמידה היה כרוך בחיפוש אחר חומר בנושא הלמידה – מכונה ולימודו באופן מקוון, בין היתר קורסים קצרים, קריאת מאמרים ומדריכים שונים והעמקה בנושא הרשות נוירונים, שהזחו האלגוריתם הייעיל והאמין ביותר, בעיקר בתחום העבודה תמורה בין האלגוריתמים האפשריים.

שנית, לצורך פיתוח המודול למדתי על ספריות קוד בפיתוח דוגמת:

:**TensorFlow** ✓
ספריית קוד פתוח ללמידה מכונה, לבניה ואיימון רשות עצביות.

:**Keras** ✓
המשק הידידותי ביותר ללמידה מכונה, למעשה הוא המשק ברירת המחדל של TensorFlow

:**NumPy** ✓
מאפשרת לעבוד עם מערכים רב-ממדיים, וספקת פונקציות מתמטיות לעבודה עם מערכים.

:**Matplotlib** ✓
הציג נתונים באמצעות תרשימים (חשוב כשםנתחים מיידי).

:**SciKit-Learn** ✓



ספרייה בעלת כלים פשוטים ויעילים לניתוח ובו נתונים, השתמשתי בספריה זו רק כדי לעבד את הנתונים הגולמיים לפני הלמידה מכונה שנעשית בפועל באמצעות הספרייה TensorFlow. ועוד.

אתגר נוסף היה איסוף התמונות, לצורך דיקט מקסימלי בתוצאות זיהוי הרשות יש לאמן אותה על מה שיוצר תמונות, לצורך כך אספתי מספר רב של תמונות ע"י צילום עצמי של תמרורים וכן מtower תמונות הקיימות באינטרנט. לבסוף, בניתי מחלוקת עם פונקציות לעיבוד תמונה, על – מנת לעמוד את מאגר התמונות עד שהגעתי לכמויות תמונות ש כוללת כמעט 23,000 תמונות!

וכן אתגר נוסף הייתה תהליכי משמעותי בפרויקט, היה בנית הרשות נירונים שתבחר לזרחי התמרורים. על – מנת להגיע לרשות הנonta תוצאה אופטימלית בזיהוי, בניתי מספר רשתות מסווגים שונים והרצתי כל אחת מהן מספר פעמים (בשינוי ערכים, לדוגמה: שינוי מספר ה epochs – כמה פעמים תרצו כל תמונה (ע"מ לכוון את המשקלות)). עד לקבלת התוצאה האופטימלית שהיא יכולה להפיק, ולאחר מכן בחירת הרשות הטובה ביותר מבין הרשותות שאימנתי – שנותנת את אחוז הדיקט הגבוהים ביותר בזיהוי.

אתגר נוסף, פיתוח הקוד בשפת Python שהצריך הרבה לימוד עצמי – הנטיה הראשונית שלי הייתה לפתח את הפרויקט בשפת C# – 'שפה - אם' (מבחינה תכנית ..), אותה למדתי קודם (וכן עבדתי אליה הרבה), אך דווקא בגלל זה רציתי לפתח בשפת Python – שפה שפותה מתמקדים בה במהלך הלימודים, להעמק בה, ולדעת אותה היטב וכך לרכוש לעצמי ידע והתקচות בשפה פופולרית וחזקה כיום בשוק.



מדד הצלחה למערכת:

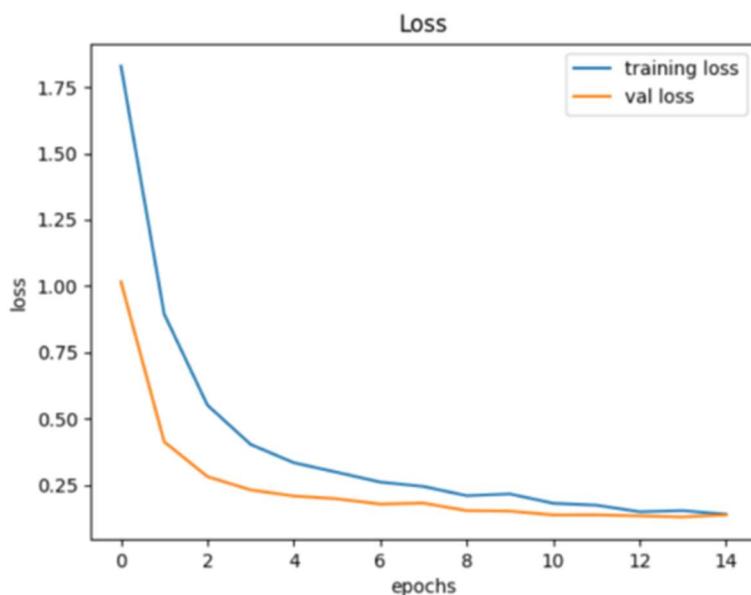
האתגר המשמעותי בפרויקט הינו ניסיון הבאת הרשות לאחוזי זיהוי אופטימליים.

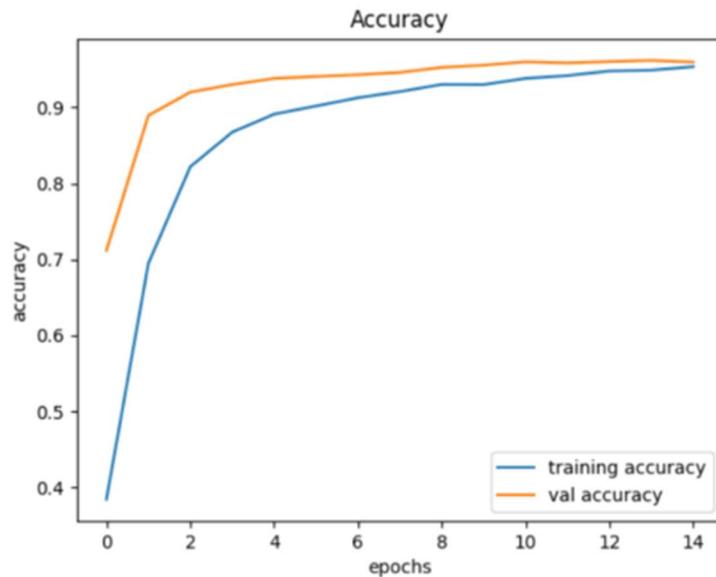
לצורך כך בנית רשותות שונות ולבסוף בחזרתי ברשת עלת אחוזי ההצלחה הגבוהים והמדויקים ביותר.

לאחר בדיקות והערכת הרשות שנבחרה הינה רשת מבודדת CNN שאחוז הזיהוי שלה הימם מעל 96% זיהוי!

Accuracy: 96.78%

בנוסף, ניתן לראות עפ"י הגרפים הבאים, את המודל המוצלח, שנוצר לאחר פיתוח ואיימון מסוף מודלים (כפי שיפורט בסעיף 'אלגוריתמים מרכזיים'), המתבטא ב – loss נמוך (loss = שגיאה), וב – fit good – מידת התאמה טוביה, ככלומר מידת שגיאה (loss) נמוכה ודוגמאות ביקורת טובות (training_loss) באותה מידה או קצת פחות מהמודל מאומן עליו (val_loss) (val_loss) נמוך מזה של קבוצת הביקורת (val_loss) נמוך מזה של קבוצת האימון (training_loss), שזה מצב הרצוי המעיד על התאמה טוביה.





רקע תיאורטי:

לאחר החלטה על אופן פעולה האלגוריתם לזיהוי התמרורים (cmpeqורת לעיל), ניתן היה לבנות סוגים שונים של רשתות ניירונים, המכילות שכבות שונות. התמקדת בשני סוג הרשתות הבאות,

רשת מבוססת **CNN**, Convolutional Neural Network, רשת המבוססת על מודל VGG16.

שאלו רשתות מבוססות שנקראו ע"י טובי המוחות בתחום הבינה המלאכותית לצורך פתרון בעית בסוג שונה.

במהלך בניית המערכת אמנתי כל אחת מהרשתות, תוך התמקדות בכל אחת מהן ושיפורה ע"י הוספת שכבות, הוספה בלוקי קובולציה, שינוי ערכי פרמטרים (cmpeqורת בהרחבבה בהמשך, בסעיף אלגוריתמים מרכזיים), עד לקבלת התוצאה האופטימלית שניתן להפיק מכל אחת. כאמור, בחרתי את הרשת בעלת אחוז הדיוק הגבוה ביותר.

ניתוח חלופות מערכתי:

ישנם מספר דרכים לפתרון הבעיה האלגוריתמית של זיהוי תמרורים,



ישנה אפשרות להשתמש באלגוריתמים שונים של עיבוד תמונה. אלגוריתמים אלה – עובדים ע"פ העיקון הבא – העברת התמונה למטריצה של פיקסלים והפעלת פעולות מתמטיות שונות. כגון – הכפלת כמות הפיקסלים, סיווג אובייקטים, זיהוי צורות, צבע ונקודות אוור במקומות מוגדרים ועוד.

אך באופציה זו, ישנה בעיה מהותית והוא הדיק, נניח שבמהלך הנסעה יהיה תמרור שחייב מטושטש, על – פי האלגוריתם הנ"ל, לא לקבל זיהוי תמרור, כיון שהוא ינסה לחתך דגימה של הצורה שאמורה להיות במרכז התמרור, אך הדגימה לא תהיה נכונה כיון שחילק מהצורה מטושטשת ולא ברורה.

אם כך, עלי למצוא שיטה שתביא את התוצאה האופטימלית ביותר לזיהוי תמרורים בדרך.

ועל כן, האפשרות שתתן זיהוי תקין וכן גם במקירה קצה, כדוגמת המקירה שתואר לעיל, הינה, זיהוי התמרורים באמצעות בניית רשת נוירונים המשמשת בעיקון הבינה המלאכותית. תחום זה מתבסס על הרעיון הבא: מודל מתמטי חישובי שפותח בהשראת תהליכי מוח האדם. הרשת מכילה מספר רב של יחידות מידע הקשורות זו לזו. רמת הקישור בין היחידות מכילה מידע על חזק הקשר, מדומה את אופן חיבור הנוירונים במוח.

תהליך השימוש ברשת כולל תהליכי אימון אשרקובע את המשקלות לזיהוי אוטומטי של תבניות ברשת העצבית.

טיור החלופה הנבחרת, כולל נימוקים לבחירתה:

הפתרון הנבחר הוא כאמור, בניית רשת נוירונים שתזהה תמרור, ובנוסף תדע להתריע לנאג על משמעותו בהתאם.

היתרון שבביהה הרשת נוירונים על – פני האלגוריתם הנ"ל, היא יכולה **להקיש** מודגם אחד לחברתה, גם אם היא לא בדיק מכילה את כל הפרמטרים הניצרכים לזיהוי הנכון, זאת בעוד האלגוריתם הנ"ל, לא יודע להקיש מקרוה אחד לשנהו, כיון שלמד לזהות כל תמונה בפני עצמה, ואם אין את כל הנתונים הנדרשים לזיהוי הספציפי, הוא יזהה עפ"י הפרמטרים החלקיים שמצא, מה שברוב המקרים ייתן תוצאה שגوية.



אפיון המערכת שהוגדרה:

नितוח דרישות המערכת

- ⊗ כתיבה בסטנדרטים גבוהים ומקצועיים, באופן מסודר תוך הקפדה על תיעוד ברור בכל שלב.
- ⊗ זיהוי גובה ומדדיק של התמരורים ע"י הרשת נוירונית.
- ⊗ שייהי בארכיטקטורת שרת - ל��וח.
- ⊗ חישוב וכנתיבת האלגוריתם בסביבות היילה ביתר.
- ⊗ בניית הממשק (UI) באופן מקצועי וכן שימוש חדשני וידידותי למשתמש.
- ⊗ ניתוח התוצאות באופן מהיר והציג הפלט.

מודול המערכת

חלק א': הכנת הנתונים הגלומיים כתשתית לקראת הלמידת מכונה.

1. איסוף תמונות.
2. עיבוי מאגר התמונות, ע"י פונקציות לעיבוד תמונה.
3. חלוקת התמונות שבמאגר לשולשה תיקיות: train, test, validate בראון רנדומלי, כאשר בכל תיקיה ישנו שלושה – עשר קטגוריות שונות של תמרורים.

חלק ב': בניית הרשות ואיומה על התמונות שבתיקיות הנ"ל.

חלק ג': תחילת הפרויקט ופעולתו מצד ה - Server.

1. עיבוד הסרטון שהתקבל לפריימים, ושליחת הפריים לזיהוי ברשות הנוירונים.
2. זיהוי קווארדינטות התמרור בפריימ, במידה וקיים, וחיתוכו בהתאם מתוך ה פריימ.
3. זיהוי וקלסיפיקציה של התמרור ע"י הרשת נוירונית.
4. שליחת מספר התמרור המתתקבל מזיהוי הרשות (במידה ואכן הופיע תמרור) לפונקציה שתקרא מתוך קובץ Excel את מאפייניו.
5. כתיבת השניה בה זהה תמרור בסרטון, וכן את המשמעות שלו לקובץ Log.
6. שליחת המידע שנכתב לקובץ -log - תמרורים זהוו במהלך העבודה, לצד ה – client, דרך ממשק API שבנית.



חלק ד': תהליכי הפרויקט ופעולותיו מצד ה – Client.

1. כניסה המשמש לאפליקציה ובחירה הפעולה הרצiosa:
 - 1) קבלת התראות (2) צפיה בתמוריים הנתמכים.
2. במידה ונבחרה האופציה הראשונה – קבלת התראות:
 - 2.1 העלאת סרטון של מהלך נסעה ע"י המשתמש.
 - 2.2 שליחתו לשרת, לצורך עיבוד.
- 2.3 הצגת הנתונים, שהתקבלו מהשרת (תוצאת העיבוד).
- 2.4 הפעלת הסרטון ע"י המערכת ובמקביל, קבלת התראות על שימושם התמוריים המופיעים במהלך הסרטון באופן ברור ונעים.
3. במידה והאופציה השנייה – צפיה בתמוריים הנתמכים נבחרה:
 - 3.1 הצגת התמוריים הנתמכים
 - 3.2 אפשרות של חיפוש אחר מידע על תמרור ספציפי ע"י הכנסת שם או קטgorיה מובקשים.

אפיקון פונקציונלי וביצועים עיקריים

חלק א': הכנת תשתיית למידת מכונה.

- processing – בחלוקת עבודה תמונה, תפקידה עיבוי מאגר התמונות, באמצעות שינויים קלים בהן.
- randomTheFile – בחלוקת ארגון הנתמכים, תפקידה לחלק את מאגר התמונות כולם לשולשה תיקיות train, test, validate, אשר כל אחת מהתיקיות מחלוקת לשולשה – עשר קטגוריות של התמוריים הנתמכים.

חלק ב': בניית הרשת.

- buildModel – האלגוריתם הראשי כולל, טעינת התמונות מתוך שלושת התקיות שנוצרו כנ"ל. בניית שכבות המודל עפ"י קרייטרוניים רצויים, וקימפול המודל. לאחר מכן, אימון המודל, וקבלת רשות ניירונים לזיהוי תמוריים.

חלק ג': תהליכי הפרויקט ופעולותיו מצד ה - Server.

- get_frame_video – עיבוד הידעו שהועלה ע"י המשתמש, הפונקציה מקבלת קובץ וידעו שהועלה ע"י המשתמש, וחوتכת אותו לפרויימים.



- Execute – זיהוי האם מופיע בפריטים תמרור, ואם כן את סוגו, ע"י הרשות ניירונית. קריית הנתונים על התמרור שזוהה מקובץ ה – Excel, ושליחת הצד ה – client. וכן כתיבת תיעוד של התמരורים שזוינו לקובץ log.
- getAlert dictionary – מחזירה המכיל מספר תמרור שזוהה במהלך הסרטון, ואת המאפיינים הקשורים אליו.
- upload_videoFile – פונקציית ממשק API, מקבלת את קובץ הideo שנשלח ע"י המשתמש, ושולחת אותו לעיבוד במחלקה הראשית, שתሪיצ את כל האלגוריתם.
- get_list –מחזירה את כל התמരורים ומאפייניהם, לצורך צפייה בתמരורים הנתמכים.
- get_filteredList –מחזירה רשימה של תמരורים, שנמצאו תואמים בשם או בקטgorיה אליה הם משתיכים, עפ"י בקשת המשתמש.

חלק ד': תהליך הפרויקט ופעולותיו העיקריות מצד ה – Client.

- changingFileSelection – העלת קובץ וידאו ע"י המשתמש.
- onClickSend – שליחת הקובץ הנבחר לשרת, לצורך עיבוד.
- showAlert – הצגת התראות.
- getTrafficSigns – הצגת התמരורים הנתמכים.
- onSearch – הצגת התמרור/ים עפ"י חיפוש המשתמש.

אילוצים

- על האלגוריתם לבצע את עיבוד הסרטון שהתקבל במרירות ולהתируע באופן מיידי.
- על הרשות ניירונית לנתן אחזוז דיק מספיק גובה ותקין.



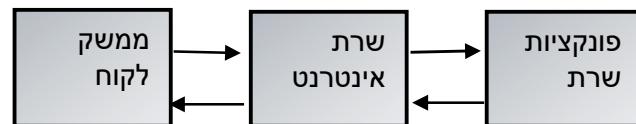
תיאור ארכיטקטורה:

הארכיטקטורה של הפתרון המוצע בפורמט של Top-Down level Design

תכנון הפרויקט נעשה מן המאקרו אל המיקרו, כלומר מן הכלל אל הפרט.

בתחילת תכנון הפרויקט נעשה באופן כללי, ובכל שלב ירדתי פנימה, יותר לפרט ולהשיבה מעמיקה יותר, עד לביצוע מלא ומקיף של הפרויקט.

תיאור הרכיבים בפתרון



ארכיטקטורת רשת
לא רלוונטי.

תיאור פרוטוקולי התקשורת
Http

תיאור טכנולוגיות השרת והלקוח הנבחרות

הפרויקט פותח **בצד שרת – Server**:

בשפת Python עם תוכנת PyCharm, גרסה 3.7, עם טכנולוגיית Flask - ספרייה ב – Python המשמשת סביבה לפיתוח Web ולקוח ה- API, בשפת Python.

בצד לקוח – Client:

פותח בשפות Html, CSS, JS, CSS, HTML, עם תוכנת React, VisualStudioCode, TypeScript.

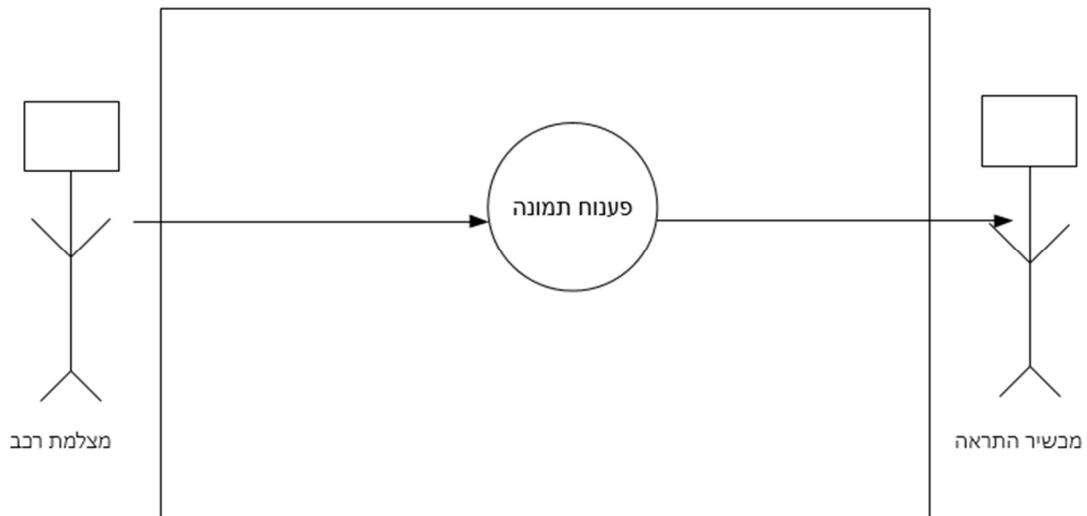
תיאור הצפנות
לא רלוונטי.



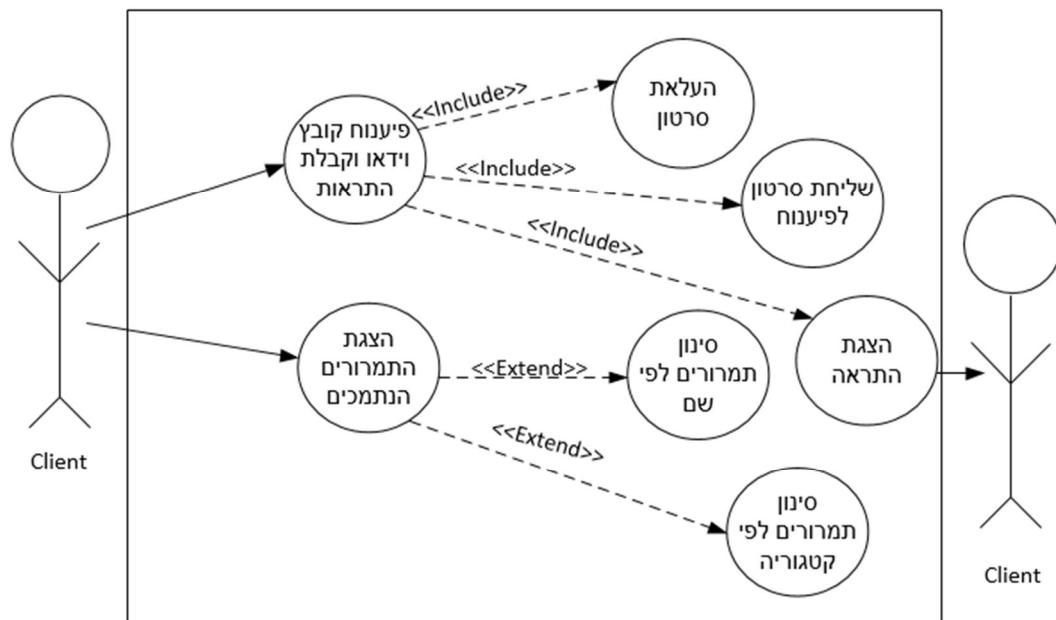
ביתוח ותרשים Use cases/UML של המערכת

המערכת שבנית, תופעל ברכב ועל מנת להמחיש אותה בהגשת הפרויקט הקמתי מערכת שתציג את פעילותה. אשר על כן, אציג כתה שני תרשימי use case הראשון – יציג את המערכת כפי שהיא במציאות, והשני – יציג את המערכת כפי שתוצג בהגשת הפרויקט,

תרשים ה UML הראשון – מציג את המערכת כפי שהיא במציאות



תרשים ה UML השני – מציג את המערכת עבור הגשת הפרויקט



כעת, אפרט עבור תרשيم ה UML השני הממחיש את פעילות המערכת עבור הגשת הפרויקט.

רישימת Use cases

1. UC1 - פיענוח קובץ וידאו וקבלת התראות.

2. UC2 - הצגת התמורות הנתמכים.

תיאור ה-Use cases

:UC1

Name: פיענוח קובץ וידאו וקבלת התראות. ⊖

Identifier: UC1 - פיענוח קובץ וידאו וקבלת התראות. ⊖

Description: פיענוח קובץ וידאו וקבלת התראות על משמעותם התמורות המזוהים בסרטון. ⊖

כאן מומוש האלגוריתם הראשי של הפרויקט, על – מנת להתריע על משמעות התמורות יש לetest את הסרטון שהועלה ע"י המשתמש, זאת באמצעות השלבים הבאים:



1. חיתוך הסרטון לפרייםים (frames).
2. שליחת הפרייםים לזיהוי ופיענוח ברשת נירוניים.
3. קריאה ממוקור נתונים על מאפייניו של מספר התמരור שזזהה.
4. שליחת המאפיינים לצד הלקוח, והציגם למשתמש באופן ברור ונעים.

.Client- initiator :**Actors** ⊖

:**Frequency** ⊖

.: **Pre-conditions** ⊖

⊖ **Post-conditions** : המשמש מקבל במהלך נסעה. המשמש את המשמעות התמוריים המזוהים, וכן סיכום בקובץ Log המכיל את משמעויות התמוריים שזזו במהלך הנסעה.

----- : **Extend Use Case** ⊖

:**Included use case** ⊖

1. העלאת סרטון – בחירת קובץ ידאו.

2. שליחת סרטון לפיענוח – תבצע שליחת של קובץ זה לשרת.

3. הציג התראה – לאחר עיבוד הסרטון מצד שרת, תופענה ההתראות מצד הלקוח.

.: **Assumptions** ⊖

⊖ **Basic course of action** : לאחר העלאת הסרטון, המערכת תבצע זיהוי והתרעה של התמוריים המופיעים בו.

.: **Change history** ⊖ Version1 ע"י יהודית ארואס.

⊖ **Decisions** : בגרסת הבאה האפקטיבית תשודרג, ויורחבו התמוריים בהם המערכת תומכת.

בנוסף, המערכת תוכל להתריע על מרחק לא בטוח בין המכונית הנוכחיות לאחרות.

וכן תדע לזיהות ולהתריע על הולך רגל הנמצא בטווח הקרוב של המכונית, כך שאם חלילה הולך רגל התפרץ לכביש, המערכת תתריע ע"כ לנаг, שיוכל לבLOW בזמן ולמנוע תאונה.

ובכך תבטיח תוספת בטיחות והקלת באופן ישיר לנаг, ובעקיפין גם להולך הרגל.

:**UC2**

:**Name** ⊖ הציג התמוריים הנתמכים.

:**Identifier** ⊖ UC2 הצגת התמוריים הנתמכים.

:**Description** ⊖ הציג התמוריים ומאפייניהם בהם המערכת תומכת.



בנוסף, יכול המשתמש לחפש מידע על תמרור ספציפי ע"י הכנסה שם תמרור או קטgorיה רצiosa לתיבת ה – **אקטואן המתאימה**, ולצפות רק בתמורות המעניינים אוטו.

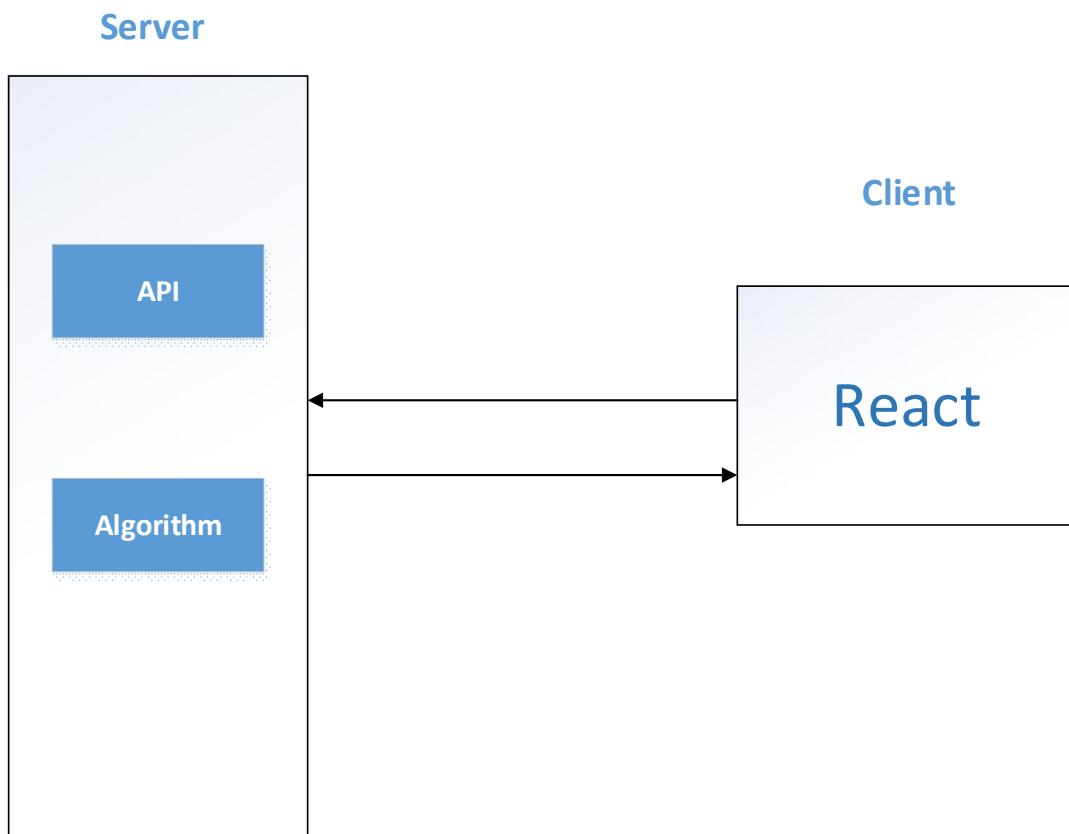
- .Client- initiator :**Actors** ⊖
 - :**Frequency** ⊖
 - ⊗ **Pre-conditions**: כתיבה של שם תמרור או קטgorיה רצiosa לתיבת ה – **אקטואן**, ובלחיצה על מקש ה – Enter יציגו התמורות הרצויים.
 - ⊗ **Post-conditions**: טבלה עם התמורות הנתמכים או התמורות המתאימים לאופן הסינון בהתאם.
 - ⊗ : **Extend Use Case** ⊖
1. סינון תמרורים לפי שם.
 2. סינון תמרורים לפי קטgorיה.
- :**Included use case** ⊖
 - ⊗ **Assumptions**: לצורך סינון על המשתמש להכנס שם תמרור או קטgorיה באיות נכון.
 - ⊗ **Basic course of action** ⊖
(מצגת התמרור/ים הרצוי/ים (באם התבצע חיפוש על תמרור ספציפי) והודיע על כל אחד מהם).
 - ⊗ **Change history** ⊖
Version1 ע"י יהודית ארואו.
 - ⊗ **Decisions** •
ביצוע חיפוש אחר מידע על תמרור, ע"י מצגת תמונה התמרור הרצiosa.

תיאור מבני הנתונים בפרויקט

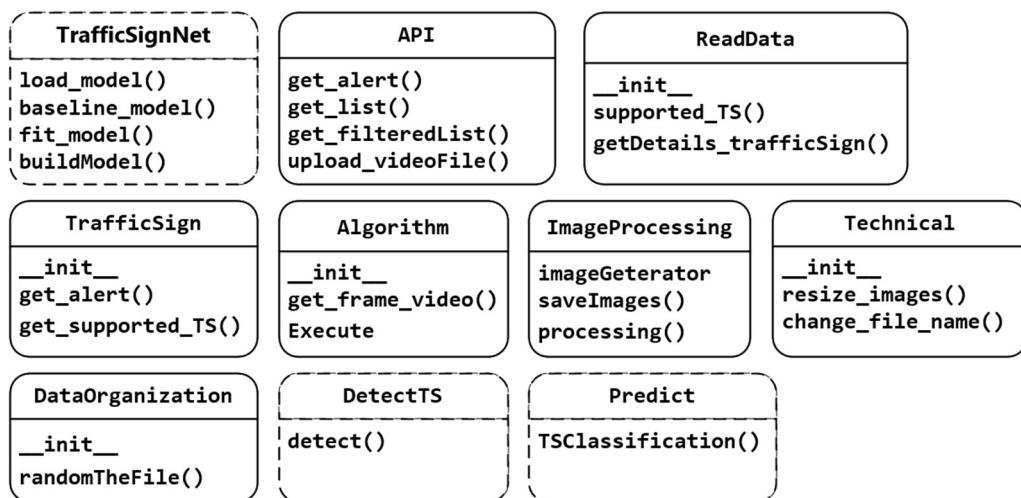
- ✓ **Tuples** – לאחסן הפרימיטיבים לאחר הזיהוי, אחת מתכונות tuple שהוא **tuple** שאינו מutable, תכונה זו מבטיחה הגנה על הפרימיטיבים ומיקומם, לאחר העבודה.
- ✓ **Dictionary** – לאחסן מאפייני התמרור החוזרים ממחלה קריאה מ – Excel.
- ✓ **רשימות** – אחסון הפרימיטיבים שהתקבלו לאחר עבודה סרטון, אחסון תמונות לאחר העבודה ועוד.
- ✓ **מטריצות** – אחסון תמונות שונות במטריצה, על – מנת שאוכל להפעיל עליהם כל מיני פעולות, כגון עבודה התמונה ועוד.



תרשים מודולים:



תרשים מחלקות – Class diagram :





תיאור המחלקות המוצעות – בצד ה - Server:

כעת אכנס ועמיק בפרוט המחלקות ורכיביהם המופיעים בתרשימים:

חלק א' - Algorithm:

1. מחלקה TrafficSignNet . מחלקה:

תפקיד המחלקה: המחלקה העיקרית בפרויקט, ותפקידה פיתוח ואימון המודל, עד לקבלת רשת נירונית חזקה ומדויקת.

פונקציות המחלקה:

1. `dataset load_model` – הפונקציה טעונה את תמונות ה - dataset מתוך שלושת התיקיות `train`, `test`, `validate` (כפי שיפורט בהמשך במחלקה `DataOrganization`.) וכן מבצעת נרמול נתונים לתמונות.
2. `baseline_model` – שלב פיתוח וקימפול המודל, הפונקציה מחזירה מודל המורכב מהשכבות שהתווסף בשלב הפיתוח.
3. `fit_model` – שלב אימון המודל שפותח, הפונקציה מחזירה רשת נירונית בעלת משקלות שנוצרו במהלך הלמידה.
4. `buildModel` – הפונקציה מפעילה את שלושת הפונקציות הנ"ל. (יפורט בהמשך בהרחבה, בסעיף אלגוריתמים מרכזיים).

2. מחלקה TrafficSign . מחלקה:

תפקיד המחלקה: מחלקה תמרור, מזמנת את הפונקציות של מחלקה `ReadData` שקוראת את הנתונים על התמורות ממקור מקור הנתונים (כפי שתפורט בהמשך) ומחזירה את הנתונים על התמרור/ים בהתאם.

פונקציות המחלקה:

1. `getAlert` – הפונקציה מקבלת מספר תמרור, מבצעת קרייה לפונקציה `getDetails_trafficSign` (ש�포רט במחלקה `ReadData`) ומחזירה את כל הנתונים לגביו.



.2 supported_TS – הפקציה מבצעת קריאה לפונקציה get_supported_TS (ש时报וט במחלקה ReadData) ומחזירה נתונים על כל התמורות הנתמכים.

3. מחלקת DataOrganization

תפקיד המחלקה: חלוקת מאגר התמונות (לאחר עיבוד והגדלה), לשולשה תיקיות: Train, Test, Validate

פונקציית המחלוקת:

.1 randomTheFile – הפקציה מקבלת את מאגר התמונות וביצעת חלוקה רנדומלית של התמונות לשולשת התקיות כנ"ל. כר ש – 70% מהתמונות ימצאו בתיקיית Train, 20% בתיקיית Test, ו – 10% בתיקיית Validate. (פרוטו מורחב בסעיף אלגוריתמים מרכזים).

4. מחלקת Algorithm

תפקיד המחלקה: מחלקת ראשית, מפעילה את כל תהליך המערכת. המחלקה מקבלת את קובץ הויידאו שהתקבל לעיבוד, וביצעת את עיבוד הסרטון הכלול את שליחת הפריים לזיהוי וסוג במידה וקיים תמרור בפריים. והזרת של התמורות שזוועו במהלך הסרטון.

פונקציות המחלוקת:

.1 get_frame_video – הפקציה מעבדת את הסרטון וחותכת אותו לפריים, ומחזירה רשימה המורכבת מהפריים. הערה: מכל שנייה הסרטון ישנים 24 פריים, ארצה לבדוק כל חצי שנייה את מצב הכיביש, ולכן רשימת הפריים המקורית אחזר רשימה שמורכבת מ- 2 פריים עבור כל שנייה.

.2 Execute – הפקציה מפעילה את כל המערכת, היא מקבלת את רשימת הפריים שהוחזרה מהfonkzia הנ"ל, ושולחת אותה לזיהוי. בנוסף, הפקציה כותבת את משמעות הזיהויים של התמורות (omidu נוסף לגביהם) לקובץ log.



5. מחלקה DetectTS

תפקיד המחלקה: זיהוי התמרור המופיע בפ輻ים, וחיטוכו מתוך הרקע.

פונקציות המחלקה:

1. detect – הפקzia מזזה באמצעות אלגוריתם סופי וספרית OpenCV,

את התמרור המופיע בפ輻ים (במידה וקיים) ומהזירה תמונה חדשה המכילה

רק את התמרור, ללא רקע. (פרוטו מורחב בסעיף אלגוריתמים מרכזים)

6. מחלקה ReadData

תפקיד המחלקה: קריית נתונים מתוך קובץ Excel.

פונקציות המחלקה:

1. supported_TS – הפקzia מקבלת קובץ Excel המכיל נתונים ומידע על

התמרורים הנתמכים במערכת (כמפורט בסעיף מסד נתונים)

וקוראת את הנתונים מתוכו.

הפקzia מהזירה רשימה המכילה את כל הנתונים על כל התמרורים.

2. getDetails_trafficSign – הפקzia מקבלת מספר מזזה של תמרור,

וקוראת את הנתונים השייכים לתמרור זה מתוך קובץ ה - Excel.

הפקzia מהזירה dictionary המכיל את כל הנתונים והמאפיינים של תמרור זה.

7. מחלקה ImageProcessing

תפקיד המחלקה: עיבוד תמונה לtresuna חדשה באמצעות שינוי פרמטרים שונים

בתמונה המקורית.

מטרת המחלקה היא הרחבת והגדלה של כמות התמונות במאגר, הכנה למידת המcona.

פונקציות המחלקה:

1. imageGenerator – הפקzia מעבדת את התמונה שהתקבלה, למספר

תמונות רצוי, ע"י שינוי פרמטרים של התמונה המקורית כגון: סיבוב התמונה,

שינוי חdots ובהירות התמונה ועוד.

הפקzia מהזירה מספר תמונות שונות.



2. saveImages – הfonקציה נוטנת גודל אחד לרשימת התמונות שছזרה מהfonקציה imageGenerator, ושמרת אותם.
3. processing – הfonקציה מפעילה את שני הfonקציות הנ"ל עבור כל תמונה במאגר התמונות.

8. מחלוקת Predict

תפקיד המחלוקת: 'חיזוי' הפריים שהתקבל באמצעות הרשת ניירונים.

fonקציית המחלוקת:

1. TSClassification – הfonקציה מקבלת רשימת תמונות (לאחר חיתוך הפריים) [במידה ולא זהה תمرור בפריים, הוא נשלח ישירות להfonקציה TSClassification ללא עיבוד מקדים].
וכן, טוענת את הרשת ניירונים שנוצרה בחלוקת TrafficSignNet, ועפ"י
משמעות הרשת מבצעת קלסיפיקציה עבור כל פריים, ומחזירה רשימה עם
מספרי התמורות שזויה.

חלק ב' - API:

תפקידו הינו שליחת המידע המתבקש לצד ה-client של המערכת ולהפוך, מਮמשת את התקשרות בין ה-client וserver.

fonקציות:

1. get_alert – הfonקציה מקבלת מספר תמרור ומחזירה נתונים על התמרור על פי המספר שהתקבל.
2. get_list – הfonקציהמחזירה נתונים על כלל התמורות.
3. get_filteredList – הfonקציה מקבלת שם תמרור או קטגוריה, ומחזירה נתונים על התמורות/ים שנמצאו תואמים בקטgorיה או בשם.
4. upload_videoFile – הfonקציה מקבלת קובץ וידאו, שנשלח ע"י המשתמש, שולחת אותו לעיבוד, ומחזירה את רשימה המכילה את התמורות שזויה במהלך הסרטון.

תיאור הקומפוננטות הצד ה-client



cut apart על מבנה הקומפוננטות הצד הלקוח:

.1. – Main component – הקומפוננטה הראשית.

תפקיד הקומפוננטה: מסך הפתיחה משמש כנקודת כניסה לאפשרות הפונקציונליות של המערכת.

אפשרות אחת, הינה העלאת קובץ וידאו המכיל מהלך נסעה, שליחתו לשרת, וקבלת התראות על שימוש התמוריים המזהים בסרטון.

אפשרות נוספת, היא צפיה במאפייני התמוריים הנתמכים ע"י המערכת.

.2. – קומפוננטת ההתראות.

תפקיד הקומפוננטה: שליחת קובץ הוידאו, שהועלה ע"י המשתמש לשרת, ולאחר עיבוד הקובץ, הצגת ההתראות לנגן.

פונקציות הקומפוננטה:

1. changingFileSelection – בעת לחיצה על כפתור בחירת קובץ, מופעלת פונקציה זו, ומתחילה בתוך משתנה דינאמי את הקובץ שהועלה.

2. onClickSend – בעת לחיצה על כפתור שליחה לשרת, מופעלת פונקציה זו. תפקידה שליחת הקובץ הנבחר לשרת, וכן קבלת ה-`response` מהשרת לאחר העיבוד, והפעלת הפונקציה `setInterval`.

3. setInterval – הפעלת `interval` שיקרא לפונקציה שמציגה את ההתראות על התמוריים המופיעים בסרטון כל חצי שנייה. ועיצוב האינטראול כשמגיעו סוף הסרטון.

4. showAlert – הפונקציה נקראת ע"י ה – `interval` כל חצי שנייה ומציגה את ההתראה במידה וקיימים שלושה זיהוי תמרור שוויים באופן רצוף (מהראשימה שחרזה לאחר העיבוד, ובها מספר התמוריים שזוהו). כזכור כל זיהוי מורכב מחצי שנייה בסרטון).

5. getAlertToDriver – פונקציית API המקבלת מספר מזהה של תמרור, ומחזירה את כל המאפיינים עליו.

.3. – קומפוננטת הצגת התמוריים הנתמכים.



תפקיד הקומפוננטה: הצגת התמרורים ומאפייניהם. כגון: משמעות התמרור, הקטגוריה אליה הוא משתייך ועוד.

פונקציות הקומפוננטה:

1. componentDidMount – פונקציה מובנית המופעלת לפני פונקציית ה – render (ג"כ מובנית), שתפקידה להציג את המשתנים והנתונים השונים על המסך, לפיכך הפעלתה בה את פונקציית getTrafficSigns, שהנתונים עליהם התמרורים (שחזרים מהfonkzia API) יהיו זמינים כאשר הטללה תוצג על גבי המסך.
2. getTrafficSigns – כאמור לעיל, פונקציית API המחזירה את המאפיינים של כל התמרורים.
3. getTableByQuery – פונקציית API המקבלת שם תמרור או קטgorיה לחיפוש, ומחזירה את התמרור הראשי או לחילופין את כל התמרורים עם הקטgorיה התואמת לחיפוש.
4. onSearch – הפונקציה תציג את התמרור/ים, עפ"י חיפוש המשתמש.
5. search_ts – הפונקציה הזאת, בלחיצה על מקש Enter תשליח את תוכן תיבת החיפוש לפונקציה onSearch.

תיאור התוכנה:

סביבת פיתוח:

חומרה: מעבד i7, RAM 8GB.

עמדת פיתוח: מחשב Intel.

מערכת – הפעלה: Windows 10.

הפרויקט פותח בגרסת Python 3.7

שפות תוכנה: **צד שרת:** פותח בשפת Python עם טכנולוגיית Flask.

עם תוכנת PyCharm, גרסה 3.7.

צד לקוח: פותח שילוב זה מאפשר בטכנולוגיית React.

עם תוכנת VisualStudioCode.

חיבור לאינטרנט: לא נדרש.

דפדפניים: Explorer, Chrome



אלגוריתמים מרכזיים:

האלגוריתם המרכזי נבנה על שלושה שלבים עיקריים:

1. ראשית, על - מנת להביא את הרשות לאחזוי זיהוי אופטימליים, עליה להתאמן על כמה - שיטור נתונים.

אחת הדרכים **להגדלת כמות התמונות למידת מכונה** הינה באמצעות גישה של **.augmentation**.

ספריית Keras מציעה את הפונקציה **ImageDataGenerator** שיכולה להפיק תמונות מגוונות ו��וות על בסיס תמונה מקורית שאספוק לה. התמונות המגוונות באוט לידי ביטוי בשינויים ויזואליים כגון חdots התמונה, בהירותה ועוד.

וכך באמצעות פקודה זו הכפלתי את מאגר הנתונים המקורי שלו.

חשוב לציין שבין אוניות השיני בתמונה כמעט ולא יORGש, אך BRAהיה ממוחשבת זהו SHINI גדול, ולמעשה תמונה שונה לגמרי. זאת מכיוון SMIKOMI הפיקסלים וצבעם משתנה לגמרי מהתמונה לתמונה וbraהיה ממוחשבת זהה לתמונה שונה.

להלן אופן הביצוע:

הספריות הנדרשות:

```
import os
from pathlib import Path
import numpy as np
from PIL import Image as im
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
from skimage import io
from skimage import transform
from ProjectNeurelNetwork.BL.Constant import Constants
```



אוף הפעלה:

```

class ImageProcessing:
    """Class processing images."""

    def imageGenerator(self, imagePath):
        """
        Class method processing an image.

        Parameters:
        imagePath (str): Image path for processing.

        Returns:
        list: a list of processed images.
        """

        image_gen = ImageDataGenerator(
            # הParamטרים שאיתם נשים את התמונות. וונלן בלאו גראן
            # טווח בירוב התמונות
            rotation_range=Constants.rotationRange,
            # טווח גודלת התמונות לבודה
            width_shift_range=Constants.widthShiftRange,
            # טווח גודלה התמונות לאורה
            height_shift_range=Constants.heightShiftRange,
            # טווח סיבת תבניות התמונות
            brightness_range=[Constants.fromBrightnessRange, Constants.toBrightnessRange],
            # טווח מידת חסיגון/זום של התמונות
            zoom_range=[Constants.fromZoomRange, Constants.toZoomRange],
            # אייד מתמגן גודלה מוחזק גירען
        )

        # אייד מתמגן גודלה מוחזק גירען
        fill_mode="nearest",
        # הפיכת גזרה של התמונות
        horizontal_flip=False,
        # הפיכת גזרה של התמונות
        vertical_flip=False,
        # הפיכת התמונות למתחמת עם גוונים מודפסים של אור/闇ה לא ברורה
        samplewise_center=False
    )

    # וידוא שנטיב התמונה שתהיה, אכן כן יתגלה
    img_path = imagePath
    my_file = Path(img_path)
    if my_file.is_file():
        print('OK')
    else:
        print('No image')
    # ביציאם התמונה מודפסת והפיכת מודר התמונה למשמע צבעה הטעון
    img = np.expand_dims(io.imread(img_path), 0)
    # ציירם גזרה של התמונות גודלה מוגדרת
    aug_iter = image_gen.flow(img)
    # גזרת מספר התמונות מרצוני
    aug_images = [next(aug_iter)[0].astype(np.uint8) for i in range(Constants.multiple_images)]
    return aug_images

```



כעת, מגוון התמונות מאוחסנות ברשימה `.aug_images`:
דוגמת הרצה מהטהלי'ך שנעשה בפונקציה על תמונה, לפני ואחרי העיבוד:

לפני:



אחרי:



פונקציה נוספת מחלוקת, השומרת את התמונות החדשנות



```

def save_images(self, images, picture_name, count=1):
    """
    Class method save the list of images processed.

    Parameters:
    images : The list of images processed.
    picture_name (str): The name given to the list of images.
    count (int): Number of images in the list (default is 1)

    Returns:
    none.
    """
    for r in range(count):
        # מודifies each image in the list
        data = im.fromarray(images[r])
        # R,G,B
        # הפיכת כל התמונה שוגר לוגדש בReLU 3 מרכיבים
        images[r] = transform.resize(images[r], (
            Constants.image_size, Constants.image_size, Constants.channels))

        # Generating thumbnail and watermark on it using pillow
        data.thumbnail((Constants.image_size, Constants.image_size), im.ANTIALIAS)

        # Generating a new background with size img_w
        img_w, img_h = data.size
        background = im.new('RGB', (Constants.image_size, Constants.image_size),
                            (Constants.backgroundColor, Constants.backgroundColor,
                             Constants.backgroundColor))

        bg_w, bg_h = background.size
        offset = ((bg_w - img_w) // 2, (bg_h - img_h) // 2)
        # This is where we paste the image to the background
        background.paste(data, offset)
        # Generating name for the image
        namepic = picture_name + str(r) + ".JPG"
        # Saving the generated image
        background.save(namepic)

```

כל זאת כהכנה תשתיית למידת המכונה.

2. שלב הבא, הינו זיהוי מיקום וקואורדינטות התמரור מתוך התמונה (הפריים) המתקבלת, על ידי שימוש באלגוריתם *soy* וספריית *OpenCV*, המזהה תמרור מתוך תמונה, לאחר מכן חתכים את התמരור המזוהה.



להלן אופן הביצוע:

הספריות הנדרשות:

```
import cv2
import numpy as np

from ProjectNeurelNetwork.BL.Constant import Constants
```

אופן הפעולה:

```
class DetectTS:
    """Identifying the sign from the background of the image."""

    @staticmethod
    def detect(list_pic):
        """
        Static class method. Identifying the sign from the background of the image,
        and cropping the identified sign.

        Parameters:
            list_pic (list): List of frames after video processing.

        Returns:
            The folders train, test and number of categories to classify.
        """

        classNames = []
        pre_img = []
        # מזהה yolo - הינה שמות הlayerים שאנו צריכים
        configFile = Constants.classFile
        with open(configFile, 'rt') as f:
            classNames = f.read().rstrip('\n').split('\n')
        # מזהה configPath - מזהה configPath
        configPath = Constants.configPath
        # מזהה weightsPath - מזהה weightsPath
        weightsPath = Constants.weightsPath

        # טרינינג בוצוי וחרוגים ומשקלים
        net = cv2.dnn_DetectionModel(weightsPath, configPath)
        net.setInputSize(320, 320)
        net.setInputScale(1.0 / 127.5)
        net.setInputMean((127.5, 127.5, 127.5))
```



```

net.setInputSwapRB(True)

# עבור כל פוטרים ברשימה הפריטים שמתגבלם
for pic in list_pic:
    # הגדרת הפרמטרים ליזיינו
    classIds, confs, bbox = net.detect(np.asarray(pic), confThreshold=0.5)
    # אוחזת הרשימה עם הפריטים שמתגבלם, בקרה של זיהוי
    # הפריטים המתגבלים בדרך ע"י הפריטים לאחר הדיווח (פזרט בתמונה).
    pre_img.append((pic, False))
    # בחרת פוטו אובייקט
    if len(classIds) != 0:
        # ארכז בזיכרון על פני שלושת מושימות שחוגדרו לפoto (כל חאובייקטים שנמצאו בתמונה).
        for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
            # מרכיב שזמה אובייקט מסeq 13 (מפה)
            if classId == 13:
                # אשמoga את גונדריניות תמגרוב הנקהן שזומן
                left = box[0]
                top = box[1]
                right = box[2] + box[0]
                bottom = box[3] + box[1]
                # נחתון נפ"י, גונדריניות תמגרוב שזומן
                crop_image = pic.crop(left, top, right, bottom)
                # מנג את תמונה צמלה רב מפה אבוג מפה תיבת
                np_img = np.array(crop_image)
                # אחילו את מזגום הפלרים שזומן בו מפה, לפוטום מתהנו (אם מפה תמגרוב בלבב)
                pre_img[-1] = (np_img, True)
return pre_img

```

הסבר מוחרב:

`classIds, confs, bbox = net.detect(np.asarray(pic), confThreshold=0.5)`

הגדרת הפרמטרים ליזיוי, במקרה של זיהוי אובייקט בוודאות של 50% לפחות, נשמר את מספר האובייקט שזזה, את אחוז הזיהוי ואת קואורדינטות האובייקט בתמונה.

-**ClassId**- רשימה של מספרי האובייקטים שזזהו.

confs- רשימה של אחוזי הזיהוי.

bbox-רשימה של קואורדינטות האובייקטים שזזהו.

confThreshold – רק במקרה של 50% זיהוי יש להתייחס לאובייקט.

`for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):`

נród בלולאה על פני שלושת הרשימה שהוגדרו לעיל (כל האובייקטים שנמצאו בתמונה).



```
if classId == 13:  
    # אושטוג את קואורדינטות התמרור חנוכתן שזורה  
    left = box[0]  
    top = box[1]  
    right = box[2] + box[0]  
    bottom = box[3] + box[1]  
    # נחתוך עפ"י קואורדינטות התמרוג שזורה  
    crop_image = pic.crop((left, top, right, bottom))  
    # גמיב את התמונה למשהה רב מימין עבור המשג חביבות  
    np_img = np.array(crop_image)  
    # אחילנו את מיקום הפרוייקט שזורה בו תמרוג, לפיראים חתוג (עם תומנת התמרוג בלבד)  
    pre_img[-1] = (np_img, True)
```

במידה והאובייקט הנוכחי הוא בעל מספר 13 (– מספר זה מבטא זיהוי תמרור),

נחותר את התמרור מתוך התמונה עפ"י קואורדינטות התמרור בתמונה.

ונשמר את הפיראים החדש המכיל את תומנת התמרור בלבד.

[נדרש את הפיראים הקודם, עם התמרור בלבד רקע].

דוגמת הרצה (– מתוך סרטון, בזמן אמת):



לאחר זיהוי התמרור:





לאחר חיתוך התמרור המזוהה:

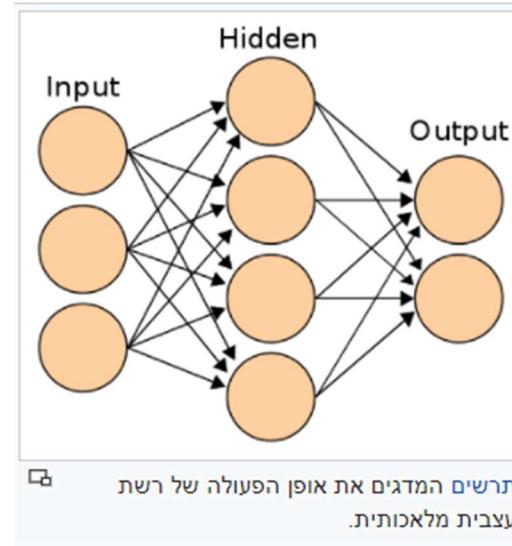


3. החלק הבא והעיקרי בפרויקט הינו סוג התמונה (הפרוייקט) המתקבלת, על ידי הרשות ניירונית וועוגה בהתאם.

על – מנת להגיע לאחוזי הזיהוי המרביים, הפעלתו על מאגר התמונות (שהוגדל כנ"ל, ומכל כרגע מעל 23,000 תמונות!), מודלים שונים של רשותות ניירונית.

רקע קצר על משמעותה של רשות ניירונית,
ニירון הוא מקום בזיכרון המחשב שמחזיק מספר. לפיכך, הרשות מורכבת מכמה מספרים
שמוחברים ביחד.

הרשות מסודרת מכמה שכבות של ניירונית. השכבה הראשונה קולטת את התמונה,
והאחרונה פולטת את התוצאה. בין שתי השכבות קיימת שכבה חבוכה אחת או יותר שבהם
מתבצעת הלמידה. כפי שניתן לראות בתמונה הבאה:



תרשים המדגים את אופן הפעולה של רשת עצביות מלאכותית.

למידה משמעותה פיתוח היכולת לבצע משימה מוגדרת. בפרויקט שלי הלמידה משמעותה אימון הרשת לזהות תמרורים שונים. כדי לרכוש את היכולת צריך לאמן את הרשת. במקרה שלי, יצא לרשת כ – 23,000 תמונות של תמרורים מסוגים שונים, וכן קטgorיה נוספת של *nothing*, כך שהרשת תהיה מסוג 'זהו אם לא הופיע תמרור בתמונה שהתקבלה' (חלק מהמחלק הפרויקט הינו זיהוי מתוך סריטון של מהלך נסעה, וطبعי שיקבל פרימיום במהלך העבודה שלא מופיע בהם תמרור).

אחרי ששסיימתי להציג לרשת את כל תמונות התמרורים, יצא לה עוד מספר תמרורים מסט ביקורת ששמרתי לצד, במקרה של זיהוי נכון, הרשת תקבל פידבק חיובי, ובמקרה של זיהוי שגוי, פידבק שלילי. אחזור על התהילה של הצגת התמרורים לאימון וכן תמונות מסט בקרה במסך מספר ריצות רצופות ואראה כיצד הרשת משפרת את אחוז הדיווק בכל פעם.

השאלה הנשאלת מآلיה היא, מה גורם לרשת ללמידה? התשובה היא שככל חלק של הרשת מתמחה בזיהוי של דפו מסויים, והשילוב של החלקים הוא שנוטן את התוצאה.

לדוגמא תמרור 'אין – כניסה' מורכב מעיגול שאותו יזהו אוטם חלקים של הרשת נירוניים שמתמחים בזיהוי צורת עיגול, אותו חלק של הרשת יכול לזהות גם תמרור 'נסעה עד 30 קמ"ש', רק שבמקרה זה נדרש גם הפעלה של חלק אחר של הרשת שמצויה ספרות. רק חיבור של שני חלקים הרשת יכול לבדוק בין התמורות השונים.

כלומר, החלקים השונים ברשת מתמחים בזיהוי חלקים שונים שימושתלבים ביניהם כדי לתת את התמונה הכוללת.



א"כ מדוע שחלקים מסוימים של הרשת יזהו דפוס מסוים ולא אחר, למה שנירונים מסוימים
יזהו עיגול ואחרים ספרה?

כל קשר בין ניירון לבין הנירון שכבה הבאה מקבל משקל (weight) ואם סכום המשקלים
הוא גבוה משמעותית עבור מאפיין מסוים בתמונה (דוגמת עיגול או ספרה) אז אותו חלק של
הרשת הופך למומחה בזיהוי המאפיין.

סכום המשקלים יכול לקבל כל ערך אבל ארצה שהערך יהיה בין 0 ל-1. לשם כך, משתמש
בפונקציה סיגmoidית (לוגריתמית), אשר דוחשת את הערכים שהוא מקבלת לטוווח שבין 0
ל-1. הערכים ששואפים ל-1 הם הסבירים והערכים הקרובים ל-0 אינם סבירים.

הפונקציה הסיגmoidית היא דוגמה לפונקציית אקטיבציה, ותפקידה לברור את החלקים
המשמעותיים לרשת מכל הרוש.

בדרכם כלל, פונקציית האקטיבציה היא לוגריתמית ולא לינארית כדי שנייתן יהיה לפענה
באמצעותה דפוסים מורכבים שלא ניתן לתאר באמצעות קו ישר בלבד. במידה וביעת הרוש
מתמידה בכל הקישורים שמצוינים את הנירון איז הרשת מושיפה הטיה (bias) לדוגמה, אם
רמת הרוש התמידית היא 10 איז הרשת מפחיתה 10 מסכום המשקלים לפני שהיא מכניסה
את הסכום לתוך פונקציית האקטיבציה.

כל הערכים המתקיים כתוצאה מהאימון הם הגורמים להסתגלות המערכת לזרחי
התמורות.

מודל ראשון שיוצרתי מבוסס על רשת ניירונים מתכנסת, רשות: CNN, Convolutional Neural Network
שהיא השימושית ביותר לזרחי של תМОנות בגל
שהיא מסוגלת להתחשב במרחב הרוחב והגובה של הפריטים מהם מורכבת התמונה.

המודל נכתב באמצעות הספרייה TensorFlow, של למידת מכונה שכתובה ב-Python.

פיתוח המודל:

ראשית, יבואתי את הספריות הבאות:



```
import numpy
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.models import Sequential
from ProjectNeurelNetwork.BL.Constant import Constants
```

כדי לאתחל את הרשת ניירונים יש להזין אותה במספרים אקראיים. הבעה עם הגישה זו שכאשר מתחילה במספרים אקראיים מקבלים תוצאות שונות בכל פעם. אחת הדרכים לקבל תוצאות עקביות היא באמצעות הגדנת מספר קבוע לפונקציה שמייצרת את המספרים האקראיים שמהם אתחל את התהילר. לדוגמה: 5.

```
 Gebiyut Massaf HaTzalim Regdomli CDI LeKabel Tovazot Ugebivot #  
numpy.random.seed(Constants.seed)
```

טעינת הנתונים:

טעינת תמונות התמוריים ממאגר התמונות, שולקו לשולשה תיקיות train ,test, validate לשילושה תיירות .DataOrganization כמפורט בהרבה בהמשך, בסעיף קודים נוספים, מחלוקת



```
class TrafficSignNet:
    """Model development class"""

    @staticmethod
    def Load_model():
        """
        Static class method. load the data for model learning.

        Returns:
        The folders train, test and number of categories to classify.
        """

        # create generator
        datagen = ImageDataGenerator()
        # prepare an iterators for each dataset
        # load and iterate training dataset
        train_it = datagen.flow_from_directory(Constants.train_path,
                                                class_mode='categorical',
                                                # either binary- if i only have two categories, or categorical if more,
                                                # In my case I have 13 categories so I will choose,
                                                batch_size=Constants.num_trainPics,
                                                target_size=(Constants.image_size, Constants.image_size))
```

```
# load and iterate test dataset
test_it = datagen.flow_from_directory(Constants.test_path,
                                       class_mode='categorical',
                                       batch_size=Constants.num_testPics,
                                       target_size=(Constants.image_size, Constants.image_size))

# confirm the iterator works
# אישר שהiterator פועל
train_x, train_y = train_it.next()
test_x, test_y = test_it.next()
# normalize input from 0-255 to 0-1
train_x = train_x / 255
test_x = test_x / 255
num_classes = test_y.shape[1]
return train_x, train_y, test_x, test_y, num_classes
```

הסבר מורה:

```
# normalize input from 0-255 to 0-1
train_x = train_x / 255
test_x = test_x / 255
```

בצעתי נרמול נתונים על – מנת לקבל ערכים בין 0 ל -1, כיוון שישנם 255 גוונים אפשריים נחלק כל אחד מסט הערכים ב-255 ונקבל ערך שהוא בין 0 ל-1.

רשת נירונית עובדת טוב יותר על ערכים, כאשר הם בין 0 ל -1, לפיכך נרמול הנתונים יעזר לרשת לעבוד טוב יותר על התמונות, ומילא להפיק תוצאות טובות יותר.



```
num_classes = test_y.shape[1]
```

מספר הקטגוריות לשׂוֹג כָּבוֹד הַוּקְטוּר בְּמִטְרִיצַת `test_y`. קלומר ישנים שני מטריצות, מטריצת X ומטריצת Y, מטריצת X – רב ממדית, ומטריצת Y – דו – ממדית.

הממדים במטריצת X – מטריצת הפלט, מורכבים:

ממד 0 – מספר תמונה.

ממד 1 – המיקום ה – X בתמונה.

ממד 2 – המיקום ה – Y בתמונה.

ממד 3 – ערך נורט ה – RGB.

ערך לדוגמא במטריצת X: 5,10,15,2 מבטא:

5 – תמונה מספר 5.

10 – פיקסל מספר 10.

15 – על 15 (פיקסל 10 על 15).

2 – G, הערך השני ב – RGB.

והממדים במטריצת Y – מטריצת הפלט, מורכבים:

ממד 0 - מספר תונות.

ממד 1 – גודל הוקטור, קלומר מספר הקטגוריות שעלה המודל לשׂוֹג לאחת מהן (במקרה שלי על המודל לשׂוֹג לאחת מ – 13 קטגוריות).

על – כן, מספר הקטגוריות נמצאת ב – `test_y.shape[1]`.



בנייה המודל:

```
@staticmethod
def baseline_model(num_classes):
    """
    Static class method. Building the model.

    Parameters:
    num_classes (int): number of categories to classify.

    Returns:
    The model.
    """
    # Create a layer type model
    model = Sequential()
    # convolution layer: 50 filters, 3X3, use with relu for activation function
    model.add(Conv2D(Constants.num_filters, (Constants.matrix, Constants.matrix),
                     input_shape=(Constants.image_size, Constants.image_size, Constants.channels),
                     activation='relu'))
    # מפוקה על מושבאות (שכבות פולינום)
    model.add(MaxPooling2D(pool_size=(2, 2)))
    # ignore from 20% from the noironim
    model.add(Dropout(0.2))

    # we have 3 channels. flat them to one long vector
    model.add(Flatten())
    # another noinim layer (with activation function)
    model.add(Dense(128, activation='relu'))
    # output
    model.add(Dense(num_classes, activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.summary()
    return model
```

הסבר מורחב:

יצירת מודל מסווג שכבות - ()

בלוק הconvolution:

השכבה החבוייה, היא שכבת הconvolution (convolution):

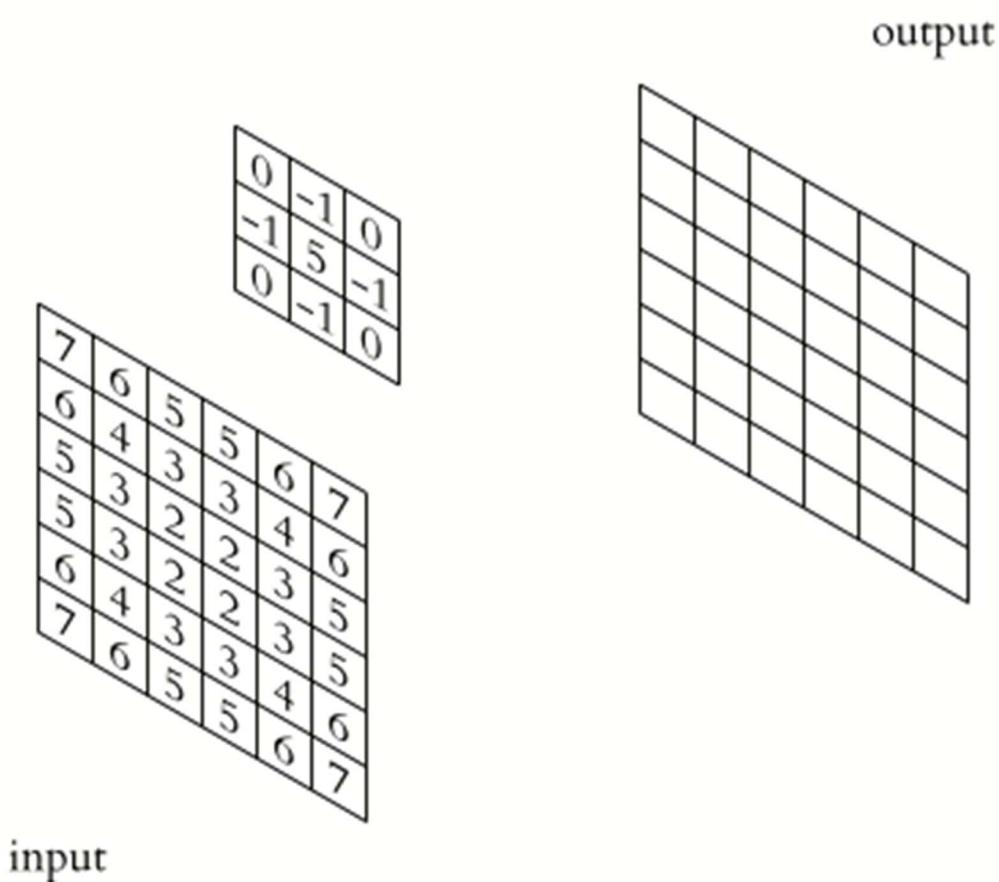
```
# convolution layer: 50 filters, 3X3, use with relu for activation function
model.add(Conv2D(Constants.num_filters, (Constants.matrix, Constants.matrix),
                 input_shape=(Constants.image_size, Constants.image_size, Constants.channels),
                 activation='relu'))
```

שכבה זו מרים מספר פילטרים רצויים בגודל רצוי של פיקסלים על כל תמונה (במקרה זה בחרתי 50 פילטרים בגודל 3X3 פיקסלים).
הפילטרים הם הנאורים של השכבה הכוללים פונקציית אקטיבציה (relu).



כל פילטר עובר על כל התמונה בסדרה של **צעדים** שמחזים אותו בכל פעם בפיקסל 1 בודד. התוצאה היא סידרה של ייצוגים חופפים של התמונה המכונה **feature map** כל פילטר מנפיק ייצוג אחד של התמונה, ולפיirc במקרה של שכבות הקובולציה היא **50 feature maps**.

הפעולה לדוגמא:



שכבה נוספת, היא שכבת הדגימה (pooling):

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```



תפקידה לצמצם את המטריצה (ריבוע הפיקסלים) שסמכיו 2×2 פיקסלים, שנמסר לה מהשכבה הקודמת, לרבע, כלומר כל מטריצה צאת תצומצם לריבוע אחד בלבד.

זה נעשה ע"י דגימת הערך הגבוה מבין ארבעת הפיקסלים הנ웃ונים, והוא מעבירה לשכבה הבאה.

מטרת ה pooling - היא להפוך את הדוגמאות לפחות קונקרטיות ועל ידי כך לעודד הכללות שייקלו על המודל בזיהוי דוגמאות שלא בכללו בסט האימון.

יתרנו נוסף הוא בכך שהיא מופכת לפחות מרכיבת וכתוצאה מכך פוחתים משאבי המחשב הדרושים ללמידה.

השכבה הבאה, היא שכבת ה - **Dropout**:

```
model.add(Dropout(0.2))
```

שכבה זו, אינה בדיק שכבה, המטרתה היא למנוע מהמערכת ללמידה יותר מדי טוב את הדוגמאות (לאימון), מה שיגרום לירידת יכולת להזוהות תמרורים שלא בכללו בסט הדוגמאות ששישמשו לאימון. וכן אומר לו להתעלם מ – 20% מנירוניים מסוימים, כלומר השכבה שומרת באקראי 20% מהມ"ע מהשכבה הקודמת.

במצב זה לא יהיה נוירון שהוא יבנה עליו.

עד כאן בлок הקונבולציה.

הסיווג בפועל של התמונות נעשה על ידי רשת נאורתית מסוג **Dense**.

על – מנת להעביר את המידע מבלוק הקונבולציה (שכולל את שלוש השכבות המפורחות לעיל) לרשת הנאורתית מסוג Dense, משתמשים בפונקציית flatten שתפקידה לשתח את המידע לווקטור (שכבה שגובהה פיקסל 1 בלבד) משום שכבת ה Dense חייבת לעבוד על ווקטור ולא על מטריצה.

```
model.add(Flatten())
```

פונקציה זו מקבלת מטריצה ומשטחת את הערכים במטריצה לווקטור אחד.

שכבת ה - **Dense**:

```
model.add(Dense(num_classes, activation='softmax'))
```



מכילה 128 פיקסלים בלבד, ובה כל ניירון מחובר לכל אחד מנירוני הקלט,

השכבה האחורונה היא שכבת הפלט, המכילה, במקרה של¹, 13 אפשרויות שניתן לסוג את התמונה. כל ניירון מייצג תמרור שונה והערך שהニーירון מקבל נע בטווח שבין 0 ל-1. ערך קרוב ל-0 הוא בלתי סביר וערך קרוב ל-1 הוא סביר.

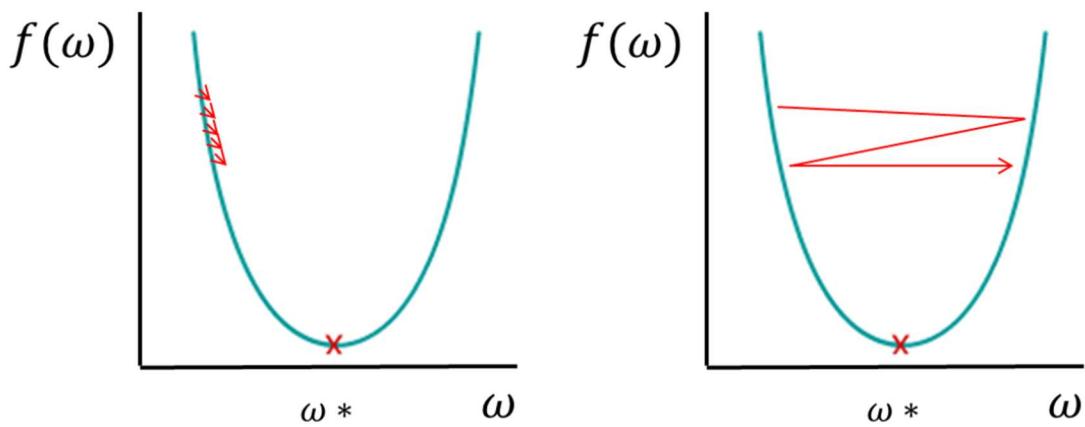
בשכבה הפלט פועלת פונקציית אקטיבציה מסווג **softmax** הגורמת לכך שסכום ההסתברויות של הנירונים בשכבה יהיה 1. הודות לפונקציית האקטיבציה תמיד לקבל אחד מ-13 הנירונים בשכבה הוא הפתרון הסביר ביותר לתמונה שהזנתנו לרשת הנירונים.

קימפול המודל:

```
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

בקומpileציה של המודל השתמשתי בפונקציית Adam **שגדילה** את השינויים שעושים לפרמטרים של המודל - **carshookim** מהמינימום שהוא הפתרון המתמטי של המודל, **ומקטינה** את השינויים ככל שמתקרבים **למינימום**. גודל השינויים הוא חשוב מפני שינויים גדולים מדי יכולים לגרום לפספוס המינימום בעוד שינויים קטנים מדי יגרמו לאימון לרוץ יותר מדי זמן.

תרשים המדגים את פועלות פונקציית האופטימיזציה Adam



אימון המודל:

כדי לאמן את המודל נרץ אותו מספר פעמים רצוי (epochs) על כל התמונות, וכך להימנע מלהכבד על המחשב יותר מדי נזין את התמונות באצווות של 64 (batch_size).

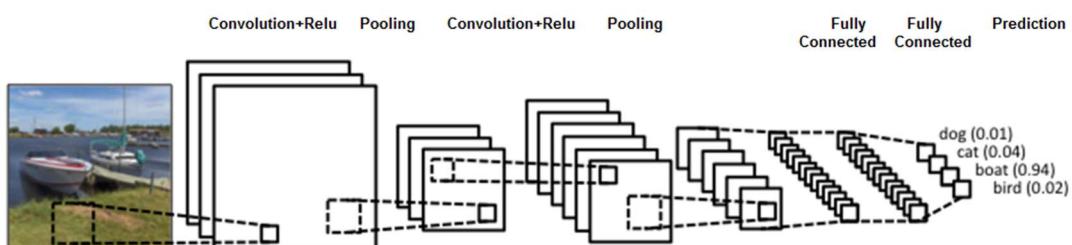


```
@staticmethod
def fit_model(model, train_x, train_y, test_x, test_y):
    """
    Static class method. Training the model.

    Parameters:
    model: The model built.
    train_x, train_y: The folder train.
    test_x, test_y: The folder test.

    Returns:
    none.
    """
    # Fit the model
    model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=Constants.epochs,
              batch_size=Constants.batch_size,
              verbose=1)
    model.save("network@100.h5")
    scores = model.evaluate(test_x, test_y, verbose=1)
    print("Accuracy: %.2f%%" % (scores[1] * 100))
```

תרשים המדגים את תהליכי אימון שכבות המודל:



לאחר אימון המודל הראשון, אחוז הדיוק בזיהוי הינו כ- 95%.



```
Epoch 1/15
152/152 [=====] - 212s 1s/step - loss: 1.2838 - accuracy: 0.6416 - val_loss: 0.4691 - val_accuracy: 0.8676
Epoch 2/15
152/152 [=====] - 223s 1s/step - loss: 0.2795 - accuracy: 0.9247 - val_loss: 0.2794 - val_accuracy: 0.9334
Epoch 3/15
152/152 [=====] - 232s 2s/step - loss: 0.1079 - accuracy: 0.9713 - val_loss: 0.2504 - val_accuracy: 0.9412
Epoch 4/15
152/152 [=====] - 228s 2s/step - loss: 0.0580 - accuracy: 0.9850 - val_loss: 0.2670 - val_accuracy: 0.9497
Epoch 5/15
152/152 [=====] - 230s 2s/step - loss: 0.0343 - accuracy: 0.9940 - val_loss: 0.2889 - val_accuracy: 0.9467
Epoch 6/15
152/152 [=====] - 232s 2s/step - loss: 0.0289 - accuracy: 0.9932 - val_loss: 0.2891 - val_accuracy: 0.9534
Epoch 7/15
152/152 [=====] - 233s 2s/step - loss: 0.0209 - accuracy: 0.9952 - val_loss: 0.2699 - val_accuracy: 0.9560
Epoch 8/15
152/152 [=====] - 250s 2s/step - loss: 0.0081 - accuracy: 0.9989 - val_loss: 0.2914 - val_accuracy: 0.9564
Epoch 9/15
152/152 [=====] - 248s 2s/step - loss: 0.0120 - accuracy: 0.9973 - val_loss: 0.3266 - val_accuracy: 0.9571
Epoch 10/15
152/152 [=====] - 234s 2s/step - loss: 0.0137 - accuracy: 0.9976 - val_loss: 0.3630 - val_accuracy: 0.9449
Epoch 11/15
152/152 [=====] - 231s 2s/step - loss: 0.0250 - accuracy: 0.9941 - val_loss: 0.3968 - val_accuracy: 0.9497
Epoch 12/15
152/152 [=====] - 229s 2s/step - loss: 0.0210 - accuracy: 0.9955 - val_loss: 0.3686 - val_accuracy: 0.9471
Epoch 13/15
152/152 [=====] - 230s 2s/step - loss: 0.0061 - accuracy: 0.9987 - val_loss: 0.3885 - val_accuracy: 0.9475
Epoch 14/15
152/152 [=====] - 231s 2s/step - loss: 0.0088 - accuracy: 0.9982 - val_loss: 0.4715 - val_accuracy: 0.9408
Epoch 15/15
152/152 [=====] - 248s 2s/step - loss: 0.0171 - accuracy: 0.9955 - val_loss: 0.4444 - val_accuracy: 0.9504
```

Accuracy: 95.04%

מודל נוסף שיצרתי, מרכיב מאותן השכבות של המודל הראשון, אך הוסיף **בלוק קונבולוציה**, לצורך בדיקה האם המודל יחזיר אחוזי זיהוי גבוהים יותר.

וכן לאחר האימון, המודל הגיע למעלה 96% דיווק!

להלן המודל:



טעינת הנתונים (כ"ל במודל א')

```
class TrafficSignNet:  
    """Model development class"""  
  
    @staticmethod  
    def load_model():  
        """  
        Static class method. Load the data for model learning.  
  
        Returns:  
        The folders train, test and number of categories to classify.  
        """  
  
        # create generator  
        datagen = ImageDataGenerator()  
        # prepare an iterators for each dataset  
        # load and iterate training dataset  
        train_it = datagen.flow_from_directory(Constants.train_path,  
                                              class_mode='categorical',  
                                              # either binary- if i only have two categories, or categorical if more,  
                                              # In my case I have 13 categories so I will choose,  
                                              batch_size=Constants.num_trainPics,  
                                              target_size=(Constants.image_size, Constants.image_size))  
  
        # load and iterate test dataset  
        test_it = datagen.flow_from_directory(Constants.test_path,  
                                              class_mode='categorical',  
                                              batch_size=Constants.num_testPics,  
                                              target_size=(Constants.image_size, Constants.image_size))  
  
        # confirm the iterator works  
        # אפליגון פונקציית next()  
        train_x, train_y = train_it.next()  
        test_x, test_y = test_it.next()  
        # normalize input from 0-255 to 0-1  
        train_x = train_x / 255  
        test_x = test_x / 255  
        num_classes = test_y.shape[1]  
        return train_x, train_y, test_x, test_y, num_classes
```

בנייה המודל – הוספה בלוק קונבולציה.



```
@staticmethod
def baseline_model(num_classes):
    """
    Static class method. Building the model.

    Parameters:
    num_classes (int): number of categories to classify.

    Returns:
    The model.
    """
    # Create a layer type model
    model = Sequential()
    # convolution layer: 50 filters, 3X3, use with relu for activation function
    model.add(Conv2D(Constants.num_filters, (Constants.matrix, Constants.matrix),
                     input_shape=(Constants.image_size, Constants.image_size, Constants.channels),
                     activation='relu'))
    # (פונקציית פולינומיאלית)
    model.add(MaxPooling2D(pool_size=(2, 2)))

    # [add a convolution block to decrease the loss]
    model.add(Conv2D(Constants.num_filters, (Constants.matrix, Constants.matrix),
                     input_shape=(Constants.image_size, Constants.image_size, Constants.channels),
                     activation='relu'))
    # [add this layer to decrease the loss]
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
# ignore from 20% from the noironim
model.add(Dropout(0.2))
# we have 3 channels. flat them to one long vector
model.add(Flatten())
# another noironim layer (with activation function)
model.add(Dense(128, activation='relu'))
# output
model.add(Dense(num_classes, activation='softmax'))
# Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
return model
```

אימון המודל (כ"ל במודל א')



```

@staticmethod
def fit_model(model, train_x, train_y, test_x, test_y):
    """
        Static class method. Training the model.

    Parameters:
    model: The model built.
    train_x, train_y: The folder train.
    test_x, test_y: The folder test.

    Returns:
    none.
    """
    # Fit the model
    model.fit(train_x, train_y, validation_data=(test_x, test_y), epochs=Constants.epochs,
              batch_size=Constants.batch_size,
              verbose=1)
    model.save("network@100.h5")
    scores = model.evaluate(test_x, test_y, verbose=1)
    print("Accuracy: %.2f%%" % (scores[1] * 100))

```

ואכן, כאמור לעיל, אחוז זיהוי המודל הגיעו ל- 96% דיק!

```

Epoch 1/12
152/152 [=====] - 162s 1s/step - loss: 1.0096 - accuracy: 0.6732 - val_loss: 0.4076 - val_accuracy: 0.8380
Epoch 2/12
152/152 [=====] - 157s 1s/step - loss: 0.3206 - accuracy: 0.8956 - val_loss: 0.2591 - val_accuracy: 0.9053
Epoch 3/12
152/152 [=====] - 146s 964ms/step - loss: 0.1951 - accuracy: 0.9354 - val_loss: 0.1701 - val_accuracy: 0.9446
Epoch 4/12
152/152 [=====] - 153s 1s/step - loss: 0.1281 - accuracy: 0.9564 - val_loss: 0.1373 - val_accuracy: 0.9560
Epoch 5/12
152/152 [=====] - 155s 1s/step - loss: 0.0886 - accuracy: 0.9700 - val_loss: 0.2283 - val_accuracy: 0.9290
Epoch 6/12
152/152 [=====] - 154s 1s/step - loss: 0.0747 - accuracy: 0.9749 - val_loss: 0.1750 - val_accuracy: 0.9516
Epoch 7/12
152/152 [=====] - 156s 1s/step - loss: 0.0626 - accuracy: 0.9792 - val_loss: 0.1629 - val_accuracy: 0.9553
Epoch 8/12
152/152 [=====] - 161s 1s/step - loss: 0.0502 - accuracy: 0.9826 - val_loss: 0.1528 - val_accuracy: 0.9671
Epoch 9/12
152/152 [=====] - 159s 1s/step - loss: 0.0432 - accuracy: 0.9862 - val_loss: 0.1512 - val_accuracy: 0.9689
Epoch 10/12
152/152 [=====] - 158s 1s/step - loss: 0.0335 - accuracy: 0.9897 - val_loss: 0.1351 - val_accuracy: 0.9671
Epoch 11/12
152/152 [=====] - 159s 1s/step - loss: 0.0317 - accuracy: 0.9904 - val_loss: 0.1439 - val_accuracy: 0.9734
Epoch 12/12
152/152 [=====] - 162s 1s/step - loss: 0.0571 - accuracy: 0.9820 - val_loss: 0.1644 - val_accuracy: 0.9678

```

Accuracy: 96.78%

מודל רשת נוספת שבחرتית לאמן, כאמור, הינה רשת המבוססת על מודל vgg16.

מודל זה מגדים למידת העברה - transfer learning שהיא ענף של למידת מכונה המתמקד בשימוש במידע שנרכש תוך פתרון בעיה אחת לפתרון בעיה אחרת, איבר אחד מוחדר את מודל 16vgg שיווכל להבחן בין סוגים שונים, ע"י החלפת השכבות המס滂גות של המודל המקורי.



ראשית מודל 16vgg עובד על תמונות בגודל (224, 224), לצורך כך שיניתי את גודל התמונות לגודל המתאים, באמצעות הפונקציה `resize_images` (שתפורט בהמשך בסעיף קוד התוכנית, מחלוקת .(Technical).

פיתוח המודל:

הספריות הנדרשות:

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from ProjectNeurelNetwork.BL.Constant import Constants
```

טעינה הנתונים מתיקיות `train`, `test`, `validate` קודם (כ"ל).

```
class TrafficSignNetBasedVGG16:
    """Model development class"""

    @staticmethod
    def load_data():
        """
        Static class method. load the data for model learning.

        Returns:
        The folders train, test and validate.
        """

        # create generator
        datagen = ImageDataGenerator()
        classes = ['BumpyRoad', 'CloseToVehicles', 'CurvesAlongTheWay', 'DangerOfSlipping', 'DangerousPlace',
                   'GiveWay', 'NoEntry', 'Nothing', 'RoadConstruction', 'UpTo30kmh', 'UpTo50kmh', 'UpTo80kmh',
                   'movement circle']

        # prepare an iterators for each dataset
        # load and iterate training dataset
        train_batches = datagen.flow_from_directory(directory=Constants.train_path,
                                                     classes=classes,
                                                     class_mode='categorical',
                                                     target_size=(Constants.imageSize, Constants.imageSize),
                                                     batch_size=Constants.batchSize,
                                                     shuffle=True)

        # load and iterate valid dataset
        valid_batches = datagen.flow_from_directory(directory=Constants.valid_path,
                                                     classes=classes,
```



```
        class_mode='categorical',
        target_size=(Constants.imageSize, Constants.imageSize),
        batch_size=Constants.batchSize,
        shuffle=True)

# load and iterate test dataset
test_batches = datagen.flow_from_directory(directory=Constants.test_path,
                                             classes=classes,
                                             class_mode='categorical',
                                             target_size=(Constants.imageSize, Constants.imageSize),
                                             batch_size=Constants.batchSize,
```

ה – generator נדרש שלא לערבות את המידע שהוא מושך מתיקית כדי שאוכל להשוות בהמשך את הקטגוריות שהמודל חזה עם הקטגוריות בפועל אליו משתיכות התמונה.

```
        shuffle=False)
return train_batches, valid_batches, test_batches
```

בנייה המודל:

```
@staticmethod
def baseline_model():
    """
    Static class method. Building the model.

    Returns:
    The model.
    """
    # יייאו מודל
    vgg16_model = VGG16()
    # מגדיר מודל למודל המקורי נל שכבת
    model = Sequential()
    for layer in vgg16_model.layers:
        model.add(layer)

    convolution_model = Sequential()

    # I'm interested in categorizing into 13 categories,
    # not like the original model categorized into 1000 categories
    # So I will download the last three layers and attach the rest
    # To build a new model that will include only the convolution layers and named 'convolution_model'.
    for layer in vgg16_model.layers[:-4]:
        convolution_model.add(layer)

    # make the layers of the 'convolution_model' untrainable.
    for layer in convolution_model.layers:
        layer.trainable = False
```



```
# I have to set the input and output of the model, after running model.summary()
# i saw the output in the last layer is block5_pool
```

```
transfer_layer = model.get_layer('block5_pool')
# Define the input and output of the model I will create
convulation_model = Model(inputs=convulation_model.input,
                           outputs=transfer_layer.output)
```

```
# Number of categories to classify.
num_classes = Constants.num_classes
```

```
# Define the model I will build of the layer type.
```

```
my_new_model = Sequential()
```

```
# Add the convolutional layers of the VGG16 model
```

```
my_new_model.add(convulation_model)
```

```
# flatten the output of the VGG16 model because it is from a
```

```
# convolutional layer
```

```
my_new_model.add(Flatten())
```

```
# add a dense (fully-connected) layer.
```

```
# this is for combining features that the VGG16 model has
```

```
# recognized in the image.
```

```
my_new_model.add(Dense(1024, activation='relu'))
```

```
# add a dropout layer.
```

```
my_new_model.add(Dropout(0.5))
```

```
# add the final layer for the 13 category classification.
```

```
my_new_model.add(Dense(num_classes, activation='softmax'))
```

```
# in the process we use the Adam optimizer with a low learning-rate so we don't
# distort the original weights from the pre-trained VGG16 model
```

```
optimizer = Adam(lr=1e-5)
```

```
# loss function should by 'categorical_crossentropy' for multiple classes
```

```
# compile the model
```

```
loss = 'categorical_crossentropy'
```

```
my_new_model.compile(optimizer=optimizer, loss=loss, metrics=['binary_accuracy'])
```

```
return my_new_model
```



```
@staticmethod
def fit_model(model, train_batches, valid_batches, test_batches):
    """
    Static class method. Training the model.

    Parameters:
    model: The model built.
    train_batches: The folder train.
    valid_batches: The folder validate.
    test_batches: The folder test.

    Returns:
    none.
    """
    # early stopping stops the learning process in the case that the process doesn't progress for 2 epochs
    # or in the case of over fitting to the training data over the validation data
    es = EarlyStopping(monitor='val_loss',
                        min_delta=0,
                        patience=2,
                        verbose=1,
                        mode='auto')

    # מגדיר מנגנונים נתונים. כדי שיבריזט את כל סעיפים
    # רק פעם אחת בכל Epoch גדרה את פרמטר זהה
    # מנגנון חותם מנגנונים ביא מופיע תחילה בסעיף הראשון וילך מנגנונים
    step_size_train = train_batches.n // train_batches.batch_size
    step_size_valid = valid_batches.n // valid_batches.batch_size
```

```
# Fit the model
history = model.fit_generator(train_batches,
                               epochs=Constants.epochs,
                               steps_per_epoch=step_size_train,
                               validation_data=valid_batches,
                               validation_steps=step_size_valid,
                               callbacks=[es],
                               verbose=1)

step_size_test = test_batches.n // test_batches.batch_size
result = model.evaluate_generator(test_batches, steps=step_size_test)
print("Test set classification accuracy: {:.2%}".format(result[1]))
model.save("network@224.h5")
```

הסבר מורחב:

בנייה המודל:

"בוא מודל VGG16 שעליו אטיבסס, מקובל בשיטת transfer learning".

```
vgg16_model = VGG16()
```

המרת המודל VGG16 למודל מסווג שכבות.



```
model = Sequential()
for layer in vgg16_model.layers:
    model.add(layer)
```

מודל VGG16 מסוג למספר מסוים של קטגוריות, אך אני מעוניינת לסוג לשולשה – עשר קטגוריות, ולכן אוריד את שלושת השכבות האחרונות של המודל, ואת יתר אוצרם לבניית המודל שלי שיכלול רק את שכבות הקונבולוציה ושם `convulation_model`.

```
convulation_model = Sequential()
```

```
for layer in vgg16_model.layers[:-4]:
    convulation_model.add(layer)
```

ארצאה לשמר את המידע שניצבר במודל למטרות שתיכף אותו על התמונות שלי, (בהנחה שאנו בשיטה של למידת מכונה – transfer learning – מונע מהתהילך האימון לשנות את המשקלות נוצרך כך אקפיא את שכבות הקונבולוציה, באופן שימנע מהתהילך האימון לשנות את המשקלות שלו).

```
for layer in convulation_model.layers:
    layer.trainable = False
```

עליה להציג את הקלט והפלט של המודל, לאחר הרצתה של `model.summary()`, ראייתי שהפלט בשכבה الأخيرة, היא `pool5`, וכך הגדרתי:

```
transfer_layer = model.get_layer('block5_pool')
```

```
convulation_model = Model(inputs=convulation_model.input,
                           outputs=transfer_layer.output)
```

לאחר ההכנה של שכבות המודל VGG16 לפי מה שאני צריכה למודל שלי, אוצר את המודל שלי (מסוג שכבות).

```
my_new_model = Sequential()
```



הוספת שכבות הconvolutional של מודל VGG16 למודל שאבנה.

```
my_new_model.add(convulation_model)
```

יש צורך לבצע flatten לשכבה הפלט של מודל VGG16 (כזכור, flatten זו שכבה המתקבלת מטריצה משטחת אותה לווקטור אחד), מפני ששכבה הפלט היא משכבות הconvolutional.
ועל – מנת להעביר את המידע משכבות הconvolutional להמשך שכבות המודל, משתמש בפונקציית flatten שתפקידה לשתח את המידע לווקטור (שכבה שאובגה פיקסל 1 בודד).

```
my_new_model.add(Flatten())
```

הוספת שכבת Dense למודל, עם פונקציית אקטיבציה relu, מכילה 1024 פיקסלים בלבד. ובה כל ניירון מחובר לכל אחד מנירוני השכבה הבאה.

```
my_new_model.add(Dense(1024, activation='relu'))
```

הוספת שכבה של Dropout, שכבה זו, אינה בדיק שכבת, המטרתה היא למנוע מהמערכת ללמידה יותר מדי טוב את הדוגמאות (לאימון) מה שייגרום לירידת היכולת לזהות תמרורים שלא נכללו בסט הדוגמאות ששימושו לאימון.

ולכן אומר לו להתעלם מ – 50% מנירונים מסוימים, ככלומר השכבה שומנת באקראי 50% מהמידע מהשכבה הקודמת.

במצב זה לא יהיה ניירון שהוא יבנה עליו.

```
my_new_model.add(Dropout(0.5))
```

הוספת שכבה נוספת וסופה של Dense, עם פונקציית אקטיבציה softmax, מפני שזו שכבת הפלט, המכילה, במקרה של 13 אפשרויות שניתן לסוג את התמונה. כל ניירון מייצג תמרור שונה והערך שהנירון מקבל נעה בטוחה שבין 0 ל-1. ערך קרוב ל-0 הוא בלתי סביר וערך קרוב ל-1 הוא סביר.

כאמור בראשת הקודמת, פונקציית אקטיבציה מסוג softmax גורמת לכך שסכום ההסתברויות של הנירונים בשכבה יהיה 1. הודות לפונקציית האקטיבציה תמיד קיבל אחד מ-13 הנירונים בשכבה הוא הפתרון הסביר ביותר לתמונה שהזנת לרשף הנירונים.

קימפול המודל:



לצורך קומפקציית המודל, הגדרתי פונקציית אופטימיזציה מסווג Adam עם קצב למידה נמוך למדי.

פונקציית Adam (כאמור לעיל) **מגדילה** את השינויים שעושים לפרמטרים של המודל - **כשרוחקים** מהמינימום שהוא הפתרון המתמטי של המודל, **ומקטינה** את השינויים ככל שמתקרבים **למינימום**. גודל השינויים הוא חשוב מפני ששינויים גדולים מדי יכולים לגרום לפספסות המינימום בעוד שינוי קטנים מדי יגרמו לאימון לרוץ יותר מדי זמן.

```
optimizer = Adam(lr=1e-5)
```

```
loss = 'categorical_crossentropy'  
my_new_model.compile(optimizer=optimizer, loss=loss, metrics=['binary_accuracy'])
```

אימון המודל:

פונקציית עצירה מוקדמת להפסקת תהליכי הלמידה במידה ואין בו התקדמות, או במקרה של over fitting לסת האימון לעומת הוועידה.

```
es = EarlyStopping(monitor='val_loss',  
                   min_delta=0,  
                   patience=2,  
                   verbose=1,  
                   mode='auto')
```

הגדרתי שאימון המודל יעשה ע"י ריצה של לכל היותר 100 epochs, אך לא סביר שיגיע לשם, בפועל הוא יעצור את האימון עד קודם, הודות ל – .early stopping

```
history = model.fit_generator(train_batches,  
                               epochs=Constants.epochs,  
                               steps_per_epoch=step_size_train,  
                               validation_data=valid_batches,  
                               validation_steps=step_size_valid,  
                               callbacks=[es],  
                               verbose=1)
```

הערכת ביצועי המודל:

הערכת ביצועי המודל על סמך מידת הדיק של סוג התמונות בסט המבחן (תיקית ה – test), אילו תמונות שהמודל לא נחשף אליהם במהלך הלמידה.



וכן שמירת הרשות שהמודל יצר.

```
result = model.evaluate_generator(test_batches, steps=step_size_test)
print("Test set classification accuracy: {:.2%}".format(result[1]))
model.save("network@224.h5")
```

Test set classification accuracy: 93.08%

מודל זה נתן לי 93% אחוזי דיוק – זה אומנם יפה מאד, אך הרשות הקודמת נתנה לי אחוזי זיהוי גבוהים יותר (96%), לפיכך בחרתי את הרשות שנוצרה ממהמודל הראשון.

קוד התוכנית

:Server – הקוד התוכנית בצד ה-

כעת אפרט על שאר המחלקות בצד שרת, שחלקן יהיו תשתיית למידת המכונה וחלקן כפונקציות ממשק להצגת הפלט.

חלק א' - Algorithm :

מחלקה DataOrganizaiton :

קלט – נתיב למאגר התמונות שברצוני לחלק לשולשה תיקיות.

פלט – שלושה תיקיות, ובכל אחת מהן מספר תמונות שהוגדר.

תפקיד המחלקה הינה חלוקת מאגר התמונות לשולשה מאגרים: ,train, test, validate שעליהם יתאמנו הרשותות השונות.

חלוקת מתבצעת באופן הבא:

20% מכלל התמונות במאגר – יוכנסו באופן רנדומלי (זאת בעזרת הספרייה SciKit-Learn)

.Test – לתיקיות ה – .Train – לשאר (כ – 80%), יוכנסו לתיקיות ה –



לאחר מכן, מטור התמונות שבתיקיית ה – Train, ניקח 10% לתיקיית ה – Validate.

תפקידה של כל תיקיה:

Train – עם התמונות שבמאג'ר זה תתאמן הרשות ותלמד איך נראה כל תמרור.

Test – עם תמונות אלו תבחן את נכונותה (עדין בשלב האימון והלמידה).

Validate – תמונות אלו יוצגו לרשות, בסיום האימון, ככלומר אלו תמונות שהרשות לא ראתה כלל במהלך האימון, ואיתם בוחנתי האם אכן הרשות מזזה נכון.

אוף הפעולה:

הספריות הנדרשות:

```
import os
import shutil
from sklearn.model_selection import train_test_split
from ProjectNeurelNetwork.BL.Constant import Constants

class DataOrganization:
    """Class organizes the image dataset after processing into 3 folders."""

    def __init__(self, path):
        self.path = path

    def randomTheFile(self, name_dir):
        """
        Class method divides the image dataset into 3 folders

        Parameters:
            name_dir (str): Dataset path for partition.

        Returns:
            none.
        """
        X = []
        directory = os.path.join(self.path, name_dir)
        # מבוקע כל תיקיותו במאוגת מתמונות
        for folder_name in os.listdir(directory):
            currentPath = os.path.join(directory, "/", folder_name)
            os.chdir(currentPath) # מסתובב בפניה טר חטבון בתקבציהם
            # X מזמין את המתמונות של הנקודות הנקודות במאוגן לרשימה
            for i in sorted(os.listdir()):
                X.append(i)
        train_valid_name = folder_name + '_train_valid'
        test_name = folder_name + '_test'
        train_name = folder_name + '_train'
        valid_name = folder_name + '_valid'
```



```
train_valid_name, test_name = train_test_split(X, test_size=0.2, random_state=1)
# validate -> 10% -> train -> 70% -> .train
train_name, valid_name = train_test_split(train_valid_name, test_size=0.1, random_state=1)
# המתקיים של התמונות מתווגן חרישיות מחזוריות כנ"ל גזירה ותבניות
for item in train_name:
    ori = os.path.join(currentPath, "/", item)
    dest = os.path.join(self.path, '/Train@100f', folder_name)

    shutil.copy(ori, dest)

for item1 in test_name:
    ori1 = os.path.join(currentPath, '/', item1)
    dest1 = os.path.join(self.path, '/Test@100f/', folder_name)
    shutil.copy(ori1, dest1)

for item2 in valid_name:
    ori2 = os.path.join(currentPath, '/', item2)
    dest2 = os.path.join(self.path, '/Validate@100f/', folder_name)
    shutil.copy(ori2, dest2)
# אתחול את חרישיות X לירגמה עבוגה חטגורית
X = []
```

תוצאת הרצה הינה, שלושה תיקיות המכילות סך תמונות כפי שהואגדה.



:Predict

קלט – רשימה פרימיום (תמונות מתוך קובץ וידאו לאחר עיבודם בפונקציית detect, כמוポート לעיל, מחלוקת DetectTS, סעיף אלגוריתמים מרכזיים).

פלט – רשימה המכילה את מספרי התמורות שזוהו ע"י הרשת נירונית.

תפקיד המחלוקת הינה 'חיזוי' התמונה שהתקבלה (במקרה של, הפריים) וווג התמוך, במידה ומופיע.

אחרת מסווג לקטגורית nothing.

פונקציית המחלוקת TSClassification, תשליך כל פריים ופריים לזיהוי ברשת נירונית ותחזיר רשימה של מספרי התמורות שזוהו בסרטון.

אוף הפעולה:



הספריות הנדרשות:

```
import numpy as np
from skimage import transform
from tensorflow.keras.models import load_model
from ProjectNeurelNetwork.BL.Constant import Constants as const
```

```
class Predict:

    @staticmethod
    def LTSClassification(list_pic):
        """
        Static class method. Classifies the frame into the appropriate category.

        Parameters:
        list_pic (list): List of frames after processing in the detect function.

        Returns:
        List of traffic signs numbers identified during the video.
        """
        LTSClassification = []
        # טעינה חישובית בירכונים
        loaded_model = load_model('../Server/network.h5')
        for p in list_pic:
            if p[1] == True:
                # איזורת מערך מהתמונה, עבור חישוב הערך
                np_image = np.array(p[0]).astype('float32')
                # ברגז הטראנס הילס, כפוי תפלוקם של מושג באנטוגרף
                np_image = np_image / 255
                # גנטית גודל כפוי הגדיל שטחים אטומים בירקון
                np_image = transform.resize(np_image, (const.image_size, const.image_size, const.channels))
                # פולינומית מודרנית המבזבז גודל כפוי
                np_image = np.expand_dims(np_image, axis=0)
                # ביצוע מושג תחומרה, נס"י. הרשות שאומנת
                predict = loaded_model.predict(np_image)

                # קביעת מושג תחומרה מערך אחד
                y_pred = np.argmax(predict, axis=1)
                # הפיכת המושג למספרים
                listToInt = int(y_pred)
                # הגדלת מושג תחומרה
                LTSClassification.append(listToInt)
            else:
                # nothing - יתלווה
                LTSClassification.append(8)
        return LTSClassification
```

מחלקה ReadData

קלט – נתיב לקובץ Excel, בו מידע ונתונים על התמורות השונים.

פלט – המכיל את מאפייני התמורה כפי שנקרו או מקובץ ה – Excel, dictionary



תפקיד המחלקה הינו קריית נתונים נדרשים מתוך קובץ Excel – מקור הנתונים.

אוף הפעולה:

הספריות הנדרשות:

```
import openpyxl as xl
from ProjectNeurelNetwork.BL.Constant import Constants as const
```

פונקציית מחלקה ראשונה, `supported_TS`, המחזירה מידע על כל התמורות.

```
class ReadData:
    """Class reads data from Excel"""

    def __init__(self, path):
        self.path = path

    def supported_TS(self):
        """
        Class method reads all the data on all the traffic signs.

        Returns:
        list: a list of dictionary containing all the details of the signs.
        """

        # Excel-ה פוןction שמייצרת מילויים אטום ומקבילה ל-Excel, מיכילות מידע על התמורות
        wb_obj = xl.load_workbook(self.path)
        sheet_obj = wb_obj.active
        LIST_TS = [None] * const.num_classes
        i = 0
        # מתרגם בולגלאט על כל שורות ה-Excel, מיכילות מידע על התמורות
        for j in range(sheet_obj.max_row - 1):
            cell_objNumber = sheet_obj.cell(row=j + const.begin_data, column=const.number_TS)
            Number_trafficSign = cell_objNumber.value
            cell_objName = sheet_obj.cell(row=j + const.begin_data, column=const.name_TS)
            Name_trafficSign = cell_objName.value
            cell_objCategory = sheet_obj.cell(row=j + const.begin_data, column=const.category_TS)
            Category_trafficSign = cell_objCategory.value

            cell_objSignificance = sheet_obj.cell(row=j + const.begin_data, column=const.significance_TS)
            Significance_trafficSign = cell_objSignificance.value
            cell_objPicture = sheet_obj.cell(row=j + const.begin_data, column=const.picture_TS)
            Picture_trafficSign = cell_objPicture.value
            # מגדיר מילויים אטום שמייצרת מילויים אטום
            LIST_TS[i] = {"NumberTrafficSign": Number_trafficSign,
                          "NameTrafficSign": Name_trafficSign,
                          "CategoryTrafficSign": Category_trafficSign,
                          "SignificanceTrafficSign": Significance_trafficSign,
                          "PictureTrafficSign": Picture_trafficSign}

            i += 1
        return LIST_TS
```

פונקציית המחלקה הבאה, `getDetails_trafficSign()`, המחזירה מידע על תמרור ספציפי שהתקבל.



```
def getDetails_trafficSign(self, id):
    """
    Class method reads data of specific traffic sign.

    Parameters:
    id (int): Sign ID number.

    Returns:
    dictionary: a dictionary containing the details of the specific sign.
    """

    # Excel-ה אובייקט מקבל את גובץ
    wb_obj = xl.load_workbook(self.path)
    sheet_obj = wb_obj.active
    # בדיקת תבנית טריבוט נספחים
    cell_objNumber = sheet_obj.cell(row=id + const.begin_data, column=const.number_TS)
    Number_trafficSign = cell_objNumber.value
    cell_objName = sheet_obj.cell(row=id + const.begin_data, column=const.name_TS)
    Name_trafficSign = cell_objName.value
    cell_objCategory = sheet_obj.cell(row=id + const.begin_data, column=const.category_TS)
    Category_trafficSign = cell_objCategory.value
    cell_objSignificance = sheet_obj.cell(row=id + const.begin_data, column=const.significance_TS)
    Significance_trafficSign = cell_objSignificance.value
    cell_objPicture = sheet_obj.cell(row=id + const.begin_data, column=const.picture_TS)
    Picture_trafficSign = cell_objPicture.value
    cell_objAlert = sheet_obj.cell(row=id + const.begin_data, column=const.alert_TS)
    Alert_trafficSign = cell_objAlert.value

    # מילוי המרכיבים שנדרשו מטריבוט
    return {"NumberTrafficSign": Number_trafficSign,
            "NameTrafficSign": Name_trafficSign,
            "CategoryTrafficSign": Category_trafficSign,
            "SignificanceTrafficSign": Significance_trafficSign,
            "PictureTrafficSign": Picture_trafficSign,
            "AlertSignification": Alert_trafficSign})
```

מחלקה **TrafficSign**:

קלט – קובץ Excel, המכיל נתונים על כל התמרורים.

פלט – כל מאפייני התמרורים בהתאם.

אופן הפעולה:

פונקציית מחלוקת ראשונה, ()`getAlert`, תפקידה, מקבלת מספר תמרור, מבצעת קריאה לפונקציה `ReadData` (שמפורטת במחלקה `getDetails_trafficSign`) ומחזירה את כל המאפיינים על המספר תמרור שהתקבל.



```
class TrafficSign:  
    """Class that activates the call functions from Excel"""  
    def __init__(self, path):  
        self.path = path  
  
    def getAlert(self, id_TS):  
        """  
        Class method activates the method that returns data about a specific sign.  
  
        Parameters:  
        id_TS (int): Sign ID number.  
  
        Returns:  
        dictionary: a dictionary containing the details of the specific sign.  
        """  
        r = ReadData(self.path)  
        return r.getDetails_trafficSign(id_TS)
```

פונקציה נוספת, `get_supported_TS()`, קרייה לפונקציה `supported_TS` (שמפורטת במחלקה `ReadData`), ומחזירה נתונים על כל התמורות הנתמכים.

```
def get_supported_TS(self):  
    """  
    Class method activates the method that returns all the data about all the traffic signs.  
  
    Returns:  
    list: a list of dictionary containing all the details of the signs.  
    """  
    r = ReadData(self.path)  
    return r.supported_TS()
```

מחלקה :Technical

קלט – נתיב למאגר התמונות.

פלט – מאגר תמונות בעל גודל אחיד (ושמות תמונות באופן עקי ומסודר).

תפקיד המחלקה, הינו בפן הטכני לצורך בהירות וסדר הקוד.

אופן הפעולה:

הספריות הנדרשות:

```
import os  
import cv2
```

פונקציית מחלקה ראשונה, `resize_images()`, תפקידה, נתינת גודל אחיד לכל התמונות במאגר.



```

class Technical:
    """Helped with system clarity and order."""

    def __init__(self, path):
        self.path = path

    def resize_images(self):
        """
        method resize the images.

        Returns:
        none.
        """

        for dir in os.listdir(self.path):
            # מ放开 כל גלובץ בתיקינו
            for f in os.listdir(
                os.path.join(self.path, '/{}'.format(dir))
            ):
                # במנדרט והרנפער תמונה
                if f.endswith('.jpg') or f.endswith('.JPG'):
                    # resize
                    img_dir = dir
                    img_name = f
                    # בונא לתמונה
                    img = cv2.imread(
                        os.path.join(self.path, '/{}/{}'.format(img_dir, img_name))
                    )
                    # גתינעם הגדיל מרצף
                    dim = (const.resize_image, const.resize_image)

                    resized = cv2.resize(img, dim)
                    # מונענו דבשנה בגדיל מרצף
                    status = cv2.imwrite(
                        os.path.join(self.path, '/{}/{}'.format(img_dir, img_name), resized))

```

פונקציית מחלקת הבאה, הינה (`change_file_name()`, תפקידה הוא שינוי שמות התמונות
[איסוף תמונות התמוריים הרבים, יצר מאגר מלא תמונות עם שמות לא אחידים ומראה לא מסודר –
לצורך כך יצרתי את הפונקציה זו], אך אין לה השכלה כלשהם על פעילות המערכת].



```

def change_file_name(self):
    """
    method rename the name of images.

    Returns:
    none.
    """
    i = 1
    for folder in os.listdir(self.path): # עבוק כל תיקה במאגר
        currentPath = os.path.join(self.path, '\\', folder) # סתכל ברגע על תיקה הנוכחית
        os.chdir(currentPath) # עבוק כל גובץ בתיקה הנוכחית
        for file in os.listdir(): # מזג את פירומת הגובץ
            f_ext = ".JPG" # שם חדש
            new_name = '{}@{}@{}.'.format(folder, i, f_ext) # בנית השם החדש
            os.rename(file, new_name) # איזום שרד ה-ז' עבוק הגבאים שבתיקה הנוכחית
            i += 1 # אתחול שרד ה-ז' מזג עבוק תיקה הבא
    i = 1

```

מחלקה Algorithm

קלט – נתיב לティקיה ובה נשמרים קבצי הויידאו שנשלחו ע"י המשתמש, וכן שם קובץ הויידאו הנוכחי.
פלט – רשימה המכילה את מספרי התמוריים שזוהו בסרטון.

תפקיד המחלקה, הרצת כל המערכת בצד שרת, כולל את עיבוד קובץ הויידאו שהועלה ע"י המשתמש, שליחת הפריים (שנוצרו מעבודה הסרטון) לזהוי במחלקה Detect ולסואג במחלקה Predict (מפורטות לעיל), והחזירה של רשימת התמוריים שזוהו במהלך הסרטון, וכן כתיבת תיעוד מפורט של התמוריים שזוהו בקובץ Log.

אוף הפעולה:

הספריות הנדרשות:

```

import logging
import os
import cv2
import numpy as np
from ProjectNeurelNetwork.BL.Constant import Constants
from ProjectNeurelNetwork.DAL.trafficSign import TrafficSign
from ProjectNeurelNetwork.Server.DetectTS import DetectTS
from ProjectNeurelNetwork.Server.predict import Predict

```



פונקציה מחלקה ראשונה, ()`get_frame_video`, תפקידה הינו עיבוד קובץ הוידאו (שהועלה ע"י המשתמש) לפרייםים.

מכל שנייה בסרטון ישנו 24 פרייםים, ארצתה 2 פרייםים בלבד, קלומר כל חצי שנייה לבדוק את מצב הכביש.

אוף הביצוע:

```
class Algorithm:
    """Main class that runs the system process on the server side"""

    def __init__(self, path, video):
        self.base_path = path
        self.video = video

    def get_frame_video(self):
        """
        class method that processes the video to frames.

        Returns:
        buf_2frame (list): The list of frames.
        """

        # capture the video
        vid = os.path.join(self.base_path, '/{}'.format(self.video))
        cap = cv2.VideoCapture(vid)
        frameCount = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        frameWidth = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        frameHeight = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

        # תכנת רשימת שטיגוט את הפרייםים לפ. מסך מינדיים:
        # 1. מסך הפרייםים 2. גובה הפרייםים 3. אורך הפרייםים 4. שטוחות שנמצאים
        buf = np.empty((frameCount, frameHeight, frameWidth, 3), np.dtype('uint8'))
        buf_2frame = []
        fc = 0
        ret = True
```

```
# כתיבת ה-frames לרשימה שהוכנה מראש
while fc < frameCount and ret:
    ret, buf[fc] = cap.read()
    fc += 1

# in every second in a video there is 24 frames, and i dont need all 24 i want only 2
for i in range(0, len(buf), 12):
    buf_2frame.append(buf[i])
# סגירת הצליל
cap.release()
return buf_2frame
```

הfonkzia הבאה הינה, `Execute()`, תפקידה הרצת כל המערכת בצד שרת.



ההרצאה כוללת קריאה לפונקציות השונות לצורכי זיהוי הסרטון, בין היתר פונקציית DetectTS (המופורטת במחלקה detect, get_frame_video,-algorithms מרכזים), predict (המופורטת במחלקה Predict, TSClassification סעיף קוד התוכנית).

אוף הביצוע:

```
def Execute(self):
    """
    class method. activates the main methods in the system, and writes the data returned to a log file.

    Returns:
        none.
    """
    # נציג מופעים עבוג מהזגוגית משוגעת
    t = TrafficSign(Constants.ExcelDirectory)
    a = Algorithm(self.base_path, self.video)
    p = Predict()
    d = DetectTS()
    # מכיל את הפרמטרים שבודקו מנגנון הרכשות
    listOfImages = a.get_frame_video()
    # השימת מכילה את הפרמטרים לאחכ ולחזור תמונות מוגדרות וחישובן
    listDetect = d.detect(listOfImages)
    # מכיל את ממצאים תומכון במתן דיווחים
    listRecognize = p.TSClassification(listDetect)
    # כתיבת הממצאים לוג ותומכון
    for i in listRecognize:
        logging.basicConfig(filename='LogFile_{0}'.format(self.video),
                            format='%(asctime)s - %(message)s', datefmt='%d-%b-%y %H:%M:%S')
        logging.warning(t.get_alert(i))
    return listRecognize
```

חלק ב' – API

שליחת המידע המתבקש לצד ה-client של המערכת ולהפוך, מ ממשת את התקשרות בין ה-client ו-server

אוף הפעולה:

הספריות הנדרשות:

```
from flask import Flask, jsonify, request
from flask_cors import CORS, cross_origin
```

נתינת הרשאה לשילוח הנתונים לצד הלקוח, וקבלת נתונים מהלקוח.



```
app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
```

אוף הפעולה:

קיבלה נתונים על מספר תמרור ספציפי.

```
@app.route('/getAlert/<int:id>', methods=['GET'])
@cross_origin()
def get_alert(id):
    """
    A method that returns data about a specific sign.

    Parameters:
        id (int): Number of traffic sign.

    Returns:
        Dictionary within json: a dictionary containing details about a traffic sign.
    """
    return jsonify(t.get_alert(id))
```

קיבלה נתונים על כלל התמരורים.

```
@app.route('/getTrafficSigns', methods=['GET'])
@cross_origin()
def get_list():
    """
    A method that returns data about all the signs.

    Returns:
        list within json: a list of dictionary containing all the details of the signs supported by the system.
    """
    return jsonify(t.get_supported_TS())
```

קיבלה נתונים על תמרור/ים מסויימים התואמים לשם או הקטגוריה הרצויים.



```
@app.route('/getTrafficSignsByQuery<string:query>', methods=['GET'])
@cross_origin()
def get_filteredList(query):
    """
    A method that returns data about a wanted sign(s).

    Parameters:
        query (str): Name of the sign or category to search.

    Returns:
        Dictionary within json: a dictionary containing sign(s) that matched the search.
    """
    list_TS = t.get_supported_TS()
    new_list = list(
        filter(lambda item: item['CategoryTrafficSign'] == query or item['NameTrafficSign'] == query, list_TS))
    return jsonify(new_list)
```

קיבלה קובץ הودיאו שנשלח ע"י המשתמש, העברתו לעיבוד, ושליחת תוצאה העיבוד לצד ה client.

```
@app.route('/uploadFile', methods=['POST'])
@cross_origin()
def upload_videoFile():
    """
    A method that receives a video file from the client side, and sends it for processing.

    Returns:
        string: The name of the video for further processing.
    """
    if request.method == 'POST':
        f = request.files['file']
        print(f.filename)
        f.save(Constants.videoFolder + '\{}\'.format(f.filename))
        a = Algorithm(Constants.videoFolder, f.filename)
        data = a.Execute()
        return jsonify(data)
    else:
        return 'file wasn\'t uploaded'
```

קוד התוכנית מצד ה Client

Main component

מטרת הקומפוננטה, הינה ניות בין האפשרויות הפונקציונליות של המערכת.

אוף הפעולה:



```
import React, { Component } from 'react';
import './StyleMain.css';

// קומפוננטה ראשית המשמשת לנירוט בין אפשוריות המערכת
export default class Main extends React.Component {
  render() {
    return (
      <div className="main1">
        <div className="main">
          <div className="alert">
            <h2>Alert</h2>
            <span className="exp">Receive alerts about the signs that appear during a road trip</span>
            <div className="btn">
              <a className="text_link" href="/alertToDriver">Get alert</a>
            </div>
          </div>
          <div className="table">
            <h2>Table</h2>
            <span className="exp">View the signs and their details supported by the system</span>
            <div className="btn">
              <a className="text_link" href="/showTable">View traffic - signs</a>
            </div>
          </div>
        </div>
      </div>
    );
  }
}
```

בלחיצה על אחת מתגיות הקישור, תופיע הקומפוננטה הרצויה.

```
<a className="text_link" href="/showTable">View traffic - signs</a>
```

Alert component

מטרת הקומפוננטה הינה, הצגת התראות על זיהוי תמרורים לנהג.

אוף הפעולה:



```
import React, { Component } from 'react';
import { element } from "prop-types";
import './alertStyle.css';
import { ITrafficSign } from '../interfaceTS/TrafficSigns';
import MoreDetails from '../MessageBox/MoreDetails';
import UploadFile from '../UploadFile/uploadVideo';
import '../UploadFile/input.css';
import { func } from 'prop-types';

export default class AlertComponent extends React.Component {
  constructor(props: any) {
    super(props);
    this.changingFileSelection = this.changingFileSelection.bind(this);
    this.onClickSend = this.onClickSend.bind(this);
  }

  TSDetection: Array<number> = new Array<number>();
  interval: any;
  id: any;

  state = {
    alertToDriver: {
      "CategoryTrafficSign": "", "NameTrafficSign": "",
      "NumberTrafficSign": 0, "PictureTrafficSign": "", "SignificanceTrafficSign": "",
      "AlertSignification": ""
    },
    show: false, selectedFile: null, showPopup: true, index: 0
  }
}
```

פונקציה לקליטת הקובץ שהועלה ע"י המשתמש. מתרחשת בלחיצה על כפתור 'העלאת קובץ'.

```
// העלאת קובץ
changingFileSelection(event: any) {
  let file = event.target.files[0];
  this.setState({
    selectedFile: file
  });
}
```

פונקציית API לשילוח הקובץ לשרת, בנוסף מפעילה את הפונקציה להציג ההתראות.



```
// - response - שליחת הקובץ הנבחר לשרת וקבלת ה
// רשיימה המכילה את מספרי התמורות שזוהה והפעלת פונקציה להציג התראות
async onClickSend() {
    this.setState({ showPopup: false })
    // במידה ונבחר קובץ
    if (this.state.selectedFile) {
        const data = new FormData();
        data.append('file', this.state.selectedFile || "");
        const requestOptions = {
            method: 'POST',
            body: data
        };
        // שליחת הקובץ לשרת
        await fetch("http://127.0.0.1:5000/uploadFile", requestOptions)
            .then(res => res.json())
            // קבלת התגובה מהשרת
            .then(item => {
                item.map((i: any) => this.TSDetection.push(i))
            })
            .catch();
        // הפעלת הסרטון במקביל לשילוח התראות
        var v = document.getElementsByTagName("video")[0];
        v.play();
        // הפעלת פונקציית שליחת התראות
        this.setInterval();
    }
}
```

הפעלת הפונקציה השולחת התראות כל חצי שנייה (ע"י ה – interval). [כזכור זההו בצד השרת ה被执行 עבור כל חצי שנייה בסרטון].



```
setInterval() {
    // הפעלת האינטראול כל דצ"נ שנה
    this.interval = setInterval(this.showAlert, 500)
}

// כל יוזת החתראות
showAlert = () => {
    let index = this.state.index;
    // במידה וקיים שלושה רצפים של זיהויים שונים של חדת התראה
    if (this.TSDetection[index] == this.TSDetection[index + 1]
        && this.TSDetection[index] == this.TSDetection[index + 2]) {
        // כל יוזת מס' התמרור
        this.getAlertToDriver(this.TSDetection[index]).
            then(item => { this.state.alertToDriver = item; })
    }
    else {
        //nothing
        this.getAlertToDriver(8).
            then(item => { this.state.alertToDriver = item; })
    }
    // קידום אינדקס
    this.setState({ index: index + 1 });
    // אם הגיענו לסוף המערך - נרצה לעצור את האינטראול
    if (this.TSDetection.length == index + 1) {
        clearInterval(this.interval);
    }
}
```

פונקציית API המקבלת מס' התמרור, ומחזירה מהשרת את כל הנתונים לגביו.



```
// קבלת הנתונים מהשרת על מספר תמרור לצורך הצגת החטר&gt;
async getAlertToDriver(i: number) {
    let obj: ITrafficSign = {
        "AlertSignafication": "", "CategoryTrafficSign": "", "NameTrafficSign": "",
        "NumberTrafficSign": 0, "PictureTrafficSign": "", "SignificanceTrafficSign": ""
    };
    this.id = i
    await fetch("http://127.0.0.1:5000/getAlert/" + this.id)
        .then(res => res.json())
        .then(data => {
            debugger;
            obj = ({
                AlertSignafication: data.AlertSignafication,
                NumberTrafficSign: data.NumberTrafficSign,
                NameTrafficSign: data.NameTrafficSign,
                CategoryTrafficSign: data.CategoryTrafficSign,
                SignificanceTrafficSign: data.SignificanceTrafficSign,
                PictureTrafficSign: data.PictureTrafficSign,
            })
        }).catch()
    return obj;
}
```

הציגת הקומפוננטה:

```
// בלחיצה על כפתור יופיעו נתוניםigosים על תמרור
getMessageBox = (bool: boolean) => {
    this.setState({ show: bool });
}

render() {
    const show = this.state.show;
    const showFile = this.state.showPopup;

    return (<div>
        <div className="body">
            {!showFile && <div className="main_alert">
                <div className="container">
                    <span className="blink">
                        <img src={`PictureForAlert/${this.state.alertToDriver.PictureTrafficSign}`}></span>
                    <p className="alert_driver">{this.state.alertToDriver.SignificanceTrafficSign}</p>
                    <button className="details" onClick={() => this.getMessageBox(true)}>More details</button>
                {show && <MoreDetails messageCategory={this.state.alertToDriver.CategoryTrafficSign}
                    messageName={this.state.alertToDriver.NameTrafficSign}
                    messageNumber={this.state.alertToDriver.NumberTrafficSign}
                    clickAgain={this.getMessageBox}>
                </MoreDetails>}
                <audio className="voice" ref="audio_tag"
                    src={`AlertSound/${this.state.alertToDriver.AlertSignafication}`}>
                    <controls autoPlay />
                </audio>
            </div>
        </div>
    )
}
```



```
<div>
    <div>
        <div>
            <div>
                <div>
                    <div>
                        <div>
                            <div>
                                <div>
                                    <div>
                                        <div>
                                            <div>
                                                <div>
                                                    <div>
                                                        <div>
                                                            <div>
                                                                <div>
                                                                    <div>
                                                                        <div>
                                                                            <div>
                                                                                <div>
                                                                                    <div>
                                                                                        <div>
                                                                                            <div>
                                                                                                <div>
                                                                                                    <div>
                                                                                                        <div>
                                                                                                            <div>
                                                                                                                <div>
                                                                                                                    <div>
                                                                                                                        <div>
                                                                                                                            <div>
                                                                                                                                <div>
                                                                                                                                    <div>
................................................................
```

תת קומפוננטה להציג הנתונים הנוספים על התמורה.

```
import React from "react";
import { ITrafficSign } from '../interfaceTS/TrafficSigns';
import './styleMessageBox.css';

// מקבלת נתונים מהקונטנרט הרשאית
export interface IMessageToDriver {
    messageCategory: string;
    messageName: string;
    messageNumber: number;
    clickAgain: (bool: boolean) => void;
}

export default class MoreDetails extends React.Component<IMessageToDriver>{
    constructor(props: IMessageToDriver) {
        super(props);
    }
    state = {show: true}

    // בליחיצה על כפתור תציג או תסגור
    hide() {
        this.setState({ show: !this.state.show });
        this.props.clickAgain(false);
    }
}
```



```
render() {
  const show = this.state.show;
  return (
    <div>
      {show && <div>
        <div className="containerMB">
          <span role="button" className="btn-close" onClick={() => this.hide()}>&times;</span>
          <span className = "content">
            Category: {this.props.messageCategory}<br>
            Name: {this.props.messageName}<br>
            Number: {this.props.messageNumber}<br>
          </span>
          <button className="ok" onClick={() => this.hide()}>OK</button>
        </div>
      </div>
    )
}
```

showTable component

מטרת הקומponentה הינה הצגת התמורות הננתמכים והמידע עליהם.

אוף הפעולה:

```
import React from "react";
import { element } from "prop-types";
import { ITrafficSign } from '../interfaceTS/TrafficSigns';
import SearchTS from '../SearchTS/SearchTS';
import './StyleTable.css'

export default class ShowTableComponent extends React.Component {
  signsTable: Array<ITrafficSign> = [];
  query = "";

  state = { results: new Array<ITrafficSign>() }

  //ונציגיה שטפניאל את render(עומדת לפני ה- render( מופעלת (מחזרת)
  //חזרו מוכנים לפניה שהקומponentה תונצח
  async componentDidMount() {
    this.getTrafficSigns().then(item => { this.signsTable = item; this.setState({ results: item }) });
  }
}
```



```
פונקציה לקבלת הנתונים מהשרת על כל התמורות//
async getTrafficSigns() {
    let arr: Array<ITrafficSign> = new Array<ITrafficSign>();
    await fetch("http://127.0.0.1:5000/getTrafficSigns")
        .then(res => res.json())
        .then(data => {
            debugger;
            data.map((i: any) => arr.push({
                CategoryTrafficSign: i.CategoryTrafficSign,
                NameTrafficSign: i.NameTrafficSign,
                NumberTrafficSign: i.NumberTrafficSign,
                SignificanceTrafficSign: i.SignificanceTrafficSign,
                PictureTrafficSign: i.PictureTrafficSign,
                AlertSignification: i.AlertSignification
            }))
        }).catch()
    return arr;
}
```

```
פונקציה לקבלת הנתונים על התמורות שנמצאו תואמים לחיפוש//
async getTableItemByQuery() {
    let arr: Array<ITrafficSign> = new Array<ITrafficSign>();
    await fetch("http://127.0.0.1:5000/getTrafficSignsByQuery/" + this.query)
        .then(res => res.json())
        .then(data => {
            debugger;
            data.map((i: any) => arr.push({
                CategoryTrafficSign: i.CategoryTrafficSign,
                NameTrafficSign: i.NameTrafficSign,
                NumberTrafficSign: i.NumberTrafficSign,
                SignificanceTrafficSign: i.SignificanceTrafficSign,
                PictureTrafficSign: i.PictureTrafficSign,
                AlertSignification: i.AlertSignification
            }))
        })
    }).catch()
    return arr;
}
```



```
// מנגנון חיפוש מיידני על מחרוזת
onSearch = (query: string) => {
    this.query = query;
    this.getTableItemByQuery().then(items => this.setState({ results: items }))
}

render() {
    return (<div className="body">
        <SearchTS onSearch={this.onSearch}></SearchTS>
        <h1>Supported signs</h1>
        <p className="match">{this.state.results.length} matching traffic - sign/s found</p>
        <table>
            <tbody>
                <tr><th>number</th><th>name</th><th>category</th><th>significance</th><th>picture</th></tr>
                {this.state.results.map(i => <tr><td>{i.NumberTrafficSign}</td><td>{i.NameTrafficSign}</td>
                    <td>{i.CategoryTrafficSign}</td><td>{i.SignificanceTrafficSign}</td>
                    <td><img src={require(`../PictureForTable/${i.PictureTrafficSign}`)} height={60} width={65} /></td></tr>)}
            </tbody>
        </table>
    </div>
}
```

תת קומפוננטה להציג התמורות/ים הרצויים (עפ"י חיפוש המשתמש).



```
import React from "react";
import './ShowTable/StyleTable.css';

// ממשק לקבלת רשימת תוכן תיבת ה לחיפוש
export interface ISearchItem {
    onSearch: (item: string) => void;
}

export default class SearchItem extends React.Component<ISearchItem>{
    constructor(props: ISearchItem) {
        super(props)
    }

    // באחיזה על מקש ה Enter או על חיבור/ריץ הרצויים
    search_ts(e: React.KeyboardEvent<HTMLInputElement>) {
        if (e.key === 'Enter') {
            let value = (document.getElementById("search") as HTMLInputElement).value;
            this.props.onSearch(value);
        }
    }

    render() {
        return (<div>
            <div>
                <input className="searchTS" id="search"
                    placeholder="Search traffic - sign by category or name here..." 
                    onKeyPress={(e) => this.search_ts(e)}></input>
            </div>
        </div>
    }
}
```

מוד נתונים:

1. מקור נתונים המאחסן מידע על כל תמרור בקובץ אקסל המכיל:
 - מספר זהה של תמרור.
 - מספר הסמל.
 - שמו.
 - הקטגוריה אליה הוא משתייך.
 - משמעתו.
 - תמונה שלו.
 - צילול התראה המתאים לו.

מידע זה נלמד והינו על פי כל כללי משרד התחבורה הישראלי.



2. מסד נתונים מורכב מתמונות רבות של תמרורים שאספתי ע"י צילום עצמי וכן מתוך תמונות קיימות באינטרנט.

בנוסף, על – מנת לעובות את מאגר התמונות בנייטי קוד עם יכולת פונקציונלית של עיבוד תמונה שיצרת מכל תמונה – מספר תמונות שונות (אופן הפעולה – כמפורט לעיל בסעיף קוד התוכנית ImageProcessing) עד שהגעתי לכמות שכוללת של כמעל 23,000 תמונות!

בנוסף, נתינת גודל אחיד לכל התמונות במאגר עבר אימון תקין ונכון של הרשותות השונות (אופן פעולות הפונקציה, כמפורט בסעיף קוד התוכנית, מחלוקת Technical בפונקציית `UniformSize`).

עבור הרשת המבוססת על CNN – גודל זה הוחלט לאחר שיקול שכלל כמה היבטים, מצד אחד, הצורך שהתמונה תהיה כמה שיותר קטנה – כך שהקלט יהיה קטן יותר, מצד שני, לא מדי קטן שהרשת תדע לזהות את התמרור. לאחר מספר ניסיונות נראה שהגודל הנ"ל מתאים.

ופעם נוספת בגודל 224 – שזהו הגודל הנדרש לרשת נוירונים מבוססת `VGG16`.

לאחר מכן, חלוקת מאגר הנתונים ל – 3 תת-מאגרים עליהם יתאמנו הרשותות. (כמפורט בסעיף קוד התוכנית מחלוקת `DataOrganization` בפונקציית `TheFile`)

בתמונה הבאה ניתן לראות את החלוקה המפורטת לתיקיות, כל תיקייה מכילה 13 קטגוריות שונות של תמרורים.



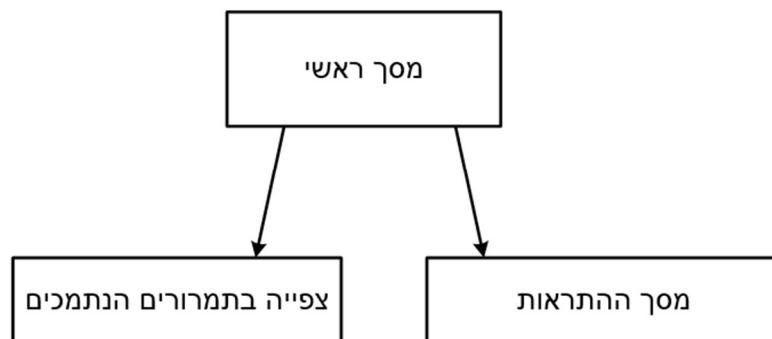
```
RoadConstruction
UpTo30kmh
UpTo50kmh
UpTo80kmh
Validate
BumpyRoad
CloseToVehicles
CurvesAlongTheWay
DangerOfSlipping
DangerousPlace
GiveWay
Movement circle
NoEntry
Nothing
RoadConstruction
UpTo30kmh
UpTo50kmh
UpTo80kmh
```

```
Test
BumpyRoad
CloseToVehicles
CurvesAlongTheWay
DangerOfSlipping
DangerousPlace
GiveWay
Movement circle
NoEntry
Nothing
RoadConstruction
UpTo30kmh
UpTo50kmh
UpTo80kmh
Train
BumpyRoad
CloseToVehicles
CurvesAlongTheWay
DangerOfSlipping
DangerousPlace
GiveWay
Movement circle
NoEntry
Nothing
```



תיאור מסכימים:

תרשים מסכימים – Screen flow diagram



תרשים מסכימים

תפקידי המרכיבים:

- מסך הצגת התראות לנהג.



- מסך הצגת התמרוריהם הנתמכים.



Search traffic - sign by category or name here...

Supported signs

12 matching traffic - sign found

| number | name | category | significance | picture |
|--------|-----------------------|-----------------------------------|--|---------|
| 401 | Close to vehicles | Prohibition and restriction signs | Notice! road closed in both directions to every vehicle | |
| 303 | Movement circle | Priority signs | Yield to vehicles already in the traffic circle or entering it in front of you | |
| 101 | Bumpy road | Warning signs | Notice! bumpy road ahead | |
| 105 | Curves along the road | Warning signs | Curve left and then right on the road ahead | |
| 141 | Danger of slipping | Warning signs | Caution! Slippery road ahead | |
| 150 | Dangerous place | Warning signs | Dangerous place ahead for which no special sign has been set | |
| 301 | Give way | Priority signs | Yield to vehicles in the intersection or on the track in front of you | |

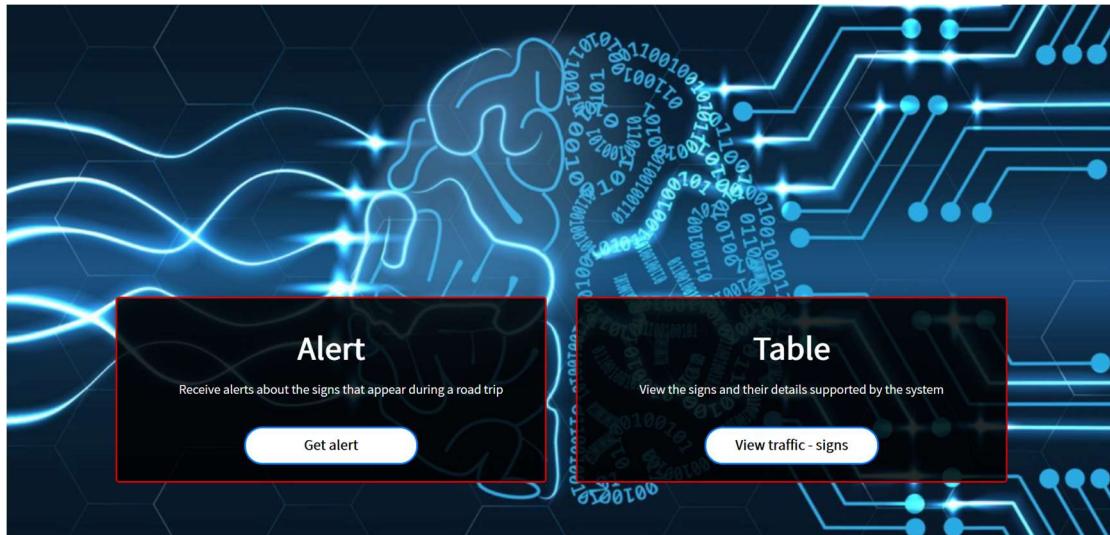
| | | | | |
|-----|-----------------------|-----------------------------------|---|--|
| 105 | Curves along the road | Warning signs | Curve left and then right on the road ahead | |
| 141 | Danger of slipping | Warning signs | Caution! Slippery road ahead | |
| 150 | Dangerous place | Warning signs | Dangerous place ahead for which no special sign has been set | |
| 301 | Give way | Priority signs | Yield to vehicles in the intersection or on the track in front of you | |
| 402 | No entry | Prohibition and restriction signs | .Notice! entry to any vehicle is prohibited | |
| 901 | Road construction | Construction site signs | .Construction site ahead. Follow the orange markings, if marked | |
| 426 | Up to 30 km/h | Warning signs | Driving at a speed exceeding 30 km/h is forbidden | |
| 426 | Up to 50 km/h | Warning signs | Driving at a speed exceeding 50 km/h is forbidden | |
| 426 | Up to 80 km/h | Warning signs | Driving at a speed exceeding 80 km/h is forbidden | |

תיאור מסך הפתיחה

מסך הפתיחה משמש כנקודות ניוט לאפשרויות הפונקציונליות של המערכת.

אפשרות אחת, הינה העלאת סרטון המכיל מהלך נסעה, שליהתו לשרת, וקבלת התראות על משמעות התמורות המזוהים.

אפשרות נוספת, היא צפיה במאפייני התמורות הנתמכים ע"י מערכת, וכן אפשרות של חיפוש מידע על תמורה ספציפי, באמצעות הקלדת שמו או הקטגוריה אליה הוא משתייך לתיבת ה –սעוקו.



מסכי האפליקציה בלוני תיעוד והסביר מפורטים:
במסך זה המשמש יעלת סרטון של מהלך נסעה למערכת, ע"י לחיצה על כפתור ה

- עללאת קובץ.

בחירה קובץ

ולאחר מכן שליחתו לשרת, באמצעות לחיצה על כפתור ה

upload
.

Upload

Upload a video



לא נבחר קובץ

Upload

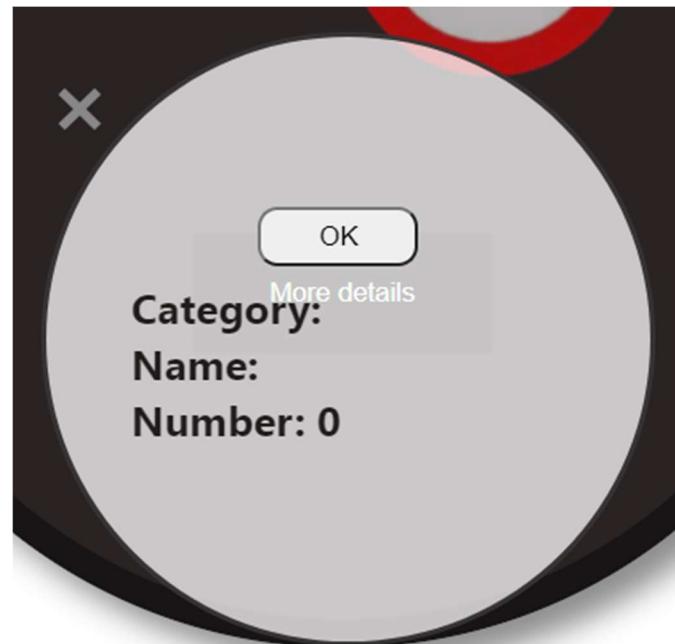
השרת יעבד את הסרטון שהתקבל, ושלח התרעות לנגן בהתאם לתרומות המזוהים, התרעות מתקבלות בمعنى מכשיר המוצג על הדפסן, ותפקידו להציג את התרעות המתקבלות מהשרת.



הנהג רואה את התטרעה ע"י הצגת תמונה התמרור שזזה באופן מהbehav, מה שמסב את תשומת ליבו, והמשמעות מוצגת על גבי מסך המכשיר, בនוסף ישמע צליל התטרעה עם שם התמרור שזזה. וכל זאת על – מנת להסביר את תשומת לב הנהג באופן המקסימלי, ולשמור על ערנותו לתמרורים בזמן נהיגה.

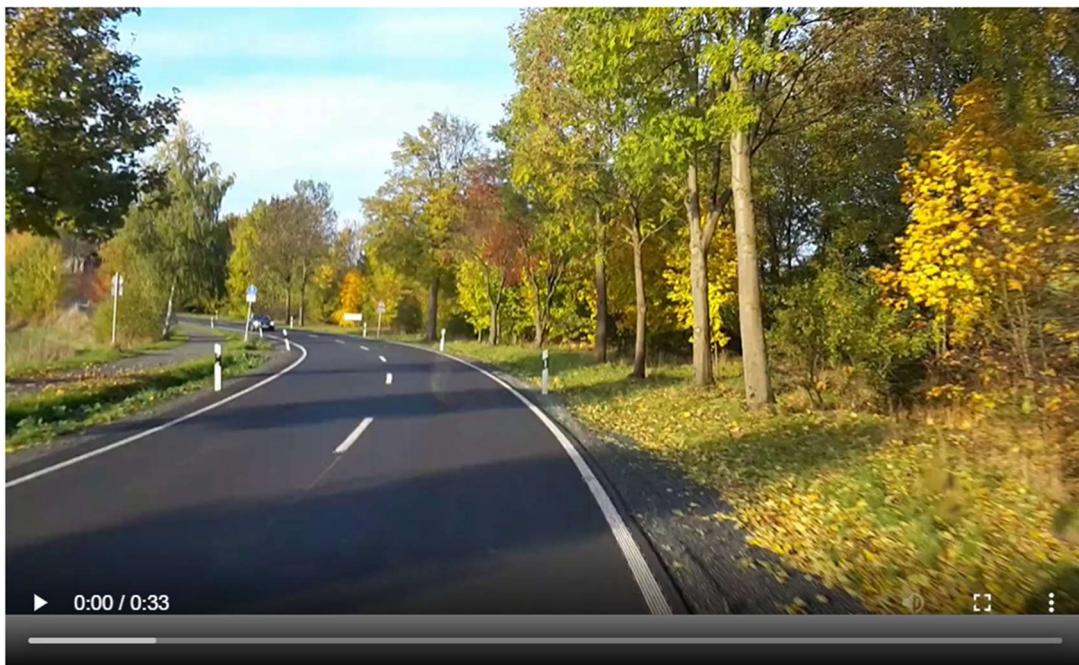
וכן, ישנה אפשרות אף לראות את מאפייני התמרור שזזה – בלחיצה על כפתור ה – more details – יופיע窗口 message – שמציג את מספר התמרור, שם התמרור, והקטגוריה אליו משתייך התמרור.





ובלחיצה על כפתור OK או על X – ה – message box ייסגר, אך תמכתי באפשרות שניתן יהיה לראותו שוב, ע"י לחיצה נוספת על כפתור ה – more details.

במקביל ניתן לצפות בסרטון של מהלך הנסיעה שהועלה ע"י המשתמש, ולראות שאכן כאשר מופיע תמרור, ישנה התראעה מתאימה המופיעה ב'מכשיר ההתרעות'.





מסך נוסף, הימנו הצגת התמרורים הנתמכים ע"י המערכת, ולכל תמרור מוצג מידע רלוונטי לגביו.

Search traffic - sign by category or name here...

Supported signs

12 matching traffic-sign found

| number | name | category | significance | picture |
|--------|-----------------------|-----------------------------------|--|---------|
| 401 | Close to vehicles | Prohibition and restriction signs | Notice! road closed in both directions to every vehicle | |
| 303 | Movement circle | Priority signs | Yield to vehicles already in the traffic circle or entering it in front of you | |
| 101 | Bumpy road | Warning signs | Notice! bumpy road ahead | |
| 105 | Curves along the road | Warning signs | Curve left and then right on the road ahead | |
| 141 | Danger of slipping | Warning signs | Caution! Slippery road ahead | |
| 150 | Dangerous place | Warning signs | Dangerous place ahead for which no special sign has been set | |
| 301 | Give way | Priority signs | Yield to vehicles in the intersection or on the track in front of you | |

| | | | | |
|-----|-----------------------|-----------------------------------|---|--|
| 105 | Curves along the road | Warning signs | Curve left and then right on the road ahead | |
| 141 | Danger of slipping | Warning signs | Caution! Slippery road ahead | |
| 150 | Dangerous place | Warning signs | Dangerous place ahead for which no special sign has been set | |
| 301 | Give way | Priority signs | Yield to vehicles in the intersection or on the track in front of you | |
| 402 | No entry | Prohibition and restriction signs | .Notice! entry to any vehicle is prohibited | |
| 901 | Road construction | Construction site signs | .Construction site ahead. Follow the orange markings, if marked | |
| 426 | Up to 30 km/h | Warning signs | Driving at a speed exceeding 30 km/h is forbidden | |
| 426 | Up to 50 km/h | Warning signs | Driving at a speed exceeding 50 km/h is forbidden | |
| 426 | Up to 80 km/h | Warning signs | Driving at a speed exceeding 80 km/h is forbidden | |

כאמור לעיל, ניתן לסנן תמרורים על פי שם או קטגוריה בתיבת ה – **טבקו!**, ויוצגו רק התמרורים המתאימים וכן מספר התמרורים שנמצאו מתאימים לשם או הקטגוריה הנבחרת.

תיבת ה – **:input**:

Search traffic - sign by category or name here...



מספר התמרורים שנמצאו מתאימים:

12 matching traffic - signs found

מדריך למשתמש:

בעת כניסה למערכת, עומדות בפני המשתמש 2 אפשרויות,
הachat, אפשרות לקבלת התרעה על שימוש התמרורים במהלך נסיעה.
בעת בחירת אפשרות זו, על המשתמש להעלות קובץ וידאו של מהלך נסעה ויכול לצפות בהתראות
המתתקבלות במקביל ובהתאם למופיע בסרטון.
השנייה, צפייה בכל התמרורים בהם המערכת תומכת.
במידה והמשתמש בחר את האפשרות זו, על המסר יופיעו התמרורים הנתמכים ע"י המערכת
ומאפייניהם.
בנוסף אם המשתמש רוצה לחפש מידע אחר תמרור ספציפי, הוא יכול כתוב בתיבת ה – **Search** המופיעה
בראש העמוד את שם התמרור אותו הוא מחפש או הקטגוריה אליה הוא משתמש, ובעת לחיצה על
מקש ה – **Enter**, יופיע לו התמרור יחד עם מאפייניו, במידה והמערכת תומכת בתמרור זה.

בדיקות והערכתה:

בדיקות והערכתה:

על – מנת להגיע לרשות עם אחוזי הדיווק הגבוהים ביותר, בצעדי מספר בדיקות, והערכתה של
הרשות שהמודול יוצר בכל הרצה.
ניתן לראות שבהרצתה הראשונה המודול דijk ב – 95%.



```

Epoch 1/15
152/152 [=====] - 212s 1s/step - loss: 1.2838 - accuracy: 0.6416 - val_loss: 0.4691 - val_accuracy: 0.8676
Epoch 2/15
152/152 [=====] - 223s 1s/step - loss: 0.2795 - accuracy: 0.9247 - val_loss: 0.2794 - val_accuracy: 0.9334
Epoch 3/15
152/152 [=====] - 232s 2s/step - loss: 0.1079 - accuracy: 0.9713 - val_loss: 0.2504 - val_accuracy: 0.9412
Epoch 4/15
152/152 [=====] - 228s 2s/step - loss: 0.0580 - accuracy: 0.9850 - val_loss: 0.2670 - val_accuracy: 0.9497
Epoch 5/15
152/152 [=====] - 230s 2s/step - loss: 0.0343 - accuracy: 0.9940 - val_loss: 0.2889 - val_accuracy: 0.9467
Epoch 6/15
152/152 [=====] - 232s 2s/step - loss: 0.0289 - accuracy: 0.9932 - val_loss: 0.2891 - val_accuracy: 0.9534
Epoch 7/15
152/152 [=====] - 233s 2s/step - loss: 0.0209 - accuracy: 0.9952 - val_loss: 0.2699 - val_accuracy: 0.9560
Epoch 8/15
152/152 [=====] - 250s 2s/step - loss: 0.0081 - accuracy: 0.9989 - val_loss: 0.2914 - val_accuracy: 0.9564
Epoch 9/15
152/152 [=====] - 248s 2s/step - loss: 0.0120 - accuracy: 0.9973 - val_loss: 0.3266 - val_accuracy: 0.9571
Epoch 10/15
152/152 [=====] - 234s 2s/step - loss: 0.0137 - accuracy: 0.9976 - val_loss: 0.3630 - val_accuracy: 0.9449
Epoch 11/15
152/152 [=====] - 231s 2s/step - loss: 0.0250 - accuracy: 0.9941 - val_loss: 0.3968 - val_accuracy: 0.9497
Epoch 12/15
152/152 [=====] - 229s 2s/step - loss: 0.0210 - accuracy: 0.9955 - val_loss: 0.3686 - val_accuracy: 0.9471
Epoch 13/15
152/152 [=====] - 230s 2s/step - loss: 0.0061 - accuracy: 0.9987 - val_loss: 0.3885 - val_accuracy: 0.9475
Epoch 14/15
152/152 [=====] - 231s 2s/step - loss: 0.0088 - accuracy: 0.9982 - val_loss: 0.4715 - val_accuracy: 0.9468
Epoch 15/15
152/152 [=====] - 248s 2s/step - loss: 0.0171 - accuracy: 0.9955 - val_loss: 0.4444 - val_accuracy: 0.9564

```

בעמ נוספת, לאחר הוספה בлок קונבולוציה, המודל דיק ב – 96% זיהוי מוצלח!

```

Epoch 1/12
152/152 [=====] - 162s 1s/step - loss: 1.0096 - accuracy: 0.6732 - val_loss: 0.4076 - val_accuracy: 0.8380
Epoch 2/12
152/152 [=====] - 157s 1s/step - loss: 0.3206 - accuracy: 0.8956 - val_loss: 0.2591 - val_accuracy: 0.9053
Epoch 3/12
152/152 [=====] - 146s 964ms/step - loss: 0.1951 - accuracy: 0.9354 - val_loss: 0.1701 - val_accuracy: 0.9445
Epoch 4/12
152/152 [=====] - 153s 1s/step - loss: 0.1281 - accuracy: 0.9564 - val_loss: 0.1373 - val_accuracy: 0.9560
Epoch 5/12
152/152 [=====] - 155s 1s/step - loss: 0.0886 - accuracy: 0.9700 - val_loss: 0.2283 - val_accuracy: 0.9290
Epoch 6/12
152/152 [=====] - 154s 1s/step - loss: 0.0747 - accuracy: 0.9749 - val_loss: 0.1750 - val_accuracy: 0.9516
Epoch 7/12
152/152 [=====] - 156s 1s/step - loss: 0.0626 - accuracy: 0.9792 - val_loss: 0.1629 - val_accuracy: 0.9553
Epoch 8/12
152/152 [=====] - 161s 1s/step - loss: 0.0502 - accuracy: 0.9826 - val_loss: 0.1528 - val_accuracy: 0.9671
Epoch 9/12
152/152 [=====] - 159s 1s/step - loss: 0.0432 - accuracy: 0.9862 - val_loss: 0.1512 - val_accuracy: 0.9689
Epoch 10/12
152/152 [=====] - 158s 1s/step - loss: 0.0335 - accuracy: 0.9897 - val_loss: 0.1351 - val_accuracy: 0.9671
Epoch 11/12
152/152 [=====] - 159s 1s/step - loss: 0.0317 - accuracy: 0.9904 - val_loss: 0.1439 - val_accuracy: 0.9734
Epoch 12/12
152/152 [=====] - 162s 1s/step - loss: 0.0571 - accuracy: 0.9820 - val_loss: 0.1644 - val_accuracy: 0.9678

```

לאחר מכן, יצרתי רשת נוספת, המבוססת על מודל 16vgg, וקבעתי 93% זיהוי.

Test set classification accuracy: 93.08%

ולכן הרשות הנבחנת (לאחר הבדיקה הסופית [כמפורט בהמשך]), היא הרשות בעלת 96% זיהוי.

בדיקה סופית:



על מנת לבדוק את תקינות המערכת, ולודא שכן רשות המירונים שנבנתה פועלת כמצופה ממנה, ניסיתי את הרשות על עשרות תמונות שהיא אינה ראתה קודם לכן, ושלא התאימה עליה. הרשות סיפקה את התוצאות הרצויות עבור כל תמונה, וסיווגה כראוי את סוג התמரור.

פיתוח עילות:

האלגוריתם המרכזי בפרויקט הינו בניית רשות נירוניים שתסואג את סוג התמרור עבור כל פריטים שהתקבלו מהסרטון.

סיבוכיות האלגוריתם של פיתוח המודל הינה סיבוכיות גבוהה וכן הרצתי מספר מוגדר של epochs, עד לקבלת התוצאה הרצiosa.

א"כ הסיבוכיות הינה כמספר ה – epochs – מספר סבבי האימון שהמודל מריץ כל תמונה הנמצאת בתיקייה ה – .train.

עבור הרשות הנבחרת, ביצעת 15 epoch. המודל יעצר מלהתאמן עקב אחת מ-2 הסיבות:

- גמר את סבבי האימון- הגיעו למספר הepoch המרבי שהוגדרו.
- "יעצר עקב אימון יתר, כך שאם ימשיך לבצע סבבים- ייתן תוצאות פחותות מהסביר הקודם (overfitting)

לאחר מכן כאשר הרשות מוכנה המערכת רצה כמספר שכבות הרשות.

אבלחתת מידע

לא רלוונטי.

מסקנות:

כשניגשתי לתוכנן פרויקט הגמר ראייתי כי המשימה מורכבת, ודורשת השקעה רבה הן בלימוד החומר קודם תחילת בנייתו, הן בתכנון בנייתו - תכנון האלגוריתם והן בבניה עצמה.

בשלב זה ניגשתי לכתיבת הקוד שהוא המרכיב החשוב והמרכזי בפרויקט, היה צורך לכתוב תוך חשיבה עמוקה ולימוד מקיף של התחום.
לכן נדרשנו לתקן ולשפרו מספר פעמים עד אשר הגיעו לתוצאה מדויקת ומושלמת.



כתיבת הקוד הייתה מורכבת ומאתגרת אף בעicker מלמדת, בחלק זה למדתי רבות על תחום הבינה המלאכותית ובפרט על התת תחום: רשות ניירונים, וכן רכשתי לעצמי ידע רחב ומكيف בשפת Python, כיצד להשתמש בפונקציות וספריות רבות ומגוונות ולשמור כל העת על רמת כתיבה גבוהה ונכונה.

ניתן לומר כי הפרויקט טרם לי רבות סטודנטית וכמו הנדסת, הוא הכנס שעניינו עמוק לתוך המבנה הפנימי היוצר כל תוכנת מחשב באשר היא, החל מניתוח ואפיון של המערכת, תוך הקפדה על כך שלשימוש יהיה קל, נעים וברור בזמן השימוש במערכת, וכן שימת לב לעילות האלגוריתם ומתן דגש לדיווקו המרבי, על – מנת שישיג באמצעות את התועלות שניתן להפקה מיפוי זה.

לאחר שעות רבות של عمل ויעז, אני חשה כי באופן אישי הפקתי תועלות רבה מאד מיפוי הפרויקט, ומקווה שההו נדבך נוספת לנסעה בטוחה וקליה יותר, וכן בתחום הרכיבים האוטונומיים.

פיתוחים עתידיים – רעיונות לשיפור:

בגרסה הבאה האפליקציה תשודרג ויורחבו התמורות בהם המערכת תומכת.

בנוסף, המערכת תוכל להתריע על מרחק לא בטוח בין המכונית הנוכחית לאחוריות.

וכן תדע לzechות ולהתריע על הולך רגל הנמצא בטוחה הקרוב של המכונית, כך שאם חילתה הולך רגל התפרץ לככיש, המערכת תתריע ע"כ לנаг, שיוכל לבЛОם בזמן ולמנוע תאונה.

כל זאת ועוד על – מנת לשמר על חי הנוהג באופן ישיר, ועל חי הולך הרגל בעקיפין, ולהפוך את הנסעה לקלה ובטוחה הרבה יותר.

ביבליוגרפיה:

אלגוריתם:

- ✓ [למד למידת מכונה - all-the-tutorials](https://reshetech.co.il/machine-learning-tutorials/all-the-tutorials)
- ✓ [/https://data-flair.training/blogs/ - DataFlair – learn about AI](https://data-flair.training/blogs/)
- ✓ [/https://www.pyimagesearch.com - Learn Deep Learning & OpenCV](https://www.pyimagesearch.com)
- ✓ [- Learn Machine Learning with Python](#)
- ✓ [/https://www.murtazahassan.com/courses/machine-learning-with-python](https://www.murtazahassan.com/courses/machine-learning-with-python)



[למידת מכונה עם GeeksforGeeks](https://www.geeksforgeeks.org/machine-learning/#mp) ✓

[למידת עיבוד תמונה - library](https://reshetech.co.il/python-tutorials/pillow-imaging-library) ✓

[רשות VGG16](https://keras.io/api/applications/vgg/#vgg16-function) ✓
[/https://www.ai-blog.co.il/category/deep - AI-Blog](https://www.ai-blog.co.il/category/deep) ✓

:Python

GitHub - [/https://github.com](https://github.com) ✓
stackOverflow - [/https://stackoverflow.com](https://stackoverflow.com) ✓

[ליימוד Python](https://reshetech.co.il/python-tutorials/all-the-tutorials) ✓
[/https://opencv.org](https://opencv.org) - **OpenCV** ✓

[https://www.programiz.com/python- programming/methods/built-in/min](https://www.programiz.com/python-programming/methods/built-in/min) ✓
[/https://keras.io](https://keras.io) - **Keras** ✓
[/https://www.tensorflow.org](https://www.tensorflow.org) - **TensorFlow** ✓
- **NumPy** ✓

<https://numpy.org/doc/stable/reference/generated/numpy.ndarray.tolist.html>

UI

Mosh Hamedani – I learn Python & React ✓
[/https://programmingwithmosh.com](https://programmingwithmosh.com)

<https://he.reactjs.org/tutorial/tutorial.html> - **React** ✓
[/https://developer.mozilla.org/he](https://developer.mozilla.org/he) – **MDN** ✓
/ <https://www.w3schools.com> - **W3School** ✓
Wikipedia ✓