מסמך פרויקט *ImageToText* -
תשפ"א

מגיש: עומרי חליפה
תעודת זהות: 213765787
מורה מנחה: גלית ברעם
תאריך הגשה: קיץ 2021

תוכן עניינים :

פרק 1 – יזום הפרויקט.....עמודים 4-6

- תיאור כללי
- מטרת הפרויקט
- ליבה טכנולוגית
- טכנולוגיות עיקריות ושיקולים עיקריים
- אתגרים טכנולוגיים
- סוגי משתמשים / קהל היעד
- דרישות חומרה
- פתרונות קיימים

פרק 2 – אפיון הפרויקט.....עמודים 6-11

- תפקוד – עיבוד/ קלט/ פלט
- סביבת עבודה
- ממשק משתמש
- תקשורת נתונים
- אחסון הפרויקט
- מודלים
- פיצ׳רים ותהליכים עיקריים
- טכנולוגיות – תהליכים
- מבנה בסיס נתונים

פרק 3 – ארכיטקטורת הפרויקט.....עמודים 12-13

- מבט על
- Client application
- Server
- Learning machine
- עיצוב הנתונים וישויות המידע
- טכנולוגיות עיקריות
- התאמה לאפיון

פרק 4 – תוכנית עבודה.....עמודים 13-19

- מתן עדיפות לפיצ׳רים, חלוקה לשלבים מסודרים
- חלוקה לאיטרציות (איטרציה 1,2,3,4,5)

פרק 5 – עיצוב – מדי איטרציה.....עמודים 19-23

- איטרציה 1 (מטרה, תהליך עבודה, תוצר)
- איטרציה 2 (מטרה, תהליך עבודה, תוצר)
- איטרציה 3(מטרה, תהליך עבודה, תוצר)
- איטרציה 4 (מטרה, תהליך עבודה, תוצר)
- איטרציה 5 (מטרה, תהליך עבודה, תוצר)

פרק 6 – קוד הפרויקט.....עמודים 24-25
- הסבר קבצים
- פונקציות בImageToTextBeckend – Server side react.py
- פונקציות בCore_ProjectImageToText.py
- פונקציות בimageProcessing.py
- פונקציות בCnn.py
- פונקציות בFeatures_ProjectImageToText.py
- פונקציות בApp.js

פרק 7 – שימוש בפרויקט.....עמודים 25-30
פרק 8 – קבצים בפרויקט.....עמודים 30-39
רפלקציה.....עמוד 39
ביבליוגרפיה .....עמוד 40

פרק 1 : יזום הפרויקט

תיאור כללי

הפרויקט שלי הוא אפליקציית סריקת תמונה לטקסט (OCR) שמאפשרת למשתמש לעלות תמונה עם טקסט באנגלית לאפלקציה, משם התמונה נשלחת אל השרת ובצד השרת ממירה את הטקסט בתמונה לטקסט, ויוצרת בנתיב שנבחר על ידי המשתמש תיקייה שכוללת בתוכה :

אם מדובר בזיהוי תו אחד בודד :

1. קובץ txt עם סיווג התו.
2. קובץ txt שמכיל בתוכו את כל סיווגי התווים האחרונים של המשתמש.

אם מדובר בזיהוי טקסט בתמונה :

1. קובץ txt עם סיווג הטקסט שנמצא בתמונה.
2. קובץ txt שמכיל בתוכו את התרגום של הטקסט מאנגלית לעברית.
3. קובץ pdf שמכיל בתוכו את סיווג הטקסט בתמונה.
4. קובץ word שמכיל בתוכו את סיווג הטקסט בתמונה.
5. תמונה שמראה את רקע התמונה.

מטרת הפרויקט

הפרויקט עונה על הצורך להמיר קבצי תמונה אל קבצי טקסט, לממש את טכנולוגיית OCR.
בחרתי לעבוד על פרויקט זה משום שזהו נושא המסקרן אותי מאוד והידע שלי בנושא זה מאוד דל כך שזו הזדמנות טובה להרחיב את הידע. בחרתי ברעיון הזה כי אני חושב שהפרויקט הוא יעיל ושימושי ליום יום, בין אם זה בבית ספר ובין אם זה לדברים אחרים. לדעתי הפרויקט מכיל בתוכו תחומי עניין מגוונים וחשובים ברמה הטכנולוגית, והם נושאים שמעניינים אותי ואני רוצה להתעמק בהם. המוטיבציה שמניעה אותי לעשות את הפרויקט הזה, הוא הרצון ללמוד מעבר לתכנית הלימודים בתחום בינה מלאכותית ועיבוד תמונה.

ליבה טכנולוגית

ליבת הפרויקט של הפרויקט שלי הוא היכולת לסווג אות מתמונה, כלומר לקחת תמונה ולדעת באיזה אות מדובר ולסווג אותה.

טכנולוגיות עיקריות ושיקולים עיקריים

הטכנולוגיות המרכזיות בהם מימשתי את הליבה הטכנולוגית של הפרויקט הם :

1. זיהוי אות בתמונה –מכונה לומדת אשר מזהה אות בתמונה לפי תבניות של אותיות מוכנות במאגר המידע שלי , ומבינה כי מדובר באות.

2. סיווג האות שזוהתה – מכונה לומדת אשר לומדת את הבסיס של כל אות, יוצרת תבניות ובכך היא יכולה לסווג את סוג האות באמצעות מאגר המידע של האותיות , כלומר לומר מהי האות שצולמה.

אתגרים טכנולוגיים

האתגרים הטכנולוגיים שהיו לי בתחום החקר הם חוסר מידע לגבי חומר מסוים או חומר יותר מדי מורחב כך שהייתי צריך לצמצם חומר מיותר בשביל שיישאר רק מה שרלוונטי לי. את האתגרים האלו פתרתי בעזרת מחקר.
בנוסף לכל קיימים אתגרים במימוש שאני נתקלתי בהם כמו קוד שלא עובד כמו שצריך, ופתרתי באמצעות debug.
גם למידת המכונה שלמדתי עליה עיכבה אותי.
האתגר העיקרי שלי שהיה לי בפרויקט הוא הלמידה העצמאית של הטכנולוגיות המורכבות, למידת המכונה ועיבוד התמונה, כיוון שאני חסר ידע מקדים בנושא.
החומרים שמצאתי באינטרנט הם ברובם תיעודיים, או מאגרי מושגים והסברים טכניים על חלקים כאלו ואחרים בטכנולוגיות, אך חסר הסבר אחיד, מעיין מדריך, על הטכנולוגיות שממנו ניתן ללמוד כיצד ניתן לבצע שימוש בטכנולוגיה ולהתממשק עם מה, המידע הנ״ל קיים אבל הוא מפוזר בין הרבה דפים ולא מסודר.
בשל כך אני עברתי תהליך למידה ארוך שסידרתי את החומרים תחילה לשם נוחיות על מנת שאוכל ללמוד באופן רציף, ולאחר מכן מיקדתי את החומרים לחומרים הספציפיים שהייתי צריך בפרויקט.

סוגי משתמשים / קהל היעד

סוגי המשתמשים אליהם מיועד הפרויקט הם כל מי שמעוניין ב OCR ,כלומר בזיהוי טקסט מתמונה, הפרויקט הנ״ל ישמש בעיקר סטודנטים ותלמידים ויהיה יעיל מאוד עבורם ושימושי מאוד מכיוון שהם יוכלו להמיר את מחברותיהם לקובץ וירטואלי כדי שיהיה להם יותר מסודר ומובן, להמיר דפים שהם מקבלים מהמורה לקבצים כך שהכל יהיה מסודר להם בצורה מקוונת בטלפון/מחשב.

דרישות חומרה

הפרויקט לא דרש חומרה מיוחדת מעבר למחשב להריץ את הבדיקות.

פתרונות קיימים

קיימים אין-ספור פתרונות כאשר מדובר ב-OCR אך כל אחד מהם ממומש בצורה שונה . מספר פתרונות קיימים הם :

- פרויקט **ocr** - פרויקט אשר משתמש באותן ספריות שאנו רוצים להשתמש בהן. שונה ברמת הפיצרים שיש בו לעומת הפרויקט שלנו .
- דוגמה לocr- שימוש בocr.

פרק 2 : אפיון הפרויקט

תפקוד – עיבוד / קלט / פלט

קלט:
בצד הלקוח, המשתמש מעלה תמונה בצד הלקוח בנוסף כותב את הנתיב שבו הוא רוצה שסיווג התמונה יתבצע ושולח אל השרת.
עיבוד:
השרת מקבל את התמונה מצד הלקוח, מפצל את התמונה לשורות, מילים ואותיות ומכניס אל קובץ האותיות. משם של האותיות של המילה למודל, המודל מסווג את האותיות, מחבר את האותיות csvהשרת שולח את קובץ ה למחרוזת, ומחזיר לשרת.
פלט:
השרת מקבל את המחרוזת מהמודל, ויוצר בנתיב שהמשתמש יצר :
אם מדובר בזיהוי תו אחד בודד :

1. קובץ txt עם סיווג התו.
2. קובץ txt שמכיל בתוכו את כל סיווגי התווים האחרונים של המשתמש.

אם מדובר בזיהוי טקסט בתמונה :

1. קובץ txt עם סיווג הטקסט שנמצא בתמונה.
2. קובץ txt שמכיל בתוכו את התרגום של הטקסט מאנגלית לעברית.
3. קובץ pdf שמכיל בתוכו את סיווג הטקסט בתמונה.
4. קובץ word שמכיל בתוכו את סיווג הטקסט בתמונה.
5. תמונה שמראה את רקע התמונה.

סביבת עבודה

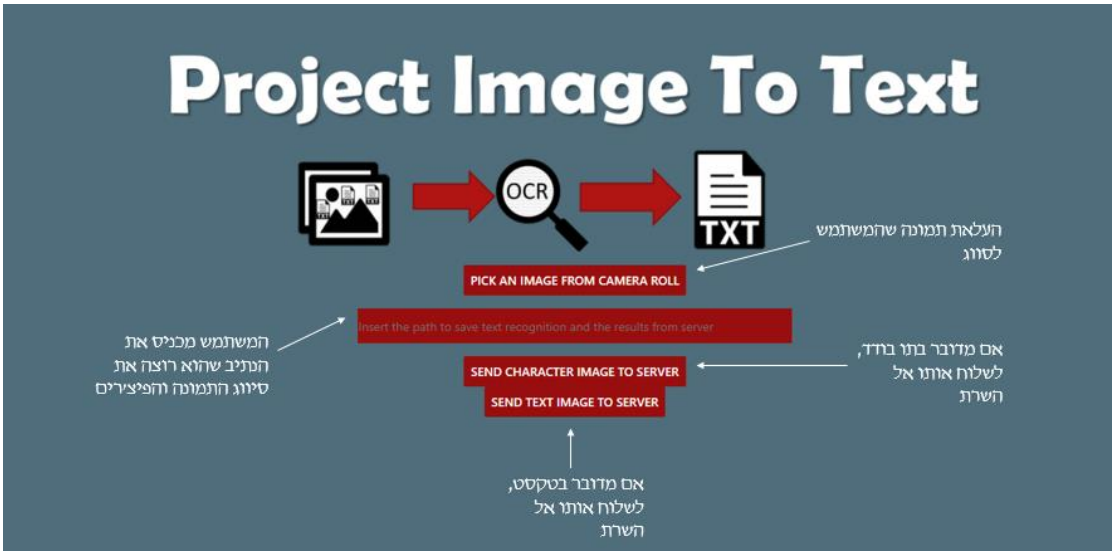
סביבת העבודה שלי בפרויקט הוא בעיקר פייתון.

את כתיבת צד השרת כתבתי בפייתון.

את כתיבת צד הלקוח כתבתי בreact native בjava script.

את מודל המכונה כתבתי בפייתון.

ממשק משתמש



תקשורת נתונים

השרת והלקוח מדברים באמצעות בקשת post:

POST היא אחת ממתודות בקשה (request) משרת הנתמכת על ידי הפרוטוקול HTTP. הלקוח שולח את התמונה, אם זה משפט או תו לפי כפתור השליחה, ואת הנתיב שבחר, ההודעה נראת כך :

{image": "data: GscaqiKAqqowAB0AFOoooAKKKKACiiigAooooAKKKKAOZ8Z/DHwd8RoUh8W
eE9D8UQoMLH /9k=","choice": "1","path": "C: \\filesMagshimim"}

השרת מקבל את ההודעה ומנתח אותה בהתאם.

אחסון הפרויקט

אחסון הפרויקט הוא על המחשב, כאשר את מודל המכונה בניתי בענן של גוגל – google colab

מודלים

מודל שרת לקוח :

הלקוח הוא מי שיוזם את התקשורת כי הוא מעוניין לקבל מהשרת את הסיווג בצד הלקוח .

השרת הוא מי שהסיווג נמצא אצלו , והוא משרת את הלקוחות לפי בקשה, אם מדובר בזיהוי של תו בודד או שמדובר בסיווג טקסט עם הפיצ׳רים.

מודל CNN – מודל המכונה :

המודל מבוסס על רשת נוירונית שמייצרת בינה מלאכותית. בינה פירושה שמתכנתים לא צריכים לקודד את כל המצבים האפשריים לתוכנה, ובמקום זאת ניתן לכתוב תוכנית שתגרום למחשב להבין דברים בעצמו.

המוח שלנו מורכב מנוירונים אבל במודל הממוחשב שאותו בניתי בפרויקט נוירון הוא מקום בזיכרון המחשב שמחזיק מספר.הרשת מסודרת מכמה שכבות של נוירונים. השכבה הראשונה קולטת את התמונה, והאחרונה פולטת את התוצאה. בין שתי השכבות קיימת שכבה חבויה אחת או יותר שבהם מתבצעת הלמידה.

המודל מייצר רשת CNN, Convolutional Neural Network, שהיא השימושית ביותר לזיהוי של תמונות בגלל שהיא מסוגלת להתחשב בממד הרוחב והגובה של הפריטים מהם מורכבת התמונה. הרשת שבה נהוג להשתמש לצורך קונבולציה מתאפיינת במבנה מתכנס, שמאפשר לנוירון להרכיב תמונה כוללנית יותר מהנוירונים בשכבה הקודמת כיוון שהוא ניזון ממספר נוירונים בשכבה שלפניו. הודות למבנה המתכנס השכבה האחרונה של הרשת רואה את התמונה הכללית, ומסיקה מה רואים בתמונה.

פיצ׳רים ותהליכים עיקריים

---

1. צילום תמונה באמצעות האפליקציה – המערכת מצלמת מהמשתמש תמונה שמכילה טקסט עבורה המערכת ממירה את הטקסט בתמונה לטקסט המוקלד בה. כתבתי web application בreact native וקלטתי את התמונה מהמשתמש. השלב ייחל בעת פתיחת האפלקציה .

2. קליטת התמונה מהמשתמש – המערכת קולטת מהמשתמש תמונה שמכילה טקסט עבורה המערכת ממירה את הטקסט בתמונה לטקסט המוקלד בה.

3. שליחת התמונה לשרת – לאחר שהמשתמש ערך את התמונה ובחר את החלק אותו הוא רוצה להמיר, הוא שולח אותה לשרת שמכיל את המכונה. השתמשתי בפייתון באמצעות socket. (בקשת **post**)

4. דגימת התמונה כדי למצוא איזה גוון מבדיל בין הטקסט לרקע התמונה – לאחר שהמשתמש שולח את התמונה למכונה, המכונה למעשה מבדילה בין הטקסט לבין רקע התמונה, שלב זה מבוצע בשביל שלב זיהוי האותיות בהמשך. השתמשתי בטכנולוגיית עיבוד התמונה לצורך הפיצר.

5. הפרדת השורות השלמות למילים על פי מרווחים – המכונה מפרידה את השורות למילים לצורך זיהוי האותיות בהמשך. השתמשתי בטכנולוגיית עיבוד התמונה לצורך הפיצר.

6. הפרדת השורות השלמות לאותיות על פי המילים – המכונה מפרידה את המילים לאותיות לצורך זיהוי האותיות בהמשך. השתמשתי בטכנולוגיית עיבוד התמונה לצורך הפיצר.

7. חיפוש בתוך מאגר המידע של האותיות – לאחר שהפיצ׳רים של עיבוד התמונה לפני למידת המכונה מוכנים , נבנה מכונה שמחפשת במאגר המידע באיזה אות מדובר. השתמשתי במכונה שנבנתה לזיהוי האותיות בתמונה והשתמשתי בלימוד במאגר המידע EMNIST.

12. זיהוי האותיות בתמונה שהתקבלה – לאחר שהפיצרים של עיבוד התמונה לפני למידת המכונה מוכנים , המכונה לומדת תבניות של אותיות בכך שעוברת על עשרות אלפי דוגמאות של אותיות, בונה לעצמה תבניות של אותיות ובכך מזהה את התו ומעביר את התו לשרת. השתמשתי במכונה שנבנה לזיהוי האותיות בתמונה והשתמשתי במאגר המידע EMNIST

13. סיווג האותיות בתמונה שהתקבלה – לאחר חיפוש במאגר והבנה באיזה אות מדובר, היא מסווגת את האות (אומרת באיזה אות או תו מדובר). השתמשתי במכונה שנבנה לזיהוי וסיווג האותיות בתמונה והשתמשתי במאגר המידע EMNIST

14. זיהוי מקום של האות בתוך התמונה – לאחר שהמכונה לומדת את התבניות ומסווגת את האותיות המכונה תסדר את האות שסיווגנו לפי הסדר שהם היו בתמונה. השתמשתי במכונה שנבנה לזיהוי וסיווג האותיות בתמונה והשתמשתי במאגר המידע EMNIST , בנוסף השתמשתי בpython.

15. הפיכת הסיווג לפונט קריא – לאחר שהמכונה לומדת את התבניות , מסווגת את האותיות , מסדרת את המקום בתמונה , האותיות שסווגו הופכות לפונט קריא שניתן להבין. השתמשתי בפייתון והעברתי את הטקסט אל תוך קובץ word.

16. קבלת לינק לקריאה בפונט קריא – לאחר סידור וסיווג הטקסט שהיה בתמונה והפיכתה לפונט קריא, המערכת מעבירה את הטקסט שהומר לקובץ txt או word.

17. תרגום הטקסט שזוהה בתמונה – לאחר שהקובץ מוכן וקריא , המשתמש יוכל לבחור באופציית תרגום הטקסט, הטקסט שהיה בתמונה בשפה האנגלית יתורגם לשפה העברית . השתמשתי בפייתון בספרייה של גוגל טרנסלייט ושלחתי את הטקסט אל השרת של google translte.

טכנולוגיות – תהליכים

פיצ'ר / תהליך :	טכנולוגיות ושפות תכנות :	משאבים נדרשים ושירותים חיצוניים :
קליטת התמונה מהמשתמש	כתבתי web application בreact native, וצד שרת בפייתון.	לא נדרשים כרגע.
שליחת התמונה לשרת	השתמשתי בפייתון בsocket, בקשת post	לא נדרשים כרגע.
דגימת התמונה כדי למצוא איזה גוון מבדיל בין הטקסט לרקע התמונה	באמצעות ספריית opencv, השתמשתי בפונקציה cnvtColor והפכתי את התמונה לצבע שחור-אפור-לבן	לא נדרשים כרגע.

הפרדת השורות השלמות לאותיות ומילים על פי מרווחים	באמצעות ספריית pytesseract בפייתון.	לא נדרשים כרגע.
זיהוי האותיות בתמונה שהתקבלה	השתמשתי במודל שנקרא CNN שהוא השימושי ביותר לזיהוי תמונות, כתבתי את המודל באמצעות ספרייה שנקראת 2TensorFlow ,ספרייה של למידת מכונה שכתובה ב-Python	השתמשתי בפייתון בסביבת google collab.
חיפוש בתוך מאגר המידע של האותיות	המכונה שנבנה לזיהוי האותיות בתמונה צריכה לחפש במאגר המידע EMNIST את האות אותה זיהתה.	השתמשתי בפייתון בסביבת google collab.
סיווג האותיות בתמונה שהתקבלה.	לאחר שהאות שזוהתה נמצאה במאגר המידע EMNIST , המכונה מסווגת אותה לפי האות שנמצאה במאגר.	השתמשתי בפייתון בסביבת google collab.
זיהוי מקום של האות בתוך התמונה	השתמשתי בפייתון בספרייה.	לא נדרשים כרגע.
קבלת לינק לקריאה בפונט קריא	השתמשתי בפייתון והעברתי את הטקסט אל תוך קובץ word	לא נדרשים כרגע.
תרגום הטקסט שזוהה בתמונה	השתמשתי בפייתון בספרייה של גוגל טרנסלייט ושלחתי את הטקסט אל השרת של google translte.	לא נדרשים כרגע.
העברת קובץ הטקסט ל PDF	השתמשתי בספרייה המעבירה את קובץ הtxt אל קובץ pdf	לא נדרשים כרגע.

מבנה בסיס נתונים

ה***Datasets*** שלי הם: *MNIST* ו- *KAGGLE*

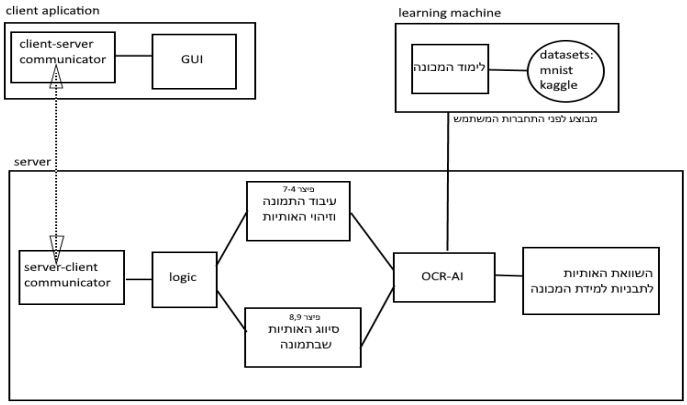
מבנה נתונים שמאפשר מיוחד שנבנה בצורה ספציפית בשביל המטרה והוא בנוי לכך שבזכותו המכונה הלומדת תעבור על מאגר גדול יותר מאשר אנו מסוגלים לבצע ובכך להביא תוצאות טובות יותר עבור המכונה, השתמשתי במאגרים אלו משום שהן המאגרים של אותיות ותווים והם הנפוצים ביותר והשימושיים ביותר.

:Emnist Dataset

* בכל שורה יש בתו הראשון את סיווג האות ובהמשך תמונה של האות

פרק 3 – ארכיטקטורה:

מבט על:



- **:Client application**
  - רכיב האחראי על הצגת המידע למשתמש בצורה נוחה וקלה לשימוש.
  - Client-server communicator – אחראי על התקשורת עם השרת.
  - Application GUI – פלטפורמת האפליקציה שדרכה המשתמש ניגש למערכת. רכיב זה משמש למעשה כשכבת האפליקציה ונכתב בפייתון.
- **:Server**
  - Server-client communicator – אחראי על ביצוע התקשורת מול הלקוח, כולל טיפול בתקלות תקשורת.
  - רכיב זה נכתב בpython.
  - Logic רכיב לוגיקה – אחראי על עיבוד הנתונים והבקשות המתקבלים בתקשורת, אימות ושליפת המידע, והעברת הבקשות למענה המתאים.
  - סיווג האותיות שבתמונה – אם הבקשה מתאימה לו, ישלח למערכת הבינה המלאכותית ויסווג את האות שבתמונה.

- עיבוד התמונה וזיהוי האותיות, אם הבקשה מתאימה לו, יעבד את התמונה ויערוך אותה בהתאם לבקשת הלקוח , ויזהה את האותיות בתמונה באמצעות הבינה המלאכותית.
- OCR AI- אחראי על זיהוי וסיווג האותיות בתמונה ועובדת עם learning machine (ראה המשך learning machine).
- השוואת האותיות לתבניות למידת המכונה – לאחר שהמכונה לומדת באמצעות המאגר מידע (ראה המשך learning machine).

- **Learning machine :**

- תהליך שמתבצע בהרצה הראשונית של השרת, שכביכול מלמדת את המכונה את התבניות של האותיות לפי הEMNIST ופועל באופן ישיר עם מערכת הבינה המלאכותית OCRAI ולאחר מכן הוא נגמר.

עיצוב הנתונים ויישויות המידע :

- בקשות ומענות בין השרת ללקוח :
  - בקשת פתיחת התקשורת.
  - בקשת שליחת התמונה.
  - מענה על שליחת התמונה וסיווגה.
  - סיווג התקשורת.
- המידע המועבר בין GUI ללקוח הוא בהקשר לחיצת הכפתורים.
- בקשות ומענות בין הlogic ל OCR AI :
  - עיבוד התמונה ועריכתה בהתאם לבקשת הלקוח.
  - זיהוי האותיות מהתמונה וסיווגן.
- לימוד המכונה לבין המכונה עצמה באמצעות מאגרי המידע .

טכנולוגיות עיקריות :

- לצורך התקשורת בין השרת ללקוח השתמשתי בsocket בשפת פייתון.
- לצורך עיבוד התמונה השתמשתי בספריית pytesseract ו opencv בשפת פייתון.
- לצורך לימוד המכונה השתמשתי בtensorflow2 ובמאגרי המידע EMNIST.
- לצורך האפליקציה השתמשתי בספריית React native

** השתמשי בשפת פייתון מכיוון שאני מאוד אוהב אותה והיא מאוד נוחה ושימושית , הספריות בהן השתמשתי נמצאות בה.

התאמה לאפיון :

פיצ'ר / תהליך:	רכיבים רלוונטיים
קליטת התמונה מהמשתמש	לקוח (APP GUI) -> client-server communicator -> server-client communicator
שליחת התמונה לשרת	לקוח (APP GUI) -> client-server communicator -> server-client communicator
דגימת התמונה כדי למצוא איזה גוון מבדיל בין הטקסט לרקע התמונה	server-client communicator -> Logic -> עיבוד התמונה -> ocr AI
זיהוי שורות הטקסט ויישור השורה	server-client communicator -> Logic -> עיבוד התמונה -> ocr AI
הפרדת השורות השלמות לאותיות ומילים על פי מרווחים	server-client communicator -> Logic -> עיבוד התמונה -> ocr AI
זיהוי האותיות בתמונה שהתקבלה	server-client communicator -> Logic -> זיהוי האותיות -> ocr AI
חיפוש בתוך מאגר המידע של האותיות	ocr AI <-Learning machine
סיווג האותיות בתמונה שהתקבלה.	communicator -> Logic -> סיווג האותיות -> ocr AI

זיהוי מקום של האות בתוך התמונה	< -Logic <-communicator עיבוד התמונה -> ocr AI
קבלת לינק לקריאה בפונט קריא	server- < - logic <-Ocr AI gui – client-server <-client app
תרגום הטקסט שזוהה בתמונה	server- < - logic <-Ocr AI gui – client-server <-client app
העברת קובץ הטקסט ל PDF	server- < - logic <-Ocr AI gui – client-server <-client app

פרק 4 : תוכנית עבודה

מתן עדיפות לפיצ׳רים, חלוקה לשלבים מסודרים

מס׳ פיצר	משימה	תלויות תשתית (האם המשימה תלויה במשימה אחרת)	נימוק והערות	רכיב רלוונטי
1	קבלת תמונה לשרת	לא תלוי במשימה אחרת	השרת מקבל תמונה.	Client-server communicator
2	בדיקה וזיהוי הצבע שחוזר על עצמו הכי הרבה בתמונה	תלוי בתמונה שהתקבלה	השרת בודק מהו הצבע הכי נפוץ בתמונה.	logic
3	שליחת התמונה לשרת	לא תלוי במשימה אחרת	לאחר שהמשתמש ערך את התמונה ובחר את החלק אותו הוא רוצה להמיר, הוא שולח אותה לשרת שמכיל את המכונה.	Client-server communicator

1	דגימת התמונה כדי למצוא איזה גוון מבדיל בין הטקסט לרקע התמונה	תלוי במשימות הקודמות	לאחר שהמשתמש שולח את התמונה למכונה, המכונה למעשה מבדילה בין הטקסט לבין רקע התמונה, שלב זה מבוצע בשביל שלב זיהוי האותיות בהמשך.	logic
3	הפרדת השורות השלמות למילים על פי מרווחים	תלוי במשימות הקודמות	המכונה מפרידה את השורות למילים לצורך זיהוי האותיות בהמשך.	logic
4	הפרדת השורות השלמות לאותיות על פי המילים	תלוי במשימות הקודמות	המכונה מפרידה את המילים לאותיות לצורך זיהוי האותיות בהמשך.	logic
1	חיפוש בתוך מאגר המידע של האותיות	תלוי במשימות הקודמות	המכונה מחפשת במאגר המידע באיזה אות מדובר.	OCR AI
2	זיהוי האותיות בתמונה שהתקבלה	תלוי במשימות הקודמות	המכונה לומדת תבניות של אותיות בכך שעוברת על עשרות אלפי דוגמאות של אותיות, בונה לעצמה תבניות של אותיות ובכך מזהה את התו ומעביר את התו לשרת.	OCR AI
3	סיווג האותיות בתמונה שהתקבלה	תלוי במשימות הקודמות	לאחר חיפוש במאגר והבנה באיזה אות מדובר, היא מסווגת את האות (אומרת באיזה אות או תו מדובר).	OCR AI
4	זיהוי מקום של האות בתוך התמונה	תלוי במשימות הקודמות	מסדרים את האות שסיווגנו לפי הסדר שהם היו בתמונה.	OCR AI

5	הפיכת הסיווג לפונט קריא	תלוי במשימות הקודמות	האותיות שסווגו הופכות לפונט קריא שניתן להבין.	OCR AI
6	קבלת לינק לקריאה בפונט קריא	תלוי במשימות הקודמות	לאחר סידור וסיווג הטקסט שהיה בתמונה, המערכת מעבירה את הטקסט שהומר לקובץ txt או word.	logic
1	בניית האפליקציה עם GUI בסיסי.	תלוי במשימות הקודמות		Client application
2	תמיכה ותקשורת בין השרת ללקוח	תלוי במשימות הקודמות	ובדיקת צד השרת לאחר קליטת התמונה מהמשתמש (תשובה מהשרת וקבלת הטקסט).	Client-server communicator
1	תרגום הטקסט שזוהה בתמונה	תלוי במשימות הקודמות	אם המשתמש בחר בתרגום הטקסט, הטקסט שהיה בתמונה בשפה האנגלית יתורגם לשפה העברית	Server-client communicator
3	העברת קובץ הטקסט ל PDF-	תלוי במשימות הקודמות	אם המשתמש בחר בהעברת קובץ הטקסט לpdf, הטקסט שהומר ועבר לקובץ txt או word מועבר לקובץ pdf.	Server-client communicator

חלוקה לאיטרציות (ספרינטים)

איטרציה 1:

מטרה: בניית אב טיפוס של שרת שמקבל תמונה ופועל לפי עיבוד תמונה, הוא מאפשר לערוך את התמונה, הוא בודק ומזהה את הצבע שחזר על עצמו הכי הרבה בתמונה, ומציג את התמונה לאחר העריכה ואת הצבע שחזר על עצמו הכי הרבה בתמונה.

פיצ'רים באיטרציה 1:

1. קבלת תמונה לשרת – השרת מקבל תמונה.

2. עריכת התמונה: חיתוך התמונה - המשתמש יוכל לחתוך את התמונה, לבחור את הערך שהוא רוצה שיומר על ידי המכונה

3. עריכת התמונה: סיבוב התמונה - המשתמש יוכל לסובב את התמונה

4. עריכת התמונה : הבהרת התמונה - המשתמש יוכל להבהיר את התמונה במידת הצורך.

5. בדיקה וזיהוי הצבע שחוזר על עצמו הכי הרבה בתמונה – השרת בודק מהו הצבע הכי נפוץ בתמונה.

6. שליחת התמונה לשרת – לאחר שהמשתמש ערך את התמונה ובחר את החלק אותו הוא רוצה להמיר, הוא שולח אותה לשרת שמכיל את המכונה.

איטרציה 2:

מטרה: בניית אב טיפוס לאפליקציה מצד הקליינט עם GUI בסיסי שתומכת ויכולה לדבר עם השרת שכתבנו, להעביר לו תמונה (שניתן לערוך באפליקציה באמצעות השרת) , ולקבל ממנה את התשובה. כמו כן, לחקור על מודל CNN וכיצד יש לכתוב את המודל לצרוך למידת המכונה בספרינט הבא.

פיצ'רים באיטרציה 2 :

1. בניית האפליקציה עם **GUI** בסיסי.

2. תמיכה ותקשורת בין השרת ללקוח ובדיקת צד השרת לאחר קליטת התמונה מהמשתמש (תשובה מהשרת וקבלת הטקסט).

איטרציה 3:

מטרה: לימוד המכונה את האותיות לפי הdataset, ובאמצעותה נשלח למכונה תמונה והיא תדע לזהות ולסווג את האות.

פיצ'רים באיטרציה 3 :

1. חיפוש בתוך מאגר המידע של האותיות – המכונה מחפשת במאגר המידע באיזה אות מדובר.

2. זיהוי האותיות בתמונה שהתקבלה – המכונה לומדת תבניות של אותיות בכך שעוברת על עשרות אלפי דוגמאות של אותיות, בונה לעצמה תבניות של אותיות ובכך מזהה את התו ומעביר את התו לשרת.

3. סיווג האותיות בתמונה שהתקבלה – לאחר חיפוש במאגר והבנה באיזה אות מדובר, היא מסווגת את האות (אומרת באיזה אות או תו מדובר).

4. זיהוי מקום של האות בתוך התמונה – מסדרים את האות שסיווגנו לפי הסדר שהם היו בתמונה.

5. הפיכת הסיווג לפונט קריא – האותיות שסווגו הופכות לפונט קריא שניתן להבין.

6. קבלת לינק לקריאה בפונט קריא – לאחר סידור וסיווג הטקסט שהיה בתמונה, המערכת מעבירה את הטקסט שהומר לקובץ txt או word.

איטרציה 4:

מטרה : ממשיך עם עיבוד התמונה, מבצעים עוד מספר פיצ'רים לצורך זיהוי המכונה הלומדת את האותיות בתמונה , מעין להכין את השטח לפני לימוד המכונה.

פיצ'רים באיטרציה 4 :

1. דגימת התמונה כדי למצוא איזה גוון מבדיל בין הטקסט לרקע התמונה – לאחר שהמשתמש שולח את התמונה למכונה, המכונה למעשה מבדילה בין הטקסט לבין רקע התמונה, שלב זה מבוצע בשביל שלב זיהוי האותיות בהמשך.

2. זיהוי שורות הטקסט ויישור השורה כך שהאותיות יהיו אופקיות – המכונה למעשה מזהה שורות טקסט ומיישרת אותם במידת הצורך , שלב זה פותר בעיות של דפים שסריקתם נעשתה בזווית, מה שמקשה על התוכנה לזהות את צורת האותיות בצורה נכונה.

3. הפרדת השורות השלמות למילים על פי מרווחים – המכונה מפרידה את השורות למילים לצורך זיהוי האותיות בהמשך.

4. הפרדת השורות השלמות לאותיות על פי המילים – המכונה מפרידה את המילים לאותיות לצורך זיהוי האותיות בהמשך.

איטרציה 5:

מטרה : הוספת פיצ'רים לצד השרת ושדרוג האפליקציה והוספת עיצוב.

פיצ'רים באיטרציה 5 :

1. תרגום הטקסט שזוהה בתמונה – אם המשתמש בחר בתרגום הטקסט, הטקסט שהיה בתמונה בשפה האנגלית יתורגם לשפה העברית

2. הרצת קוד שזוהה בתמונה – אם המשתמש בחר בסריקת קוד, תמונת הקוד שצילם המשתמש בשפת python, מומרת לקוד וניתן להריץ אותה באמצעות המערכת.

3. העברת קובץ הטקסט ל **PDF**- אם המשתמש בחר בהעברת קובץ הטקסט לpdf, הטקסט שהומר ועבר לקובץ txt או word מועבר לקובץ pdf.

פרק 5 : עיצוב- מדי איטרציה

איטרציה 1 - דוגמה לתקשורת בין שרת ללקוח בפייתון

מטרה:

מטרת הספרינט הייתה בניית אב טיפוס של שרת שמקבל תמונה ופועל לפי עיבוד תמונה, הוא מאפשר לערוך את התמונה, הוא בודק ומזהה את הצבע שחזר על עצמו הכי הרבה בתמונה, ומציג את התמונה לאחר העריכה ואת הצבע שחזר על עצמו הכי הרבה בתמונה.

תהליך העבודה:

חילקתי את הספרינט למספר חלקים.

תחילה , כתבתי ממשק GUI זמני למשתמש לצורך בדיקת עריכת התמונה והתקשורת עם השרת.

לאחר מכן, מימשתי פונקציות של עריכת התמונה לפיצ'רים של חיתוך התמונה, סיבוב התמונה, הבהרת התמונה (עריכת התמונה מצד הלקוח) , מימשתי אותם בשפת פייתון באמצעות הספרייה Opencv2.

כמו כן, מימשתי פונקציה אשר בודקת את הצבע שחוזר על עצמו הכי הרבה בתמונה ומציג אותו.

יתר על כן, כתבתי את צד השרת ובדקתי תקשורת בין השרת ללקוח ומימשתי את שליחת התמונה אל השרת וקבלת התמונה מצד השרת והצגתה.

איטרציה 2 - האפליקציה, צד הלקוח של הפרויקט

מטרה:

בניית אב טיפוס לאפליקציה מצד הקליינט עם GUI בסיסי שתומכת ויכולה לדבר עם השרת, להעביר לו תמונה (שניתן לערוך באפליקציה) , ולקבל ממנה את התמונה ערוכה. כמו כן, חקרתי על מודל CNN וכיצד יש לכתוב את המודל לצורך למידת המכונה בספרינט הבא.

תהליך העבודה:

בתחילת הספרינט חקרתי את React native , ספרייה בJavaScript לצורך פיתוח web application.

לאחר מכן ניסיתי להתממשק עם הספרייה, בניתי אפליקציית דוגמה לצורך בדיקה וניסיון.

בהמשך, בניתי ממשק GUI בסיסי לאפלקציה.

לאחר זאת, ביצעתי בקוד האפלקציה כפתור אשר פותח תמונה מהגלריה, ומציג אותה במסך האפלקציה. לאחר מכן קראתי וחקרתי כיצד לשלוח לשרת בפייתון את התמונה , והחלטתי לשלוח את התמונה מצד האפליקציה באמצעות שליחת הודעת post. בנוסף, יצרתי במקביל את השרת אשר מקבל הודעות Post. שלחתי בצד האפלקציה את התמונה בשליחת הודעת Post, קיבלתי אותה בצד השרת , פתחתי אותה , שמרתי אותה והצגתי אותה בצד השרת והוספתי פונקציה אשר בודקת את הצבע הנפוץ בתמונה ומציגה אותו לצורך לימוד המכונה בהמשך. אחרי כן, שדרגתי את האפלקציה מבחינת העיצוב.

כמו כן, חקרתי על מודל CNN ובנייתה לצורך הספרינט הבא.

איטרציה 3 - למידת המכונה:

מטרה:

מטרת הספרינט הייתה בניית המודל, אימון (לימוד) המכונה את האותיות לפי המאגר המידע, (EMNIST). אל המכונה שלחתי למודל תמונה מתוך המאגר מידע שהכנו לצורך הבדיקה והמכונה תדע לזהות ולסווג את האות.

תהליך העבודה:

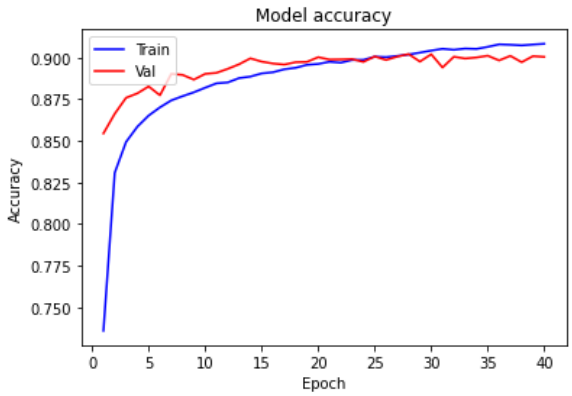
בתחילת הספרינט חקרתי את מודל CNN, הכנתי את סביבת העבודה שבה השתמשתי בספרינט , google collab, הורדתי את הספריות איתן עבדתי בספרינט שהם TensorFlow וkeras, ואת מאגר המידע EMNIST.

בהמשך, בניתי את מודל CNN באמצעות keras והאמנתי את המכונה עם מאגר המידע שהורדתי. הגעתי ל85 אחוזי הצלחה. רציתי לשפר את אחוזי ההצלחה של המכונה, לכן הוספתי למודל עוד שכבות ותמונות, וביצעתי יותר אימונים על המודל, עד שהגענו ל90 אחוזי הצלחה.

לאחר מכן, בדקתי את המכונה על ידי מתן תמונות לבדיקת המודל שהכנתי מראש וראיתי את ההצלחות שלו ואת מה צריך לשפר.

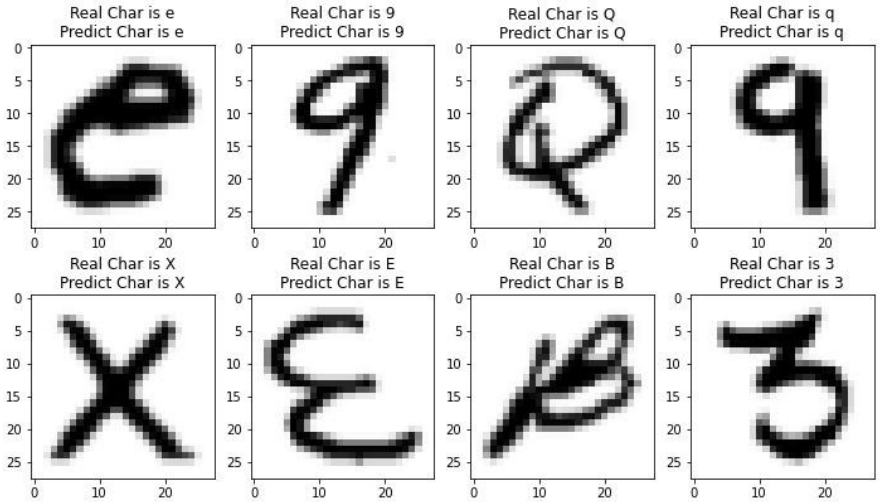
לאחר זאת, קלטתי מהמשתמש תמונה של אות שעל המכונה לסווג, והמכונה החזירה את האות שקיבלה מהמשתמש.

התוצר הסופי הוא מכונה אשר יודעת לזהות את האותיות בדיוק של כמעט 90%.




```
Test loss: 0.299619197845459
Test accuracy: 0.8995159268379211
```

ניתן לראות גם על פי הדוגמה של סט הנתונים שהכנו לשם בדיקת המודל, את דיוק המודל:



המודל :

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
batch_normalization (BatchNo (None, 26, 26, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 32)        25632
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 32)        128
_________________________________________________________________
dropout (Dropout)            (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 64)          102464
_________________________________________________________________
batch_normalization_5 (Batch (None, 4, 4, 64)          256
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 1, 1, 128)         131200
_________________________________________________________________
batch_normalization_6 (Batch (None, 1, 1, 128)         512
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 47)                6063
=================================================================
Total params: 332,015
Trainable params: 331,183
Non-trainable params: 832
_________________________________________________________________
```

איטרציה 4 - איחוד חלקי הפרויקט וסיווג הטקסט :

מטרה:

מטרת הספרינט הייתה איחוד כל חלקי הפרויקט לכדי מערכת גדולה.

קבלת תמונה של אות מהמשתמש בreact , וסיווגה של האות על ידי המודל.

עיבוד התמונה של הטקסט המלא לכדי פירוקו של תמונה לטקסט, לאחר מכן לשורה, למילה ולאות, לשלוח אל המכונה הלומדת ולקבל תשובה שלמה של הטקסט בתמונה.

תהליך העבודה:

בספרינט הנ"ל איחדתי את כל חלקי הפרויקט, כאשר חיברתי את המודל עם צד השרת והלקוח בreact.

לאחר מכן, קלטתי תמונה של אות מהלקוח, והעלתי אותה אל השרת, השרת שלח אל המודל והמודל החזיר לשרת את סיווג האות.

לאחר מכן, פיצלתי את התמונה השלמה לכדי פסקאות, לאחר שחילקתי לפסקאות, חילקתי לשורות, ומשורות למילים. לאחר מכן עברתי על כל מילה בתמונה, וכל מילה פיצלתי לאותיות, והכנסתי אל קובץ csv, את קובץ הcsv העברתי אל המודל, המודל החזיר תשובה למילה שסווגה והמשיך לרוץ על כל המילים, שבין כל מילה יש רווח, והכנסתי אל תוך מחרוזת שלמה סופית שנכנסת אל קובץ txt.

איטרציה 5 - הפרויקט השלם והוספת פיצ'רים :

מטרה:

העברת הטקסט מהשרת אל הלקוח, הוספת פיצ'רים לצד הלקוח ושדרוג האפליקציה בreact.

הפיצ'רים באיטרציה 5 :

1. תרגום הטקסט שזוהה בתמונה – אם המשתמש בחר בתרגום הטקסט, הטקסט שהיה בתמונה בשפה האנגלית יתורגם לשפה העברית

2. דגימת התמונה והרקע שלה.

3. העברת קובץ הטקסט ל PDF- אם המשתמש בחר בהעברת קובץ הטקסט לpdf, הטקסט שהומר ועבר לקובץ txt או word מועבר לקובץ pdf

4. העברת קובץ הטקסט לWord.

תהליך העבודה:

בספרינט הנ"ל התחלתי עם הפיצ'רים, הוספתי את הפיצ'רים של הפרויקט באמצעות פונקציות שממשתי, ולאחר מכן שדרגתי את צד הלקוח ואת האפלקציה . בנוסף יצרתי את הפונקציה אשר יוצרת את התיקייה עם סיווג הטקסט והפיצ'רים.

פרק 6 : קוד הפרויקט :

הפרויקט כולל בתוכו מספר קבצים :

1. **ImageToTextBeckend – Server side react.py-** צד השרת של הפרויקט אשר מקבל מreact את בקשת הPost , מנתח אותה ושולח אותה אל Core_ProjectImageToText.py בהתאם לבקשה.
2. **Core_ProjectImageToText.py** – ליבת הפרויקט, מנתח מה הבקשה של המשתמש, אם מדובר בטקסט הוא מפצל את השורות, המילים והאותיות באמצעות imageProcessing.py , מכניס לקובץ csv, שולח לCnn.py , מקבל את תשובת הסיווג , יוצר תיקייה עם קובץ טקסט של הסיווג והפיצ'רים בהתאמה לבקשת המשתמש.
3. **imageProcessing.py-** משתמש בטכנולוגיית עיבוד תמונה, מפצל את התמונה לפסקאות, שורות, מילים ואותיות ומכניס אל קובץ csv ושולח ל- Core_ProjectImageToText.py.
4. **Cnn.py-** מודל המכונה אשר מקבלת את קובץ csv, מסווגת את התמונה ומחזירה אל Core_ProjectImageToText.py.
5. **Features_ProjectImageToText.py-** פונקציות עם הפיצ'רים של הפרויקט.
6. **App.js-** צד הלקוח של הפרויקט בreact native עם עיצוב האפלקציה ושליחת המידע של המשתמש לשרת.

הפונקציות בImageToTextBeckend – Server side react.py :

פונקציית fileUpload- פונקציה המקשיבה ומתקשרת עם הלקוח, מקבלת את בקשת השרת, מנתחת אותו ושולחת אותו לפונקציה המטפלת בבקשה.

הפונקציות בCore_ProjectImageToText.py :

1.פונקציית Choice(img,imagePath,sentence_or_character,pathFolder):

פונקציה אשר מקבלת את בקשת הלקוח, בודקת אם מדובר בתו או טקסט ומפנה אותו לפונקציה המתאימה לבקשה עם התמונה והנתיב שבחר המשתמש.

2. פונקציית csvWriterLetter(fil_name, nparray) - פונקציה אשר ממירה לקובץ csv את התמונה של התו שנבחר.

3. פונקציית forLetter(img,imagePath,pathFolder)- פונקציה מרכזית בשביל תו, אשר מקבלת את התמונה של התו, ואת הנתיב שבחר, שולחת אל פונקציית סיווג התו בcnn.py .

4. פונקציית:csvWriter(fil_name, nparray - פונקציה אשר ממירה לקובץ csv את התמונה של המילה בטקסט שנבחר.

5. פונקציית make_square - פונקציה שגורמת לתמונה להיראות טובה יותר ולעזור למכונה לזהות את המילה טוב יותר. זה מוסיף חזית לבן ולכן כאשר גודל התמונה ישתנה, האות לא תהיה על כל המסך ויהיה מרוכז יותר.

6. פונקציית toFileTxt- פונקציה אשר כותבת לקובץ את הטקסט שסווג בתמונה.

7. פונקציית CoreFunction- זו הפונקציה העיקרית לזיהוי המשפט, היא כוללת את כל הקריאות לפונקציות, מפצלת את התמונה לפסקאות, שורות, מילים ואותיות באמצעות ImageProcsseing.py , שולחת אל המכונה, מקבלת תשובה ומכניסה אל תוך התיקייה את התשובה יחד עם הפיצ'רים של הפרויקט שנמצאים ב Features_ProjectImageToText.py.

הפונקציות בimageProcessing.py :

פונקציית cutLettersFromWord - מקבלת תמונה של מילה שפוצלה בCoreFunction , ומפצלת לאות ומכניסה אל קובץ csv ושולחת אל המודל.

הפונקציות בCnn.py :

1. פונקציית Model(count_letters) - פונקציית סיווג התמונה של מילה מהמכונה הלומדת, אשר פותחת את הבינה המלאכותית, מקבלת קובץ csv של מילה, מסווגת לפי הקובץ ומחזירה אל CoreFunction

2. פונקציית model_CNN(pathFolder) - פונקציית סיווג התמונה של תו מהמכונה הלומדת, אשר פותחת את הבינה המלאכותית, מקבלת קובץ csv של תו, מסווגת לפי האות, מכניסה אל קובץ ומוסיפה לקובץ של כל התווים שזוהו לאחרונה.

הפונקציות בFeatures_ProjectImageToText.py :

1. **פונקציית CommonColor -** פונקציה שמכניסה לתיקיה שהמשתמש בחר את הרקע של התמונה עם הטקסט.

2. **פונקציית convert_to_word -** פונקציה שמכניסה לתיקיה שהמשתמש בחר קובץ word יחד עם הטקסט שזוהה בתמונה.

3. **פונקציית translateImageText -** פונקציה שמכניסה לתיקיה שהמשתמש בחר קובץ txt יחד עם תרגום הטקסט שהיה בתמונה.

4. **פונקציית convert_to_pdf** - פונקציה שמכניסה לתיקיה שהמשתמש בחר קובץ Pdf יחד עם הטקסט שזוהה בתמונה.

הפונקציות בApp.js :

1. **פונקציית HomeScreen**- פונקציית מסך הבית של האפלקציה שבה יש את הכפתורים שמעלים את התמונה, כותבים את הנתיב, ושולחים אל השרת בהתאם למה שמבקשים.

2. **פונקציית responseForText** – פונקציה ששולחת לפונקציה handleUploadImage עם הבחירה של טקסט, ופותחת מסך של קבלת תשובה של טקסט.

3. **פונקציית responseForLetter** – פונקציה ששולחת לפונקציה handleUploadImage עם הבחירה של תו, ופותחת מסך של קבלת תשובה של אות.

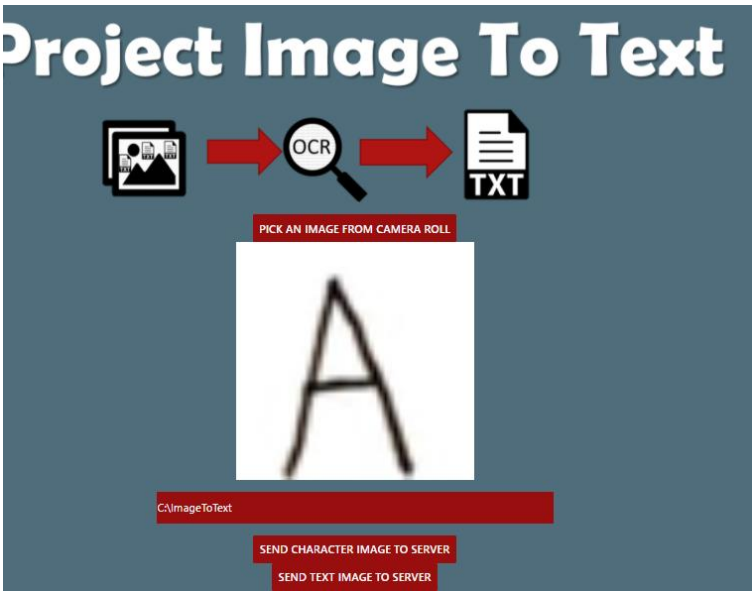
4. **פונקציית handleUploadImage** – פונקציה אשר פותחת את הגלריה בשביל שהמשתמש יבחר את התמונה של הטקסט שברצונו לסווג, ובנוסף שולחת אל השרת יחד עם הנתיב, והבחירה אם תו או טקסט.

פרק 7 : שימוש בפרויקט :

אם מדובר בסיווג של תו בודד:

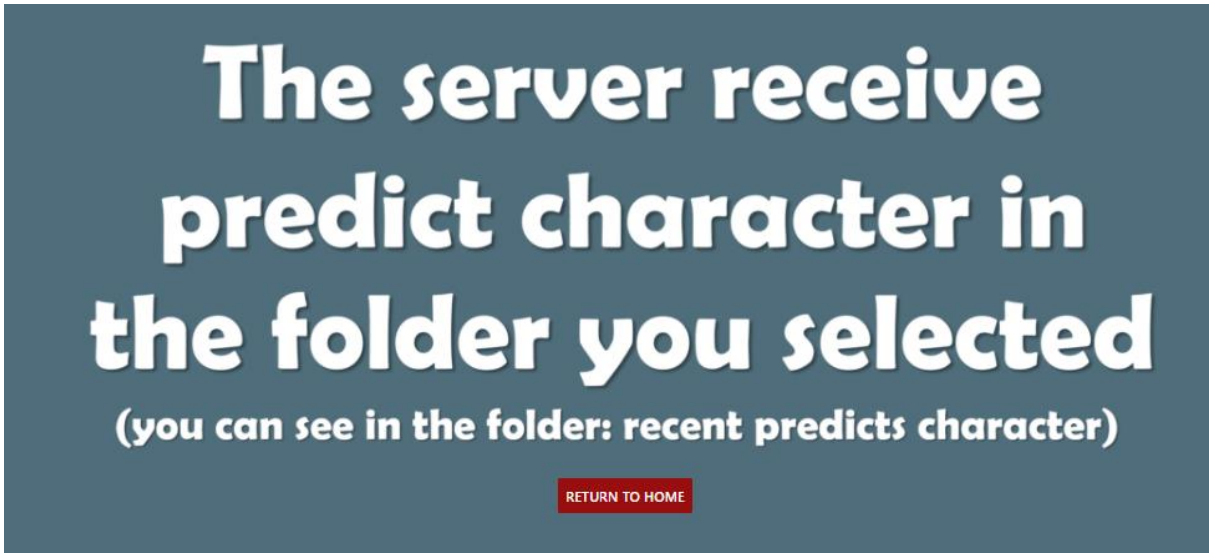
1. לוחצים על הכפתור PICK AN IMAGE FROM CAMERA ROLL ומעלים תמונה של תו.
2. בוחרים נתיב בו אנו רוצים לשמור את הסיווג של התו.
3. לוחצים על הכפתור SEND IMAGE CHARACTER TO SERVER

לדוגמה:



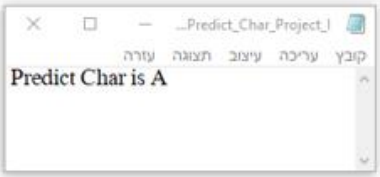
שולחים אל השרת..

השרת מחזיר קובץ של סיווג התו בנתיב שבחר המשתמש, וקובץ עם סיווגים אחרונים.



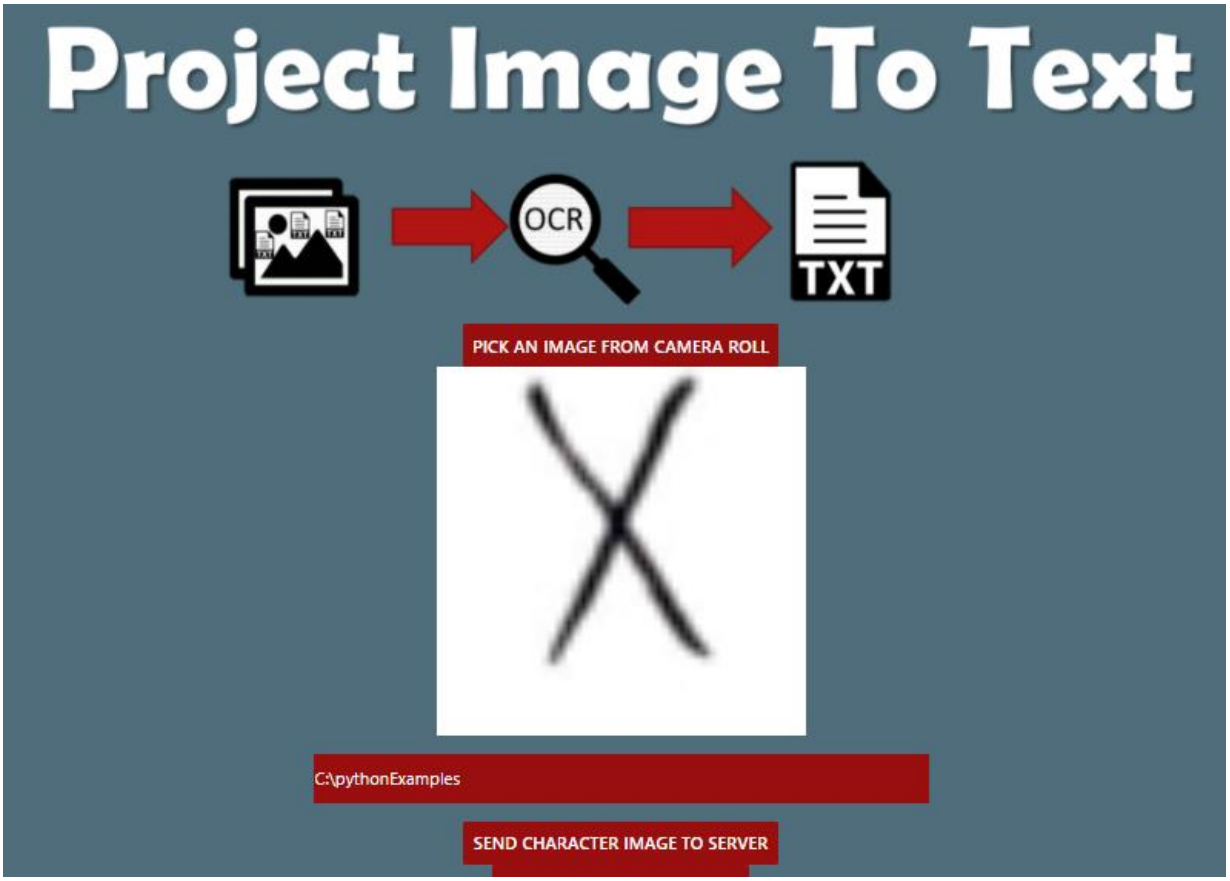
בנתיב שבחרנו..

קובץ סיווג האות

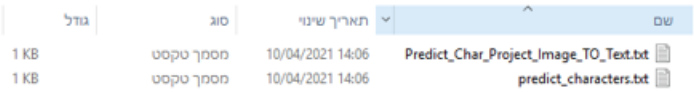


קובץ סיווגים אחרונים

אם נבצע את אותו התהליך גם לתו נוסף :



בנתיב שבחרנו..



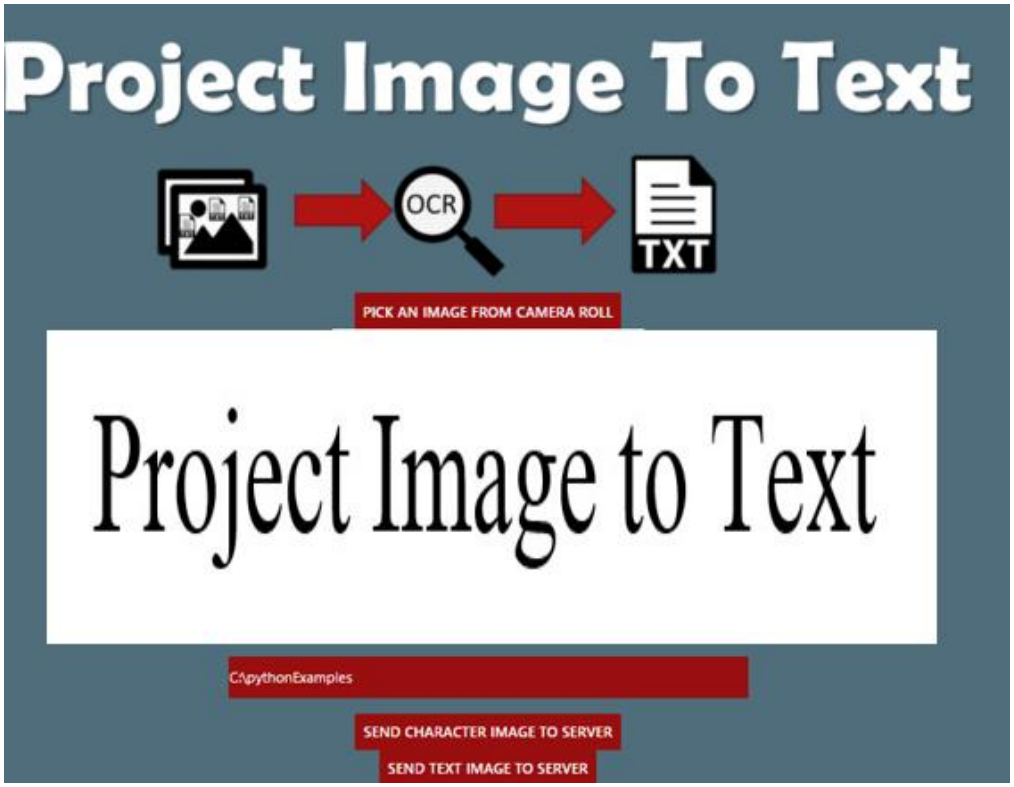
הוספה של האות לקובץ סיווגים אחרונים

קובץ סיווג האות

אם מדובר בסיווג של טקסט:

1. לוחצים על הכפתור PICK AN IMAGE FROM CAMERA ROLL ומעלים תמונה של טקסט.
2. בוחרים נתיב בו אנו רוצים לשמור את הסיווג של הטקסט.
3. לוחצים על הכפתור SEND TEXT IMAGE TO SERVER

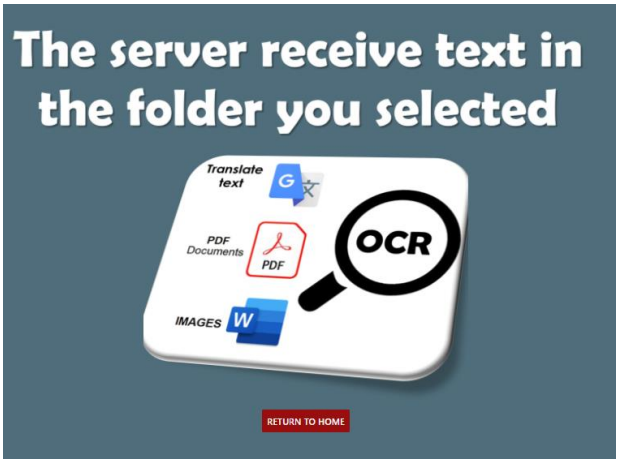
לדוגמה,



שולחים אל השרת...

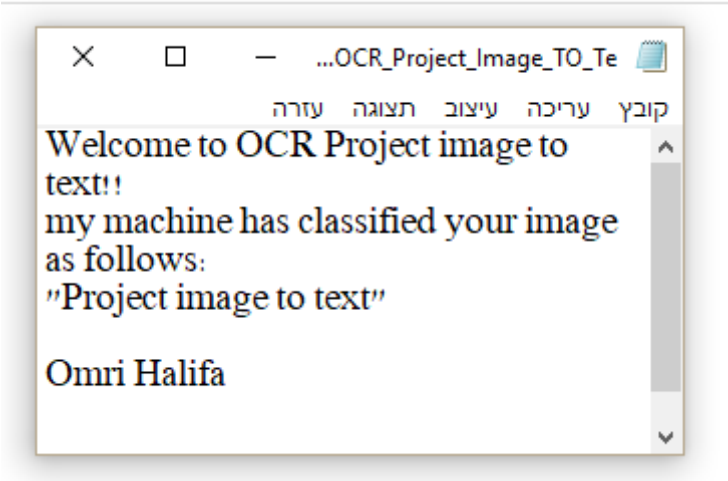
השרת מחזיר קובץ של סיווג הטקסט בנתיב שבחר המשתמש, ובנוסף מחזיר קבצים :

1. קובץ txt עם סיווג הטקסט שנמצא בתמונה.
2. קובץ txt שמכיל בתוכו את התרגום של הטקסט מאנגלית לעברית.
3. קובץ pdf שמכיל בתוכו את סיווג הטקסט בתמונה.
4. קובץ word שמכיל בתוכו את סיווג הטקסט בתמונה.
5. תמונה שמראה את רקע התמונה.

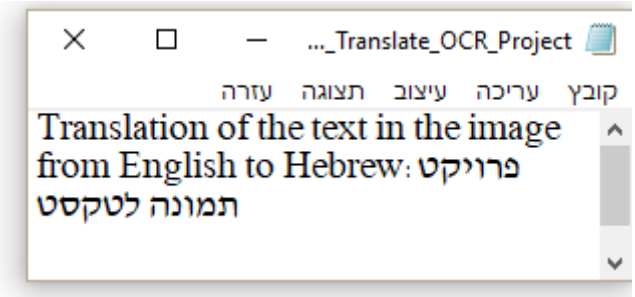


תשובות השרת :

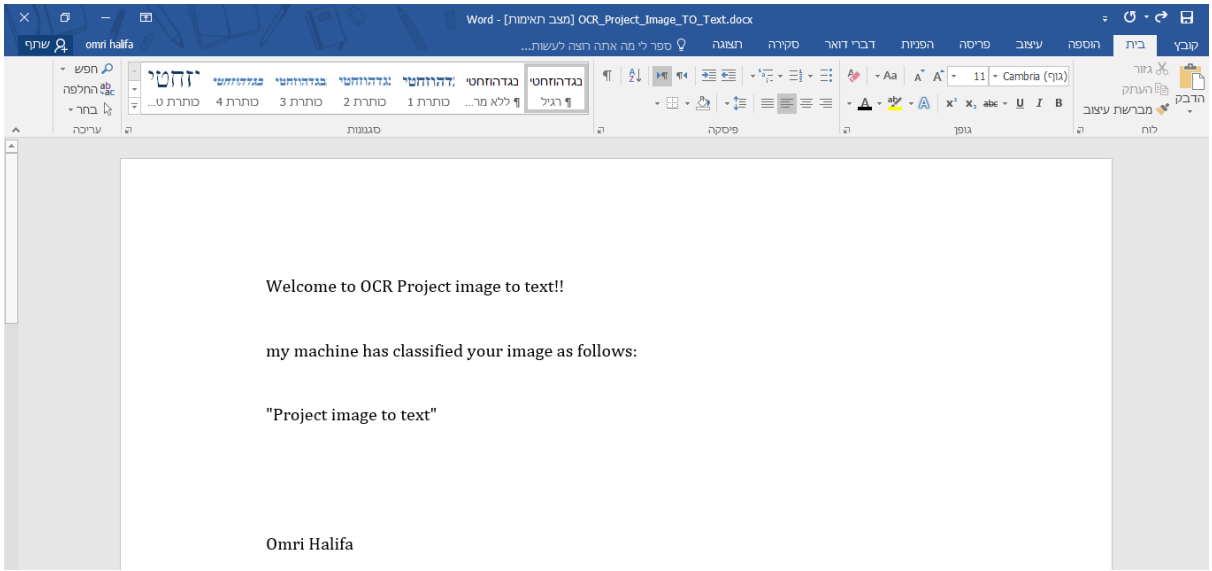
קובץ הטקסט של סיווג התמונה:



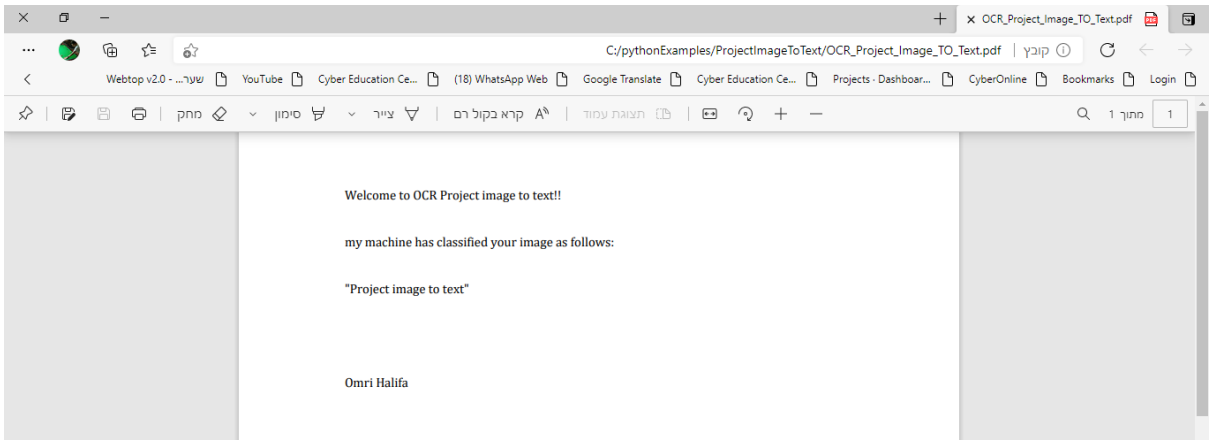
קובץ הטקסט של תרגום התמונה:



קובץ word עם סיווג התמונה:



קובץ Pdf עם סיווג התמונה:



קבצים בפרויקט:

ImageToTextBeckend – Server side react.py:

```
import os
from flask import Flask, request
from flask_cors import CORS
from PIL import Image
import base64
import io
import cv2
import numpy as np

import Core_ProjectImageToText

UPLOAD_FOLDER = 'C:\\filesMagshimim'

app = Flask(__name__)


"""
A function that listens and communicates with the client, receives the server's
request,
analyzes it and sends it to the function that handles the request.
"""
@app.route('//upload', methods=['POST'])
def fileUpload():
    target=os.path.join(UPLOAD_FOLDER,'test_docs')
    if not os.path.isdir(target):
        os.mkdir(target)
    clientRequest = request.data
    clientRequestStr = clientRequest.decode("utf-8") #convert client request to
str
    print(clientRequestStr)
    sentence_or_character =
clientRequestStr[clientRequestStr.find('"choice":')+9:clientRequestStr.find(',"pat
h":')] # Analyzes the user request, whether to return an answer for one character
or text
    pathFolder = clientRequestStr[clientRequestStr.find('"path":')+7:-1] #Analyzes
the path selected by the user to save the text
    pathFolder = pathFolder[1:-1] #remove the quotes
    if (not "null" in clientRequestStr):
        #Receives the user's image, turns it into an image, saves it, and sends it
to a function according to the user's request:
        place = clientRequestStr.find('base64,') + 7
        imagestr = clientRequestStr[place : -1]
        image = base64.b64decode(imagestr)
        fileName = 'result.png'
        imagePath = ('C:\\ProjectImageToText\\' + "result.png") #const
        img = Image.open(io.BytesIO(image))
        img.save(imagePath, 'png')
        img = cv2.imread(imagePath)
        cv2.imwrite(imagePath, img)
        img = np.array(Image.open(imagePath))
        print("Image received.")
        img = cv2.imread(fileName)  # Opens the image as an image
        pathFolder = pathFolder+ "\\ProjectImageToText"
        try:
```

```
            os.mkdir(pathFolder)
        except OSError as error:
            print(error)

Core_ProjectImageToText.Choice(img,imagePath,sentence_or_character,pathFolder)
#Sends the function to the function, whether it is a single character or text, the
image path, and the path selected by the user
        response = "Ok."
    return response

if __name__ == "__main__":
    app.secret_key = os.urandom(24)
    app.run(debug=True,host="127.0.0.1",use_reloader=False)

CORS(app, expose_headers='Authorization')
```

ב: Core_ProjectImageToText.py

```
import cv2
import pytesseract
from PIL import Image
import csv
import numpy as np
from pytesseract import Output
import CNN
import Features_ProjectImageToText
import imageProcessing
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-
OCR\tesseract.exe'


#this function is the one called.
#this seperates the sentence recognition from the word recognition.
def Choice(img,imagePath,sentence_or_character,pathFolder):
    if (sentence_or_character =='"1"'):#if the given in is 1 than it means that it
is for a sentence
                CoreFunction(img,pathFolder)
    elif (sentence_or_character =='"2"'):#if the given code is 2 than it meens
that it is for a letter.
                forLetter(img,imagePath,pathFolder)


#this function it the csv writer for singular letters.
#it transforms the letter into a csv in a correct way for the machine
#for a singular letter.
def csvWriterLetter(fil_name, nparray):
    example = nparray
    with open(fil_name + '.csv', 'w', newline='') as csvfile:
        writer = csv.writer(csvfile, delimiter=',')
        writer.writerow(example)
        writer.writerow(example)

#this function is the main function for the letter recognition,
#it is given the image, the path, and the path to save it to.
def forLetter(img,imagePath,pathFolder):
    res = cv2.resize(img, (28, 28))
    cv2.imwrite(imagePath, res)#writes the image
    img = np.array(Image.open(imagePath))#create an arr for the file.
```

```
    newimg = []
    for line in img:
        for pixle in line:
            newimg.append(255 - pixle[0])#invert the colors for the machine.
    img = newimg
    csvWriterLetter("data to use", img)#create csv file.
    CNN.model_CNN(pathFolder)#call the machine to make a prediction.


def csvWriter(fil_name, nparray):
    example = nparray
    with open(fil_name + '.csv', 'w', newline='') as csvfile:  # opening the file
and calling it csvfile
        writer = csv.writer(csvfile, delimiter=',')  # creating a csv writer with
delimiter as comma.
        writer.writerows(example)  # using premade functions to add the letters's
images from the list into a csv file.


#make square is a function that makes the image look beter, and help
#the machine recognize the word better
#it adds a white backround so when the image will be resized, the letter wont be
on the whole screen
#and will be more centered.
def make_square(im, min_size=256, fill_color=(255, 255, 255, 255)):
        x, y = im.size
        size = max(x+4, y+4)
        new_im = Image.new('RGBA', (size, size), fill_color)
        new_im.paste(im, (int((size - x) / 2), int((size - y) / 2)))
        return new_im

def toFileTxt(final_string,pathFolder):
    pathFolder = pathFolder +"\\OCR_Project_Image_TO_Text.txt"
    f = open(pathFolder, "w")
    f.write("Welcome to OCR Project image to text!!\n")
    f.write("Our machine has classified your image as follows:\n")
    f.write('"' + final_string + '"\n')
    f.write("\n")
    f.write("Omri Halifa and Shachar Sangier (:")
    f.close()


#this is the main functio for the sentence recognition, it includes all the calls
for the
#minor functions, and preforms most of the progress ,for example the word
seperation,letter seperation
#resizing, corrections, and preparing the csv for the machin to make its
predictions.
def CoreFunction(img,pathFolder):
    print("Loads and recognize the text in the image...")
    h, w, _ = img.shape

    img_words = img.copy()
    d = pytesseract.image_to_data(img_words, output_type=Output.DICT)#this
function takes the image
    # and crops it into boxes of words, lines and paragraphes.
    n_boxes = len(d['level'])
    ROI_number = 0
    for i in range(d['level'].count(5)+d['level'].count(4)-1+d['level'].count(3)-
```

```
1+d['level'].count(2)-1):
        # line to add a space to the end code, and to save the words in a file in
a dir we create
        # so in the futer we can seperate them into letters.
        i = i + d['level'].index(5)# we want to skip the 5 first places because
they represent the start of the proces(first paragraph,line and word).
        if (d['conf'][i] != '-1'):
            (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i],
d['height'][i])#we save the dimantions and the position of the word
            crop_word = img_words[y:y + h, x:x + w]#we crop the initial image
            cv2.imwrite('.\words\word{}.png'.format(ROI_number), crop_word)#we
save it in a dir with the index as its name.
            ROI_number += 1
    count_letter = 0
    final_string = " "
    num_word = 0
    #in this for loop we seperate every word in the dir by the index, and the
ammount
    #saved in the last for.
    #here we will also split the word into letters and save the image in a csv
file for out ai machine.
    for i in range(ROI_number):#for every number of words counted.
        filename ='.\words\word{}.png'.format(num_word)
        temp_img = []#defining a list for the letters in the word to be in.
        num_word+=1
        img = cv2.imread(filename)#loading file from filename.

        h, w, _ = img.shape# using .shape to take dimantions of the image.
        boxes = pytesseract.image_to_boxes(img)#main way of spliting the word into
letters,
        # if doesnt work we have a secondery method longer in the code.
        count_letter = 0
        for b in boxes.splitlines():#for every letter found
            b = b.split(' ')
            start_point = (int(b[1]), h - int(b[2]))#save start pos
            end_point = (int(b[3]), h - int(b[4]))#save end pos
            crop = img.copy()

            out = crop[(h-int(b[4])):(h-
int(b[2])),int(b[1]):int(b[3])].copy()#crop by the start and end pos's.
            count_letter+=1
            cv2.imwrite("temp.png",out)#save the image temporarly
            t = Image.open("temp.png")
            new = make_square(t)#using a function that shahar made to make the
image a bit eazier to recognize
            new.save("temp.png")
            out = cv2.imread("temp.png")
            out = cv2.cvtColor(out,cv2.COLOR_BGR2GRAY)#truning the image into
greyscale
            #cv2.imshow("letter",out)
            #cv2.waitKey()
            res = cv2.resize(out,(28,28));# resizing the image for the ai to
understand
            cv2.imwrite("temp.png",res)
            nparr = np.array(Image.open("temp.png"))#adding the image into a list
that contains all of the letters,
            # so it will be easier for the futer steps
            newimg = []
            for line in nparr:
```

```
                for pixle in line:
                    newimg.append(255 - pixle)#reverting the images colors for the
machine to understand.
            temp_img.append(newimg)#add to the final list
        if (count_letter==0):#if the first split method didnt work as sapose to we
move into a secondery method
            count_letter = imageProcessing.cutLettersFromWord(filename)#function
in difrent file.
        else:
            csvWriter("word", temp_img)#this function transforms the list given to
it into a csv file, for the machine.
        final_string += CNN.Model(count_letter)#call the cnn modle to make its
predictions based on the information given
        final_string += " "
    final_string = final_string.lower()#lower all leters from detection
    final_string = final_string.replace("0", "o")#make some corrections
    final_string= final_string.split()
    final_string[0] =final_string[0].capitalize()#capitalize the string based of
the list positions..
    final_string = " ".join(final_string)#join the list into a final string
    final_string = final_string.strip()
    toFileTxt(final_string,pathFolder)#use a function to make a text file
    print("The recognize text is:" + final_string)
    print("The machine created a text file with image classification successfully
in: "+pathFolder+" and also created several features that our project supports!")
    print("1. translate the text in the image from Hebrew to English")
    print("2. To move the txt file to a word file")
    print("3. To transfer the txt file to a pdf file")
    print("4. Show background of the image text")
    Features_ProjectImageToText.translateImageText(final_string,pathFolder)
    Features_ProjectImageToText.convert_to_word(pathFolder)
    Features_ProjectImageToText.convert_to_pdf(pathFolder)
    Features_ProjectImageToText.CommonColor(img,pathFolder)
```

imageProcessing.py בקובץ :

```
import cv2
from imutils import contours
from PIL import Image
import numpy as np
import csv


# this function makes adds some pixles to the side of the image
# so the process of resizing it will not harm the image, and so it will be
# more centered and will not fill the image(adds alittel for the prediction).
def make_square(im, min_size=256, fill_color=(255, 255, 255, 255)):
    x, y = im.size
    size = max(x + 4, y + 4)  # deciding what size to resize it to
    new_im = Image.new('RGBA', (size, size), fill_color)  # make a new image just
made from black with new size
    new_im.paste(im, (int((size - x) / 2), int((size - y) / 2)))  # paste the
letters image on the new black image.
    return new_im  # return the new image.


# the csv writer for the main sentence recognition
# it recives a list of images, that are made as an arr of grey-scale pixles
# it creats a csv file that contains all of the images(pixles seperated by commas
```

```
and letters seperated by \n)
def csvWriter(fil_name, nparray):
    example = nparray
    with open(fil_name + '.csv', 'w', newline='') as csvfile:  # open a new csv
file and name it as csvfile
        writer = csv.writer(csvfile, delimiter=',')
        writer.writerows(example)  # write the arr into the file with a writer
that seperates by commas.


# this function is called on the case when the main function cant cut the word
correctly
# this function cuts the word into a letters by finding when one starts and when
it ends.
#
def cutLettersFromWord(image):
    image = cv2.imread(image)  # read the image from where it was saved.
    image = cv2.bitwise_not(image)  # revert the colors for the secondery method.
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  # grey scale the image.
    thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU + cv2.THRESH_BINARY)[1]
    cnts = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if len(cnts) == 2 else cnts[1]
    cnts, _ = contours.sort_contours(cnts, method="left-to-right")
    temp_img = []  # defining a list for the letters in the word to be in.
    ROI_number = 0
    count = 0
    for c in cnts:
        area = cv2.contourArea(c)
        if area > 10:  # ignore all the dots and small points (like i's and j's)
            x, y, w, h = cv2.boundingRect(c)
            ROI = 255 - image[y:y + h, x:x + w]
            cv2.imwrite("temp.png", ROI)
            t = Image.open("temp.png")
            new = make_square(t)  # square the image for the machine.
            new.save("temp.png")  # save the new image
            out = cv2.imread("temp.png")
            out = cv2.cvtColor(out, cv2.COLOR_BGR2GRAY)  # convert to grey

            res = cv2.resize(out, (28, 28))  # resizing the image for the ai to
understand
            cv2.imwrite("temp.png", res)
            nparr = np.array(Image.open("temp.png"))  # make an array of the
letters from the image
            newimg = []
            for line in nparr:
                for pixle in line:
                    newimg.append(255 - pixle)  # invert color.
            temp_img.append(newimg)  # add to image list.
            count += 1
            csvWriter("word", temp_img)  # csv write the word.
    return count
```

Cnn.py בנ :

```
import numpy as np
from keras.models import load_model
import pandas as pd
import os
import datetime

# Constants
HEIGHT = 28
WIDTH = 28


# in this function we have the prediction for the sentence
# we load the modls save that we created to use to make
# predictions for each letter.
def Model(count_letters):
    model = load_model('emnist_model.h5')  # load modle
    # dictionary of Chars:
    # we use this dict so we will be able to change the reply\ predict into the
actual letter.
    chars = {
        0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9:
'9',
        10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F', 16: 'G', 17: 'H',
18: 'I', 19: 'J',
        20: 'K', 21: 'L', 22: 'M', 23: 'N', 24: 'O', 25: 'P', 26: 'Q', 27: 'R',
28: 'S', 29: 'T',
        30: 'U', 31: 'V', 32: 'W', 33: 'X', 34: 'Y', 35: 'Z',
        36: 'a', 37: 'b', 38: 'd', 39: 'e', 40: 'f', 41: 'g', 42: 'h', 43: 'n',
44: 'q', 45: 'r', 46: 't'}

    image_csv = pd.read_csv(".\\word.csv", delimiter=',')  # open csv file
    test_csv = pd.read_csv(".\\emnist-balanced-test.csv", delimiter=',')

    # Normelizing
    # we normelize the images so we will be able to feed them into the machine.
    image_csv_x = image_csv.iloc[:, :]
    image_csv_x = np.asarray(image_csv_x)
    image_csv_x = image_csv_x.astype('float32')
    image_csv_x /= 255

    image_csv_x = image_csv_x.reshape(-1, HEIGHT, WIDTH, 1)  # last reshape to
28x28
    y_pred = model.predict(image_csv_x)  # make predictions
    word = " "
    for i in range(count_letters):
        word += chars[y_pred[i - 1].argmax()]  # add all predicted letters to one
string, and return it.
    return word


# modle code for the letter recognition
# this is where the prediction is made, and where
# we use the ai that we created.
def model_CNN(pathFolder):
    model = load_model('emnist_model.h5')  # here we load the ai machine that we
created and saved.
    # dictionary of Chars:
    # we use this dict so we will be able to change the reply\ predict into the
```

```
actual letter.
    chars = {
        0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9:
'9',
        10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E', 15: 'F', 16: 'G', 17: 'H',
18: 'I', 19: 'J',
        20: 'K', 21: 'L', 22: 'M', 23: 'N', 24: 'O', 25: 'P', 26: 'Q', 27: 'R',
28: 'S', 29: 'T',
        30: 'U', 31: 'V', 32: 'W', 33: 'X', 34: 'Y', 35: 'Z',
        36: 'a', 37: 'b', 38: 'd', 39: 'e', 40: 'f', 41: 'g', 42: 'h', 43: 'n',
44: 'q', 45: 'r', 46: 't'}
    image_csv = pd.read_csv(".\\data to use.csv", delimiter=',')  # open the csv
file.

    # Reshape and rotate EMNIST images
    def rotate(image):
        image = image.reshape([28, 28])
        image = np.fliplr(image)
        image = np.rot90(image)
        return image

    # Flip and rotate image
    image_csv = np.asarray(image_csv)

    # Normalise
    # we normelize the images so we will be able to feed them into the machine.
    image_csv = image_csv.astype('float32')
    image_csv /= 255
    image_csv = image_csv.reshape(-1, 28, 28, 1)  # reshape into 28x28
    # from test dataset:
    pred = model.predict(image_csv)  # make prediction
    image_csv = image_csv.reshape(image_csv.shape[0], 28, 28)
    print("Predict Char is " + chars[pred.argmax()])
    pathFolderchar = pathFolder + "\\Predict_Char_Project_Image_TO_Text.txt"  #
create text file
    pathFolderchars = pathFolder + "\\predict_characters.txt"  # create text file

    # here we insert the letter and the information into the text file.
    f1 = open(pathFolderchar, "w")
    f1.write("Predict Char is " + chars[pred.argmax()])
    f1.close()
    f2 = open(pathFolderchars, "a")
    filesize = os.path.getsize(pathFolderchars)
    if filesize == 0:
        f2.write('Character recognition - ImageToText project:\n')
    f2.write(str(datetime.datetime.now()) + " -Predict Char is: " +
chars[pred.argmax()] + '\n')
```

ב: Features_ProjectImageToText.py

```
from docx import Document
import cv2
import numpy as np
from google_trans_new import google_translator
from fpdf import FPDF

"""
A feature that inserts into the folder that the user has selected the background
of the image with the text.
"""
def CommonColor(image,pathFolder):
    img_arr = image.copy()
    img_temp = img_arr
    unique, counts = np.unique(img_temp.reshape(-1, 3), axis=0,
return_counts=True)
    img_temp[:, :, 0], img_temp[:, :, 1], img_temp[:, :, 2] =
unique[np.argmax(counts)]
    pathFolder =pathFolder+"\\Background_of_the_image"+".jpg"
    cv2.imwrite(pathFolder, img_temp)

"""
A feature that inserts into the folder that the user selected a word file along
with the text identified in the image.
"""
def convert_to_word(pathFolder):
    # Prepare document
    document = Document()
    pathFolderText = pathFolder + "\\OCR_Project_Image_TO_Text.txt"
    with open(pathFolderText, 'r') as textfile:
        for line in textfile.readlines():
            document.add_paragraph(line)
    pathFolderWord = pathFolder + "\\OCR_Project_Image_TO_Text.docx"
    document.save(pathFolderWord)

"""
A feature that inserts into the folder that the user selected a txt file along
with the translation of the text that was in the image.
"""
def translateImageText(final_string,pathFolder):
    translator = google_translator()
    translate_text = translator.translate(final_string, lang_tgt='he')
    pathFolderText = pathFolder+"\\Translate_OCR_Project_Image_TO_Text.txt"
    # open the text file in read mode
    f = open(pathFolderText, "w")
    f.write("Translation of the text in the image from English to Hebrew: "+
translate_text)
    f.close()

"""
A feature that inserts into the folder that the user selected a pdf file along
with the text identified in the image.
"""
def convert_to_pdf(pathFolder):
    pdf = FPDF()
    # Add a page
    pdf.add_page()
    # set style and size of font
```

```
    # that you want in the pdf
    pdf.set_font("Arial", size=15)
    pathFolderText = pathFolder+"\\OCR_Project_Image_TO_Text.txt"
    # open the text file in read mode
    f = open(pathFolderText, "r")
    # insert the texts in pdf
    for x in f:
        pdf.cell(200, 10, txt=x, ln=1, align='C')
    pathFolderPdf = pathFolder + "\\OCR_Project_Image_TO_Text.pdf"
    # save the pdf with name .pdf
    pdf.output(pathFolderPdf)
```

רפלקציה:

הפרויקט שלי הוא אפליקציית סריקת תמונה לטקסט (OCR) שמאפשרת למשתמש לעלות תמונה עם טקסט באנגלית לאפלקציה, משם התמונה נשלחת אל השרת ובצד השרת ממירה את הטקסט בתמונה לטקסט, ויוצרת בנתיב שנבחר על ידי המשתמש תיקייה שכוללת בתוכה:

אם מדובר בזיהוי תו אחד בודד:

1. קובץ txt עם סיווג התו.
2. קובץ txt שמכיל בתוכו את כל סיווגי התווים האחרונים של המשתמש.

אם מדובר בזיהוי טקסט בתמונה:

1. קובץ txt עם סיווג הטקסט שנמצא בתמונה.
2. קובץ txt שמכיל בתוכו את התרגום של הטקסט מאנגלית לעברית.
3. קובץ pdf שמכיל בתוכו את סיווג הטקסט בתמונה.
4. קובץ word שמכיל בתוכו את סיווג הטקסט בתמונה.
5. תמונה שמראה את רקע התמונה.

הדבר הכי חשוב שלמדתי בפרויקט הוא קודם כל בפן האישי לא לוותר, לעשות הכל כדי שיהיה תוצר ולעמוד בזמנים גם כשעמוס. בפן המקצועי, למדתי לחקור בצורה מיטבית באינטרנט, לחקור וללמוד דברים חדשים שלא נגעתי בהם בעבר, ולדעת להבין ולהתנסות ולעבוד איתם.
החלק הכי מהנה בפרויקט היה בסיום כל ספרינט שראיתי את התוצר הסופי של כל ספרינט, ולהתעסק בבניית מודל, נושא שלא התעסקתי בו כלל. להתעמק בעיבוד תמונה ולשפר את יכולותיי בפייתון וב*react*.
החלק הכי מאתגר בפרויקט היה להתמודד עם פערים, לדעת להבין שיש בעיה ולעשות הכל כדי לפתור אותה. בנוסף, למדתי כיצד לחפש באינטרנט עד שאני מוצא את הפתרון לבעיה ולצלוח אותה בסוף, או במידה ולא למצוא חלופות. האתגר המקצועי הכי קשה שהיה לי בפרויקט הוא לעסוק בreact native משום שלא עסקתי בו בעבר, לא השתמשתי כלל בשפת java script כך שהייתי צריך ללמוד מהבסיס, לכן כדי לפתור את זה הייתי צריך לחקור וללמוד אותו מההתחלה. בסופו של דבר, אני מרגיש שהתממשקתי עם react ולמדתי עליו ויודע להשתמש בו , ואני שמח על כך כי react מאוד נפוץ בתעשייה.
לסיכום, לדעתי הפרויקט היה מהנה, גם כשהוא היה כולו כמעט דרך מחשב בתקופה מורכבת זו וגם לימוד והעשרת הידע בנושא. אני מרוצה מהפרויקט שעשיתי כי אני חושב שבאמת והשקעתי ונתנו את כל כולנו בשבילו. הפרויקט תרם לי מאוד ברמה האישית ונהנתי לעשות זאת.

ביבליוגרפיה:

1. לצורך פיתוח האפליקציה :

https://realpython.com/mobile-app-kivy-python/

2. לצורך הפרויקט עצמו :

https://www.pyimagesearch.com/2020/08/17/ocr-with-keras-tensorflow-and-deep-learning/

https://www.pyimagesearch.com/2020/08/24/ocr-handwriting-recognition-with-opencv-keras-and-tensorflow/

3. לצורך עיבוד התמונה :

https://www.learnopencv.com/image-alignment-feature-based-using-opencv-c-python/

4. לצורך למידת המכונה :

https://reshetech.co.il/machine-learning-in-the-browser/how-to-develop-hand-written-digits-recognition-web-app

https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5