

שיטות למידת מכונה לזיהוי וירוסים

סמינר באבטחת המרחב המקוון, 20927
הדר תפארת, 205492507

תוכן עניינים

1	מבוא
2	חלק 1: למידת מכונה
2	פרק 1: מבוא ללמידת מכונה
2	1.1 – מהי למידת מכונה
2	1.2 – חשיבות טיב הנתונים
2	1.3 – מוטיבציה לגישה
3	1.4 – סוגי למידת מכונה
4	פרק 2: למידה מפוקחת
4	2.1 – הצגת התהליך
4	2.2 – דוגמאות מודלים ללמידה מפוקחת
4	2.2.1 – עצי החלטה, Decision trees
6	2.2.2 – יער אקראי, Random forest
7	2.2.3 – למידה בייסיאנית פשוטה, naïve bayes
7	2.2.4 – K Nearest Neighbor
8	2.2.5 – רגרסיה לינארית, Linear regression
8	2.2.6 – מכונת וקטורים תומכים, Support vector machines
10	פרק 3: למידה לא מפוקחת
10	3.1 – הצגת התהליך
10	3.2 – דוגמאות מודלים ללמידה לא מפוקחת
10	3.2.1 – ניתוח אשכולות, Clustering
12	3.2.2 – חוקי הקשר, Association rules
13	פרק 4: רשתות נוירונים ולמידה עמוקה
13	4.1 – מבוא לרשתות נוירונים
14	4.2 – הצגת תהליך הלמידה
14	4.2.1 – התפשטות לפני, forward propagation
14	4.2.2 – פונקציות מחיר, loss functions
15	4.2.3 – אלגוריתם מורד הגרדיאנט, gradient descent
16	4.2.4 – התפשטות לאחור, back propagation
16	4.2.5 – פונקציות אקטיבציה, activation functions
16	4.2.6 – שכבות נרמול, normalization layers
17	4.2.7 – שכבת dropout
17	4.3 – יחידת הפרספטרון
18	4.4 – רשתות נוירונים בעלות קישוריות מלאה, feed-forward neural networks
18	4.4.1 – שכבה לינארית
18	4.4.2 – מבנה הרשת
18	4.5 – רשתות קונבולוציה, Convolutional Neural Networks
18	4.5.1 – מוטיבציה
19	4.5.2 – שכבת הקונבולוציה
20	4.5.3 – מבנה רשת קונבולוציה
20	4.6 – רשתות נשנות, Recurrent Neural Networks
20	4.6.1 – מוטיבציה
21	4.6.2 – תא נשנה, RNN Cell
21	4.6.3 – מבנה רשת נשנית
21	4.7 – ארכיטקטורת מקודד-מפרש, Encoder-Decoder
21	4.7.1 – מוטיבציה
22	4.7.2 – מבנה הרשת
22	4.7.3 – שימוש נוסף: העברת לימוד
23	חלק 2: זיהוי וירוסים
23	פרק 5: הכרת האיום
23	5.1 – וירוס: הגדרה
23	5.2 – סיווג וירוסים
24	5.3 – ניתוח נזקקות, malware analysis

24	5.3.1 – ניתוח סטטי, static analysis
25	5.3.2 – ניתוח דינאמי, dynamic analysis
26	פרק 6: שיטות אמפיריות לזיהוי נזקות
26	6.1 – זיהוי מבוסס חתימה
26	6.2 – זיהוי מבוסס היוריסטיקה
27	פרק 7: שיטות למידת מכונה קלאסית לזיהוי נזקות
27	7.1 – מאפיינים סטטיים
27	7.1.1 – ניתוח רצפים, string analysis
27	7.1.2 – n-gram מבוסס בתים ופקודות בקוד אסמבלי, byte/opcode n-grams
28	7.1.3 – קריאות לפונקציות ממשק, API function calls
28	7.1.4 – ניתוח אנטרופיה
28	7.1.5 – ייצוג קוד כתמונה
29	7.1.6 – גרף קריאות פונקציה, function call graph
29	7.1.7 – גרף בקרת זרימה
29	7.2 – מאפיינים דינאמיים
29	7.2.1 – זיכרון ושימוש באוגרים
29	7.2.2 – מעקב אחרי פקודות, instruction trace
30	7.2.3 – ניטור תעבורת תקשורת
30	7.2.4 – מעקב אחרי פונקציות ממשק, API call trace
30	7.3 – שימוש במודלים של למידת מכונה קלאסית (ממאפיינים למודל)
31	פרק 8: שיטות למידה עמוקה לזיהוי נזקות
31	8.1 – למידה עמוקה לעומת למידת מכונה קלאסית
31	8.2 – ייצוג כווקטור מאפיינים, Feature vector representation
31	8.3 – ייצוג כתמונה, Image based representation
31	8.4 – מעקב אחרי פונקציות ממשק \ קריאות לפונקציות, API call trace / Instruction trace
32	8.5 – ייצוג מבוסס בתים
33	פרק 9: מבט לעומק – שיטת למידת מכונה מבוססת תמונת זיכרון לסיווג נזקות
33	9.1 – ניתוח זיכרון נדיף, Volatile memory forensics
33	9.2 – הצגת השיטה
34	9.2.1 – איסוף מידע אודות זיכרון
34	9.2.2 – ייצוג המידע כתמונה
35	9.2.3 – חילוץ מאפיינים ויזואליים
36	9.2.4 – סיווג על ידי טכניקות למידת מכונה
36	9.2.5 – מדדי הצלחה
37	9.3 – תוצאות הניסוי
39	פרק 10: מבט לעומק – שיטת למידה עמוקה מבוססת תמונת קובץ הרצה לזיהוי נזקות
39	10.1 – בניית בסיס נתונים מתאים ללמידה
39	10.2 – הצגת השיטה
39	10.2.1 – ייצוג קובץ הרצה כתמונה
40	10.2.2 – שימוש ברשת עמוקה לניתוח מאפיינים
40	10.2.3 – שימוש בלמידת מכונה קלאסית למשימת זיהוי הנוזקות
40	10.2.4 – הרציונל מאחורי השיטה
41	10.3 – מהלך הניסוי ותוצאותיו
41	10.3.1 – השוואה בין ביצועי מודלים ללמידה עמוקה למשימת זיהוי נזקות
42	10.3.2 – בחינת ביצועי המודל המוצע לזיהוי נזקות
44	פרק 11: הצגת שיטת למידה עמוקה מבוססת מקודד עצמי
44	11.1 – מוטיבציה לשיטה
44	11.2 – הצגת השיטה
44	11.2.1 – בנייה ואימון של מקודד עצמי
44	11.2.2 – איסוף מידע אודות זיכרון וייצוג כתמונה
44	11.2.3 – סיווג באמצעות מודל מבוסס למידה עמוקה
45	11.3 – מהלך הניסוי ותוצאותיו
47	סיכום:
48	ביבליוגרפיה

מבוא

מערכות תוכנה מהוות חלק בלתי נפרד מחיינו, החל מהפלאפונים החכמים, דרך שרתי ענק המספקים תוכן למיליוני אנשים מסביב לעולם, מערכת הבנקאות העולמית וכלה במערכות ביטחוניות הגנתיות והתקפיות כאחד.

מערכות אלו חשופות לתוכנות מזיקות, אשר מטרתן לנצל, לפגוע או לגרום נזק למערכת ולמשתמשים בה. קיים מרוץ תמידי בין יצרני מערכות הגנה, אשר מנסים לגלות ולהתמודד עם האיומים השונים לבין כותבי הנוזקות אשר מייצרים נוזקות חדשות ומנסים לערום על אמצעי ההגנה הקיימים לשם השגת מטרתם.

בשנים האחרונות, עם התפתחות ענף למידת המכונה, נעשה שימוש נרחב בשיטות למידת מכונה לזיהוי וסיווג נוזקות בשל מורכבות המשימה ויכולתן של שיטות אלו ללמוד דפוסים שונים.

עבודה זו מחולקת לשני חלקים.

בחלק הראשון נסביר מהי למידת מכונה באופן כללי ונציג מספר שיטות ללמידת מכונה המשויכות ללמידה מפוקחת, למידה לא מפוקחת ולמידה עמוקה.

בחלק השני של עבודה זו נגדיר נוזקות באופן כללי ווירוסים בפרט, נציג מספר שיטות לזיהוי נוזקות, כאשר רובן עושה שימוש בטכניקות למידת מכונה, נבחן לעומק שתי שיטות ספציפיות, אשר מתבססות על מאמרים קיימים ולבסוף נבחן בעצמנו את הפוטנציאל של שיטה מבוססת למידה עמוקה.

העבודה נכתבה מתוך כוונה להעביר את הנושאים המוצגים בצורה שתהיה מובנת לפחות רעיונית גם לקורא שאינו בהכרח מכיר את התחום ולכן מוצגים בנוסף לשיטות המסוקרות גם הסברים בדבר ההיגיון בשימוש בהן וגם מושגים אשר חיוניים להגדרתן.

חלק 1: למידת מכונה

פרק 1: מבוא ללמידת מכונה

1.1 – מהי למידת מכונה

למידת מכונה היא תת תחום של בינה מלאכותית, אשר מתמקד בשימוש בנתונים ואלגוריתמים בכדי להתחקות אחרי תהליך הלמידה האנושי.

המושג למידת מכונה מיוחס לחוקר IBM – Arthur Samuel, אשר בשנת 1962 פרסם מאמר ובו הציג מכונה אשר למדה לשחק את המשחק דמקה.

מאז התחום התפתח רבות וכעת ניתן לראות שימוש נרחב בשיטות של למידת מכונה לפתרון מגוון רחב של בעיות בתחומים שונים.

[What is machine learning?, n.d.]

מכונה נחשבת כלומדת כאשר היא קולטת נתונים, בונה מודל בהתאם לנתונים ומשתמשת במודל זה בכדי להסיק מסקנות.

1.2 – חשיבות טיב הנתונים

כאמור, למידת מכונה מתבססת על נתונים בכדי להסיק מסקנות.

לכן, ישנה חשיבות עליונה לאיכות הנתונים אליהם המודל נחשף בעת שלב האימון. על הנתונים להיות אמינים ולייצג בצורה מדויקת ככל הניתן את המציאות אליה המודל ייחשף.

נציג מספר דוגמאות לבעיות שעלולות לצוץ בעקבות החלטה מסוימת לבחירת נתוני האימון:

- מודל אשר אומן לסווג בין תמונות של כלבים וחתולים. במידה והמודל ייתקל לאחר האימון בתמונה שאינה מכילה כלב או חתול – הוא יסווג בשוגג את התמונה לאחת משתי הקטגוריות הללו, שכן זהו המרחב שהמודל מכיר.
- מודל אשר אומן לזהות חשד למחלת הסרטן על סמך נתונים רפואיים של בן אדם, במידה ואימון המודל יתבצע רק על סמך נתוני החולים של מחלקה אונקולוגית בבית חולים – ייתכן והמודל יהיה מוטה לטובת סיווג אנשים כחולים במחלה מכיוון ורוב הדוגמאות אליהן נחשף המודל בעת אימונו היו של חולים והמודל אינו נחשף בצורה מספקת לדוגמאות של אנשים שאינם חולים.

1.3 – מוטיבציה לגישה

למידת מכונה מצריכה קבלת נתונים אמינים ומתאימים למשימה והרבה מהם, בכדי להניב תוצאות טובות. בנוסף, תהליך האימון לוקח לרוב זמן רב וכתלות במודל יכול לדרוש משאבים רבים.

אם כן, מדוע נבחר בלמידת מכונה על פני תכנות "מסורתי" - כתיבת סדר פעולות שעל המכונה לבצע בזו אחר זו?

נציג שני סוגי בעיות בהן יכולת הלמידה של מודל תסייע במציאת פתרון:

• משימות קשות במיוחד לתכנות

- ישנן משימות אשר לעיתים ניתן לחשוב על קונספט לפתרון, אך קשה ליישם פירוט של פקודות אותן מכונה צריכה לבצע בכדי להשיג פתרון זה.
- משימות אלו כוללות משימות אשר אנו כבני אדם מבצעים בקלות יחסית, כגון: זיהוי אובייקטים בתמונה, תרגום שפה טבעית וזיהוי קולי, ומנגד גם משימות אשר מהוות קושי רב למוח האנושי, בשל צורך במספר רב מאוד של חישובים או צורך בניתוח נתונים בסדרי גודל אסטרונומיים, כדוגמת: מיטוב מנועי חיפוש והסקת מסקנות מתוך מאגרי נתונים גדולים.
- בעיות כאלו הינן קשות להגדרה כפקודות מכונה שלב אחר שלב ודורשות יכולת הסקת מסקנות והבחנה בין עיקר לתפל. תכונות אשר קשה עד בלתי אפשרי להשיג באמצעות תכנות מסורתי.

• משימות אשר דורשות גמישות

תכנות מסורתי דורש הגדרה מלאה ומדויקת של המשימה לביצוע. עם זאת, ישנן משימות בהן אנו מעוניינים בגמישות באופן הפתרון, הן בשל רצון להשתמש בפתרון ידוע וטוב בעבור מספר של משימות בעלות אופי דומה והן בשל מצבים בהם מאגר הידע הקיים משתנה (למשל גדל או מתעדכן). בתכנות מסורתי נצטרך לשנות את הפתרון הקיים בכדי שיתאים למצב החדש, אך מודל של מכונה לומדת יוכל ללמוד מהשינויים בנתונים אליהם הוא נחשף ולהתאקלם למצב החדש.

על כן, למרות הקשיים הנובעים משימוש בלמידת מכונה, ישנן משימות בהן השיטה מהווה פתרון אטרקטיבי מאוד ואף מניבה תוצאות טובות יותר משיטות אחרות.

1.4 – סוגי למידת מכונה

למידת מכונה היא תחום רחב אשר כולל בתוכו טכניקות שונות שמתאימות לצרכים שונים.

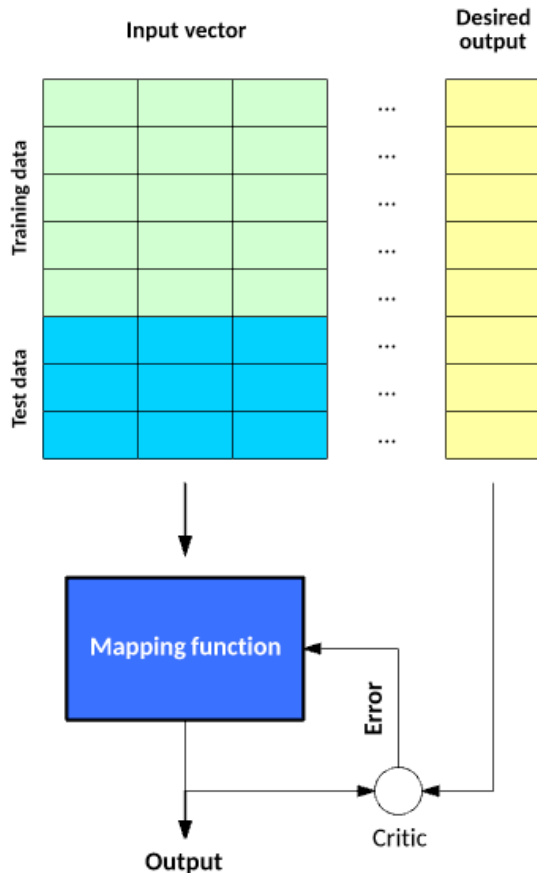
מודל יכול לכלול את התכונות הבאות:

1. למידה מפוקחת לעומת לא מפוקחת.
בלמידה מפוקחת, בעת אימון, המודל מקבל כקלט רשומות של נתונים (משתנים) ובעבור כל רשומה – משתנה מטרה תואם. משתנה המטרה הוא הסיווג (או התוצאה) שמצופה מהמודל ללמוד בעבור סט הנתונים. בעת מבחן (חיזוי) – המודל מסיק את משתנה המטרה המתאים לנתונים המוזנים לו, בהתאם למה שלמד.
לעומת זאת, בלמידה לא מפוקחת – המודל אינו מקבל בנוסף לנתונים משתנה מטרה כלשהו, ועליו להסיק על סמך הנתונים בלבד כיצד לשייך או להבדיל בין סט נתונים אחד למשנהו.
קיימת גם גישה שלישית, הנקראת למידת חיזוק, בגישה זו – המודל אינו מקבל משתנה מטרה מוזן ידנית בנוסף לסט הנתונים, אלא ערך כלשהו המבוסס על הנתונים עצמם ומשמש למטריקה בעבור שיפור דיוק הליך הלמידה.
2. למידה אקטיבית לעומת למידה פסיבית.
מודל אקטיבי מסוגל להשפיע על הסביבה ממנה הוא לומד, בעוד שמודל פאסיבי לומד מן הסביבה הנתונה לו מבלי לשנות אותה כלל.
3. מידת התמיכה של המורה.
נרצה להתאים את מאגר הנתונים אליו נחשף המודל לסוג המשימה.
למשל, מודל אשר תפקידו לחזות מאורע אשר מתרחש בטבעיות, ללא קשר להימצאו – יכול ללמוד בצורה טובה מנתונים המייצגים באופן סטטיסטי את הסביבה.
לעומת זאת, מודל אשר תפקידו לזהות פעילות זדונית יהיה מוצלח יותר אם במהלך אימונו ייחשף לנתונים אשר הוזנו במתכוון במטרה לערום עליו.

פרק 2: למידה מפוקחת

2.1 – הצגת התהליך

למידה מפוקחת מאופיינת בכך שבתהליך האימון, המודל מקבל בנוסף לרשומות הנתונים גם ערך מטרה בעבור כל רשומה.



תרשים 1: תרשים תהליך למידה מפוקחת, [Jones, 2017]

בתהליך הלמידה המודל קורא רשומות של נתונים, מייצר ערך מטרה משוער בעבור נתונים אלו, בוחן את ערך המטרה המשוער כנגד ערך המטרה הידוע בעבור רשומת הנתונים באמצעות פונקציית מחיר ולבסוף משנה את ערכי המודל בכדי למזער את השגיאה שזוהתה. התהליך חוזר על עצמו במטרה למזער את השגיאה ככל הניתן.

חלקי המודל:

- מאגר הנתונים: רשומות של נתונים אותם לומד המודל, בצירוף ערכי מטרה אשר הוזנו בעבור כל רשומת נתונים במאגר.
- מודל להסקת ערך מטרה מתוך רשימת נתונים
- פונקציית מחיר להסקת שגיאה בין ערך המטרה המוסק מהמודל לבין ערך המטרה האמיתי

מודלים של למידה מפוקחת יכולים לשמש לביצוע אחת משתי משימות:

- סיווג – המודל מתאים לכל רשומת נתונים חדשה אליה הוא נחשף את ערך המטרה המתאים לה ביותר מבין ערכי המטרה להם נחשף במהלך האימון.
- רגרסיה – המודל מחזיר בעבור כל רשומת נתונים חדשה אליה הוא נחשף ערך רציף המחושב על ידי מציאת המקום של רשומת הנתונים ביחס לנתונים אליהם נחשף המודל בתהליך האימון.

2.2 – דוגמאות מודלים ללמידה מפוקחת

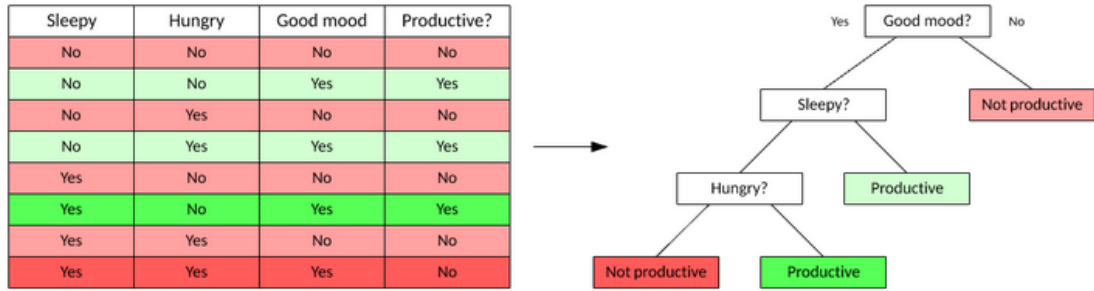
נציג מספר מודלים של למידה מפוקחת:

2.2.1 – עצי החלטה, Decision trees

עץ החלטה הוא מבנה עץ, בו כל צומת פנימית מייצגת מבחן על תכונה מבין תכונות הנתונים, כל ענף מייצג את תוצאת המבחן על הצומת ממנה הוא יצא וכל עלה (צומת קצה) מבטא החלטה על ערך מטרה.

[Han, Kamber, & Pei, 2012, p. 330]

עץ החלטה יכול להיות בינארי – במידה וכל מבחן מניב עד שתי תוצאות או לא בינארי במידה ויתכנו יותר משתי תוצאות למבחן מסוים.



תרשים 2: דוגמה למימוש עץ החלטה, [Jones, 2017]

לאחר בניית עץ ההחלטה, נוכל לסווג סט נתונים חדש אשר משתנה המטרה עבורו אינו ידוע. נבצע זאת על ידי מעבר משורש העץ דרך תוצאות כל המבחנים אשר תואמות לסט הנתונים שברשותנו, עד להגעה לעלה, כאשר משתנה המטרה התואם לעלה אליו הגענו קובע את הסיווג לסט הנתונים.

עצי החלטה קלים להבנה וייצוגם אינטואיטיבי לעין האנושית, בנוסף תהליך הלמידה והסיווג שלהם מהיר יחסית.

כיצד נבנה עץ החלטה?

בגרסה הבסיסית, העץ נבנה מלמעלה למטה (באופן חמדני) לפי גישה של הפרד ומשול. בהגעה לצומת חדש, נרצה לבחון את התכונה שתורמת הכי הרבה לסיווג משתנה התוצאה מבין התכונות אשר לא נבחנו בצמתים קודמים במעלה העץ. נבחר תכונה זו, הנקראת תכונת הפיצול, לפי מדד פיצול כלשהו ונחזור על התהליך עד שניתן יהיה להסיק את משתנה התוצאה מתוך כל אחד מעלי העץ או שלא יישארו עוד תכונות לפיצול הענף.

מדד הפיצול מהווה יוריסטיקה לקביעת התכונה הבאה לפיצול, עבורה סיווג נתוני האימון מתחלק בצורה הטובה ביותר.

ישנם מדדי פיצול אפשריים רבים, ביניהם:

1. אנטרופיה - רווח אינפורמטיבי (information gain) ויחס רווח (gain ratio):

כאשר D - מאגר נתוני האימון, C_i - משתנה התוצאה i במספר i - $p_i = \frac{|C_i D|}{|D|}$ - ההסתברות לכך

שסט נתונים כלשהו מנתוני האימון שייך למשתנה התוצאה C_i :

נחשב את האנטרופיה הדרושה בכדי לסווג סט נתונים D כ: $Info(D) = -\sum_{i=1}^m p_i \log_2(p_i)$ כעת, לאחר פיצול מאגר הנתונים לפי v הערכים האפשריים של תכונה A , נקבל כי המידע הדרוש

לסיווג המאגר הוא: $Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} * Info(D_j)$

נשתמש בנתון זה לחישוב רווח האינפורמציה בעבור התכונה A :

$$Gain(A) = Info(D) - Info_A(D)$$

לבסוף, לפי האלגוריתם - בבואנו לבחור את התכונה הבאה לפיצול, נבחר בתכונה עבורה רווח האינפורמציה הוא הגבוה ביותר.

נשים לב כי למדד הרווח האינפורמטיבי יש העדפה טבעית לתכונות בעלות מספר רב של ערכים אפשריים ולכן ניתן להשתמש במדד אחר, יחס רווח, אשר מבצע סוג של נרמול על ערכי הרווח האינפורמטיבי ובכך מנסה להתמודד עם ההטיה של מדד זה.

ראשית נגדיר את פיצול המידע כ: $Split Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} * \log_2 \left(\frac{|D_j|}{|D|} \right)$

וכך נוכל להסיק את ערך יחס הרווח: $Gain Ratio(A) = \frac{Gain(A)}{Split Info_A(D)}$

לבסוף, נבחר בתכונה בעלת יחס הרווח הגבוה ביותר כתכונת הפיצול.

2. מדד פיצול *Twoing*:

מדד המפצל כל קודקוד באופן בינארי על ידי חיפוש אחר קבוצות שמחלקות את התצפיות באופן שווה, ללא תלות באופי המשתנה.

בעבור p_L ו- p_R פרופורציות הנתונים המתחלקים לענף השמאלי והימני בהתאמה, j – אינדקס משתנה המטרה ו- $p(\frac{j}{t_L}), p(\frac{j}{t_R})$ – הסתברות השייכות של נתון למשתנה המטרה j מתוך כלל הנתונים בענף השמאלי והימני בהתאמה, נבחר בפיצול אשר גורר את ערך המדד המקסימלי עבור:

$$\frac{p_L * p_R}{4} * \left[\sum_j \left| p\left(\frac{j}{t_L}\right) - p\left(\frac{j}{t_R}\right) \right| \right]^2$$

3. מדד ג'יני:

מדד המניח כי התכונות רציפות ומבצע פיצול בינארי, בעדיפות למצב בו שתי הקבוצות יהיו יחסית שוות בגודלן. המדד בוחן את מידת ה'טוהר' של מאגר הנתונים D בתור: $Gini(D) = 1 - \sum_{i=1}^m p_i^2$ כאשר p_i היא ההסתברות לכך שסט נתונים במאגר D יהיה שייך למשתנה המטרה C_i . כאשר הסכימה מבוצעת מעל כל m משתני המטרה.

כאשר אנו בוחנים פיצול בינארי סביב התכונה A , אנו מחשבים את מדד הפיצול של D ל- D_1, D_2 לפי $Gini_A(D) = \frac{|D_1|}{|D|} * Gini(D_1) + \frac{|D_2|}{|D|} * Gini(D_2)$ בעבור כל תכונה A . התכונה עבורה $Gini_A(D)$ מקבל את הערך הנמוך ביותר נבחרת כתכונה הבאה לפיצול.

ניתן לבחון את תוצאות המודל בעבור בחירת מדדי פיצול שונים ולבחור בפיצול המניב את התוצאה הטובה ביותר בצורה אמפירית, אך בדרך כלל אנטרופיה פופולרית בספרות על למידת מכונה, מדד Twoing מניב תוצאות טובות בעבור בעיות המכילות מספר רב של משתני מטרה אפשריים ומדד ג'יני אופטימלי בעבור בעיות בעלות משתנה מטרה בינארי.

טיפול בהתאמת יתר:

בעת בניית עץ החלטה, ענפים מסוימים עלולים להיווצר בעקבות טיפול באנומליות או רעש בנתוני האימון ובכך לתרום ליצירת עץ אשר מתאים במידה רבה לנתוני האימון, אך פחות תואם למקרה הכללי.

בכדי לטפל בבעיה זו ניתן להשתמש בגיזום ענפים. עצים לאחר ביצוע גיזום לרוב יהיו קטנים יותר, פחות מסובכים, יותר מהירים לריצה ולרוב יתאימו במידה טובה יותר לנתוני מבחן.

ישנן שתי גישות לגיזום:

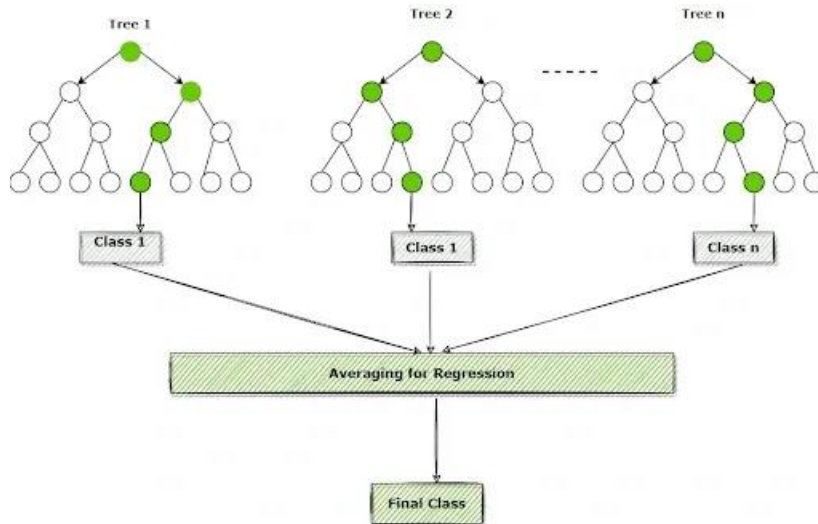
1. גיזום מוקדם – קביעת חסם תחתון בעבור ערך מדד הפיצול. במהלך בניית העץ, נמשיך לבנות את הענפים כל עוד ערך מדד הפיצול בעבור התכונה שנבחרה גבוה מהחסם התחתון. במידה והערך הגבוה ביותר מבין מדד הפיצול של התכונות הנותרות בעבור ענף מסוים נמוך מהחסם התחתון – נעצור את הבנייה בענף זה.
2. גיזום מאוחר – בסיום בניית העץ המקורי ניצור סט של עצים לאחר גיזום, על ידי בחירת צומת פנימית, גיזום תת העץ המתחיל בצומת זו והפיכת הצומת עצמה לעלה. לבסוף, נריץ את כל העצים בסט שבנינו על נתוני אימון חדשים ונבחר כעץ החדש את העץ אשר מסווג נתונים אלו באופן המדויק ביותר.

2.2.2 – יער אקראי, *Random forest*

שיטה סטטיסטית המנצלת את תכונות עץ ההחלטה בכדי להניב תוצאות טובות יותר.

בשיטה זו מתקיים תהליך בו נבחרים באופן אקראי מספר תכונות מבין כלל התכונות המצויות במאגר ובעבור תכונות אלו נבנה עץ החלטה שאינו גזום.

התהליך חוזר על עצמו כמספר הפעמים שנבחר עד לקבלת מספר גדול של עצים אשר לא מתואמים זה עם זה. הסיווג הסופי נקבע על פי הצבעת הרוב או מיצוע התחזיות של כלל העצים שנבנו.



תרשים 3: דוגמה למבנה יער אקראי, [Chaudhary, n.d.]

שיטה זו פשוטה למימוש, מתמודדת היטב עם נתונים לא מאוזנים, נמנעת מהתאמת יתר בצורה טובה יותר מעץ החלטה בודד ומניבה רמת דיוק גבוהה יותר אך במחיר של זמן חישוב ארוך יותר בשל הצורך בבניית מספר רב של עצי החלטה.

2.2.3 – למידה בייסיאנית פשוטה, naïve bayes

למידה בייסיאנית מסתמכת על חוק בייס (bayes theorem) לפיו ההסתברות המותנית של מאורע A בהינתן

מאורע B היא הסיכוי להתרחשותו בהנחה ש B אכן התרחש, או בנוסחה: $P(A|B) = \frac{P(A \cap B)}{P(B)}$

בתהליך הלמידה אנו משתמשים דווקא במשפט ההפוך:

בעבור אמת כלשהי X, נגדיר היפותזה H כסיכוי שסט נתונים יהיה שייך למשתנה מטר C ואז:

$$P(H|X) = \frac{P(H \cap X)}{P(X)} = \frac{P(X|H) * P(H)}{P(X)}$$

נוכל לנצל חוק זה בכדי לסווג מידע באמצעות Maximum posteriori hypothesis:

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D) = \operatorname{argmax}_{h \in H} P(D|h) * P(h)$$

אך מדובר בפעולה מסובכת לחישוב, שכן נדרש ידע קודם של הסתברויות רבות. לכן, נעדין את הנוסחה לכדי סיווג משתנה המטרה הסביר ביותר בהינתן מאגר מידע האימון.

בשלב הבא, בכדי להקל על החישוב נניח מספר הנחות מקלות:

1. D הוא מאגר האימון והוא מורכב מסטים של n תכונות אותן נסמן כ $X = (x_1, x_2, \dots, x_n)$ ומשתנה מטר.

2. נניח שישנם m משתני מטר: C_1, C_2, \dots, C_m

3. הסיווג נגזר מההסתברות המותנית המרבית, לפי $P(C_i|X) = \frac{P(X|C_i) * P(C_i)}{P(X)}$

4. נעדין את הבעיה למצב בו לכל תכונה יש תרומה אחידה ולכן נוכל לחשב את ההסתברויות לפי: $P(C_i|X) = P(X|C_i) * P(C_i)$

2.2.4 K Nearest Neighbor

שיטה המתבססת על למידה מתוך הסביבה.

לפי השיטה, בהינתן מאגר נתוני אימון המכיל סטים של נתונים n ממדיים (כלומר כל סט נתונים כולל n תכונות) ומשתנה מטר תואם לכל סט – נייצג כל סט נתונים כנקודה במרחב n ממדי ונקודה זו תשוך למשתנה המטרה התואם לסט.

כעת, כאשר ברצוננו לסווג סט נתונים חדש – נבנה ייצוג תואם לסט זה כנקודה במרחב ונחפש במאגר את k הנקודות הקרובות ביותר לנקודה זו.

הקרבה נבחנת על ידי מטריקת מרחק, כדוגמת מרחק אוקלידי: $distance(x_1, x_2) = \sqrt{\sum_{i=1}^n (x_{1i} - x_{2i})^2}$

לבסוף, סט הנתונים הנבחן יסווג כשייך למשתנה המטרה התואם לרוב הנקודות מבין k הנקודות השכנות שנמצאו.

נציין כי במקרים רבים הנתונים אינם תואמים בסדר גודלם בין תכונה לתכונה ולכן, בכדי לאפשר למידה יעילה מכל התכונות הקיימות, ננרמל את ערכי התכונות של סט נתונים לפני שנבחן את הקרבה שלו לסט נתונים אחר. שיטה נפוצה לנרמול ערכים היא נרמול Min-Max: כאשר \min_A ו- \max_A הם ערכי המינימום והמקסימום אשר תכונה A יכולה לקבל, ננרמל את ערך התכונה בעבור סט נתונים ספציפי, v , לפי: $v' = \frac{v - \min_A}{\max_A - \min_A}$. בנוסף, ניתן להשתמש במטריקת מרחק הכוללת רק חלק מהתכונות הקיימות ולבצע גיזום של תכונות שאינן תורמות לדיוק השיטה לאחר בדיקה אמפירית, בכדי לשפר את מהירות החישוב, במיוחד בעבור מאגרי נתונים הכוללים מספר רב של נתונים ושל תכונות.

[Han, Kamber, & Pei, 2012, pp. 423-424]



תרשים 4: המחשת שימוש באלגוריתם K-Nearest-Neighbors לסיווג סט נתונים חדש, [Raj, 2021]

2.2.5 – רגרסיה לינארית, Linear regression

השיטה מתאימה לחיזוי ערכים רציפים במקרה בו קיימת מידת התאמה לינארית כלשהי בין משתנה המטרה לבין אחת או יותר מהתכונות.

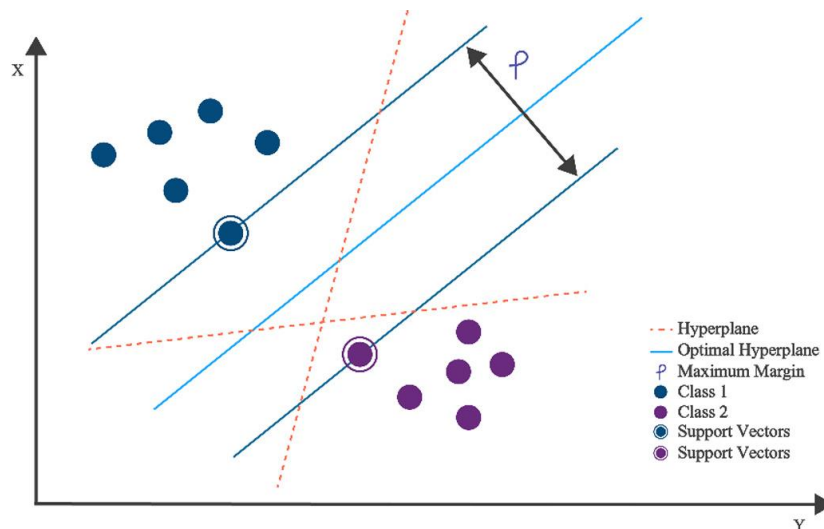
נשים לב לכך ששיטה זו מתאימה רק לערכים רציפים ולכן נצטרך להמיר חלק מהתכונות לתצורה רציפה במידה ואינן כאלו.

השיטה פשוטה לביצוע וניתנת להמחשה גרפית בקלות יחסית, כך שאם קיים קשר לינארי טוב – נוכל להציג מגמה באופן קל ונוח לצפייה.

מנגד, השיטה רלוונטית רק אם קיים קשר לינארי חזק מספיק. במידה ולא יימצא קשר חזק דיו בין התכונות נקבל מודל חלש מאוד.

2.2.6 – מכונת וקטורים תומכים, Support vector machines

Support vector machines, או בקיצור – SVM, הינו אלגוריתם לסיווג נתונים על ידי מציאת קו מפריד בעל רווח מקסימלי בין סוגי הנתונים הקיימים.



האלגוריתם משתמש במיפוי לא לינארי בכדי לייצג את מאגר הנתונים בממד גבוה יותר. לאחר מכן, בממד החדש, מבוצע חיפוש אחר גבול לינארי אשר יחלק את המרחב בצורה האופטימלית בין משתני המטרה הקיימים.

[Han, Kamber, & Pei, 2012, pp. 408-415]

תרשים 5: מבנה מכונת וקטורים תומכים, [Shaukat, Luo, & Varadharajan, 2023, p. 9]

נפריד את השימוש במכונת וקטורים תומכים לשני מקרים:

- במקרה בו הנתונים ניתנים להפרדה על ידי פונקציה לינארית – נוכל להעביר אינסוף גבולות לינאריים כך שהנתונים הנמצאים מכל צד של גבול זה יהיו שייכים למשתנה מטרה יחיד. לכן, בחיפוש אחר הגבול האופטימלי, נבחר בגבול אשר יוצר את המרווח הגדול ביותר בינו לבין דוגמאות הנתונים הקרובות אליו ביותר מכל אחד ממשתני המטרה (maximum marginal hyper-plane). גבול מפריד מאופיין על ידי הנוסחה:

$$W \cdot X + b = 0$$
 כאשר W הינו וקטור משקלים המתאים משקל לכל תכונה בסט נתונים X הינו וקטור תכונות. המישורים המקבילים לגבול המפריד, אשר מתלכדים עם הדוגמאות הקרובות ביותר אליו מכל אחד ממשתני המטרה נקראים וקטורים תומכים ומציאת הגבול האופטימלי מתבצעת בפועל על ידי חיפוש המרחק המקסימלי בין וקטורים תומכים אלו. כאשר ברצוננו לסווג סט נתונים חדש, נשתמש בנוסחה: $d(X^T) = \sum_{i=1}^l y_i \alpha_i X_i X^T + b_0$ כאשר y_i הוא משתנה המטרה המתאים לווקטור התומך X_i , X^T הינו סט נתוני המבחן, α_i, b_0 הם פרמטרים הנקבעים על פי אופטימיזציה של השיטה ו- l הוא מספר הווקטורים התומכים. שימוש בנוסחה זו יקנה לנו מידע באשר מיקום סט המבחן ביחס לווקטורים התומכים ובכך גם את סיווגו.
- במקרה בו הנתונים אינם ניתנים להפרדה על ידי פונקציה לינארית – נשתמש במיפוי לא לינארי בכדי להמיר את הנתונים לייצוג בממד גבוה יותר, אשר בו הנתונים יהיו ניתנים להפרדה על ידי פונקציה לינארית. המיפוי מבוצע בצורה אופטימלית באמצעות שימוש בפונקציית גרעין. פונקציות גרעין נפוצות כוללות:
 1. גרעין פולינומיאלי מדרגה h : $K(X_i, X_j) = (X_i \cdot X_j + 1)^h$ (Polynomial kernel of degree h)
 2. גרעין גאוסיאן פונקציית בסיס רדיאלי (Gaussian radial basis function kernel):

$$K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$$
 3. גרעין סיגמואיד (sigmoid kernel): $K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$
 לאחר שימוש בגרעין מתאים, נקבל נתונים מממד חדש, הניתנים להפרדה על ידי פונקציה לינארית ולכן נטפל בהם בהתאם למקרה הראשון, לקבלת מודל סיווג מתאים.

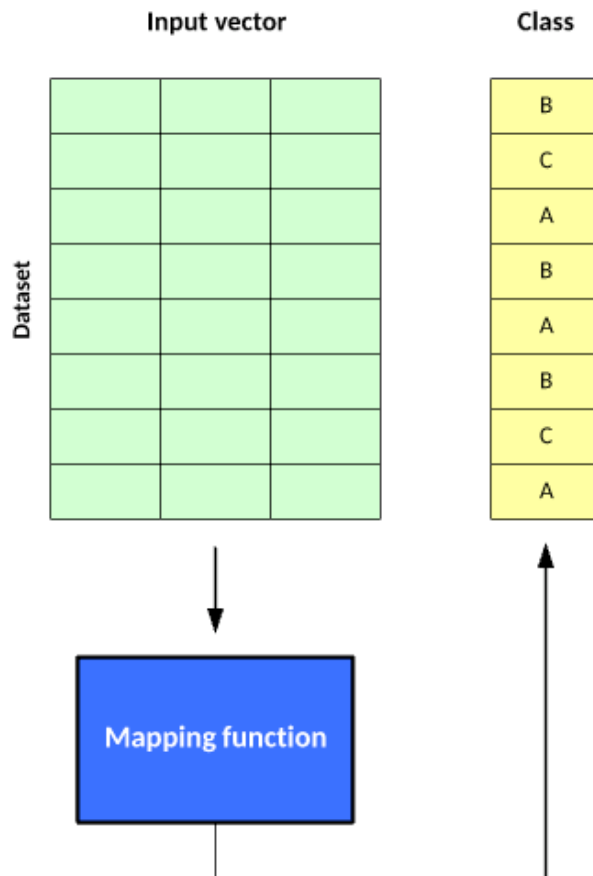
פרק 3: למידה לא מפקחת

3.1 – הצגת התהליך:

למידה לא מפקחת היא שיטה ללמידת מכונה בה על המודל ללמוד דפוסים מנתוני האימון בהיעדר ערכי תוצאה מוגדרים מראש אחרים עליו להתחקות.

היתרון המרכזי בלמידה לא מפקחת היא האפשרות לספק כמויות עצומות של מידע לשם אימון המודל, שכן לא נדרשת התערבות ידנית לשם סיווג כל רשומת מידע עם משתנה מטרה תואם, אך כמובן יתרון זה בא במחיר שאין אנו 'מכוונים' את המודל להתמקד במטרה בה אנו מעוניינים, אלא המודל לומד לבד כיצד להבדיל בין הרשומות השונות.

ניתן לשלב בין שתי גישות הלמידה על ידי שימוש בלמידה לא מפקחת לחשיפת דפוסים המתקיימים טבעית בנתוני בעיה כלשהי ולאחר מכן להגדיר מודל ללמידה מפקחת בהתאם לדפוסים שהתגלו.



תרשים 6: תרשים תהליך למידה לא מפקחת, [Jones, 2017]

3.2 – דוגמאות מודלים ללמידה לא מפקחת:

נציג מספר דוגמאות למודלים של למידה לא מפקחת:

3.2.1 – ניתוח אשכולות, Clustering

ניתוח אשכולות הוא תהליך בו מחלקים את רשומות הנתונים לתת קבוצות, לפי הדמיון בין הרשומות, כך שכל קבוצה מהווה אשכול. תהליך החלוקה לאשכולות מבוצע על ידי אלגוריתם חלוקה ולכן ייתכן ששימוש בשיטה שונה לחלוקה יגרום לקבלת אשכולות שונים בעבור אותו מאגר נתונים.

באמצעות ניתוח אשכולות ניתן לעיתים לגלות מגמות שלא היו ידועות לפני כן מתוך המידע הנתון. מכיוון שאשכול הוא אוסף של רשומות נתונים אשר דומים זה לה ושונים מרשומות השייכות לאשכולות אחרים – ניתן להתייחס לכל אשכול כאל סיווג ובכך להשתמש בשיטת ניתוח האשכולות ככלי מסווג בעבור רשומות מידע, למרות שבשונה משיטות בלמידה מפקחת – אין אנו קובעים את מספר הסיווגים מראש או את השייכות של רשומות המידע המשמשות לאימון המודל לסיווגים השונים.

קיימות מספר גישות לביצוע ניתוח אשכולות:

1. שיטות על בסיס הפרדה:

בהינתן מאגר של n אובייקטים, ניצור k קבוצות, כך ש $n \geq k$ וכל קבוצה מייצגת אשכול. כל קבוצה חייבת להכיל לפחות אובייקט אחד וניתן לקבוע שכל אובייקט יימצא בקבוצה אחת או להקל על התנאי ולאפשר לאובייקט להשתייך למספר קבוצות. השיטה מתבססת על מטריקת מרחק כלשהי בכדי לאמוד את מידת הדמיון בין כל אובייקט למשנהו.

2. שיטות על בסיס היררכיה:

מודל מבוסס היררכיה בונה מערכת אשר מסדרת את האובייקטים בקבוצות בסדר היררכי. שיטה היררכית יכולה להיות בונה או מפצלת.

שיטה בונה, הנקראת גם מלמטה למעלה, מתחילה כאשר לכל אובייקט יש קבוצה נפרדת משלו ובכל שלב מתבצע איחוד של קבוצות, עד שכל הקבוצות אוחדו לכדי קבוצה יחידה, או שהגענו לתנאי סיום אשר נקבע מראש.

לעומת זאת, בשיטה המפצלת, אשר נקראת גם מלמעלה למטה, מתחילים עם קבוצה יחידה אשר מכילה את כל האובייקטים ובכל שלב מפצלים את האשכולות לכדי אשכולות קטנים יותר, התהליך מסתיים כאשר לכל אובייקט יש אשכול משלו או שהגענו לתנאי סיום אשר נקבע מראש.

3. שיטות מבוססות צפיפות:

בעוד שרוב השיטות המבוססות על עיקרון הפרדה מגלות רק אשכולות בצורת ספירה (sphere), ניתן לבנות מודלים המבוססים על צפיפות כך שאשכולות יתגלו גם בצורה לא טריוויאלית.

שיטות אלו בוחנות גם הן את המרחק בין אובייקט אחד למשנהו על ידי מטריקת מרחק, אך מתבססות על יצירת אשכולות סביב מספר אובייקטים והוספת אובייקטים אשר לא סומנו עוד לאשכול הקרוב ביותר, עד להבאת האשכול למכסה מוגדרת מראש.

שיטה ספציפית המתבססת על עיקרון ההפרדה הינה K Means Clustering.

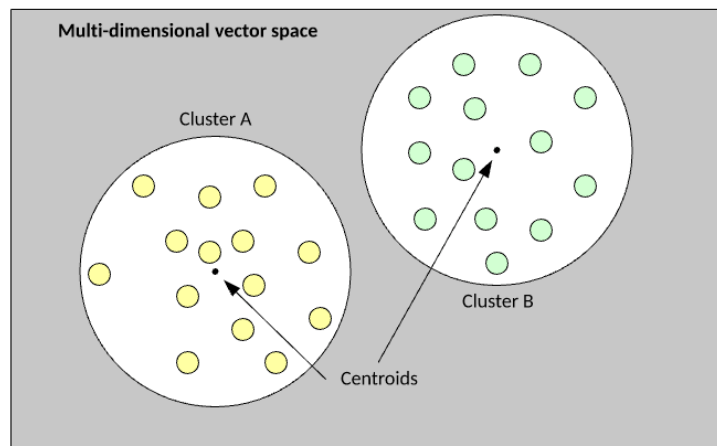
לפי אלגוריתם זה, כל אשכול מיוצג על ידי הממוצע הרב ממדי של האובייקטים השייכים אליו.

האלגוריתם פועל לפי השלבים הבאים:

1. האלגוריתם מגריל רנדומלית k אובייקטים ומגדיר אותם כמרכזי האשכולות הראשוניים.
2. כל האובייקטים משויכים לאשכול אליו הם הכי דומים, מבחינת מרחק אוקלידי של תכונות האובייקט ומרכז האשכול.
3. מחשבים את מרכזי האשכולות מחדש לפי ממוצע כל האיברים המשויכים לאשכול.
4. חוזרים על התהליך החל משלב 2 באיטרציה חדשה, עד לקבלת שתי איטרציות בהן לא התבצע שינוי.

תנאי העצירה, קבלת שתי איטרציות אשר מקנות את אותה התוצאה מספק יציבות לתוצאת האלגוריתם.

לאחר השלמת תהליך הלמידה, נוכל להציג בפני המודל אובייקט חדש ונקבל את האשכול אליו המודל מסווג את האובייקט בהתאם למרחק המינימלי ממרכזי האשכולות.



תרשים 7: דוגמה למימוש אלגוריתם K Means Clustering בעבור שני אשכולות, [Jones, 2017]

3.2.2 – חוקי הקשר, Association rules

חוקי הקשר הם כלי המאפשר לחלץ קשרים נפוצים בין תכונות במאגר נתונים נתון.

חוק הקשר מתואר בדרך כלל על ידי הנוסחה: $X_1, X_2, \dots, X_n \rightarrow Y$

כאשר כלל האיברים בנוסחה זו הם תכונות המתארות סט נתונים.

X_1, X_2, \dots, X_n מוגדרים כגוף החוק Y מוגדר כראש החוק. משמעות החוק היא שכאשר התכונות המצוינות בגוף החוק מתקיימות כולן – גם ראש החוק יתקיים בסבירות גבוהה.

בתהליך כריית חוקי הקשר משתמשים בשני מדדים מרכזיים: תמיכה ואמון.

התמיכה (support) בקבוצת תכונות מסוימת היא שיעור הדוגמאות במאגר בהן קבוצה זו מופיעה, מתוך כלל הדוגמאות במאגר.

האמון (confidence) בחוק מהצורה $X \rightarrow Y$ מוגדר כשיעור התמיכה בקבוצה $X \cup Y$ מתוך התמיכה ב- X .

בנוסחאות:

$$\text{support}(X \rightarrow Y) = P(X \cup Y)$$

$$\text{confidence}(X \rightarrow Y) = P(Y|X) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

כאשר אנו מחפשים אחר חוקי הקשר במאגר נתונים, נגדיר סף תמיכה מינימלית ואמון מינימלי ונחפש אחר חוקים אשר רמת התמיכה ורמת האמון בהם גדולה מהסף שהגדרנו. חוקים שהתגלו באופן זה נקראים חוקים חזקים.

קיימים מספר אלגוריתמים המשפרים את תהליך כריית חוקי הקשר. נציג שיטה אחת - אלגוריתם אפריורי (apriori algorithm):

בהתבסס על התכונה האפריורית, שקבוצה חלקית מתוך קבוצת פריטים שכיחה חייבת גם היא להיות שכיחה, אלגוריתם אפריורי פועל באופן איטרטיבי, כאשר באיטרציה k מוצאים את כל קבוצות הפריטים השכיחות אשר מכילות k פריטים.

כל איטרציה מורכבת משלושה שלבים:

1. יצירת מועמדים בגודל k על ידי צירוף שתי קבוצות פריטים שכיחות בגודל $k-1$ שנמצאו באיטרציה הקודמת.
2. גיזום מועמדים אשר לא מקיימים את התכונה האפריורית.
3. בדיקת התמיכה של המועמדים שלא נגזמו בשלב הקודם וסימון מועמדים מתאימים כקבוצות שכיחות בגודל k .

תהליך זה מתבצע עד לאיטרציה בה לא נמצאות קבוצות שכיחות כלל, או הגעה לתנאי עצירה מוגדר מראש.

פרק 4: רשתות נוירונים ולמידה עמוקה

4.1 – מבוא לרשתות נוירונים

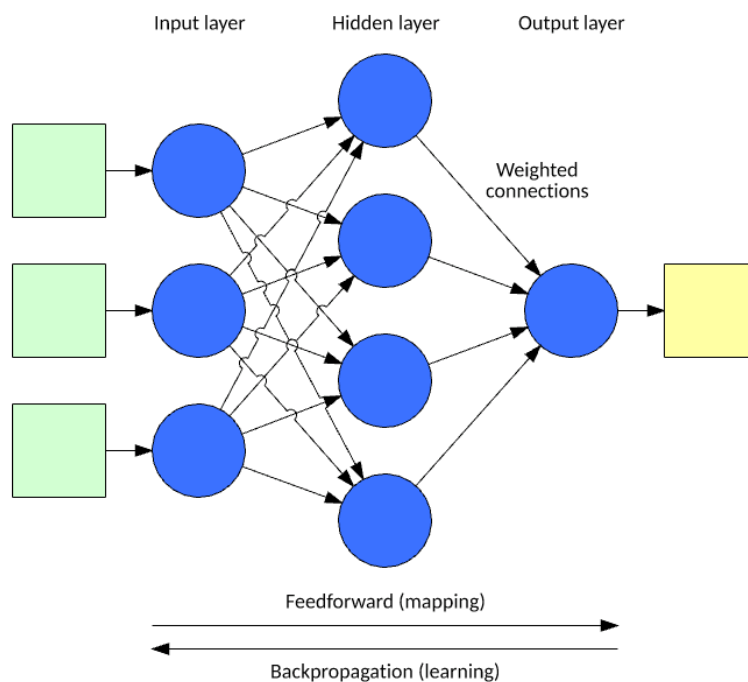
רשת נוירונים מלאכותית היא מודל חישובי בהשראת המבנה של רשתות הנוירונים הביולוגיות במוח.

המוח, במודל פשוט, מכיל מספר רב של נוירונים, תאי עצב אשר מסוגלים לקלוט ולשלוח מידע, המחוברים זה לזה ליצירת רשת תקשורת מורכבת, דרכה מסוגל המוח לבצע משימות מורכבות.

[Shalev-Shwartz & Ben-David, 2014, p. 268]

בביצוע משימה נתונה במוח משתתפים נוירונים רבים. הקשרים המרובים הקיימים בין נוירונים שונים במוח מאפשרים לבצע את המשימה ואף לרוב לבצע מספר משימות מורכבות בו זמנית, כגון ראייה, דיבור, תנועה ועוד.

בשל תכונות אלו, ובשל יכולתו הטובה של המוח ללמוד ולהשתפר בביצוע משימות, מהוות רשתות הנוירונים הביולוגיות השראה לפיתוחן של רשתות נוירונים מלאכותיות לשם למידת מכונה.



רשת נוירונים מלאכותית היא קבוצה של נוירונים המחוברים זה לזה בקשרים בעלי משקל.

הדומה למקבילתה הביולוגית הן באופן רכישת הידע - באמצעות תהליך הלמידה, והן באופן אכסון המידע - באמצעות הקשרים הקיימים בין הנוירונים.

הקשרים אמנם נשארים קבועים, אך המשקל המשוך לכל קשר, אשר מבטא את עוצמת הקשר משתנה.

תהליך הלמידה מתבצע על ידי ויסות משקלי הקשרים במטרה להשיג את המבנה והפונקציונליות הדרושה מהרשת.

בבסיסן, רשתות נוירונים מאופיינות במספר תכונות, כגון:

- מעבר בין שכבות לינאריות לשכבות לא לינאריות.
- שימוש בכלל השרשרת, בתהליך הנקרא פעפוע לאחור, לשם עדכון הפרמטרים של המודל.
- סביב שנות התשעים ושנות האלפיים המוקדמות, נצפתה האטה בקצב ההתקדמות ובפופולריות של שיטות מבוססות רשתות נוירונים. להאטה זו מיוחסים בעיקר שני גורמים:
- הסיבוכיות החישובית הגבוהה אשר כלולה בתהליך אימון רשת נוירונים, בעיקר בעבור רשת אשר מכילה מספר רב של שכבות.
- מאגרי הנתונים הנגישים לאימון מודלים היו קטנים יחסית, בעוד שרשתות נוירונים דורשות מאגר נתונים גדול בכדי להניב תוצאה טובה לתהליך האימון.

בשל שני גורמים אלו, עד אמצע העשור הראשון של המאה הנוכחית, שיטות סטטיסטיות כדוגמת עצי החלטה היו פופולריות יותר מרשתות הנוירונים של התקופה, בשל יכולתן להניב תוצאות טובות תוך כדי שימוש בסיבוכיות נמוכה יותר ויכולת אימון מהירה יותר.

אך בשנים האחרונות נצפו שיפורים בענף אשר הפכו את התחום לאטרקטיבי במיוחד.

החל משיפור ניכר בביצועי המחשבים בשל התקדמות הטכנולוגיה, דרך התבונה כי ניתן להשתמש במאיץ הגרפי (gpu) לשם ביצוע מספר רב של חישובים מקביליים במהירות וכלה בפיתוחם של מודלים מתוחכמים לשימוש ברשתות נוירונים כגון רשתות קונבולוציה (CNN) רשתות נשנות (RNN), שכבות נרמול ועוד.

[Zhang, Smola, Li, & Lipton, pp. 1.4 - 1.5]

4.2 – הצגת תהליך הלמידה

תהליך הלמידה של רשת נוירונים לפרטיו תלוי בסוג ומבנה הרשת, אך כולל בחובו מספר מאפיינים כלליים:



תרשים 9: מבנה מופשט של מודל רשת נוירונים

- הקלט הוא ערך נתונים או אצווה (batch) של ערכי נתונים מתוך מאגר הנתונים, אשר אנו מעוניינים לעבד ברשת הנוירונים.
- רשת הנוירונים עצמה מורכבת משכבה אחת או יותר, בהתאם למימוש הספציפי שנבחר. הפרמטרים הנלמדים, אותם ניתן לאמן ברשת הם המשקלים (weights) וערכי ההטיה (biases) השייכים לכל שכבה.
- הפלט הוא הערך אותו מחזירה רשת הנוירונים לקלט – הערך החזוי של המודל.

מודל יכול לפעול במצב אימון (training mode) או במצב חיזוי (evaluation mode).

במצב אימון המודל חוזר פלט בעבור הקלט הנתון ולאחר מכן משווה את הערך החזוי לערך המטרה הידוע בעבור הקלט הנתון. המודל מחשב את השוני בין שני ערכים אלו באמצעות שימוש בפונקציית מחיר (loss function) ולאחר מכן משתמש בתהליך הנקרא פעפוע לאחור (back propagation) בכדי לעדכן את הפרמטרים שלו בניסיון לשפר את דיוק המודל.

במצב חיזוי לעומת זאת, המודל אינו יודע את ערך המטרה בעבור הקלט ולכן מבוצע רק חיזוי של ערך פלט מתאים, המתבסס על אימון קודם.

4.2.1 – התפשטות לפנים, forward propagation

התפשטות לפנים, הנקרא גם מעבר לפנים, מתייחס לחישוב והאחסון של ערכי הביניים (כולל ערך הפלט) של רשת הנוירונים בסדר משכבת הקלט ועד שכבת הפלט.

[Zhang, Smola, Li, & Lipton, p. 5.3.1]

תהליך ההתפשטות לפנים מבוצע גם במצב אימון וגם במצב חיזוי. תהליך זה מסתיים בהגעה לערך חזוי בעבור הקלט הנתון.

4.2.2 – פונקציות מחיר, loss functions

תהליך הלמידה של אלגוריתם ללמידת מכונה מסתמך על ערך אותו אנו מעוניינים לייעל.

בעבור רשתות נוירונים, ערך זה מיוצר באמצעות שימוש בפונקציה הנקראת פונקציית מחיר, אשר מכמתת את המרחק בין ערך המטרה האמיתי לבין הערך החזוי אותו החזיר המודל.

המחיר יהיה לרוב לא שלילי, קטן יותר ככל שהערך החזוי על ידי המודל קרוב יותר לערך המטרה האמיתי ושווה לאפס במקרה של חיזוי מושלם.

[Zhang, Smola, Li, & Lipton, p. 3.1.1.2]

קיים מגוון רחב של פונקציות מחיר וחלקן מתאימות טוב יותר בעבור משימות למידה בעלות אופי ספציפי.

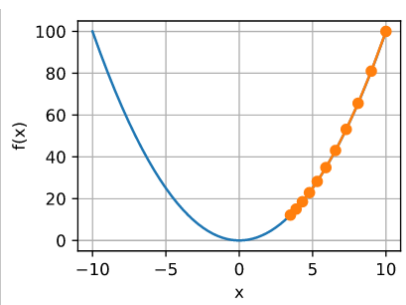
נציין מספר פונקציות מחיר פופולריות::

בהינתן x וקטור ערכים חזויים y וקטור של ערכי המטרה התואמים, כאשר שניהם בגודל $n \geq 1$

- ערך שגיאה ממוצעת, mean absolute error: $MAE(x, y) = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$
 - ערך שגיאה ממוצעת בריבוע, mean squared error: $MSE(x, y) = \frac{\sum_{i=1}^n (y_i - x_i)^2}{n}$
 - אנטרופיה צולבת, cross entropy loss – פונקציה אותה סיקרנו בסעיף 2.2.1, כקריטריון פיצול של עצי החלטה. פונקציה זו נבחרת לרוב כאשר ברצוננו לסווג נתונים לערכי מטרה מבין רשימה מוגדרת וסופית של ערכים אפשריים.
 - Negative log likelihood loss – פונקציה דומה לאנטרופיה הצולבת, אשר מעצימה הסתברויות קרובות למטרה ומקטינה הסתברויות רחוקות ממנה על ידי שימוש בלוגריתמים.
- בכדי לבצע למידה נכונה במודל, ננסה למזער את ערך המחיר בין הערכים החזויים לבין ערכי המטרה התואמים בעבור דוגמאות הנתונים במאגר האימון ובכך להשיג תוצאות חזויות מדויקות יותר.

4.2.3 – אלגוריתם מורד הגרדיאנט, gradient descent

כפי שראינו בסעיף הקודם, תהליך הלמידה מונע על ידי מזעור פונקציית מחיר. מזעור זה מבוצע באמצעות שימוש באלגוריתם הנקרא מורד הגרדיאנט (gradient descent).

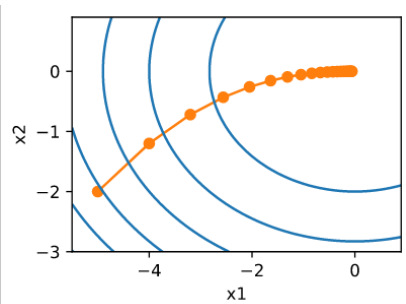


תרשים 10: דוגמה לשלבי אלגוריתם מורד הגרדיאנט בממד יחיד

מכיוון שמדובר בפונקציה מורכבת, ננסה להגיע למינימום על ידי נקיטת צעדים קטנים יחסית בכיוון הנכון, לפי השיפוע בנקודה בה אנו נמצאים.

גודל הצעד אותו ניקח מושפע מקצב הלמידה (learning rate), או בקצרה (lr), אותו נגדיר בעבור המודל.

בעבור פונקציה חד ממדית, הצעד הבא בכל שלב מחושב על פי הנוסחה: $step = -lr * \frac{df(x)}{dx}$ כך שנעדכן את הערך x לכדי $x + step$. דוגמה לתהליך זה מצויה בתרשים 10.



תרשים 11: דוגמה לשלבי אלגוריתם מורד הגרדיאנט בשני ממדים

אך כאשר ברצוננו למזער פונקציה המכילה יותר משתנים – נבחר להתקדם בכיוון אשר מפחית את תוצאת הפונקציה במהירות הגבוהה ביותר, זהו כיוון השיפוע המירבי. הצעד הבא במקרה זה יחושב על פי:

$step = -lr * \nabla C = -lr * (\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n})$ וכך, נעדכן את וקטור הערכים x לכדי $(x_1, \dots, x_n) + step$. דוגמה לתהליך הנ"ל, במקרה הפרטי של פונקציה דו ממדית, מצויה בתרשים 11.

[Zhang, Smola, Li, & Lipton, p. 12.3]

ישנם מספר אלגוריתמים נוספים המתבססים על אלגוריתם מורד הגרדיאנט ומציעים שיפורים לו, מהם נציג את השניים הבאים:

1. Stochastic gradient descent (SGD) – שימוש באלגוריתם מורד הגרדיאנט הבסיסי כולל הרצה של כל מאגר הנתונים ורק לאחר מכן חישוב הגרדיאנטים בעבור הצעד הבא, תהליך הדורש משאבים רבים. אלגוריתם SGD מתמודד עם בעיה זו על ידי שימוש בצעדים שחושבו על אצוות קטנות (mini batches) של נתונים, אשר נדגמו באופן רנדומלי מתוך מאגר הנתונים כך שישמשו כקירוב טוב למאגר הנתונים כולו תוך כדי שמירה על כמות חישובים נדרשת קטנה יחסית בעבור כל צעד.
2. Adam – אלגוריתם זה מוסיף תכונה הנקראת מומנטום (momentum) לחישוב צעדי הגרדיאנט, תכונה אשר עוזרת 'להתעלם' מרעשי רקע שנוספו משימוש בגרדיאנט האקראי. בנוסף, האלגוריתם מבצע נרמול של ערכי הגרדיאנט בעבור כל פרמטר (gradient step scaling) ובכך מאפשר לפרמטרים המכילים ערכים קטנים יחסית לתרום גם לתהליך הלמידה (אחרת, ערכים אלו היו 'נעלמים' ביחס לערכי פרמטרים אחרים גדולים יותר).

4.2.4 – התפשטות לאחור, *back propagation*

פעפוע לאחור (back propagation) מתייחס לשיטת החישוב של ערכי הגרדיאנט בעבור הפרמטרים הנלמדים המצויים ברשת נוירונים. השיטה עוברת על רשת הנוירונים בסדר הפוך, משכבת הפלט לשכבת הקלט לפי כלל השרשרת. האלגוריתם שומר את כל ערכי הביניים (שהם נגזרות חלקיות) הדרושים לשם חישוב הגרדיאנט לפי כל אחד מפרמטרי הרשת.

[Zhang, Smola, Li, & Lipton, p. 5.3.3]

4.2.5 – פונקציות אקטיבציה, *activation functions*

רוב השכבות החישוביות ברשתות נוירונים מבצעות טרנספורמציה לינארית על הקלט. אם כך, שרשר של מספר שכבות לינאריות בזו אחר זו מקביל לביצוע טרנספורמציה לינארית בודדת.

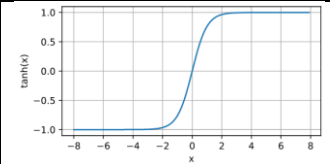
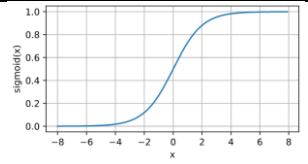
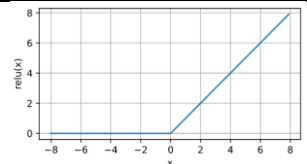
לינאריות יכולה להיות בעייתית בעבור מודלים של רשתות נוירונים, שכן לינאריות מניחה מונוטוניות – עלייה בערכו של אחד הנתונים חייב לגרור תמיד עלייה או תמיד ירידה בפלט ובאופן כללי, הכוח החישובי של מודל לינארי, מסובך ככל שיהיה, מוגבל ביחס למודל לא לינארי.

לכן, בכדי הגדיל את כוחה החישובי של רשת נוירונים אנו נדרשים לשבור את הלינאריות בין השכבות החישוביות באמצעות שימוש בפונקציות הנקראות פונקציות אקטיבציה לא לינאריות (non-linear activation functions).

פונקציית אקטיבציה קובעת בעבור כל נוירון האם יש להפעילו באמצעות ביצוע חישוב על סכום המשקלים וערך ההטיה המתאימים לו.

[Zhang, Smola, Li, & Lipton, p. 5.1]

נציג מספר פונקציות אקטיבציה נפוצות:

שם	נוסחה	המחשה
Tanh	$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
Sigmoid	$sigmoid(x) = \frac{1}{1 + e^{-x}}$	
ReLU	$ReLU(x) = \max(x, 0)$	

4.2.6 – שכבות נרמול, *normalization layers*

רשתות נוירונים עמוקות מורכבות ממספר רב של שכבות ומביעות כוח חישובי רב. עם זאת, תהליך האימון של רשתות מסוג זה הינו מורכב ובאופן מיוחד קשה לבצע אופטימיזציה של פרמטרי הרשת בזמן סביר.

ישנן מספר דרכים להתמודד עם בעיה זו כאשר המרכזית בהן היא הוספת שכבות נרמול:

שכבות שתפקידן לנרמל את הנתונים העוברים דרכן לסקאלה ברורה, לרוב $[-1, 1]$, בכדי לקבל נתונים בהתפלגות סטנדרטית, עליהם קל יותר לבצע חישובי גרדיאנט מורכבים. פעולת הנרמול מוודאת שתכונות שונות מצויות בסקאלה ברורה וכך מסייעת לייצב את שלבי מורד הגרדיאנט.

השימוש בשכבות נרמול מאפשר להשתמש בקצב למידה (lr) גבוה יותר, או להתייצב סביב תוצאה טובה מהר יותר.

נשתמש בשכבות אלו בדרך כלל לפני או אחרי שכבות האקטיבציה במודל.

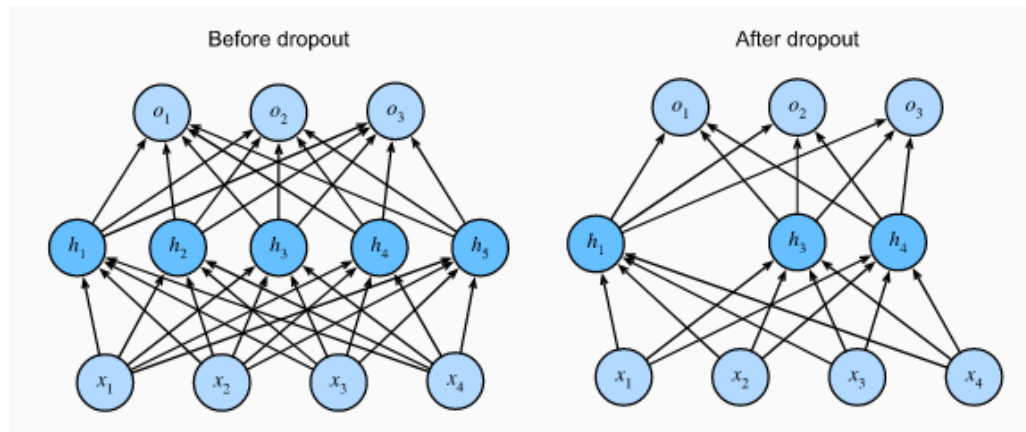
נציג שתי שכבות נרמול פופולריות:

- Batch norm – שכבה זו שומרת ממוצע (mean) וסטיית תקן (standard deviation) בעבור כל תכונה בקלט על פני האצוה הנוכחית ומשתמשת בערכים אלו לשם נרמול התכונות.

- Layer norm – שכבה זו שומרת ממוצע וסטיית תקן בעבור כל קלט, על פני כל תכונות הקלט (ללא תלות בשאר הקלטים באצווה) ומשתמשת בערכים אלו לשם נרמול התכונות.

4.2.7 – שכבת dropout

שכבה שתפקידה לייצר יתירות ביכולת החישוב של המודל. השכבה מנסה לכפות על המודל ללמוד לבצע חיזוי באופן שאינו מסתמך על אף נוירון אחד ספציפי על ידי איפוס חלק מהנוירונים הפעילים בקלט באופן רנדומלי כאשר המודל נמצא במצב אימון.



תרשים 12: דוגמה לשימוש בשכבת dropout, [Zhang, Smola, Li, & Lipton, p. 5.6.1]

לשימוש בשכבת dropout מספר יתרונות:

- מכיוון שבכל מעבר על נתוני אימון חלק מהנוירונים מתאפסים – הרשת לומדת לבצע חיזוי על סמך מספר קטן יותר של נוירונים ובכך חוסכת שימוש במשאבים.
- רשתות המממשות שכבות dropout נוטות לבצע הכללה בצורה טובה יותר, בשל חוסר היכולת של המודל לבסס את החיזוי על אף נוירון ספציפי.
- שכבות אלו מוסיפות רעש לתהליך בעת הלמידה ובכך מלמדות את הרשת להתמודד יותר טוב עם רעש חיצוני.

4.3 – יחידת הפרספטרון

פרספטרון הינו מודל ללמידת מכונה המכיל נוירון בודד. הפרספטרון מהווה גם יחידה בסיסית של רשתות נוירונים.

לנוירון מחוברים אוסף של קלטים (inputs) אותם נסמן ב- x_1, \dots, x_n , כאשר עוצמת הקשר בין כל קלט לנוירון מיוצגת על ידי משקל תואם, w_1, \dots, w_n . הנוירון מכיל גם ערך הטיה (bias), אותו נסמן ב- b המאפשר להטות את עוצמת האות העובר דרכו.

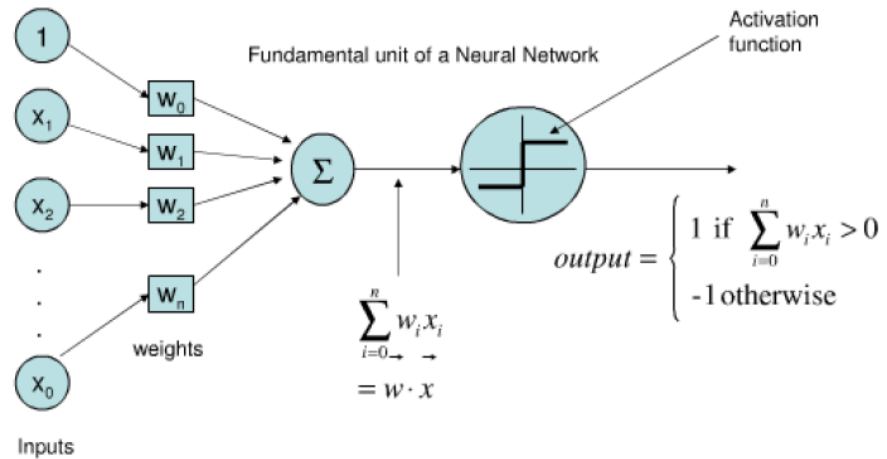
ערכו הצבור של הנוירון מושג באמצעות חישוב סכום כל הערכי הקלט בהתחשב במשקל המשויך לכל אחד מהם, בתוספת ערך ההטיה של הנוירון.

$$value = \sum_{i=0}^n w_i x_i + b$$

לבסוף ערך הנוירון מועבר לפונקציית אקטיבציה (activation function) והתוצאה מהווה את הפלט של הנוירון, אשר יסומן כ- y .

נשים לב לכך שהמודל מקבל אוסף של קלטים ונדרש ללמוד להחזיר פלט מתאים על ידי תהליך הלמידה. המודל מבצע זאת על ידי ביצוע שינויים בערכי המשקלים ובערך ההטיה, עד להפסקת האימון או הגעה להתייצבות בערכים.

הפרספטרון הינו מודל פשוט וביכולתו להחזיר ערך תוצאה בודד ולכן יכולתו החישובית מוגבלת לעומת מודלים מורכבים יותר.



תרשים 13: מבנה הפרספקטרום

4.4 – רשתות נוירונים בעלות קישוריות מלאה, feed-forward neural networks

4.4.1 – שכבה לינארית

שכבה לינארית מורכבת משני רכיבים נלמדים: משקלים (weights) והטיה (bias).

כל קלט מחובר לכל פלט באמצעות משקל מסוים ובנוסף, לכל פלט קיים ערך הטיה המתאים לו.

הפעולה המבוצעת בשכבה היא: $xW^T + b = y$

כאשר x הינו וקטור הקלט בגודל n , y הוא וקטור הפלט בגודל m , W היא מטריצת המשקלים ו- b הוא וקטור ההטיות. מטריצת המשקלים היא מגודל (m, n) . החישוב של כל אחד מ- i איברי הפלט $1 \leq i \leq m$ הוא:

$$y_i = \sum_{j=1}^n (x_j * w_{i,j}) + b_i$$

4.4.2 – מבנה הרשת

רשתות במבנה קישוריות מלאה מספקות גמישות רבה, שכן הרשת יכולה ללמוד להסיק תובנות לכל פלט מקומבינציה של כל נתוני הקלט.

רשת ממוצעת בנויה משרשר של בלוקים המכילים: שכבה לינארית, פונקציית אקטיבציה (כדוגמת ReLU) ושכבת נרמול (batch/layer normalization).

לאחר הבלוק האחרון יימצא לרוב ראש סיווג אשר יכול להכיל שכבה לינארית ולבסוף שכבת softmax המשמשת לשם השגת הסתברויות משתני המטרה לפי חיזוי המודל, או ראש רגרסיה אשר לרוב יכיל שכבה לינארית עם נירון פלט יחיד, כתלות בסוג הרשת ומשימתה.

4.5 – רשתות קונבולוציה, Convolutional Neural Networks

4.5.1 – מוטיבציה

בעוד שרשתות נוירונים בעלות קישוריות מלאה הינן בעלות כוח הבעה רב ועל ידי בחירת משקלים מתאימים ביכולתן להשיג תוצאות טובות במספר רב של משימות שונות, קיימות מספר משימות אליהן רשתות מסוג זה אינן אופטימליות, וביניהן – ניתוח תמונה.

בבעיות של ניתוח תמונה נתוני הקלט שלנו הם ערכי הצבעים (RGB) של כל פיקסל בתמונה. ככל שרזולוציית התמונה גבוהה יותר – כך גדל משמעותית מספר ערכי הקלט בעבור כל תמונה. משום ששכבה לינארית בעלת קישוריות מלאה מכילה חיבור בין כל נירון קלט לכל נירון פלט – במהרה רשתות מסוג זה בעבור ניתוח תמונה הופכות ליקרות מאוד עד לבלתי ניתנות להרצה מבחינה חישובית. מה גם שבעקבות הכמות העצומה של הפרמטרים שנקבל ברשת מסוג זה יידרש סט אימון גדול במיוחד בכדי לאמן את הרשת.

בנוסף, רשתות בעלות קישוריות מלאה מכילה גמישות בה אין אנו בהכרח מעוניינים – הרשת תלמד את מיקומי ערכי הפיקסלים ותייחס חשיבות לערך פיקסל לעומת התמונה כולה. זאת בעוד שבתמונות טבעיות ישנה חשיבות ללוקאליות של ערכים, לשם זיהוי עצמים בתמונה.

בעיות אלו ועוד רשתות קונבולוציוניות נועדו לפתור, באמצעות גרעין בהשראת שיטות קודמות של עיבוד תמונה.

4.5.2 – שכבת הקונבולוציה

Input		Kernel		Output																	
<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>	0	1	2	3	4	5	6	7	8	*	<table><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	=	<table><tr><td>19</td><td>25</td></tr><tr><td>37</td><td>43</td></tr></table>	19	25	37	43
0	1	2																			
3	4	5																			
6	7	8																			
0	1																				
2	3																				
19	25																				
37	43																				

תרשים 14: דוגמה לביצוע פעולת קונבולוציה עם קלט וגרעין נתונים

פעולת הקונבולוציה (convolution), או בשמה המתמטי – קורלציה צולבת (cross correlation) מבוצעת על ידי הצבת גרעין הקונבולוציה אל מול תת מטריון של הקלט, ביצוע כפל איבר-איבר בין שתי המטריונות וסכימת התוצאה לקבלת ערך פלט יחיד. חוזרים על פעולה זו עם תת מטריון קלט אשר הוזזה ימינה \ למטה עד למעבר על כל תת המטריונות האפשריות. נוסחה מתמטית לביצוע הפעולה תופיע בסוף תת סעיף זה, לאחר הצגת שאר המושגים הרלוונטיים לנושא.

בעבור קונבולוציה פשוטה, עם קלט בגודל $n_h \times n_w$ וגרעין בגודל $k_h \times k_w$, נקבל פלט בגודל $(n_h - k_h + 1) \times (n_w - k_w + 1)$.

[Zhang, Smola, Li, & Lipton, p. 7.2.1]

פעולת הקונבולוציה מושפעת בין היתר ממספר רכיבים:

ערוצים (channels) – לרוב, תמונות מקור יכללו שלושה ערוצים בעבור עוצמת האדום, הירוק והכחול (RGB) בכל פיקסל בתמונה.

בתהליך ביצוע הקונבולוציה אנו ננסה לרוב להוריד את הרזולוציה של התמונה בתמורה להגדלת מספר הערוצים. אינטואיטיבית ניתן לחשוב על ערוץ כאל פילטר, מסכה אשר נועדה לזהות תכונה מסוימת בקלט.

Input		Kernel		Output																																													
<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>2</td><td>0</td></tr><tr><td>0</td><td>3</td><td>4</td><td>5</td><td>0</td></tr><tr><td>0</td><td>6</td><td>7</td><td>8</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	1	2	0	0	3	4	5	0	0	6	7	8	0	0	0	0	0	0	*	<table><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	=	<table><tr><td>0</td><td>3</td><td>8</td><td>4</td></tr><tr><td>9</td><td>19</td><td>25</td><td>10</td></tr><tr><td>21</td><td>37</td><td>43</td><td>16</td></tr><tr><td>6</td><td>7</td><td>8</td><td>0</td></tr></table>	0	3	8	4	9	19	25	10	21	37	43	16	6	7	8	0
0	0	0	0	0																																													
0	0	1	2	0																																													
0	3	4	5	0																																													
0	6	7	8	0																																													
0	0	0	0	0																																													
0	1																																																
2	3																																																
0	3	8	4																																														
9	19	25	10																																														
21	37	43	16																																														
6	7	8	0																																														

תרשים 15: דוגמה לביצוע קונבולוציה לאחר ריפוד של שורה ועמודה מכל צד של מטריון הקלט

ריפוד (padding) – נשים לב כי בקונבולוציה פשוטה אנו נוטים לאבד מידע הצבור בפיקסלים אשר מצויים בקצוות הקלט. לרוב אנו משתמשים בגרעינים קטנים יחסית, כך שאיבוד המידע אינו משמעותי בכל פעולה אך לאחר ביצוע מספר רב של קונבולוציות כמות המידע הנאבד גדל משמעותית.

בכדי לפתור בעיה זו – נרפד את הקלט בשורות ועמודות של ערכים מוגדרים מראש (לרוב 0 או 1) לפני ביצוע פעולת הקונבולוציה.

[Zhang, Smola, Li, & Lipton, p. 7.3.1]

Input		Kernel		Output																																	
<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>2</td><td>0</td></tr><tr><td>0</td><td>3</td><td>4</td><td>5</td><td>0</td></tr><tr><td>0</td><td>6</td><td>7</td><td>8</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	1	2	0	0	3	4	5	0	0	6	7	8	0	0	0	0	0	0	*	<table><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	=	<table><tr><td>0</td><td>8</td></tr><tr><td>6</td><td>8</td></tr></table>	0	8	6	8
0	0	0	0	0																																	
0	0	1	2	0																																	
0	3	4	5	0																																	
0	6	7	8	0																																	
0	0	0	0	0																																	
0	1																																				
2	3																																				
0	8																																				
6	8																																				

תרשים 16: דוגמה לביצוע קונבולוציה עם קצב פסיעה באורך 2 וברוחב 2

קצב פסיעה (stride) – לעיתים, בכדי לחסוך במשאבים או להאיץ את תהליך הקטנת הרזולוציה, נרצה להעביר את גרעין הקונבולוציה על גבי הקלט ביותר מפיקסל אחד ימינה ולאו למטה בכל צעד. כך נדלג על ערכי הביניים בין כל שני צעדים ונוכל להקטין את גודל תמונת הפלט ואף להשתמש בגרעינים גדולים יותר מבלי לאפשר לפיקסל קלט מסוים להשפיע על תוצאת מספר פיקסלים של פלט.

[Zhang, Smola, Li, & Lipton, p. 7.3.2]

לבסוף, נציין שבעבור שכבת קונבולוציה עם קלט בעל c_{in} ערוצי קלט בגודל $n_h \times n_w$, עם c_{out} גרעינים בגודל $k_h \times k_w$ המתאימים ל c_{out} ערוצי פלט רצויים, עם ריפוד p_h, p_w וקצב פסיעה s_h, s_w – נקבל פלט בגודל:

$$c_{out} \times \left\lfloor \frac{n_h - k_h + p_h + s_h}{s_h} \right\rfloor \times \left\lfloor \frac{n_w - k_w + p_w + s_w}{s_w} \right\rfloor$$

וערכי הפלט יחושבו על ידי:

$$Y_{r,t}^m = \sum_{l=1}^{c_{in}} \sum_{i=1}^{k_h} \sum_{j=1}^{k_w} k_{i,j}^{m,l} * X_{p_{r*s_h+i, t*s_w+j}}^l$$

בעבור $1 \leq m \leq c_{out}$ ערוצי פלט ו- X_p^l מטריצת הקלט בערוץ h לאחר הפעלת הריפוד.

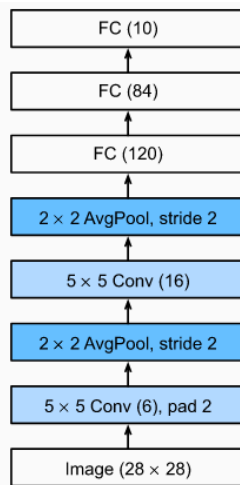
4.5.3 – מבנה רשת קונבולוציה

מבנה טיפוסי של רשת קונבולוציה מתחלק לשני חלקים:

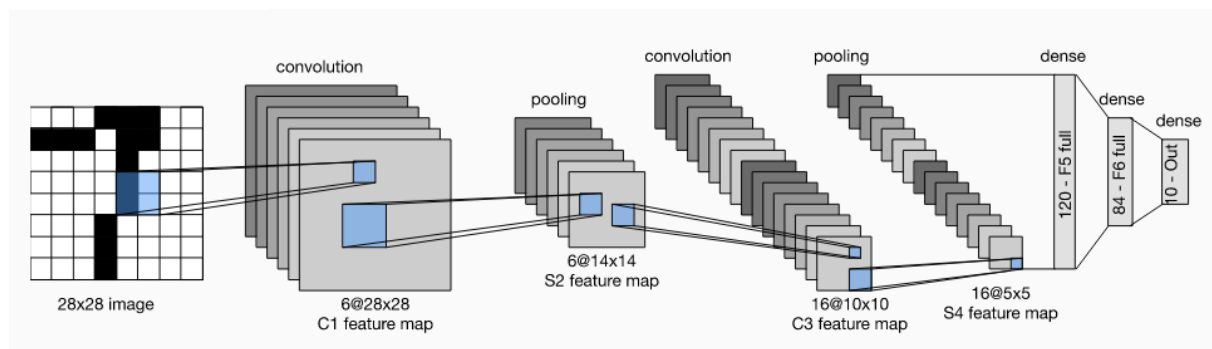
- שרשר של בלוקים מחלצי מאפיינים, הכוללים שכבת קונבולוציה, פונקציית אקטיבציה, שכבת נרמול ולא שכבת איגום – שכבה המבצעת הפחתה (downsampling) במימדי הקלט.
- בלוקים אלו מקטינים את מימדי הקלט ובו בעת מגדילים את כמות הערוצים ובכך מנסים לחלץ מאפיינים בתמונה כאשר שכבות עמוקות יותר ברשת בוחנות מאפיינים ברמה גבוהה יותר.
- ראש סיווג, אשר מורכב משכבה לינארית אחת או יותר. ראש הסיווג מצמצם את המאפיינים אשר אותרו בחלק הראשון של הרשת לכדי סיווג לערך המטרה.

נוכל להציג מבנה של רשת קונבולוציה באמצעות דיאגרמת בלוקים או כתרשים זרימה. כדוגמה, מבנה רשת *LeNet* מוצג בתרשימים 17 ו-18.

[Zhang, Smola, Li, & Lipton, p. 7.6.1]



תרשים 17: דיאגרמת בלוקים של מבנה רשת *LeNet*



תרשים 18: תרשים זרימת מידע ברשת *LeNet*

קיימות רשתות קונבולוציה מודרניות יותר, כדוגמת *ResNet*, אשר כוללת פונקציונליות של חיבור דילוג – חיבור הפלט של שכבה לקלט באמצעות פרמטר נלמד, דבר המאפשר שימוש ברשתות עמוקות יותר ובעלות יכולת אופטימיזציה גבוהה יותר, אך בבסיסן מבנה רשתות דומה.

4.6 – רשתות נשנות, Recurrent Neural Networks

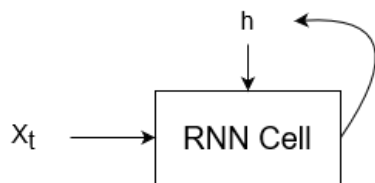
4.6.1 – מוטיבציה

עד כה המודלים שהצגנו התמודדו עם סט נתונים לא סדרתיים.

בטיפול בנתונים סדרתיים, כגון מדידות שבוצעו לאורך זמן או ניתוח משפטים בשפה טבעית, יש לתת את הדעת בעת ניתוח הנתונים לסדר הופעתם. למשל, שני המשפטים: "טוב מאוד, לא רע" ו- "רע מאוד, לא טוב" מכילים את אותן המילים אך בסדר שונה ועובדה זו משנה לחלוטין את משמעותם.

לשם טיפול בנתונים מסוג זה, נציג מבנה רשת אשר מביא בחשבון את סדר הופעת הנתונים – רשת נשנית (Recurrent Neural Network – RNN).

4.6.2 – תא נשנה, RNN Cell

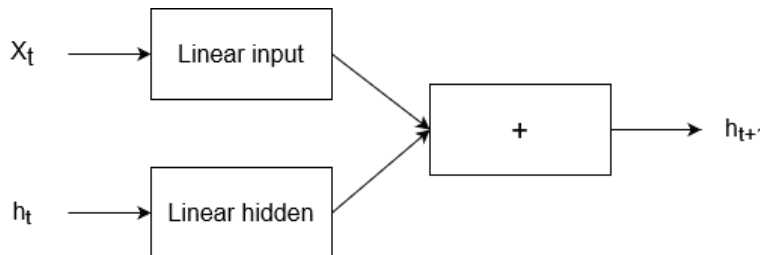


תרשים 19: סכמת פעולת תא נשנה, כאשר x_t הוא הקלט ה- t בסדרה ו- h הוא המצב החבוי של התא

רכיב הרשת הבסיסי המאפשר לרשת נוירונים להביא בחשבון את סדר הופעת הקלטים נקרא תא נשנה, או תא אלמן (Elman cell). התא מקבל את הקלטים לפי סדרם ומכיל מצב חבוי (hidden state) אשר מתעדכן עם כל הזנת קלט. בכל הזנת קלט – המצב החבוי מתעדכן בהתאם לקלט הנוכחי והמצב החבוי השמור בתא.

התא מפיץ את המצב החבוי החדש על ידי העברת הקלט החדש דרך שכבה לינארית אחת, העברת המצב החבוי הקודם דרך שכבה לינארית שנייה וביצוע פונקציית אקטיבציה (לרוב \tanh) על חיבור הפלטים של שתי השכבות הלינאריות.

בנוסחה: $h_{t+1} = \tanh(W_{input} * X_t + b_{input} + W_{hidden} * h_t + b_{hidden})$



כאשר הפרמטרים הנלמדים בתא הינם:

$W_{input}, b_{input}, W_{hidden}, b_{hidden}$

הקלטים מוזנים בזה אחר זה, דבר הגורר עדכון של המצב החבוי בכל פעם, עד לקבלת המצב החבוי הסופי אשר מייצג את עיבוד כל הקלט בצורה סדרתית.

תרשים 20: סכמת עדכון מצב חבוי בתא נשנה בצעד יחיד, על סמך קלט חדש ומצב חבוי קודם

4.6.3 – מבנה רשת נשנית

רשת נשנית טיפוסית מכילה שני חלקים:

- יצירת ייצוג חבוי – נתוני הקלט מועברים בזה אחר זה דרך תא נשנה בכדי לייצר מצב חבוי המייצג את הקלט כולו. ניתן להרחיב את המודל לשימוש במספר שכבות של תאים נשנים, כך שכל תא לאחר התא הראשון יקבל כקלט את הייצוג החבוי של השכבה הקודמת לו.
- ראש סיווג אשר מורכב משכבה לינארית אחת או יותר ומסווג את הייצוג החבוי שהושג בחלק הראשון למשתנה המטרה המתאים ביותר.

קיימים שיפורים לתא הנשנה הבסיסי, כדוגמת $Long\ Short\ Term\ Memory - LSTM$, אשר מוסיפים יכולת שכחה וחיבור דילוג לתא ובכך מאפשרים למידה יותר מוצלחת של קלטים ארוכים יותר ותורמים להתמודדות עם בעיית הגרדיאנט הנעלם/מתפוצץ.

4.7 – ארכיטקטורת מקודד-מפרש, Encoder-Decoder

4.7.1 – מוטיבציה

המודלים אשר הצגנו עד כה היו מודלים של סיווג או רגרסיה, המשוויכים לאסכולה של למידה מפקחת.

בשל היותם מודלים של למידה מפקחת, על סט הנתונים הדרוש לאימוןם לכלול ערך מתויג של משתנה יעד, אשר לרוב מקטין את גודל סט הנתונים האפשרי, או מחייב השקעה רבה של זמן ומשאבים להכנת נתוני האימון מראש. בנוסף, ככל שמודל מורכב יותר ומכיל יותר פרמטרים נלמדים – נצטרך להשתמש בכמות גדולה יותר של נתוני אימון בכדי להגיע לתוצאה טובה של תהליך הלמידה.

האינטרנט מכיל כמויות אסטרונומיות של נתונים, כאשר רובם המוחלט אינו מתויג. כעת נציג ארכיטקטורת רשת המאפשרת שימוש בנתונים אלו לכדי ביצוע תהליך למידה – ארכיטקטורת מקודד מפרש בכלל ומקודד עצמי (autoencoder) בפרט.

4.7.2 – מבנה הרשת

רשת מקודד-מפרש בנויה, לפי השם, מרכיב מקודד ורכיב מפרש.

תפקידו של המקודד הוא לבנות מהקלט ייצוג חבוי ממצא ככל שניתן. המפרש בתורו מנסה לבנות את הפלט המבוקש מתוך ייצוג חבוי זה.

ארכיטקטורה זו כוללת מגוון רחב של סוגי רשתות, כתלות במשימה אותה אנו מעוניינים לבצע ובסוג המידע בו אנו עוסקים.

סוג אחד של רשתות מקודד מפרש נקרא מקודד עצמי (autoencoder). ברשתות אלו, תפקיד המפרש הוא לבנות מהייצוג החבוי המגיע מהמקודד את הקלט המקורי, ברמת הדיוק הקרובה ביותר שניתן.

המקודד והמפרש של רשתות מסוג זה יכולים להיות מורכבים שניהם משכבות לינאריות, משכבות קונבולוציה (וקונבולוציה משולפת בהתאמה) ואף משכבות נשנות, באם הקלט לרשת הוא טבלאי, תמונות או נתונים סדרתיים בהתאם.

רשתות אלו יכולות לשמש ללמידת ביצוע פעולות של דחיסת \ שחזור מידע, ניקוי רעשים ולדגימת נתונים חדשים.

היתרון הגדול בשימוש ברשתות מסוג זה הוא ניצול היכולת ללימוד עצמי – המודל מנסה לייצר את הקלט הידוע מראש מחדש, ועל כן ניתן לכמת את מידת הצלחתו וכך לאפשר למידה מנתונים שאינם מכילים משתנה מטרה כלשהו (על ידי שימוש בקלט כמעין משתנה מטרה בפני עצמו)

סוג נוסף של רשתות מקודד מפרש הינו רשתות תרגום שפה טבעית. במקרה זה, המקודד מייצר ייצוג חבוי של המשפט בשפת המקור והמפרש מקבל ייצוג חבוי זה ומנסה לייצר ממנו את המשפט המתאים בשפת היעד.

כמובן שרשת מסוג זה תבצע שימוש נרחב בשכבות נשנות הן בפעולת הקידוד והן בפירוש, ולשם האימון עלינו לקבל בעבור כל משפט קלט גם את משתנה המטרה המתאים – המשפט בשפת היעד.

4.7.3 – שימוש נוסף: העברת לימוד

העברת לימוד הינה שיטה בה אנו לוקחים רשת או חלק ממנה אשר אומנו למטרה מסוימת, ומשתמשים בה (לרוב על ידי שינוי שכבה או הוספת ראש סיווג) לשם ביצוע משימה חדשה. הצלחת השימוש בשיטה תלויה ביכולת ההכללה של המודל בעת הלמידה שלו.

כאמור, רשתות מקודד עצמי לומדות לקודד נתוני קלט לכדי מצב חבוי בשלב המקודד ולפרש מצב חבוי זה לקבלת הקלט המקורי בשלב המפרש.

משום שפלט המודל הינו הקלט עצמו – נוכל להשתמש בכמויות גדולות של נתוני אימון בקלות יחסית, שכן לא נצטרך לתייג כל נתון למשתנה מטרה כלשהו.

בשלב זה, נצפה לקבל מקודד אשר יודע לייצר ייצוג חבוי, שאומן על סט נתונים נרחב ולכן מצופה להניב תוצאות טובות מאוד.

נוכל לקחת מקודד זה ולצרף לו ראש סיווג מתאים, לאמן רשת חדשה זו על סט נתונים מתויג חדש, כאשר תהליך הלמידה ושינוי המשקלים מתבצע רק על ראש הסיווג שצורף וכך להשיג רשת הבנויה ממחלץ תכונות אשר אומן על כמות נרחבת של נתונים (המקודד) וראש סיווג אשר יודע לקבל את הייצוג החבוי ולהמירו למשתנה מטרה מתאים לצרכינו.

בשיטה זו נוכל להרוויח אימון נרחב יותר וכך גם דיוק גבוה יותר של מודל ואף לקבל תוצאות בעלות יכולת הכללה גבוהה יותר, שכן שכבת המקודד נלמדה על סט נתונים שונה.

חלק 2: זיהוי וירוסים

פרק 5: הכרת האיום

5.1 – וירוס: הגדרה

וירוס (virus) הוא סוג של תוכנה מזיקה, נזקקה (malware), מונח כללי המתאר תוכנה אשר מוחדרת למערכת, לרוב במסווה, במטרה לפגוע בסודיות, שלמות או זמינות של מידע המשתמש, תוכנות או מערכת ההפעלה.

ישנם סוגים רבים של נזקות, כגון וירוסים, תולעים וסוסים טרויאנים, וניתן לסווג, בין היתר, על פי:

- דרך הפצה (propagation method) – כיצד הנוזקה מתפשטת בכדי להגיע ליעדה.
- פעולה (payload) – הפעולות אותן הנוזקה מבצעת כאשר היא מגיעה ליעדה.

[Stallings & Brown, 2018, p. 206]

וירוס, באופן ספציפי, הוא פיסת תוכנה אשר מסוגלת "להדביק" תוכנות אחרות.

תוכנה זו משנה את התוכנות אותן היא מדביקה כך שיקללו רוטינה ליצירת עותקים של קוד הוירוס, במטרה להדביק קבצים נוספים.

אופן הפעולה של וירוס מחשב דומה לוירוס ביולוגי - מקטע קטן של קוד גנטי – DNA או RNA, אשר מסוגל להשתלט על תא ולגרום לו לייצר עותקים זהים של הוירוס המקורי.

וירוס אשר הצליח להיקשר לתוכנה או קובץ הפעלה מסוים – מסוגל לבצע כל משימה אשר מצויה בהרשאות הקובץ והוא יבצע את פעולתו בחשאי בעת הרצת הקובץ הנגוע.

כך, על ידי הדבקת קובץ מתאים, יכול וירוס בין היתר למחוק, להצפין או לשנות את קבצי המשתמש.

בדרך כלל, וירוס כולל שלושה חלקים:

- מנגנון הדבקה (infection mechanism) – הדרך בה הוירוס מתפשט ומתרבה.
- מטען (payload) – הפעולה אותה מבצע הוירוס, מלבד תהליך ההתפשטות.
- מנגנון הפעלה (trigger) – מאורע או תנאי אשר גורמים להפעלת המטען של הוירוס.

ועובר בין ארבעה שלבים:

- שלב רדום (dormant phase) – הוירוס אינו פעיל, אך יופעל בהתאם למאורע שנקבע מראש (שלב זה לא מצוי בכל וירוס)
- שלב התפשטות (propagation phase) – הוירוס משעתק את עצמו לתוך קבצים או תוכנות אחרות במערכת. ההעתק אינו בהכרח זהה לגרסת המקור ולעתים יראה שונה לשם הסוואה. כעת כל קובץ נגוע יכיל העתק של הוירוס אשר בתורו יכנס לשלב ההתפשטות.
- שלב הפעלה (triggering phase) – התנאי של מנגנון ההפעלה מתקיים והוירוס מופעל בכדי לבצע את מטרותיו.
- שלב ביצוע (execution phase) – מטען הוירוס מבוצע. המטען יכול להיות ללא נזק, כמו הצגת הודעה או מסוכן כמו מחיקת קבצי משתמש.

[Stallings & Brown, 2018, p. 211]

5.2 – סיווג וירוסים

ישנו מרוץ תמידי בין כותבי וירוסים ליצרני תוכנות אנטי וירוס. ככל שהאמצעים לזיהוי והגנה מפני וירוסים מתייעלים ומתפתחים – כך כותבי הוירוסים מפתחים וירוסים מסוגים חדשים, בניסיון לערום על אמצעי ההגנה.

לכן, לא קיימת מוסכמה אוניברסלית על שיטה יחידה לסיווג וירוסים. לחלופין, נציג שתי דרכים לסיווג, לפי תכונות שונות של הוירוס.

ניתן לסווג וירוס לפי סוג המטרה אותה מנסה הוירוס להדביק:

- Boot sector infector – וירוס אשר מדביק את סקטור האתחול של המערכת ומתפשט בעת עליית המערכת מדיסק נגוע.
- File infector – וירוס אשר מדביק קבצים שמוגדרים על פי מערכת ההפעלה כקבצי הרצה.
- Macro virus – וירוס אשר מדביק קבצים המכילים פקודות מאקרו או סקריפט המפורש על ידי תוכנה.
- Multipartite virus – וירוס אשר מדביק קבצים במספר דרכים שונות.

מנגד, ניתן לסווג וירוס לפי מנגנון ההסתרה שלו:

- וירוס מוצפן (encrypted virus) – וירוס אשר משתמש בהצפנה בכדי להסתיר את התוכן שלו. חלק מהוירוס מכיל מפתח הצפנה שנוצר רנדומלית ושאר הוירוס מוצפן באמצעות מפתח זה. מפתח ההצפנה שמור לצד קוד הוירוס המוצפן, וכאשר הוירוס מדביק קובץ אחר – מוגרל בעבור עותק זה מפתח הצפנה חדש, כך שלא ניתן לזהות רצף ביטים אחיד בעבור קבצים נגועים בוירוס. כאשר הוירוס עובר לשלב הביצוע, קוד הוירוס מפורש באמצעות מפתח ההצפנה השמור ואז פעולתו מבוצעת.
- וירוס מוסתר (stealth virus) – וירוס שתוכנן ספציפית להסתתר מאמצעי הגנה ולכן הוירוס כולו מוסתר ולא רק המטען שלו.
- וירוס פולימורפי (polymorphic virus) – וירוס אשר מנגנון ההתפשטות שלו יוצר עותקים זהים בפונקציונליות שלהם, אך שונים במראה בכדי להתחמק מזיהוי על ידי תוכנות הגנה. תכונה זו יכולה להיות מושגת על ידי הוספת פקודות רנדומליות בין שורות קוד הוירוס, החלפת סדר של שורות בלתי תלויות מקוד הוירוס או שימוש בהצפנה.
- וירוס מטמורפי (metamorphic virus) – וירוס אשר משנה את מהותו בכל הדבקה. וירוס מסוג זה שונה מוירוס פולימורפי בכך שהוירוס משכתב את עצמו בכל הדבקה, באמצעות שימוש במספר שיטות טרנספורמציה, ובכך מגדיל את הקושי בזיהוי ואף עלול לגרום לשינוי בפונקציונליות בנוסף למראה הוירוס.

[Stallings & Brown, 2018, pp. 215-216]

5.3 – ניתוח נוזקות, malware analysis

מקטע זה ואילך נעבור לדון בנוזקות באופן כללי, שכן השיטות לזיהוי וירוסים אותן נציג משמשות גם לזיהוי נוזקות מסוגים שונים.

ניתוח נוזקות הינו מונח המתייחס לתהליך חקירת נוזקה לשם הבנת אופן פעולתה, קביעת הפונקציונליות שלה, מקורה וההשפעה הפוטנציאלית שלה.

ישנן שתי גישות לביצוע ניתוח נוזקות – ניתוח סטטי וניתוח דינאמי.

5.3.1 – ניתוח סטטי, static analysis

ניתוח סטטי מתייחס לבחינת קוד או מבנה קובץ הרצה ללא הרצתו בפועל.

גישה זו מאפשרת לבחון האם קובץ נחשב כמסוכן, לספק מידע בדבר הפונקציונליות שלו ואף לייצר אוסף חתימות ייחודי, לרוב באמצעות שימוש בפונקציית גיבוב (hash function), לשם זיהוי הקובץ.

[Gilbert, Mateu, & Planes, 2020, p. 4]

קיימות מספר שיטות לניתוח סטטי, כאשר הנפוצות ביניהן הינן:

- חיפוש רצפים של תווים בקוד – רצפים בקוד הבינארי של תוכנה יכולים לכלול אזכורים של מיקומי קבצים ששוננו או נקראו על ידי תוכנה, קישורים (URLs) שמשמשים את התוכנה, פקודות מזיקות, קריאות לספריות של מערכת ההפעלה ועוד, אשר ביכולתם לרמוז בדבר מטרת התוכנה.
- הסתכלות על רשימת הספריות המיובאות והפונקציות של קובץ הרצה, בצירוף המטא דאטה על הקובץ המצוי בכותרות (headers) – פרטים אלו כוללים מידע על פעולות ופונקציות אשר מצויות בשימוש נפוץ בתכנות ויכולים לספק רעיון כללי לגבי פעולת הקובץ.

- חיפוש קטעים מוצפנים או דחוסים – כאמור, כותבי נוזקות משתמשים לעיתים בהצפנה או דחיסה של קוד בכדי להקשות על מציאת הנוזקה. בדרך כלל, קטעי קוד אשר עברו דחיסה או הצפנה יכללו מספר מועט יותר של רצפי תווים ואנטרופיה גבוהה יותר בהשוואה לקטעי קוד לגיטימיים.
- תרגום קוד המכונה לשפת אסמבלי – תהליך הנדסה לאחור (reverse engineering) הכולל טעינת קובץ ההרצה לתוכנת disassembler בכדי לגלות מה קובץ ההרצה מבצע.

5.3.2 – ניתוח דינאמי, dynamic analysis

ניתוח דינאמי כולל הרצת תוכנה ומעקב אחרי השפעתה על המערכת. בדרך כלל תהליך זה מבוצע לאחר ששימוש בניתוח סטטי לא הניב תוצאה מספקת וזאת משום ששיטות בגישה זו לרוב דורשות יותר משאבים וזמן מאשר שיטות ניתוח סטטי.

בניגוד לניתוח סטטי, ניתוח דינאמי עוקב אחרי הפעולות המבוצעות על ידי התוכנה במדויק.

יחד עם זאת, יש לבצע את תהליך הניתוח בסביבה מוגנת בכדי להימנע מלחשוף את המערכת לסיכונים מיותרים. כסביבה מוגנת נוכל להשתמש במערכת פיזית שאינה מחוברת לרשת או למחשבים אחרים, בכדי לחסום את יכולת ההדבקה של נוזקה פוטנציאלית, או שנבחר להשתמש במכונה וירטואלית, אשר תדמה הרצת מערכת הפעלה ותאפשר את הפונקציונליות של מערכת פיזית תוך כדי שמירה על בידוד ממערכת ההפעלה של המכונה הפיזית עליה פועלת הווירטואליזציה.

הפעולות אשר נוכל לבצע בניתוח דינאמי כוללות מעקב אחרי קריאות API ומערכת הקבצים, ניתוח התעבורה ברשת וחקר מבנה הזיכרון בעת הרצת התוכנה. פעולות אלו יתאפשרו לנו באמצעות שימוש בכלי ניטור שונים. כלי חשוב במיוחד הוא שימוש בתוכנות debugger, אשר מאפשר לעקוב אחרי פעולת התוכנה תוך כדי הרצתה, שלב אחר שלב באופן דינאמי.

למרות היכולת לקבל מידע רב יותר על תוכנה מסוימת מאשר בניתוח סטטי, השימוש בניתוח דינאמי אינו נטול חסרונות.

כפי שצינו, השיטה דורשת משאבים רבים יותר בין אם מדובר במערכת פיזית נוספת לביצוע התהליך או בהרצת מכונה וירטואלית על מערכת קיימת.

בנוסף – קיימת סכנה שביצוע תהליך הניתוח בסביבה מוגנת תגרום לשינוי בהתנהגות התוכנה ואף לפגיעה פוטנציאלית במערכת:

- בשימוש במערכת פיזית המנותקת מהרשת – ייתכן שתוכנה נגועה לא תפעל כהלכה שכן מספר רב של נוזקות דורשות חיבור לרשת לשם עדכון, שליטה ובקרה. במקרים מסוימים ייתכן ונסווג קובץ כלגיטימי למרות היותו נגוע ובעל יכולת להזיק למערכת בעת פעילותו התקינה.
- בשימוש במכונה וירטואלית – ישנן נוזקות אשר מסוגלות להבחין בכך שהן רצות על מכונה וירטואלית ובמצב כזה הן יציגו מהלך ריצה שונה מאשר זה שיתרחש על מכונה פיזית, דבר שיגרום לסיווג פוטנציאלי לא נכון של תוכנה נתונה.

ראוי לציין כי גם בעת שימוש בכל אמצעי הבטיחות המוכרים – ניתוח דינאמי של נוזקות כרוך בסיכון. לעיתים מתגלות חולשות במערכות הווירטואליזציה אשר מאפשרות לתוכנה זדונית לחדור למערכת הראשית (למשל שימוש בתיקיות משותפות).

[Gilbert, Mateu, & Planes, 2020, pp. 4-5]

פרק 6: שיטות אמפיריות לזיהוי נזקות

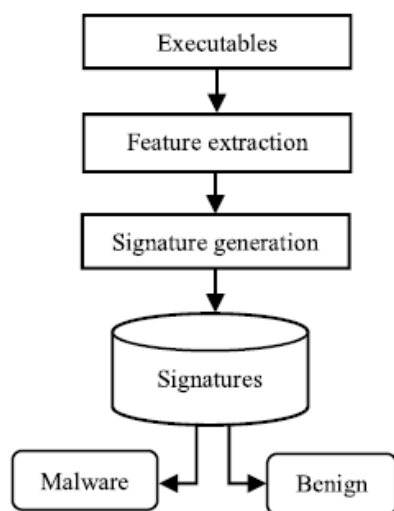
לפני תחילת השימוש בטכניקות למידת מכונה לזיהוי נזקות, השתמשו במספר טכניקות שונות בכדי להתמודד עם האיום. בתחילה הנחת היסוד הייתה כי מבנה נזקה הוא אחיד ואינו משתנה אך במרוצת השנים השתפרו הנוזקות והשתכללו באמצעים שונים להסוואה ושינוי הקוד בכדי להתחמק מזיהוי.

בפרק זה נציג בקצרה מספר שיטות אשר היו בשימוש לזיהוי נזקות, לפני שהחל השימוש הנרחב בלמידת מכונה.

6.1 – זיהוי מבוסס חתימה

כפי שהצגנו בסעיף 5.3.1 בקצרה, חתימה היא מאפיין אשר מתמצת את מבנה התוכנה ומשמש כמזהה ייחודי לתוכנה זו.

זיהוי על בסיס חתימה היא שיטה פשוטה, מהירה ואפקטיבית לזיהוי נזקות ידועות, אך אינה מספקת בעבור זיהוי נזקות שאינן ידועות עוד (כדוגמת zero day attacks). בנוסף, נזקות מאותה משפחה של נזקה ידועה יכולות להתחמק מזיהוי על ידי שימוש בשיטות להסתרה כגון הצפנה או דחיסה.



תרשים 21: סכמת מודל לזיהוי נזקות מבוסס חתימה, [Aslan & Samet, 2020, p. 7]

יצירת חתימה בעבור קובץ כוללת חילוץ מאפיינים מהקובץ ואז העברת מאפיינים אלו במנוע יצירת חתימה (למשל פונקציית גיבוב).

השימוש בשיטה זו מבוסס על מאגר חתימות פעיל. המאגר מכיל חתימות רבות של קבצים וסיווגם כקבצים תקינים או נזקות.

בעת ביצוע ניתוח של קובץ – מייצרים חתימה ממאפייני הקובץ באמצעות אותו מנוע יצירת חתימה ומשווים את החתימה המתקבלת לרשימת החתימות המצויה במאגר. על פי תוצאת ההשוואה הקובץ מוגדר כתקין או כנגוע.

שיטה זו לבדה הינה ריאקטיבית בלבד, שכן היא מסתמכת על מאגר חתימות מעודכן של נזקות לשם תהליך הזיהוי ולא תוכל להגן מפני נזקות חדשות או שלא עודכנו עוד במאגר עד לאחר ביצוע הנזק.

6.2 – זיהוי מבוסס היוריסטיקה

בשיטה זו משתמשים בחוקים היוריסטיים בכדי לחפש קבצים החשודים בסבירות גבוהה להיות נזקות.

ניתן למשל לחפש בקבצים חלקי קוד אשר נפוצים בנזקות ידועות (כגון קוד הצפנה המצוי בוירוסים פולימורפיים) – זיהוי של קטע קוד כזה יעלה את החשד שמדובר בנזקה ובמידה וציון הקובץ הניתן על ידי החוקים היוריסטיים עולה על רף מסוים – הקובץ יאובחן כמסוכן.

פרק 7: שיטות למידת מכונה קלאסית לזיהוי נזקות

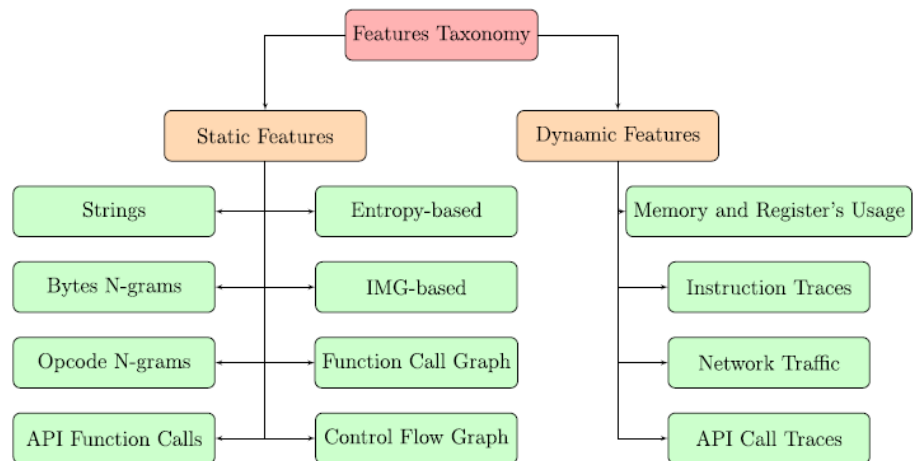
נזקות חדשות מופיעות כל הזמן ואלו הקיימות משתכללות ולעיתים תכופות יכולות לשנות את תבניתן. לכן, יכולת לימוד המאפיינים והסקת המסקנות של מודלים מבוססים על למידת מכונה, בצירוף למאגרים המתעדים נזקות מתוגות אשר הולכים וגדלים הן בכמות והן באיכות הנתונים – הופכות את למידת המכונה לדרך אטרקטיבית להתמודדות עם המשימה של זיהוי וסיווג נזקות.

כאשר ברצוננו לרתום את תחום למידת המכונה הקלאסית לשם התמודדות עם איום הנזקות, עלינו לנקות ולהתאים את הנתונים אשר בידינו בכדי שיוכלו לעבור שימוש במודלים השונים.

במקום להתמודד עם תוכנה בצורתה המקורית, נבצע תהליך עיבוד מקדים של חילוץ מאפיינים (feature extraction) מתוך התוכנה, אשר יספק מאפיינים שיתארו את התוכנה בצורה מופשטת. במאפיינים אלו נשתמש לשם אימון מודל למידת מכונה מתאים לצרכינו.

נזכיר כי בתהליך אימון המודל ברשותנו סט של קבצי הרצה של תוכנות בצירוף הגדרת ערך התוצאה שלהן (נוזקה/קובץ חוקי בעבור משימת זיהוי או הסיווג התואם לסוג הנוזקה/קובץ חוקי בעבור משימת סיווג)

את מודל הסיווג נבחר בהתאם למאפיינים, אשר מחולצים לפי שיטת ניתוח הנוזקות הנבחרת.



תרשים 22: גרף שיטות חילוץ מאפיינים, בהתאם לתהליך ניתוח הנוזקות, [Gilbert, Mateu, & Planes, 2020, p. 6]

7.1 – מאפיינים סטטיים

מאפיינים סטטיים מחולצים מתוכנה בעת תהליך ניתוח סטטי, ללא הרצת קובץ הריצה, כפי שהצגנו בסעיף 5.3.1

7.1.1 – ניתוח רצפים, string analysis

בשיטה זו מחלצים רצפים בקוד הבינארי של התוכנה, שכן אלו יכולים לכלול אזכורים של מיקומי קבצים ששונים או נקראו על ידי התוכנה, קישורים (URLs) שמשמשים את התוכנה, פקודות מזיקות, קריאות לספריות של מערכת ההפעלה ועוד, אשר ביכולתם לספק מידע בדבר מטרת התוכנה.

7.1.2 – n-gram מבוסס בתים ופקודות בקוד אסמבלי, byte/opcode n-grams

שיטה נפוצה זו לחילוץ מאפייני נזקות משתמשת בn-grams, סדרה רציפה של n פריטים הנבנית מתוך רצף נתון של טקסט.

ניתן לחלץ n-grams של רצף בתים על ידי קריאת הקוד הבינארי של התוכנה כרצף של בתים והוספת כל סדרה של n בתים ברצף זה כמאפיין חדש.

בנוסף, ניתן לחלץ n-grams של פקודות אסמבלי על ידי המרת הקוד הבינארי של התוכנה לשפת אסמבלי, בניית רצף הכולל רק את שמות הפקודות המופיעות בקוד האסמבלי (כגון "ADD", "MUL" וכו') והוספת כל קומבינציה ייחודית של n פקודות ברצף זה כמאפיין חדש.

נציין כי לצד ההצלחה של השימוש במאפיינים אלו למשימת זיהוי נזקות, שימוש ב-n-grams כרוך גם בבעיות כגון דרישות חישוביות לא סבירות לשם מיצוי כל ה-n-grams האפשריים, אשר גורר צורך בשימוש בבחירת חלק מהמאפיינים ושיטות רדוקציה נוספות, לצד בעיה של התאמת יתר אפשרית לנתוני האימון בשל הדרך בה נבחרים המאפיינים בהם נשתמש.

[Gilbert, Mateu, & Planes, 2020, p. 8]

7.1.3 – קריאות לפונקציות ממשק, API function calls

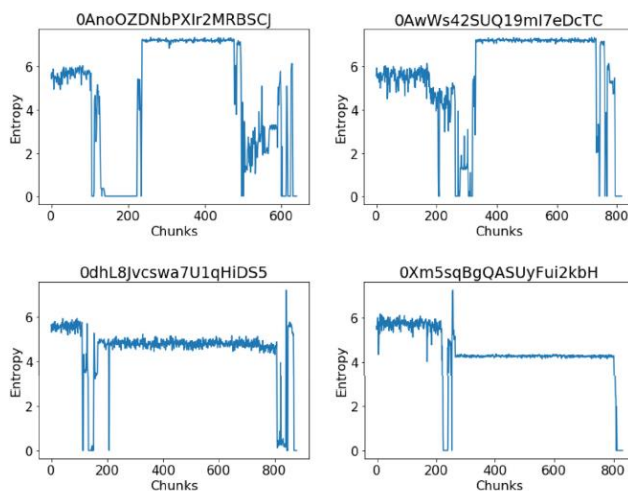
פונקציות ממשק וקריאות מערכת משמשות להבנת התנהגות תוכנות, שכן הן משמשות לתקשורת עם שירותים המסופקים על ידי מערכת ההפעלה (תקשורת, אבטחה, מערכת הקבצים ועוד).

לכן, אפשר לבנות מאפיינים לפי תדירות השימוש בקריאות השונות בתוכנה אשר בתורם יספקו מידע חשוב על פעילות התוכנה.

7.1.4 – ניתוח אנטרופיה

אנטרופיה של בתים מהווה מדד למידת השונות הסטטיסטית שלהם.

כאמור, כותבי נזקות לרוב משתמשים בשיטות כמו הצפנה ודחיסה בכדי להסוות את מטרת הקוד שכתבו ולהתחמק מזיהוי בתהליך ניתוח סטטי.



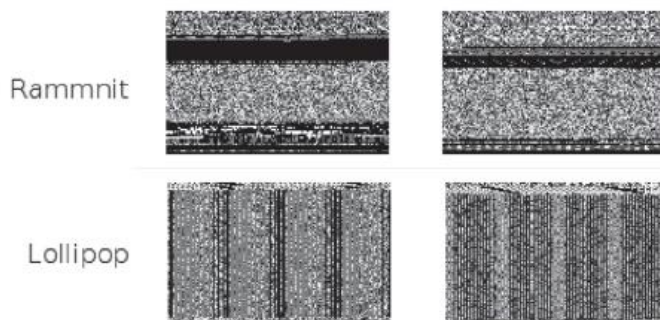
השימוש במדד האנטרופיה מאפשר לנו לנסות להתגבר על מכשול זה, שכן קטעי קוד אשר עברו דחיסה או הצפנה נוטים להיות בעלי ערכי אנטרופיה גבוהים יותר מאשר קטעי קוד רגילים.

נזקות מתקדמות מנסות לעיתים להסתיר את השימוש בהצפנה או דחיסה כך שייראה כרצף קוד תקין רגיל, אך גם במצב זה ישנה נטייה לקבצים מסוג שונה (מכילים הצפנה, מכילים דחיסה או רגילים) לכלול רמות אנטרופיה שונות המתאימות לסוג זה.

בשיטה אחת מנתחים את האנטרופיה המבנית של קובץ – הצגת רצף הבתים בקוד הבינארי של הקובץ כרצף של ערכי אנטרופיה, כאשר כל ערך מייצג את גודל האנטרופיה ביחס לפיסת קוד קטנה במיקום ספציפי.

7.1.5 – ייצוג קוד כתמונה

לפי שיטה זו, נציג את הקוד הבינארי של קובץ ההרצה כתמונה בגווי אפור (grey scale) על ידי פירוש כל בית כערך של פיקסל יחיד בתמונה, בטווח 0-255 וסידור המערך הנוצר כמערך דו ממדי.



כפי שניתן לראות בתרשים 24, הייצוג כתמונה של נזקה ממשפחה מסוימת דומה לייצוג של נזקות ממשפחה זו ושונה מהייצוג של נזקות ממשפחה אחרת. הדמיון הוויזואלי נובע משימוש חוזר בקטעי קוד ליצירת הקובץ הבינארי, כך שבאמצעות ייצוג קבצי הרצה כתמונה נוכל לנסות לזהות שינויים קלים בין דגימות מאותה המשפחה.

תרשים 24: דוגמה לייצוג כתמונה של קוד בינארי השייך לקבצי נזקה ממשפחות שונות, [Gilbert, Mateu, & Planes, 2020, p. 9]

המאפיינים אותם ניתן לחלץ בשיטה זו כוללים עוצמות ממוצעות של פיקסלים, שוני, מספר פיקסלים בעלי ערך עוצמה מסוים ועוד.

נציין כי לייצוג של תוכנה כתמונה ישנם גם חסרונות:

קטעי הקוד אינם במקור תמונות דו ממדיות ועל ידי הפיכתם לכאלו נוכל לייצר תנאים מקדימים שאינם מופיעים בקוד המקורי. בין היתר, נצטרך לקבוע ידנית את ממדי התמונה שברצוננו להרכיב (ערך גובה ורוחב) ותוצאת התמונה תשתנה בהתאם.

בנוסף, בתמונות ישנה קורלציה בין פיקסלים סמוכים, תכונה שאינה בהכרח רלוונטית לקטעי קוד של תוכנה. ולבסוף, כמו מירב המאפיינים הסטטיים – השיטה אינה מתמודדת בצורה טובה עם נזקות המכילות שיטות הסתרה מזיהוי כגון הצפנה ודחיסה, אשר עלולות לשנות לחלוטין את מבנה הבתים של קובץ ההרצה וכך גם את מבנה ואופי התמונה המתאימה לקובץ.

7.1.6 – גרף קריאות פונקציה, *function call graph*

גרף קריאות פונקציה הינו גרף מכוון המורכב מקודקודים המתארים את הפונקציות אותן מכילה התוכנה ומצלעות המתארות את אופן הקריאה לפונקציות.

הפונקציות המתוארות מחולקות לשני סוגים:

1. פונקציות מקומיות, אשר נכתבו על ידי המתכנת לשם ביצוע פעולה מסוימת
2. פונקציות חיצוניות – פונקציות מערכת וספריות חיצוניות.

הגרף נבנה תוך כדי שמירה על תנאי לפיו פונקציות מקומיות יכולות לקרוא לפונקציות חיצוניות אך לא להיפך.

השיטה בונה גרף קריאות פונקציה מתוך פירוש הקוד הבינארי של קובץ הרצה לקוד אסמבלי.

ניתן להתייחס לגרף כמאפיין ולהשתמש בו להשוואה אל מול גרפים תואמים של קבצים אחרים.

לחלופין, ניתן לייצר וקטור מייצג מתוך הגרף ולהשתמש בו כווקטור מאפיינים.

7.1.7 – גרף בקרת זרימה

גרף בקרת זרימה הוא גרף מכוון כאשר קודקודיו הם בלוקים של קוד וצלעותיו הם מסלולי בקרת הזרימה.

בלוק קוד הוא רצף לינארי של פקודות בתוכנה המכיל נקודת מוצא (הפקודה הראשונה בבלוק) ונקודת סוף (הפקודה האחרונה בבלוק).

באופן זה, גרף בקרת זרימה מייצג את כל המסלולים אשר יכולים להתבצע במהלך הרצת התוכנה.

השיטה בונה גרף בקרת זרימה מתוך פירוש הקוד הבינארי של קובץ הרצה לקוד אסמבלי, בו נוכל להשתמש כמאפיין להשוואה מול גרפים תואמים בקבצים אחרים.

7.2 – מאפיינים דינאמיים

מאפיינים דינאמיים מחולצים מתוכנה במהלך ניתוח דינאמי אשר כולל הרצה של קובץ הריצה, כפי שהצגנו בסעיף 5.3.2

7.2.1 – זיכרון ושימוש באוגרים

נוכל לתאר את התנהגותה של תוכנה לפי הערכים הצבורים בזיכרון במהלך ריצתה.

שיטה זו מתעדת את ערכי אוגרי הזיכרון בעת הרצת התוכנה כמאפיינים בהם ניתן להשתמש לשם הבדלה בין קובץ נגוע לקובץ נקי.

נוכל למשל לשמור את ערכי האוגרים בעת ביצוע קריאות API במהלך ריצת תוכנה ולהשוות ערכים אלו למקבילים שלהם בעבור תוכנות אחרות.

7.2.2 – מעקב אחרי פקודות, *instruction trace*

מעקב דינאמי אחרי פקודות מייצר סדרה של פקודות מעבד אשר נקראו בעת מהלך הריצה של התוכנה.

בניגוד למעקב סטטי אחרי פקודות, שעוקב אחרי הפקודות לפי סדר הופעתן בקובץ ההרצה הבינארי – מעקב דינאמי מאפשר להציג את הסדר לפיו הפקודות מופיעות בפועל בזמן הריצה ולכן הינו אמצעי מדויק יותר לתיאור התוכנה, במיוחד במקרים בהם מדובר בנוזקה מתקדמת אשר עושה שימוש בשיטות הסתרה.

לאחר יצירת רצף הפקודות, נוכל לייצר n-grams מתוך הרצף ולייצר ממנו מאפיינים למשימתנו, או לחלופין לייצר גרף קריאות פונקציה מתאים.

7.2.3 – ניטור תעבורת תקשורת

תוכנה זדונית עלולה במהלך הריצה שלה ליצור קשר עם שרת חיצוני לשם קבלת פקודות מרחוק, הורדת עדכונים, הורדת קבצי נזקות נוספים או הדלפת מידע רגיש מהמחשב הנגוע.

לכן, ניטור של תעבורת הרשת במהלך ריצתה של תוכנה מספק מידע שימושי לזיהוי התנהגויות זדוניות.

ניתן לחלץ מאפיינים ברמות אבסטרקציה שונות, החל מתוכן פקטות רשת בודדות, זרימה ברשת, מידע המועבר בפרוטוקולי רשת ספציפיים ועד מידע יותר כללי כגון כתובות IP, פורטים בשימוש וספירת פקטות.

7.2.4 – מעקב אחרי פונקציות ממשק, API call trace

בדומה לסעיף 7.2.2 – מעקב אחרי קריאות אלו יספק מידע מדויק יותר בסדר המבוצע בפועל על ידי התוכנה, לעומת השיטה הסטטית המקבילה.

7.3 – שימוש במודלים של למידת מכונה קלאסית (מאפיינים למודל)

בשני הסעיפים הקודמים הצגנו מאפיינים רלוונטיים לזיהוי וסיווג נזקות וכיצד ניתן לחלץ אותם מתוכנה נתונה.

לאחר בחירת המאפיינים אותם נחלץ מתוך התוכנה, נרצה להשתמש במאפיינים אלו כסט הנתונים באחת מהשיטות ללמידת מכונה, שאת חלקן הצגנו בחלק א' של עבודה זו, לשם יצירת מודל לזיהוי או סיווג נזקות.

אך ברוב המקרים נצטרך לבצע עיבוד מקדים נוסף בכדי להכין את המאפיינים שחילצנו לסוג הקלט האופייני לשיטת למידת המכונה שנבחר. לדוגמה:

- בעבור חילוף מאפייני n-gram, כפי שתוארו בסעיף 7.1.2, אשר לרוב יניב מספר רב מאוד של מאפיינים – נצטרך להשתמש בחישוב לאמדתן החשיבות היחסית של כל מאפיין על הסיווג הסופי (כדוגמת information gain), לבחור את המאפיינים המשמעותיים ביותר ולהשתמש במאפיינים הנבחרים בלבד כקלט למודל כדוגמת עץ החלטה, יער אקראי, רשת נוירונים פשוטה או למידה בייסיאנית פשוטה.
- בעבור חילוף מאפיינים על ידי ייצוג תמונה, כפי שתואר בסעיף 7.1.5, נוכל להשתמש בטכניקות של ניתוח תמונה (לדוגמת gist) בכדי לחלץ מאפיינים מצומצמים בעבור התמונה שהושגה ולהשתמש במאפיינים אלו כקלט למודל כמו k nearest neighbors לשם מטרת האימון ופיתוח יכולת הסיווג של קבצים חדשים.

פרק 8: שיטות למידה עמוקה לזיהוי נזקות

8.1 – למידה עמוקה לעומת למידת מכונה קלאסית

שיטות מבוססות למידת מכונה קלאסית מסתמכות ברובן על מאפיינים שהוגדרו ותוכננו על ידי מומחים בתחום הרלוונטי למשימה. מטרת מאפיינים אלו היא לספק מידע יעיל ככל הניתן בעבור קלט נתון.

במקרה הספציפי של סיווג נזקות, המאפיינים (אשר הוצגו בפרקים 7.1 ו-7.2) מספקים מבט כללי לתוכנה החשודה כנוזקה, באמצעות ערכים אותם ניתן להעביר למסווג מבוסס למידת מכונה לשם בניית מודל.

התהליך של בניית מאפיינים רלוונטיים וייצוגם הינו הכרחי לבניית מודלים מבוססי למידת מכונה קלאסית, בעל השפעה רבה על איכות התוצאה ודורש זמן רב.

לעומת זאת, למידה עמוקה מאפשרת לנו לבנות מודלים ניתנים לאימון, אשר כוללים שכבות פנימיות של חילוף מאפיינים ללא התערבות אדם. בגישה זו, נוכל להעביר למודל מתאים את הקלט כפי שהוא, או לאחר ביצוע תהליך עיבוד מקדים קל (כזה שאינו כולל יצירת מאפיינים ידנית) וזה בתורו ילמד לחלץ מאפיינים רלוונטיים מהקלט ולהשלים את משימת הסיווג בהתאם למאפיינים אלו.

בפרק זה נציג מספר שיטות לשימוש בלמידה עמוקה למטרת זיהוי וסיווג נזקות, בהתאם לתהליך העיבוד המקדים המתאים לכל שיטה.

8.2 – ייצוג כווקטור מאפיינים, Feature vector representation

שיטה זו לביצוע עיבוד מקדים דומה לעיבוד המקדים המבוצע בעבור שיטות למידת מכונה קלאסיות.

מבוצעות פעולות לחילוף מאפיינים מתוך קובץ נתון, לפי השיטות אשר הוצגו בסעיפים 7.1 ו-7.2, כאשר ניתן להשתמש ביותר משיטה אחת. מאפיינים אלו יהוו כמעין ייצוג חבוי לקובץ הנתון.

לאחר מכן, המאפיינים שחולצו מורכבים לווקטור מאפיינים אשר ישמש כקלט למודל הלמידה העמוקה שנבחר, אשר יהיה לרוב רשת נוירונים לינארית עמוקה (deep feed forward neural network).

8.3 – ייצוג כתמונה, Image based representation

שיטה זו, אשר מפרשת את קוד התוכנה כתמונה כפי שנעשה בסעיף 7.1.5 מבצעת פעולה שונה עם המידע המחולץ.

במקום להשתמש בשיטות ידניות לחילוף מאפיינים מהתמונה המתקבלת – התמונה מועברת ישירות כקלט לרשת קונבולוציה (CNN) שבתורה לומדת גם לבצע בחירת מאפיינים מהתמונה וגם סיווג.

לרוב, רשת הקונבולוציה בה נשתמש תהיה מורכבת מבלוקים לחילוף מאפיינים ולבסוף ראש סיווג.

כאשר בלוק לחילוף מאפיינים יכלול שכבת קונבולוציה, פונקציית אקטיבציה, שכבת איגום ושכבת נרמול.

וראש הסיווג יכיל שכבה לינארית כשכבת הפלט.

8.4 – מעקב אחרי פונקציות ממשק \ קריאות לפונקציות, API call trace / Instruction trace

בפרק הקודם תיארו שיטות לחילוף מאפיינים מתוך קריאות לפונקציות ממשק ופונקציות בקוד אשר תוכנה מבצעת ושימוש במאפיינים אלו לשם אימון מודל לזיהוי נזקות. אך השימוש הניתן לביצוע באמצעות שיטות למידת מכונה קלאסיות אינו מייחס חשיבות לסדר הופעתן של הקריאות במהלך התוכנה.

כעת, כאשר אנו נעזרים בשיטות ללמידה עמוקה, נוכל לחלץ רשימה של הקריאות המבוצעות לפונקציות לפי סדר הופעתן ולאמן על פי רשימה זו רשת נוירונים נשנית (RNN) אשר תוכל ללמוד את מלאכת הסיווג גם לפי הפקודות המופיעות וגם לפי סדר הופעתן.

לרוב, הרשת הנשנית בה נשתמש תכלול שכבת embedding, אשר ממירה כל קלט לייצוג הניתן לשימוש על ידי רשת נוירונים, שכבה נשנית אחת או יותר (כדוגמת RNN layer או LSTM layer) ולבסוף שכבה לינארית כראש סיווג.

8.5 – ייצוג מבוסס בתים

הדרך הפשוטה ביותר לייצג תוכנת מחשב היא על ידי רצף הבתים המופיעים בקוד הבינארי של קובץ ההרצה. בייצוג זה כל בית מהווה יחידה ברצף הקלט.

דרך זו מאפשרת לייצג נזקות בצורה שאינה תלויה במערכת ההפעלה או בחומרה ספציפית.

אך ייצוג קובץ הרצה כרצף של בתים כרוך במספר בעיות:

- ייצוג זה יכול לכלול רצף עצום של בתים, כאשר כל בית משמש כקלט, לקבלת רצף קלטים המכיל מיליוני צעדים. זאת בזמן שהשימוש במודלים לעיבוד רצפים נעשה משמעותית מסובך יותר ככל שגודל הקלט גדל.
- כל בית בקלט יכול להתפרש באופנים שונים כתלות בהקשר בו הוא מופיע. למשל, בית יכול להיות חלק מקוד בינארי, טקסט בשפה טבעית, תמונה ועוד.
- קבצים בינאריים מאופיינים ברמות שונות של קישוריות מרחבית (spatial correlation). פקודות מכונה סמוכות זו לזו נוטות להיות קשורות זו לזו, אך לעיתים לא כך הדבר – בשל קפיצות בקוד וקריאה לפונקציות.

בשל בעיות אלו, מודל מתאים לשימוש בקלט זה חייב להיות מסוגל להתאים את עצמו לגודל רצף הקלט וגם להתאים להקשר המקומי והכללי של כל הקובץ בו זמנית.

השימוש בשיטות למידה עמוקה מאפשר לנו לנסות ולהתמודד עם המשימה.

ניתן להשתמש ברשת קונבולוציה בכדי לבצע זאת, על ידי בניית מודל המורכב משכבת embedding הממירה כל בית לזוקטור קלט, שכבות קונבולוציה לחילוף מאפיינים ולבסוף שכבה לינארית לסיווג המאפיינים שחולצו.

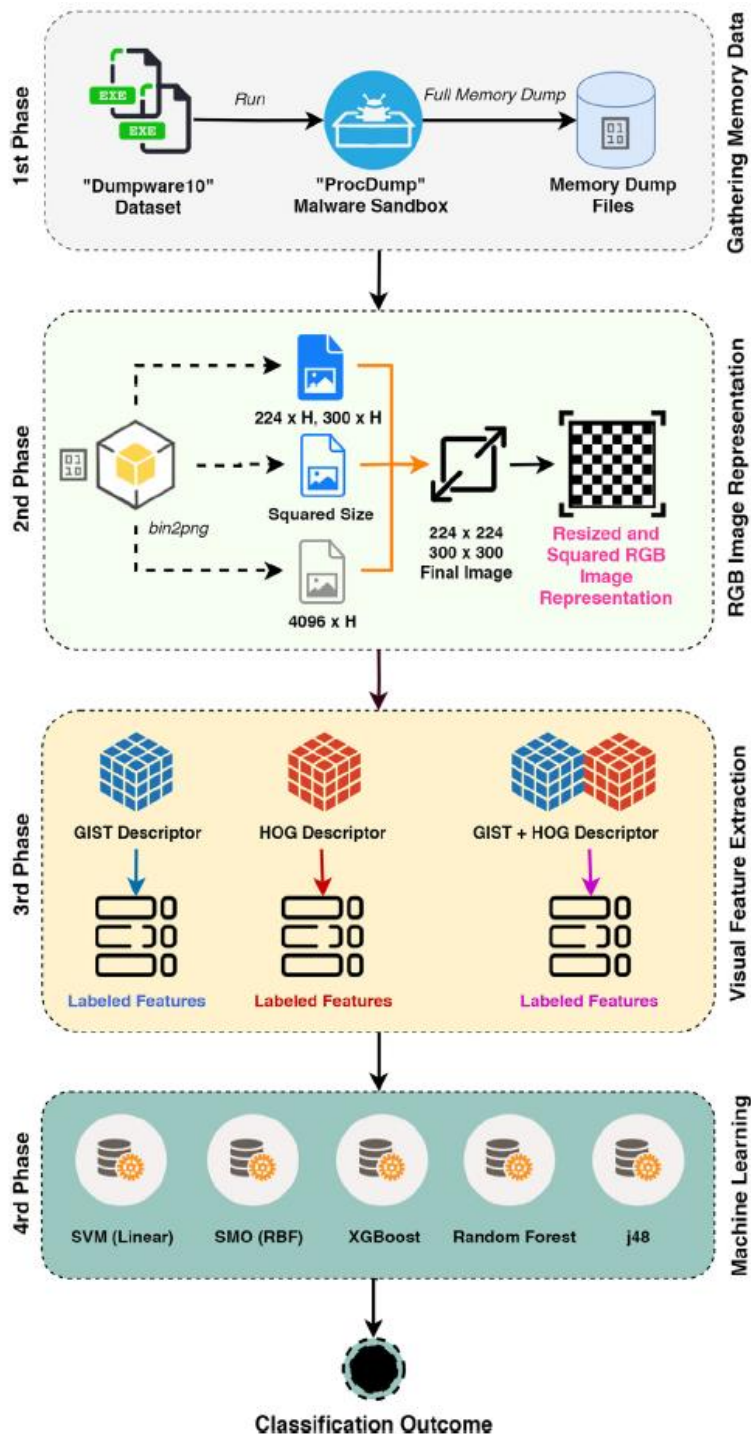
נציג שיטה נוספת לביצוע משימת הסיווג:

נאמן רשת מקודד עצמי להפחתת רעשים (denoising autoencoder) למשימת עיבוד נתחים (chunks) של קוד בינארי, כך ששכבות המקודד ירכיבו ייצוג חבוי לנתח הקוד ושכבות המפרש ינסו לשחזר את תוכן הקוד המקורי מתוך הייצוג החבוי.

לאחר מכן, נחלק את קוד התוכניות לנתחים בגודל קבוע, נעביר כל נתח בשכבת המקודד של המקודד העצמי שבנינו בשלב הקודם, לקבלת רצפים של ייצוגים חבויים המייצגים את הקוד כולו בעבור כל תוכנה. לבסוף, נאמן רשת נשנית (RNN) על רצפים אלו ושכבה לינארית סופית שתשמש כראש סיווג.

פרק 9: מבט לעומק – שיטת למידת מכונה מבוססת תמונת זיכרון לסיווג נוזקות

בפרק זה נציג שיטה לסיווג נוזקות מבוססת למידת מכונה אשר משתמשת במאפיינים שחולצו בתהליך ניתוח זיכרון נדיף בעת הרצת תוכנה נבדקת, כפי שתוארה במאמר [Bozkir, Tahilioglu, Aydos, & Kara, 2021].



תרשים 25: מבנה השיטה לסיווג נוזקות על סמך תמונת זיכרון נדיף, [Bozkir, Tahilioglu, Aydos, & Kara, 2021, p. 7]

9.1 – ניתוח זיכרון נדיף, Volatile memory forensics

ניתוח זיכרון נדיף הינה גישה לתהליך ניתוח נוזקות, בדומה לניתוח סטטי וניתוח דינאמי.

הגישה מורכבת משני שלבים:

השגת תוכן הזיכרון – שלב זה מתייחס להמרת המידע הצבור בזיכרון הפיזי או הווירטואלי לקובץ memory dump על ידי שימוש בדרייברים של גרעין המערכת או אימולטורים.

ניתוח התוכן – שלב זה עוסק בהסקת מידע שמיש מתוך קובץ memory dump באמצעות שיטות כדוגמת למידת מכונה.

גישה זו מתמודדת היטב עם שיטות הסתרה של נוזקות כגון הצפנה ודחיסה, משום שתהליך הניתוח מבוצע בעת שהתוכנה מצויה בזיכרון הנדיף במהלך הרצתה, כאשר היא חשופה מטכניקות הסתרה לשם ביצוע פעילותה התקינה. בנוסף, הגישה מאפשרת התמודדות עם נוזקות ללא קבצים (fileless malware), אשר מופעלות ישירות מהזיכרון הנדיף של המערכת ולא משאירות חותם על מערכת הקבצים.

9.2 – הצגת השיטה

השיטה המוצגת במאמר מייצרת תמונה מתוך נתוני הזיכרון בעת הרצת תוכנה ומנתחת תמונה זו לשם סיווג התוכנה כנוזקה או קובץ תמים.

השיטה מורכבת מארבעה שלבים, כמתואר בתרשים 25, אותם נפרט כעת:

9.2.1 – איסוף מידע אודות זיכרון

תפקידו של השלב הראשון בשיטה הוא לאסוף מידע אודות מצב הזיכרון של המערכת בעת ריצת תוכנה. את המידע המבוקש נשיג באמצעות memory dump - תהליך חילוץ המידע הזמני מתוך הזיכרון הקשיח והזיכרון הנדיף של המחשב. בעת ביצוע תהליך זה על תוכנה ספציפית, נוכל ללמוד פרטים רבים על התוכנה, לדוגמת מקטעי הקוד (text segment) ומקטעי הנתונים (data segment), מבנה המחסנית, אזור הערימה ועוד.

קיימות מספר תוכנות בהן ניתן להיעזר לשם ביצוע תהליך memory dumping.

בניסוי המתואר ב [Bozkir, Tahilioglu, Aydos, & Kara, 2021, pp. 7-13], התוכנות הנבדקות רצות על windows 10 בסביבה וירטואלית, וקבצי memory dump מושגים באמצעות התוכנה ProcDump [ProcDump, n.d.] על ידי איסוף כלל המידע המצוי בזיכרון בעבור התהליך הנבדק בעת ריצתו.

השימוש בסביבה הווירטואלית נועד להגן על המערכת הפיזית מפני הנוזקות הנבדקות.

גודל קובץ memory dump המתקבל משתנה מתהליך לתהליך בהתאם לאזורי הזיכרון בהם הוא משתמש, משתני הסביבה וקטעי הקוד והמידע המופיעים בו.

9.2.2 – ייצוג המידע כתמונה

בשלב זה נבצע המרה של המידע שאספנו לכדי תמונה, אשר תאפשר שימוש בשיטות לניתוח תמונות לשם מטרתנו. בכדי לייצג את המידע שאספנו בקובץ memory dump בתור תמונה, עלינו לקבוע את גודל אורך התמונה, הרוחב או את היחס בין האורך לרוחב. זאת משום שקבצים בעלי גדלים שונים יניבו תמונות בעלות רזולוציות שונות בהתאם.

כעת, נוכל לקרוא את הקובץ כרצף של בתים ולפרש אותו בתור תמונה בשני אופנים:

1. תמונת גווני אפור (grayscale image) על ידי המרת כל בית לערך העוצמה של פיקסל יחיד.
2. תמונה צבעונית (RGB image) על ידי המרת כל רצף של שלושה בתים לכדי ערכי העוצמות של הצבעים אדום, ירוק וכחול (RGB) של פיקסל יחיד.

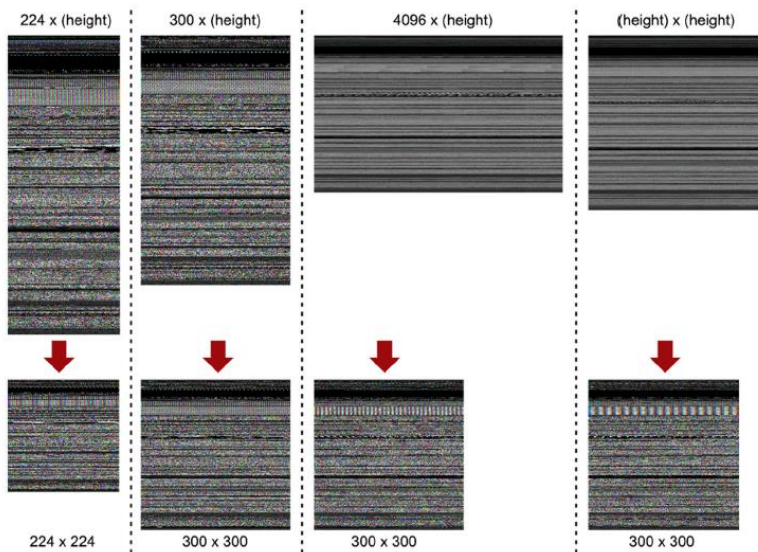
בניסוי המתואר ב [Bozkir, Tahilioglu, Aydos, & Kara, 2021, pp. 7-13], מבוצעת המרה של קובץ הנתונים שנאספו לתמונה צבעונית. הסיבה לבחירה זו היא היכולת לבטא יותר מידע בכל שורה בתמונה המתקבלת וכך לאפשר לדמיון בין דגימות שונות לבלוט. בנוסף, על ידי המרה של כל 3 בתים לכדי פיקסל יחיד מתקבלת תמונה קטנה פי 3 מאשר התמונה המקבילה לה בגווני אפור בלבד וכך צפוי להתווסף פחות רעש בעת סידור התמונה מחדש לשם התאמה למודל.

תהליך ההמרה עצמו בוצע באמצעות גרסה מעודכנת של הסקריפט bin2png [Sultanik, n.d.].

מכל קובץ memory dump, הופקו 4 תמונות בטכניקות שונות:

- תמונה בעלת אורך של 224 פיקסלים
- תמונה בעלת אורך של 300 פיקסלים
- תמונה בעלת אורך של 4096 פיקסלים
- תמונה ריבועית באורך (ורוחב) שורש מספר הפיקסלים הדרוש לייצוג (מספר הבתים בקובץ חלקי 3)

שלושת הטכניקות הראשונות מייצרות תמונות בעלות גדלי שורה זהים, ללא תלות בגודל הקובץ, בעוד שהטכניקה האחרונה תייצר תמונות בגדלים שונים אך בעלי יחס גובה לרוחב של 1:1.



תרשים 26: דוגמאות לתהליך עיבוד התמונות, [Bozkir, Tahilioglu, Aydos, & Kara, 2021, p. 9]

יצירת התמונות במספר טכניקות שונות מאפשרת להשוות את תוצאות המודל על פני מספר סוגי קלט ולבחור במאפייני התמונה המיטיבים מבין אפשרויות אלו.

לאחר יצירת התמונות, הן צומצמו לכדי תמונות ריבועיות ברזולוציה של 224 ו-300 פיקסלים באמצעות שיטה בשם Lanczos interpolation אשר מאפשרת הפחתה לוקאלית של כל אזור בתמונה.

תהליך זה אמנם גורם לאובדן מינורי של מידע אך מאפשר חילוץ ווקטור מאפיינים שווה בגודלו ללא תלות בתמונה ספציפית ומייעל את מהירות ניתוח הקלט.

מאגר התמונות שנוצרו בעבור בסיס הנתונים בניסוי מופיע ב [Dumpware10, n.d.]

9.2.3 – חילוץ מאפיינים ויזואליים

בשלב השלישי מבוצע שימוש בתמונות אשר הושגו בשלב הקודם לשם חילוץ מאפיינים בטכניקות של ראייה ממוחשבת. מאפיינים אלו יהוו כקלט לשיטות למידת המכונה לסיווג בהן נשתמש בשלב הבא.

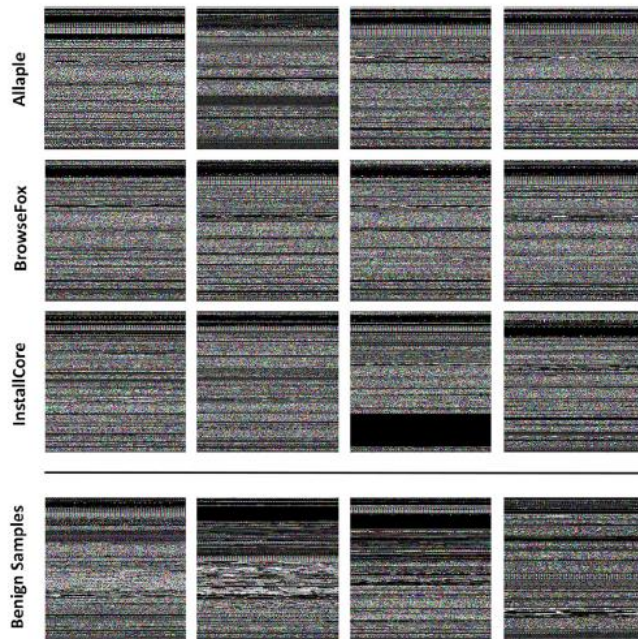
בניסוי המתואר ב [Bozkir, Tahilioglu, Aydos, & Kara, 2021, pp. 7-13], תהליך חילוץ המאפיינים מבוצע בשלושה אופנים:

1. GIST – אלגוריתם אשר מייצר ייצוג לתמונה על סמך ניתוח המבנה המרחבי הדומיננטי של התמונה תוך כדי התחשבות בחמישה ממדים תפיסתיים: פתיחות, חספוס, טבעיות, הרחבות וקשיחות.
2. Histogram of Oriented Gradients – שיטה המאפשרת ללכוד מאפיינים מקומיים בתמונה על ידי שימוש בעוצמות הגרדיאנט וכיווני הקצוות. וקטור HOG מיוצר על ידי חישוב הגרדיאנטים, איגוד קצוות ולבסוף נרמול בלוקים.
3. שילוב של מאפייני GIST ומאפייני HOG.

השימוש בשלושת אופנים אלו מאפשר ביצוע השוואה ביעילות התהליך.

9.2.4 – סיווג על ידי טכניקות למידת מכונה

השלב הרביעי והאחרון בתהליך כולל הזנת וקטור המאפיינים אשר חולץ בשלב הקודם כפלט לשיטת למידת מכונה נבחרת ראשית לשם אימון המודל ולבסוף לשם סיווג דגימות חדשות.



תרשים 27: ייצוג כתמונה של נתוני זיכרון השייכים לתוכנות ממשפחות שונות, [Bozkir, Tahilioglu, Aydos, & Kara, 2021, p. 10]

בניסוי המתואר ב [Bozkir, Tahilioglu, Aydos, & Kara, 2021, pp. 7-13] נעשה שימוש בחמש שיטות למידת מכונה שונות לשם השוואה:

1. Random Forest – יער אקראי, מבנה המורכב ממספר עצי החלטה.
2. XGBoost – מבנה של עצי החלטה בו כל עץ חדש מנסה למזער את השגיאה של קודמו ולבסוף נבחרת תוצאת הרוב.
3. Linear SVM – שיטה לרגרסיה לינארית.
4. Sequential Minimal Optimization
5. J48 – אלגוריתם עץ החלטה אשר כולל גיזום ומספר שיפורים נוספים.

שיטות 1, 2 ו-5 הן שיטות מבוססות עצי החלטה, בעוד ששיטות 3 ו-4 מבוססות מבנה. בדרך זו נבחנות מספר גישות לביצוע תהליך הסיווג ומתבצעת השוואה במידת הצלחתן.

9.2.5 – מדדי הצלחה

בעת ניתוח תוצאות של מודלי סיווג, ניתן להגדיר משתנה מטרה רצוי כחיובי ולהתייחס לדגימות המשוכות למשתנה מטרה זה כחיוביות ולאלו שאינן שייכות למשתנה זה כשליליות. כך ניתן למנות מספר תכונות:

- True Positive – בקצרה TP, מספר הדגימות החיוביות אשר סווגו כחיוביות.
- True Negative – בקצרה TN, מספר הדגימות השליליות אשר סווגו כשליליות.
- False Positive – בקצרה FP, מספר הדגימות השליליות אשר סווגו כחיוביות.
- False Negative – בקצרה FN, מספר הדגימות החיוביות אשר סווגו כשליליות.

בכדי לבחון את מידת הצלחה של כל אחד מהיישומים של השיטה המוצעת, נעשה במאמר שימוש במספר מטריקות להשוואה:

1. מדד דיוק (Accuracy) – מספר הדגימות אשר סווגו נכונה, מתוך כלל הדגימות:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

2. מדד מדויקות (Precision) – מספר הדגימות החיוביות אשר סווגו נכונה, מתוך כלל הדגימות אשר סווגו כחיוביות:

$$Precision = \frac{TP}{TP + FP}$$

3. מדד Recall – מספר הדגימות החיוביות אשר סווגו נכונה מתוך כלל הדגימות החיוביות:

$$Recall = \frac{TP}{TP + FN}$$

4. ציון F1:

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

5. יחס FP – מספר הדגימות השליליות אשר סווגו כחיוביות מתוך כלל הדגימות השליליות:

$$FPR = \frac{FP}{FP + TN}$$

9.3 – תוצאות הניסוי

הניסוי המתואר במאמר בחן את המודל המוצג בסעיף 9.2 לזיהוי וסיווג נוזקות תוך כדי בחינת ההשפעה של בחירת אופן חילוף מאפייני התמונה, ממדי התמונה המייצגת את קובץ הקס Memory dump ושיטת למידת המכונה לסיווג על יעילות המודל.

הטבלאות הבאות מציגות את התוצאות שהתקבלו בהתאם למאפיינים שחולצו:

Table 2 – GIST based performance comparison for the various classifiers and row length settings. Best configuration was highlighted with bold characters.

ML.Algorithm	Initial Column Width	Feature	Accuracy	FPR	Precision	Recall	F1-Score
Random Forest	224 pixels	GIST	86.06%	0.017	0.867	0.861	0.857
SMO (RBF)	224 pixels	GIST	87.45%	0.014	0.880	0.875	0.875
SVM (Linear)	224 pixels	GIST	85.36%	0.016	0.858	0.854	0.853
XGBoost	224 pixels	GIST	88.26%	0.011	0.885	0.882	0.881
J48	224 pixels	GIST	72.70%	0.029	0.731	0.727	0.726
Random Forest	300 pixels	GIST	89.08%	0.013	0.892	0.892	0.891
SMO (RBF)	300 pixels	GIST	91.40%	0.010	0.915	0.914	0.914
SVM (Linear)	300 pixels	GIST	88.03%	0.014	0.879	0.880	0.879
XGBoost	300 pixels	GIST	88.61%	0.011	0.888	0.886	0.884
J48	300 pixels	GIST	74.09%	0.027	0.756	0.741	0.746
Random Forest	4096 pixels	GIST	93.14%	0.09	0.931	0.931	0.932
SMO (RBF)	4096 pixels	GIST	94.65%	0.006	0.948	0.947	0.947
SVM (Linear)	4096 pixels	GIST	92.91%	0.08	0.928	0.929	0.928
XGBoost	4096 pixels	GIST	91.63%	0.008	0.919	0.916	0.917
J48	4096 pixels	GIST	79.55%	0.02	0.804	0.797	0.797
Random Forest	Square root scheme	GIST	88.03%	0.016	0.885	0.880	0.878
SMO (RBF)	Square root scheme	GIST	91.75%	0.009	0.917	0.918	0.916
SVM (Linear)	Square root scheme	GIST	88.73%	0.012	0.887	0.887	0.885
XGBoost	Square root scheme	GIST	88.96%	0.011	0.889	0.889	0.888
J48	Square root scheme	GIST	71.19%	0.033	0.724	0.712	0.716

Table 3 – HOG based performance comparison for the various classifiers and row length settings. Best configuration was highlighted with bold characters.

ML.Algorithm	Initial Column Width	Feature	Accuracy	FPR	Precision	Recall	F1-Score
Random Forest	224 pixels	HOG	86.87%	0.017	0.874	0.869	0.866
SMO (RBF)	224 pixels	HOG	90.84%	0.010	0.910	0.909	0.909
SVM (Linear)	224 pixels	HOG	85.71%	0.015	0.859	0.859	0.859
XGBoost	224 pixels	HOG	87.45%	0.012	0.875	0.874	0.872
J48	224 pixels	HOG	67.71%	0.036	0.687	0.677	0.679
Random Forest	300 pixels	HOG	86.06%	0.018	0.861	0.861	0.861
SMO (RBF)	300 pixels	HOG	89.31%	0.012	0.894	0.893	0.892
SVM (Linear)	300 pixels	HOG	85.48%	0.016	0.861	0.855	0.856
XGBoost	300 pixels	HOG	84.90%	0.015	0.848	0.849	0.846
J48	300 pixels	HOG	70.84%	0.031	0.726	0.708	0.712
Random Forest	4096 pixels	HOG	88.61%	0.015	0.892	0.886	0.884
SMO (RBF)	4096 pixels	HOG	92.68%	0.008	0.927	0.927	0.926
SVM (Linear)	4096 pixels	HOG	88.85%	0.012	0.888	0.889	0.888
XGBoost	4096 pixels	HOG	90.12%	0.010	0.904	0.901	0.900
J48	4096 pixels	HOG	70.84%	0.033	0.717	0.708	0.710
Random Forest	Square root scheme	HOG	88.05%	0.014	0.896	0.889	0.886
SMO (RBF)	Square root scheme	HOG	89.79%	0.012	0.901	0.898	0.898
SVM (Linear)	Square root scheme	HOG	85.13%	0.017	0.857	0.851	0.853
XGBoost	Square root scheme	HOG	88.15%	0.012	0.882	0.881	0.880
J48	Square root scheme	HOG	66.89%	0.037	0.683	0.669	0.674

Table 4 – GIST+HOG based performance comparison for various classifiers and row length settings. Best configuration was highlighted with bold characters.

ML.Algorithm	Initial Column Width	Feature	Accuracy	FPR	Precision	Recall	F1-Score
Random Forest	224 pixels	GIST+HOG	89.89%	0.010	0.899	0.899	0.897
SMO (RBF)	224 pixels	GIST+HOG	94.54%	0.006	0.946	0.945	0.945
SVM (Linear)	224 pixels	GIST+HOG	92.10%	0.009	0.923	0.921	0.921
XGBoost	224 pixels	GIST+HOG	91.86%	0.008	0.922	0.918	0.918
J48	224 pixels	GIST+HOG	73.28%	0.028	0.741	0.733	0.735
Random Forest	300 pixels	GIST+HOG	90.59%	0.012	0.908	0.906	0.904
SMO (RBF)	300 pixels	GIST+HOG	93.14%	0.008	0.932	0.931	0.931
SVM (Linear)	300 pixels	GIST+HOG	90.24%	0.010	0.904	0.902	0.902
XGBoost	300 pixels	GIST+HOG	89.69%	0.010	0.897	0.896	0.895
J48	300 pixels	GIST+HOG	76.42%	0.025	0.781	0.764	0.770
Random Forest	4096 pixels	GIST+HOG	92.91%	0.009	0.933	0.929	0.929
SMO (RBF)	4096 pixels	GIST+HOG	96.39%	0.004	0.965	0.964	0.964
SVM (Linear)	4096 pixels	GIST+HOG	93.26%	0.007	0.932	0.933	0.932
XGBoost	4096 pixels	GIST+HOG	93.61%	0.006	0.936	0.936	0.935
J48	4096 pixels	GIST+HOG	80.37%	0.022	0.813	0.804	0.806
Random Forest	Square root scheme	GIST+HOG	91.17%	0.011	0.915	0.912	0.911
SMO (RBF)	Square root scheme	GIST+HOG	95.35%	0.005	0.954	0.954	0.953
SVM (Linear)	Square root scheme	GIST+HOG	92.21%	0.008	0.922	0.922	0.922
XGBoost	Square root scheme	GIST+HOG	90.82%	0.009	0.911	0.908	0.909
J48	Square root scheme	GIST+HOG	73.86%	0.028	0.759	0.739	0.746

תרשים 28: טבלאות תוצאות הניסוי, [Bozkir, Tahilioglu, Aydos, & Kara, 2021, pp. 11-12]

מתוצאות אלו ניתן להסיק מספר מסקנות:

ראשית, ניתן לראות לפי מדיי ההצלחה כי מאפייני GIST תורמים יותר ליכולת הסיווג של המודל מאשר מאפייני HOG, כלומר חילוף מאפיינים על פי GIST נוטה להניב מאפיינים המתארים את התמונה הנתונה בצורה טובה יותר. יחד עם זאת, התוצאות הטובות ביותר הושגו לאחר שילוב המאפיינים שחולצו על ידי GIST עם אלו שחולצו על ידי HOG ללא תלות בסוג השיטה לסיווג שנבחרה ולכן נסיק כי שימוש במספר שיטות לחילוף מאפיינים מספק תיאור עשיר ושימושי יותר של תמונת הזיכרון.

שנית, נשים לב לכך שהמדדים הטובים ביותר הושגו בעת ניתוח תמונות הזיכרון המכילות ממד רוחב קבוע של 4096 פיקסלים וכי ברוב המקרים טיב התוצאות נמצא ביחס ישיר לרוחב תמונת הזיכרון הראשונית.

בנוסף, מבין שיטות למידת המכונה לסיווג שנבחנו, Sequential Minimal Optimization הניבה את התוצאות הטובות ביותר. תוצאה זו עולה בקנה אחד עם הסברה כי בעיית סיווג הנוזקות על פי תמונת הזיכרון הינה מורכבת ולא לינארית. סברה זו נתמכת גם על ידי העובדה כי מבין שיטות למידת המכונה מבוססות עצים – Random forest ו-XGBoost, אשר משתמשות במספר עצים ולכן גם מכילות כוח הבעה רב יותר, נטו להניב תוצאות מדויקות יותר מאשר J48 אשר כוללת עץ בודד וכן גם כוח הבעה נמוך יותר.

לבסוף, המודל הטוב ביותר: שימוש בגיט GIST וגם HOG לחילוף מאפיינים מתוך תמונת זיכרון בעלת רוחב התחלתי של 4096 פיקסלים והזנת מאפיינים אלו למסווג מבוסס SMO נבדק אל מול מספר שיטות אחרות לשם השוואת יעילות המודל.

Table 5 – Comparison of our best classifier with other references works.

Study	Accuracy	Precision	Recall	F1
Nataraj et al., 2011	91.40%	0.915	0.914	0.915
Dai et al., 2018	94.54%	0.946	0.945	0.945
Rezende et al., 2018	96.93%	0.970	0.969	0.969
Our best	96.36%	0.964	0.964	0.964

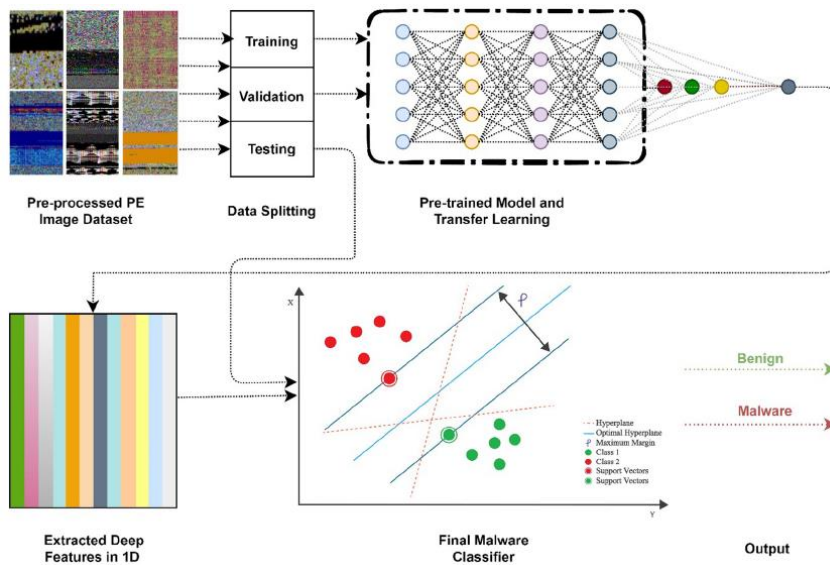
תרשים 29: השוואת המודל הטוב ביותר למודלים מעבודות קודמות, [Bozkir, Tahilioglu, Aydos, & Kara, 2021, p. 13]

המודל הראשון ברשימה זו מבוסס על שימוש במאפייני GIST וסיווג על פי שיטות למידת מכונה קלאסית, המודל השני ברשימה מבוסס על שימוש במאפייני HOG וסיווג על פי רשת נוירונים בעוד שהמודל השלישי משתמש בסיווג על פי רשת CNN עמוקה.

המודל המוצע במאמר התעלה על פני שני המודלים הראשונים ואף הגיע לתוצאות דומות לאלו של מודל מבוסס רשת עמוקה, אשר בעלת כוח הבעה רב יותר באופן משמעותי.

פרק 10: מבט לעומק – שיטת למידה עמוקה מבוססת תמונת קובץ הרצה לזיהוי נזקות

בפרק זה נציג שיטה לזיהוי נזקות הכוללת חילוף מאפייני קובץ הרצה כתמונה, ניתוח מאפיינים באמצעות



תרשים 30: מבנה השיטה לזיהוי נזקות על סמך תמונת קובץ הרצה בהעברת לימוד, [Shaukat, Luo, & Varadharajan, 2023, p. 7]

רשת עמוקה בהעברת לימוד (transfer learning) וראש סיווג מבוסס למידת מכונה קלאסית, כפי שתוארה במאמר [Shaukat, Luo, & Varadharajan, 2023].

10.1 – בניית בסיס נתונים מתאים ללמידה

בעיה נפוצה של מודלים ללמידת מכונה הינה הטיה כלפי משתני מטרה שגויים לאחר האימון. בעיה זו עלולה להתרחש כאשר המודל נחשף במהלך האימון לכמות לא מספקת או לא פרופורציונלית של נתונים המתאימים לאחד או יותר ממשתני המטרה האפשריים.

אחת הדרכים להתמודד עם בעיה זו (והדרך אשר נבחרה במאמר) היא העשרת מאגר הנתונים באמצעות דוגמאות סינתטיות.

באמצעות שימוש בטרנספורמציות פשוטות כגון שיקופים, סיבובים, שינויי צבע, חיתוך וכדומה, ניתן לייצר מתוך דוגמה נתונה מספר רב של דוגמאות חדשות מבלי לשנות את משתנה המטרה של הדוגמאות. תהליך זה יבוצע עד ליצירת מאגר נתונים המכיל מספיק דוגמאות בעבור כל משתנה מטרה ובאופן פרופורציונלי בין אחד למשנהו.

יתרון נוסף לשימוש בשיטה הוא שיפור יכולת המודל לבצע הכללה ולהימנע מהתאמת יתר, זאת משום שהמודל נחשף במהלך האימון לדוגמאות רבות יותר ווריאציות שונות אשר אינן מופיעות במאגר הנתונים המקורי.

לשם בניית המודל המוצע במאמר, נעשה שימוש בשלושה מאגרי מידע בעבור קבצי הרצה של נזקות (Microsoft malware dataset, Maling, VirusShare) ובמספר מאגרים שונים בעבור קבצי הרצה נקיים מנזקות (softpedia.com, download.com) (ועוד).

מאגרים אלו, בדומה לרוב מאגרי קבצי הנזקות הזמינים לשימוש, סובלים מייצוג חסר של נזקות חדשות לעומת ישנות ומוכרות יותר ומחוסר איזון במספר הנזקות הקיימות מכל סוג.

לאחר תהליך יצירת התמונות בעבור הקבצים שנאספו (תהליך אשר יתואר בסעיף הבא), נעשה שימוש ברצף של טרנספורמציות אשר הופעלו כל אחת אחרי השנייה בהסתברות נתונה על תמונה מקורית, לשם יצירת תמונה חדשה השייכת לאותה מחלקה.

כך הורחב מאגר נתוני האימון לכדי מאגר המכיל כמות שווה של דוגמאות מכל מחלקה.

10.2 – הצגת השיטה

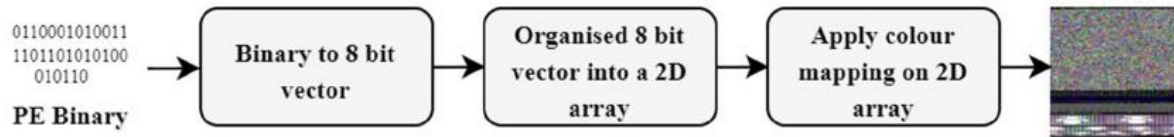
השיטה המוצגת במאמר מייצרת תמונה מתוך קובץ הרצה נבדק ומשלבת למידה עמוקה ולמידת מכונה קלאסית לשם סיווג הקובץ כנזקה או קובץ תמים.

10.2.1 – ייצוג קובץ הרצה כתמונה

כשלב ראשון בתהליך, נייצר תמונה צבעונית בעבור קובץ הרצה כתהליך חילוף מאפיינים לקובץ.

נוכל לבצע זאת על ידי קריאת קובץ ההרצה כרצף של בתים או על ידי בניית קובץ אסמבלי מתוך הקוד הבינארי של הקובץ והמרת הקובץ המתקבל לתמונה.

תהליך ההמרה, כפי שהוצג בפרקים קודמים, כולל קריאת קובץ כרצף של בתים, המרת כל בית לערך מספרי חיובי (unsigned integer), התייחסות לכל רצף של שלושה ערכים כאל מאפייני פיקסל בודד (RGB) והגדרת מאפייני הרוחב והאורך המבוקשים לכדי קבלת תמונה.



תרשים 31: אילוסטרציה של תהליך המרת קובץ הרצה בינארי לייצוג כתמונה, [Shaukat, Luo, & Varadharajan, 2023, p. 5]

בניסוי המתואר ב [Shaukat, Luo, & Varadharajan, 2023], הופקה מכל קובץ במאגר הנתונים תמונה על ידי קריאת קובץ ההרצה כרצף של בתים, המרת כל רצף של שלושה בתים לכדי ייצוג של פיקסל וסידור כתמונה על ידי קביעת רוחב התוצאה בהתאם לגודל קובץ ההרצה. תמונה זו תשמש כקלט למודל זיהוי הנוזקות.

10.2.2 – שימוש ברשת עמוקה לניתוח מאפיינים

בשלב השני, ננתח את התמונה שהושגה בשלב הקודם ונחלץ ממנה וקטור מאפיינים באמצעות שימוש במודל למידה עמוקה אשר אומן מראש למשימת עיבוד וסיווג תמונה.

מודלים כאלו, כדוגמת ResNet50 ו-VGG16, אומנו על מאגר התמונות ImageNet המכיל מיליוני תמונות ואלף מחלקות אפשריות לשם סיווג תמונה למחלקה המתאימה לה.

משום שאנו עוסקים במשימת זיהוי נוזקות ולא סיווג תמונות גנריות, אנו נשתמש במודל ללא שכבת הסיווג המופיעה בסופו וכך ננצל את תהליך הלמידה שהמודל עבר לשם חילוף וקטור מאפיינים יעיל ככל שניתן אשר יועבר לראש סיווג מתאים למשימת זיהוי הנוזקות אשר יתואר בסעיף הבא.

בניסוי המתואר ב [Shaukat, Luo, & Varadharajan, 2023], נבחנו 15 מודלים שונים של למידה עמוקה לשם ניתוח התמונה וחילוף וקטור המאפיינים. המודלים הותאמו למשימה על ידי הסרת ראש הסיווג המופיע בסוף המודל ויצירת וקטור מאפיינים לתמונה נבחנת מתוך פלט השכבה הלינארית האחרונה במודל.

10.2.3 – שימוש בלמידת מכונה קלאסית למשימת זיהוי הנוזקות

בשלב השלישי והאחרון, נשתמש בשיטת למידת מכונה קלאסית לשם סיווג קובץ נתון כנוזקה או כקובץ נקי על סמך וקטור המאפיינים שהושג בשלב הקודם.

בניסוי המתואר ב [Shaukat, Luo, & Varadharajan, 2023], נבחן שימוש ב-Support vector machine כמסווג על סמך וקטור המאפיינים. בנוסף, מבוצעת השוואה של מסווג זה לשיטות למידת מכונה קלאסית נוספות.

10.2.4 – הרציונל מאחורי השיטה

שיטות למידה עמוקה מאפשרות בניית מודל הכולל רשת אחת ויחידה אשר תחלץ מאפיינים מהקלט, תנתח מאפיינים אלו ותניב פלט מתאים, כדוגמת רשת קונבולוציה אשר מכילה שני נירוני פלט בשכבה האחרונה של ראש הסיווג לפני מעבר לפונקציית softmax לשם בחירת משתנה המטרה הצפוי.

עם זאת, השיטה המוצעת במאמר מחלקת את עבודת המודל לשני חלקים: חילוף מאפייני התמונה על ידי רשת נירונים עמוקה וסיווג לפי וקטור המאפיינים על ידי מודל למידת מכונה קלאסית.

בעוד שבניית מודל המסתמך על רשת עמוקה יחידה יהיה קל יותר לביצוע, המאמר מציג מספר סיבות לפיצול המשימה:

- מודל מבוסס למידה עמוקה מתחילה ועד סופו ידרוש משאבים רבים יותר למשימת האימון ובעת שימוש בהעברת למידה – יידרש עדכון של פרמטרי המודל לשם התאמתו לבסיס התמונות החדש, דבר אשר עלול לגרום להתאמת יתר לנתוני האימון.

- מודל מבוסס למידה עמוקה מתחילה ועד סוף יצורך מאגר נתונים רב מאוד לשם ביצוע אימון איכותי, אך כפי שצינו בסעיף 10.1 – ישנם קשיים בהשגת מאגרי נתונים גדולים, מעודכנים ומגוונים של נוזקות לשם ביצוע הליך למידה עשיר.

10.3 – מהלך הניסוי ותוצאותיו

נחלק את הניסוי המתואר במאמר לשני חלקים מרכזיים.

בחלק הראשון מבוצעת השוואה בין מספר שיטות למידה עמוקה מקצה לקצה לשם זיהוי הנוזקות, תוך התייחסות לבעיית ההטיה האפשרית במאגר הנתונים.

ובחלק השני נבחנת יעילותה של השיטה המוצעת במאמר, אותה הצגנו בסעיף 10.2.

10.3.1 – השוואה בין ביצועי מודלים ללמידה עמוקה למשימת זיהוי נוזקות

בחלק זה נבחנו מספר מודלים של למידה עמוקה לזיהוי נוזקות מבוסס תמונה.

ראשית, בכדי לבחון את השפעת הגיוון בייצוג של מאגר הנתונים על יעילות המודלים, מתוך מאגר כולל של 20000 קבצי נוזקות ו-10000 קבצים נקיים - הוגדרו שלושה מאגרי נתונים:

- מאגר A: מאגר הכולל 2000 קבצי נוזקות ו-10000 קבצים נקיים המחולקים לאימון, ולידיה ומבחן ביחס של 60:20:20, כלומר 1200 נוזקות ו-6000 קבצים נקיים לאימון ו-400 נוזקות ו-2000 קבצים נקיים לקבוצת הוולידציה והמבחן כל אחת.
- מאגר B: מאגר הכולל 20000 קבצי נוזקות ו-10000 קבצים נקיים המחולקים לאימון, ולידיה ומבחן ביחס של 60:20:20, כלומר 12000 נוזקות ו-6000 קבצים נקיים לאימון ו-4000 נוזקות ו-2000 קבצים נקיים לקבוצת הוולידציה והמבחן כל אחת.
- מאגר C: מאגר אשר דומה בתכולתו למאגר B, אך באמצעות הוספת נתונים סינתטיים כמתואר בסעיף 10.1, נתוני האימון מכילים 12000 קבצים נקיים, כאשר 6000 מהם הינם הקבצים המקוריים וה-6000 הנוספים נוצרו באופן סינתטי על מנת להשוות בין כמות הקבצים הנקיים לכמות קבצי הנוזקות הקיימים במאגר.

השימוש בשלושת מאגרים אלו נועד לבחון את התמודדות השיטות עם בעיית חוסר האיזון הנפוצה במאגרי נתונים של קבצי נוזקות. מאגר A מכיל כמות קטנה של נוזקות ביחס לקבצים הנקיים ומייצג מצב יחסית קרוב יותר לסביבה אמיתית איתה יצטרך מודל לזיהוי נוזקות להתמודד.

מאגר B מכיל רבה יותר של נוזקות ביחס לקבצים נקיים וכך מהווה דוגמה סבירה למאגר נתונים ממוצע עליו מאומן מודל זיהוי נוזקות.

ומאגר C משתמש בהוספת נתונים סינתטיים בכדי לנסות ולעזור למודל להימנע מלמידה מוטה לכיוון המחלקה הגדולה.

על כל אחד מהמאגרים הורצו 15 מודלים נפוצים של למידה עמוקה ומודל נוסף אשר מהווה כמעין פשרה בין מודל למידה עמוקה מקצה לקצה לבין השיטה המוצעת במאמר – מודל CNN-SVM.

מודל CNN SVM מורכב מרשת CNN המכילה 4 בלוקים של שכבת קונבולוציה, שכבת איגום MaxPooling, שכבת Dropout ושכבה לינארית מלאה ולאחר מכן ראש סיווג המורכב משתי שכבות שיטוח מלאות לפני השכבה הסופית. לבסוף, השכבה הסופית משתמשת בנורמת L2 במקום ביצוע softmax לשם קבלת ראש סיווג מבוסס SVM ומשתנה המטרה נבחר לפי נירון הפלט בעל הערך המירבי.

15 המודלים הנפוצים משתמשים בהעברת לימוד: המודלים הותאמו לבעיית זיהוי הנוזקות על ידי שמירת ערכי הפרמטרים של שכבות חילוף המאפיינים כפי שאומנו על מאגר התמונות של ImageNet, והחלפת השכבה הלינארית האחרונה של ראש הסיווג (המכילה 1000 משתני פלט, בעבור כל אחת ממחלקות מאגר התמונות ImageNet) ברצף של שכבת dropout, שיטוח ושכבה לינארית המכילה שני משתני פלט (בעבור קובץ נוזקה וקובץ נקי).

כל המודלים בשלב זה קיבלו כקלט תמונות שחולצו מתוך קבצי ההרצה במאגר והניבו כפלט חיזוי של נוזקה או קובץ נקי. המודלים אומנו כולם בתנאים של 150 epochs ו-batch size 32

להלן התוצאות הניסוי כפי שהופיעו במאמר [Shaukat, Luo, & Varadharajan, 2023, pp. 11-16]:

Model	Dataset A		Dataset B		Dataset C	
	Test loss	Test accuracy	Test loss	Test accuracy	Test loss	Test accuracy
VGG16	0.4575	90.71%	0.487	90.75%	0.2509	93.68%
ResNet50	31.42	69.38%	54.62	74.58%	30.54	74.84%
InceptionV3	3.599	72.55%	2.72	80.23%	1.9	80.46%
VGG19	0.4613	89.84%	0.3785	93.04%	0.2789	93.64%
MobileNet	5.44	70.25%	5.15	76.04%	4.39	76.13%
DenseNet169	4.702	77.95%	5.94	83.59%	2.66	84.15%
Xception	3.88	62.04%	4.29	73.93%	3.41	74.21%
DenseNet201	10.99	75.05%	14.24	76.23%	8.84	77.44%
Inception ResNet V2	4.63	52.80%	5.29	75.35%	4.21	75.55%
MobileNetV2	10.84	70.81%	9.12	73.97%	8.8	74.01%
ResNet152V2	9.099	77.44%	11.71	80.21%	8.95	80.31%
NasNetMobile	5.53	50.70%	3.52	78.43%	1.93	79.62%
AlexNet	8.49	68.54%	8.02	73.11%	7.14	77.48%
SqueezeNet	8.38	70.27%	8.26	74.95%	7.94	81.28%
RegNetY320	2.41	78.92%	1.93	82.54%	1.19	88.52%
CNN-SVM	0.4366	91.17%	0.3373	93.39%	0.2303	95.15%

כפי שניתן לראות, מודל CNN-SVM הניב את מדדי הדיוק הגבוהים ביותר מבין המודלים שנבדקו. מודל זה, בניגוד לשאר המודלים בניסוי, לא השתמש בטכניקה של העברת לימוד אלא אומן מתחילתו ועד סופו על מאגרי הנוזקות של הניסוי הספציפי שבוצע ולכן למד לבצע חילוף מאפיינים המתאים במיוחד לבעיה המוצגת.

ביצועי המודלים על פי רוב השתפרו בין מאגר A למאגר B ואף יותר למאגר C. מגמת שיפור זו ניתנת להסברה באמצעות שני גורמים – ראשית, מאגר C מכיל יותר דוגמאות ללמידה מאשר מאגר B, אשר מכיל בעצמו משמעותית יותר דוגמאות הרצה ממאגר A. רשתות עמוקות זקוקות למאגר נתונים גדול בכדי לבצע תהליך למידה איכותי ולכן ככל שנאפשר למודל ללמוד מיותר דוגמאות – על פי רוב כך גם ישתפרו ביצועיו. בנוסף, בניגוד למאגרים A ו-B – מאגר C מבצע אוגמנטציה של נתוני הלימוד וכך משיג כמות זהה של דוגמאות משתי המחלקות האפשריות, דבר זה מקל על המודלים ללמוד נכונה לסווג דוגמאות חדשות, שכן תהליך הלמידה לא היה מוטה לטובת מחלקה אחת ספציפית.

10.3.2 – בחינת ביצועי המודל המוצע לזיהוי נוזקות

בחלק זה בוצעה בחינה לעומק של השיטה המוצעת. לשם הניסוי, נבדקו 15 המודלים ללמידה עמוקה בהעברת לימוד אשר הופיעו גם בחלק הקודם ונבחנו פעם אחת לאחר עדכון ראש הסיווג למאגר הנתונים כמודלים מקצה לקצה (מופיע תחת רובריקת accuracy בטבלת התוצאות) ופעם כרכיב חילוף וניתוח המאפיינים בעבור מודלים המבוססים על השיטה המוצעת, כאשר פעולת הסיווג מבוצעת על ידי טכניקות למידת מכונה שונות. כל המודלים אומנו על מאגר C, בתנאים של 150 epochs ו-batch size 32.

המודל המוצע על ידי המאמר משתמש ספציפית במ SVM כטכניקת למידת המכונה לסיווג הקבצים.

Model	Accuracy	Bagging	Adaboost	Gradient Boosting	XGBoost	HistGradient Boosting	Logistic Regression	Random Forest	Naïve Bays	Decision Tree	K-Nearest Neighbour	Linear Discriminator Analysis	Proposed framework
VGG16	93.68	93.41	92.11	91.81	93.59	96.12	97.23	95.31	84.87	90.31	96.72	89.21	97.91
ResNet50	74.84	93.2	92.29	90.23	93.18	96.04	96.53	94.39	81.51	89.37	94.78	84.43	96.87
InceptionV3	80.46	91.61	90.33	89.49	92.34	94.31	95.21	92.37	83.82	84.41	92.12	93.81	96.02
VGG19	93.64	95.23	93.48	93.58	94.28	96.52	97.36	95.85	86.43	91.57	96.53	90.53	97.82
MobileNet	76.13	93.1	92.14	91.33	92.57	96.33	97.24	95.89	86.57	88.43	95.47	95.81	97.34
Xception	74.21	91.48	90.23	91.13	93.31	95.11	96.31	94.71	82.89	88.81	94.58	95.77	96.74
DenseNet169	84.15	92.93	89.99	92.44	94.73	96.27	97.43	95.77	85.23	90.21	96.81	96.79	97.55
DenseNet201	77.44	90.23	87.81	89.23	94.18	96.51	97.78	95.56	86.57	89.9	96.51	97.23	97.91
InceptionResNetV2	75.55	91.44	90.25	90.93	94.03	96.48	97.41	94.97	87.1	88.53	95.91	96.14	97.9
MobileNetV2	74.01	92.51	91.14	91.82	93.81	96.43	97.35	95.35	87.41	89.43	95.58	95.44	97.5
NasNetMobile	79.62	93.14	92.1	91.22	93.48	95.93	96.33	94.56	84.86	88.33	95.21	95.23	96.84
ResNet152V2	80.31	93.22	90.98	91.58	92.13	95.22	96.24	94.23	85.75	88.58	92.47	95.35	96.73
AlexNet	77.48	89.97	88.54	90.27	90.85	94.28	95.28	93.43	83.29	88.56	93.86	86.59	96.25
SqueezeNet	81.28	90.72	88.96	91.55	91.67	95.08	95.94	94.27	83.84	89.18	94.28	88.51	97.4
RegNetY320	88.52	94.6	92.23	91.58	93.98	96.87	98.82	96.41	85.33	88.81	96.61	96.61	98.93

תרשים 32: תוצאות השימוש בשיטה על מאגר נתונים C, בחלוקה לפי המודל הנבחר לחילוף מאפיינים בהעברת לימוד והמודל הנבחר כראש הסיווג, [Shaukat, Luo, & Varadharajan, 2023, p. 17]

את אחוז השיפור המתקבל משימוש בשיטה המוצעת, כתלות בשיטת למידת המכונה המחליפה את ראש הסיווג, על פני שימוש במודל למידה עמוקה קצה לקצה ניתן לראות בטבלה הבאה:

Model	Accuracy	Bagging	AdaBoost	Gradient Boosting	XGBoost	HistGradient Boosting	Logistic Regression	Random Forest	Naïve Bays	Decision Tree	K-Nearest Neighbour	Linear Discriminator Analysis	Proposed Framework
VGG16	93.68	-0.27	-1.57	-1.87	-0.09	2.44	3.55	1.63	-8.81	-3.37	3.04	-4.47	4.23
ResNet50	74.84	18.36	17.45	15.39	18.34	21.2	21.69	19.55	6.67	14.53	19.94	9.59	22.03
InceptionV3	80.46	11.15	9.87	9.03	11.88	13.85	14.75	11.91	3.36	3.95	11.66	13.35	15.56
VGG19	93.64	1.59	-0.16	-0.06	0.64	2.88	3.72	2.21	-7.21	-2.07	2.89	-3.11	4.18
MobileNet	76.13	16.97	16.01	15.2	16.44	20.2	21.11	19.76	10.44	12.3	19.34	19.68	21.21
Xception	74.21	17.27	16.02	16.92	19.1	20.9	22.1	20.5	8.68	14.6	20.37	21.56	22.53
DenseNet169	84.15	8.78	5.84	8.29	10.58	12.12	13.28	11.62	1.08	6.06	12.66	12.64	13.4
DenseNet201	77.44	12.79	10.37	11.79	16.74	19.07	20.34	18.12	9.13	12.46	19.07	19.79	20.47
InceptionResNetV2	75.55	15.89	14.7	15.38	18.48	20.93	21.86	19.42	11.55	12.98	20.36	20.59	22.35
MobileNetV2	74.01	18.5	17.13	17.81	19.8	22.42	23.34	21.34	13.4	15.42	21.57	21.43	23.49
NasNetMobile	79.62	13.52	12.48	11.6	13.86	16.31	16.71	14.94	5.24	8.71	15.59	15.61	17.22
ResNet152V2	80.31	12.91	10.67	11.27	11.82	14.91	15.93	13.92	5.44	8.27	12.16	15.04	16.42
AlexNet	77.48	12.49	11.06	12.79	13.37	16.8	17.8	15.95	5.81	11.08	16.38	9.11	18.77
SqueezeNet	81.28	9.44	7.68	10.27	10.39	13.8	14.66	12.99	2.56	7.9	13	7.23	16.12
RegNetY320	88.52	6.08	3.71	3.06	5.46	8.35	10.3	7.89	-3.19	0.29	8.09	8.09	10.41
Average Increase		11.70	10.08	10.46	12.45	15.08	16.08	14.12	4.28	8.21	14.41	12.41	16.56

תרשים 33: אחוז השיפור המתקבל משימוש בשיטה המוצעת, כתלות בשיטת למידת המכונה המחליפה את ראש הסיווג, על פני שימוש במודל למידה עמוקה מקצה לקצה, [Shaukat, Luo, & Varadharajan, 2023, p. 18]

כפי שניתן לראות, על פי רוב – השיטה המוצעת מייצרת מודלים מדויקים יותר. השיפור הממוצע המתקבל על ידי ביצוע חילוך מאפיינים באמצעות מודל למידה עמוקה בהעברת לימוד וניתוח מאפיינים אלו על ידי שיטת למידת מכונה קלאסית עומד על 14.11% בניסוי זה.

המודל המוצע ספציפית, אשר משתמש ב-SVM כטכניקת למידת המכונה לסיווג המאפיינים הניב את התוצאות הטובות ביותר, עם ממוצע שיפור של 16.56% ביחס למודל המקורי, בעבור המודלים שנבדקו.

לבסוף, מבוצעת השוואה של המודל הטוב ביותר שנבחן בעבור השיטה (חילוך מאפיינים באמצעות RegNetY320 וסיווג באמצעות SVM) לבין מודלים ממספר עבודות קודמות על ידי הרצתם על מאגר Malimg:

year	Dataset	Reference#	Models	Accuracy	Precision	Recall	F-score
2011	Malimg	Nataraj et al. (2011)	K-NN	98.08%	–	–	–
2017	Malimg	Luo and Lo (2017)	CNN+LBP	93.72%	94.13%	92.54%	93.33%
2017	Malimg	Rezende et al. (2017)	ResNet50	97.48%	–	–	–
2017	Malimg	Rezende et al. (2017)	ResNet-50	98.62%	–	–	–
2017	Malimg	Makandar and Patrot (2017)	GIST+SVM	98.88%	–	–	–
2018	Malimg	Lo et al. (2019)	Xception	98.52%	–	–	–
2019	Malimg	Bhodia et al. (2019)	ResNet34	94.80%	–	–	–
2019	Malimg	Roseline et al. (2019)	Ensembling using RF	97.82%	98%	98%	98%
2019	Malimg	Agarap (2017)	CNN-SVM	77.22%	84%	77%	79%
2019	Malimg	Ben Abdel Ouahab et al. (2019)	K-NN	97%	–	–	–
2019	Malimg	Gibert et al. (2019)	CNN	97.18%	–	–	–
2019	Malimg	Vinayakumar et al. (2019)	CNN+LSTM	96.3%	96.3%	96.2%	96.2%
2019	Malimg	Vinayakumar et al. (2019)	RF	78.6%	–	–	–
2019	Malimg	Vinayakumar et al. (2019)	NB	80.5%	–	–	–
2019	Malimg	Vinayakumar et al. (2019)	KNN	41.8%	–	–	–
2019	Malimg	Vinayakumar et al. (2019)	DT	79.5%	–	–	–
2019	Malimg	Singh et al. (2019)	ResNet50	96.08%	95.76%	96.16%	95.96%
2020	Malimg	Vasan et al. (2020b)	VGG16,	97.59%	–	–	–
2020	Malimg	Vasan et al. (2020b)	ResNet50	95.94%	–	–	–
2021	Malimg	Hemalatha et al. (2021)	DenseNet	98.23%	97.78%	97.92%	97.85%
2022	Malimg	Proposed framework		99.06%	98.47%	98.52%	98.49%

תרשים 34: השוואת המודל הטוב ביותר שנבחן בעבור השיטה אל מול מודלים ממאמרים קודמים, על מאגר הנתונים Malimg, [Shaukat, Luo, & Varadharajan, 2023, p. 18]

פרק 11: הצגת שיטת למידה עמוקה מבוססת מקודד עצמי

בפרק זה נבנה ונבחן שיטה נוספת לזיהוי נוזקות, המורכבת ממחלף מאפיינים הבנוי מרכיב המקודד של מקודד עצמי ושכבות לינאריות כראש סיווג.

מחברת הקוד של הניסוי נמצאת ב [Tiferet, n.d.].

11.1 – מוטיבציה לשיטה

הרעיון לגישה הגיע לאחר קריאת החומרים לביצוע העבודה. בפרט, רציתי לבחון שיטה אשר תשלב את היתרונות של הגישות המוצגות בשני הפרקים הקודמים ותוכל להתמודד בצורה סבירה עם מגבלותיהן.

בפרק 9 הצגנו גישה שמתבססת על תמונות זיכרון בעת הרצת תוכנה לשם השגת מידע אודות התוכנה. בכך, הגישה נחשפת לפעולת התוכנה ללא יכולת הסתרה ובאופן אשר מתאר את פעולתה בצורה מדויקת. מנגד, השיטה משתמשת בחילוף מאפיינים מתוך התמונה על פי שיטות ראייה ממוחשבת – שיטות אשר דורשות תכנון ובחינה של מומחים בתחום לשם הסקה ועדכון של מאפיינים רלוונטיים למשימה.

בפרק 10 הצגנו גישה אשר שילבה למידה עמוקה באמצעות העברת לימוד לשם חילוף מאפייני תמונות קובץ הרצה, עם סיווג על סמך מאפיינים אלו מבוסס שיטות למידת מכונה קלאסית. בדרך זו הוצגו תוצאות טובות מאוד, אך חילוף מאפייני התמונה באמצעות מודלים אשר אומנו על מאגר תמונות מעולם תוכן אחר, במיוחד בשל השוני הרב בין התמונות המתקבלות מניתוח נתוני קבצי נזקות כתמונה לבין התמונות המופיעות במאגר ImageNet, עלול לגרום לפספוס מאפיינים רלוונטיים או הסקת מאפיינים שאינם תואמים למשימה הנדרשת.

בנוסף, בשני המאמרים שסיקרו, נדרשה התייחסות לבעיות המחסור וחוסר האיזון של מאגרי הנוזקות הקיימים. בין אם באמצעות יצירת מאגר חדש המתאים לצורכי המשימה לבין עיבוי מאגר קיים באמצעות דוגמאות סינתטיות.

השיטה אותה נציג בסעיף הבא מתבססת על ניתוח תמונות זיכרון, בדומה לשיטה שהוצגה בפרק 9 אך מבצעת את תהליך חילוף המאפיינים באמצעות למידה עמוקה ובכך חוסכת את התלות בידע ספציפי לתחום לטובת יכולת למידה של הרשת עצמה. בנוסף, בשל השימוש במקודד עצמי מתאפשרת למידה של חילוף מאפייני תמונות גם ממאגרים נוספים שאינם מסווגים, מתמונות שנוצרו באופן סינתטי ועוד.

11.2 – הצגת השיטה

השיטה המוצגת מורכבת משני מודלים ללמידה עמוקה.

- מקודד עצמי מבוסס שכבות קונבולוציה – מקודד עצמי אשר מאומן על תמונות ללא סיווג בכדי לייצר וקטור מאפיינים מתוך תמונה נתונה בשלב המקודד, ולשחזר את התמונה המקורית לפי וקטור המאפיינים שיוצר בשלב המפרש.
- מודל לסיווג אשר מכיל את שכבות המקודד מתוך המקודד העצמי, בהעברת לימוד, וראש סיווג הבנוי מבולוקים של שכבות לינאריות.

מבנה המודל:

11.2.1 – בנייה ואימון של מקודד עצמי

כשלב ראשון, עלינו לבנות מקודד עצמי מבוסס שכבות קונבולוציה ולאמנו על מאגר תמונות למטרה של יצירת מצב חבוי בתהליך הקידוד ושחזור התמונה המקורית בתהליך הפירוש.

11.2.2 – איסוף מידע אודות זיכרון וייצוג כתמונה

כשלב שני נאסף מידע אודות מצב הזיכרון בעת ריצת התוכנה הנבדקת ונייצג מידע זה כתמונה. השיטה המוצגת, בדומה לזו המוצגת בפרק 9, מתבססת על תמונת זיכרון נדיף ולכן נבצע את תהליך בניית התמונה בהתאם לסעיפים 9.2.1 ו-9.2.2.

11.2.3 – סיווג באמצעות מודל מבוסס למידה עמוקה

החלק האחרון בתהליך כולל יצירת מודל לסיווג המורכב מרכיב המקודד מתוך מודל המקודד העצמי שאומן בסעיף 11.2.1, ואחריו ראש סיווג. מודל זה מאומן על מאגר תמונות הזיכרון של נזקות מסווגות כאשר הפרמטרים הנלמדים הינם רק אלו השייכים לראש הסיווג.

לאחר אימון המודל יהיה ניתן לסווג קובץ החשוד כנוזקה על ידי יצירת תמונת זיכרון מתאימה וניתוחה באמצעות מודל זה.

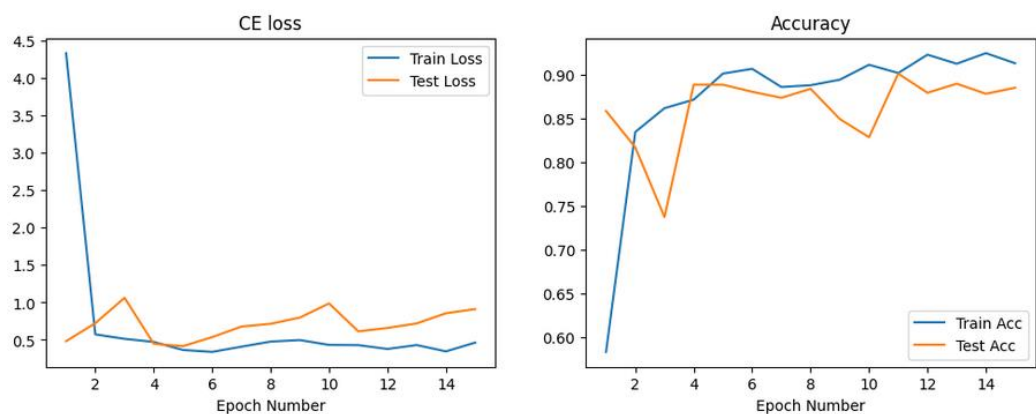
11.3 – מהלך הניסוי ותוצאותיו

לשם הניסוי השתמשתי במאגר תמונות הזיכרון בעבור קבצים, שפורסם במאמר שהוצג בפרק 9, ומצוי ב [Dumpware10, n.d.].

מתוך המאגר בחרתי בתמונות הזיכרון אשר הורכבו עם רוחב 4096 פיקסלים והוקטנו לכדי אורך ורוחב של 300 פיקסלים. על מאגר זה נבחנו מספר שיטות:

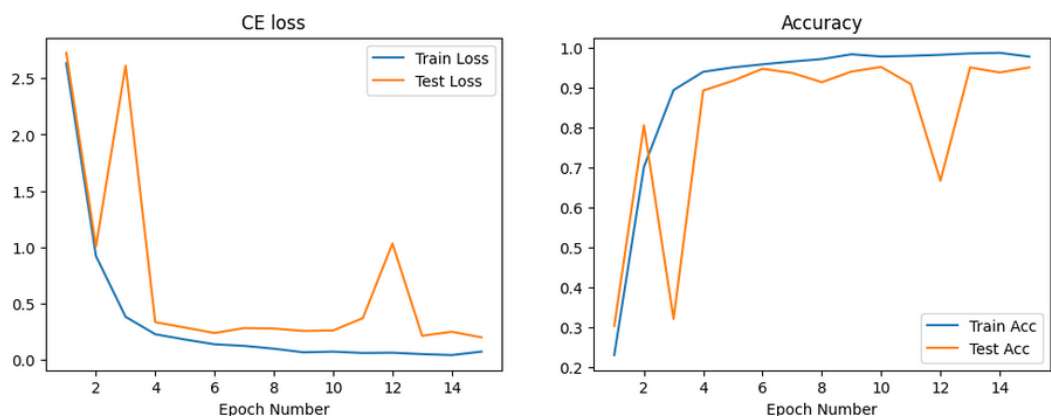
ראשית נבחן מודל המתבסס על העברת לימוד של רשת resnet18 לשם חילוץ מאפייני התמונה וסיווג על בסיס שכבה לינארית נלמדת.

מודל זה, המורכב משכבות חילוץ המאפיינים של רשת resnet18 ושכבה לינארית נלמדת כמסווג, אומן בתנאים של 0.03 learning rate, 15 epochs והגיע לתוצאות המתוארות בגרף:



לאחר מכן נבחן מודל רשת קונבולוציה הבנויה מבלוקים בסיסיים לפי מודל resnet אשר אומנה כולה על מאגר נתוני האימון.

מודל זה, המורכב מארבעה בלוקים שיריים ושכבה לינארית כמסווג, אומן בתנאים של 0.03 learning rate, 15 epochs והגיע לתוצאות המתוארות בגרף:



ולבסוף נבחן המודל המוצע באופן הבא:

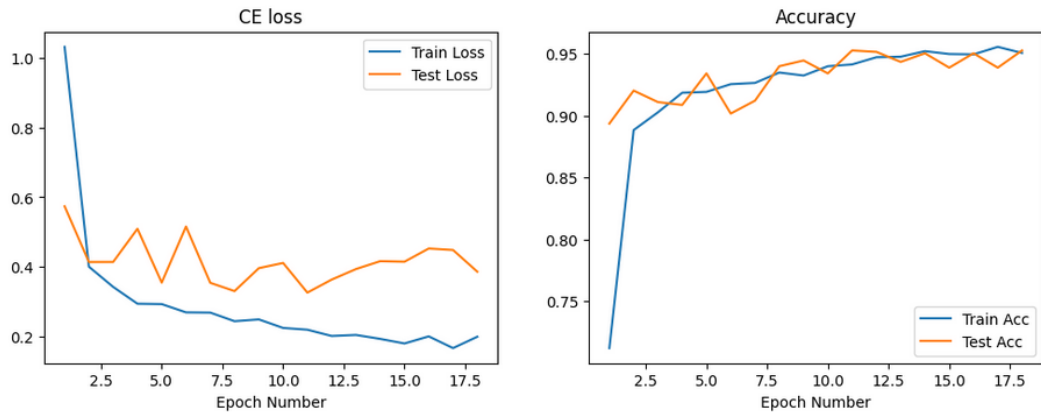
נבנה מקודד עצמי המכיל במקודד 8 שכבות קונבולוציה המייצרות מתוך קלט מסוג [3: 300: 300] ייצוג חבוי מסוג [1: 1: 1024] ומפרש הכולל 8 שכבות קונבולוציה משוחלפת אשר הפוכות לשכבות המקודד.

מקודד עצמי זה אומן על מאגר תמונות האימון, לצד בחינת יכולתו לשחזר תמונות מתוך מאגר תמונות המבחן.

נבנה מודל חדש המכיל את חלק המקודד מתוך המקודד העצמי, בהעברת לימוד, וראש סיווג המורכב מצירוף בלוקים של שכבה לינארית, שכבת ReLU, נרמול dropout.

ולבסוף אומן מודל זה על מאגר האימון ונבחן על מאגר המבחן, כאשר הפרמטרים שאומנו שייכים לראש הסיווג.

מודל זה אומן בתנאים של 0.03 learning rate, 18 epochs והגיע לתוצאות המתוארות בגרף:



תוצאות המודלים הסופיות מופיעות בטבלה הבאה:

Model	Training loss	Training acc	Testing loss	Testing acc
Resnet based transfer learning	0.4638	91.34%	0.9121	88.54%
CNN based on residual blocks	0.0713	97.71%	0.1983	95.02%
Evaluated model: Encoder and classifier	0.198	95.05%	0.3853	95.25%

המודל המוצע מספק תוצאות תחרותיות ביחס למודלים האחרים שנבדקו בניסוי זה וקרובות למודל המוצע במאמר [Bozkir, Tahilioglu, Aydos, & Kara, 2021], אותו הצגנו בפרק 9.

היתרון המרכזי בשיטה המוצעת נובע מהשימוש במקודד עצמי לשם חילוץ מאפייני התמונה. באופן זה ניתן לשפר את ביצועי מחלץ המאפיינים על ידי אימון מודל המקודד העצמי על גבי מאגרי תמונות ללא סיווג, תמונות שנוצרו באופן סינתטי על ידי ביצוע טרנספורמציות ואף תמונות מעולמי תוכן אחרים. ככל שהייצוג החבוי של התמונה יהיה עשיר ומדויק יותר – כך תהיה לראש הסיווג קל יותר לסווג נכונה את הקובץ.

עם זאת, משום שמדובר בשיטה מבוססת למידה עמוקה יהיה צורך בשימוש במשאבים רבים ובמאגר נתונים גדול יותר בכדי להגיע לתוצאה אופטימלית באמצעותה.

הניסוי שביצעתי מהווה בדיקה רעיונית של השיטה. ניתן לחדד את השיטה אף יותר במגוון דרכים כגון העשרת נתוני האימון של המקודד העצמי בנתונים סינתטיים ומאגרי מידע נוספים, בחינת החלפת ראש הסיווג בשיטות למידת מכונה קלאסיות כפי שנעשה בניסוי המתואר בפרק 10 ובחינת איסוף נתוני קבצים בתהליכים נוספים בנוסף לתמונת זיכרון נדיף.

סיכום:

זיהוי וסיווג נזקקות הינה משימה מורכבת בשל קצב הגידול הרב של נזקקות חדשות לצד טכניקות מתקדמות בהן נוקטים כותבי הנוזקות בכדי להסתירן ממנגנוני ההגנה. למידת מכונה מהווה כלי שימושי לזיהוי נזקקות, שכן היכולת ללמוד מכמויות גדולות של נתונים ולזהות דפוסים מאפשרת להתמודד עם הנוזקות אשר ממשיכות להתפתח בסיכויי הצלחה גבוהים מאשר שיטות אחרות.

בעבודה זו הצגנו את תחום למידת המכונה ולאחר מכן הדגמנו את אופן השימוש בו לשם זיהוי נזקקות.

הצגנו שיטות למידת מכונה קלאסיות אשר שייכים לאסכולה של למידה מפוקחת, שיטות ללמידה לא מפוקחת וגם שיטות ללמידה עמוקה, אשר מתבססות על רשתות נוירונים.

לאחר מכן הצגנו בסקירה כללית מהן נזקקות בכלל ווירוסים בפרט וסקרנו שיטות שונות למשימת זיהוי הנזקקות, משיטות שאינן משתמשות בלמידת מכונה ועד שיטות המערבות למידה עמוקה.

בפרקים 9 ו-10 ביצענו סקירה לעומק של שתי שיטות ספציפיות למשימת הזיהוי אשר הוצגו במאמרים שנבחרו בקפידה, תוך התחשבות ברלוונטיות לנושא העבודה.

השיטה המוצגת בפרק 9, לפי [Bozkir, Tahilioglu, Aydos, & Kara, 2021], ביצעה ניתוח של תמונות הזיכרון הנדיף בעת הרצת תוכנה נבדקת באמצעות חילוץ מאפיינים של ראייה ממוחשבת אשר הועברו למודל למידת מכונה קלאסית לשם סיווג הקובץ כנוזקה או קובץ נקי. שיטה זו הניבה תוצאות טובות, אשר מתחרות עם שיטות מבוססות למידה עמוקה שלהן פוטנציאל כוח הבעה רב יותר מהסיווג על פי למידת מכונה קלאסית.

השיטה המוצגת בפרק 10, לפי [Shaukat, Luo, & Varadharajan, 2023], ביצעה ניתוח של קובץ הרצה של תוכנה נבדקת כתמונה. התמונה הועברה למודל רשת קונבולוציה אשר אומנה על מאגר תמונות נפרד, באמצעות העברת לימוד, לשם חילוץ המאפיינים לאחר מכן בוצע סיווג בשיטת למידת מכונה קלאסית על פי ווקטור המאפיינים שהושג. בניסוי התקבלו תוצאות טובות יותר בעבור שיטות המורכבות ממודל אחד לחילוץ מאפייני הקובץ ומודל שני לסיווג, על פני מודל יחיד אשר מאומן לבצע שני שלבים אלו יחדיו. בנוסף, הבחינה של מודל ביניים, בסעיף 10.3.1, אשר אומן ספציפית על מאגר הקבצים והניב תוצאות טובות ממודלים אשר השתמשו בהעברת לימוד העלתה את האפשרות כי ישנו יתרון פוטנציאלי לרכיב חילוץ מאפיינים אשר אומן על תמונות בעלות מאפיינים דומים לאלו אותן עליו לסייע לסווג נכונה.

לבסוף, בפרק 11 הצגנו שיטה נוספת, אותה בנינו בניסיון לשלב את יתרונות השיטות שסקרנו קודם לכן ולסייע בטיפול בנקודות התורפה שלהן, לצד בעיית המחסור של מאגרי נתונים איכותיים לאימון רשת נוירונים עמוקה לסיווג נזקקות. השיטה אותה הצגנו כוללת שני מודלים: מודל מקודד עצמי אשר אומן על תמונות זיכרון של קבצים לשם יצירת ייצוג חבוי ושחזור התמונה המקורית על בסיס הייצוג החבוי שנוצר ומודל לסיווג אשר מורכב מחלק המקודד של המקודד העצמי בצירוף ראש סיווג אשר מאומן ספציפית למשימת סיווג הנוזקות.

בניסוי שביצענו התקבלו תוצאות תחרותיות ביחס לשיטות שסקרנו. בפרט, המודל אותו ניסינו הגיע למדד דיוק של 95.25% כאשר השיטה המוצגת בפרק 9, הגיעה לדיוק של 96.39% על אותו מאגר נתונים. יחד עם זאת, היתרון הגדול בשיטה נובע מהשימוש במודל מקודד עצמי לשם ביצוע חילוץ מאפיינים מקובץ נבחן, שכן מודל מסוג זה ניתן לאמן על מגוון רחב של תמונות שאינן בהכרח מסווגות ואף כאלו שיוצרו באופן סינתטי. בשל תכונה זו – המודל המוצע צפוי להתמודד בצורה טובה עם המחסור בנתוני נזקקות איכותיים לאימון ועם הנטייה של מודלים מתחום הלמידה העמוקה לצרוך כמויות גדולות של נתוני אימון בכדי להניב תוצאות טובות.

לדעתי, בעתיד יעשה שימוש נרחב אף יותר בשיטות למידה עמוקה לשם זיהוי וסיווג נזקקות הן בשל כוח ההבעה הרב של שיטות אלו והן בשל היכולת ללמוד לעבד נתונים רלוונטיים באופן עצמאי וללא צורך בהתערבות מומחים בתחום היעד.

ביבליוגרפיה

- Aslan, O., & Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. *IEEE Access*, 8.
- Bozkir, A. S., Tahilioglu, E., Aydos, M., & Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers & Security*.
- Chaudhary, M. (n.d.). *Random Forest Algorithm - How It Works & Why It's So Effective*. Retrieved from turing: <https://www.turing.com/kb/random-forest-algorithm>
- Dumpware10. (n.d.). Retrieved from Hacettepe University: <https://web.cs.hacettepe.edu.tr/~selman/dumpware10/>
- Gilbert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*(153).
- Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques - Third Edition*. Morgan Kaufmann.
- Jones, M. T. (2017, December 4). *Models for machine learning*. Retrieved from ibm: <https://developer.ibm.com/articles/cc-models-machine-learning/>
- ProcDump. (n.d.). Retrieved from Microsoft Learn: <https://learn.microsoft.com/en-us/sysinternals/downloads/procdump>
- Raj, A. (2021, April 8). *Introduction to Classification Using K Nearest Neighbours*. Retrieved from towardsdatascience: <https://towardsdatascience.com/getting-acquainted-with-k-nearest-neighbors-ba0a9ecf354f>
- Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press. Retrieved from <https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf>
- Shaukat, K., Luo, S., & Varadharajan, V. (2023). A novel deep learning-based approach for malware detection. *Engineering Applications of Artificial Intelligence*(122).
- Stallings, W., & Brown, L. (2018). *Computer Security - Principles and Practice Fourth Edition*. Pearson Education Limited.
- Sultanik, E. (n.d.). *bin2png*. Retrieved from GitHub: <https://github.com/ESultanik/bin2png>
- Tiferet, H. (n.d.). *Experimental testing of malware classification approaches*. Retrieved from GitHub: https://github.com/Hadar-Tiferet/Malware_Classification_DL/blob/main/malware_classification.ipynb
- What is machine learning?* (n.d.). Retrieved from ibm: <https://www.ibm.com/topics/machine-learning>
- Zhang, A., Smola, A. J., Li, M., & Lipton, Z. C. (n.d.). *Dive into Deep Learning*. Retrieved from <https://d2l.ai/>