

תיק פרויקט הגנת סייבר



מגיש: יובל מנדל (216496349( תיכון הרצוג כפר סבא יב'3

שם פרויקט: DigitNet

שם מנחה: אופיר שביט

שם חלופה: הגנת סייבר ומערכות הפעלה

תאריך הגשה:24/05/2025

תוכן עניינים

[מבוא 4](#_Toc197381609)

[ייזום 4](#_Toc197381610)

[תיאור כללי של המערכת 4](#_Toc197381611)

[הגדרת הלקוח 4](#_Toc197381612)

[מטרות ויעדים 5](#_Toc197381613)

[בעיות תועלות וחסכונות 5](#_Toc197381614)

[סקירת פתרונות קיימים 6](#_Toc197381615)

[טכנולוגיה 7](#_Toc197381616)

[תיחום 7](#_Toc197381617)

[פירוט תיאור המערכת 9](#_Toc197381618)

[תיאור מפורט והיכולותך שיש לכל רכיב 9](#_Toc197381619)

[פירוט הבדיקות (קופסא שחורה) 10](#_Toc197381620)

[לוח זמנים 12](#_Toc197381621)

[ניהול הסיכונים ודרכי התמודדות 13](#_Toc197381622)

[תיאור תחום הידע 14](#_Toc197381623)

[פירוט מעמיק של היכולות 14](#_Toc197381624)

[יכולות צד שרת 14](#_Toc197381625)

[יכולות צד לקוח 16](#_Toc197381626)

[מבנה הפרויקט 19](#_Toc197381627)

[תיאור הארכיטקטוקרה 19](#_Toc197381628)

[תיאור הטכנולוגיה הרלוונטית 19](#_Toc197381629)

[תיאור זרימת המידע במערכת 20](#_Toc197381630)

[תיאור האלגוריתמים המרכזיים בפרויקט 24](#_Toc197381631)

[פתרונות אפשריים 24](#_Toc197381632)

[הפתרון שלי – CNN 24](#_Toc197381633)

[תיאור סביבת הפיתוח 25](#_Toc197381634)

[תיאור הפרוטוקול 27](#_Toc197381635)

[תיאור מסכי המערכת 31](#_Toc197381636)

[מסכי המערכת 31](#_Toc197381637)

[תרשים מסכים 35](#_Toc197381638)

[מבני נתונים 36](#_Toc197381639)

[מסד נתונים 36](#_Toc197381640)

[תור משימות בלקוח 36](#_Toc197381641)

[הורדת תמונות צד לקוח 37](#_Toc197381642)

[סקירת חולשות ואיומים 37](#_Toc197381643)

[מימוש הפרויקט 40](#_Toc197381644)

[סקירת המודולים 40](#_Toc197381645)

[מודולים חיצוניים 40](#_Toc197381646)

[מודולים פנימיים: 41](#_Toc197381647)

[הסבר אלגורתמים 54](#_Toc197381648)

[Backward של שכבת קונבלוציה: 54](#_Toc197381649)

[מסמך בדיקות 57](#_Toc197381650)

[הבדיקות שתכננתי בשלב האפיון 57](#_Toc197381651)

[הבדיקות נוספות שביצעתי 58](#_Toc197381652)

[מדריך למשתמש 59](#_Toc197381653)

[כלל קבצי המערכת 59](#_Toc197381654)

[התקנת המערכת 62](#_Toc197381655)

[פירוט הסביבה הנדרשת 62](#_Toc197381656)

[מיקומי הקבצים 62](#_Toc197381657)

[משתמשי המערכת 62](#_Toc197381658)

[רפלקציה 63](#_Toc197381659)

[בבליוגרפיה 64](#_Toc197381660)

[נספחים 65](#_Toc197381661)

# מבוא

## ייזום

### תיאור כללי של המערכת

בפרויקט זה הינו מערכת שיכולה לזהות ספרות (0–9) מתוך תמונות, תוך שימוש בטכניקות של למידת מכונה, ובעיקר רשת נוירונים קונבולוציונית (CNN).

רשת הנוירונים הקונבולוציונית תמומש מבלי ספריות כמו pytorch ו tensorflow אלא רק באמצעות numpy.

המערכת תכלול ממשק גרפי פשוט ונוח, שבו משתמשים יוכלו להעלות תמונות שכוללות ספרות, ולקבל את תוצאת הזיהוי בצורה של טקסט. בנוסף, תהיה אפשרות לצפות בתמונות שהועלו על ידי משתמשים רשומים, לראות איזו ספרה זוהתה בכל תמונה , להוריד את התמונות ששמורות בשרת ולהירשם/להתחבר (login/sign up).

התמונות והמידע הנלווה (כמו תוצאת הזיהוי) שנשלחו על ידי משתמשים שעברו התחברות (login) יישמרו במסד נתונים, כך שיהיה אפשר לשמור היסטוריה של הפעולות שהתבצעו במערכת. המערכת תשלב בין צד לקוח (GUI) שבו המשתמש פועל, לבין צד שרת שמטפל באימון המודל, בביצוע הזיהוי בפועל, ובניהול מסד הנתונים.

בחרתי בפרויקט הזה כי תחום הבינה המלאכותית מעניין אותי מאוד, במיוחד נושא של רשתות נוירונים ולמידה עמוקה. אני רוצה להבין איך רשת נוירונים מצליחה ללמוד לזהות תבניות מתוך מידע חזותי, ואיך אפשר ליישם את זה בפועל על בעיה אמיתית כמו זיהוי ספרות מתמונה. בנוסף, אני רואה בפרויקט הזה הזדמנות לשלב בין תיאוריה ומימוש מעשי, וללמוד על כל התהליך – החל מאיסוף הנתונים והכנתם, דרך אימון המודל, ועד לשילובו במערכת שלמה שפועלת מול משתמשים.

האתגרים שאני צופה לי בפרויקט הם אימון מדויק של מודל הבינה המלאכותית כך שהזיהוי יהיה אמין והבנה ומימוש של של רשתות נוירונים קונבולוציונית.

### הגדרת הלקוח

המערכת מיועדת לכל אדם שמעוניין לזהות ספרות מתוך תמונות בצורה מהירה, פשוטה ונגישה. היא מתאימה לשימושים חינוכיים, ניסיוניים או יישומיים — למשל לצורך זיהוי של ספרות שנכתבו ביד. קהל היעד כולל תלמידים, מורים, חוקרים ומפתחים המעוניינים לבדוק מודלים של למידת מכונה או להשתמש בתוצרי הזיהוי לצורכי ניסוי ולמידה.

המערכת מאפשרת למשתמשים להעלות תמונות, לקבל את תוצאת הזיהוי באופן מיידי, ולצפות בהיסטוריית הזיהויים שבוצעו בעבר. הממשק תוכנן להיות נוח לשימוש גם עבור מי שאין לו רקע טכני, והמערכת שמה דגש על חוויית משתמש פשוטה וברורה.

יתרון מרכזי של המערכת הוא באפשרות לאגור את התמונות שזוהו יחד עם תוצאות הזיהוי. מאגר זה משמש לא רק לתיעוד, אלא גם ככלי עזר ללמידת מכונה — ניתן לעשות בו שימוש חוזר לצורך אימון ושיפור של מודלים אחרים, מה שהופך את המערכת לכלי עבור תהליכים מתקדמים של ניתוח ולמידה.

### מטרות ויעדים

המטרות שלי הן לפתח מערכת שמסוגלת לזהות ספרות מתוך תמונות באופן אוטומטי בעזרת למידת מכונה. לממש ולהבין רשת נוירונים קונבולוציונית. לצבור ידע וניסיון מעשי בתחום של בינה מלאכותית וזיהוי תמונה.

### בעיות תועלות וחסכונות

הפרויקט עוסק בזיהוי אוטומטי של ספרות מתוך תמונות בעזרת בינה מלאכותית. מטרת המערכת היא לבצע את הזיהוי בצורה מהירה, מדויקת ויעילה, גם בסביבות עתירות נתונים ובתנאי קלט לא אחידים — כגון תמונות באיכות משתנה, עם רעש חזותי או כתב יד לא אחיד.

בנוסף לפונקציית הזיהוי הבסיסית, המערכת כוללת רכיב חשוב של אגירת מידע: כל תמונה שעוברת זיהוי נשמרת באופן שיטתי במאגר פנימי יחד עם תוצאת הסיווג ופרטים נוספים. המאגר הזה אינו רק אמצעי שמירה, אלא משמש תשתית ללמידה מתקדמת — ניתן לאמן על בסיסו מודלים חדשים או לשפר את המודל הקיים באמצעות הנתונים שהמערכת עצמה צברה.

באופן זה, המערכת לא רק מזהה ספרות מתמונה, אלא גם בונה בסיס נתונים שיכול לעזור באימון של מודלים אחרים בעתיד.

### סקירת פתרונות קיימים

הפרויקט שלי מתמקד בלפתור 2 בעיות, זיהוי ספרה ובניית מאגר תמונות יש להן שיוך למספר המתאים.

יש הרבה מוצרים שמסוגלים לזהות ספרות ואפילו מספרים מילים ומשפטים.

לדוגמה:  
<https://www.newocr.com/>

מערכת ocr שמסוגלת לזהות טקסט בתמונה (כולל ספרות)

האתר הזה נותן יותר יכולת מהפרויקט שלי (מסוגל טקסט בכללי).

|  |  |  |
| --- | --- | --- |
| תמונה | תוצאה של הפתרון הקיים | תוצאה שלי |
|  | 5 | 3 |
|  | SL | 7 |
|  | O) | 0 |

האתר אינטרנט נתן תוצאות לא נכונות.

הבדל נוסף בין הפתרון הזה לבין הפתרון שלי הוא שהפתרון הזה הוא עמוד web ובעוד שלי הוא שרת ולקוח עד פרוטוקול יחודי. בנוסף לכך האתר הזה אינו מאפשר להסתכל על תוצאות של אחרים.

הבעיה השנייה שאוצתה אני פותר היא מאגר מידע, המאגר מידע של ספרות הכי פופולרי הואMNIST שבו אני משתמש כדי לאמן את המודל שלי. בעוד שמאגר MNIST כולל תמונות סינתטיות בפורמט אחיד — ספרות ממורכזות, ללא רעש, בעובי וקונטרסט קבועים — המאגר שאני בונה מורכב מתמונות קלט אמיתיות כפי שהועלו על ידי המשתמשים. לכן, הוא מייצג בצורה נאמנה יותר אתגרים מהעולם האמיתי ויכול לשמש ככלי עזר לmnist באימון של מודלים אחרים. (יתרון של mnist הוא שהוא מקוטלג על ידי בני אדם ולכן הסיכוי שיהיה טעות בזיהוי הוא אפסי לעומת המאגר שלי)

### טכנולוגיה

הטכנולוגיה שעליה מבוסס הפרויקט אינה חדשה, היא עושה שימוש ברשתות קונבולוציה- סוג של אלגוריתם בלמידת מכונה שמיועד במיוחד להבנה של תמונות. הרשת "לומדת" לזהות דפוסים וצורות חוזרות בתמונה, כמו קווים, עיקולים או מבנה של ספרות, בדיוק כפי שעין אנושית שמה לב לפרטים. בזכות היכולת הזו, רשתות קונבולוציה מתאימות במיוחד למשימות של זיהוי חזותי.

מכיוון שאני מממש את רשת ה־CNN מהיסוד וללא שימוש בספריות מתקדמות או מודלים מוכנים מראש, תהליך האימון פועל גבי מעבד (CPU) רגיל ולא על כרטיס גרפי (GPU) מה שמגביל את היכולת לבצע אימונים כבדים או להשתמש במודלים גדולים. בנוסף, מאחר שאין ברשותי חומרה חזקה במיוחד, נדרש איזון בין איכות המודל לזמן הריצה והיעילות. שני הגורמים האלו מגבילים את היכולת שלי לאמן מודלים גדולים וחזקים ופוגעים באיכות הסופית של המוצר. כדי להתגבר על המכשולים האלו ניסיתי לממש את הרשת בצורה יעילה.

### תיחום

הפרויקט עוסק במגוון של תחומים טכנולוגיים:

* למידת מכונה: הפרויקט מתמקד בזיהוי ספרות מתוך תמונות באמצעות רשתות נוירונים קונבולוציוניות תוך מימוש עצמי של תהליך האימון והחישוב, ללא שימוש במודלים מוכנים מראש.
* רשתות תקשורת: המערכת כוללת תקשורת בין לקוח לשרת באמצעות פרוטוקול TCP (בשביל אמינות בהעברת המידע) ומממשת פרוטוקול ייחודי להעברת בקשות וקבצים בצורה אמינה ויעילה. התקשורת תומכת בהעברת מידע בינארי (תמונות) ובניהול תהליך שליחה וקבלה בשכבת האפליקציה.
* הצפנה ואבטחת מידע: כל התקשורת בין הלקוח לשרת מוצפנת באמצעות AES (הצפנה סימטרית), כאשר מפתח ההצפנה מועבר בצורה מאובטחת באמצעות אלגוריתם RSA (הצפנה אסימטרית). שילוב זה מאפשר הגנה מלאה על המידע לאורך כל שלבי ההעברה.
* מערכות הפעלה: הפרויקט כולל שימוש ב- Threads לניהול תהליכים מקבילים בשרת ובלקוח, וכן עבודה מול מערכת הקבצים (קריאה, כתיבה ושמירה של תמונות ונתונים), כחלק מהאינטגרציה עם מערכת ההפעלה.

הפרויקט אינו עוסק ביצירת מערכת הרשאות או ממשק משתמשים עם שם משתמש וסיסמה. אין בו מנגנון התחברות, שמירת סיסמאות או ניהול משתמשים, והוא לא כולל אימות זהות של הלקוח.

## פירוט תיאור המערכת

### תיאור מפורט והיכולותך שיש לכל רכיב

המערכת נועדה לזהות ספרות (0–9) מתוך תמונות המועלות על ידי המשתמשים, ולהציג את תוצאת הזיהוי בצורה נגישה, מהירה וברורה. היא מבוססת על רשת נוירונים קונבולוציונית (CNN), שזו שיטה מתקדמת בלמידת מכונה המתמחה בניתוח מידע חזותי. רשת זו מסוגלת לזהות תבניות חזותיות מורכבות מתוך תמונה, גם כאשר הקלט אינו אחיד — כלומר, גם כאשר הספרה אינה ממורכזת, הקווים דקים או עבים מהרגיל, קיימים רעשים בתמונה, או שהתמונה צולמה או נסרקה באיכות משתנה.

אחד היעדים המרכזיים של המערכת, מעבר לזיהוי המדויק עצמו, הוא בניית מאגר תמונות ותיוגים שנוצרו מתוך קלטים אמיתיים של משתמשים. מאגר זה מהווה תשתית חשובה לאימון עתידי של מודלים נוספים, כך שהמערכת לא רק מזהה אלא גם **לומדת ומשתפרת** לאורך זמן. בכך, נוצרת פלטפורמה שמבוססת על נתוני אמת ולא רק על מאגר סטטי כמו MNIST, ומאפשרת גמישות והתאמה לשימושים רחבים יותר.

המערכת מחולקת לשני חלקים עיקריים:

**1. צד לקוח:**

חלק זה של המערכת הוא הממשק הגרפי שמולו פועל המשתמש. הוא נועד להיות פשוט, ברור ונגיש גם למי שאין לו רקע טכנולוגי קודם. הממשק מאפשר למשתמש להעלות קבצי תמונה מהמחשב האישי או מכל מקור אחר. לאחר ההעלאה, התמונה נשלחת לשרת, ותוצאת הזיהוי (הספרה שזוהתה) מוצגת מיד בצורה ברורה.

בנוסף לכך, הממשק כולל אפשרות לעיין בתמונות קודמות שעברו זיהוי — כולל תמונות שהועלו על ידי משתמשים אחרים. כל תמונה מוצגת יחד עם הספרה שזוהתה בה, וניתן גם להוריד את התמונות למחשב המקומי. תכונה זו מאפשרת למשתמשים לצפות בנתונים שכבר עברו דרך המערכת, לבחון את איכות הזיהוי ולהשתמש בתמונות לצרכים לימודיים או ניסיוניים.

יכולות: בקשה מהשרת לזהות תמונה, העלאת תמונה,התחברות והרשמה למשתמש (log in / sign in) הצגת תוצאה של השרת, הצגת/הורדת תמונות ותוצאות שנאגרו בשרת.

**2. צד שרת:**

חלק זה אחראי על עיבוד הנתונים. השרת מקבל את התמונות שנשלחות מהלקוח, מפענח את המידע ומכין אותו לקראת ניתוח. לאחר מכן, התמונה מועברת למודל ה- CNN שמבצע את תהליך החיזוי כלומר, קובע איזו ספרה מופיעה בתמונה שהתקבלה.

התוצאה מוחזרת ללקוח לצורך הצגה בממשק, אך גם נשמרת בצד השרת לצורך תיעוד וניתוח עתידי. כל תמונה מזוהה, יחד עם תוצאת הזיהוי שלה ופרטי עזר נוספים, נשמרים במסד נתונים פנימי. בכך המערכת בונה לעצמה באופן שוטף מאגר של דוגמאות מתויגות – בסיס נתונים חשוב שיכול לשמש לאימון נוסף, סטטיסטיקות, או מחקר עתידי בתחום הזיהוי החזותי.

יכולות: זיהוי ספרה בתמונה, שמירת תוצאות ותמונות, תקשורת, שמירה של לקוחות ומתן שירות ללקוח

### פירוט הבדיקות (קופסא שחורה)

1. בדיקת העלאת תמונה תקינה

* מטרה: לבדוק האם המערכת מזהה תמונה שמכילה ספרה בודדת באופן תקין.
* אופן ביצוע: המשתמש יעלה תמונה איכותית, ברורה, של ספרה אחת (למשל, ספרה 3 כתובה בכתב יד ברור).
* תוצאה צפויה: המערכת תזהה את הספרה בצורה נכונה, והתוצאה תוצג בממשק ותישמר במסד הנתונים.

1. בדיקת העלאת תמונה באיכות נמוכה / עם רעש

* מטרה: לבדוק את עמידות המערכת לתמונות פחות איכותיות – לדוגמה, עם רעש, טשטוש קל או כתב יד לא ברור.
* אופן ביצוע: העלאת תמונה שצולמה מטלפון נייד בתאורה חלשה או עם רעש מלאכותי שנוסף לקובץ.
* תוצאה צפויה: המערכת תצליח לזהות את הספרה למרות הירידה באיכות.

1. בדיקת העלאת קובץ שאינו תמונה

* מטרה: לבדוק שהמערכת יודעת להתמודד עם קלט שגוי.
* אופן ביצוע: ניסיון להעלות קובץ מסוג PDF, קובץ טקסט או קובץ אקראי שאינו תמונה.
* תוצאה צפויה: המערכת תסרב לקבל את הקובץ ותציג הודעת שגיאה מתאימה (למשל: "סוג קובץ לא נתמך").

1. בדיקת תצוגת ההיסטוריה

* מטרה: לוודא שניתן לצפות בתמונות שזוהו בעבר יחד עם תוצאת הזיהוי שלהן.
* אופן ביצוע: לאחר מספר העלאות, המשתמש ילחץ על כפתור צפה בתוצאות קודמות.
* תוצאה צפויה: תוצג רשימה של תמונות קודמות עם תוצאת הזיהוי שלהן, כולל אפשרות להוריד כל אחת מהן.

1. בדיקת תגובתיות הממשק

* מטרה: לבדוק שהממשק הגרפי (GUI) מגיב בצורה חלקה לכל פעולה – העלאה, הצגה, מעבר בין כפתורים.
* אופן ביצוע: ביצוע רצף של פעולות משתמש – העלאה, מעבר להיסטוריה, חזרה למסך ראשי וכו'.
* תוצאה צפויה: כל פעולה תתבצע באופן מיידי ללא תקיעות או קריסות של הממשק.

1. בדיקת תקשורת עם השרת

* מטרה: לבדוק האם התקשורת בין הלקוח לשרת תקינה, והאם מתבצע תהליך שליחה וקבלה תקני.
* אופן ביצוע: ניתוק יזום של החיבור לרשת, או הרצת הלקוח כשהשרת כבוי.
* תוצאה צפויה: המערכת תציג הודעה על כשל בחיבור ולא תקרוס.

1. בדיקת אחסון במסד הנתונים

* מטרה: לבדוק שהמידע – תמונה, תוצאה וכו' – אכן נשמרים במאגר הנתונים לאחר כל זיהוי.
* אופן ביצוע: ביצוע זיהוי ולאחר מכן בדיקה (לוגית או תצוגתית) של הופעת הנתון החדש ברשימת ההיסטוריה.
* תוצאה צפויה: הנתון החדש יופיע עם כל הפרטים במיקום הנכון.

### לוח זמנים

|  |  |  |
| --- | --- | --- |
| משימה | תכנון | בפועל |
| חקר והבנה של תהליך למידת מכונה. להבין את השלבים בלמידה ובעיבוד של רשתות נוירונים Fully connected | 23.11.24-1.1.25 | 12.12.24-20.1.25 |
| מימוש של רשת Fully connected | 2.1.25-15.2.25 | 21.1.25 – 24.1.25 |
| להכין את השרת/לקוח(ממשק גרפי) + חיבור למסד נתונים | 20.2.25 – 3.4.25 | 20.2.25 – 3.4.25 |
| הבנה של רשת קונבולוציונית | 3.4.25 – 15.4.25 | 3.4.25 – 20.4.25 |
| מימוש של רשת קונבולוציונית | 15.4.25 – 30.4.25 | 20.4.25 – 2.5.25 |
| הוספה של פיצ'רים נוספים/שיפור של המערכת | 1.5.25-10.5.25 | 2.5.25 – 3.5.25 |

### ניהול הסיכונים ודרכי התמודדות

|  |  |  |
| --- | --- | --- |
| בעיה | דרך מוצעת להתמודדות | מה שנעשה בפועל |
| עיבוד תמונות ברזולוציות שונות | Resize לתמונה טרם העיבוד | Resize לתמונה טרם העיבוד |
| תפיסה של מקום גדול באחסון של השרת בגלל השמירה של תוצאות עבר | לשמור רק מספר קבוע של תמונות אחרונות | לשמור רק מספר קבוע של תמונות אחרונות |
| אי זיהוי תווים כתוצאה מודל לא מאומן מספיק | הרחבה של סט הנתונים | שינוי המבנה של המודל, אימון ארוך יותר והוספת מוטציות לתמונות על מנת לשפר generalization |
| קבצים גדולים מאוד עולים לקחת הרבה מקום באכסון של השרת | לשמור את התמונות אחרי resize | לשמור את התמונות אחרי resize |
| |  | | --- | | משתמש מעלה אותה תמונה מספר פעמים |  |  | | --- | |  | | מניעת כפילויות במסד הנתונים  על ידי hash של התוכן | מניעת כפילויות במסד הנתונים  על ידי hash של התוכן |
| קריסת הלקוח כשהשרת קורס | לסגור את הלקוח במקרה שהשרת קורס | להפריד בין הgui לתקשורת ובמקרה של התנתקות מהשרת לנסות להתחבר בחזרה כל כמה שניות |

# תיאור תחום הידע

## פירוט מעמיק של היכולות

### יכולות צד שרת

* 1. קבלת תמונות מהלקוח זיהוי התמונה ושליחת תוצאות בצורה מאובטחת-

לאפשר ללקוח לשלוח תמונה לשרת ולקבל תוצאת זיהוי – בצורה אמינה ומוצפנת.

פעולות/יכולות שנדרשות:

* הלקוח יתחבר לשרת באמצעות פרוטוקול TCP.
* שרת יקבל את התמונה (בבייטים).
* השרת יזהה את הספרה שבתמונה.
* לאחר זיהוי – שליחת התוצאה (ספרה + רמת ביטחון) חזרה ללקוח.
* כל זה יתבצע תוך הצפנה מאובטחת של המידע.

אובייקטים/מודולים נחוצים:

* TCP Socket
* הצפנה סימטרית (AES)
* החלפת מפתחות (RSA)
* פרוטוקול תקשורת אפליקטיבי
* encryptor/decryptor
* מודל לזיהוי ספרות מאומן
  1. עיבוד תמונה לזיהוי ספרה באמצעות מודל CNN-

עיבוד של תמונה שהתקבלה מהלקוח וביצוע זיהוי של הספרה שבתוכה באמצעות רשת נוירונים קונבולוציונית מאומנת.

**פעולות/יכולות שנדרשות:**

* המרת התמונה לפורמט אחיד (שחור-לבן, גודל קבוע, נירמול).
* המרת התמונה למערך מספרי שניתן להזין למודל.
* הפעלת המודל המאומן לקבלת תוצאה – הספרה שזוהתה.
* שמירה על מהירות תגובה סבירה לביצוע בזמן אמת.

**אובייקטים/מודולים נחוצים:**

* מודל CNN יעיל מאומן לזיהוי ספרות (לממש maxpooling, dense layer, conv layer ועוד)
* פונקציות preprocessing שמממירות את התמונה לפורמט מתאים ועושות שינוויים כדי לשפר את אחוזי ההצלחה בזיהוי
  1. שמירת תוצאות במסד נתונים-

שמירה של כל תוצאה של זיהוי במסד נתונים, כולל מידע כמו הספרה שזוהתה, רמת הביטחון, והקובץ עצמו, על מנת לאפשר שליפה עתידית ובקרה.

**פעולות/יכולות שנדרשות:**

* יצירת רשומה חדשה במסד הנתונים עבור כל תמונה.
* שמירה של מזהה ייחודי (UUID), תוצאה מזוהה, אחוז ביטחון, ונתיב לקובץ
* שמירה של קובץ התמונה עצמו במערכת הקבצים.
* הגבלת כמות התמונות השמורות (למשל ל־100 אחרונות בלבד).
* מניעת כפילויות באמצעות hash של תוכן התמונה.
* מחיקת קבצים ישנים שאינם קיימים יותר במסד הנתונים
* שליפה של תמונות לפי דרישות – לדוגמה לפי ספרה מסוימת.

**אובייקטים/מודולים נחוצים:**

* מסד נתונים (SQLite)
* מחלקה לניהול מסד הנתינים
* פונקציות לניהול קבצים ( read, open, write, delete)
* מחולל hash (SHA-256) למניעת כפילויות.
* שאילתות SQL לשליפה, עדכון ומחיקה.
  1. אימון מודל CNN לזיהוי ספרות-

לבנות ולאמן רשת נוירונים קונבולוציונית (CNN) מאפס, כך שתלמד לזהות ספרות מתוך תמונות באופן מדויק. האימון מתבצע על גבי מאגר נתונים הכולל תמונות מתויגות.

**פעולות/יכולות שנדרשות:**

* בניית ארכיטקטורה של רשת
* בניית כל החלקים שמרכיבים את הרשת (dense conv2d flatten maxpool)
* הגדרת פונקציית עלות (loss) מתאימה.
* ביצוע forward pass לקבלת חיזוי.
* חישוב backward pass (גרדיאנטים לפי השרשרת) לעדכון המשקלים.
* שימוש באלגוריתם אופטימיזציה (לדוגמה SGD)
* חלוקה ל־epochs ומעקב אחר שיפור באחוזי הדיוק.
* אפשרות לבצע augmentations (עיוותים אקראיים) בתמונות לשיפור היכולת להכליל.
* שמירה של המודל המאומן לשימוש עתידי.

**אובייקטים/מודולים נחוצים:**

* מחלקת CNN הכוללת את כל השכבות והפונקציות הדרושות.
* מחלקת optimizer
* נתונים לאימון (mnist)
* פונקציות serialization כדי לשמור את האימון שנעשה
* פונקציות שעושות עיוותים למידע המתקבל
  1. **ניהול משתמשים – הרשמה והתחברות**  
     הוספת מנגנון הרשמה והתחברות של משתמשים למערכת, לצורך התאמה אישית, הפרדה בין משתמשים, שמירה על פרטיות, ושיוך תמונות למשתמשים רשומים בלבד.

**פעולות/יכולות שנדרשות:**

* אימות זהות המשתמש עם התחברות מבוססת שם משתמש וסיסמה.
* אפשרות יצירת חשבון חדש במערכת (רישום).
* שמירת מידע משתמשים (כולל סיסמה מוצפנת).
* בדיקת תקינות סיסמה בעת התחברות.
* קישור תוצאות זיהוי למשתמש שנכנס למערכת.
* חסימת פעולות מסוימות (כמו צפייה בהיסטוריה) למשתמשים לא מחוברים.
* שמירה על session פעיל לאחר התחברות.

**אובייקטים/מודולים נחוצים:**

* מסד נתונים וטבלת users בSQLite
* פונקציית hash לסיסמאות (SHA-256 או bcrypt)
* מערכת session לשיוך לקוח מחובר
* שכבת הרשאות בסיסית לפעולות מוגנות

### יכולות צד לקוח

* 1. יצירת חיבור מאובטח לשרת-

התחברות לשרת בצורה מאובטחת לצורך שליחה וקבלה של מידע רגיש (תמונות ותוצאות זיהוי).

פעולות/יכולות שנדרשות:

* פתיחת חיבור TCP לשרת.
* קבלת מפתח ציבורי מהשרת לצורך הצפנה אסימטרית (RSA).
* יצירת מפתח AES אקראי.
* הצפנת המפתח הסימטרי באמצעות RSA ושליחתו לשרת.
* מעבר לתקשורת מוצפנת ב-AES לכל ההודעות הבאות.

אובייקטים/מודולים נחוצים:

* TCP Socket
* RSA Encryptor
* AES Encryptor/Decryptor
* ניהול session להצפנה
* ממשק תקשורת מול השרת
  1. העלאת תמונה לשרת לצורך זיהוי-

לאפשר למשתמש לבחור תמונה מהמחשב ולשלוח אותה לשרת בצורה פשוטה ומוצפנת.

פעולות/יכולות שנדרשות:

* הצגת כפתור לבחירת קובץ מהמחשב.
* קריאת קובץ התמונה וטעינתו לזיכרון.
* בנייה של ההודעה לפי הפרוטוקול
* שליחת ההודעה לשרת במבנה פרוטוקול מוגדר מראש.
* המתנה לקבלת תשובה מהשרת.

אובייקטים/מודולים נחוצים:

* File chooser
* מנגנון קריאת קבצים - read()
* AES Encryptor
  1. קבלת תוצאות מהשרת והצגתן למשתמש-

להציג למשתמש את הספרה שזוהתה בתמונה ואת אחוז הדיוק של המודל, באופן ברור ונוח.

פעולות/יכולות שנדרשות:

* קבלת תגובת שרת בפורמט מוגדר (מוצפן).
* פענוח
* פירוק תוכן ההודעה – ספרה + אחוז ביטחון.
* הצגת התוצאה על גבי הממשק.

אובייקטים/מודולים נחוצים:

* AES Decryptor
* Protocol parser
* רכיבי GUI להצגת טקסט
* לוגיקת ניתוח הודעות
  1. הצגת היסטוריית תמונות ותוצאות-

לאפשר למשתמש לצפות בתמונות קודמות שזוהו בעבר (את כולן וסינון לפי ספרות) כולל תוצאה מזוהה ואפשרות להורדה.

פעולות/יכולות שנדרשות:

* שליחת בקשה לשרת לקבלת היסטוריית תמונות.
* קבלת רשימת תמונות + תוצאות.
* הצגת התמונות בתצוגת גלריה.
* הצגת הספרה שזוהתה בכל תמונה.
* אפשרות להוריד את התמונות

אובייקטים/מודולים נחוצים:

* התמונות שהתקבלו מהבקשה
* מציד תמונות תמונות (image viewer)
  1. הורדת תמונות מהשרת-

לאפשר למשתמש להוריד למחשב האישי שלו את התמונות שנשלחו וזוהו בעבר.

פעולות/יכולות שנדרשות:

* הצגת כפתור "הורד" ליד כל תמונה בהיסטוריה.
* שליחת בקשה לשרת לקבל הקבצים וקבלתם.
* שמירה של הקובץ במחשב המקומי בפורמט מתאים.

אובייקטים/מודולים נחוצים:

* File dialog לבחירת תיקייה
* יוצר קבצי zip
  1. **התחברות והרשמה למשתמשים-**

לאפשר למשתמשים להירשם למערכת ולהיכנס עם שם משתמש וסיסמה לפני ביצוע פעולות.

**פעולות/יכולות שנדרשות:**

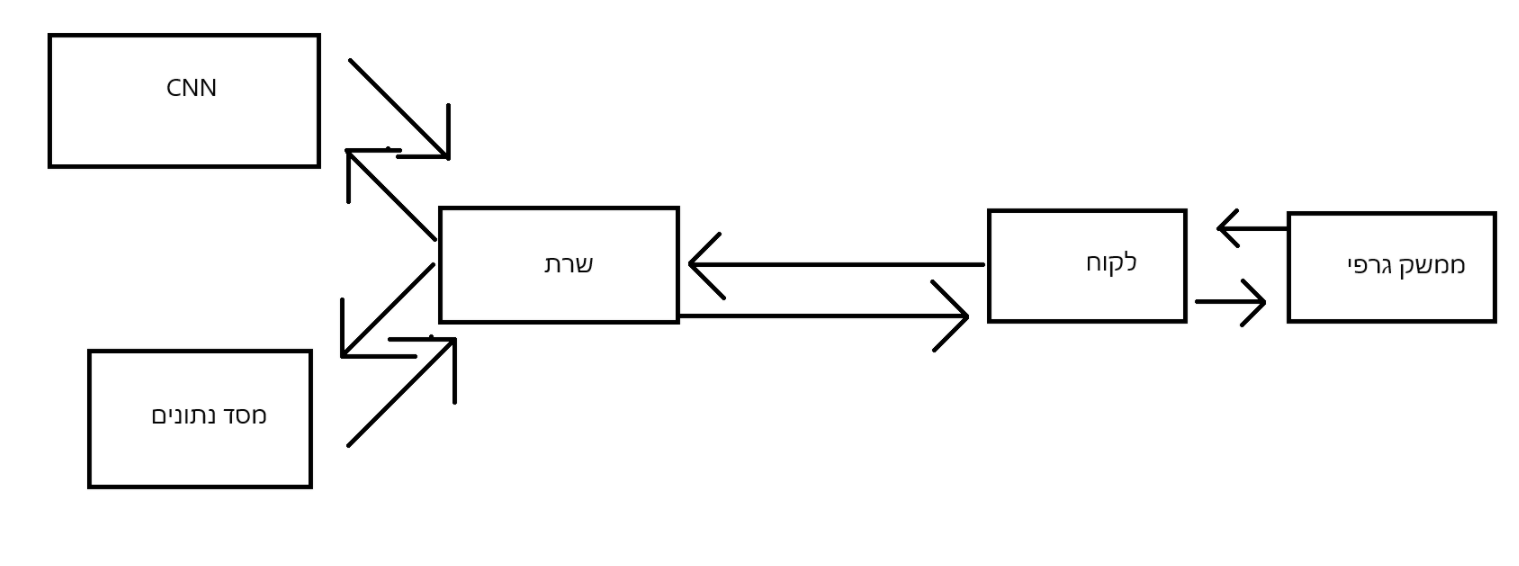
* מסך כניסה ומסך רישום עם שדות שם משתמש וסיסמה.
* שליחת בקשה לשרת לצורך התחברות או רישום.
* הצגת הודעות שגיאה (למשל משתמש כבר קיים, סיסמה שגויה וכו').
* מעקב אחר סטטוס התחברות והצגתו בממשק הראשי.
* הגבלת פעולות מסוימות (למשל שליחה/צפייה בהיסטוריה) למשתמשים מחוברים בלבד.

**אובייקטים/מודולים נחוצים:**

* רכיבי GUI לשדות קלט וכפתורים
* משתנה סטטוס התחברות (logged\_in\_user)
* הודעות סטטוס ועדכונים בממשק

# מבנה הפרויקט

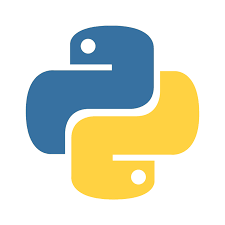
## תיאור הארכיטקטוקרה

לפרויקט 5 רכיבים עיקריים: ממשק גרפי, לקוח, שרת, רשת נוירונים קונבולוציונית ומסד נתונים

## תיאור הטכנולוגיה הרלוונטית

כל הקוד בפרויקט נכתב בשפת python (בפרט כדי לכתוב את ה- CNN השתמשתי בספריית Numpy).

בחרתי לכתוב את הפרויקט בשפת פייתון מכיוון שהיא שפה פשוטה, קריאה וגמישה, שמתאימה במיוחד לפרויקטים שמשלבים תקשורת. לפייתון יש קהילה רחבה וכלים מוכנים לעבודה עם תמונות ומערכים רב מימדיים בצורה יעילה (Numpy, PIL). היא מתאימה מאוד לפרויקטים לימודיים שמטרתם הבנה של הלוגיקה מאחורי המימוש – ולא רק שימוש בספריות מוכנות מראש.



השרת והלקוח רצים במערכת הפעלה windows 11.

התקשורת במערכת מנוהלת בפרוטוקול TCP בשביל אמינות של המידע. בנוסף התקשורת פועלת לפי פרוטוקול יעודי שהגדרתי למען הפרויקט.

הפרויקט עוסק בשילוב של תחומים טכנולוגיים מגוונים: למידת מכונה וזיהוי תבניות חזותיות בעזרת רשתות נוירונים קונבולוציוניות (CNN), הצפנה ואבטחת מידע (RSA, AES), ופיתוח מערכות מבוססות לקוח-שרת עם תקשורת אמינה בפרוטוקול TCP.

בנוסף, הוא כולל ניהול נתונים באמצעות מסד נתונים מקומי, עבודה עם קבצים ותמיכה בממשק גרפי אינטראקטיבי. תחומי עניין אלו מבטאים חיבור בין מדעי המחשב העיוניים לבין יישומים מעשיים של תוכנה ובינה מלאכותית.

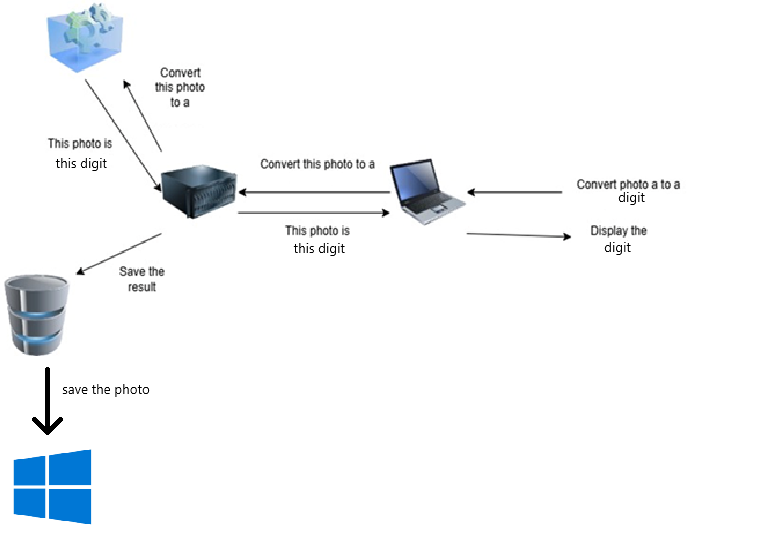
## תיאור זרימת המידע במערכת

\*חיצים בין הלקוח לשרת הם בקשות/תגובות ברשת בעוד שחצים אחרים הם מעבר של מידע בתוך process של השרת/לקוח

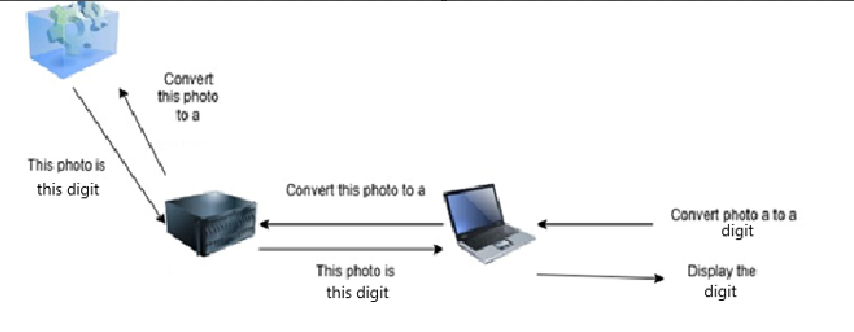
בקשה של הלקוח לפענח ספרה מסויימת:

המשתמש יזין בממשק הגרפי תמונה שהוא רוצה להמיר, הממשק הגרפי יודיע ללקוח לשלוח הודעה לשרת, הלקוח ישלח את התמונה לשרת לפיענוח, השרת ישתמש בCNN כדי לפענח את התמונה, אם המשתמש רשום ישמור במסד הנתונים את התוצאה (שישמור את התמונה מול מערכת ההפעלה) וישלח את התוצאה ללקוח שיציג את התוצאה בממשק הגרפי של המשתמש.

משתמש רשום:

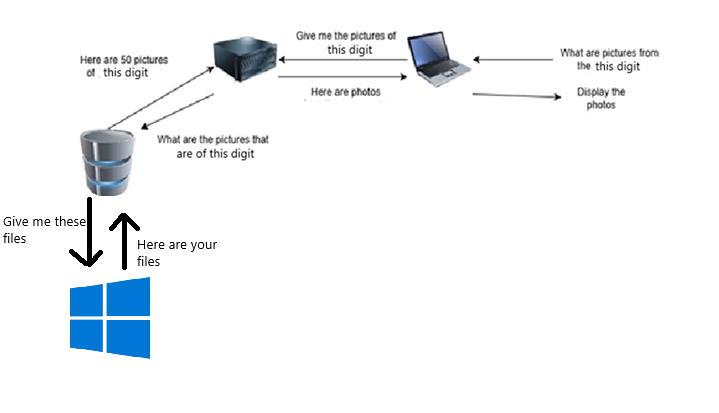


משתמש לא רשום:



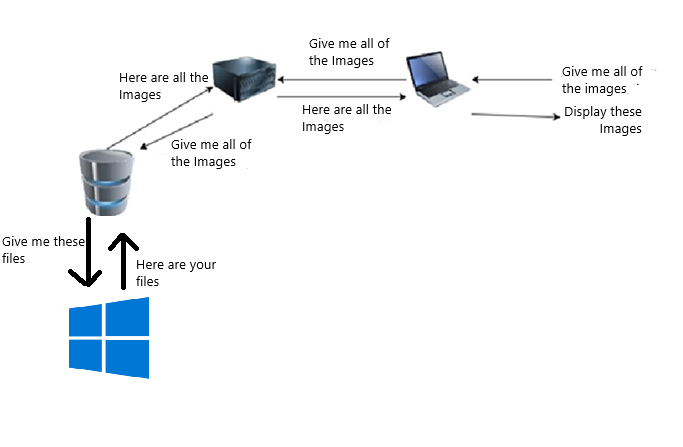
בקשה של הלקוח של תמונות של ספרה מסויימת:

המשתמש יבקש תוצאות קודמות של ספרה מסויימת דרך הממשק הגרפי, הממשק הגרפי יודיע ללקוח לשלוח הודעה לשרת, הלקוח יבקש מהשרת תשובות קודמות, השרת ישלוף ממסד הנתונים את התוצאות הקודמות, השרת ישלח את התוצאות ללקוח שיציג את התוצאות בממשק הגרפי.

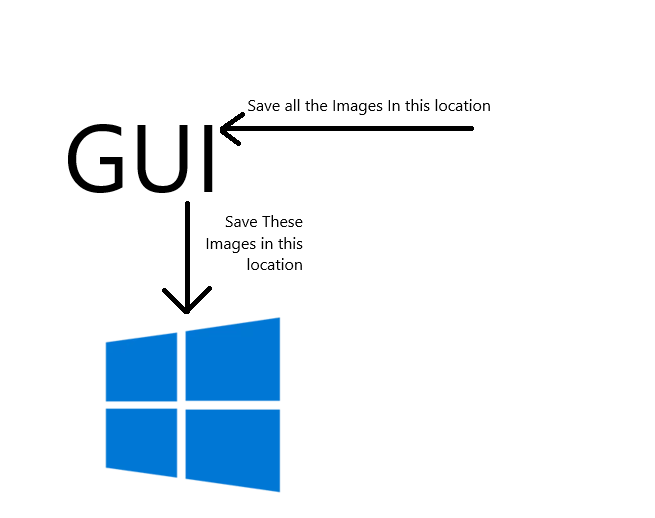


בקשה של הלקוח של כל התמונות שיש לשרת:

המשתמש יבקש את כל התמונות שיש לשרת דרך הממשק הגרפי, הממשק הגרפי יודיע ללקוח לשלוח הודעה לשרת, הלקוח יבקש מהשרת תשובות קודמות, השרת ישלוף ממסד הנתונים את התוצאות הקודמות, השרת ישלח את התוצאות ללקוח שיציג את התוצאות בממשק הגרפי.

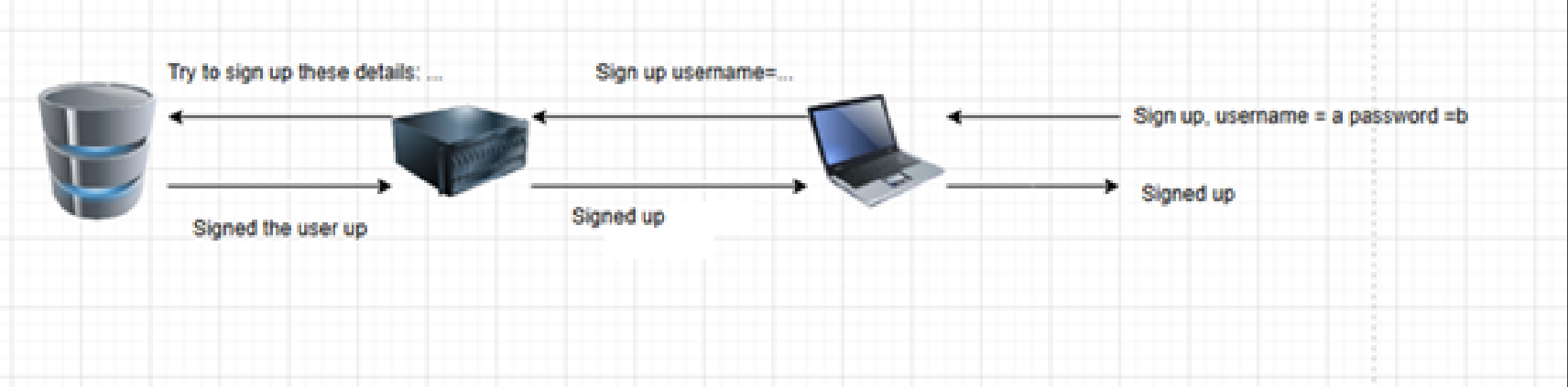


המשתמש רוצה להוריד את כל הקבצים שנשלחו לו (יכול לקרות רק אחרי שקיבל תמונות (בשני הדרכים שצויינו לעיל)



רישום של משתמש sign up-

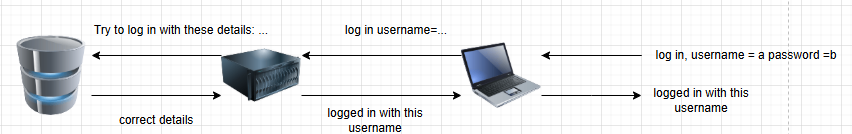
המשתמש מכניס פרטים של משתמש, שולח לשרת בקשה, אם השם משתמש פנוי השרת שומר את המשתמש (אם לא אז הוא שולח ללקוח שההרשמה נכשלה) אם ההרשמה הושלמה אז השרת שולח שההרשמה נעשתה בהצלחה ומחזיר את השם משתמש.

A computer next to a graph

AI-generated content may be incorrect.

התחברות של משתמש sign in-

המשתמש מכניס פרטים שולח בקשה להתחבר. אם הנתונים נכונים אז השרת מאשר שהלקוח נכנס אם לא שולח הודעה שהנתונים לא נכונים.

A close up of a blue object

AI-generated content may be incorrect.

## תיאור האלגוריתמים המרכזיים בפרויקט

זיהוי תמונות (Image Recognition) הוא תחום בלמידת מכונה ובראייה ממוחשבת, שמטרתו להבין את תוכן התמונה – לזהות עצמים, תבניות או מאפיינים בתמונה, ולהמיר מידע חזותי לייצוג סמנטי או מספרי. בפרויקט זה, המשימה ממוקדת בזיהוי **ספרות** (0–9) מתוך תמונות, לרוב בכתב יד, שהן בעלות גיוון גדול במראה: מיקום שונה בתמונה, סגנון כתיבה אישי, רעש רקע, עיוותים, שינויי תאורה ועוד.

לבני אדם, לזהות ספרה בתמונה זה דבר קל שבא בטבעיות — גם אם הספרה כתובה בעט, בגיר, בעובי שונה, בזווית, או אפילו עם טיפה רעש ברקע. מערכת הראייה האנושית יודעת להכליל, להשלים פרטים חסרים ולהשתמש בהקשר כדי להבין במהירות מה מופיע בתמונה. לעומת זאת, עבור מחשב, משימה זו מורכבת בהרבה: הוא מקבל מערך של פיקסלים מספריים חסרי משמעות סמנטית, וצריך ללמוד כיצד תכונות מקומיות בתמונה מרמזות על ספרה שלמה.

### פתרונות אפשריים

לבעיה הזו כמה פתרונות אפשריים.

Random Forest-

Random Forestהוא אלגוריתם של למידת מכונה שמבוסס על ריבוי של עצי החלטה (Decision Trees) כל עץ מקבל החלטה עצמאית לגבי הסיווג, ו"היער (forest) "מצביע על התשובה הסופית לפי הרוב. העיקרון מאחורי השיטה הוא שכאשר משקללים את ההחלטות של הרבה עצים שונים, מקבלים תוצאה מדויקת ועמידה יותר לשגיאות.

מכיוון שתמונה היא מערך של פיקסלים לא כדאי להזין את הפיקסלים עצמם לRandom Forest – זה היה סובל מרעש ומורכבות גבוהה מדי. לכן, קודם כל מחולצים מהתמונה מאפיינים, שמספקים ייצוג קומפקטי של צורת הספרה: קווים, כיוונים, קימורים וכו'.

לאחר מכן, הפיצ'רים הללו מוזניםRandom Forest שמסווג את הקלט לאחת מ־10 הספרות האפשריות.

בעיה אחת בפתרון הזה היא שצריך לבחור את הפיצ'רים בעצמינו ואם הפיצ'רים לא טובים זה עלול לפגוע מאוד באיכות של המודל.

k-Nearest Neighbors –

שיטה פשוטה שמבוססת על השוואת תמונה חדשה לדוגמאות שכבר סומנו. כל תמונה נמדדת לפי “מרחק” לתמונות אחרות, והספרה הכי שכיחה מבין השכנים הקרובים היא הבחירה.

הבעיה בפתרון הזה היא שהוא איטי מאוד בזיהוי (חייב לעבור על כל הדוגמאות), רגיש לרעש, לא יעיל בזיכרון.

### הפתרון שלי – CNN

מה זה CNN?

CNN (Convolutional neural network)היא רשת נוירונים מיוחדת שנועדה לעבודה עם תמונות. הרשת מקבלת את תמונת הפיקסלים עצמה, ולומדת מתוכה לבד איך לזהות תבניות. היא עושה זאת באמצעות שכבות קונבולוציה – מסננים (filters) שמזהים קווים, עיקולים, גבולות ועוד, ושלבים של pooling להקטנת המידע תוך שימור המבנה החשוב.

איך CNN מזהה ספרות?

* התמונה עוברת עיבוד מקדים (נרמול שינוי גודל וכו')
* היא מוזנת לרשת CNN, שבה שכבות לומדות תכונות – החל מקווים פשוטים בשכבות הראשונות ועד למבנה הספרה השלמה בשכבות העמוקות.
* בסוף, שכבת Fully Connected מוציאה חיזוי: מהי הספרה בתמונה ומה מידת הביטחון בזיהוי.

למה זה מתאים?

* הרשת לומדת לבד את התכונות החשובות מתוך התמונות – בלי צורך בהנדסת פיצ'רים.
* היא יודעת לזהות גם תמונות עם רעש, סיבוב, שינוי גודל או כתב לא אחיד.
* היא מסוגלת להגיע לרמות דיוק גבוהות מאוד.

יתרונות:

* דיוק גבוה מאוד.
* עמידות לשינויים בתמונה.
* מתאים במיוחד לזיהוי תבניות חזותיות.

חסרונות:

* דורש הרבה יותר כוח חישוב (במיוחד בזמן אימון).
* קשה יותר להבנה ולהסבר פנימי של תהליך ההחלטה.
* דורש מאגר נתונים איכותי ומגוון כדי להכליל היטב.

מקורות שעזרו לי ללמוד ולחקור על רשתות נוירונים:

<http://neuralnetworksanddeeplearning.com/chap1.html>

<https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi>

### תיאור סביבת הפיתוח

כלי הפיתוח הנדרשים

הפרויקט נבנה כולו בשפת Python, תוך שימוש ב IDE (PyCharm)

לצורך פיתוח המודל והמערכת נדרשו הכלים הבאים:

* שפת פיתוח: Python 3.9
* ספריות לעיבוד תמונה: PIL (Pillow), io, os
* ספריות חישוביות: NumPy
* ספריות להצפנה: Crypto.Cipher.AES, Crypto.Cipher.PublicKey.RSA
* עבודה עם מסד נתונים: sqltie3
* GUI: tkinter
* תקשורת: threading

**כלים וסביבה הדרושים לבדיקה:**

* מערכת הפעלה: Windows 11
* סביבת הרצה: Interpreter של Python 3.9
* הבדיקות כללו בדיקות תקשורת מקצה לקצה (Client ↔ Server), בדיקות תקינות הצפנה, בדיקת תגובות לזיהוי שגוי ובדיקות GUI.

### תיאור הפרוטוקול

מטרת הפרוטוקול היא לאפשר שירות זיהוי תמונות. הפרוטוקול המידע שעובר צריך להיות מוצפן והפרוטוקול צריך לאפשר מעבר של קבצים למען בקשות של הלקוח ולמען תגובות של השרת.

מאפייני הפרוטוקול:

* טקסטואלי (Text-Based Protocol) – הנתונים עצמם מקודדים ב־Base64 ומופרדים באמצעות תו מפריד “~” .
* מבוסס TCP – תקשורת אמינה וסדר הודעות שמור.
* סינכרוני – הלקוח שולח בקשה וממתין לתשובה.
* מוצפן לחלוטין לאחר handshake
* שדה אורך בתחילת כל הודעה (7 תווים לא מוצפנים), כדי לאפשר קריאה רציפה של ההודעה, גם עבור קבצים גדולים.
* כדי להימנע מפתיחה של מספר קבצים בו זמנית על ידי אותו הthread בשרת כאשר השרת שולח תמונות ללקוח הוא ישלח הודעה מראש שאומרת שהוא הולך לשלוח הודעות של תמונות וכמה מהן הוא מתכנן לשלוח, אז ישלח מספר הודעות עם התמונות וכשיסיים ישלח הודעה שהוא סיים. (נשלח הודעת סיום **וגם** כמות ההודעות שישלחו מראש כדי להימנע ממצב שבו השרת לא מוצא תמונה ואז הלקוח נתקע בrecv)
* כל ההודעות נשלחות בצ'אנקים של עד 4K (הודעה יכולה להיות יותר גדולה מ4K אך פשוט תפורק למספר צאנקים המקבל ידע מתי להפסיק לעשות recv לפי שדה האורך)

\*סכנה תאורטית אפשרית היא אם מישהוא רוצה לשנות את 4K למספר קטן יותר מאורך שדה האורך, ואז לא כל שדה האורך נשלח בצ'אנק הראשון מה שיגרום לבעיה.

תהליך ה-handshake:

1. השרת שולח ללקוח את המפתח הציבורי שלו (RSA – PEM)
2. הלקוח יוצר מפתח AES ו־IV אקראיים, מצפין אתהמפתח AES עם המפתח הציבורי של השרת ושולח אותם אותם לשרת (2 הודעות נפרדות)
3. השרת מפענח, שומר את המפתח ואת הIV, ושולח הודעת ACK (GKSC) מוצפנת באמצעות AES.
4. מכאן והלאה, כל ההודעות מוצפנות ב־AES במצב CBC.

מבנה הודעה כללי:

* שדה אורך: 7 תווים (לא מוצפן)
* גוף ההודעה: שדות BASE64 מופרדות באמצעות ~
* השדה הראשון יהיה קוד הודעה
* הפענוח מתבצע לאחר קבלת כל ההודעה המוצפנת.

סוגי בקשות – Client → Server:

RIPP- בקשה לזיהוי ספרה מתוך תמונה

פרמטרים:

1. שם הקובץ (מחרוזת)
2. תוכן התמונה (בייטים)

הגבלות:

* גודל מקסימלי של קובץ: 3,000,000 בתים (3MB)
* פורמט תמונה חייב להיות חוקי ונתמך (בדיקה עם PIL.Image.verify())
* התמונה יכולה להיות צבעונית או בשחור-לבן – המרה תתבצע בצד שרת.

RIHP- בקשה לקבל את כל התמונות

פרמטרים: אין

RIHD- בקשת שליפת תמונות לפי ספרה מסוימת

פרמטרים:

1. ספרה אחת (0–9) במחרוזת

CRSU- בקשה להירשם sign up

פרמטרים:

* 1. שם משתמש
  2. סיסמה

CRSI- בקשה להיכנס login

פרמטרים:

1. שם משתמש
2. סיסמה

תגובות – Server → Client:

RIPR- תגובת זיהוי לתמונה

פרמטרים:

1. ספרה מזוהה (0–9)
2. אחוז ביטחון (Confidence) – מספר עשרוני בין 0 ל־1 (טקסט)

RIHL- תגובה לבקשה של תמונות, מתחיל את תהליך העברת הקבצים. לאחר ההודעה הזו השרת שולח הודעות מסוג RILF עד שמסיים את התמונות ואז שולח RIHE

פרמטרים:

* מספר התמונות המשוער (אם אחת התמונות לא נמצאה אז פשוט ישלחו פחות הודעות RIHE בכל מקרה הלקוח מפסיק כאשר מקבל RIHE)

RILF- הודעה ששולחת הודעה אחת ללקוח (הודעה זו נשלחת בשרשרת ולאחר השרשרת תמיד תופיע RIHE)

פרמטרים:

1. מזהה תמונה (UUID)
2. תוכן התמונה (בייטים בפורמט PNG)
3. ספרה מזוהה
4. אחוז ביטחון

RIHE- הודעה שמסמנת את סוף שליחת התמונות מהשרת

פרמטרים- אין

CRSA- תוגבה חיובית לבקשה להירשם (sign up)

פרמטרים: אין

CRSD – תגובה שלילית לבקשה להירשם (sign up)

פרמטרים: אין

CRLA – תגובה חיובית לבקשה להתחבר (login)

פרמטרים:

* שם המתמש שאיתו נרשמו

CRFD- תגובה שלילית לבקשה להתחבר (login)

פרמטרים: אין

ERRR- שגיאה

פרמטרים:

1. קוד שגיאה (מחרוזת):
   * 1- קובץ אינו תמונה חוקית או בפורמט לא נתמך
   * 2- קובץ גדול מדי
   * 3 - בקשה לא תקינה
   * 4 - שגיאה כללית

GKSC- אישור של השרת שהוא מוכן לתקשורת (אחרי handshake), אין פרמטרים

דוגמה לתקשורת מלאה:

1. הלקוח פותח חיבור TCP ומקבל את מפתח ה־RSA של השרת.
2. הלקוח שולח מפתח AES מוצפן ואת ה־IV.
3. השרת משיב ב GKSC
4. הלקוח שולח RIPP עם שם ותוכן תמונה.
5. השרת שולח RIPR עם ספרה ורמת ביטחון.
6. הלקוח שולח RIHD עם הספרה 3.
7. השרת משיב ב־RIHL
8. השרת שולח הודעות RILF כדי לשלוח את התמונות (הודעה עבור כל תמונה)
9. השרת שולח הודעת RIHE כדי לסמן שכל התמונות נשלחו

פרטים נוספים:

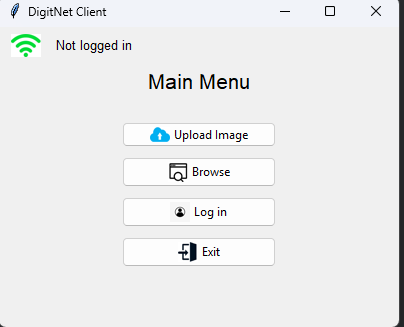
* כל ההודעות ב־AES מוצפנות במצב CBC עם padding.
* לכל לקוח Session הצפנה ייחודי משלו.
* המודל בצד השרת שומר את התמונה, מזהה ספרה, ומעדכן מסד נתונים (עד 100 תמונות).
* קבצים נשמרים בפורמט PNG, בגודל מרבי של 256\*256 ( עוברים resize)

## תיאור מסכי המערכת

### מסכי המערכת

מסך ראשי-

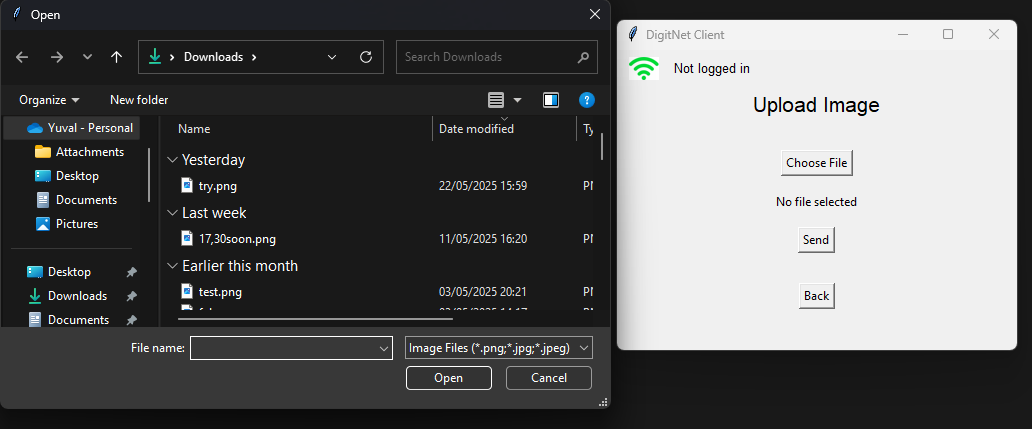
נותן לעבור למסככים האחרים (או לסגור את התוכנה), מוצג בפינה השמאלית למעלה את מצב החיבור (מחובר או מנותק מהשרת, מחובר למשתמש או לא)



מסך העלאת תמונות-

נותן את האפשרות להעלות תמונות ולשלוח אותן (מראה את השם של התמונה שכרגע בחרו אותה) וכפתור חזרה למסך הראשי, שליחה של תמונה תביא הודעה עם שגיאה או תגובה של השרת. A screenshot of a computer

AI-generated content may be incorrect.

 A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

מסך עיון –

מאפשר לבקש לצפות בתמונות (browse) ולבחור תמונות של איזה ספרה לבקש (או לקבל את כל התמונות)

(נפתח תפריט שאפשר לבחור ממנו ספרה אך לא מאפשר לצלם אותו)

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

מסך צפייה בתמונות-

מאפשר לראות את התמונות, מתחת לכל תמונה רשום (משמאל לימין) את חמשת האותיות הראשונות של הid (השם), את הספרה שהמודל עשה לו predict, והconfidence (0 עד 1)

מאפשר לחזור למסך הראשי

מאפשר להוריד את כל הקבצים שהתקבלו (מעלה בוחק מקום של מערכת ההפעלה) A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

מסך הרשמה-

מאפשר להירשם/להתחבר למשתמש על מנת שהתמונות שנשלחות מהלקוח ישמרו במסד הנתונים. מאפשר הכנה של הנתונים (שם משתמש וסיסמא) והרשמה/התחברות

A screenshot of a computer screen

AI-generated content may be incorrect.

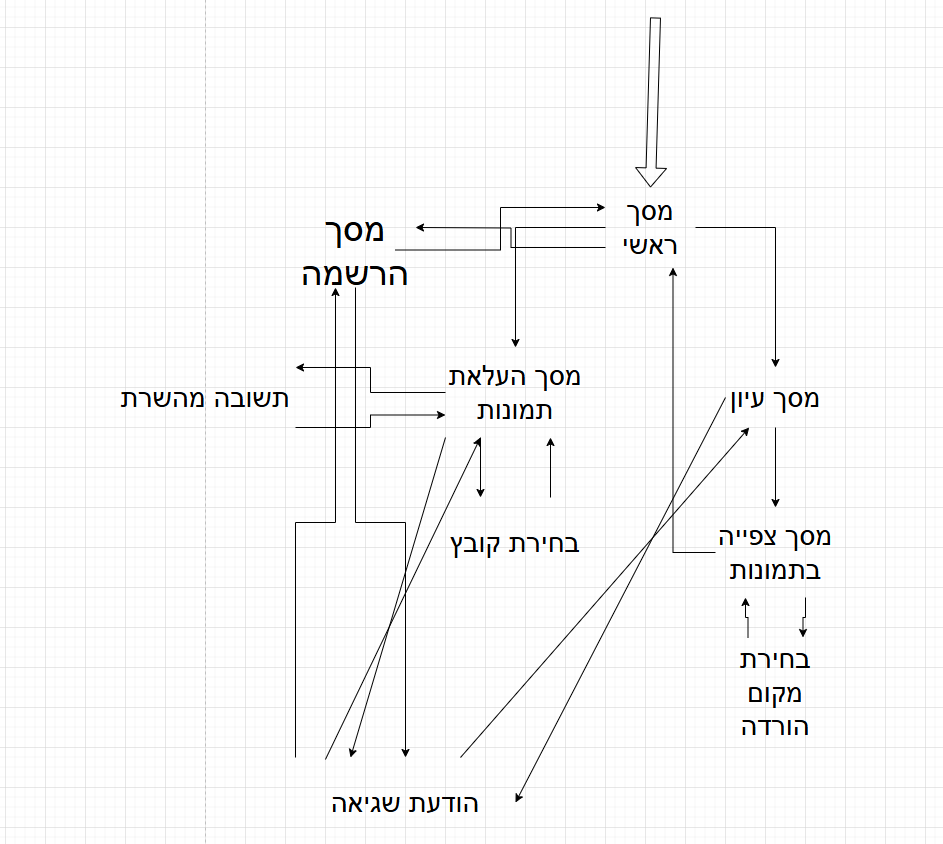
A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer login

AI-generated content may be incorrect.

## תרשים מסכים

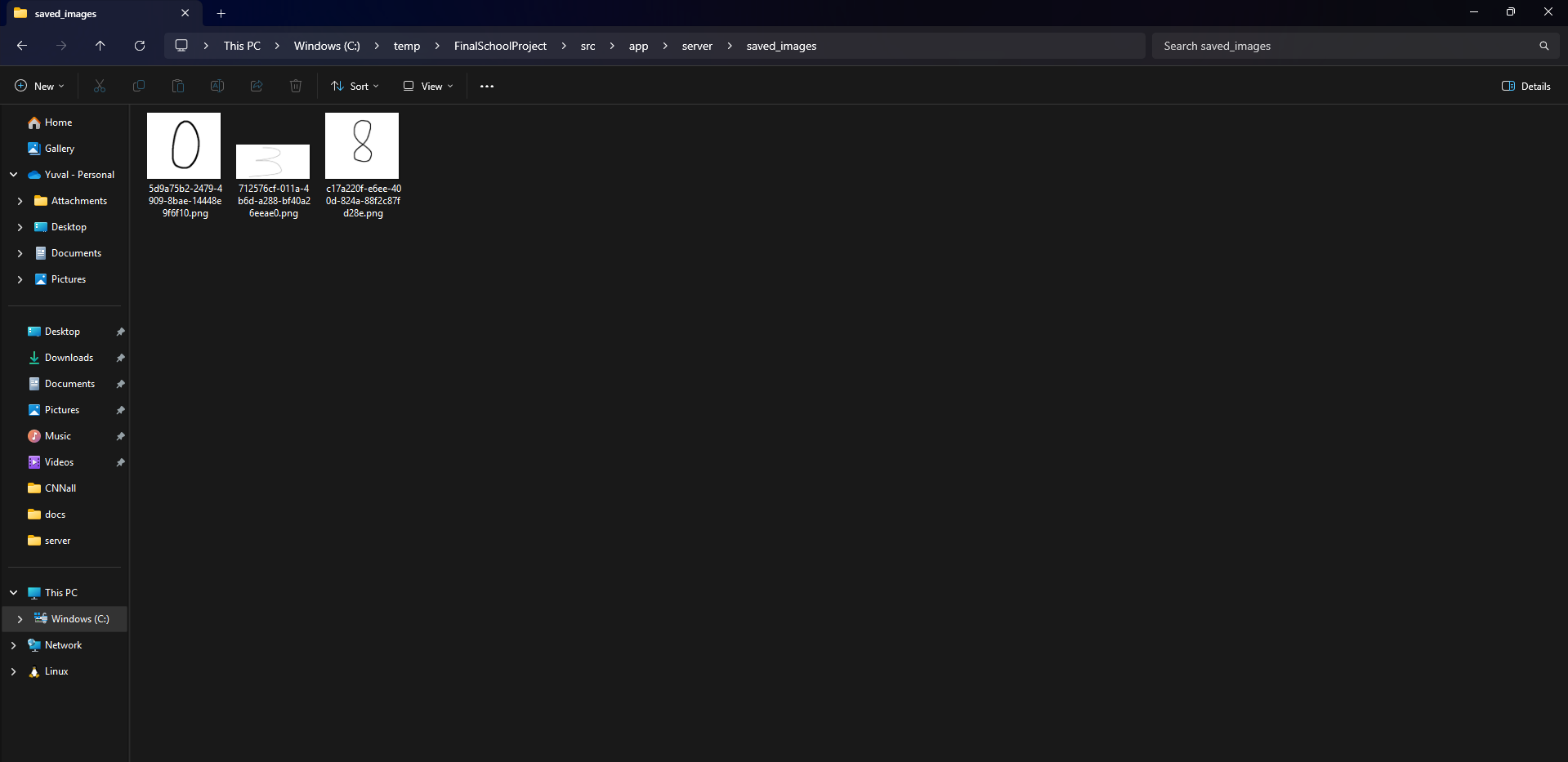
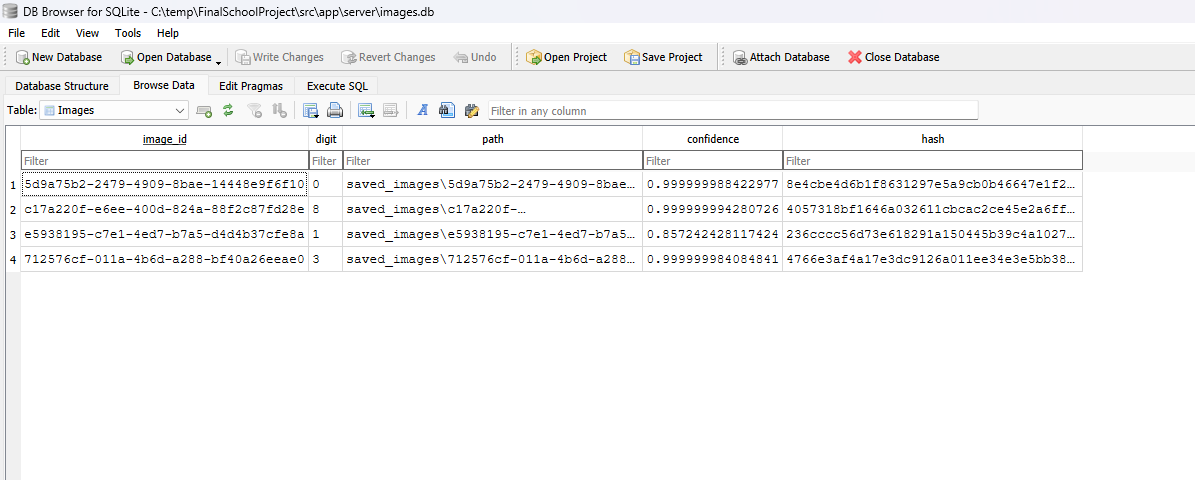


## מבני נתונים

### מסד נתונים

מסד הנתונים הוא SQLite עם טבלה אחת, הכוללת את השדות הבאים: מזהה תמונה ייחודי (image\_id TEXT PRIMARY\_KEY), הספרה שסווגה (digit TEXT), הנתיב לקובץ התמונה המקומי (path TEXT), רמת הביטחון של המודל (confidence REAL) וגיבוב מסוג SHA-256 של התמונה (hash TEXT UNIQUE), המאפשר לזהות תמונות כפולות. השדה hash מוגדר כייחודי כדי למנוע שמירה של אותה תמונה פעמיים, גם אם היא התקבלה ממקורות שונים.

התמונות עצמן שמורות בתת התקייה saved\_images עם הimage\_id בתור השם, בפרומט png, מוקטן לעד 256\*256



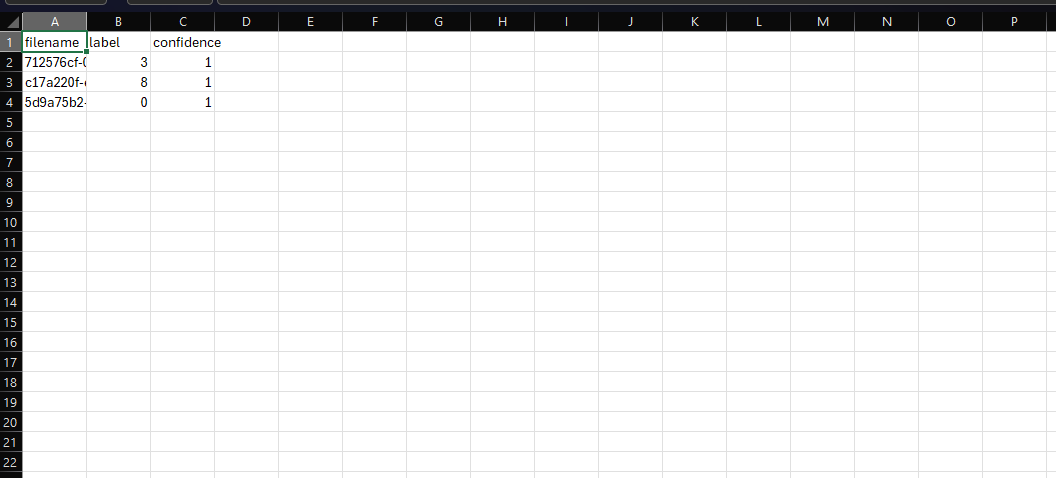
### תור משימות בלקוח

כדי לנהל את המשימות מהמשתמש הthread של הלקוח (מנהל את התקשורת) משתמש בתור, ללקוח יש פעולות שהgui משתמש בהם ששמות משימות בתור. אז הthread של הלקוח מקבל את המשימה בqueue, המשימה מורכבת משני דברים "קוד למשימה" (קוד שהוסכם מראש שיהיה הקוד של המשימה- לרוב יהיה פשוט הקוד הרלוונטי בפרוטוקול) והפרמטרים (arguments), אם המשימה היא None זה סימן של הgui שהאפליקציה נסגרת.

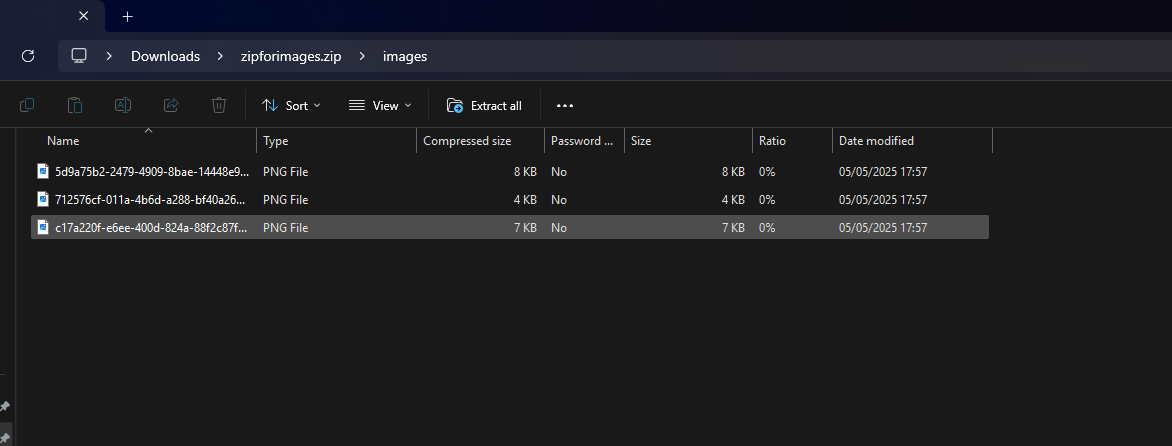
### הורדת תמונות צד לקוח

הלקוח מוריד zip של התמונות, בקובץ zip יש קובץ csv שמכיל את נתונים על הקבצים. בנוסף לכך יש תקייה בשם /images שמכילה את הקבצים עצמם.

קובץ הcsv:



תקייה images:



## סקירת חולשות ואיומים

SQL injection-

אחת מהפגיעויות הנפוצות והמסוכנות ביותר באפליקציות מבוססות־מסד נתונים היא SQL injection מדובר בטכניקת תקיפה שבה תוקף מזין קלט זדוני (למשל במקום ערך של שם משתמש או מזהה) שמכיל קוד SQL. כאשר הקלט לא מסונן כראוי ומוזן ישירות לבקשת SQL, תוקף יכול לשנות את משמעות השאילתה – למשל כדי לחשוף מידע רגיש, למחוק טבלאות, או לעקוף מנגנוני אימות.

בפרויקט שלי, אני עושה שימוש מודע ובטוח ב־SQLite באמצעות מודול 3sqlite של פייתון. כל השאילתות שמקבלות קלט מהמשתמש נכתבות באמצעות prepared statements (הצהרות מוכנות) – כלומר, אני משתמש ב־? בתור מציין מקום לערכים, ומספק את הנתונים כטופל נפרד.

שימוש בשיטה זו מבטיח שהקלט של המשתמש לא יפורש לעולם כחלק מהתחביר של שאילתות SQL אלא רק כערך, גם אם הוא מכיל תווים מסוכנים כמו גרשיים, נקודה־פסיק, או מילות מפתח של SQL.

הסיבה שצריך להגן על מסד הנתונים למרות שאין בו סיסמאות או מידע מאוד רגיש היא שלא רוצים שהמשתמש יכול לקבל את הpath של התמונות שעלול לרמוז על המבנה של המחשב של השרת.

הצפנות-

כדי שלא יוכלו להקשיב לתקשורת בין השרת והלקוח אני מצפין את התקשורת. המידע שעובר הוא לא בהכרח מידע רגיש (כל לקוח שרוצה יכול להתחבר ולקבל את כל התמונות) אך ההצפנות מסייעות בכך שמסתירות את איזו תמונה נשלחה על ידי איזה משתמש.

כדי להצפין את התקשורת בצורה בטוחה אני משתמש בהצפנה אסימטרית (RSA) כדי להחליף את המפתחות להצפנה הסימטרית (AES) שבה אני משתמש כדי להצפין את המידע שעובר בתקשורת.



העלאת קבצים-

בתקשורת שלי אני מעביר קבצים ושומר קבצים שלקוחות שולחים. יש הרבה סכנות בזה, מה אם לקוח שולח קובץ מאוד גדול וזה תופס הרבה מקום (בRAM בהתחלה ואז גם בדיסק). מה אם שולחים הרבה מאוד קבצים, יגמר המקום בדיסק. מה אם שולחים הרבה קבצים ואז אחסון של השרת "נאכל". משתמש זדוני עלול לנסות לנצל את חולשות אלו כדי לפגוע בשרת.

על מנת שהשרת יהיה בטוח מהתקפות מהסוג הזה קבעתי כמה הגבלות.

קודם כל יש כמות מוגבלת של מידע שיכול לעבור בהודעה (יש שדה של גודל הודעה בגודל 7 כלומר עשר מליון זה הכי הרבה בתים שיכולים להיות בהודעה), אז בלתי אפשרי לשלוח תמונות גדולות מידי, מעבר לזה הגבלתי את הגודל של התמונות שהשרת מוכן לקבל ל3000000. מעבר לכך אני עושה resize לתמונות לפני שאני שומר אותך לגודל 256x256 כדי שלא יוכלו לקחת לי כמה מקום באחסון שהלקוח ירצה. בנוסף לכך גם הגבלתי את כמות הקבצים שהשרת שלי שומר למספר קבוע (קבוע שאפשר לשנות בקובץ שמנהל את מסד הנתונים) (מסד הנתונים מזהה כפילויות כך שזה באמת יהיו תמונות שונות).

כך השרת שלי מוגן מתקיפות שמשתמש זדוני עלול לבצע באמצעות העמסה של קבצים.

בנוסף לכך יש סכנה של שליחה של הקבצים ללקוחות. נגיד מחוברים 100 לקוחות ויש לי 100 תמונות וכל הלקוחות מבקשים את כל התמונות. לפתוח את כל התמונות במקביל יקח הרבה מאוד מקום בRAM לכן פיצלתי את השליחה של התמונות לתמונה אחת עבור כל לקוח (כלומר שולח תמונה ואז פותח את התמונה הבאה כך שמתפנה המקום של התמונה הקודומת בRAM)



# מימוש הפרויקט

## סקירת המודולים

### מודולים חיצוניים

* Socket: משמש לתקשורת TCP בין הלקוח לשרת. המודול מאפשר פתיחת חיבורים, שליחת מידע וקבלת נתונים.
* Threading: מאפשר הפעלת תהליכונים (threads) במקביל, מה שמאפשר לשרת לנהל מספר לקוחות בו־זמנית וללקוח להריץ תקשורת ברקע.
* Queue: תור בטוח־לשרשורים, משמש בלקוח כדי לנהל משימות שמתקבלות מה־GUI ונשלחות לשרת.
* Os, sys: מספקים פונקציונליות לעבודה עם מערכת הקבצים והנתיבים, ולשליטה בתצורת הריצה של הקוד.
* Io: משמש ליצירת אובייקטים מסוג buffer שניתן להעביר בהם תמונות בפורמט בינארי (bytes).
* Pickle: מאפשר שמירה וטעינה של מודלים מאומנים לדיסק בצורה נוחה.
* Base64: שימושי לקידוד מידע בצורה בטוחה להעברה דרך פרוטוקול טקסטואלי, במיוחד כאשר יש צורך בשליחה של מידע בינארי (כמו תמונות) מעל TCP.
* Numpy: ספריית חישוב מטריציונית מרכזית שמשמשת לאורך כל הפרויקט לכל פעולות ה־forward, backward, עדכוני משקולות, עיבוד תמונות ועוד.
* PIL (pillow): משמשת לפתיחה, שמירה, שינוי קנה מידה של תמונות ועוד פעולות עיבוד תמונה.
* Cv2 (opencv): משמש בעיקר במודול ההגדלה (augment) של התמונות, לדוגמה סיבוב, תזוזה, דילול והוספת רעש.
* Crypto (pycryptodome): משמש להצפנת תקשורת הלקוח־שרת. נעשה שימוש ב־AES להצפנה סימטרית וב־RSA להצפנה א־סימטרית של מפתח AES.
* Tkinter: משמש לבניית הממשק הגרפי של הלקוח.
* time: משמש למדידת זמני הריצה של שלבי אימון המודל והדפסת משך האפוק.
* Copy: משמש לשמירה של עותק עמוק של המודל במהלך הלמידה במקרה שעוצרים את הלמידה מוקדם מהמתוכנן
* Gzip: משמש לפתיחת קבצי ה־MNIST הדחוסים בעת טעינת הדאטאסט.
* Urllib.request: משמש להורדת קבצי הדאטאסטים (MNIST, CIFAR-10) מהאינטרנט במידת הצורך.
* Tarfile: משמש לחילוץ (extract) של קובצי הארכיון של CIFAR-10 לאחר הורדתם.
* Abc: משמש להגדרת מחלקות אבסטרקטיות (כמו Layer, Loss) שאינן ניתנות להופעה ישירה אלא משמשות כבסיס למחלקות מורשות.
* Zipfile: משמש ליצירת הקובץ zip שמכיל את התמונות והמידע הנלווה בשביל המשתמש
* csv: משמש לכתיבה של הטבלת הנתונים בתוך קובץ הzip שהשלוח מוריד

## מודולים פנימיים

**Client (client.py)**

* **תפקיד:** אחראי על התקשורת מול השרת – פתיחת חיבור, שליחת קבצים, קבלת תגובות וביצוע Handshake מוצפן (RSA+AES).
* **תכונות:**
  + dest: כתובת השרת
  + Sock- socket TCP
  + Request\_queue: תור משימות
  + Connected: סטטוס חיבור
  + Crypto: אובייקט הצפנה מסוג ClientCrypto
  + Gui\_callback: הממשק הגרפי בשביל תצוגה
* **מתודות:**
  + –connect() יוזם את החיבור עם השרת ומבצע תהליך Handshake מוצפן. אין פרמטרים. לא מחזיר ערך.
  + –run() מפעיל את הלקוח כ thread -ומריץ את לולאת העבודה. אין פרמטרים. לא מחזיר ערך.
  + Handshake()- מחליף מפתחות הצפנה עם השרת באמצעות RSA ומאמת את הקשר. אין פרמטרים. לא מחזיר ערך.
  + – queue\_task(task\_code, \*args) מוסיף משימה לתור לביצוע עתידי. מקבל קוד פעולה וארגומנטים. לא מחזיר ערך.
  + – activate() מנהל את לולאת הביצוע המרכזית שמבצעת משימות ומתקשרת עם השרת. אין פרמטרים. לא מחזיר ערך.
  + – handle\_task(code, args) מבצע שליחת בקשה לפי קוד פעולה ופרמטרים. מחזיר ערך בוליאני – האם לצפות לתגובה מהשרת.
  + – send(\*msg) שולח הודעה מוצפנת לשרת. מקבל כל הודעה בפורמט תואם. לא מחזיר ערך.
  + – recv() מקבל הודעה מוצפנת מהשרת, מפענח ומחזיר אותה כמבנה נתונים מנותח (רשימת שדות).
  + – close() סוגר את החיבור TCP ומסיים את פעולת הלקוח. לא מחזיר ערך.
  + – send\_file(path) שם בqueue בקשת קובץ. לא מחזיר ערך.
  + – request\_images(digit=None) שם בqueue בקשת תמונות מהשרת. אם מצוין ספרה – מקבל רק תמונות מתויגות. לא מחזיר ערך.
  + request\_sign\_up(username, password) – שם בqueue בקשה להרשמה sign up
  + request\_log\_in(username, password) – שם בwork queue בקשה להרשמה login
  + – business\_logic(response) מפרש את התגובה שהתקבלה מהשרת ומבצע פעולה מתאימה בממשק. לא מחזיר ערך.
  + – convert\_image\_string\_to\_tuple(list) ממיר רשימת מחרוזות מהשרת למבנה רשומות של תמונות. מחזיר רשימת טאפלים (id, image, digit, confidence)
  + recv\_files\_process(, amount\_of\_files)- עושה את התהליך של קבלת התמונות מהשרת (מקבל amount\_ of\_files כמות הקבצים המצופה) מחזיר רשימת טאפלים (id, image, digit, confidence)

**ClientCrypto (client.py)**

* **תפקיד:** מטפל בהצפנה ופענוח של הודעות בין הלקוח לשרת באמצעות AES במצב CBC, כאשר מפתח AES עצמו מוצפן ב־RSA.
* **תכונות:**
  + : aes\_key מפתח סימטרי באורך 32 בתים
  + : aes\_iv וקטור אתחול באורך 16 בתים
* **מתודות:**
  + – encrypted\_key\_iv(rsa\_key) מקבל מפתח RSA ציבורי ומחזיר את המפתח הסימטרי AES כשהוא מוצפן וכן את ה־IV. משמש בעת Handshake.
  + – encrypt(plaintext) מצפין טקסט בפורמט string בעזרת AES. מחזיר מחרוזת מוצפנת ב־bytes.
  + – decrypt(encrypted\_text) מפענח מחרוזת bytes ומחזיר את הטקסט המקורי כ־string.

**ClientGUI (gui.py)**

* תפקיד:  
  ממשק משתמש גרפי עבור לקוח מערכת לזיהוי ספרות. מאפשר העלאת תמונות, עיון בגלריה, התחברות למשתמש, והורדת תמונות**.**
* **תכונות:**
* Root: חלון ראשי של Tkinter.
* Client: מופע של מחלקת Client המטפל בתקשורת עם השרת.
* : logged\_in\_user שם המשתמש המחובר כעת, או None אם אין חיבור.
* : connection\_label תווית המציגה סטטוס חיבור לרשת (אייקון).
* : status\_frame מסגרת עליונה הכוללת סטטוס חיבור וסטטוס התחברות.
* : login\_status\_label תווית המציגה את שם המשתמש המחובר או סטטוס אי-התחברות.
* : file\_path הנתיב לתמונה שנבחרה להעלאה.
* : file\_label תווית המציגה את שם הקובץ הנבחר.
* : current\_images רשימת התמונות שהתקבלו מהשרת בעת עיון בגלריה.
* : image\_refs הפניות לאובייקטים של תמונות (PhotoImage) למניעת מחיקתן מהזיכרון.
* : no\_internet\_img, connected\_img תמונות המייצגות מצב חיבור.
* : filter\_var ערך תיבת הבחירה לסינון תמונות בגלריה.
* : filter\_dropdown: תיבת בחירה להצגת תמונות לפי תווית ספרה.
* **מתודות:**
* activate()  
  מפעיל את לולאת הממשק הגרפי ומתחיל בדיקת סטטוס חיבור תקופתית.
* display\_result(message, message\_type)  
  מציג הודעה למשתמש – result כהודעת מידע, error כהודעת שגיאה.
* create\_main\_screen()  
  יוצר את תפריט הבית: העלאה, עיון, התחברות ויציאה.
* open\_login\_screen()  
  יוצר מסך התחברות/הרשמה, מאפשר שליחת בקשות התחברות/הרשמה דרך Client.
* gui\_set\_logged\_in\_user(username)  
  מעדכן את שם המשתמש המחובר להצגה בממשק.
* update\_login\_status(username)  
  משנה את תווית סטטוס ההתחברות בהתאם למשתמש הפעיל.
* create\_status\_frame()  
  יוצר את המסגרת העליונה עם סטטוס חיבור וסטטוס התחברות.
* update\_connection\_status()  
  בודק אם הלקוח מחובר ומעדכן את האייקון בהתאם.
* start\_connection\_polling(interval\_ms)  
  מתחיל בדיקה מחזורית (כל כמה אלפיות שניה) של מצב החיבור.
* open\_upload\_screen()  
  יוצר את מסך העלאת התמונה.
* upload\_image()  
  פותח חלון לבחירת קובץ תמונה ושומר את הנתיב.
* send\_image()  
  שולח את התמונה שנבחרה לשרת דרך Client.
* open\_browse\_screen()  
  יוצר את ממשק העיון בתמונות ומאפשר לבחור פילטר.
* browse\_images()  
  שולח בקשה לשרת להחזרת תמונות, בהתאם לפילטר שנבחר.
* display\_images(images)  
  מציג את התמונות בגלריה, כולל תצוגה מוקטנת של כל תמונה עם מזהה, תווית וביטחון.
* download\_zip()  
  שומר את התמונות הנוכחיות בקובץ ZIP, כולל labels.csv עם מידע על כל תמונה.
* exit\_gui()  
  סוגר את הממשק, סוגר את הלקוח ומסיים את התהליך.
* handle\_server\_response(response)  
  מציג תגובת שרת שהתקבלה מהלקוח (למשל תוצאה או שגיאה).

### ImagesORM (img\_db\_orm.py)

**תפקיד:**  
ניהול מסד נתונים מקומי (SQLite) הכולל שמירת תמונות ומטא־נתונים, הגבלת היסטוריה, ניהול משתמשים, אימות, ושיוך תמונות למשתמשים.

#### תכונות:

db\_path  
נתיב לקובץ מסד הנתונים (ברירת מחדל: 'images.db').

image\_dir  
תיקייה מקומית לשמירת קבצי תמונה.

conn  
חיבור פעיל למסד הנתונים (אובייקט sqlite3.Connection).

cursor  
אובייקט cursor לביצוע שאילתות מול בסיס הנתונים.

#### מתודות לניהול בסיס נתונים:

open\_DB()  
פותחת חיבור למסד הנתונים. חובה לפני פעולות כתיבה/קריאה.

close\_DB()  
סוגרת את החיבור למסד הנתונים.

commit()  
שומרת שינויים למסד הנתונים.

create\_tables()  
יוצרת את הטבלאות Users ו־Images אם אינן קיימות.

#### מתודות לניהול משתמשים:

register\_user(username, password)  
רושמת משתמש חדש. מחזירה user\_id אם הצליח, או None אם שם המשתמש תפוס.

authenticate\_user(username, password)  
מאמתת משתמש לפי הסיסמה. מחזירה user\_id אם הסיסמה נכונה, אחרת None.

#### מתודות לטיפול בתמונות:

save\_image\_file(image\_bytes, max\_size=256)  
שומרת תמונה מוקטנת, מחזירה מזהה, נתיב, ו־hash של התמונה.

insert\_image(image\_id, digit, path, confidence, hash\_val, user\_id)  
מוסיפה תמונה לטבלה אם לא קיימת לפי hash, מוחקת תמונות ישנות אם יש יותר מ־100.

delete\_old\_files()  
מוחקת קבצים מקומיים שאינם נמצאים במסד הנתונים ומנקה רשומות חסרות.

process\_and\_store(image\_bytes, digit, confidence, user\_id)  
מבצע תהליך מלא: שמירה, הכנסת שורה למסד, וניקוי קבצים מיותרים.

#### מתודות לשליפת תמונות:

get\_all\_images\_files()  
מחזירה את כל התמונות שנשמרו עם המידע המלא כקבצים.

get\_image\_by\_digit\_files(digit)  
מחזירה רק את התמונות שתויגו עם ספרה מסוימת.

**CNN (cnn.py)**

* **תפקיד:** מייצג את המודל המורכב של רשת קונבולוציה (CNN) כתצורת שכבות מותאמות אישית, המאפשרת ביצוע מעבר קדימה ואחורה ואימון.
* **תכונות:**
  + : layers רשימת השכבות המרכיבות את המודל.
  + : layer\_inputs קלטים זמניים לכל שכבה במהלך המעבר קדימה.
* **מתודות:**
  + – forward(input) מבצע מעבר קדימה דרך כל השכבות. מקבל טנזור קלט ומחזיר את פלט הרשת.
  + – backward(grad) מחשב את גרדיאנט השגיאה לאחור דרך כל השכבות. מקבל את הגרדיאנט מהשכבה האחרונה.
  + – update\_parameters(learning\_rate) מעדכן את פרמטרי כל שכבה נלמדת לפי גרדיאנט ואחוז הלמידה.

**Trainer (trainer.py)**

* **תפקיד:** מבצע את תהליך האימון של המודל עם תמיכה בירידת למידה, אימות ו־early stopping.
* **תכונות:**
  + : model המודל לאימון
  + :optimizer אובייקט SGD
  + : loss\_function פונקציית איבוד
  + : learning\_rate, decay\_epochs, decay\_rate פרמטרי שליטה בקצב הלמידה
  + : validator אובייקט ממחלקת Tester לצורך הערכה
  + : early\_stopping, patience בקרים להפסקת אימון מוקדמת
* **מתודות:**
  + – train(dataloader, num\_epochs, val\_dataloader=None) מאמן את המודל על בסיס הנתונים, מבצע אימות תקופתי, שומר את המודל הטוב ביותר ומטפל בהפסקת אימון מוקדמת אם נדרש.

**Tester (tester.py)**

* **תפקיד:** מודול להערכת ביצועי המודל המאומן באמצעות סט בדיקה או אימות.
* **תכונות:**
  + : model המודל להערכה
  + : loss\_function פונקציית איבוד
* **מתודות:**
  + – test(dataloader) מבצע מעבר קדימה על כל הדאטאסט ובודק דיוק ואיבוד ממוצע. מחזיר איבוד ודיוק.

**Conv2D (conv.py)**

* **תפקיד:** שכבת קונבולוציה דו־ממדית. מבצעת סינון של התמונות באמצעות פילטרים נלמדים (kernels).
* **תכונות:**
  + : weights משקלי הפילטרים
  + : biases ההטיות (bias) לכל פילטר
  + : weights\_gradient, biases\_gradient גרדיאנטים של הפילטרים וההטיות לצורך עדכון באימון
* **מתודות:**
  + – forward(input) מבצע קונבולוציה בין הפילטרים לבין הקלט, מחזיר מפה חדשה.
  + – backward(output\_grad) מחשב את הגרדיאנט של הקלט והמשקלים לפי גרדיאנט היציאה.
  + – update\_parameters(learning\_rate) מעדכן את הפילטרים וההטיות לפי הגרדיאנטים שנצברו.

**Dense (dense.py)**

* **תפקיד:** שכבה מלאה (Fully Connected) הממפה וקטור כניסה לווקטור יציאה בעזרת מטריצת משקולות.
* **תכונות:**
  + : weights מטריצת המשקולות
  + : biases הטיות
  + : weights\_gradient, biases\_gradient גרדיאנטים של הפרמטרים
* **מתודות:**
  + – forward(input) מחשבת מכפלה מטריציונית בין הקלט למשקולות ומוסיפה את ההטיות.
  + – backward(output\_grad) מחשבת את הגרדיאנט לפי חוק השרשרת עבור כל פרמטר.
  + – update\_parameters(learning\_rate) מעדכנת את הפרמטרים לפי הגרדיאנטים ואחוז הלמידה.

**Flatten (flatten.py)**

* **תפקיד:** ממירה טנזור מרובה ממדים לווקטור חד־ממדי לצורך מעבר לשכבות צפופות (Dense).
* **תכונות:**
  + : input\_shape שמירה על מבנה הקלט המקורי לצורך חישוב backward
* **מתודות:**
  + – forward(input) ממירה את הקלט לפורמט שטוח (batch\_size, -1).
  + – backward(output\_grad) מחזירה את הגרדיאנט למימדים המקוריים של הקלט.

**MaxPool2D (maxpool.py)**

* **תפקיד:** שכבת מקסימום שמבצעת דגימה מרחבית על ידי בחירת הערך המקסימלי בכל חלון (pooling window).
* **תכונות:**
  + : pool\_size גודל החלון
  + : stride קפיצת החלון בין אזורים
  + : input, mask קלט ו־mask לזכירת מיקום המקסימום, בשביל חישוב backward
* **מתודות:**
  + – forward(input) מחזירה מפת מאפיינים מדוללת עם הערכים המקסימליים מכל חלון.
  + – backward(output\_grad) מפזרת את הגרדיאנט חזרה רק למקומות שהכילו את המקסימום המקורי.

**InputLayer (input\_layer.py)**

* **תפקיד:** שכבת קלט פשוטה שמשמשת להעברת מידע לתוך הרשת בצורה עקבית.
* **מתודות:**
  + – forward(input) מחזירה את הקלט כפי שהוא.
  + – backward(output\_grad) מחזירה את הגרדיאנט כפי שהוא.

**ReLU (relu.py)**

* **תפקיד:** פונקציית אקטיבציה רקטיפית (Rectified Linear Unit) שמאפסת ערכים שליליים בקלט.
* **מתודות:**
  + – forward(input) מחזירה קלט שבו כל ערך שלילי מאופס, וכל ערך חיובי נשמר. משמשת כאקטיבציה לא־ליניארית.
  + – backward(output\_grad) מחשבת את הגרדיאנט של ReLU שומרת על גרדיאנט עבור ערכים חיוביים ומאפסת את השאר.

**Sigmoid (sigmoid.py)**

* **תפקיד:** פונקציית אקטיבציה שממפה את הקלט לערכים בין 0 ל־1 באמצעות נוסחת הסיגמואיד.
* **מתודות:**
  + – forward(input) מחזירה את הערכים שעברו דרך פונקציית הסיגמואיד
  + –backward(output\_grad) מחשבת את הגרדיאנט על פי הנגזרת של פונקציית הסיגמואיד ומחזירה את הגרדיאנט המותאם.

**Softmax (softmax.py)**

* **תפקיד:** פונקציית אקטיבציה לרמות פלט (output layer), שמעבירה את הקלט לחלוקה נורמלית (סכום 1) לצורך חישוב הסתברויות.
* **מתודות:**
  + – forward(input)מחשבת את הפלט הרך של הרשת כהסתברויות לכל מחלקה.
  + – backward(output\_grad)מחשבת גרדיאנט של פונקציית Softmax בהנחה שתחושב יחד עם פונקציית איבוד CCE.

**CategoricalCrossentropy (cce.py)**

* **תפקיד:** פונקציית איבוד למדידה של מרחק בין התפלגות הפלט של המודל לבין תיוג אמיתי.
* **תכונות:**
  + : predictions, targets מאחסן את הקלטים עבור חישוב קדימה ואחורה.
* **מתודות:**
  + – forward(predictions, targets) מחזירה את ערך ה־loss של CCE על סמך חישוב הלוגריתם של הפלט מול התיוג האמיתי.
  + – backward() מחזירה את גרדיאנט האיבוד: ההפרש בין הפלטים לבין התיוג )מניח שכבר עבר softmax)

**MeanSquaredError (mse.py)**

* **תפקיד:** פונקציית איבוד נפוצה לבעיות רגרסיה. מחשבת את ממוצע ריבועי ההבדלים בין הפלט לתשובה האמיתית.
* **תכונות:**
  + : predictions, targets ערכי פלט ותשובות לצורך חישוב ההפרש.
* **מתודות:**
  + – forward(predictions, targets) מחשבת את ממוצע ריבועי הסטיות בין הפלט לתשובה.
  + – backward() מחשבת את הגרדיאנט של פונקציית האיבוד ביחס לפלט.

**Layer (base.py)**

* **תפקיד:** מחלקה אבסטרקטית עבור שכבות רשת. משמשת כבסיס לכל שכבה עם תבנית לפונקציות forward backward.
* **מתודות:**
  + – forward(input) הגדרה אבסטרקטית. כל שכבה מממשת זאת בעצמה.
  + – backward(output\_grad) הגדרה אבסטרקטית. חובה למימוש בשכבות ירושה.
  + – update\_parameters(learning\_rate) פעולה ריקה ברירת מחדל, שכבות נלמדות ידרסו אותה לצורך עדכון פרמטרים.

**SGD (optimizer.py)**

* **תפקיד:** מבצע את עדכון המשקולות לפי אלגוריתם SGD (stochastic gradient descent)
* **מתודות:**
  + – step(layers, learning\_rate) עובר על כל שכבה שניתנת לאימון (TrainableLayer) ומעדכן את משקליה לפי הגרדיאנט שנצבר.

**Augment (augment.py)**

* **תפקיד:** מבצע אוגמנטציה של תמונות עבור הגדלת המגוון של הדאטאסט, כולל רעש, סיבוב, שיקוף ועוד.
* **מתודות:**
  + – random\_transform(image) מקבל תמונה ומחזיר גרסה משופצת שלה לפי פרמטרים אקראיים.
  + – rotate\_image(image) מסובב את התמונה בזווית אקראית.
  + – flip\_image(image) מחזיר את התמונה במצב שיקוף אופקי או אנכי באופן אקראי.
  + – add\_noise(image) מוסיף רעש גאוסיאני לתמונה.

**Initializer (initializers.py)**

* **תפקיד:** פונקציות אתחול עבור שכבות – מגדירות ערכים התחלתיים עבור משקולות.
* **מתודות:**
  + – xavier(shape) מחזיר מטריצה מאותחלת לפי Xavier initialization, כלומר ערכים שנדגמו מאחידות סביב אפס לפי גודל השכבה.

**TensorPatches (tensor\_patches.py)**

* **תפקיד:** עוזר פנימי למודול Conv2D – מספק פעולות המרה של טנסור ל־columns וחזרה לצורך קונבולוציה יעילה.
* **מתודות:**
  + – im2col(input\_data, filter\_h, filter\_w, stride=1) ממיר את טנסור התמונות לטבלת טלאים (patches) לצורך פעולת קונבולוציה מהירה.
  + – col2im(col, input\_shape, filter\_h, filter\_w, stride=1) מבצע את ההמרה ההפוכה: מטבלת טלאים חזרה לטנסור המקורי

**DataLoader (dataloader.py)**

* **תפקיד:** אחראי על חלוקת הדאטאסט לבאצ'ים, ניהול ערבוב וסדר, ותמיכה בלולאת אימון.
* **תכונות:**
  + : images, labels נתוני הקלט והתשובות
  + : batch\_size גודל באץ'
  + : shuffle האם לערבב את הסדר בכל אפוק
  + : indices, current\_index מעקב אחר מקום בלולאה
* **מתודות:**
  + –\_\_iter\_\_() מאתחל את האיטרטור, כולל ערבוב אם נדרש.
  + –\_\_next\_\_() מחזיר את הבאץ' הבא לפי גודל ומיקום.
  + – \_\_len\_\_() מחזיר את מספר הדגימות הכולל בדאטאסט.

**MNISTLoader (mnist.py) ו־CIFAR10Loader (cifar10.py)**

* **תפקיד:** טוענים את הדאטאסטים MNIST ו־CIFAR-10, כולל הורדה מהאינטרנט, המרה לפורמט NumPy וחלוקה לסטים.
* **מתודות עיקריות (בשניהם):**
  + – download\_dataset() מוריד את קבצי הדאטאסט מהאינטרנט אם לא קיימים מקומית.
  + – extract\_dataset() פותח את הקבצים (gzip / tar) ומכין את הדאטא לטעינה.
  + – load\_data() טוען את הנתונים לקובצי NumPy ומחזיר tuple של (images, labels).

**Server (server.py)**

* **תפקיד:** שרת ראשי שמאזין לבקשות TCP מהלקוחות, יוצר תהליכונים (threads) עבור כל לקוח, מנהל Handshake ומעבד בקשות תמונה.
* **תכונות:**
  + : rsa\_key מפתח RSA ל־Handshake.
  + : online דגל סטטוס השרת.
  + Sock: TCP socket
  + : clients רשימת לקוחות פעילים.
  + : server\_lock מנגנון נעילה למניעת התנגשויות בעת קבלת חיבורים.
* **מתודות:**
  + – activate\_server() מאזין לחיבורים חדשים, יוצר ClientHandler ומפעיל כל חיבור ב־Thread נפרד.
  + – close() עוצר את השרת ומסיים את כל התקשורת עם הלקוחות.

**ClientHandler (server.py)**

* **תפקיד:** מנהל את התקשורת עם לקוח ספציפי: מבצע Handshake, מקבל בקשות ומחזיר תגובות.
* **תכונות:**
  + : crypto הצפנת AES ו־RSA מול הלקוח.
  + : soc החיבור מול הלקוח.
  + : connected דגל שמציין אם החיבור פעיל.
  + : ai המודל הטעון לזיהוי ספרות.
  + : db\_orm גישה למסד הנתונים של התמונות.
* **מתודות:**
  + – run() מפעיל Handshake ומאזין ללולאת הבקשות.
  + – handshake() שולח מפתח ציבורי ומקבל מפתח AES מוצפן מהלקוח.
  + – send(\*msg) שולח הודעה מוצפנת ללקוח.
  + – recv() מקבל הודעה מהלקוח ומפענח אותה.
  + – business\_logic() מפענח את סוג הבקשה (העלאת תמונה, שליפה לפי ספרה וכו') ומגיב בהתאם.
  + – build\_return\_images\_msg(files) בונה את הפורמט של ההודעה הכוללת את כל התמונות מהמסד.
  + – identify\_num(picture\_content) מבצע עיבוד תמונה, העברת המידע למודל, ושמירה במסד.

**ServerCrypto (server.py)**

* **תפקיד:** טיפול בהצפנת AES/PKCS1 בצד השרת.
* **תכונות:**
  + : rsa\_key, aes\_key, aes\_iv מפתחות להצפנה ופענוח.
* **מתודות:**
  + – get\_public() מחזיר את המפתח הציבורי של השרת (RSA).
  + – decrypt\_aes\_key(aes\_key, aes\_iv) מפענח את מפתח ה־AES וה־IV שנשלחו מהלקוח.
  + – encrypt(plaintext) מצפין טקסט בפורמט AES CBC
  + – decrypt(encrypted\_text) מפענח הודעה לפי מפתח AES.

**Protocol Functions (protocol.py)**

* **תפקיד:** מנהל את פרוטוקול התקשורת בין השרת ללקוח, כולל קידוד, קידוד בסיסי, שליחה לפי גודל, לוגים.
* **קבועים:**
  + HOST, PORT, SEPERATOR, MAX\_FILE\_SIZE, OPCODESועוד.
* **פונקציות:**
  + – recv\_by\_size(sock, return\_type="bytes") קורא הודעה לפי גודל שהוקצה בתחילה.
  + – send\_by\_size(sock, data) שולח נתונים עם ציון גודל.
  + – format\_message(args) מקודד רשימת שדות למחרוזת אחת בעזרת Base64 (לפי הפרוטוקול)
  + – unformat\_message(msg) מפענח את ההודעה לפורמט רשימת שדות. (לפי הפרוטוקול)
  + –\_\_log(prefix, data, max\_to\_print=100) הדפסת Debug להודעות.

## הסבר אלגורתמים

### Backward של שכבת קונבלוציה

def backward(self, output\_grad):  
 *"""Computes the gradient of the loss with respect to the input (and with respect to the weights and biases).  
 - output\_grad: The gradient of the loss with respect to the output.  
 - Returns the gradient of the loss with respect to the input.  
 """* # reshape the output gradient to match the shape of the patches  
 output\_grad\_flat = output\_grad.transpose(0, 2, 3, 1).reshape(-1, self.out\_channels)  
 # Calculate the gradient of the loss with respect to weights  
 self.weights\_gradient = np.dot(output\_grad\_flat.T, self.patches)  
 self.weights\_gradient = self.weights\_gradient.reshape(self.out\_channels, self.in\_channels, self.kh, self.kw)  
  
 # Calculate the gradient of the loss with respect to bias  
 # sum of the output gradient (batch-wise)  
 self.biases\_gradient = np.sum(output\_grad\_flat, axis=0)  
  
 # Calculate the gradient of the loss with respect to the input (for the next layer)  
 # ----------------------------------------------------------  
 # Gradient w.r.t. the input (∂L/∂input) - Intuition:  
 #  
 # For each input pixel:  
 # - Look at all output pixels that were computed using it.  
 # - Each of those output pixels has a gradient telling how much it wants to change.  
 # - The input pixel affected that output pixel through a specific weight in the filter.  
 # - So we multiply the output gradient by that weight (the strength of the connection).  
 # - Then we sum up all of these contributions.  
 #  
 # From the output's perspective:  
 # - Each output pixel "spreads" its gradient back to the input patch it came from.  
 # - It does so proportionally to the filter weights used during the forward pass.  
 #  
 # The final result tells each input pixel: "Here's how much you need to change to reduce the loss."  
 # ----------------------------------------------------------  
 N, C, H, W = self.input\_data.shape  
 out\_h = (H - self.kh) // self.stride + 1  
 out\_w = (W - self.kw) // self.stride + 1  
 input\_grad\_patches = np.dot(output\_grad\_flat, self.weights.reshape(self.out\_channels, -1))  
 input\_grad\_patches = input\_grad\_patches.reshape(N, out\_h, out\_w, C, self.kh, self.kw)  
 input\_grad\_patches = input\_grad\_patches.transpose(0, 3, 4, 5, 1, 2)  
 # input\_grad\_patches holds the gradient of the loss with respect to each input patch.  
 # Shape: (N, C, kh, kw, out\_h, out\_w)  
 #  
 # For each image in the batch (N), each input channel (C), and each output location (out\_h, out\_w),  
 # this array tells us how much each pixel in the receptive field (defined by the kernel window kh x kw)  
 # should change to reduce the loss.  
 #  
 # This will be scattered back into the full input gradient image, summing overlapping contributions.  
 # ----------------------------------------------------------  
 input\_grad = col2im(input\_grad\_patches, self.input\_data.shape, (self.kh, self.kw), self.stride, self.padding)  
 return input\_grad

כאשר עושים backward מקבלים את הגרדיאנט של loss ביחס לפלט של השורה וצריך לחשב את הגרדיאנט של הloss ביחס למשקולות, ביחס להטייה וביחס לקלט (כדי למסור לשכבה הבאה כמו שהשכבה הזו קיבלה)

1. גרדיאנט ביחס למשקולות (weights):

עבור כל פילטר בשכבה, נחשב את ההשפעה שהייתה לכל קלט על התוצאה, כפי שהיא באה לידי ביטוי בגרדיאנט שקיבלנו מהשכבה הבאה. לשם כך, אנו ממירים את הקלט (input) לייצוג של "טלאים" (patches) התואם לגודל הקרנל, כך שכל שורה מייצגת אזור קלט קטן שהופעל עליו פילטר. לאחר מכן, אנו מבצעים מכפלה מטריציונית בין הגרדיאנט של הפלט לבין אותם טלאים. בצורה זו, אנחנו מקבלים את השינוי הרצוי בכל משקל של הפילטרים כדי להפחית את פונקציית העלות.

1. גרדיאנט ביחס להטייה (bias):

ההטייה משפיעה באופן ישיר על כל פלט של הפילטר. לכן הגרדיאנט ביחס להטייה הוא פשוט סכום של כל גרדיאנטי הפלט – כלומר, כמה כל יציאה תרמה ל־loss, ללא קשר למיקום הספציפי בקלט. זה מתבצע על ידי סכימה על כל הדוגמאות בבאטצ׳.

1. גרדיאנט ביחס לקלט (input):

מטרת חלק זה היא להעביר את הגרדיאנט הלאה לשכבה הקודמת. כדי להבין כמה כל פיקסל קלט תרם לשגיאה, אנו "מפזרים" את גרדיאנט הפלט חזרה לפי מפת ההשפעה שהייתה בקרנלים (המשקולות) על כל אזור קלט. כלומר, עבור כל גרדיאנט פלט, אנו מכפילים אותו במשקולות המתאימות וממקמים את התוצאה באזור הקלט שהשפיע עליו, תוך כדי סכימה של אזורים חופפים. פעולה זו מתבצעת באמצעות פונקציית עזר col2im, שמפזרת את הגרדיאנטים מהטלאים חזרה למבנה הקלט המקורי, כך שניתן להמשיך ולהעביר את הגרדיאנט לאחור ברשת.

החישוב השלישי מעניין, אפשר להסתכל עליו משני דרכים

לכל פיקסל בקלט**:**

* הסתכל על כל הפיקסלים ביציאה שחושבו באמצעותו.
* לכל אחד מהפיקסלים ביציאה יש גרדיאנט שמציין עד כמה הוא "רוצה" להשתנות.
* הפיקסל בקלט השפיע על אותו פיקסל ביציאה דרך משקל מסוים במסנן (filter).
* לכן, נכפיל את גרדיאנט היציאה במשקל הזה (שמשקף את עוצמת ההשפעה).
* ואז נסכם את כל התרומות האלו.

**מנקודת המבט של היציאה:**

* כל פיקסל ביציאה "מפזר" את הגרדיאנט שלו חזרה אל הטלאי בקלט שממנו הוא הגיע.
* הוא עושה זאת באופן פרופורציונלי למשקולות המסנן ששימשו במהלך המעבר קדימה (forward pass).

התוצאה של זה תהיה עבור כל פיקסל בקלט, הנה כמה שאתה צריך להשתנות כדי שהloss יעלה.

## מסמך בדיקות

### הבדיקות שתכננתי בשלב האפיון

* 1. העלאת תמונה תקינה

המטרה הייתה לבדוק האם המערכת מזהה תמונה שמכילה ספרה ברורה באופן תקין.

בדקתי את הרשת נוירונים שלי כמו שתיכננתי והרשת הצליחה לזהות את הספרות הברורות.

* 1. העלאת תמונה באיכות נמוכה/רעש

המטרה הייתה לבדוק האם הרשת עמידה בפני תמונות פחות איכותיות

אמנם סיכויי ההצלחה יורדים מעט והרשת לא מצליחה לזהות בכמעט 100 כמו שהרשת מצליחה תמונות ברורות עדיין הרשת הצליחה לזהות את הרוב המוחלט של הספרות גם אם היו בהן רעש או שאיכות התמונה היתה נמוכה יותר

* 1. העלאת קובץ שאינו תמונה

המטרה היא לבדוק שהמערכת יודעת להתמודד על קלט לא חוקי.

בגלל שטיפלתי באופן ספציפי במקרה הזה השרת שלי זיהה שזה לא קובץ של תמונה ושלח הודעה מתאימה.

* 1. בדיקת תצוגת ההיסטוריה

המטרה הייתה לוודא שניתן לצפות בתמונות שזוהו בעבר והועלו על ידי משתמשים אחרים. כדי לעשות את זה חיברתי כמה לקוחות והעלתי מהם תמונות, יכולתי לצפות בכל התמונות דרך כל אחד מהלקוחות כפי שהיה דרוש.

* 1. בדיקת תגובתיות הממשק

המטרה הייתה לבדוק האם הממשק הגרפי עובד בצורה מהירה וחלקה, בדקתי את כל היכולות/כפתורים של הממשק הגרפי וגיליתי בעייה בגלילה עם בעכבר בגלריית תמונות. מסתבר שהקוד שהשתמשתי בו כדי לגלול עם העכבר היה עבור גרסא ישנה יותר של tkinter ולכן החלפתי את הדרך שבה אני גולל עם העכבר ואז הגלילה עבדה כראוי.

* 1. לנתק את השרת בזמן שהלקוח מחובר אליו ולנסות להפעיל לקוח כששרת לא מחובר והפוך

המטרה הייתה לבדוק האם השרת והלקוח לא קורסים כאשר הצד האחר מתנתק או לא מחובר.ביצעתי את כל הבדיקות וכל הבדיקות עבדו כראוי, הלקוח מתחבר לשרת אם יש שרת שפועל ואם שרת מתנתק הלקוח מחפש שרת חדש. והשרת לא קורס ורושם הערה כאשר לקוח מתנתק ממנו.

* 1. בדיקת אחסון במסד הנתונים

המטרה הייתה לבדוק שהמידע שאמור להיסמר במסד הנתונים נשמר כראוי. כדי לעשות את זה שלחתי תמונה וראיתי שנוצר עמודה חדשה בטבלה (בעזרת SQLite viewer) וראיתי שנוצרה תמונה חדשה במקום הנכון, לאחר מכן העלתי את אותה התמונה ולא נוצר קובץ חדשה/שורה חדשה כפי שרציתי.

### הבדיקות נוספות שביצעתי

1. מחקתי קבצים של תמונות שהיו שמורות והיה מעקב אחריהן במסד נתונים

המטרה הייתה לבדוק מה קורה במקרה של מחיקה של קבצים שהמערכת חושבת שנמצאים שם

גיליתי שנוצר error בשרת ולכן הוספתי בדיקה לפני שאני פונה לקבצים ובנוסף מחקתי שורות במסד נתונים של תמונות שלא נמצאו בתקייה שבה התמונות אמורות להיות שמורות.

1. הזנתי תמונה של תווים דקים ושל תווים שהוזזו

המטרה הייתה לבדוק האם המערכת מסוגלת לזהות תווים דקים/ שעברו הוזזה

המערכת התקשתה לזהות תווים דקים ושעברו הוזזה, כדי להתגבר על זה הגברתי את השינויים שאני עושה למידע שאני מתאמן עליו, הגדלתי את המודל (מגביר יכולת אך מאריך זמן אימון וזמן עיבוד של תצמונות) והוספתי לpreprocessing שלי חלק שהופך את הכתב להיות טיפה יותר עבה. השינוים האלו שיפרו את הביצועים של המערכת שלי בהרבה.

1. סדקתי את מערכת ההרשמה וההתחברות. בדקתי שהשרת לא שומר תמונות שאני שולח כאשר אני לא מחובר למשתמש, הכנתי משתמש, ניסיתי להכין משתמש עם אותו שם משתמש, נכנסתי למשתמש מלקוח אחר (בדקתי שהוא לא מכניס אותי אם הסיסמה שגויה) ובדקתי שלאחר שאני מחובר למשתמש הוא שומר את התמונות שאני שולח לו. המערכת פעלה כצפוי ונתנה את הפלטים הרצויים.

# מדריך למשתמש

## כלל קבצי המערכת

כל הקבצים שצריך כדי להריץ את המערכת, בזמן הרצה עלולים להיפתח תקיות וקבצים חדשים (נגיד לשמירה על dataset שהורידו בשביל אימון או בשביל לשמור תמונות בתוך /server/saved\_image)

.

├── client

│   ├── client.py

│   ├── gui.py

│   ├── main.py

│   └── static

│   ├── browse.png

│   ├── connected.png

│   ├── exit.png

│   ├── no\_connection.png

│   ├── login.png

│   └── upload.png

├── protocol.py

├── requirements.txt

└── server

├── CNNall

│   ├── CNN

│   │   ├── \_\_init\_\_.py

│   │   ├── data

│   │   │   ├── \_\_init\_\_.py

│   │   │   ├── dataloader.py

│   │   │   └── datasets

│   │   │   ├── \_\_init\_\_.py

│   │   │   ├── cifar10.py

│   │   │   └── mnist.py

│   │   ├── layers

│   │   │   ├── \_\_init\_\_.py

│   │   │   ├── activations

│   │   │   │   ├── \_\_init\_\_.py

│   │   │   │   ├── relu.py

│   │   │   │   ├── sigmoid.py

│   │   │   │   └── softmax.py

│   │   │   ├── base.py

│   │   │   ├── conv.py

│   │   │   ├── dense.py

│   │   │   ├── flatten.py

│   │   │   ├── input\_layer.py

│   │   │   └── maxpool.py

│   │   ├── losses

│   │   │   ├── \_\_init\_\_.py

│   │   │   ├── base.py

│   │   │   ├── cce.py

│   │   │   └── mse.py

│   │   ├── models

│   │   │   ├── \_\_init\_\_.py

│   │   │   └── cnn.py

│   │   ├── trainers

│   │   │   ├── \_\_init\_\_.py

│   │   │   ├── tester.py

│   │   │   └── trainer.py

│   │   └── utils

│   │   ├── \_\_init\_\_.py

│   │   ├── augment.py

│   │   ├── initializers.py

│   │   ├── optimizer.py

│   │   └── tensor\_patches.py

│   ├── init.py

│   └── main.py

├── saved\_images

├── img\_db\_orm.py

├── main.py

├── images.db

├── main\_model.pkl

└── server.py

## התקנת המערכת

### פירוט הסביבה הנדרשת

כדי להשתמש בפרויקט צריך windows 11/10 וpython3.9.

כל הקבצים נמצאים ב-

<https://github.com/YuvaMendel/FinalSchoolProject/tree/main/src/app>

הספריות שצריך להוריד מצויינות ב requirements.txt שנצבא בroot, (כדי להוריד את כל המודולים להריץ בשורת הפקודה:“pip install -r requirements.txt”).

כדי להריץ את השרת יש להריץ “python main.py” מתקיית /server

כדי להריץ את הלקוח יש להריץ “python main.py {server\_ip}” מתקיית /client (ברירת מחדל של ה ip זה 127.0.0.1) בשורת הפקודה.

### מיקומי הקבצים

המבנה הכללי של הפרויקט הוא כזה:

יש תקייה server וclient ו קובץ protocol.py אם מריצים רק את השרת אין צורך בתקיית client אבל צריך את protocol.py בתקייה שבה נמצאת התקייה server.

כמו כן לא צריך את תקיית server בשביל להריץ את הלקוח, צריך את protocol.py באותה תקייה שבה התקייה client נמצא בו.

בתוך התקייה server יש קובץ pickle שנקרא main\_model.pkl זה המודל לזיהוי תמונה שבו השרת משתמש, כדי לשנות את המודל צריך להחליף את הקובץ הזה.

כדי לאמן מודל חדש צריך להיכנס לCNNall בתוך תקיית server ולהריץ שם את main.py (אימון עלול לקחת הרבה זמן, תלוי בכמה גדול המודל וכמה מאמנים אותו, אפשר לשנות את המודל שמאמנים אבל בשביל זה צריך לשנות את main.py)

## משתמשי המערכת

יש 2 משתמשים, שרת ולקוח, אחד מנהל מאגר ואחרים פונים אליו לזיהוי תמונות.

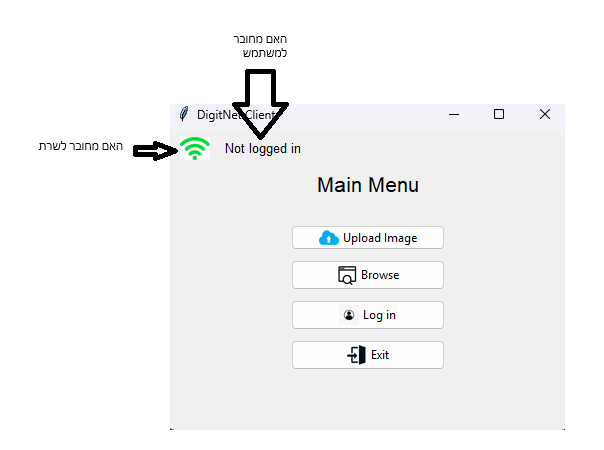
### שימוש בלקוח

כדי להריץ את הלקוח יש להריץ “python main.py {server\_ip}” מתקיית /client (ברירת מחדל של ה ip זה 127.0.0.1) בשורת הפקודה.

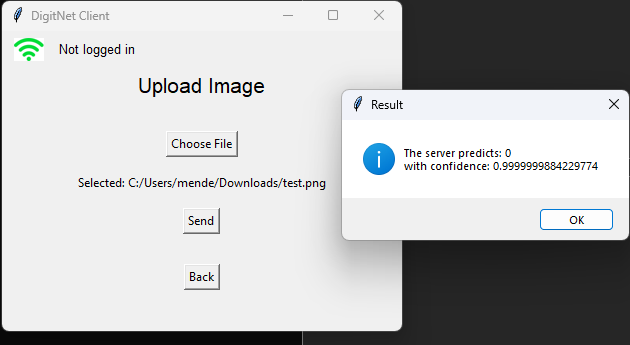
יפתח העמוד הראשי

בצד שמאל למעלה יש מידע על החיבור לשרת (האם המשתמש מחובר לשרת האם הוא התחבר למשתמש)

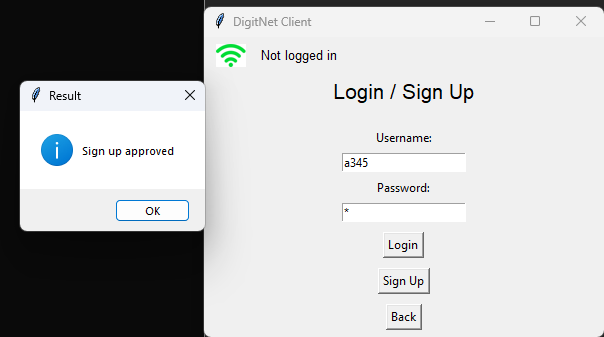
בנוסף לכך יש כפתורים למעבר להעלאת תמונות, צפייה בתמונות שהשרת שמר, והרשמה/התחברות למשתמש.



במסך העלאת תמונות אפשר ללחוץ על “Choose File” כדי לפתוח בוחר קבצים ולבחור שם תמונה להעלות. אם נבחרה תמונה אז השם של התמונה יופיע מעל הכפתור send . כדי לשלוח את התמונה לשרת יש ללחוץ send, במקרה הזה תופיע הודעה עם התשובה של השרת. אם המשתמש מחובר התמונה תישמר בשרת. כדי לחזור למסך הראשי יש ללחוץ על כפתור Back



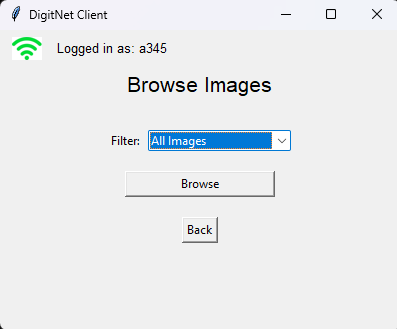
במסך רישום (sign up ו log in ) אפשר לרשום את השם משתמש והסיסמה ולהירשם/להתחבר למשתמש קיים. אם השם משתמש תפוס/סיסמה לא מתאימה הודעה תעלה בהתאם.



A screenshot of a computer login screen

AI-generated content may be incorrect.

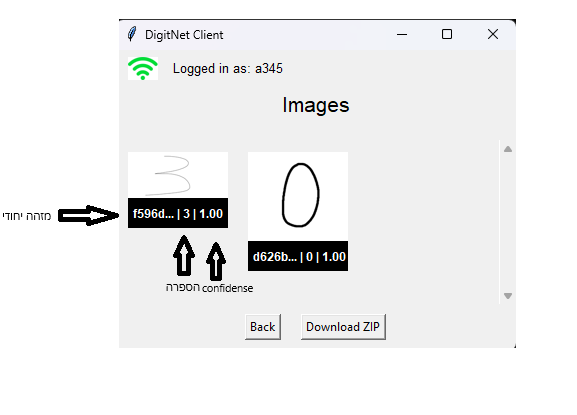
במסך בחירת תמונות אפשר לבחור את הפילטר (או כל תמונה או לבחור ספרה ספציפית). כדי לצפות התמונות יש ללחוץ browse, כדי לחזור למסך הראשי יש ללחוץ Back.



במסך צפייה בתמונות אפשר לצפות בתמונות של החיפוש שנעשה ולחזור למסך הראשי.

Download zipהכפתור Download ZIP יאפשר להוריד זיפ של התמונות ביחד עם הזיהוי שהמודל יחס להן.

אפשר לגלול עם הגלגלת של העכבר, מתחת לכל תמונה יופיע 3 פרטים: האותיות הראשונות של המזהה היחודי של התמונה, הספרה שהמודל זיהה בתמונה, ה"ביטחון" (confidence) של המודל בנכונות של התוצאה.



### שימוש בשרת

כדי להריץ את השרת יש לכתוב בשוראת הפקודה “python main.py” בתקיית /server .

המשתמשים והmetadata על התמונות ישמרו בdatabase בשם images.db התמונות שנשלחות על ידי לקוחות מחוברים (עשו login) ישמרו בתקיית /server/saved\_images .

המודל שבו משתמשים כדי לפענח את התמונות שמור בקובץ /server/main\_model.pkl , אם רוצים להחליף את המודל יש להחליף את הקובץ הזה.

כדי לאמן מודל חדש יש להריץ את “python main.py” בתוך תקיית /server/CNNall שיוציא מודל חדש בקובץ main.pkl בתקיית /server/CNNall (כדי להפוך אותו למודל שבו השרת משתמש לשנות את שמו לmain\_model.pkl ולהעביר אותו לתקיית /server כל לקוח חדש שיכנס יקבל שירות מהמודל החדש)

A screenshot of a computer screen

AI-generated content may be incorrect.

# רפלקציה

במהלך העבודה על הפרויקט התעסקתי בהרבה תחומים טכנולוגיים, תקשורת בין שרת ללקוח, הצפנה אבטחת מידע, שימוש ברשתות נוירונים לזיהוי תמונות ועוד.

לאורך הדרך נעזרתי במשאבים מגוונים – מדריכים באינטרנט, קורסים קודמים שלמדתי, ולעיתים גם בשאלות שהפניתי ל- ChatGPT שסייע לי להבין נושאים לעומק, לנסח קוד בצורה ברורה, ולחדד הסברים שכתובים בתיעוד.

הקושי הכי גדול שהיה לי היה להבין כיצד רשתות נוירונים עובדות בתאוריה וכיצד לממש אותן בפועל. לקח לי הרבה זמן לקרוא ולהבין איך לכתוב כל חלק. לדוגמה, לאחר שהבנתי איך להעביר את הגרדיאנט אחורה בצורה תאורטית ברשת קונבלוציה ניסיתי לממש את זה ונתקלתי בבעיה גדולה, להמיר את הידע התאורטיי שרכשתי לקוד היה מטלה מאוד קשה.

אחד הדברים החשובים שלמדתי הוא כיצד לפרק בעיה מורכבת לחלקים קטנים, להבין כל רכיב לעומק, ואז להרכיב אותם יחד לפתרון שלם שעובד.

מעבר לצד הטכנולוגי, למדתי גם רבות על עבודה מסודרת – כתיבת תיעוד, שמירה על קוד קריא ומודולרי, ניהול גרסאות, והצגה של המערכת בצורה שתהיה ברורה גם למי שלא כתב את הקוד.

בראייה לאחור הייתי מיישם בדרך אחרת את השרת והלקוח, אמנם ניסיתי לשמור על מודולריות אך חלק מהקוד של האוביקטים של ההצפנה של השרת והלקוח חופפים ואני חושב שלשלב את 2 המחלקות למחלה משוטפת זה פתרון יותר יפה. בנוסף לכך הייתי משנה את המבנה של הקבצים, את המודול שהכנתי לCNN הכנתי בצורה יפה מאוד ואם הייתי מיישם את השרת והלקוח מחדש הייתי מחלק את הקבצים ואת הפונקציות בצורה יותר מסודרת.

במידה והיו ברשותי עוד משאבים הייתי משפר את הפרוייקט בכך שהייתי משנה את המודל ללמוד איך לזהות דברים יותר מסובכים מאשר רק ספרות. בגלל החומרה המוגבלת שלי ובגלל שממימשתי את הפרויקט בnumpy (ולא השתמשתי בספרייה כמו pytorch) היכולת שלי לאמן מודל גדול מאוד מוגבלת. אם היו לי יותר משאבים הייתי בונה מערכת שמזהה דברים יותר מורכבים כמו משפטים או ממש ממירות תמונות של מסמכים לטקסט (OCR)

לסיכום, הפרויקט הזה היווה עבורי הזדמנות אמיתית לצמיחה אישית ומקצועית. הוא חיבר בין ידע תאורטי לבין יישום מעשי, ואתגרים טכנולוגיים הפך להזדמנויות למידה. אני מרגיש שהצלחתי לפתח לא רק מערכת שעובדת, אלא גם דרך חשיבה ויכולת פתרון בעיות שישמשו אותי בהמשך הדרך, בכל פרויקט עתידי שאקח בו חלק.

# בבליוגרפיה

3Blue1Brown. (2017, October 5). *But what is a neural network? | Deep learning chapter 1* [Video]. YouTube. <https://www.youtube.com/watch?v=aircAruvnKk>

3Blue1Brown. (2017, October 16). *Gradient descent, how neural networks learn | Deep learning chapter 2* [Video]. YouTube. <https://www.youtube.com/watch?v=IHZwWFHWa-w>

3Blue1Brown. (2017, November 3). *What is backpropagation really doing? | Deep learning chapter 3* [Video]. YouTube. <https://www.youtube.com/watch?v=Ilg3gGewQ5U>

3Blue1Brown. (2017, November 3). *Backpropagation calculus | Deep learning chapter 4* [Video]. YouTube. <https://www.youtube.com/watch?v=tIeHLnjs5U8>

Nielsen, M. A. (2015). *Chapter 1: Using neural nets to recognize handwritten digits*. In *Neural networks and deep learning*. <http://neuralnetworksanddeeplearning.com/chap1.html>

IBM Technology. (2021, October 6). What are Convolutional Neural Networks (CNNs)? [Video]. YouTube. <https://www.youtube.com/watch?v=QzY57FaENXg>

# נספחים