A picture containing graphical user interface

Description automatically generated

**LPRS**

**(License Plate Recognition System)**

**שם**: דור אברמוביץ

**ת.ז**: 325375012

**שם** מנחה: גולן מור

**חלופה:** הגנת סייבר ומערכות הפעלה

**תאריך**: מאי 2022

תוכן עניינים -

[**תקציר ורציונל הפרויקט** 3](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721813)

[**מבוא ורקע כללי** 4](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721814)

[**מטרת הפרויקט** 5](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721815)

[**שפת התכנות וסביבת העבודה** 7](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721816)

[**ניסוח וניתוח הבעיה האלגוריתמית** 8](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721817)

[**תיאור אלגוריתמים קיימים** 8](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721818)

[**הפתרון הנבחר** 9](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721819)

[**תיאור המודלים של מערכת התוכנה** 9](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721820)

[**תיעוד הקוד** 10](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721821)

[**השוואת העבודה עם פתרונות ויישומים קיימים** 14](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721822)

[**הערכת הפתרון לעומת התכנון והמלצות לשיפורו** 14](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721823)

[**תיאור של הממשק למשתמש – הוראות הפעלה** 15](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721824)

[**ממשק משתמש** 15](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721825)

[**מבט אישי על העבודה ותהליך הפיתוח** 17](file:///C:\VisualCodeProjects\finalProject\LicensePlateNumberDetectionProject\BookOfTheProject-LPRS(LicensePlateRecognitionSystem).docx#_Toc103721826)

# **תקציר ורציונל הפרויקט**

כיום בעידן המודרני והטכנולוגי מכוניות משמשות ככלי תחבורה שמשרת את בני האדם במטרה להגיע ממקום אחד לאחר בין אם זה מהבית לעבודה, לקניות ועוד.

כמו כן יחד אם השימוש הרב בכלי תחבורה יש כאלה שבוחרים להשתמש במכוניות בפזיזות ולמען מטרות לא טובות ולכן צריך משהו שיפקח על מכוניות בדרכים וכשהם מגיעים למקומות מסוימים בין אם זה חניונים לקניון, עבודה ועוד. כך כאשר מתבצע עבירה תנועה כלשהי ניתן למצוא את בין האדם האחראי למעה בעזרת איתור הכתובת האישית של המכונית, לוחית הרישוי שלה. זאת ועוד ניתן לאסור על מכוניות מסוימות להיכנס למקומות מסוימים בכך שבודקים את לוחית הרישוי שלה ואם היא מורשת לעבור.

הפרויקט שלי נועד בדיוק לשם מטרה זו. הפרויקט שלי מאפשר לי להסתכל דרך מצלמת שידור חי בזמן אמת וליראות האם מתבצע עברה או לא. במקרה ומתבצעת עבירת יש לי את היכולת לקחת snapshot של המכונית הרלוונטית ולהוציא את הלוחית רישוי שלה מהתמונה שצלמתי ואז אני יכול לבדוק אם היא נמצאת במערכת על עבירה או שהוא לא נמצא במערכת על עבירה אותו דבר בכניסה לחניות ניתן לאסור על כניסה של מכונית ע"י בדיקה האם הלוחית רישוי של המכונית נמצאת במערכת או לא.

**בחרתי לעשות את הפרויקט הזה מכמה סיבות**:

אני די אוהב מכוניות במיוחד מהסיבה שהתחלתי ללמוד השנה נהיגה. בנוסף לכך הכביש הוא מקום מאוד מסוכן וזה דבר נוסף שלמדתי במהלך שיעורי הנהיגה שלי אז כשאנשים מתנהגים בפזיזות ומבצעים עבירות תנועה הם מסכנים את החיים של אנשים אחרים בדרך וגם את של עצמם אז אני רוצה למצוא את האחראים להתנהגות הזו ולתת להם עונש מתאים ואני עושה זאת תוך הוצאה של הלוחית רישוי שלהם והכנסה שלו למערכת ואז נותן למשטרה לטפל בשאר. סיבה נוספת שבגללה בחרתי בנושא זה הוא מכיוון שהוא עסק ב-Machin learning והנושא עצמו עניין אותי לגבי איך ניתן להוציא ולזהות מספר מתמונה בעזרת אימון Dataset של מספרים

# **מבוא ורקע כללי**

לכל אדם בעולם יש מספר תעודת זהות שנועדה כדי שיהיה ניתן לזהות אותם במערכת/המדינה בה הם חיים וכשרוצים למצוא או לזהות אדם מסוים מכל סיבה שהיא ניתן להשתמש במספר תעודת הזהות כדי לקבל עליו פרטים. אותו הדבר מיוחס לכל רכב מסביב לעולם. לכל רכב יש מספר לוחית רישוי ייחודית שמזהה את הרכב ויכולה לתת עליו פרטים בנוסף לפרטי האדם שהרכב בבעלותו.

כשאדם מבצע עבירה כל שהיא בין עם זה עבירת תנועה או עבירה פלילית ניתן לדווח עליו לרשויות ולתת להם את לוחית הרישוי של הרכב של מבצע העברה וכך לאתר ולקבל פרטים עליו.

קיום עקב ההתפתחות הטכנולוגית ניתן לראות לעיתים קרובות מצלמות על כבישים שבעצם אם אתה עובר מהירות מסוימת זה לוקח תמונה של הרכב כך שיהיה אפשר לתת דוח לאותו אדם. או פשוט שימוש במצלמות לזיהוי מכוניות חשודות ועוד.

הפרויקט שלי מתייחס לתחום הזה בעצם ונותן לך את האופציה לקחת את התמונה של המכונית שאותה אתה רוצה לזהות. לאחר לקיחת התמונה הפרויקט שלי מנתח את התמונה ומוציא את לוחית הרישוי של הרכב ואת היכולת להוסיף אותו למערך נתונים.

טכנולוגית למידת המכונה((Machin learning הינה תחום במדעי המחשב העוסק בלימוד של המחשב באמצעות איסוף נתונים לתוך מאגר מידע, ניתוחם והסקת מסקנות באמצעות אלגוריתמים מתמטיים וסטטיסטיים. למידת המכונה הינה שוטפת ומתעדכנת באופן דינמי ככל שנעשה בה שימוש רב יותר וזאת באמצעות ניתוח דפוסים חוזרים.

הפרויקט שלי משתמש ב-Machin learning כדי לזהות את מספר הרישוי של הרכב ע"י הניתוח שלהם ממאגר הנתונים שבניתי לזיהוי שלהם.

# **מטרת הפרויקט**

**מה המוצר המוגמר אמור לבצע:**

המוצר נועד לזהות ולהקל על זיהוי של מכוניות עבור הלקוח בכך שהוא לוקח את התמונה של הרכב הרלוונטי ואז נעשה תהליך של זיהוי המספר של הלוחית רישוי מהתמונה ע"י ניתוח התמונה ומאגר הנתונים שיש לזיהוי מספרים.

התוכנה נותן ללקוח לראות בזמן אמת את הווידאו ממצלמה שמצלמת בלייב את הכביש ובעצם כשהלקוח רוצה לזהות מכונית כל מה שעליו לעשות זה לקחת תמונה של המכונית הרצויה לזיהוי. לאחר מכן יש תהליך זיהוי שבעצם מחזיר את מספר לוחית הרישוי ובודק אם הוא נמצא במערכת כאדם שביצע עבירות בעבר הוא לא ואם מקבל תוצאה שלילית נותן ללקוח אפשרות להוסיף את הרכב למאגר הנתונים ובכך לעקוב אחר העבירות שלו.

קהל היעד העיקרי של התוכנה הוא אנשים בתחום האבטחה או בתפקיד ציבורי כמו: עובדי עירייה, משטרה ובטחון.

**דרישות מרכזיות:**

דרישות פונקציונאליות :

1. ללקוח תהיה את היכולת לקחת תמונה ממצלמה שמשדרת בשידור חי ישירות עליו.
2. המערכת תאפשר גישה למאגר הנתונים של רכבים שנמצאים במערכת והיכולת להוסיף או להוציא רכבים מהמאגר.
3. ללקוח יוצג מסך אשר מאשר האם מספר לוחית הרישוי שזוהתה נמצאת במערכת או לא.
4. ללקוח יוצג היסטוריה של מכוניות קודמות שבדקו בהשוואה למספר לוחית הרישוי שזיהינו.

דרישות לא פונקציונאליות:

1. קלות השימוש של המשתמש בתוכנה, התוכנה תהיה ברורה וקלה לתפעול.
2. אמינות המערכת , המערכת תציג תוצאות מדויקות של זיהוי מספרי הרישוי.
3. אסתטיקה, על המערכת להראות יפה ומסודרת.

**תרחישים במערכת :**

תרחישי שימוש:

למשתמש יפתח חלון:

1. live feed from camera – יוצג ווידיאו בשידור חי מהמצלמה
2. Snapshot – כפתור שלוקח תמונה מהמצלמה בשידור חי
3. Result – יוצג במסך זה צבע אדום אם לא נמצא מספר הלוחית במאגר הנתונים וצבע ירוק אם הוא נמצא במערכת.
4. Check history – נותן לך לראות את בדיקות שבוצעו בהשוואה ללוחית הרישוי שזוהתה ללוחיות רישוי שנמצאים במאגר הנתונים.
5. Database – יהיה ניתן לפתוח חלון שנותן לך אפשרות לשנות את מאגר הנתונים כמו להכניס ולהוציא מכוניות מהמאגר.

# **שפת התכנות וסביבת העבודה**

**Python:** שפת התכנות העיקרית בה השתמשתי בפרויקט היא python תוך שימוש

בסביבת הרצה של .visual studio code פייתון זוהי שפת תכנות עילית, מונחית עצמים, המאפשרת עבודה נוחה, בעיקר בשל תמיכתה הנרחבת במודלים וספריות תשתיתיות. כמו כן, היא מאפשרת התממשקות קלה עם קבצים מסוגים שונים ובסיסי נתונים . זאת ועוד זו השפה העיקרית שלמדנו בתיכון ומהחטיבה לכן זה גרם לי להרגיש יותר בנוח להשתמש בה לעומת שפות אחרות.

# **ניסוח וניתוח הבעיה האלגוריתמית**

**בעיה 1 :** הפרויקט מסתמך על אלגוריתמים של לימוד מכונה הדורשים מצד אחד מאגרי

מידע גדולים ומצד שני יכולת לתרגם מודלים מתמטיים מורכבים לקוד. הסיכון הוא

שהישענות על כמות לא מספקת של נתונים וכן מתן מענה חלקי למודלים המתמטיים,

יפגמו ביכולת המוצר לבצע התאמות באופן מדויק והמערכת תעבוד רק באופן חלקי .

# **תיאור אלגוריתמים קיימים**

**פתרונות קיימים לבעיה 1:**

באופן כללי, ישנם 3 סוגים של אלגוריתמים ללימוד מכונה:

1. למידה מונחית: אלגוריתם זה מורכב ממשתנה יעד או תוצאה או (משתנה תלוי) אותו יש לחזות מתוך קבוצה מסוימת של מנבאים (משתנים עצמאיים). באמצעות קבוצה זו של משתנים, אנו מייצרים פונקציה הממפה קלטים לפלטים הרצויים. תהליך הלימוד נמשך עד שהמודל משיג רמת דיוק הרצויה.

דוגמאות ללמידה מונחית: Regression, Decision Tree, Random Forest, KNN CNN וכדומה.

1. למידה בלתי מונחית: באלגוריתם זה אין לנו שום יעד או משתנה תוצאה לחיזוי או

להערכה. אלגוריתם זה משמש לקיבוץ על פי אוכלוסיות בקבוצות שונות. דוגמאות

ללמידה ללא פיקוח: Apriori algorithm, K-means.

1. למידת חיזוק: באמצעות אלגוריתם זה המכונה מאומנת לקבל החלטות ספציפיות.

המכונה נחשפת לסביבה בה היא לומדת את עצמה ללא הרף באמצעות ניסוי

וטעייה. באלגוריתם זה, המכונה לומדת מניסיון העבר ומנסה לתפוס את הידע הטוב

ביותר לצורך קבלת החלטות עסקיות מדויקות. דוגמה ללמידת חיזוק: Markov Decision Process .

# **הפתרון הנבחר**

**פתרונות לבעיה 1:** לאחר שבדקתי וחקרתי באינטרנט על machine learning ומודלים שונים למימושו החלטתי שהדרך הטובה ביותר לפתור את הבעיה הראשונה היא לקיחת מבנה נתונים רחב ומגוון שבעצם עומד בכל הצרכים שלנו על מנת שיהיה ניתן לזהות מספרים שונים. כדי לבצע זיהוי של מספרים בחרתי להשתמש בספריית Tensorflow. בחרתי להשתמש בספרייה זו משום שקבעתי שהיא תהיה שימושית ומועילה בניתוח ופילוח של תמונה מכיוון שהיא עוסקת בתחום ה-machine learning, ויהיה ניתן לזהות אובייקטים בתמונה לאחר השימוש בה. זאת ועוד ספרייה זו טובה בניתוח תמונה ואימון של מודלים בעזרת CNN על מנת שיהיה ניתן לזהות את אותם האובייקטים שנתנו בתור אימון לאותם המודלים.

בנוסף לכך השתמשתי בספריית opencv מכיוון שגם היא כמו הספרייה הקודמת טובה בעיבוד וניתוח של תמונה מה שהיה לעזר רב לאורך בניית הפרויקט.

**האלגוריתם הנבחר**

* להסביר על האלגוריתם הנבחר

?

**מבנה המערכת**

?

# **תיאור המודלים של מערכת התוכנה**

* ProjectServer.py – הצד של השרת בו מתרחש כל הפעולות מאחורי הקלעים
* ProjectClient.py – הצד של הלקוח
* LicensePlateDetection.py - מבצע את הפעולות למציאת מספר לוחית הרישוי
* TrainNumbersDataset.py – יוצר את מבנה הנתונים של זיהוי המספרים ושומר אותו
* projectSnapshotGUI.py – מעלה חלון שנותן לך לראות ווידיאו מהמצלמה שלך ואפשרות לקחת תמונה ולשמור אותה.
* checksIfLPinDataset.py – בודק האם מספר לוחית הרישוי שזוהה נמצא במאגר נתונים

# **תיעוד הקוד**

**פירוט מבניי הנתונים:**

|  |  |
| --- | --- |
| **שם הרשימה** | **תפקיד המערך** |
| My\_checkpoint | מכיל את מבנה הנתונים לזיהוי המספרים |
| licensePlatesNumbers | מכיל רשימה של רכבים ולוחית הרישוי שלהם |

עתה, אפרט על כל פונקציה שנמצאת בקוד. בטבלאות הבאות (טבלה לכל טופס, מחלקה או עצם של המערכת) יצוין שם הפונקציה, מה היא מקבלת, מה היא מחזירה ומה תפקידה.

**ProjectServer.py**

**Class Server**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def \_\_init\_\_ (self,ip,port) | מקבלת ip ו-port | - | מקשיבה לחיבורים מהלקוח |
| def connect\_to\_client(self) | מקבלת את מחלקת הממשק | - | מאפשרת ללקוח להתחבר |
| def send\_message(self,msg) | מקבלת את מחלקת הממשק ומשתנה msg | - | שולח הודעה ללקוח |
| def receive\_message(self) | מקבלת את מחלקת הממשק | מחזיר data | מקבל הודעה מהלקוח |
| def receive\_image(self) | מקבלת את מחלקת הממשק | - | מקבלת תמונה מהלקוח |
| def exit(self) | מקבלת את מחלקת הממשק | - | סוגר את החיבור בין השרת ללקוח |

**ProjectClient.py**

**Class Client**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def \_\_init\_\_ (self,ip,port) | מקבלת ip ו-port | - | מתחברת לשרת |
| def send\_message(self,msg) | מקבלת את מחלקת הממשק ומשתנה msg | - | שולח הודעה לשרת |
| def receive\_message(self) | מקבלת את מחלקת הממשק | מחזיר data | מקבל הודעה מהשרת |
| def send\_image(self) | מקבלת את מחלקת הממשק | - | שולח תמונה לשרת |
| def exit(self) | מקבלת את מחלקת הממשק | - | סוגר את החיבור בין השרת ללקוח |

**LicensePlateDetection.py**

**Class GetLicensePlateNumber**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def \_\_init\_\_ (self,imgPat) | מקבלת imgPath | - | פותח תמונה |
| def img\_processing (self) | מקבלת את מחלקת הממשק | - | מעבדת את התמונה כדי לקבל את הפינות בתמונה |
| def find\_contours (self) | מקבלת את מחלקת הממשק | - | מוצאת את לוחית הרישוי בתמונה וגוזר רק אותה |
| def straighten\_licenseplate (self) | מקבלת את מחלקת הממשק | מחזיר self.result | מיישר את התמונה הגזורה |
| def find\_contours\_of\_numbers(self,dimensions, img) | מקבלת את מחלקת הממשק את המימדים של המספרים ואת התמונה של לוחית הרישוי הישרה | מחזיר img\_result | מוצא את המיקום של כל המספרים מהתמונה ומבודד אותם |
| def segment\_characters(self, image) | מקבלת את מחלקת הממשק ואת התמונה של לוחית הרישוי הישרה | מחזיר  char\_list | הופך את התמונה המיושרת לשחור לבן ולבינארית |
| def load\_saved\_weights(self) | מקבלת את מחלקת הממשק | - | מעלה את המבנה נתונים של זיהוי המספרים |
| def fix\_dimension(self,img) | מקבלת את מחלקת הממשק ואת התמונה של לוחית הרישוי | מחזיר  new\_img | מתקן את הממדים של התמונות של המספרים |
| def show\_results(self,) | מקבלת את מחלקת הממשק | מחזיר Plate\_number | מחזיר את מספר לוחית הרישוי |

**projectSnapshotGUI.py**

**Class MyVideoCapture**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def \_\_init\_\_ (self, video\_source=0) | מקבלת video\_source | - | מקבלת את מקור הווידיאו והממדים שלו |
| def get\_frame(self) | מקבלת את מחלקת הממשק | מחזיר משתנה בוליאני ואת ה-frame | מחזיר frames מהמצלמה |
| def \_\_del\_\_(self) | מקבלת את מחלקת הממשק | - | משחרר את השימוש במצלמה |

**projectSnapshotGUI.py**

**Class App**

|  |  |  |  |
| --- | --- | --- | --- |
| **כותרת הפונקציה** | **טענת כניסה** | **טענת יציאה** | **תפקיד** |
| def \_\_init\_\_ (self, window, window\_title, video\_source=0) | מקבלת video\_source את הכותרת ל-GUI ואת החלונית של ה-GUI | - | פותח ומציג את הווידיאו מהמצלמה בעזרת תמונות |
| def snapshot(self)) | מקבלת את מחלקת הממשק | - | לוקח תמונה מהווידיאו של המצלמה ושומר אותה |
| def update(self) | מקבלת את מחלקת הממשק | - | מקבל תמונות (frames) מהמצלמה |

# **השוואת העבודה עם פתרונות ויישומים קיימים**

כשאני מסתכל על הפרויקט שלי אל מול תוכנות דומות ניתן לראות חסרונות ויתרונות

לפי דעתי היתרון המרכזי בפרויקט שלי זה הפשטות של התוכנה שלי כלפי הלקוח יחד עם שילוב של רמה יחסית גבוהה של דיוק והצלחה. לעומת תוכנות אחרות השתמשתי במבנה נתונים שהוא יותר קטן ואימנתי אותו מעט בהשוואה למבני נתונים של אותם התוכנות אך בכל זאת הצלחתי להוציא 90%+ אחוזי הצלחה באימון של המערכת שהצליח לעבור מספר קטן של תוכנות בתחום זה.

חיסרון שיש בפרויקט שלי זה שיש כ-10% סיכויים של זיהוי מוטעה מכיוון שהבנה נתונים שלי קטן ואימנתי אותו פחות מרוב התוכנות שראיתי שאחוזי ההצלחה שלהם 95%+.

# **הערכת הפתרון לעומת התכנון והמלצות לשיפורו**

אני מרגיש שהצלחתי די הרבה בפרויקט הזה ושהצלחתי לסיים אותו למרות ההתחלה הקשה שהייתה לי. זאת ועוד ניתן לראות לפי אחוזי ההצלחה של 90%+ לזיהוי מספרים על לוחית רישוי שמה שעשיתי בנוסף למאגר הנתונים והאימון של המערכת היו נכונים ומראים את המחקר שערכתי כדי להצליח בפרויקט הזה אם התוספת של למידת מכונה.

הייתי משפר את המערכת – כך שמאגר הנתונים של הפרויקט יהיה יותר גדול ושהאימון של המערכת תהיה יותר גדולה ומדויקת כדי להגדיל כמה שיותר את אחוזי ההצלחה לזיהוי מספרי לוחית הרישוי. זאת ועוד אוטומציה של המערכת כך שלבד תזהה רכב תצלם אותו ותזהה את לוחית הרישוי בלי התערבות של אדם חיצוני תהיה גם לשיפור טוב.

# **תיאור של הממשק למשתמש – הוראות הפעלה**

שלום קניונים בתי עסקים ועירייה נכבדים,

באמצאות התוכנה הזו תוכלו לזהות את המכוניות שעוברות בין עם זה ברחבי העיר או בכניסה או יציאה מחניונים. תוכנה זו מאפשרת לזהות את לוחית הרישוי של אדם ואז בהתאם למה שאתה מוצא לנכון להשתמש בתוכנה זו אתה יכול להוסיף אותו למאגר נתונים.

כל שעליך לעשות זה להריץ את התוכנה על מחשב שמחובר למצלמה בין עם זה אלחוטי או חוטי.

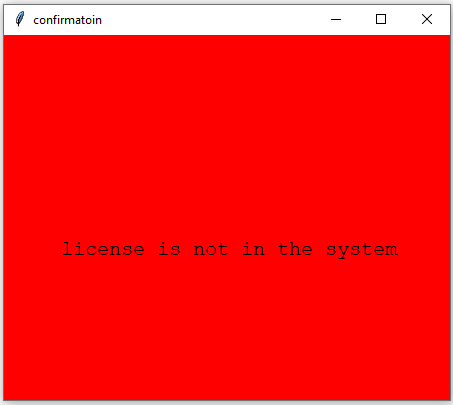
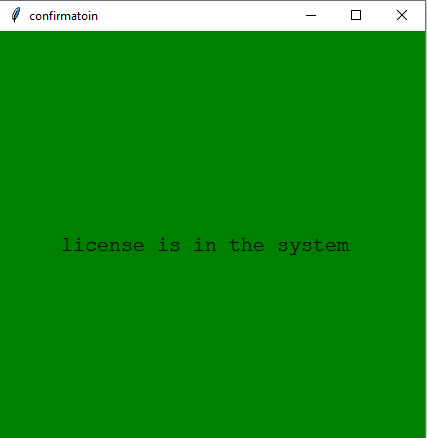
לאחר שאתה מריץ את התוכנה על המחשב יופיע לך חלונית עם ווידיאו בשידור חי מאותה המצלמה.

מתחת לווידיאו יהיה כפתור snapshot שאתה יכול ללחוץ במקרה ואתה רוצה לקחת תמונה של המכונית על מנת לזהות את לוחית הרישוי שלה.

לאחר זיהוי לוחית הרישוי של הרכב יופיע האם מספר לוחית הרישוי של הרכב נמצא במאגר הנתונים או לא בעזרת האינדיקציה של חלונית המופיע בצבע ירוק במקרה והלוחית נמצאת במערכת ואדום במקרה והיא לא.

# **ממשק משתמש**





# **מבט אישי על העבודה ותהליך הפיתוח**

בתחילת העבודה על הפרויקט העבודה הייתה זרה לי, לא ידעתי איך להתחיל וממה להתחיל. לא ידעתי מה לחקור ואיך לשיים את זה. הכל היה בלגן אחד גדול בראש שלי.

אבל כשהתחלתי את הפרויקט התחלתי לחקור ולהיעזר באתרים וקודים של אנשים אחרים בנושא כהבנה בסיסית ושם התחלתי להתקדם לאט בבניית הפרויקט שלי. עבדתי שלב אחרי שלב אם מטרות קטנות שאליהם שאפתי בקוד מה שלבסוף התקבץ לפרויקט שלם.

במהלך הפרויקט יצא לי לעבוד על נושא של למידת מכונה שהיה לנושא מאוד מעניין ושונה ממה שהייתי רגיל לעשות בבית ספר ובאופן כללי וזה הרכיב את הידע שלי על הנושא ולמדתי שנעשה שימוש רב בנושא זה בתעשיית התכנות .

במהלך הפרויקט שלי נעזרתי באתרים רבים ומגוונים על מנת לכתוב את התוכנה ולהבין אותה כשמשהו לא ברור או כשיש שגיאה מסוימת. אתר מרכזי שהייתי הולך אליו כשיש לי שאלה או תקלה בתוכנה היה "stack overflow" שמכיל שאלות ותשובות מאנשים שונים ברחבי העולם לגבי נושאים רבים בתחום התוכנה.

החוויה הייתה מלחיצה מעייפת מאיימת בהתחלה וקשה אך היא הייתה בעיקרה חוויה מעשירה שפתחה בפניי אופקים חדשים, היא נתנה לי הצצה לאיך החיים עובדים ואיך אנשים בתחום הזה עובדים. אני גאה ושמח מהפרויקט שיצאה לי ומהאתגרים הרבים שהבערתי כדי להשלים אותו.

# **ביבליוגרפיה**

<https://stackoverflow.com>

<https://www.geeksforgeeks.org>

<https://github.com>

<https://docs.opencv.org/4.x/d9/df8/tutorial_root.html>

<https://www.w3schools.com>

<https://www.hackster.io/Aasai/license-plate-detection-system-b9297d>

<https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/>

<https://learnopencv.com/edge-detection-using-opencv/>

# **קוד התוכנית**

**projectServer.py**

import os

import socket

from licensePlateDetection import plate\_number

class Server:

    def \_\_init\_\_(self,ip='0.0.0.0',port=12345):

        self.serverSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) #create a socket

        self.serverSocket.bind((ip,port))#bind it with ip and port

        self.serverSocket.listen(1)#listen to any connections

        print("Wait for Connection.....................")

    def connect\_to\_client(self,):

        self.clientSocket,self.clientAddres=self.serverSocket.accept()#accept connection to server

        print(f"connected to {self.clientAddres}")

    def send\_message(self,msg):

        self.clientSocket.send(str(msg).encode())#send encoded masseges to client

    def receive\_message(self,):

        data=self.clientSocket.recv(1024).decode()#receive encoded masseges from client decodes it and return data

        return data

    def receive\_image(self,):

        filePath='vehicle2.jpg'

        with open(filePath,'wb') as f: #open new file to write to

            receivedSize=int(self.receive\_message())#gets image size from client

            currentSize=0

            #receivce image data until receivedSize==currentSize

            while not receivedSize==currentSize:

                if receivedSize-currentSize>1024:

                    data=self.clientSocket.recv(1024)

                    currentSize+=len(data)

                else:

                    data=self.clientSocket.recv(1024)

                    currentSize=receivedSize

                f.write(data)#write data to open file

    def exit(self,):

        self.clientSocket.close()#closes connection

if \_\_name\_\_ == '\_\_main\_\_':

    server=Server()

    while True:

        server.connect\_to\_client()

        server.send\_message("connected to server")

        server.receive\_image()

**projectClient.py**

import socket

from turtle import width

from PIL import Image

from tkinter import \*

#import projectGUI

#import projectServer

#projectServer.server.

class Client:

    def \_\_init\_\_(self,ip='127.0.0.1',port=12345):

        self.clientSocket=socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)#create a socket

        self.clientSocket.connect((ip,port))#connect to server with ip port

    def send\_message(self,msg):

        self.clientSocket.send(str(msg).encode())#send encoded masseges to server

    def receive\_message(self,):

        data=self.clientSocket.recv(1024).decode()#receive encoded masseges from server decodes it and return data

        return data

    def send\_image(self,):

        filePath='vehicle1.jpg'

        img = Image.open(filePath)#open img file

        width, height = img.size#gets image dimensions

        imgSize=width\*height

        self.send\_message(str(imgSize))#sends the img size to client

        with open(filePath,'rb') as f:#open file and reads binary from it

            while True:

                data=f.read(1024)#reads data

                if not data:

                    print('{0} send over...'.format(filePath))

                    break

                self.clientSocket.send(data)#sends data

    def exit(self,):

        self.clientSocket.close()

#pops a window with a color depending if licenseplate in csv green-in| red-not in

def pop\_window(in\_dataset):

    width=500

    height=500

    if(in\_dataset):

        color='green'

        text='license is in the system'

        gui = Tk(className='confirmatoin')#build the gui window

        # set window size

        gui.geometry(f"{width}x{height}")

        #set window color

        gui.configure(bg=color)

        label1 = Label(gui, text=text, fg='black', bg=color,font=("Courier", 15)).place(x=(width/2)-(8\*len(text)),y=height/2-50)

        gui.mainloop()

    else:

        color='red'

        text='license is not in the system'

        gui = Tk(className='confirmatoin')#build the gui window

        # set window size

        gui.geometry(f"{width}x{height}")

        #set window color

        gui.configure(bg=color)

        label1 = Label(gui, text=text, fg='black', bg=color,font=("Courier", 15)).place(x=(width/2)-(7\*len(text)),y=height/2-50)

        gui.mainloop()

pop\_window(False)

if \_\_name\_\_ == '\_\_main\_\_':

    client=Client()

    data=client.receive\_message()

    print(data)

    client.send\_image()

**projectSnapshotGUI.py**

import tkinter

import cv2

import PIL.Image, PIL.ImageTk

import time

class App:

    def \_\_init\_\_(self, window, window\_title, video\_source=0):

        self.window = window

        self.window.title(window\_title)

        self.video\_source = video\_source

        # open video source (by default this will try to open the computer webcam)

        self.vid = MyVideoCapture(self.video\_source)

        # Create a canvas that can fit the above video source size

        self.canvas = tkinter.Canvas(window, width = self.vid.width, height = self.vid.height)

        self.canvas.pack()

        # Button that lets the user take a snapshot

        self.btn\_snapshot=tkinter.Button(window, text="Snapshot", width=50, command=self.snapshot)

        self.btn\_snapshot.pack(anchor=tkinter.CENTER, expand=True)

        # After it is called once, the update method will be automatically called every delay milliseconds

        self.delay = 15

        self.update()

        self.window.mainloop()

    def snapshot(self):

        # Get a frame from the video source

        ret, frame = self.vid.get\_frame()

        if ret:

            print(frame)

            cv2.imwrite("vehicle1" +".jpg", cv2.cvtColor(frame, cv2.COLOR\_RGB2BGR))

    def update(self):

        # Get a frame from the video source

        ret, frame = self.vid.get\_frame()

        if ret:

            self.photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(frame))

            self.canvas.create\_image(0, 0, image = self.photo, anchor = tkinter.NW)

        self.window.after(self.delay, self.update)

class MyVideoCapture:

    def \_\_init\_\_(self, video\_source=0):

        # Open the video source

        self.vid = cv2.VideoCapture(video\_source)

        if not self.vid.isOpened():

            raise ValueError("Unable to open video source", video\_source)

        # Get video source width and height

        self.width = self.vid.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

        self.height = self.vid.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

    def get\_frame(self):

        if self.vid.isOpened:

            ret, frame = self.vid.read()

            if ret:

                # Return a boolean success flag and the current frame converted to BGR

                return (ret, cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB))

            else:

                return (ret, None)

        else:

            return (ret, None)

    # Release the video source when the object is destroyed

    def \_\_del\_\_(self):

        if self.vid.isOpened():

            self.vid.release()

# Create a window and pass it to the Application object

App(tkinter.Tk(), "video camera feed")

**licensePlateDetection.py**

import os

import numpy as np

import pandas as pd

import cv2

import imutils

import matplotlib.pyplot as plt

import matplotlib.gridspec as gridspec

import math

from sklearn.metrics import f1\_score

import tensorflow as tf

import tensorflow.keras.backend as K

from tensorflow.keras import optimizers

from keras.layers import Flatten, Dense, Conv2D, MaxPooling2D, Input, Dropout

from keras.models import Model, Sequential

from keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import Adam

class GetLicensePlateNumber:

    def \_\_init\_\_(self,imgPath='vehicle1.jpg'):

        # Read the image file

        self.image = cv2.imread(imgPath)

        t=cv2.cvtColor(self.image, cv2.COLOR\_BGR2RGB)

        plt.imshow(t)

        plt.title('Original Image')

        plt.show()

        self.image = imutils.resize(self.image, width=500)

        self.img=cv2.cvtColor(self.image, cv2.COLOR\_BGR2RGB)

        # Display the original image

        self.fig, self.ax = plt.subplots(2, 2, figsize=(10,7))

        self.ax[0,0].imshow(cv2.cvtColor(self.image, cv2.COLOR\_BGR2RGB))

        self.ax[0,0].set\_title('Original Image')

    def img\_processing(self,):

        # RGB to Gray scale conversion

        gray = cv2.cvtColor(self.image, cv2.COLOR\_BGR2GRAY)

        self.ax[0,1].imshow(gray, cmap='gray')

        self.ax[0,1].set\_title('Grayscale Conversion')

        blur = cv2.GaussianBlur(gray, (3,3), 0)

        # Noise removal with iterative bilateral filter(removes noise while preserving edges)

        gray = cv2.bilateralFilter(blur, 11, 17, 17)

        self.ax[1,0].imshow(gray, cmap='gray')

        self.ax[1,0].set\_title('Bilateral Filter')

        # Find Edges of the grayscale image

        self.edged = cv2.Canny(gray, 170, 200)

        self.ax[1,1].imshow(self.edged, cmap='gray')

        self.ax[1,1].set\_title('Canny Edges')

        self.fig.tight\_layout()

        plt.show()

    def find\_contours(self,):

        # Find contours based on Edges

        self.cnts = cv2.findContours(self.edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)[0]

        self.cnts=sorted(self.cnts, key = cv2.contourArea, reverse = True)[:30] #sort contours based on their area keeping minimum required area as '30' (anything smaller than this will not be considered)

        self.NumberPlateCnt = None #we currently have no Number plate contour

        # loop over our contours to find the best possible approximate contour of number plate

        count = 0

        for c in self.cnts:

                peri = cv2.arcLength(c, True)

                approx = cv2.approxPolyDP(c, 0.02 \* peri, True)

                if len(approx) == 4:  # Select the contour with 4 corners

                    self.NumberPlateCnt = approx #This is our approx Number Plate Contour

                    x,y,w,h = cv2.boundingRect(c)

                    self.ROI = self.img[y:y+h, x:x+w]

                    break

        if self.NumberPlateCnt is not None:

            # Drawing the selected contour on the original image

            cv2.drawContours(self.image, [self.NumberPlateCnt], -1, (0,255,0), 3)

        plt.imshow(cv2.cvtColor(self.image, cv2.COLOR\_BGR2RGB))

        plt.title("Detected license plate")

        plt.show()

        #The green bounding box shows the detected license plate.

        # Find bounding box and extract ROI

        plt.imshow(self.ROI)

        plt.title("Extracted license plate")

        plt.show()

        #The above displayed coordinates are the coordinates of the detected plate. But the problem is that we don't know which coordinate is where, because contours can start from anywhere and form a continuous path.

    #The idea behind plate rotation is to find the bottom two coordinates. Using these two coordinates, we can easily find the angle of rotation. This will be illustrated shortly.

    # Distance between (x1, y1) and (x2, y2)

    def dist(self,x1, x2, y1, y2):

        return ((x1-x2)\*\*2+(y1-y2)\*\*2)\*\*0.5

    def straighten\_licenseplate(self,):

        """The above function returns the Euclidean distance between any two points (x1, y1) and (x2, y2).

        As discussed, we need to find the bottom two coordinates:

        To find them, we'll first find a coordinate with the maximum y-coordinate and this will be one of the two bottom-most coordinates.

        Now, the other bottom coordinate will be either to the left or right of this coordinate in the array. Since, license plates are rectangular in shape, the second required coordinate would be in a distance far away from the acquired coordinate than the other adjacent coordinate."""

        idx=0

        m=0

        # To find the index of coordinate with maximum y-coordinate

        for i in range(4):

            if self.NumberPlateCnt[i][0][1]>m:

                idx=i

                m=self.NumberPlateCnt[i][0][1]

        # Assign index to the previous coordinate

        if idx==0:

            pin=3

        else:

            pin=idx-1

        # Assign index to the next coordinate

        if idx==3:

            nin=0

        else:

            nin=idx+1

        # Find distances between the acquired coordinate and its previous and next coordinate

        p=self.dist(self.NumberPlateCnt[idx][0][0], self.NumberPlateCnt[pin][0][0], self.NumberPlateCnt[idx][0][1], self.NumberPlateCnt[pin][0][1])

        n=self.dist(self.NumberPlateCnt[idx][0][0], self.NumberPlateCnt[nin][0][0], self.NumberPlateCnt[idx][0][1], self.NumberPlateCnt[nin][0][1])

        # The coordinate that has more self.distance from the acquired coordinate is the required second bottom-most coordinate

        if p>n:

            if self.NumberPlateCnt[pin][0][0]<self.NumberPlateCnt[idx][0][0]:

                left=pin

                right=idx

            else:

                left=idx

                right=pin

            d=p

        else:

            if self.NumberPlateCnt[nin][0][0]<self.NumberPlateCnt[idx][0][0]:

                left=nin

                right=idx

            else:

                left=idx

                right=nin

            d=n

        #Extract the coordinates of the bottom-most coordinates in such a way that ```(left\_x, left\_y)``` denote the bottom-left coordinate and ```(right\_x, right\_y)``` denote the bottom-right coordinate.

        left\_x=self.NumberPlateCnt[left][0][0]

        left\_y=self.NumberPlateCnt[left][0][1]

        right\_x=self.NumberPlateCnt[right][0][0]

        right\_y=self.NumberPlateCnt[right][0][1]

        #For rotating the plate, we need to find the angle of rotation. This can be found out by calculating the sin of theta using the two coordinates. Theta can then be extracted by finding the inverse of sin. Image can finally be rotated by using ```cv2.getRotationMatrix2D()``` function.

        # Finding the angle of rotation by calculating sin of theta

        opp=right\_y-left\_y

        hyp=((left\_x-right\_x)\*\*2+(left\_y-right\_y)\*\*2)\*\*0.5

        sin=opp/hyp

        theta=math.asin(sin)\*57.2958

        # Rotate the image according to the angle of rotation obtained

        image\_center = tuple(np.array(self.ROI.shape[1::-1]) / 2)

        rot\_mat = cv2.getRotationMatrix2D(image\_center, theta, 1.0)

        result = cv2.warpAffine(self.ROI, rot\_mat, self.ROI.shape[1::-1], flags=cv2.INTER\_LINEAR)

        # The image can be cropped after rotation( since rotated image takes much more height)

        if opp>0:

            h=result.shape[0]-opp//2

        else:

            h=result.shape[0]+opp//2

        self.result=result[0:h, :]

        plt.imshow(self.result)

        plt.title("Plate obtained after rotation")

        plt.show()

        return self.result

    ## Character Segmentation

    #Character segmentation is an operation that seeks to decompose an image of a sequence of characters into subimages of individual symbols. It is one of the decision processes in a system for optical character recognition (OCR).

    #This phase contains the use of two functions: ```segment\_characters()``` and ```find\_contours()```.

    # Match contours to license plate or character template

    def find\_contours\_of\_numbers(self,dimensions, img) :

        # Find all contours in the image

        cntrs, \_ = cv2.findContours(img.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

        # Retrieve potential dimensions

        lower\_width = dimensions[0]

        upper\_width = dimensions[1]

        lower\_height = dimensions[2]

        upper\_height = dimensions[3]

        # Check largest 5 or  15 contours for license plate or character respectively

        cntrs = sorted(cntrs, key=cv2.contourArea, reverse=True)[:15]

        ii = cv2.imread('contour.jpg')

        x\_cntr\_list = []

        target\_contours = []

        img\_res = []

        for cntr in cntrs :

            # detects contour in binary image and returns the coordinates of rectangle enclosing it

            intX, intY, intWidth, intHeight = cv2.boundingRect(cntr)

            # checking the dimensions of the contour to filter out the characters by contour's size

            if intWidth > lower\_width and intWidth < upper\_width and intHeight > lower\_height and intHeight < upper\_height :

                x\_cntr\_list.append(intX) #stores the x coordinate of the character's contour, to used later for indexing the contours

                char\_copy = np.zeros((44,24))

                # extracting each character using the enclosing rectangle's coordinates.

                char = img[intY:intY+intHeight, intX:intX+intWidth]

                char = cv2.resize(char, (20, 40))

                cv2.rectangle(ii, (intX,intY), (intWidth+intX, intY+intHeight), (50,21,200), 2)

                plt.imshow(ii, cmap='gray')

                plt.title('Predict Segments')

                # Make result formatted for classification: invert colors

                char = cv2.subtract(255, char)

                # Resize the image to 24x44 with black border

                char\_copy[2:42, 2:22] = char

                char\_copy[0:2, :] = 0

                char\_copy[:, 0:2] = 0

                char\_copy[42:44, :] = 0

                char\_copy[:, 22:24] = 0

                img\_res.append(char\_copy) # List that stores the character's binary image (unsorted)

        # Return characters on ascending order with respect to the x-coordinate (most-left character first)

        plt.show()

        # arbitrary function that stores sorted list of character indeces

        indices = sorted(range(len(x\_cntr\_list)), key=lambda k: x\_cntr\_list[k])

        img\_res\_copy = []

        for idx in indices:

            img\_res\_copy.append(img\_res[idx])# stores character images according to their index

        img\_res = np.array(img\_res\_copy)

        return img\_res

    """In the above function, we will be applying some more image processing to extract the individual characters from the license plate. The steps involved will be:

    Finding all the contours in the input image. The function cv2.findContours returns all the contours it finds in the image.

    After finding all the contours we consider them one by one and calculate the dimension of their respective bounding rectangle. Now consider bounding rectangle is the smallest rectangle possible that contains the contour. All we need to do is do some parameter tuning and filter out the required rectangle containing required characters. For this, we will be performing some dimension comparison by accepting only those rectangle that have:

    1. Width in the range 0, (length of the pic)/(number of characters) and,

    2. Length in a range of (width of the pic)/2, 4\*(width of the pic)/5. After this step, we should have all the characters extracted as binary images."""

    # Find characters in the resulting images

    def segment\_characters(self,image) :

        # Preprocess cropped license plate image

        img\_lp = cv2.resize(image, (333, 75))

        img\_gray\_lp = cv2.cvtColor(img\_lp, cv2.COLOR\_BGR2GRAY)

        \_, img\_binary\_lp = cv2.threshold(img\_gray\_lp, 200, 255, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

        img\_binary\_lp = cv2.erode(img\_binary\_lp, (3,3))

        img\_binary\_lp = cv2.dilate(img\_binary\_lp, (3,3))

        LP\_WIDTH = img\_binary\_lp.shape[0]

        LP\_HEIGHT = img\_binary\_lp.shape[1]

        # Make borders white

        img\_binary\_lp[0:3,:] = 255

        img\_binary\_lp[:,0:3] = 255

        img\_binary\_lp[72:75,:] = 255

        img\_binary\_lp[:,330:333] = 255

        # Estimations of character contours sizes of cropped license plates

        dimensions = [LP\_WIDTH/6,

                        LP\_WIDTH/2,

                        LP\_HEIGHT/10,

                        2\*LP\_HEIGHT/3]

        plt.imshow(img\_binary\_lp, cmap='gray')

        plt.title('Contour')

        plt.show()

        cv2.imwrite('contour.jpg',img\_binary\_lp)

        # Get contours within cropped license plate

        char\_list = self.find\_contours\_of\_numbers(dimensions, img\_binary\_lp)

        return char\_list

    def get\_img\_numbers(self,char):

        self.new\_char=[]

        for x in range(len(char)):

            if(not 255 in char[x][2]):

                self.new\_char.append(char[x])

        for i in range(len(self.new\_char)):

            plt.subplot(1, len(self.new\_char), i+1)

            plt.imshow(self.new\_char[i], cmap='gray')

            plt.axis('off')

        plt.show()

    """The above function takes in the image as input and performs the following operation on it:

    - Resizes it to a dimension such that all characters seem distinct and clear.

    - Convert the colored image to a gray scaled image. We do this to prepare the image for the next process.

    - Now the threshold function converts the grey scaled image to a binary image i.e each pixel will now have a value of 0 or 1 where 0 corresponds to black and 1 corresponds to white. It is done by applying a threshold that has a value between 0 and 255, here the value is 200 which means in the grayscaled image for pixels having a value above 200, in the new binary image that pixel will be given a value of 1. And for pixels having value below 200, in the new binary image that pixel will be given a value of 0.

    - The image is now in binary form and ready for the next process Eroding. Eroding is a simple process used for removing unwanted pixels from the object’s boundary meaning pixels that should have a value of 0 but are having a value of 1.

    - The image is now clean and free of boundary noise, we will now dilate the image to fill up the absent pixels meaning pixels that should have a value of 1 but are having value 0.

    - The next step now is to make the boundaries of the image white. This is to remove any out of the frame pixel in case it is present.

    - Next, we define a list of dimensions that contains 4 values with which we’ll be comparing the character’s dimensions for filtering out the required characters.

    - Through the above processes, we have reduced our image to a processed binary image and we are ready to pass this image for character extraction."""

    def load\_saved\_weights(self):

        """Since the data is all clean and ready, now it’s time do create a Neural Network that will be intelligent enough to recognize the characters after training. In this project, we used CNN model for character recognition.

        - For training the model, we’ll be using ImageDataGenerator class available in keras to generate some more data using image augmentation techniques like width shift, height shift.

        - Width shift: Accepts a float value denoting by what fraction the image will be shifted left and right.

        - Height shift: Accepts a float value denoting by what fraction the image will be shifted up and down.

        For the model, we'll use 4 convolutional layers with a Max pooling layer of window size = (4,4). We'll also use 2 Dense layers where the last dense layers will have 36 output units (26 alphabets + 10 digits) and the activation function used will be 'softmax' because this is a multi-classification problem.

        The below parameters used in the model have already been optimized using hyperparamter tuning.

        We'll now go ahead and test our model. Note that the attribute ```steps\_per\_epoch``` is set to be ```train\_generator.samples // batch\_size``` because it ensures the usage of all the train data for one epoch.

        Since we saved only the model weights, we first need to create a model instance and then load the saved weights into the model."""

        # Create a new model instance

        self.loaded\_model = Sequential()

        self.loaded\_model.add(Conv2D(16, (22,22), input\_shape=(28, 28, 3), activation='relu', padding='same'))

        self.loaded\_model.add(Conv2D(32, (16,16), input\_shape=(28, 28, 3), activation='relu', padding='same'))

        self.loaded\_model.add(Conv2D(64, (8,8), input\_shape=(28, 28, 3), activation='relu', padding='same'))

        self.loaded\_model.add(Conv2D(64, (4,4), input\_shape=(28, 28, 3), activation='relu', padding='same'))

        self.loaded\_model.add(MaxPooling2D(pool\_size=(4, 4)))

        self.loaded\_model.add(Dropout(0.4))

        self.loaded\_model.add(Flatten())

        self.loaded\_model.add(Dense(128, activation='relu'))

        self.loaded\_model.add(Dense(36, activation='softmax'))

        # Restore the weights

        self.loaded\_model.load\_weights('checkpoints/my\_checkpoint')

# Predict the license plate number

#We now have our license plate and the CNN model ready! We just need to predict each character using the model. For this, we'll first fix the dimension of each character image using the function ```fix\_dimension```, in which it converts an image to a 3-channel image. The image can then be sent to ```model.predict\_classes()``` in order to get the predicted character.

# Predicting the output

    def fix\_dimension(self,img):

        new\_img = np.zeros((28,28,3))

        for i in range(3):

            new\_img[:,:,i] = img

            return new\_img

    def show\_results(self,):

        dic = {}

        characters = '0123456789'

        for i,c in enumerate(characters):

            dic[i] = c

        output = []

        for i,ch in enumerate(self.new\_char): #iterating over the characters

            img\_ = cv2.resize(ch, (28,28), interpolation=cv2.INTER\_AREA)

            img = self.fix\_dimension(img\_)

            img = img.reshape(1,28,28,3) #preparing image for the model

            y\_ = np.argmax(self.loaded\_model.predict(img)[0], axis=-1) #predicting the class

            character = dic[y\_]

            output.append(character) #storing the result in a list

        plate\_number = ''.join(output)

        return plate\_number

    def predict\_numbers\_value(self):

        plate\_number=self.show\_results()

        print(self.show\_results())

        # Segmented characters and their predicted value.

        plt.figure(figsize=(10,6))

        for i,ch in enumerate(self.new\_char):

            img = cv2.resize(ch, (28,28), interpolation=cv2.INTER\_AREA)

            plt.subplot(3,4,i+1)

            plt.imshow(img,cmap='gray')

            plt.title(f'predicted: {self.show\_results()[i]}')

            plt.axis('off')

        plt.show()

licenseplateNumber=GetLicensePlateNumber()

licenseplateNumber.img\_processing()

licenseplateNumber.find\_contours()

result = licenseplateNumber.straighten\_licenseplate()

char=licenseplateNumber.segment\_characters(result)

licenseplateNumber.get\_img\_numbers(char)

licenseplateNumber.load\_saved\_weights()

licenseplateNumber.predict\_numbers\_value()

**trainNumbersDataset.py**

import os

import numpy as np

import pandas as pd

import cv2

import imutils

import matplotlib.pyplot as plt

import matplotlib.gridspec as gridspec

import math

from sklearn.metrics import f1\_score

import tensorflow as tf

import tensorflow.keras.backend as K

from tensorflow.keras import optimizers

from keras.layers import Flatten, Dense, Conv2D, MaxPooling2D, Input, Dropout

from keras.models import Model, Sequential

from keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import Adam

train\_datagen = ImageDataGenerator(rescale=1./255, width\_shift\_range=0.1, height\_shift\_range=0.1)

path = 'data'

train\_generator = train\_datagen.flow\_from\_directory(

        path+'/train',  # this is the target directory

        target\_size=(28,28),  # all images will be resized to 28x28

        batch\_size=1,

        class\_mode='sparse')

validation\_generator = train\_datagen.flow\_from\_directory(

        path+'/val',  # this is the target directory

        target\_size=(28,28),  # all images will be resized to 28x28 batch\_size=1,

        class\_mode='sparse')

K.clear\_session()

#build model to train the dataset and saves it

model = Sequential()

model.add(Conv2D(16, (22,22), input\_shape=(28, 28, 3), activation='relu', padding='same'))

model.add(Conv2D(32, (16,16), input\_shape=(28, 28, 3), activation='relu', padding='same'))

model.add(Conv2D(64, (8,8), input\_shape=(28, 28, 3), activation='relu', padding='same'))

model.add(Conv2D(64, (4,4), input\_shape=(28, 28, 3), activation='relu', padding='same'))

model.add(MaxPooling2D(pool\_size=(4, 4)))

model.add(Dropout(0.4))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(36, activation='softmax'))

model.compile(loss='sparse\_categorical\_crossentropy', optimizer=optimizers.Adam(lr=0.0001), metrics='accuracy')

model.summary()

batch\_size = 1

result = model.fit(train\_generator,steps\_per\_epoch = train\_generator.samples // batch\_size,validation\_data = validation\_generator, epochs = 25, verbose=1, callbacks=None)

fig = plt.figure(figsize=(14,5))

grid=gridspec.GridSpec(ncols=2,nrows=1,figure=fig)

fig.add\_subplot(grid[0])

plt.plot(result.history['accuracy'], label='training accuracy')

plt.plot(result.history['val\_accuracy'], label='val accuracy')

plt.title('Accuracy')

plt.xlabel('epochs')

plt.ylabel('accuracy')

plt.legend()

fig.add\_subplot(grid[1])

plt.plot(result.history['loss'], label='training loss')

plt.plot(result.history['val\_loss'], label='val loss')

plt.title('Loss')

plt.xlabel('epochs')

plt.ylabel('loss')

plt.legend()

# Save the weights

model.save\_weights('./checkpoints/my\_checkpoint')

**checksIfLPinDataset.py**

import csv

def check\_if\_licenseplate\_in\_dataset(checkPlateNumber):

    with open('licensePlatesNumbers.csv') as file:#opens a csv file

        csvreader = csv.reader(file)#reads from it

        header=[]

        header=next(csvreader)#gets headers

        print(header)

        licenseplates = []

        for row in csvreader:#gets all the cars license plates number

            licenseplates.append(row)

        print(licenseplates)

    for plateNumber in licenseplates:#check if the license plate checkPlateNumber is in the csv

        number=int(plateNumber[0])

        if(number==checkPlateNumber):#in-true , not in-false

            return True

    return False

isIn=check\_if\_licenseplate\_in\_dataset(2952165)

print(isIn)