

זיהוי תוכנות זדוניות על פי הצגה בינארית

מגיש: אסף הראל

ת"ז: 326191442

מורה: גד לידרור

בית הספר: תיכונט



תוכן עניינים

3	מבוא
3	רקע לפרויקט
4	תהליך המחקר
6	כיצד זיהוי בעזרת הצגה בינארית פותר את החסרונות של אנטי-וירוס מסורתי?
6	קשיים עם ההצגה הבינארית במהלך העבודה
7	אתגרים מרכזיים
7	SOINN – Self Organizing Incremental Neural Network
8	הפתרון לאתגר
9	מבנה המודל
10	מבנה הפרויקט
10	איסוף הנתונים
11	הכנה וניתוח הנתונים
13	בניית המודל
15	איטרציה #1
16	איטרציה #2
18	איטרציה #3
19	איטרציה #4
20	סיכום שלב האימון
20	פונקציית השגיאה (CCE) Categorical Cross Entropy
20	פונקציית השגיאה (SCCE) Sparse Categorical Cross Entropy
21	ייעול התכנסות Adam – (Optimization)
21	התמודדות עם הטיה ושונות
21	Local Response Normalization
23	ממשק המשתמש
24	UML של היישום
24	ספריית PyQt5
24	כיצד המידע מגיע אל היישום
25	מדריך למפתח
25	הצגה בינארית
25	מבנה הפרויקט:
26	color.py
27	hilbert.py
30	progress.py
33	utils.py
36	converter.py
38	binvis.py
42	הפרויקט הסופי
42	מבנה הפרויקט:
43	utils.py
49	test_model.py
50	app.py
53	מדריך למשתמש
53	התקנה
53	תרשים מסכים
56	רפלקציה
57	ביבליוגרפיה



מבוא

רקע לפרויקט

במשך השנים אנו הופכים ליותר ויותר תלויים בטכנולוגיה ודבר זה פותח חלון לאנשים בעלי כוונות זדוניות לנצל זאת וליצור תוכנות זדוניות (וירוסים) שיפגעו במחשבים שלנו ויגנבו מאיתנו מידע. כיום יש הרבה תוכנות "אנטי-וירוס" שמטרתן לזהות תוכנות זדוניות שהוחדרו למחשב אך גם אותן האקרים מצליחים לעקוף בכך שהם יוצרים תוכנה זדונית שונה מאוד מהתוכנות הקיימות.

מטרת הפרויקט היא לענות על הצורך לתוכנת אנטי-וירוס שמסוגלת לזהות גם תוכנות זדוניות חדשות שעוד לא נראו בעולם.

הזיהוי נעשה בעזרת רשת נוירונים שלוקחת קבצי exe (executable), הופכת אותם לתמונה בעזרת ייצוג בינארי, מתמצתת מהם את התכונות החשובות ומזהה האם הם תוכנה זדונית או לא. **המוצר העתידי** יהיה תוכנת אנטי-וירוס שתסרוק את כל קבצי ה-exe במחשב, תעבד את הקובץ ותכניס את ייצוגו הבינארי לתוך המכונה ותקפיץ התראה למשתמש אם קובץ נמצא כחשוד.

קהל היעד של הפרויקט הוא כל אחד, בין אדם פרטי לחברה גדולה. בעולם של היום יש מעל 2 מיליארד מחשבים שונים בין מחשב פרטי של קופאית בסופר לשרתים הענקיים של גוגל וכל אחד מהם חשוף לאימתני ההאקרים והתוכנות הזדוניות על בסיס יומי, למען האמת, 30% מהמחשבים בארצות הברית ו-57% מהמחשבים בדרום קוריאה "נדבקו" בתוכנות וירוסים (ואלו כמובן רק המחשבים שמצאו בהם).

אני בחרתי לחקור את הנושא הזה מכיוון שאני חושב שכל התחום של האנטי-וירוסים נשאר קבוע כבר הרבה מאוד שנים ואין זה יותר ממתבקש שככל שהטכנולוגיה מתקדמת וההאקרים מתקדמים גם ככה ההגנות עלינו המשתמשים יתקדמו.



תהליך המחקר

פונקציית hash (גיבוב) – פונקציה שלוקחת פיסת מידע, ומוציאה פלט באורך קבוע הייחודי רק לה.
טבלת hash – מבנה נתונים מילוני (מבנה של "מפתח-ערך") שמשתמשים בפלט של פונקציית ה-hash בתור המפתח שלו ועל פי זה מקבלים מזהה ייחודי לכל פיסת מידע

דוגמה ל-hash:

נבצע פונקציית hash מסוג sha256 לכל הטקסט שכתוב בעמוד הקודם ←
"79ff06cba15ed7d0750b3d0258125cbd328bb03369a1cc4f43b009453554c6bf"

כמו שאפשר לראות, פונקציית ה-hash הפכה עמוד שלם למחרוזת של 64 תווים.

תחילה חקרתי כיצד האנטי-וירוס המסורתי עובד:

לתוכנה יש מאגר גדול מאוד של מזהים שהתקבלו על ידי הפעלת פונקציית ה-hash על תוכנות זדוניות קיימות שאותם מצליבים עם מזהה של קובץ על המחשב ובודקים אם הוא במאגר.

החסרונות בדרך הפעולה הנוכחית

אם תוכנת האנטי-וירוס נתקלת בתוכנה זדונית שונה ממה שקיים במאגר שלה אז היא לא תזהה אותו, מכאן שאם האקר יהיה מתוחכם ויצירתי מספיק הוא יוכל לעקוף את ההגנות של המחשב.

קבצי הרצה (exe) – קובץ המורכב מפקודות מכונה (הוראות ישירות למעבד של המחשב) שתורגמו מקוד של תוכנית, כל התוכנות על המחשב הם מסוג exe.

הצגה בינארית – הצגה של קובץ בתור תמונה, הקוד שעושה זאת מחלק 5 טווחי מספרים בינאריים לצבעים שונים:

1. 0x00 – שחור
2. low – ירוק
3. ascii – כחול
4. high – אדום
5. 0xff – לבן



דוגמה להצגה בינארית:

חלק מהערכים הבינאריים של קובץ וירוס

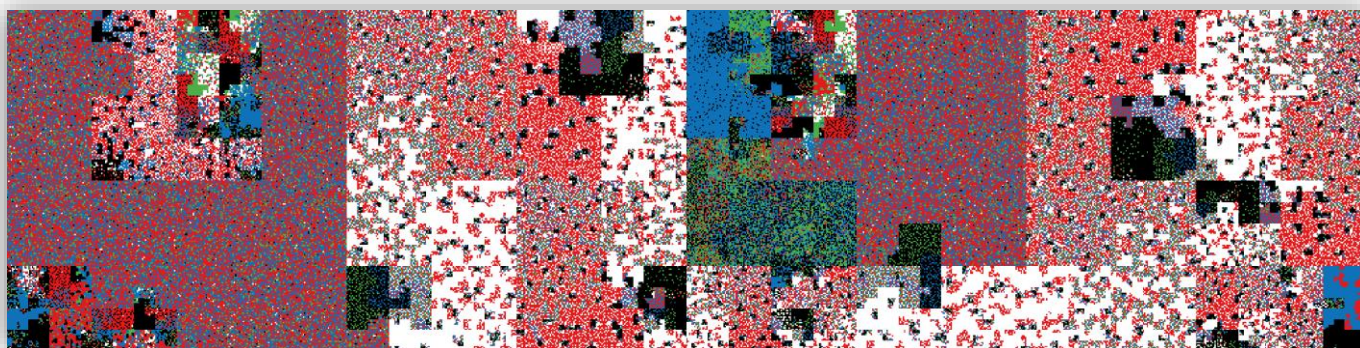
```

MalwareDatabase -- hexedit YouAreAndiot.exe -- 179x38
~/Desktop/MalwareDatabase -- hexedit YouAreAndiot.exe

00000000  5A 90 00 03 00 00 04 00 00 FF FF 00 8B 00 00 00 00 00 00 40 00 00 00 00 00 00 MZ.....@.....
00000024  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0E 1F BA E0 00 B4 09 CD .....
00000040  21 B8 01 4C CD 21 54 68 69 73 20 70 72 F6 67 72 61 6D 20 63 74 20 62 65 20 72 75 6E !..L!This program cannot be run in
0000006C  44 4F 53 20 6D 6F 64 25 00 00 0A 24 00 00 00 00 00 00 50 45 00 00 4C 01 B4 AC DOS mode...$....PE..L..xq.L...
00000090  00 00 00 00 E0 00 02 01 00 01 08 00 00 EC 03 00 00 82 02 00 00 00 00 EE 0A 04 00 00 20 00 00 00 20 04 00
000000B4  00 00 40 00 00 20 00 00 00 02 00 00 04 00 00 00 00 00 00 00 00 00 00 20 07 00 00 04 00 00 ..@.....
000000D8  00 00 00 00 02 00 40 85 00 00 10 00 10 10 00 00 00 10 00 00 10 00 00 00 00 00 00 00 00 @.....
000000FC  00 00 00 00 94 0A 04 00 57 00 00 00 40 04 00 28 AC 02 00 00 00 00 00 00 00 00 00 00 00 W....@.(.....
00000120  00 00 07 00 0C 00 00 00 00 20 04 00 1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..W....@.....
00000144  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 08 00 00 00 00 00 00 00 ..@.....
00000168  00 20 00 00 48 00 00 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 F4 EA 03 00 00 20 00 00 00 EC 03 00
0000018C  00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 74 65 78 74 00 00 A6 00 00 00 00 20 04 00
000001B0  00 02 00 00 00 F0 03 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0 2E 72 73 72 63 00 00 00 28 AC 02 00
000001D4  00 04 04 00 00 AE 02 00 00 F2 03 00 00 00 00 00 00 00 00 00 00 40 00 00 40 2E 72 65 6C 6F 03 00 00 @.....
000001F8  0C 00 00 00 00 07 00 00 02 00 00 00 A0 06 00 00 00 00 00 00 00 00 00 00 00 40 00 00 42 00 00 00 ..@.....
0000021C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 @.....
00000240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
00000264  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
00000288  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
000002AC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
000002D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
000002F4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
00000318  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
0000033C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
00000360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
00000384  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
000003A8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
000003CC  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
000003F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 D0 0A 04 00 00 00 00 48 00 00 00 02 00 05 00 2C CB 03 00
00000414  68 3F 00 00 03 00 00 00 03 00 00 06 50 20 00 6B 82 03 00 00 00 00 00 00 00 00 00 00 00 00 h?.....P .K.....
00000438  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..@.....
0000045C  91 00 00 00 6C 53 79 73 74 65 6D 2E 52 65 73 6F 75 72 63 65 33 73 2E 52 65 73 6F 75 72 63 65 52 65 72
00000480  2C 20 6D 73 63 6F 72 6C 69 62 2C 56 65 72 73 69 6F 6E 3D 74 2E 30 2E 30 2E 30 2E 43 75 6C 74 75 72 65
000004A4  3D 6E 65 75 74 72 61 61 6C 2C 20 50 75 62 6C 69 63 4B 65 79 64 6F 68 65 6C 30 62 37 37 61 35 63 35 36 31 39 33
000004C8  34 65 30 38 39 23 53 79 73 74 65 6D 2E 52 65 73 6F 75 72 63 65 73 2E 52 75 6E 74 69 60 65 52 65 73 6F 75 72
000004EC  63 65 53 65 74 02 00 00 00 00 00 00 00 00 00 50 41 44 50 41 44 50 84 00 00 DE 0E 00 CE CA EF BE 4e089\System.Resources.RuntimeResour
00000510  01 00 00 00 91 00 00 00 6C 53 79 73 74 65 6D 2E 52 65 73 6F 75 72 63 65 73 6F 75 72 63 65 52 65 52 65
----- YouAreAndiot.exe -----0x0/0x6A200-0x

```

ההצגה הבינארית שלו



כיצד זיהוי בעזרת הצגה בינארית פותר את החסרונות של אנטי-וירוס מסורתי?
בעזרת הצגה בינארית אפשר להמיר את כל קבצי ההרצה (exe) לתמונות PNG וליצור מכונה שתלמד לזהות תוכנות זדוניות ועל פי [מחקר שנערך באנגליה](#), מכונות מסוג זה מצליחות לזהות תוכנות זדוניות שעוד לא נראו במאגרים של האנטי-וירוסים הרגילים ועם תוצאות יותר טובות מהאנטי-וירוסים הרגילים.

קשיים עם ההצגה הבינארית במהלך העבודה
את האלגוריתם של ההצגה הבינארית לקחתי מהאתר binvis.io מכיוון שבמחקר שעבדתי על פיו צוין שזה האתר שהם השתמשו בו כדי להמיר את הוירוסים לתמונות אך מכיוון שהמאגר שלי כולל 858 קבצים רציתי למצוא דרך יעילה יותר להמיר את התמונות ונתקלתי בחשבון הגיטהאב של יוצר האתר ובו היה [repository](#) עם הקוד להמרת התמונות.
הבעיה בקוד של היוצר הוא שהוא לא עודכן מאז 2015 ולכן גרסת ה-python שלו לא הייתה עדכנית (2.7) וחלקים בקוד היו חסרים ולכן הייתי צריך לשכתב את רוב הקוד מחדש על מנת שיתאים לגרסה החדשה של python ולהוסיף חלקים שהיו חסרים, תוכלו לראות את התוצאה [כאן](#) ואת מבנה הקוד [כאן](#).



אתגרים מרכזיים

בעיה המרכזית איתה אני צפוי להתמודד היא החוסר בנתונים, לצערי בניגוד לרוב הדברים באינטרנט, תוכנות זדוניות פעילות הן משהו שאנשים לא אוהבים לשתף במאגרים גדולים ולכן הייתי צריך ללקט ממאגרים שונים, הגדולים ביותר הם מאגר של משתמש גיטהאב בשם [Vichingo455](#) ושל משתמש גיטהאב בשם [Endermanch](#), בסך הכל קיימים במאגר שלי 218 תוכנות זדוניות ואת שאר המאגר מרכיבים תוכנות רגילות שקיימות על המחשב שלי. האתגר העיקרי שנובע מכך הוא החשש בפגיעה באיכות הלמידה ובתוצאות המודל וזה ישפיע על בחירת המודל מתאים.

בעיה נוספת היא שהמאמר שבחרתי לעבוד על פיו לקה בחסר בכל מה שקשור למודל בו הם ישתמשו, חוץ מציון שמו – SOINN (Self-Organizing Incremental Neural Network), רשת מסוג הלמידה unsupervised learning

Supervised Learning – סוג למידת מכונה שבשלב האימון המודל משתמש במאגר נתונים מקוטלג, משמע, לכל input (למשל תמונות של חיות) יש את ה-output שהמכונה אמורה להוציא (למשל שם החיה).

Unsupervised Learning – סוג למידת מכונה שבשונה מ-supervised learning, מאגר הנתונים בו משתמש המודל ללמידה אינו מקוטלג והמודל יוצר את ההפרדה והקטלוג לבד (לדוגמה KNN), חסרונות עיקריים הם הסיבוך של רשת כזו שבדרך כלל יותר מסובכת מרשת מסוג supervised learning והצורך במאגרי מידע גדולים יותר לשלב הלמידה.

SOINN – Self Organizing Incremental Neural Network
רשת זאת היא מסוג unsupervised learning, שמה שמייחד אותה הוא שלא צריך לקבוע מראש את מבנה הרשת, הרשת מסוגלת להתאים ולסדר את עצמה על פי הנתונים שהיא מקבלת.

בנוסף ישנה עוד בעיה שאיתה אני צפוי להתמודד במהלך הפרויקט, והיא הורדת התוכנות הזדוניות הפעילות על המחשב האישי שלי מכיוון שהן יכולות לפגוע בו.



הפתרון לאתגר

- בשל החוסר בנתונים למהלך הלמידה של המודל וחוסר התייעוד והסיבוך של הרשת בה רציתי להשתמש, החלטתי לבחור רשת אחרת שהדרישות שלי ממנה הם:
1. רשת בעלת תיעוד רחב באינטרנט על מנת שמהלך המחקר יהיה יותר מהיר ונוח.
 2. רשת שמסוגלת להנפיק תוצאות טובות גם עם חוסר בנתונים.

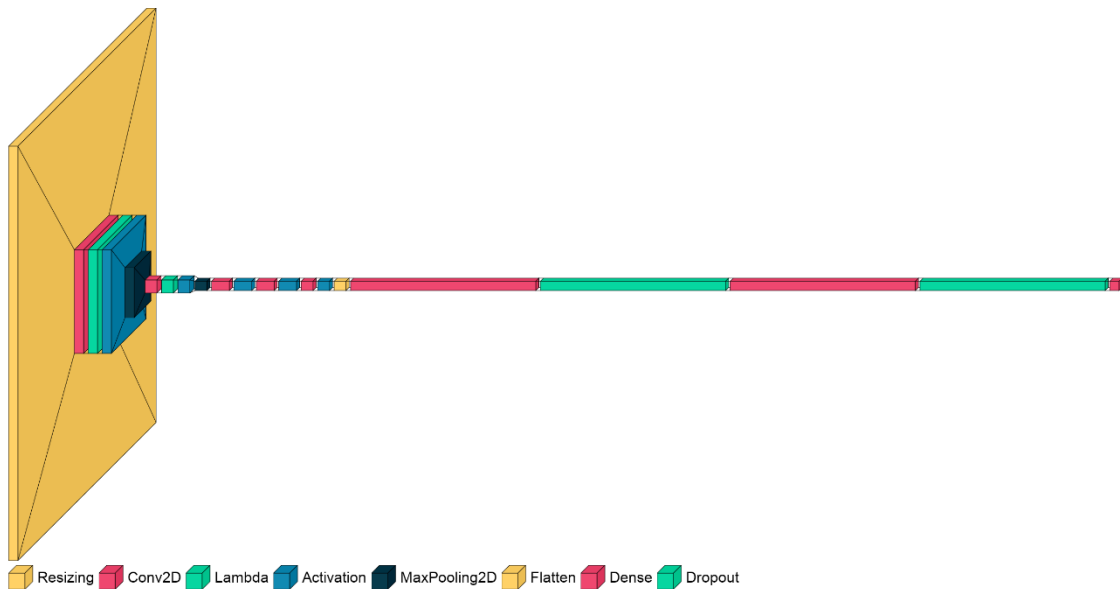
הרשת שעל פי מחקר בגוגל עונה על דרישותיי היא רשת הקונבולוציה המפורסמת, "AlexNet"

AlexNet – הרשת נוצרה על ידי בחור בשם אלכס קריז'בסקי שהתחרה בתחרות "ImageNet Large Scale Visual Recognition Challenge", הרשת שלו הוכיחה את עצמה כיעילה ובעלת תוצאות מעולות בשל הארכיטקטורה שלה, שכן מחקרו הראה שהורדה של אפילו שכבה אחת מהרשת פוגעת בתוצאות בצורה דרסטית.

בנוסף, על מנת לפתור את הבעיה שהתוכנות הזדוניות יכולות להזיק למחשב שלי, הורדתי אותן על מערכת הפעלה וירטואלית מבודדת על המחשב, המרתי שם את כל התוכנות לתמונות ומחקתי את התוכנות מהמחשב.



מבנה המודל



מבנה המודל נקלח ממדריך באינטרנט והוא כולל את השכבות הבאות:

1. שכבת קונבולוציה בעלת 96 ניוונים
2. שכבת נורמליזציה
3. שכבת 3x3 MaxPooling
4. שכבת קונבולוציה בעלת 256 ניוונים
5. שכבת נורמליזציה
6. שכבת 3x3 MaxPooling
7. שכבת קונבולוציה בעלת 384 ניוונים
8. שכבת קונבולוציה בעלת 384 ניוונים
9. שכבת קונבולוציה בעלת 256 ניוונים
10. שכבת שיטוח (Flatten)
11. שכבת Dense בעלת 4096 ניוונים
12. שכבת Dense בעלת 4096 ניוונים
13. שכבת Dense עם אקטיבציית Softmax בעלת 2 ניוונים

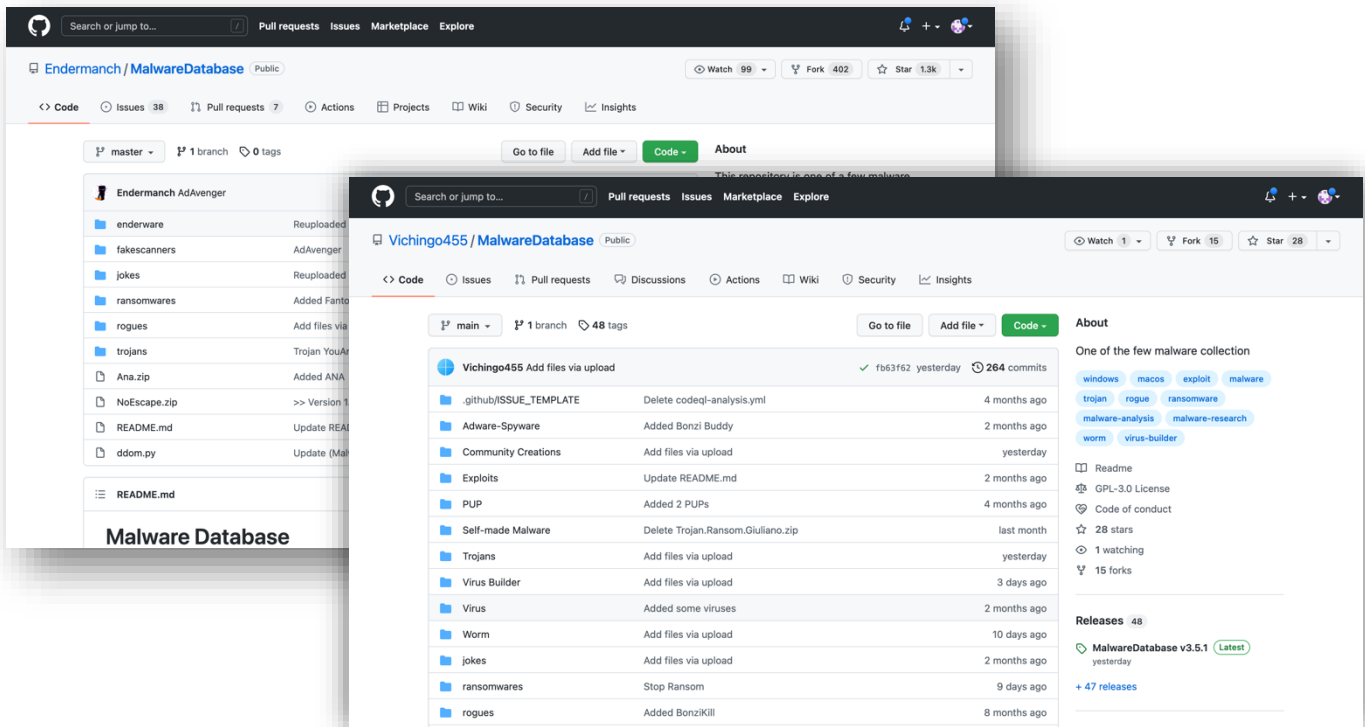
*כל השכבות חוץ מהאחרונה אותחלו בפונקציית האקטיבציה Relu



מבנה הפרויקט

איסוף הנתונים

בהמשך להסבר על הנתונים במבוא, את מאגר התוכנות הזדוניות יצרתי משילוב של שני חשבונות גיטהאב שונים - [Vichingo455](#) ו-[Endermanch](#) על מנת שיהיו לי יותר נתונים ובנוסף להבטיח גיוון ושוני בין התוכנות, שכן הן נוצרו על ידי אנשים שונים בדרכים שונות.



המאגר כלל קבצי exe של תוכנות זדוניות פעילות, משמע, אם בטעות אפתח אחת מהן על המחשב האישי שלי אני עלול להרוס אותו והדרך היחידה להתגבר על כך תהיה לפרמט אותו. בשל החשש הזה הורדתי את התוכנה VMware, תוכנה ליצירת "מכונות וירטואליות".

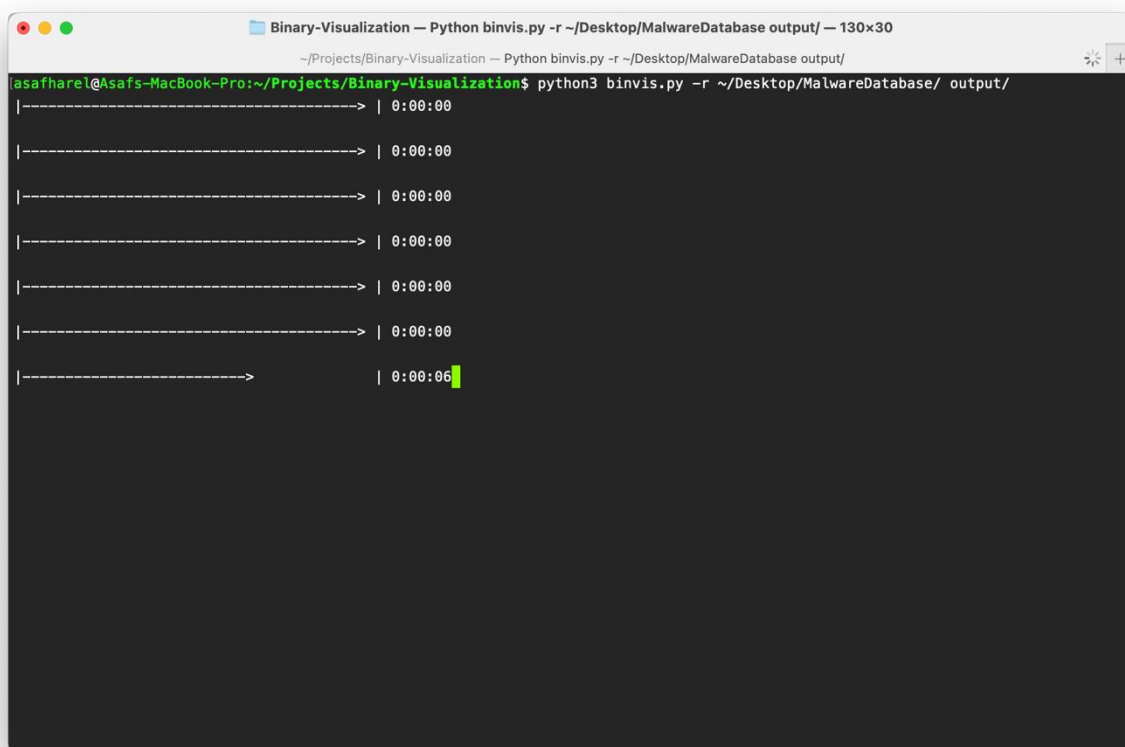
מכונה וירטואלית – מערכת הפעלה אשר מוקצב לה חלק מהמשאבים והזיכרון של המחשב והיא לא יכולה לגשת מחוץ למה שהוקצב לה, בעיקרון זה כמו עוד "מחשב" שרץ על המחשב.

לאחר שפתחתי מכונה וירטואלית יכולתי להוריד את התוכנות ללא דאגה מפגיעה במחשב שלי מכיוון שאם בטעות משהו יקרה, זה יפגע אך ורק במכונה הוירטואלית.



הכנה וניתוח הנתונים

לאחר שכל התוכנות ירדו הרצתי את הכלי שפיתחתי [להמרת התוכנות לתמונות](#), ושמתי את כולן בתיקייה שנקראת "malware", השגת התוכנות הרגילות בשביל למידת המודל הייתה הרבה יותר פשוטה, בשביל להשיג את התמונות שלהן פשוט הרצתי את הכלי להמרת התוכנות על כל המחשב שלי וברגע שהכלי ראה קובץ עם סיומת exe הוא המיר אותו לתמונה. את התוכנות הרגילות שמרתי בתיקייה שנקראת "normal".



```
Binary-Visualization — Python binvis.py -r ~/Desktop/MalwareDatabase output/ — 130x30
~/Projects/Binary-Visualization — Python binvis.py -r ~/Desktop/MalwareDatabase output/
asafharel@Asafs-MacBook-Pro:~/Projects/Binary-Visualization$ python3 binvis.py -r ~/Desktop/MalwareDatabase/ output/
|-----> | 0:00:00
|-----> | 0:00:00
|-----> | 0:00:00
|-----> | 0:00:00
|-----> | 0:00:00
|-----> | 0:00:00
|-----> | 0:00:06
```



התמונות שנוצרו הן בגודל של 256x1024 אך מכיוון שהרשת שלי מקבלת תמונות בגודל 224x224 אז כל התמונות עוברות resize כדי להיות בגודל 220x220 ואז מקבלות padding של 2x2 כדי להפוך להיות בגודל הנכון.

בנוסף לשינוי הגודל כך שיתאים לרשת, מתבצעת נורמליזציה לערכי ה-RGB של התמונה שיהיו בין 0 ל-1 על מנת להקל על תהליך החישוב ולהפוך אותו למהיר ויעיל יותר.

```
X.append(np.array(Image.open(image_path).resize((220, 220))))
```

```
x_train = tf.pad(x_train, [[0, 0], [2, 2], [2, 2], [0, 0]]) / 255
x_test = tf.pad(x_test, [[0, 0], [2, 2], [2, 2], [0, 0]]) / 255
```

על מנת לקטלג את הנתונים לתהליך הלמידה של הרשת על כל תמונה שהתווספה למערך של התמונות ששמו X התווסף המספר 0 או 1 למערך התוויות Y על פי שם התיקיה ממנה הגיעה התמונה (0 – malware, 1 – normal).

```
CLASS_NAMES = os.listdir(data_path)
classes = {}

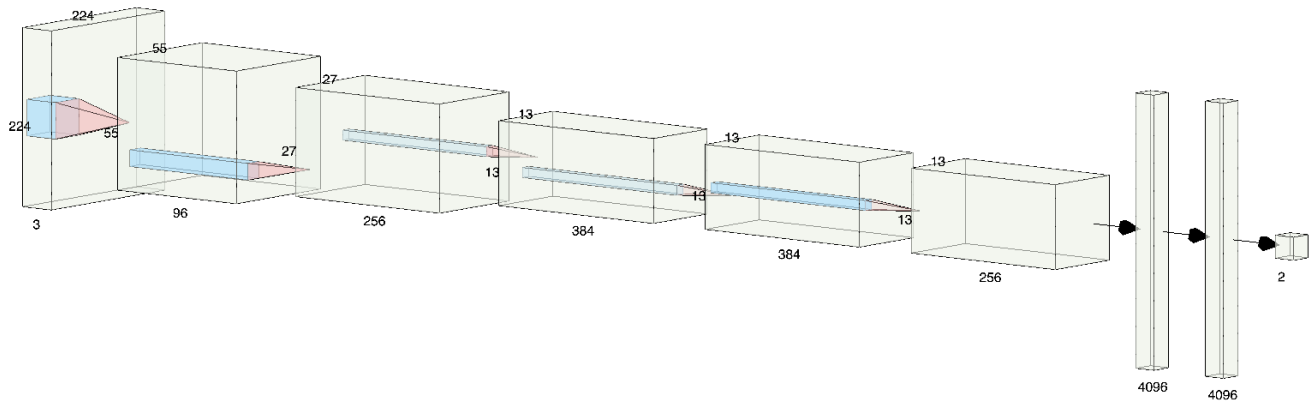
for i in range(len(CLASS_NAMES)):
    classes[CLASS_NAMES[i]] = i

X = []
Y = []
for class_name in CLASS_NAMES:
    for file in os.listdir(path.join(data_path, class_name)):
        if '.png' in file:
            image_path = path.join(data_path, class_name, file)
            X.append(np.array(Image.open(image_path).resize((220, 220))))
            Y.append(classes[class_name])
```



בניית המודל

מודל ה-AlexNet הוא בעל מבנה קבוע שבו רק השכבה האחרונה (שכבת ה-output) משתנה לפי כמות הפלטים שהרשת אמורה להוציא:



המודל נכתב בספריית TensorFlow, TensorFlow2.0 היא ספריית קוד פתוח (open-source) שנועדה ללמידת מכונה ולמידה עמוקה, אני בחרתי להשתמש בה בשל התמיכה הרחבה בה ברחבי האינטרנט וקהל המשתמשים הגדול שגורם לה להיות מעודכנת, יעילה ונטולת באגים.

שכבה דחוסה/מחוברת (Dense\Fully connected) – שכבת נירונים פשוטה, מקבלת את הקלט מהשכבה הקודמת מבצעת חלחול קדימה ואחורה מוציאה את הפלט לשכבה הבאה, הסיבה ששכבה זו נקראת "fully connected" היא שבניגוד למשל לשכבת קונבולוציה, כל נירון בשכבה מחובר לכל נירון הקודמת והשכבה שאחריה.

שכבת קונבולוציה (Convolution) – שכבה שבדרך כלל משמשת לרשתות של זיהוי תמונות מכיוון שבזיהוי תמונות, ככל שהתמונה גדולה יותר, כך מספר הפרמטרים גדל ודרוש יותר כוח מחשוב ורשת יותר מסובכת כדי להגיע לתוצאות טובות. הקונבולוציה נוצרה כדי להקטין את מספר הפרמטרים משמעותית תוך שמירה על מאפייני התמונה.

שכבת MaxPooling – מעין שכבת מעבר שגם היא נוצרה להקטנת מספר הפרמטרים בשביל להאיץ את החישוב, להקטין את מספר הקלטים לשכבה הבאה ולשמור על מאפייני הקלט מהשכבות הקודמות.

שכבת שיטוח (Flatten) – כאשר אנחנו רוצים לעבור משכבות קונבולוציה לשכבות fully connected רגילות אנחנו נשתמש בשכבה זו על מנת לשטח את ממדי הקלט לממד אחד.



ניתן לראות את סיכום המודל בתמונה הבאה:

Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 56, 56, 96)	34944
lambda (Lambda)	(None, 56, 56, 96)	0
activation (Activation)	(None, 56, 56, 96)	0
max_pooling2d (MaxPooling2D)	(None, 27, 27, 96)	0
conv2d_1 (Conv2D)	(None, 7, 7, 256)	614656
lambda_1 (Lambda)	(None, 7, 7, 256)	0
activation_1 (Activation)	(None, 7, 7, 256)	0
max_pooling2d_1 (MaxPooling2D)	(None, 3, 3, 256)	0
conv2d_2 (Conv2D)	(None, 1, 1, 384)	885120
activation_2 (Activation)	(None, 1, 1, 384)	0
conv2d_3 (Conv2D)	(None, 1, 1, 384)	1327488
activation_3 (Activation)	(None, 1, 1, 384)	0
conv2d_4 (Conv2D)	(None, 1, 1, 256)	884992
activation_4 (Activation)	(None, 1, 1, 256)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 4096)	1052672
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 2)	8194
Total params: 21,589,378		
Trainable params: 21,589,378		
Non-trainable params: 0		



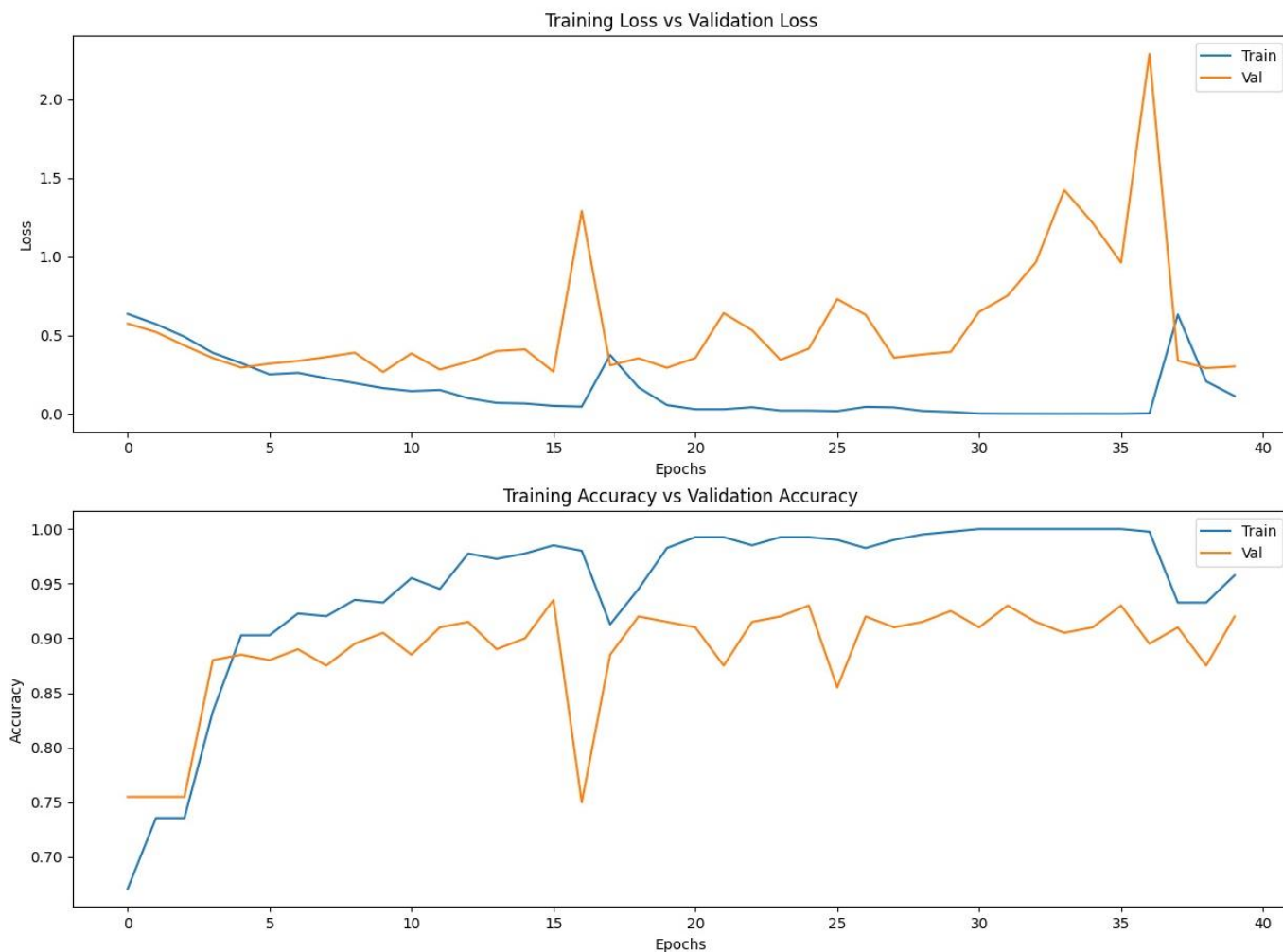
איטרציה #1

תחילה ניסיתי להריץ את המודל כמו שראיתי שהריצו אותו [באתר ממנו לקחתי השראה](#):

40 – Epochs

64 – Batch Size

התוצאות:



[0.3464984893798828, 0.8093385100364685]

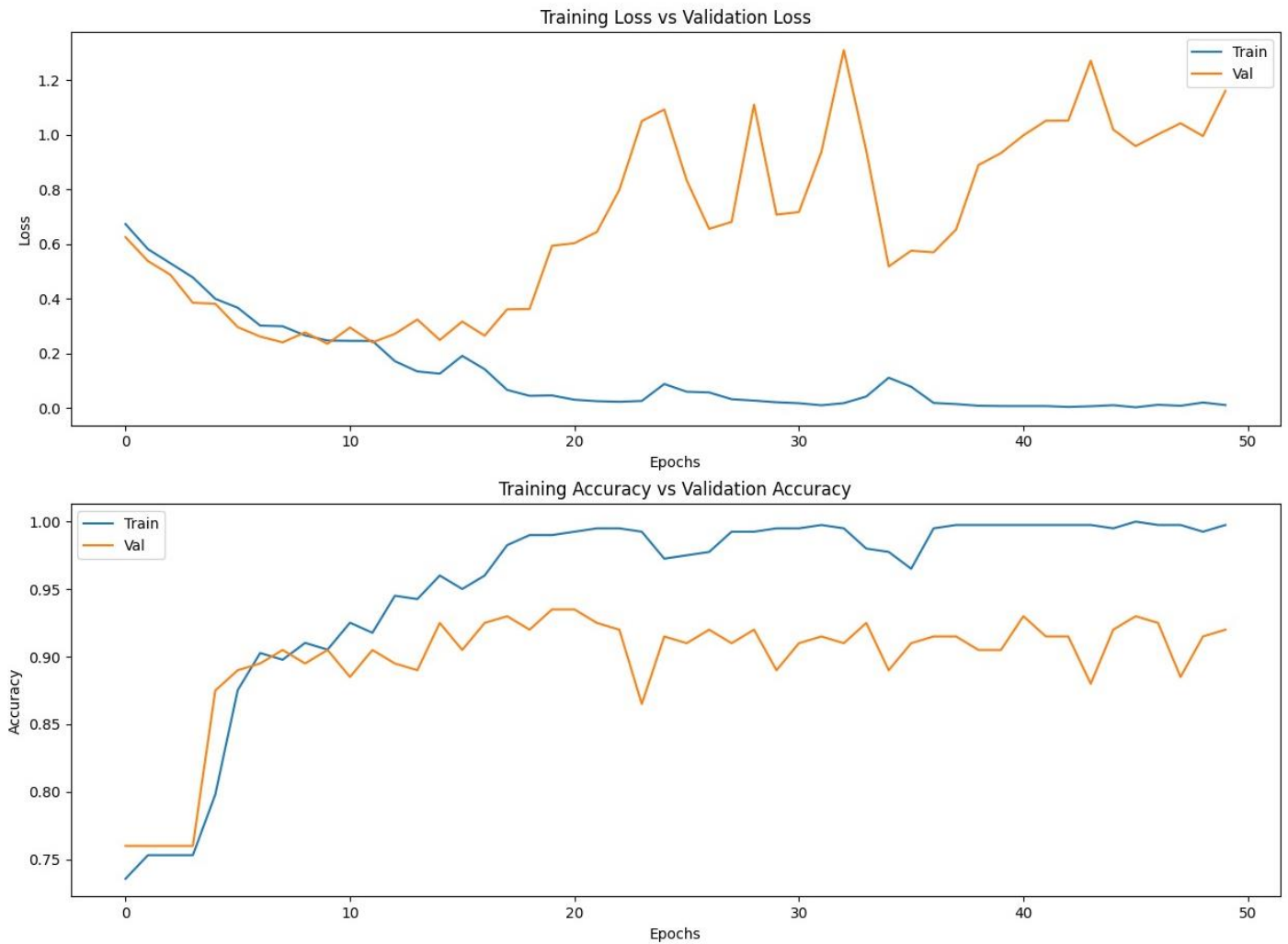
תוצאה: 81 אחוזי הצלחה

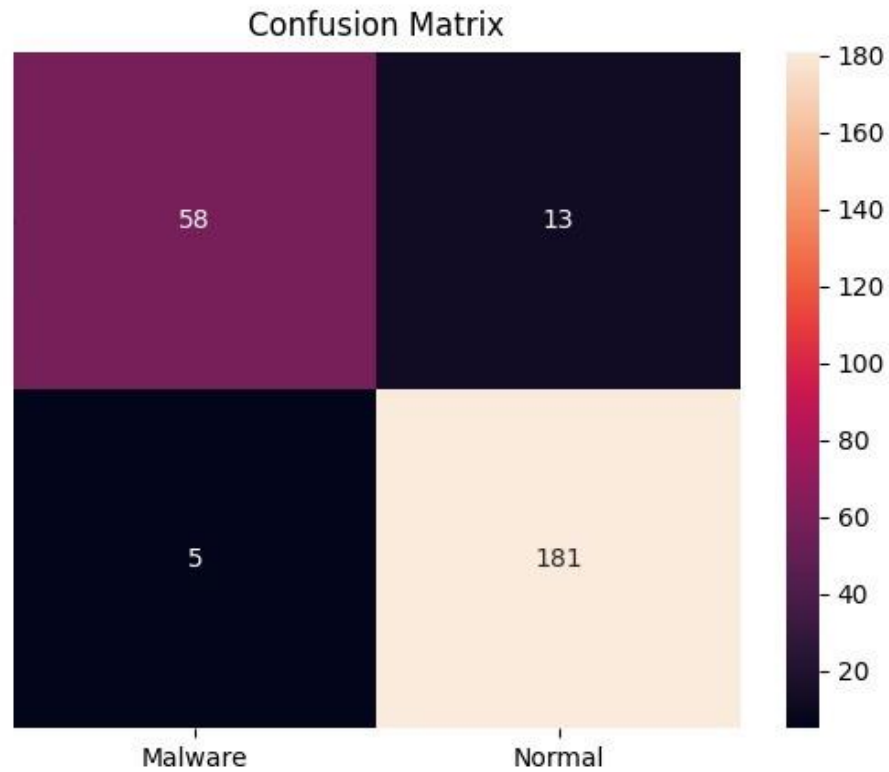


איטרציה #2

תוכנת אני-וירוס צריכה להיות בעלת אמינות גבוהה ולכן העלתי את מספר ה-epochs על מנת לבדוק אם שינוי זה יוביל לשיפור בביצועי הרשת.

50 – Epochs
64 – Batch Size





[0.7285164594650269, 0.929961085319519]

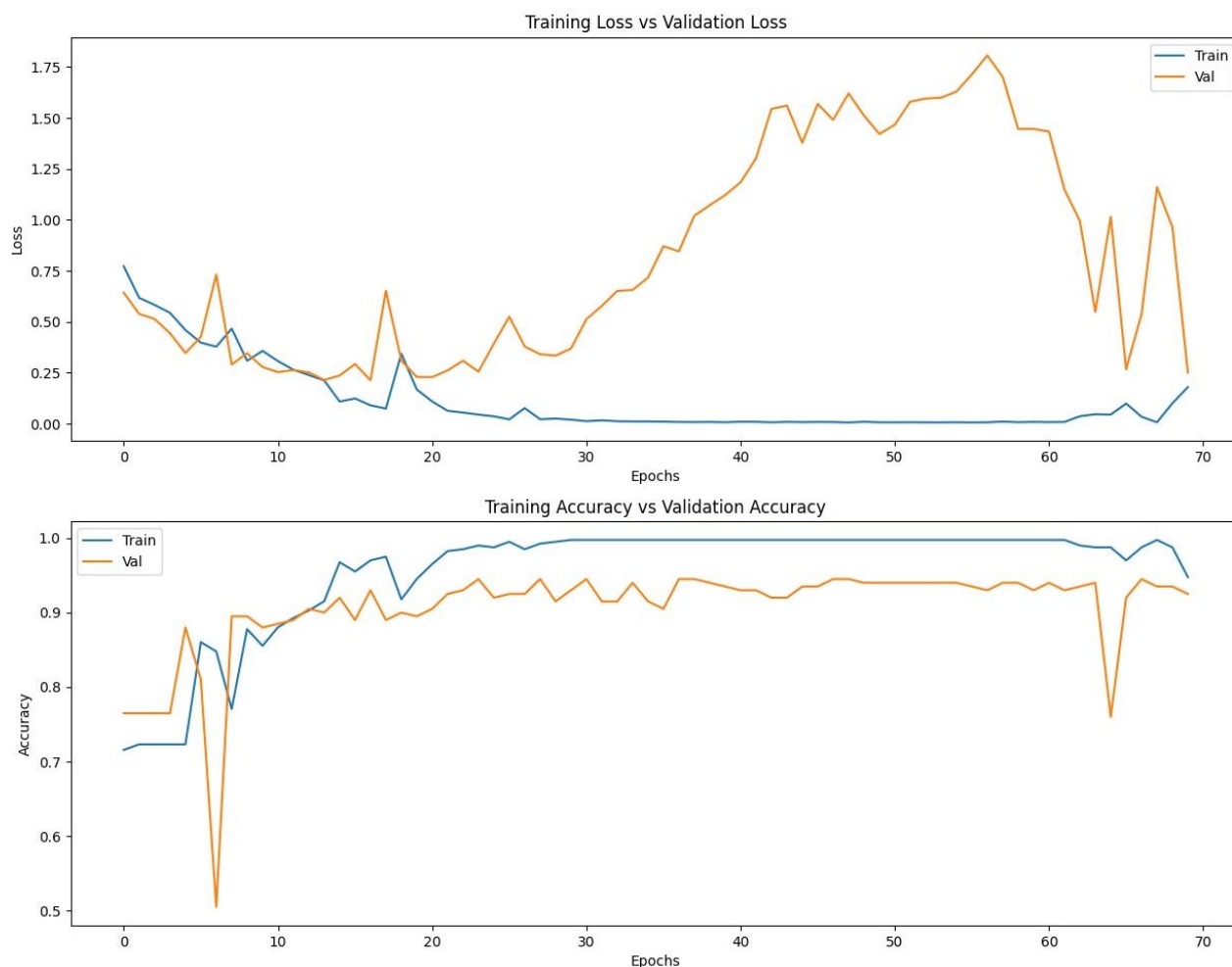
תוצאה: 93 אחוזי הצלחה



איטרציה #3

93 אחוזי הצלחה זו תוצאה מצוינת אך מתוך סקרנות בדקתי כיצד הרשת תגיד אם אוסיף לה אפילו יותר epochs.

70 – Epochs
64 – Batch Size



[0.2247806340456009, 0.9182879328727722]

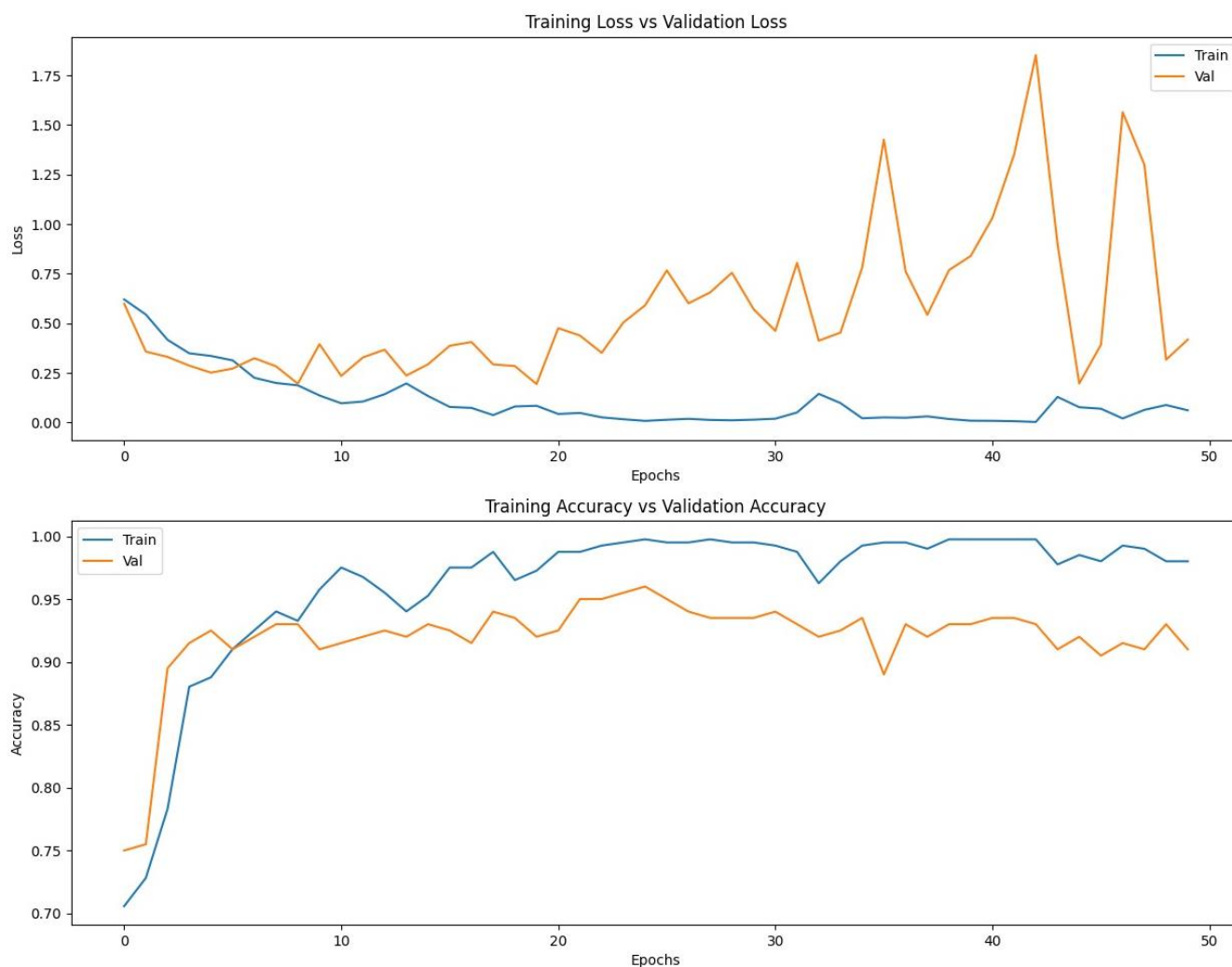
תוצאה: 92 אחוזי הצלחה



איטרציה #4

לאחר שראיתי שבמפתיע מספר כה קטן של רק 50 epochs מפיק את התוצאות הכי טובות, ניסיתי לראות האם גם הקטנת מספר הקבוצות (ה-Batch Size) תשפיע לטובה על ביצועי הרשת.

50 – Epochs
32 – Batch Size



```
[0.3465675711631775, 0.9260700345039368]
```

תוצאה: 93 אחוזי הצלחה



סיכום שלב האימון

במפתיע שלב האימון לא לקח הרבה מאוד זמן ואיטרציות ובשניים מתוך ה-4 שביצעתי קיבלתי את התוצאות הטובות ביותר (93% אחוזי הצלחה בזיהוי) אך [איטרציה מספר 2](#) הפיקה תוצאות קצת יותר טובות אז בחרתי להשתמש במשקולות שיצאו ממנה.

ה-hyper paramters של המודל:

- 50 – Epochs
- 64 – Batch Size
- אתחול משקולות – Xavier
- ייעול התכנסות – Adam
- קצב הלמידה – 0.001
- פונקציית שגיאה – Sparse Categorical Cross Entropy

פונקציית השגיאה (CCE) Categorical Cross Entropy

על מנת להבין כיצד פונקציית השגיאה Sparse Categorical Cross Entropy בה השתמשתי למודל עובדת, נצטרך להבין כיצד Categorical Cross Entropy רגילה עובדת. הפונקציה יעילה מכיוון שכמצופה מפונקציית שגיאה של הרשת, ככל שהחיזוי (\hat{y}) רחוק מהתשובה האמתית (y) פונקציית השגיאה גדלה. פונקציית השגיאה זו משמשת כאשר הרשת צריכה לבצע חיזוי שכולל כמה תוצאות ושהקלט הוא מערכים שנקראים one-hot כאשר בכל תא יש את המספר 0 או 1 למשל, אם הייתה לנו רשת שמזהה בין חתול, לכלב ולשועל ובקלט שלה הייתה התמונה הזו:



ה-Y היה נראה כך $[0, 1, 0]$ שכן ה-1 במקום השני במערך מכיוון שמדובר בכלב ולא בחתול או שועל.

פונקציית השגיאה (SCCE) Sparse Categorical Cross Entropy

ההבדל היחיד בינה לבין פונקציית ה-Categorical Cross Entropy הרגילה היא שבמקום לעבוד עם one-hot's היא עובדת עם מספרים יחידים, למשל בדוגמה למעלה ה-Y יהיה $[2]$.



החיסרון היחיד של SCCE על CCE הוא שאי אפשר לדעת מה הרשת חשבה על האפשרויות האחרות מכיוון שב-CCE בפלט יהיה משהו בסגנון של $[0.02, 0.8, 0.495]$ וב-SCEE הפלט יהיה פשוט [2] אך מכיוון שדרך זו חוסכת יותר במשאבים ורק התוצאה הגדולה ביותר מעניינת אותי החלטתי להשתמש ב-Sparse Cross Entropy.

ייעול התכנסות (Optimization) – Adam

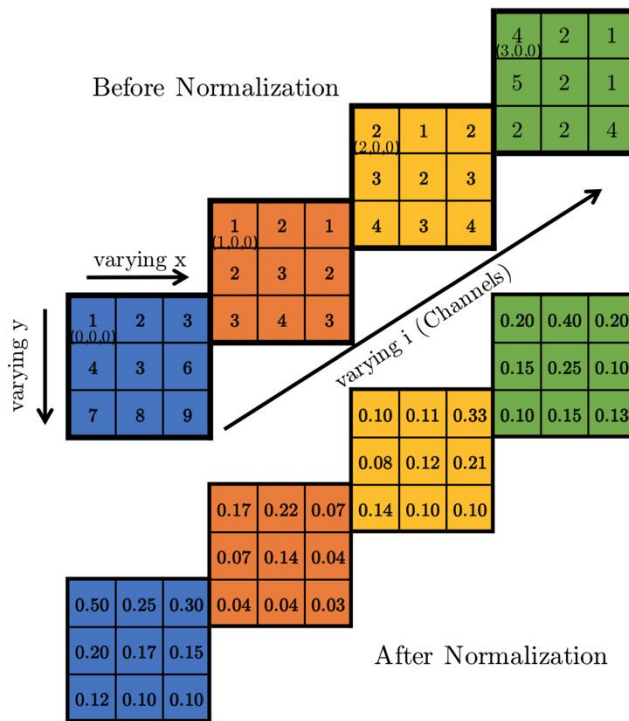
ייעול התכנסות שכן ברור מהשם שלו, נועד לייעול ולקצר את זמן הריצה של אימון המודל על מנת לחסוך זמן ומשאבים. Adam משלב בתוכו שני אלגוריתמים של ייעול התכנסות, RMSProp ומומנטום, אל שניהם לא אכנס בפרטי פרטים אך אסביר איך Adam, השילוב של שניהם עובד. היתרון של Adam הוא שהוא משנה את ה-hyper parameters בזמן אמת, למשל, את קצב הלמידה הוא מקטין ככל שמתקרבים למינימום בפונקציית ה-gradient decent על מנת להתכנס יותר טוב ויותר במהירות לנקודה. בנוסף, בזמן החלחול לאחר, Adam מיישם גם את החישוב של אלגוריתם מומנטום וגם את החישוב של RMSProp ובכך מתקן בזמן אמת את ההטיה (השגיאה) של הרשת.

התמודדות עם הטיה ושונות

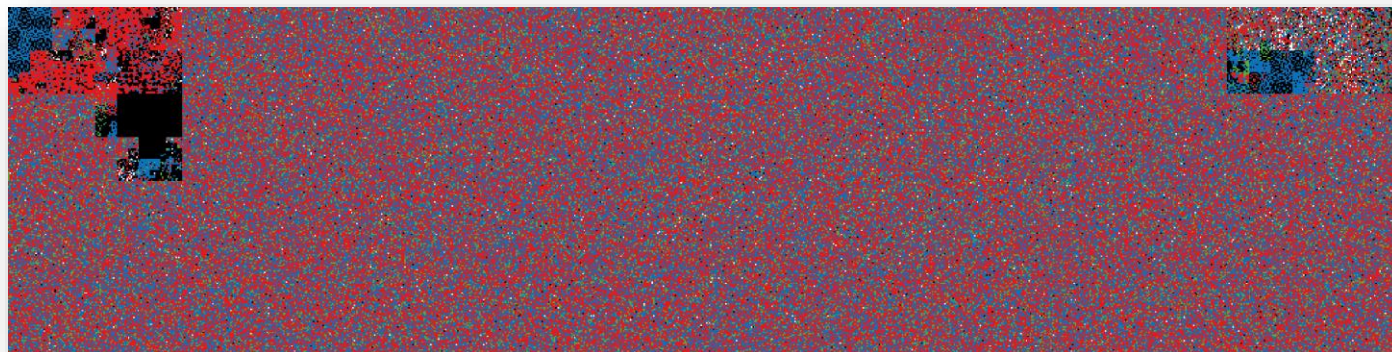
בשביל להימנע מהטיה ושונות כבר ביצירת המודל הראשון השתמשתי ב-mini batches, בפונקציות הסדרה (Regularization) מסוג Local Response Normalization ובייעול ההתכנסות מסוג Adam. בזכות זה במהלך האימון לא נתקלתי בבעיה של הטיה ושונות, מה שאפשר לי להגיע לתוצאות טובות בזמן מאוד קצר.

Local Response Normalization

שיטת נרמול שהוצגה לראשונה ברשת המפורסמת AlexNet שנועדה להקטין את הערכים של הפיקסלים ביחס לשכניהם, לדוגמה:



בסוף שלב האימון, על מנת להבטיח שהמכונה יכולה לזהות תוכנות זדוניות שונות ולא רק את התוכנות מהמאגרים שאני מצאתי, חיפשתי בגוגל עוד תוכנה זדונית רנדומלית שמישהו העלה ובדקתי אם הרשת מצליחה לזהות אותה בתור תוכנה זדונית. את התוכנה לקחתי מ-repositroy בגיטהאב של משתמש בשם [Da2dalus](#) והרצתי את התוכנה שלי על אחד מהקבצים במאגר שלו, התוצאה:



malware

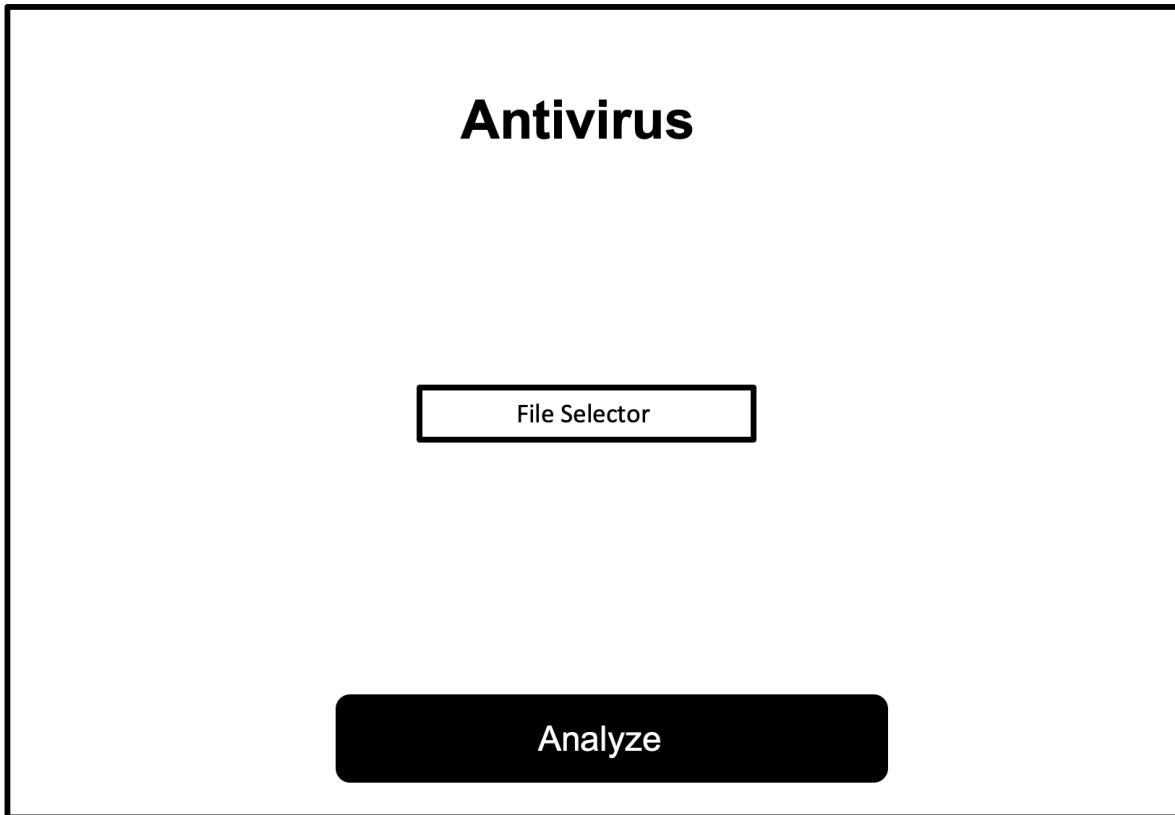
התוצאה:

כעת כאשר הבטחתי את אמינות המודל, ניתן להמשיך לשלב פיתוח ממשק המשתמש.



ממשק המשתמש

ניסיתי לשמור על ממשק המשתמש כמה שיותר פשוט שכן קהל היעד של המוצר הוא כל אדם, בין אם זה איש IT או קשיש שמתקשה עם כל הטכנולוגיות החדשות.

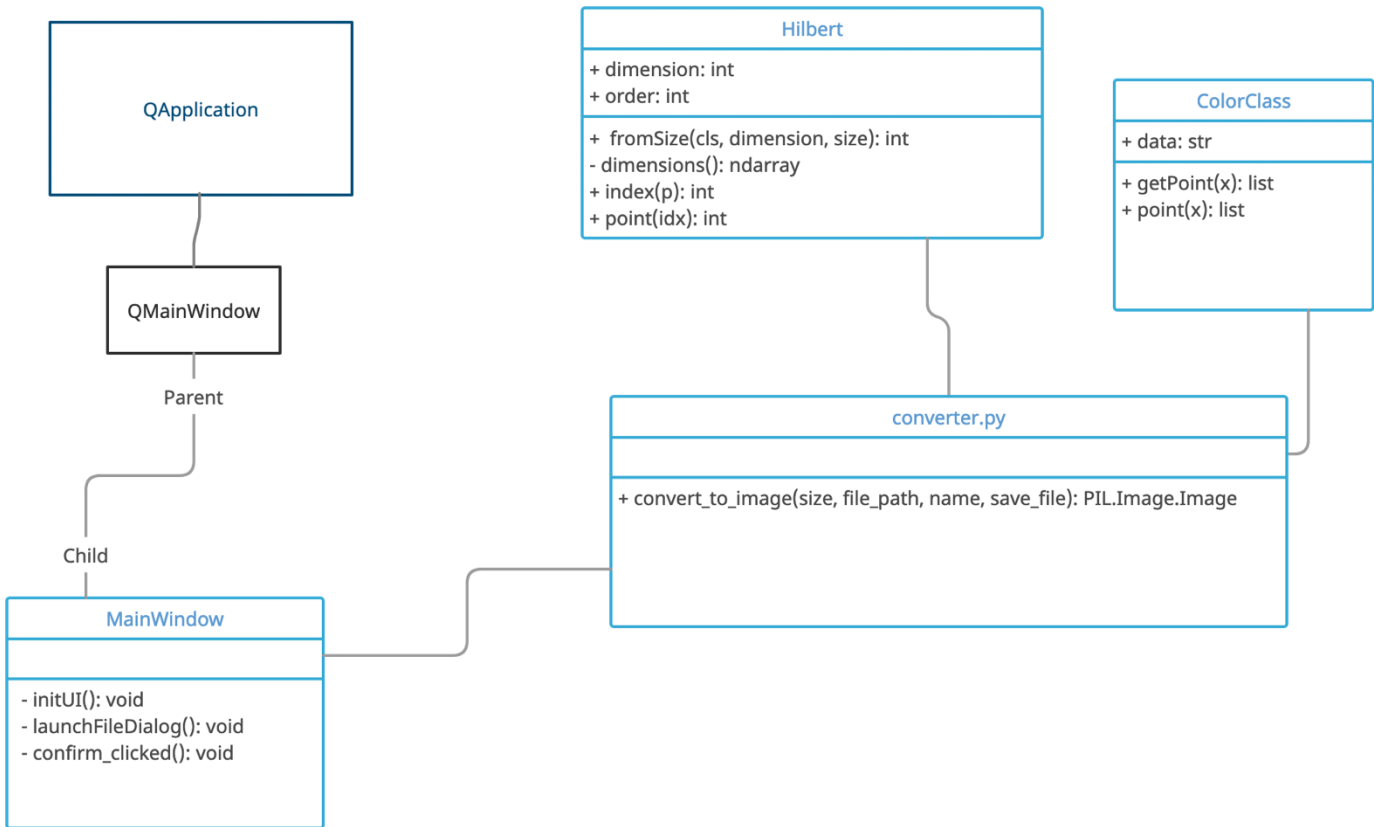


ממשק המשתמש יראה כמתואר למעלה, כפתור בחירה פשוט של קובץ ועוד כפתור ניתוח (Analyze) שיגיד אם מדובר בתוכנה זדונית או תוכנה רגילה.

השימוש המודל בתוך היישום נעשה בקלות, בתחילת הריצה, היישום יוצר את מבנה המודל וטוען לתוכו את המשקולות השמורות [משלב האימון](#). השימוש במודל נעשה רק כאשר המשתמש לוחץ על כפתור הניתוח, [התמונה עוברת את השינויים שהיא צריכה לעבור על מנת להתאים לקלט המודל](#), לאחר מכן היא נכנסת אל המודל שמחזיר [0] אם היא תוכנה זדונית או [1] אם זו תוכנה רגילה, את המספר היישום ממיר לטקסט "malware X" או "normal ☒".



UML של היישום



ספריית PyQt5

ספריית GUI לפייתון המשתמשת ביכולות של פייתון לעבוד עם קבצים בשפה C++ ו-C עליון פייתון בנויה על מנת להבטיח יעילות וריצה מהירה יותר. PyQt היא ממשק לפייתון של ערכת הכלים Qt ליצירת ממשקים גרפיים בתמיכה לכל מערכות ההפעלה, Qt הוא כלי ידוע בקרב התעשייה, בין היתר חברות מוכרות כמו למשל Adobe השתמשו ב-Qt למוצרים שלהן.

כיצד המידע מגיע אל היישום

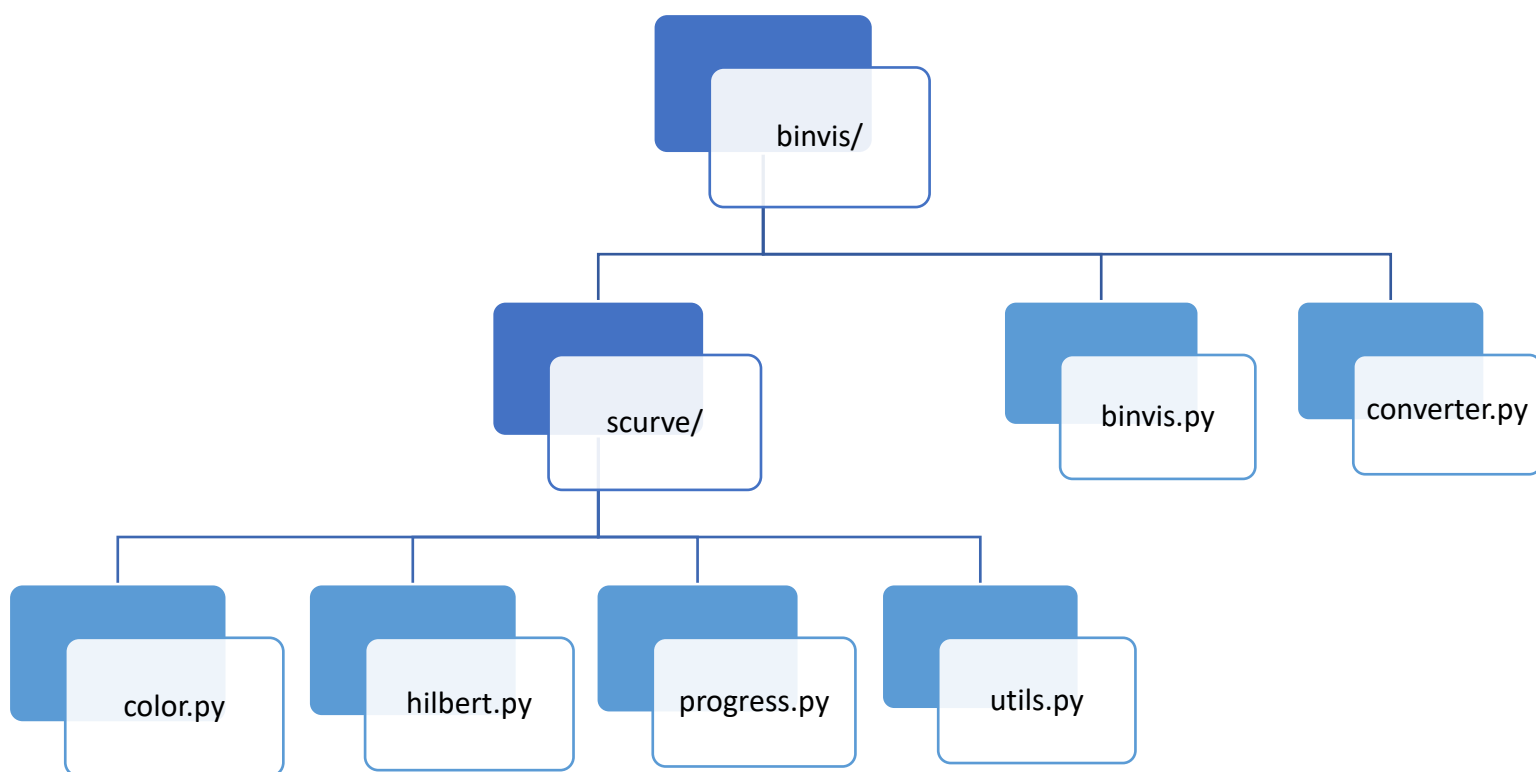
מכיוון שזו היא תוכנה שרצה על המחשב האישי של המשתמש שרוצה לבדוק את בריאות מחשבו, לתוכנה יש גישה לכל הקבצים שהמשתמש יחליט לבדוק על המחשב, המשתמש רק צריך לבחור איזה קובץ הוא רוצה לבדוק, התוכנה מקבלת את מיקום הקובץ במחשב ומיד קוראת אותו וממירה אותו לתמונה, לאחר ההמרה לתמונה התוכנה תשנה את ממדיו כדי שיתאימו לקלט המודל, תנרמל את ערכיה ותכניס אותה אל המודל לקבלת תשובה.



מדריך למפתח

הצגה בינארית

מבנה הפרויקט:



- **utils.py, hilbert.py, color.py** אחראים על ההמרה של הערכים לצבע
- **converter.py** אחראי על שילוב שלושת הקודמים ולהמרת הקובץ לתמונה
- **binvis.py** משתמש ב-converter.py והופך אותו לכלי CLI נוח



:color.py

```
import string

class ColorClass:
    """Convert binary value to RGB values"""
    def __init__(self, data):
        self.data = data # The binary values
        s = list(set(data))
        s.sort()
        self.symbol_map = {v: i for (i, v) in enumerate(s)}

    def point(self, x):
        """Get The RGB Values

        Args:
            x (int): Index of the binary value in the list

        Returns:
            list: The RGB values
        """
        c = self.data[x]

        if c == 0:
            return [0, 0, 0]
        elif c == 255:
            return [255, 255, 255]
        elif (0 < c < 32) and ((c != 9) or (c != 10) or (c != 13)):
            return [104, 172, 87]
        elif chr(c) in string.printable:
            return [55, 126, 184]
        return [228, 26, 28]

    def __len__(self):
        return len(self.data)
```



:hilbert.py

```
from . import utils
import numpy as np

def transform(entry, direction, width, x):
    assert x < 2 ** width
    assert entry < 2 ** width
    return utils.rrot((x ^ entry), direction + 1, width)

def itransform(entry, direction, width, x):
    """
    Inverse transform - simply reverse the operations in transform.
    """
    assert x < 2 ** width
    assert entry < 2 ** width
    return utils.lrot(x, direction + 1, width) ^ entry
    # There is an error in the Hamilton paper's formulation of the inverse
    # transform in Lemma 2.12. The correct restatement as a transform is as
    # follows:
    # return transform(rrot(entry, direction+1, width), width-direction-2, width,
    # x)

def direction(x, n):
    assert x < 2 ** n
    if x == 0:
        return 0
    elif x % 2 == 0:
        return utils.tsb(x - 1, n) % n
    else:
        return utils.tsb(x, n) % n

def entry(x):
    if x == 0:
        return 0
    else:
        return utils.graycode(2 * ((x - 1) / 2))
```



```
def hilbert_point(dimension, order, h):
    """
    Convert an index on the Hilbert curve of the specified dimension and
    order to a set of point coordinates.
    """
    # The bit widths in this function are:
    #   p[*] - order
    #   h   - order*dimension
    #   l   - dimension
    #   e   - dimension
    hwidth = order * dimension
    e, d = 0, 0
    p = [0] * dimension
    for i in range(order):
        w = utils.bitrange(h, hwidth, i * dimension, i * dimension + dimension)
        l = utils.graycode(w)
        l = itransform(e, d, dimension, l)
        for j in range(dimension):
            b = utils.bitrange(l, dimension, j, j + 1)
            p[j] = utils.setbit(p[j], order, i, b)
        e = e ^ utils.lrot(entry(w), d + 1, dimension)
        d = (d + direction(w, dimension) + 1) % dimension
    return p

def hilbert_index(dimension, order, p):
    h, e, d = 0, 0, 0
    for i in range(order):
        l = 0
        for x in range(dimension):
            b = utils.bitrange(p[dimension - x - 1], order, i, i + 1)
            l |= b << x
        l = transform(e, d, dimension, l)
        w = utils.igraycode(l)
        e = e ^ utils.lrot(entry(w), d + 1, dimension)
        d = (d + direction(w, dimension) + 1) % dimension
        h = (h << dimension) | w
    return h
```



```
class Hilbert:
    def __init__(self, dimension, order):
        self.dimension, self.order = dimension, order

    @classmethod
    def fromSize(cls, dimension, size):
        """
        Size is the total number of points in the curve.
        """
        x = np.math.log(size, 2)
        if not float(x) / dimension == int(x) / dimension:
            raise ValueError("Size does not fit Hilbert curve of dimension %s." %
                               dimension)
        return Hilbert(dimension, int(x / dimension))

    def __len__(self):
        return 2 ** (self.dimension * self.order)

    def __getitem__(self, idx):
        if idx >= len(self):
            raise IndexError
        return self.point(idx)

    def dimensions(self):
        """
        Size of this curve in each dimension.
        """
        return [int(np.ceil(len(self) ** (1 / float(self.dimension))))] *
                self.dimension

    def index(self, p):
        return hilbert_index(self.dimension, self.order, p)

    def point(self, idx):
        return hilbert_point(self.dimension, self.order, idx)
```



:progress.py

```
import datetime
import sys
import time

class Inplace:
    def __init__(self, title="", stream=sys.stderr):
        self.stream, self.title = stream, title
        self.last = 0

    def tick(self, s):
        if not self.stream:
            return
        w = "\r%s%s" % (self.title, s)
        self.last = len(w)
        self.stream.write(w)
        self.stream.flush()

    def inject(self, txt):
        self.stream.write("\n")
        self.clear()
        self.stream.write("%s\n" % txt)
        self.stream.flush()

    def clear(self):
        if not self.stream:
            return
        spaces = " " * self.last
        self.stream.write("\r%s\r" % spaces)

class Progress(Inplace):
    """Just a visualization of time remaining for the conversion"""
    bookend = "|"
    done = "-"
    current = ">"
    todo = " "

    def __init__(self, target, title="", width=40, stream=sys.stderr):
        Inplace.__init__(self, title, stream=stream)
        self.width, self.target = width, target
        self.prev = -1
        self.startTime = None
```



```

self.window = None

def tick(self, val):
    if not self.stream:
        return
    if not self.startTime:
        self.startTime = datetime.datetime.now()
    pp = val / float(self.target)
    progress = int(pp * self.width)
    t = datetime.datetime.now() - self.startTime
    runsecs = t.days * 86400 + t.seconds + t.microseconds / 1000000.0
    if pp == 0:
        eta = "?:?:??"
    else:
        togo = runsecs * (1 - pp) / pp
        eta = datetime.timedelta(seconds=int(togo))
    if pp > self.prev:
        self.prev = pp
    l = self.done * progress
    r = self.todo * (self.width - progress)
    now = time.time()
    s = "%s%s%s%s%s %s" % (
        self.bookend, l,
        self.current,
        r, self.bookend, eta
    )
    Inplace.tick(self, s)

def set_target(self, t):
    self.target = t

def clear(self):
    Inplace.clear(self)

def full(self):
    self.tick(self.target)

class Dummy:
    def __init__(self, *args, **kwargs): pass

    def tick(self, *args, **kwargs): pass

    def restoreTerm(self, *args, **kwargs): pass

    def clear(self, *args, **kwargs): pass

```



```
def full(self, *args, **kwargs): pass
def set_target(self, *args, **kwargs): pass
```



:utils.py

```
import numpy as np

def graycode(x):
    x = int(x)
    return x ^ (x >> 1)

def igraycode(x):
    """
    Inverse gray code.
    """
    if x == 0:
        return x
    m = int(np.ceil(np.log(x, 2))) + 1
    i, j = x, 1
    while j < m:
        i = i ^ (x >> j)
        j += 1
    return i

def bits(n, width):
    """
    Convert n to a list of bits of length width.
    """
    assert n < 2 ** width
    bin = []
    for i in range(width):
        bin.insert(0, 1 if n & (1 << i) else 0)
    return bin
```



```
def bits2int(bits):
    """
    Convert a list of bits to an integer.
    """
    n = 0
    for p, i in enumerate(reversed(bits)):
        n += i * 2 ** p
    return n
```

```
def rrot(x, i, width):
    """
    Right bit-rotation.

    width: the bit width of x.
    """
    assert x < 2 ** width
    i = i % width
    x = (x >> i) | (x << width - i)
    return x & (2 ** width - 1)
```

```
def lrot(x, i, width):
    """
    Left bit-rotation.

    width: the bit width of x.
    """
    assert x < 2 ** width
    i = i % width
    x = (x << i) | (x >> width - i)
    return x & (2 ** width - 1)
```

```
def tsb(x, width):
    """
    Trailing set bits.
    """
    assert x < 2 ** width
    i = 0
    while x & 1 and i <= width:
        x = x >> 1
        i += 1
    return i
```



```
def setbit(x, w, i, b):
    """
    Sets bit i in an integer x of width w to b.
    b must be 1 or 0
    """
    assert b in [1, 0]
    assert i < w
    if b:
        return x | 2 ** (w - i - 1)
    else:
        return x & ~2 ** (w - i - 1)

def bitrange(x, width, start, end):
    """
    Extract a bit range as an integer.
    (start, end) is inclusive lower bound, exclusive upper bound.
    """
    return x >> (width - end) & ((2 ** (end - start)) - 1)

def entropy(data, blocksize, offset, symbols=256):
    """
    Returns local byte entropy for a location in a file.
    """
    if len(data) < blocksize:
        raise ValueError("Data length must be larger than block size.")
    if offset < blocksize / 2:
        start = 0
    elif offset > len(data) - blocksize / 2:
        start = len(data) - blocksize / 2
    else:
        start = offset - blocksize / 2
    hist = {}
    for i in data[start:start + blocksize]:
        hist[i] = hist.get(i, 0) + 1
    base = min(blocksize, symbols)
    entropy = 0
    for i in hist.values():
        p = i / float(blocksize)
        # If blocksize < 256, the number of possible byte values is restricted.
        # In that case, we adjust the log base to make sure we get a value
        # between 0 and 1.
        entropy += (p * math.log(p, base))
    return -entropy
```



:converter.py

```
from PIL import Image, ImageDraw

from binvis.scurve.hilbert import Hilbert
from binvis.scurve.color import ColorClass

def convert_to_image(size, file_path, name="", save_file=True):
    """Convert file into image

    Args:
        size (int): The width of the image
        file_path (str): The path to the file
        name (str, optional): The name of the output file. Defaults to ".
        save_file (bool, optional): Either to save the otuput on the computer or
        not. Defaults to True.

    Returns:
        PIL.Image.Image: The image of the file in binary visualization
    """

    with open(file_path, 'rb') as file:
        data = file.read()

    csource = ColorClass(data)
    map = Hilbert.fromSize(2, size ** 2)
    c = Image.new("RGB", (size, size * 4))
    cd = ImageDraw.Draw(c)
    step = len(csource) / float(len(map) * 4)

   sofar = 0
    for quad in range(4):
        for i, p in enumerate(map):
            off = (i + (quad * size ** 2))
            color = csource.point(
                int(off * step)
            )
            x, y = tuple(p)
            cd.point(
                (x, y + (size * quad)),
                fill=tuple(color)
            )

    c_small = c.resize((150, 450))
```



```
if save_file:  
    c_small.save(name)  
  
return c
```



:binvis.py

```
import sys
import os
from os import path
from converter import convert_to_image

class ArgumentError(Exception):
    __module__ = Exception.__module__

    def __init__(self, message=""):
        super().__init__(message)

class DirectoryNotFoundError(Exception):
    __module__ = Exception.__module__

    def __init__(self, message=""):
        super().__init__(message)

def save_file(file_path, out_dir):
    """Save file as image into folder

    Args:
        file_path (str): Original file path
        out_dir (str): The desired directory the file should be saved to

    Raises:
        FileNotFoundError: If there is no such file in the specified 'file_path'
    """
    if path.isfile(file_path):
        if '/' in file_path:
            filename = file_path.split('/')[-1].split('.')[0]
        elif '\\' in file_path:
            filename = file_path.split("\\")[-1].split('.')[0]
        else:
            filename = file_path.split('.')[0]

        dst = path.join(out_dir, '.'.join([filename, 'png']))

        with open(file_path, 'rb') as file:
            content = file.read()

        convert_to_image(256, content, dst)
```



```

else:
    raise FileNotFoundError(f"{file_path}" does not exists')

def save_directory(directory, out_dir):
    """Convert an entire directory into images and save them

    Args:
        directory (str): The directory with all the files
        out_dir (str): The desired directory the files should be saved to

    Raises:
        DirectoryNotFoundError: If the directory specified in 'directory' does not exist
    """
    if path.isdir(directory):
        files_paths = os.listdir(directory)

        if '.DS_Store' in files_paths:
            files_paths.remove('.DS_Store')
        if '.git' in files_paths:
            files_paths.remove('.git')

        for file_path in files_paths:
            file_path = path.join(directory, file_path)
            save_file(file_path, out_dir)
    else:
        raise DirectoryNotFoundError(f"{directory}" does not exists')

def save_directory_recursive(directory, out_dir):
    """Convert every file in the directory and the in the subdirectories into images and save them

    Args:
        directory (str): The directory with all the files
        out_dir (_type_): The desired directory the files should be saved to
    """
    dir_content = os.listdir(directory)

    if '.DS_Store' in dir_content:
        dir_content.remove('.DS_Store')
    if '.git' in dir_content:
        dir_content.remove('.git')

    files_paths = [data_path for data_path in dir_content if '.' in data_path]
    dirs_paths = [data_path for data_path in dir_content if data_path not in files_paths]

    for file_path in files_paths:

```



```

    file_path = path.join(directory, file_path)
    save_file(file_path, out_dir)
for dir_path in dirs_paths:
    parent_out_dir = out_dir
    out_dir = path.join(out_dir, dir_path)
    os.mkdir(out_dir)
    dir_path = path.join(directory, dir_path)
    save_directory_recursive(dir_path, out_dir)
    out_dir = parent_out_dir

def main():
    args = sys.argv

    if len(args) < 2:
        raise ArgumentError("No file or directory specified")
    if len(args) < 3:
        raise ArgumentError("No Output directory specified")

    args = args[1:]

    out_dir = args[1] # Get the output directory

    if not path.isdir(out_dir):
        os.mkdir(out_dir) # Create the output directory if it does not exist

    if len(args) > 2:
        if '-r' in args:
            args.remove('-r')

            parent_directory = args[0]
            save_directory_recursive(parent_directory, out_dir) # Run recursive if -r was in the command
        else:
            raise ArgumentError(f'{args[2]} is not an acceptable argument')

    elif '/' in args[0][-1] or '\\ in args[0][-1]:
        directory = args[0]

        save_directory(directory, out_dir) # Run directory conversion if the specified input is a directory

    else:
        file_path = args[0]

        save_file(file_path, out_dir) # Run file conversion if the specified input is a file

```

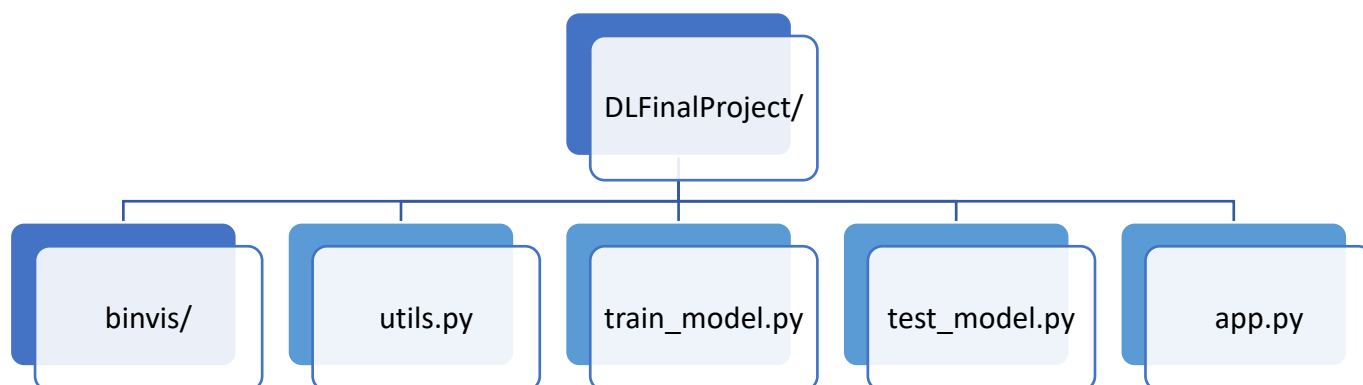



```
if __name__ == '__main__':  
    main()
```



הפרויקט הסופי

מבנה הפרויקט:



* הפרויקט מכיל גם את תיקיית `weights/` ותיקיית `data/` אך מכיוון שהן אינן כוללות קוד הן לא נכנסו לתרשים

- `binvis/` אחראי על המרת הקבצים לתמונות
- `utils.py` הוא בעל פעולות שימושיות במהלך השימוש במודל
- `train_model.py` הוא איפה שקוד אימון המודל נמצא
- `test_model.py` הוא איפה שבודקים אם המודל עובד טוב
- `app.py` הוא הקובץ עם הקוד של היישום, קובץ זה הוא הקובץ הראשי של הפרוייקט



:utils.py

```
import numpy as np
import matplotlib.pyplot as plt
import os
from os import path
from PIL import Image
import random

import tensorflow as tf
from tensorflow.keras import layers, models, losses

def get_data():
    """Get the train and test data for training

    Returns:
        Tuple: Tuple of the x_train, y_train and x_test, y_test
    """
    data_path = path.join(os.path.dirname(os.path.abspath(__file__)), 'data')
    CLASS_NAMES = os.listdir(data_path) # Get the class names from the directories names
    classes = {}

    for i in range(len(CLASS_NAMES)):
        classes[CLASS_NAMES[i]] = i

    X = []
    Y = []
    for class_name in CLASS_NAMES:
        for file in os.listdir(path.join(data_path, class_name)):
            if '.png' in file:
                image_path = path.join(data_path, class_name, file)
                X.append(np.array(Image.open(image_path).resize((220, 220)))) # Open the images and convert them
into 220x220
                Y.append(classes[class_name]) # Append the correspond class name

    X = np.array(X)
    Y = np.array(Y)

    n = round(X.shape[0] * 0.7)

    X_train, Y_train = X[:n], Y[:n]
    temp = list(zip(X_train, Y_train))
```



```

random.shuffle(temp)
X_train, Y_train = zip(*temp)
X_train = np.array(X_train)
Y_train = np.array(Y_train)

X_test, Y_test = X[n:], Y[n:]
temp = list(zip(X_test, Y_test))
random.shuffle(temp)
X_test, Y_test = zip(*temp)
X_test = np.array(X_test)
Y_test = np.array(Y_test)

return (X_train, Y_train), (X_test, Y_test)

def create_model(weights_path: str):
    """Create the AlexNet model architecture

    Args:
        weights_path (str): The saved weights path

    Returns:
        tensorflow.keras.models.Sequential: The model
    """
    model = models.Sequential()
    model.add(
        layers.experimental.preprocessing.Resizing(224, 224, interpolation="bilinear", input_shape=(224, 224, 3)))
    model.add(layers.Conv2D(96, 11, strides=4, padding='same'))
    model.add(layers.Lambda(tf.nn.local_response_normalization))
    model.add(layers.Activation('relu'))
    model.add(layers.MaxPooling2D(3, strides=2))
    model.add(layers.Conv2D(256, 5, strides=4, padding='same'))
    model.add(layers.Lambda(tf.nn.local_response_normalization))
    model.add(layers.Activation('relu'))
    model.add(layers.MaxPooling2D(3, strides=2))
    model.add(layers.Conv2D(384, 3, strides=4, padding='same'))
    model.add(layers.Activation('relu'))
    model.add(layers.Conv2D(384, 3, strides=4, padding='same'))
    model.add(layers.Activation('relu'))
    model.add(layers.Conv2D(256, 3, strides=4, padding='same'))
    model.add(layers.Activation('relu'))
    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))

```



```
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(2, activation='softmax'))

model.load_weights(weights_path)

return model

def normalize_image(X: np.ndarray):
    """Normalize image values and add padding

    Args:
        X (np.ndarray): The images array

    Returns:
        tensorflow.Tensor: normalized images
    """
    return tf.pad(X, [[0, 0], [2, 2], [2, 2], [0, 0]]) / 255
```



:train_model.py

```
import tensorflow as tf
from tensorflow.keras import layers, models, losses
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import utils

(x_train, y_train), (x_test, y_test) = utils.get_data() # Get the images
print(x_test.shape)

# Convert them into tensors
x_train = tf.convert_to_tensor(x_train)
x_test = tf.convert_to_tensor(x_test)

CLASS_NAMES = ['malware', 'normal']

x_train = tf.pad(x_train, [[0, 0], [2, 2], [2, 2], [0, 0]]) / 255
x_test = tf.pad(x_test, [[0, 0], [2, 2], [2, 2], [0, 0]]) / 255

x_val = x_train[-200:, :, :, :]
y_val = y_train[-200:]
x_train = x_train[:-200, :, :, :]
y_train = y_train[:-200]

# ----- model -----
model = models.Sequential()
model.add(layers.experimental.preprocessing.Resizing(224, 224,
                                                       interpolation="bilinear", input_shape=x_train.shape[1:]))
model.add(layers.Conv2D(96, 11, strides=4, padding='same'))
model.add(layers.Lambda(tf.nn.local_response_normalization))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(3, strides=2))
model.add(layers.Conv2D(256, 5, strides=4, padding='same'))
model.add(layers.Lambda(tf.nn.local_response_normalization))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(3, strides=2))
model.add(layers.Conv2D(384, 3, strides=4, padding='same'))
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(384, 3, strides=4, padding='same'))
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(256, 3, strides=4, padding='same'))
model.add(layers.Activation('relu'))
model.add(layers.Flatten())
```



```

model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(2, activation='softmax'))

print(model.summary())
model.compile(optimizer='adam', loss=losses.sparse_categorical_crossentropy,
              metrics=['accuracy']) # Compile model

history = model.fit(x_train, y_train, batch_size=64, epochs=50,
                   validation_data=(x_val, y_val)) # Start training

model.save_weights('./weights/weights.h5') # Save weights

# Plot the loss and accuracy
fig, axs = plt.subplots(2, 1, figsize=(15, 15))
axs[0].plot(history.history['loss'])
axs[0].plot(history.history['val_loss'])
axs[0].title.set_text("Training Loss vs Validation Loss")
axs[0].set_xlabel('Epochs')
axs[0].set_ylabel('Loss')
axs[0].legend(['Train', 'Val'])
axs[1].plot(history.history['accuracy'])
axs[1].plot(history.history['val_accuracy'])
axs[1].title.set_text("Training Accuracy vs Validation Accuracy")
axs[1].set_xlabel('Epochs')
axs[1].set_ylabel('Accuracy')
axs[1].legend(['Train', 'Val'])
plt.show()

print(model.evaluate(x_test, y_test)) # Evaluate the model test accuracy (second
                                     number in the list)

y_pred = model.predict(x_test) # Run a prediction again on the X_test

# ----- Plot the confusion matrix -----
T5_labels = ['Malware', 'Normal']

ax = plt.subplot()

cm = confusion_matrix(y_test, y_pred.argmax(axis=1))
print(cm)
sns.heatmap(cm, annot=True, fmt='g', ax=ax)

```



```
# labels, title and ticks  
ax.set_title('Confusion Matrix')  
ax.xaxis.set_ticklabels(T5_labels)  
ax.yaxis.set_ticklabels(T5_labels)  
  
plt.show()
```



:test_model.py

```
import numpy as np
import utils
from binvis import converter
import matplotlib.pyplot as plt

classes = ['malware', 'normal']

model = utils.create_model('weights/weights.h5')

file_path = '/Users/asafharel/Desktop/MalwareDatabase/Endermanch@000.exe'

# Convert random malware
image = converter.convert_to_image(256, file_path, save_file=False)

plt.imshow(image)
plt.show()

image = np.array(image.resize((220, 220)))

x_test = utils.normalize_image(np.array([image]))

y_pred = model.predict(x_test).argmax(axis=1)[0] # Get the prediction

print(classes[y_pred])
```



:app.py

```
import sys
import os
from os import path

from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox, QFileDialog,
                                QLabel, QPushButton
from PyQt5.QtGui import QFont, QPixmap, QColor

import numpy as np
from binvis.converter import convert_to_image
import utils

import string
import random

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.__model = utils.create_model('weights/weights.h5') # Load model
        self.setWindowTitle("Antivirus")
        self.setFixedSize(1200, 720)
        self.initUI()

        self.__classes = ['malware ✖', 'normal ✔']

        # Check which OS is running for later use in the file dialog
        if os.name == 'posix':
            self.__home_path = path.expanduser('~')
        else:
            self.__home_path = os.environ['USERPROFILE']

    def initUI(self):
        """Create all the necessary UI elements"""
        title_font = QFont("Arial", 48)
        title_font.setBold(True)

        self.title = QLabel("Antivirus", self)
        self.title.setFont(title_font)
        self.title.setGeometry(495, 40, 210, 55)
```



```

self.image_label = QLabel(self)

self.image_label.setGeometry(525, 165, 400, 400)

self.file_button = QPushButton("Select File", self)
self.file_button.setGeometry(500, 330, 200, 30)
self.file_button.clicked.connect(self.launchFileDialog)

self.confirmit_button = QPushButton("Analyze", self)
self.confirmit_button.setGeometry(450, 630, 300, 40)
self.confirmit_button.clicked.connect(self.confirm_clicked)

self.confirmit_button.setStyleSheet(
    """
    QPushButton {
        border: none;
        background-color: #000000;
        color: white;
        font-size: 20pt;
        border-radius: 10px;
    }

    QPushButton:disabled {
        background-color: #444444;
        color: #eeeeee;
    }
    """
)

def launchFileDialog(self):
    """Open File Dialog"""
    files_filter = "Executables (*.exe)" # Show only exe files
    file_path, file_type = QFileDialog.getOpenFileName(
        self,
        caption="Select a .exe file",
        directory=os.path.join(
            os.path.join(self.__home_path,
                "Desktop")), filter=files_filter,
        initialFilter='Executables (*.exe)'
    )

    malware_file_path = ".join(random.choices(string.ascii_letters, k=16)) + ".png" # Create random name for the
    image file

    self.malware_image = convert_to_image(256, file_path, malware_file_path) # Convert file to image

    self.file_button.setGeometry(500, 120, 200, 30)
    
```



```

image_pixmap = QPixmap(malware_file_path)
self.image_label.setPixmap(image_pixmap)
    # Show the converted file image
self.image_label.resize(image_pixmap.width(), image_pixmap.height())
os.remove(malware_file_path) # Remove the image file

def confirm_clicked(self):
    # Modify image to fit into the model
    image_array = np.array(self.malware_image.resize((220, 220)))
    modified_image = utils.normalize_image(np.array([image_array]))

    # Predict image
    result = self.__model.predict(modified_image).argmax(axis=1)[0]
    # Convert result to user friendly string
    result_str = self.__classes[result]

    self.confirm_button.setEnabled(False)

    # Open new window with result
    msg = QMessageBox()
    msg.setWindowTitle("Analyzer")
    msg.setText(result_str)
    msg.exec_()
    self.confirm_button.setEnabled(True)

app = QApplication(sys.argv)
win = MainWindow()
win.show()
sys.exit(app.exec_()) # Run Application

```



מדריך למשתמש

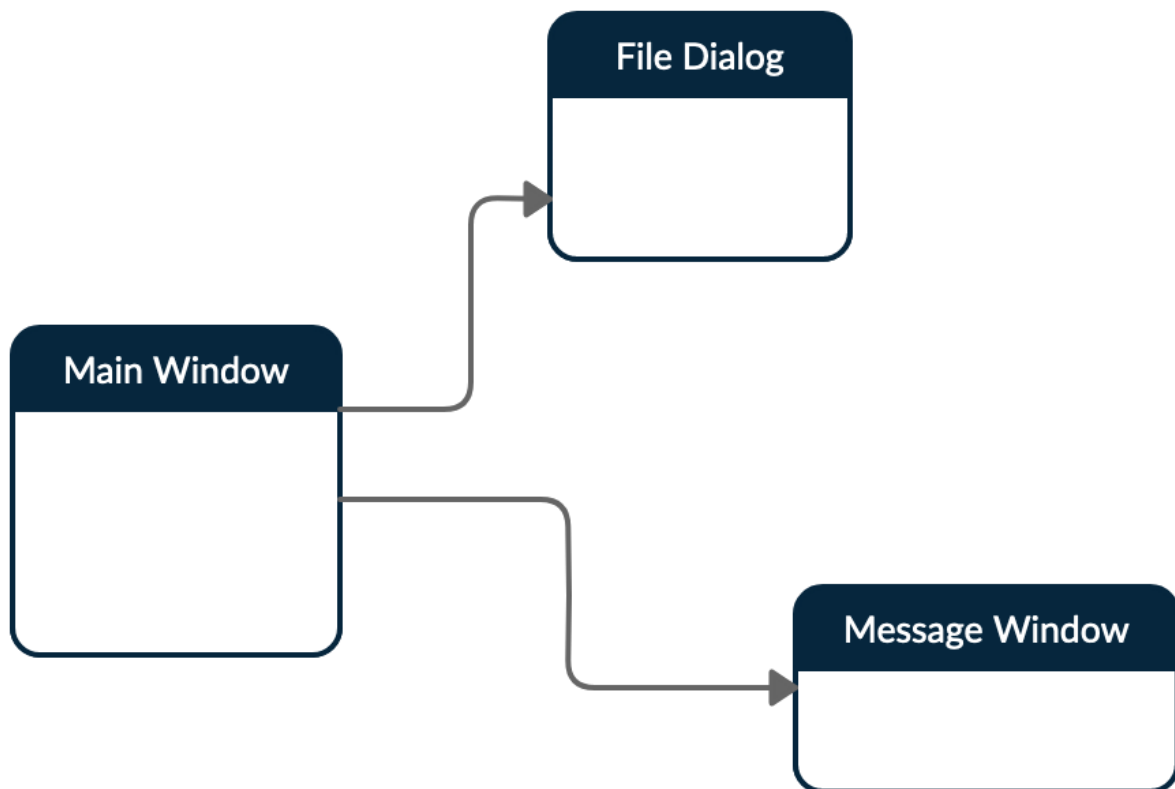
התקנה

על מנת לנסות ולעשות את חוויית המשתמש כמה שיותר פשוטה, הפכתי את קוד הפייתון של היישום לתוכנה בעזרת ספריית Pyinstaller המאפשרת להפוך קבצי פייתון לקבצי exe והמשתמש רק צריך להוריד את התיקיה מהקישור שנמצא ב-[repository](#) ולפתוח את הקובץ Antivirus.exe

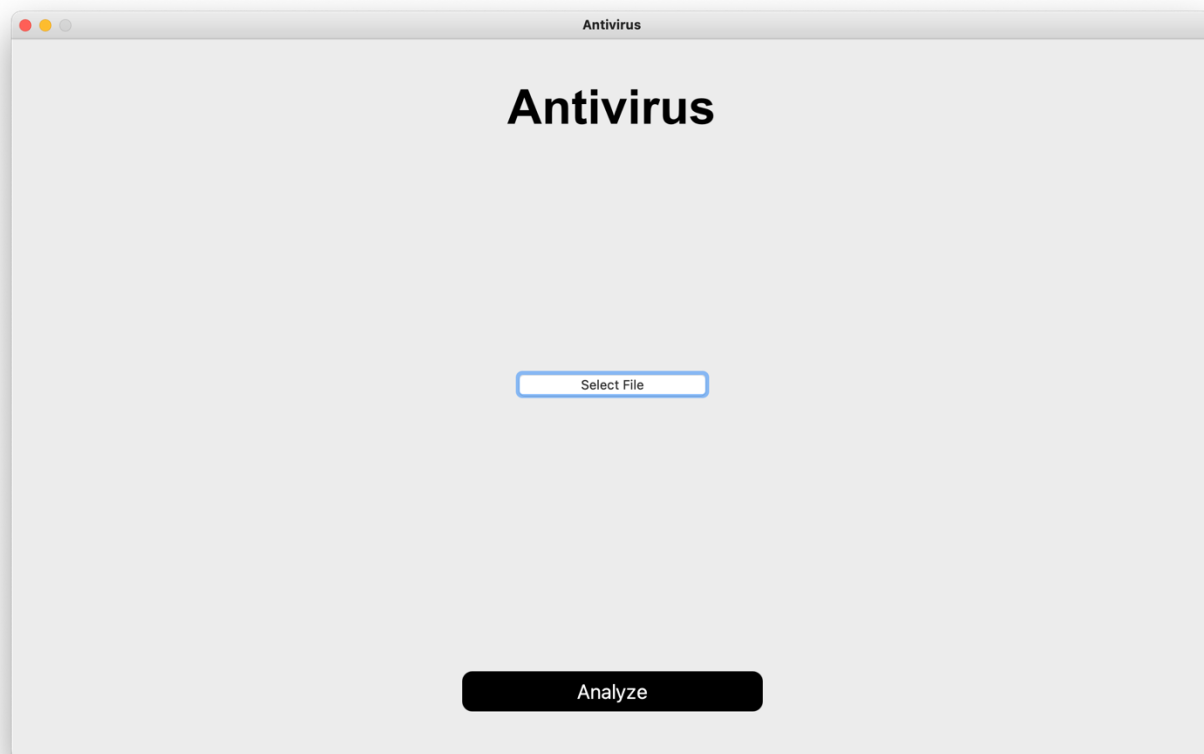
הדרישה היחידה על מנת להבטיח שימוש נטול באגים היא שעל המחשב של המשתמש יהיה פייתון.

- בנוסף, אם משתמש ירצה לשים את התוכנה במקום נוח יותר הוא יכול ליצור קיצור דרך לקובץ Antivirus.exe ולשים את הקיצור היכן שירצה.

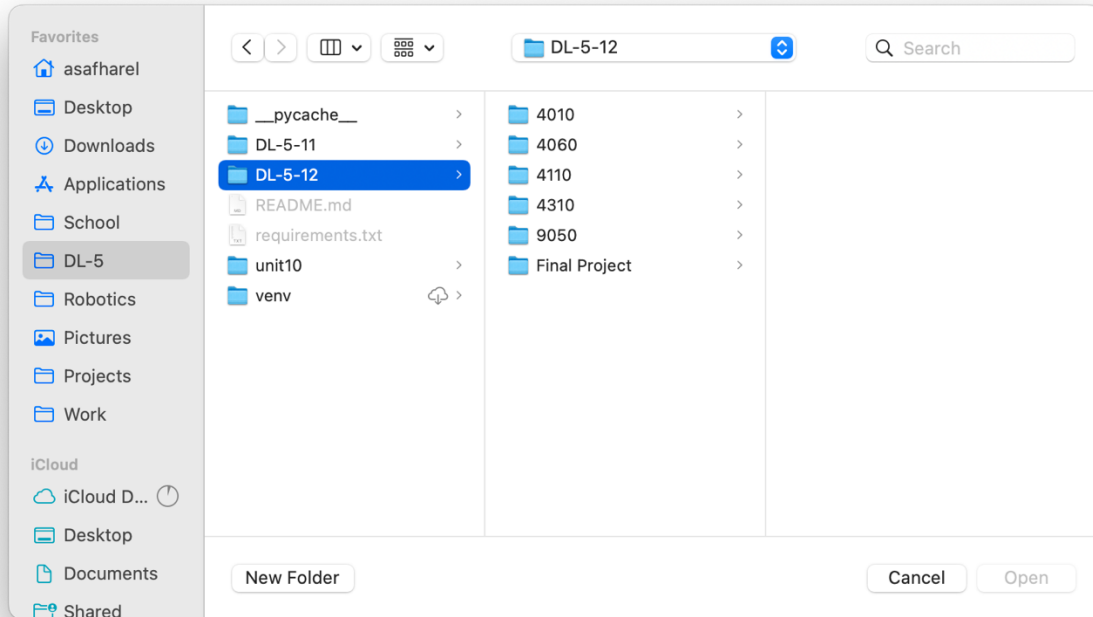
תרשים מסכים



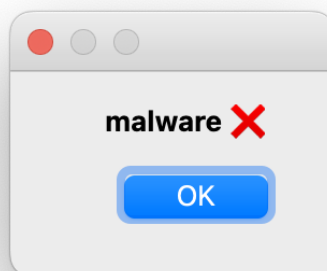
Main Window – המסך בו רוב הפעולות קורות, משם פותחים את חלון בחירת הקבצים ומשם פותחים את חלון ההודעה.



File Dialog – המסך בו נבחר הקובץ הרצוי, מסך זה הינו מסך גלישה בתכולת המחשב



Message Window – המסך בו תוצג התשובה של המודל, וירוס או לא וירוס



רפלקציה

אני נהנתי מאוד במהלך העבודה על הפרויקט, אני חושב שרוב ההנאה הגיעה מכך שניתנה לי האפשרות לעשות את הפרויקט על נושא לבחירתי, מה שנתן לי את האפשרות לבחור נושא שמעניין אותי ולכן יכולתי להיסחף בקלות לתוכו, לעבוד עליו קשה ולחקור אותו לעומק.

בשל הנושא שבחרתי שלב המחקר היה המאתגר ביותר, חוסר הנתונים בכמויות גדולות באינטרנט הקשו על תהליך מציאת הנתונים והתכנון של הפרויקט לאחר מכן. חשש גדול היה שלא יהיו לי מספיק נתונים כדי לאמן את הרשת ואצטרך להחליף נושא.

בנוסף לבעיית מציאת הנתונים הייתה את בעיית התייעוד של הרשת בה רציתי להשתמש, [רשת SOINN](#) ולכן לאחר כשבועיים של ניסיון מחקר על הרשת החלטתי להחליף [לרשת AlexNet](#) בשל אמינותה והתייעוד הרחב שלה ברחבי האינטרנט. במפתיע, הקושי הגדול ביותר היה גם השלב האהוב עליי ביותר בעבודה וזה הוא הניסיון להפוך את תהליך הפיכת הקבצים לתמונות לאוטומטי, השלב הזה [כמו שהרחבתי](#) לפניכן אילץ אותי לנסות להבין איזה קטעי קוד חסרים על מנת להשלים את הקוד המקולקל ולשכתב כמעט את כל הקוד על מנת שיתאים לגרסאות החדשות של פייתון שכן הקוד המקורי נכתב לפני כ-7 שנים.

בסך הכל אני חושב שלמדתי מהפרויקט הזה הרבה אך בעיקר למדתי עוד על תחום למידת המכונה מכיוון שחלק מהדברים שהשתמשתי בהם במודל אינם חלק מהחומר הנלמד והייתי צריך לחקור לבד מה הם הדברים האלו. בנוסף למדתי עוד על השימוש ב-Google Scholar, מנוע החיפוש של גוגל למאמרים אקדמיים, מכיוון שבמהלך מחקר על הנושא הכתבות הרגילות באינטרנט לא הספיקו והייתי צריך לחפש מאמרים יותר מעמיקים.

ככל הנראה, אם הייתי מתחיל את העבודה היום הייתי נמנע מלהיתקע על חקר מודל ה-SOINN מכיוון שחקר זה גזל לי הרבה זמן מהעבודה ולא הוביל לשום תוצאות ולכן, אם הייתי נמנע ממנו ומחליט לוותר עליו יותר מהר ולעבור ל-AlexNet אז היה לי יותר זמן להשקיע במציאת עוד נתונים. בהמשך לכך, ויתור מהיר יותר על הרשת המקורית היה מיעל לי את הזמן הרבה יותר שכן במצב הנוכחי נאלצתי לעבוד על כמה חלקים של העבודה במקביל מכיוון שלא היה לי הרבה זמן.

לסיום, הייתי מסכם את העבודה על הפרויקט מהנה ומלמדת ואין ספק שהייתי מדרג את הנושא הזה במקום הראשון במקצועות בבית הספר.



ביבליוגרפיה

המחקר עליו התבססתי:

Baptista, I., Shiaeles, S., & Kolokotronis, N. (2019, May 24). *A Novel Malware Detection System Based On Machine Learning and Binary Visualization*. Arxiv.

Retrieved January 3, 2022, from <https://arxiv.org/pdf/1904.00859.pdf>

האתר להמרת הקבצים לתמונות:

[https://binvis.io/](https://binvis.io/#/)

הקוד המקורי של המרת הקבצים לתמונות:

Cortesi. (2010, January 1). *GitHub - cortesi/scurve: A library for drawing space-filling curves like the Hilbert Curve*. GitHub. Retrieved October 23, 2021, from <https://github.com/cortesi/scurve>

3Blue1Brown. (2017, July 21). *Hilbert's Curve: Is infinite math useful?* [Video].

YouTube. <https://www.youtube.com/watch?v=3s7h2MHQtxc>



Wikipedia contributors. (2022, January 28). *AlexNet*. Wikipedia.

<https://en.wikipedia.org/wiki/AlexNet>

המאמר ממנו לקחתי את הקוד הראשוני למודל:

mrgrhn (2021, December 10). *Difference between Local Response Normalization and Batch Normalization*. Medium. <https://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization-272308c034ac>

What is the difference between sparse_categorical_crossentropy and categorical_crossentropy? (2019, October 25).

Stack Overflow.

https://stackoverflow.com/questions/58565394/what-is-the-difference-between-sparse_categorical_crossentropy-and-categorical_crossentropy

Supervised vs. Unsupervised Learning: What's the Difference? (2021, March 12). IBM.

<https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>



Anwar, A. (2021a, December 10). *Difference between Local Response Normalization and Batch Normalization*. Medium. <https://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization-272308c034ac>

