The Academic College of Tel-Aviv

THE SCHOOL OF COMPUTER SCIENCE

The impact of network architecture, training process and parameters on adversarial attacks

February 2025

Thesis submitted in partial fulfillment of the requirements for the M.Sc. degree in the School of Computer Science of the Academic College of Tel Aviv University

By

Kfir Garame

The research work for the thesis has been carried out under the supervision of

Prof. Adi Shraibman

תודות

תחילה, ברצוני להודות לפרופ' ולמנחה, עדי שרייבמן, על הליווי הצמוד בעבודת המחקר, על היכולת לערער על הנחות יסוד, לשאול את השאלות הנכונות ולפתוח כיווני מחשבה חדשים. חלק משמעותי בהתפתחות המחקר ובהתפתחותי המקצועית והאישית נזקף לזכות הליווי והתמיכה שהענקת לי לאורך הדרך. תודה מעומק הלב.

תודה מעומק לבי להראלה, אשתי היקרה, שבלעדיה, ובלעדי התמיכה הבלתי מתפשרת שלה, המחקר הזה לא היה יכול להתקיים.

תוכן עניינים

הקדמה

נושא המחקר

בשנים האחרונות אנו עדים להתפתחות משמעותית בייצור מוצרים המבוססים על למידת מכונה, ובפרט על למידה עמוקה. מוצרים אלה נשענים על ידע ומידע שנאספו במאגרים שונים לאורך עשרות שנים.

השילוב בין מוצרים מבוססי למידת מכונה לבין איסוף מידע מחיישנים שונים מאפשר לבצע פעולות שבעבר נחשבו למדעי בדיוני, וכל זאת באופן מהיר, אוטומטי ולעיתים אף אוטונומי.

תהליך זה בא לידי ביטוי בכל תחומי החיים: תעשיית הרכב, קווי ייצור ותהליכים במפעלים, שוק ההון, מוסדות ממשלתיים, צבא, רפואה וחקר החלל.
עם זאת, רגולציה ממשלתית לעיתים מגבילה אוטונומיה מלאה של מוצרים אלו. למרות ההגבלות, המוצרים הפכו לחלק בלתי נפרד משגרת חיינו, בעיקר בתחומים המוגדרים קריטיים.

אנו חיים בעיצומו של עידן חדש, שבו הקשר בין אדם למכונה הוא חזק יותר מאי פעם. ככל שאנו נשענים יותר על טכנולוגיות אלו, כך גוברת החשיבות להגן עליהן מפני מניפולציות חיצוניות – מכוונות או מקריות – המשפיעות על יכולת החיזוי והדיוק שלהן.

במאמר זה נבחן את היכולת להטעות מודלים המבוססים על למידת מכונה באמצעות תקיפות המבוצעות על ידי מניפולציה של הקלט. נציג כיצד שילוב של פרמטרים ורכיבים מרכזיים בתהליך הלמידה ובארכיטקטורת המודל משפיעים משמעותית על יכולתו להתמודד עם תקיפות נפוצות.

הצדקת המחקר

בשנים האחרונות קיימת תלות הולכת וגוברת של המין האנושי במערכות מבוססות Machine Learning (ML) ומודלים מבוססי Deep Learning. במקביל להתפתחות זו , נערכו מחקרים רבים שהדגימו את האופן הפשוט בו אפשר לייצר תקיפה אפקטיבית ויעילה תוך השקעת מאמץ נמוך מצד התוקף. [1] Fast Gradient Sign Method (FGSM) ו- [2] Projected Gradient Descent (PGD) הן דוגמאות לתקיפות המבוססות על שינויים מזעריים בקלט, שמטרתן לבלבל את המודל. מדובר בתקיפות פשוטות מאוד, המאפשרות לתוקף לייצר תקיפה משמעותית, שלעיתים קרובות קשה לזיהוי על ידי אדם והן על ידי מכונה.

מחקרים שונים הצביעו על כך שתקיפה של מודל אחד, בעל ארכיטקטורה מסוימת, יכולה להשפיע באופן משמעותי על מודל אחר, גם אם הארכיטקטורה שלו שונה. תופעה זו, שבה התוקף אינו זקוק למידע מפורט על מודל היעד, נקראת Transferability [3]. הפשטות והעוצמה של תקיפות מסוג זה יוצרות אתגר משמעותי ומעודדות מאות מחקרים מתקדמים שמטרתם לפתח שיטות הגנה ושיפור עמידות המודלים.

אחת הדרכים הנפוצות להתמודדות עם תקיפות אלו היא באמצעות Adversarial Learning [4], שבו המגן מייצר דוגמאות קלט שעברו מניפולציה כחלק מתהליך האימון. שיטה זו מאפשרת למודל ללמוד

גם את מרחב הקלטים שעברו מניפולציה ולהפוך לעמיד יותר בפני תקיפות מסוג זה. בנוסף, קיימות שיטות נוספות לשיפור עמידות המודל, כמו ביצוע Smoothing בשכבת התיוג [5] או הוספת רעש לתהליך הלמידה של המודל [6].

מחקרים נוספים מציעים גישות שונות וחדשניות לשיפור עמידות המודלים. אך עדיין, אין פתרון קסם.

מציאת שיטה פשוטה ויעילה שתאפשר עמידות בפני תקיפות מסוג WhiteBox, GrayBox או BlackBox יכולה להוביל לפריצת דרך משמעותית ביכולת השימוש במוצרים מבוססי ML ו- Deep Learning בשנים הבאות.

תיחום המחקר

המחקר מתמקד בארכיטקטורת המודל, הפרמטרים ותהליך הלמידה של מודלי Deep Learning בתחום הראייה הממוחשבת, עם דגש על בעיות סיווג. מטרת המחקר היא לפתח שיטה או מנגנון שיאפשרו שיפור בעמידות המודלים לתקיפות מסוג WhiteBox, שהן מהנפוצות ביותר ונחקרות בהקשר של למידה עמוקה. במסגרת המחקר, נתמקד במיוחד בתקיפות מסוג זה המנצלות את הידע של התוקף על הגרדיאנט של המודל, במטרה להבין את השפעתן ולפתח אמצעי הגנה יעילים נגדן. בנוסף, נדגים את האתגר שבהתמודדות עם תקיפות המבוססות על תופעת ה-Transferability , ונראה כי אין בהכרח קשר בין הגנה מפני תקיפות המתבססות על גרדיאנט לבין הגנות שמונעות Transferability של תקיפות למודלים אחרים. בכך, נדגיש את הצורך בחשיבה מחודשת על מנגנוני הגנה שמסוגלים להתמודד עם מגוון רחב יותר של תקיפות.

כחלק מהמחקר, נתמקד במסד הנתונים CIFAR-10 כמקרה בוחן עיקרי, בשל יכולתו לשלב בין יעילות גבוהה בביצוע ניסויים לבין אתגר מספק בייצוג בעיות אמיתיות. הבחירה ב-CIFAR-10 מאפשרת בחינה מעמיקה של מודלים תוך כדי שמירה על איזון בין פשטות לאתגר. בניגוד לכך, מסד הנתונים MNIST, למרות היעילות שבו, נחשב קל מדי ואינו בהכרח משקף בעיות מורכבות מהעולם האמיתי.

בנוסף, נשתמש גם במסדי הנתונים INTEL ו FOOD-101 כדי להבטיח שהתוצאות המתקבלות הן מקיפות ומותאמות למאגרים המייצגים מגוון רחב של תרחישים וברזולוציה הקרובה יותר למציאות.

המחקר יתמקד בשתי שיטות תקיפה מרכזיות: FGSM ו PGD, הנחשבות לכלים חזקים ונפוצים בתחום. כמו כן, הדיון יתבסס רק על מודלים מסוג Convolutional Neural Network (CNN) [7] , שהם מהמודלים המרכזיים והמוצלחים ביותר בבעיות סיווג בתחום הראייה הממוחשבת.

המאמר אינו מתמקד בחישובים פורמליים של השיטות הנבחנות, אלא מתמקד בהצגת תוצאות מעשיות עבור המקרים שנבדקו. המטרה היא להציג תוצאות מקיפות וברורות, המסודרות באופן שמאפשר להסיק מסקנות ולגבש תובנות רלוונטיות לגבי מצבים אמיתיים בעולם.

בנוסף, המאמר אינו עוסק בחישוב העלויות החישוביות הנדרשות ליישום השינויים הארכיטקטוניים השונים, מתוך מיקוד בהשגת תובנות מעשיות והשפעה על העמידות של המודלים לתקיפות.

גישה זו נועדה להבטיח שהמיקוד המרכזי יהיה בשיפור העמידות של מודלים מסוג CNN לבעיות סיווג, תוך בחינה מעמיקה של התוצאות המעשיות והשלכותיהן.

מגבלות המחקר

במחקר זה זוהו מספר מגבלות משמעותיות שיש להתייחס אליהן:

1. **מגבלות משאבי עיבוד וזמן:**
אחת המגבלות העיקריות במחקר הייתה מגבלת משאבי העיבוד, שיצרה מתח בין הרצון לייצר סט ניסויים אפקטיבי ומקיף לבין הצורך לסיים אותם בזמן סביר. הבחירה ב-CIFAR-10 נבעה מיכולת המסד לאזן בין קצב הבדיקות והניסויים הנדרשים לבין השאיפה לבדוק מקרים קרובים למציאות. עם זאת, בעולם האמיתי, כמויות המידע הנתונות הן לרוב גדולות משמעותית, ולכן המחקר אינו מייצג באופן מלא מצבים הכוללים מודלים ומסדי נתונים עצומים.

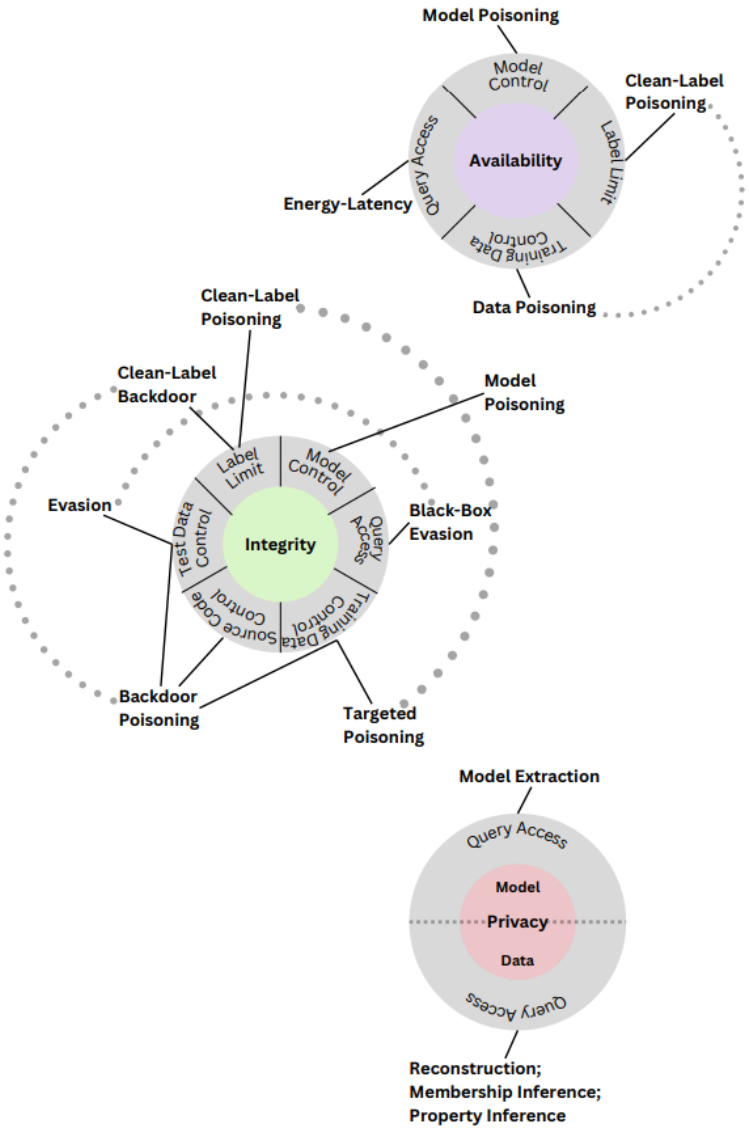
2. **מגבלות בבחירת סוגי התקיפות:**
המחקר התמקד בשתי שיטות תקיפה עיקריות בלבד: FGSM ו-PGD, ולא בדק שיטות נוספות כמו תקיפות מבוססות query. למרות שהתקיפות שנבחרו נחשבות לכלים חזקים ונפוצים, המחקר מוגבל במידת היכולת שלו לייצג את כלל סוגי התקיפות האפשריות.

3. **בחינת עוצמת התקיפות:**
עבור כל שיטת תקיפה, התמקדנו בבדיקת תוצאות על פני מרחב עוצמת התקיפה. ב-FGSM נבדקו ערכי אפסילון שונים בטווח יחסית גדול, וב-PGD נבחנו מספר איטרציות, כדי לכסות מגוון של עוצמות תקיפה, החל מחלשה ועד חזקה. עם זאת, בחינה זו מוגבלת למשתנים מסוימים ולא כוללת את כל הפרמטרים האפשריים בכל שיטה.

4. **מגבלה בבחינת תקיפות מסוג BlackBox:**
במחקר זה לא עסקנו בהרחבה בתקיפות מסוג BlackBox, אלא הצגנו טעימה בלבד ליכולות הפוטנציאליות של עמידות לתקיפות מסוג זה. תקיפות BlackBox נחשבות כמייצגות טוב יותר של העולם האמיתי, שבהם לתוקף אין גישה ישירה למודל היעד.
הבדיקות שנערכו במסגרת המחקר התמקדו בתופעת ה-Transferability ובוצעו באופן מדגמי בלבד. עם זאת, התוצאות המוצגות עשויות לשמש בסיס מצוין למחקרי המשך, שיחקרו לעומק את עמידות המודלים בפני תקיפות מסוג זה ויתמקדו בתרחישים רחבים יותר.

סקירת ספרות

עולם הבעיות הקשורות לתוקפים המנסים לפגוע באמינות או בכשירות של מערכות, ובפרט במערכות מבוססות למידת מכונה (ML), הוא רחב ומורכב מאוד. קיים ספקטרום רחב של סוגי תקיפות, ובעקבותיהם פורסמו בשנים האחרונות אינספור מאמרים ושיטות להגנה. בדו"ח האחרון של המכון הלאומי לתקנים וטכנולוגיה של ארה"ב (NIST - National Institute of Standards and Technology) מוצגת סקירה מקיפה של כלל סוגי התקיפות וההגנות.[8]



תרשים 1: סיווג של סוגי תקיפות על מערכות מבוססות AI.
מקור: NIST ***"Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations,"*** **AI 100-2e2023, 2024**

בתרשים 1 מוצגת דיאגרמה המספקת מבט על סוגי התקיפות השונות והדרכים שבהן הן מנצלות חולשות שונות, לצד המיטיגציות המיושמות להתמודדות עמן. מדובר במעין מרדף מתמשך של "חתול ועכבר", כאשר הנושא ממשיך להוות אתגר פתוח למחקרים. במאמר זה אנו מתמקדים במניעת פגיעה באמינות המודל(Integrity) , תוך התמקדות ייחודית בתחום ה -vision בלבד. בתחום זה התוקף מבצע מניפולציה על הקלט עצמו ובכך מוביל לסיווג לא נכון של המודל תוך שהוא בלתי ניתן לזיהוי לעין אנושית.

במאמר "Intriguing Properties of Neural Networks (Szegedy et al., 2014) " ישנה התייחסות למושג Blind Spots ברשתות נוירונים. תופעה זו מתרחשת לעיתים באזורים במרחב הקלט שלא נצפו כלל במהלך האימון או באזורים שבהם הנתונים אינם צפופים, מה שעלול לגרום למודל לספק סיווג שגוי. בנוסף, שילוב מאפייני השכבה האחרונה ב- Softmax ו-Cross-Entropy לייצוג מרחב הקלט באמצעות התפלגות הסתברויות מגביר את הרגישות. רגישות זו מתבטאת בכך שגם שינויים קטנים בקלט עשויים לגרום למודל לתת סיווג שגוי.

במאמר "Towards Deep Learning Models Resistant to Adversarial Attacks (Madry et al., 2019)" [4] מוצג תהליך ליצירת Adversarial Examples באמצעות שיטת תקיפה PGD, שיושמה על מסדי הנתונים MNIST ו-CIFAR-10. המסקנה המרכזית של המחקר היא כי באמצעות אימון המבוסס על דוגמאות שעברו מניפולציה על ידי התקפות, ניתן לפתח מודלים עמידים יותר. התוצאות מראות הצלחה משמעותית על MNIST , בעוד שב-CIFAR-10 ההשפעה הייתה פחותה אך עדיין ניכר שיפור בביצועים.

המאמר, כמו מאמרים נוספים העוסקים בשינוי או הוספת קלט שעבר מניפולציה או הרעשה, מתמקד בהתמודדות עם Blind Spots כפי שהוזכר קודם לכן.

במאמר On Adaptive Attacks to Adversarial Example Defenses (Tramèr et al., 2020) מוצגת עבודה מקיפה לבדיקת 13 שיטות הגנה מייצגות. עבור כל סוג הגנה, בוצע רצף של תקיפות מתגברות, שמטרתן לזהות את נקודות החולשה של ההגנה ולהבין האם היא מייצרת מודל עמיד באמת או שמא היא רק מבצעת מניפולציה שמקשה על סוג תקיפה מסוימת בלבד.

לדוגמא "החבאת גרדיאנט", עשויה להיות יעילה כנגד סוג תקיפה מסוימת, אך כושלת מול סוגי תקיפות אחרות. אחת הטענות המרכזיות של כותבי המאמר היא שההגנות המוצגות כיום אינן מציגות תוצאות עבור מגוון מספק של תקיפות ואינן עומדות במתודולוגיה סדורה. בנוסף, כותבי המאמר מדגישים את הצורך בכך שמחקרים עתידיים, העוסקים ביצירת מודלים עמידים, יכללו גם הצגה של נקודות החולשה ואסטרטגיות להתמודדות עמן.

במחקרים רבים שסקרנו, הבחַנו כי השיטות או הכלים המוצגים בהם מתיימרים לייצר הגנה רחבה מפני מגוון תקיפות. עם זאת, מנגד, קיימים מאמרים המערערים על עמידותן של שיטות אלו ומצביעים על פרצות בהגנתן.

במחקר זה הצבנו לעצמנו יעד ממוקד אך שאפתני – פיתוח שיטה פשוטה להשגת עמידות בפני תקיפות מסוג WhiteBox, תוך התמקדות בתקיפות המנצלות את הגרדיאנט של המודל. השיטה שאנו מציעים אינה עושה שימוש במניפולציה על הקלט עצמו ואינה מצריכה הוספת דוגמאות חדשות לאימון. הפשטות שלה היא עוצמתה, שכן ניתן לשלב אותה בקלות בכל פיתוח של מודל, תוך שמירה על עלות נמוכה וללא צורך בידע מומחה או בכלי פיתוח מתקדמים.

השיטה מבוססת על ביצוע מניפולציה בשתי השכבות האחרונות בתהליך הלמידה: השכבה האחרונה של המודל עצמו (Output Layer), שכבת ה Label, והתאמת פונקציית ה-Loss.

תהליך פיתוח השיטה המוצעת נשען על מגוון רחב של מאמרים המתארים את עולם הבעיה, תוך הימנעות מכוונת מהסתמכות על מאמרים המתארים את מרחב הפתרונות, שכן רובם מתמקדים במניפולציות על הקלט ואימון עליו [4], הרעשת שכבות במודל ובשיטות לזיהוי תקיפות[9] ועוד מגוון רחב של שיטות הגנה.

מתודולוגיה

במחקר התמקדנו בהשפעת תקיפה מסוג White-Box , כאשר בקריאת הספרות שמנו דגש על מאמרים מוקדמים המציגים את עולם הבעיה. השתדלנו להימנע מהתמקדות במאמרים העוסקים במרחב הפתרונות, משתי סיבות עיקריות:

1. זיהינו שרוב המאמרים מתמקדים בלימוד על קלטים שעברו מניפולציה, גישה המנוגדת ליעד שהצבנו לעצמנו במחקר.
2. רצינו לזהות ולמקד את סט הרכיבים המאפשרים לתקיפה מסוג FGSM להתקיים, מבלי להכניס הטיות מוקדמות הנובעות מפתרונות קיימים.

הגישה המחקרית שנקטנו התבססה על קילוף הבעיה בשיטת "שכבות הבצל", מתוך תקווה לזהות את האזורים או הרכיבים במודל שאחראים ליצירת התופעה. מעבר לכך, חקרנו את השפעתם של פרמטרים שונים ואת האינטראקציות ביניהם, במטרה למצוא דרכים להתמודד עם הבעיה, או לפחות להקטין אותה בצורה משמעותית תוך שמירה על פשטות הפתרון.

תהליך המחקר

שלב ראשון: חקר ראשוני

בשלב זה עסקנו בעיקר בלמידת מאפייני מודלי CNN בבעיות סיווג בתחום ה-Vision, תוך התמקדות בשני מסדי נתונים עיקריים: MNIST ו-CIFAR-10. השלב כלל ניסויים בנוגע למרכיבי המודל, כמו שכבות שונות, פונקציות אקטיבציה, Optimizer ,ועוד. במטרה לקבל הבנה בסיסית של התנהגות המודלים.

שלב שני: זיהוי השפעת השכבות האחרונות

בשלב זה הבנו כי השכבות האחרונות של המודל, ובפרט ייצוג המידע בשכבת התיוג, עשויות להיות קריטיות לשיפור עמידות המודל. ביצענו ניסויים שכללו מניפולציות שונות על שכבת התיוג כמו: Smoothing ,Image as Class ,Pseudo Label. המטרה העיקרית הייתה יצירת חיבוך ולמידה עם שינויים ושיטות קיימות ובאלה שבדקנו שיתכן ויכולות לעזור לנו להבין טוב יותר מהו המאפיין הספציפי שגורם לתופעה להתקיים בצורה רחבה. במהלך שלב זה נשארנו בתצורה "רגילה" של מודלים שמתבססים על מבנה שכבה אחרונה סטנדרטי עם Softmax ופונקציית Loss מתאימה.

שלב שלישי: מעבר למסדי נתונים מורכבים יותר

בשלב זה זנחנו את השימוש ב-MNIST, שהיווה מסד נתונים לשלב החיבוך הראשוני, ועברנו למסד הנתונים INTEL, המשקף את המציאות בצורה טובה יותר ברמת המורכבות והרזולוציה. בנוסף, בשלב זה הנחנו שיש דרך לבנות ייצוג מורכב יותר של התמונות.

- פיצול כל מחלקה למספר מיקומים בשכבת התיוג במטרה ליצור גבולות עמידים יותר.
- ניסויים על קבוצת מדגם קטנה וקבלת שיפור משמעותי כבר בפיצול מאוד קטן
- לאחר מכן, התחלנו באימון מלא על מסדי הנתונים CIFAR-10 ו-INTEL, תוך הצגת שיפור ביצועים משמעותי בעלות נמוכה.

קפיצת מדרגה ראשונה בשיפור ביצועים לעמידות מודלים לתקיפות WhiteBox

שלב רביעי: מעבר לעולם בעיה MultiClass בלתי תלויים

בניסיון לאפשר למודל גמישות בתהליך הלמידה אל מול ה MultiClass עברנו לסט ניבויים ובדיקות בבעיות של מודלים שמבוססים על תהליך שכל מחלקה יכולה להיות משויכת לכל אחד מהמיקומים בשכבת התיוג באופן בלתי תלוי. והוספת תהליך ה LabelDropOut שמאפשר שיפור נוסף בעמידות של מודלים.

קפיצת מדרגה שניה בשיפור ביצועים לעמידות מודלים לתקיפות WhiteBox

שלב חמישי: הרחבת הניסויים באופן סדור והסקת מסקנות

- הסדרת הניסויים למספר מודלים שונים, כולל הוספת מודלים מבוססי Transfer Learning
- שילוב מסד נתונים נוסף ומורכב אף יותר לסט הבדיקות – FOOD-101
- הוספת סוג תקיפות נוספות מסוג WhiteBox : PGD
- הדגמת חוסר האפקטיביות של השיטה אל מול שימוש בתקיפה מבוססת Transferability והדגמה של כיווני מחקר נוספים
- סיכום וניתוח של תוצאות הניסויים

המטרה של השלב החמישי לנסות להבין האם התוצאות שהבחנו בתהליך הבדיקות המדגמיות שביצענו אכן עומדות במבחנים מורכבים יותר וסדורים במטרה לזהות טעויות ולאפשר בחינה טובה שניתנת להצגה ולבחינת עמיתים.

הצגת השיטה – ממצאים ותוצאות

מטרת פרק זה היא להציג באופן מקיף את הנחות העבודה, התוצאות והמסקנות של שתי שיטות שנועדו לשיפור עמידותם של מודלים בפני תקיפות ומניפולציות על הקלט. הפרק נכתב כך שהוא עומד בפני עצמו, ומאפשר לקורא להבין את בסיס התוצאות והמסקנות של המחקר כולו.

מבוא והגדרת האתגר

המחקר שלנו מתמקד בהתמודדות עם תקיפה חזקה מסוג WhiteBox, שבה לתוקף יש גישה מלאה לכל המידע על הרשת: הארכיטקטורה, המשקולות, פונקציות האקטיבציה, שכבות רגולציה כמו dropoutועוד. מצב זה מאפשר לתוקף לבצע מניפולציות על הקלט באופן שממקסם את יכולתו לפגוע בביצועי המודל בצורה יעילה.

במסגרת המחקר, הגדרנו מספר מטרות מרכזיות:

1. פיתוח שיטת הגנה המספקת עמידות למודלים מבלי להוסיף רכיבי קלט נוספים או לבצע מניפולציה ישירה על התמונות .בלומר, שיטות המבוססות על שינוי פיקסלים של התמונות אינן רלוונטיות עבורנו.
2. הבטחת שמירה על ביצועי המודל. מטרה זו מהווה אתגר משמעותי ברוב שיטות ההגנה המוכרות כיום, אשר פעמים רבות פוגעות בביצועים הכלליים של המודל.

בדרך כלל, תקיפת WhiteBox מתייחסת למקרה שבו התוקף מבצע תקיפה ישירות על המודל ללא מניפולציות חיצוניות נוספות. עם זאת, אנו שואפים להרחיב את ההגדרה של תקיפת WhiteBox ולהציג שיטה המספקת הגנה גם מפני מודלים "מקורבים" למודל המקורי.

כאשר אנו מתייחסים למונח "מודל מקורב", אנו מתכוונים לגלגולים שונים של המודל – גרסאות היסטוריות (מודלים שנשמרו בשלבים מוקדמים יותר של האימון) וכן גרסאות עתידיות של המודל, במידה והתוקף יחליט לאמן את המודל מעבר לנקודה הנוכחית שלו.

הבחנות ומוטיבציה

המשכנו במספר ניסויים מקדימים שמטרתם להמחיש את עוצמת התקיפה על מודלים שונים ומגוונים. במסגרת הניסויים השתמשנו בשני סוגי תקיפות מרכזיים FGSM ו-PGD. בנוסף, נעשה שימוש במגוון מסדי נתונים, כדי להדגיש את האתגר שבהתמודדות מול תקיפת WhiteBox . מסדי הנתונים בהם השתמשנו CIFAR-10 ו-INTEL.

כדי להמחיש עוד יותר את עוצמת התקיפה, ביצענו ניסויים נוספים על מודלים שאומנו מראש על ImageNet, תוך שימוש בטכניקת Transfer Learning, והותאמו למסדי הנתונים FOOD-101 ו-INTEL.

התוצאות שיוצגו הם עבור המודלים על בסיס הארכיטקטורות שמוצגות בטבלה הבא:

Model 1 CIFAR10	Model 2 CIFAR10	Model 1 INTEL	Model 2 INTEL TransferLearning	Model 2 FOOD101 TransferLearning
Conv2D 3X3 32 relu	Conv2D 3X3 64 relu	Conv2D 3X3 32 relu	Inception -2 Top	Inception -2 Top
MaxPool 2X2	Conv2D 3X3 64 relu	MaxPool 2X2	Dense 128 relu	Dense 128 relu
Conv2D 3X3 64 relu	MaxPool 2X2	Conv2D 3X3 64 relu	Dense 10 softmax	Dense 10 softmax
MaxPool 2X2	Conv2D 3X3 128 relu	MaxPool 2X2		
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu		
MaxPool 2X2	MaxPool 2X2	MaxPool 2X2		
Conv2D 3X3 256 relu	Dense 256 relu	Conv2D 3X3 256 relu		
MaxPool 2X2	Dense 10 softmax	MaxPool 2X2		
Dense 512 relu		Dense 512 relu		
Dense 128 relu		Dense 128 relu		
Dense 10 softmax		Dense 10 softmax		

פרטי הניסוי והגדרות התקיפה

מספר ריצות ונתוני אימון:

התוצאות המוצגות מבוססות על 100 ריצות לכל מודל. עבור כל אחד ממסדי הנתונים שנבדקו, השתמשנו במספר דוגמאות שונה לאימון:

- CIFAR-10 -40,000 דוגמאות עבור 10 מחלקות שונות.
- INTEL – 11,228 דוגמאות עבור 6 מחלקות שונות.
- FOOD-101 – 3,510 דוגמאות עבור 6 מחלקות שונות.

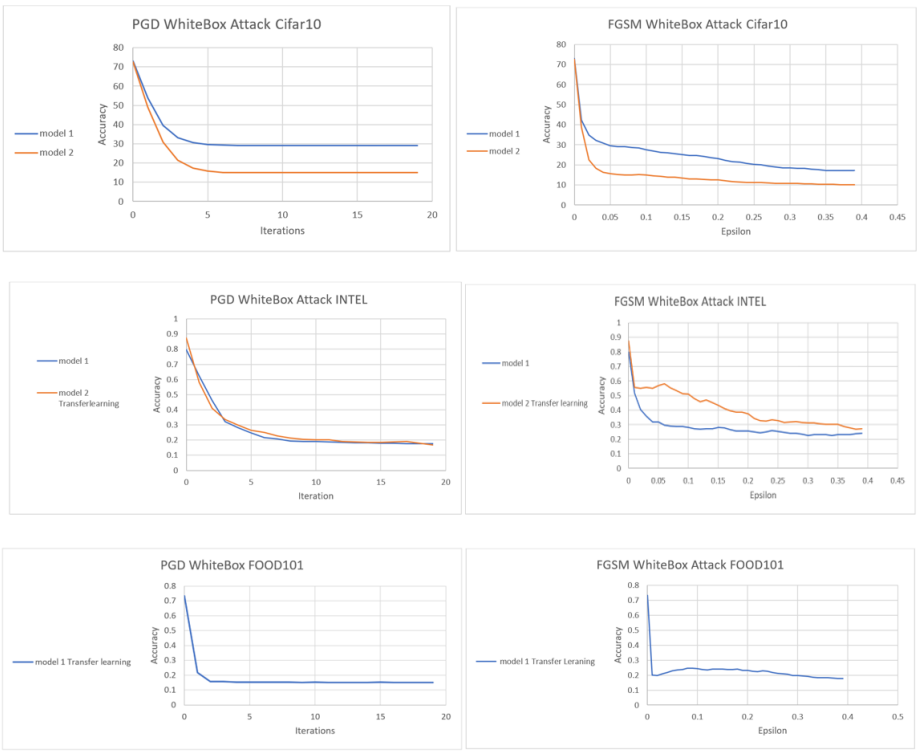
שימוש ב- Transfer Learning:

עבור מסדי הנתונים FOOD-101 ו-INTEL הוספנו שימוש בטכניקת Transfer Learning, בהתבסס על משקולות ומבנה של מודל שאומן על ImageNet בשם InceptionV3.

- במהלך השימוש ב-InceptionV3, הגדרנו את כל המשקולות בגוף המודל כלא ניתנים לאימון.
- אימנו רק את השכבות האחרונות, שהותאמו מחדש למטרות האימון הייחודיות לכל מסד נתונים.

הגדרות התקיפה:

- **תקיפת FGSM**
  בוצעה בטווח אפסילון בין 0 עד 0.4, בצעדים של 0.01
- **תקיפת PGD**
  בוצעה עם עד 20 איטרציות. בכל איטרציה, אפשרנו לתקיפה גודל צעד של 0.005 אפסילון. כאשר הוגדר חסם של אפסילון 0.3, כדי לאפשר לתקיפה גמישות מירבית.



תרשים 2 :הצגת תוצאות של עמידות המודלים לתקיפות FGSM ו PGD על מסדי הנתונים CIFAR-10 ו-INTEL בתצורות שונות. הצגת החולשה של המודלים לתקיפות כבר באפסילון קטן מאוד.

ניתוח תוצאות תרשים 2: השפעת תקיפות FGSM ו-PGD על ביצועי המודלים

בתרשים 2 ניתן לראות כי תקיפות FGSM ו-PGD גורמות לפגיעה משמעותית בביצועי המודלים, ופוגעות ביכולתם לזהות את המחלקות השונות עליהן אומנו.

ממצאים עיקריים:

- **פגיעה בביצועים כבר באפסילון קטן מאוד -** ניתן לראות ירידה חדה בדיוק כבר עבור ערכי אפסילון קטנים.

- **באפסילון גדול התייצבות בטווח דיוק של 20% עד 30% -** לאחר הירידה הראשונית, הדיוק נשאר יציב יחסית בטווח זה כאשר האפסילון גבוה.

- **תוצאות דומות גם עבור מודלים שהתבססו על Transfer Learning** - גם במודלים שעברו אימון עם InceptionV3 באמצעות Transfer Learning, ההשפעה של התקיפה נותרה משמעותית, והתוצאות אינן שונות מהותית ממודלים שאומנו מאפס.

משמעות הממצאים ומסקנות

- כלל הדוגמאות ממחישות את עוצמת התקיפה מסוג WhiteBox, אשר פוגעת במודלים על פני מגוון מסדי נתונים, ארכיטקטורות ופרמטרים שונים.
- זיהינו כי ללא למידה על דוגמאות שעברו מניפולציה(Adversarial Learning) , קשה מאוד ליצור ארכיטקטורה עמידה לתקיפות FGSM ו-PGD.

השיטה המוצעת – LabelDropOut

האתגר המרכזי והפתרון המוצע

במהלך המחקר, נתקלנו באתגר של אילוץ המודל לשנות את אופן מיפוי המרחב במהלך האימון ,כך שהמבנה הארכיטקטוני והמשקולות ימנעו את היתכנות התקיפה או לפחות יקשו עליה משמעותית.

דרך אחת לשינוי המיפוי היא הוספת דוגמאות סינתטיות ,כולל כאלה שעברו מניפולציות באמצעות FGSM או PGD, אך לא רק. עם זאת, הצבנו לעצמנו יעד לפתח מודל עמיד מבלי להסתמך על למידה מבוססת דוגמאות חדשות.

לבן, הנחנו שאם נגדיר מחדש את שייכותה של כל תמונה למספר מופעים בשכבת התיוג, נוכל לכפות על המודל לייצר גבולות חדשים, שלא היו קיימים קודם לכן, או לכל הפחות לשנות את תצורת הגבולות הקיימים.

תיוג מבוסס MultiClassPosition

בבעיה סטנדרטית עם 10 מחלקות ,האימון מתבצע על 10 מיקומים ייחודיים בשכבת התיוג. כעת, נפריד כל מחלקה למספר מיקומים שונים בהתאם למסד הנתונים ולבעיית הסיווג.

במקום שכבת תיוג בגודל 10, נגדיל אותה ל-1,000 מיקומים, כאשר כל קבוצה של מיקומים מייצגת מחלקה אחת .לצורך נוחות, קבענו שכל מיקום שהאינדקס שלו מודולו 10 תואם למחלקה המתאימה:

- **מחלקה 0** תשויך למיקומים 0, 10 ,20, 30...
- **מחלקה 1** תשויך למיקומים 1, 11 , 21, 31...
- **מחלקה 2** תשויך למיקומים 2, 12 , 22, 32...
- **וכן הלאה...**

שיטת החיזוי החדשה

בשלב החיזוי, ניתן לזהות את המחלקה בשתי דרכים:

1. מציאת המיקום בעל הערך הגבוה ביותר והגדרת המחלקה בהתאם.
2. חישוב סכום כל המיקומים שמייצגים את אותה מחלקה 'כאשר המחלקה עם הסכום הגבוה ביותר היא המחלקה הסופית שנבחרה.

LabelDropOut – הכנסת DropOut לשכבת התיוג

בשלב השני, נשתמש ב-DropOut על שכבת התיוג, כך שלפני תחילת האימון, חלק מהמיקומים יתאפסו באופן רנדומלי. התהליך יתבצע באופן מאוזן לאורך כל מאגר התמונות.

ניתן לבחור כל שיטת חלוקה למיקומים המייצגים מחלקות, כל עוד נשמרת הדרישה כי במיקומים שלא שייכים למחלקה אין ערכים שונים מ-0.

כדי לאפשר איפוס רכיבים בשכבת התיוג ,נשתמש בשכבת Dense עם אקטיבציית Sigmoid, לצד פונקציית Loss בינארית (Binary Cross-Entropy).

השוואת ביצועי המודלים – שימוש ב LabelDropOut

התוצאות הבאות הם עבור הם בסיס מודלים כפי הארכיטקטורה שמוצגות בטבלה הבאה:

LabelDropOut CIFAR-10	LabelDropOut INTEL	LabelDropOut INTEL Transfer Learning	LabelDropOut FOOD-101 Transfer Learning
Conv2D 3X3 32 relu	Conv2D 3X3 32 relu	Inception -2 Top	Inception -2 Top
Conv2D 3X3 32 relu	Conv2D 3X3 32 relu	Dense 128 relu	Dense 128 relu
MaxPool 2X2	MaxPool 2X2	Dense Sigmoid 600	Dense Sigmoid 600
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu		
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu		
MaxPool 2X2	MaxPool 2X2		
Dense Sigmoid 500	Dense Sigmoid 600		

מטרת ההשוואה היא לבחון את השפעת גודל שכבת התיוג וה-DropOut על עמידות המודלים לתקיפות WhiteBox וכן לבדוק הבדלים בין מודלים סטנדרטיים למודלים מבוססי Transfer Learning.

בכדי לבצע השוואה מקיפה, יצרנו מודלים מבוססי הארכיטקטורה כפי שמוצג בטבלה.
כל מודל עבר 100 איטרציות של אימון, תוך שימוש באותן תמונות בשלבי האימון, כך שהתנאים יהיו זהים לכל המודלים.

CIFAR-10

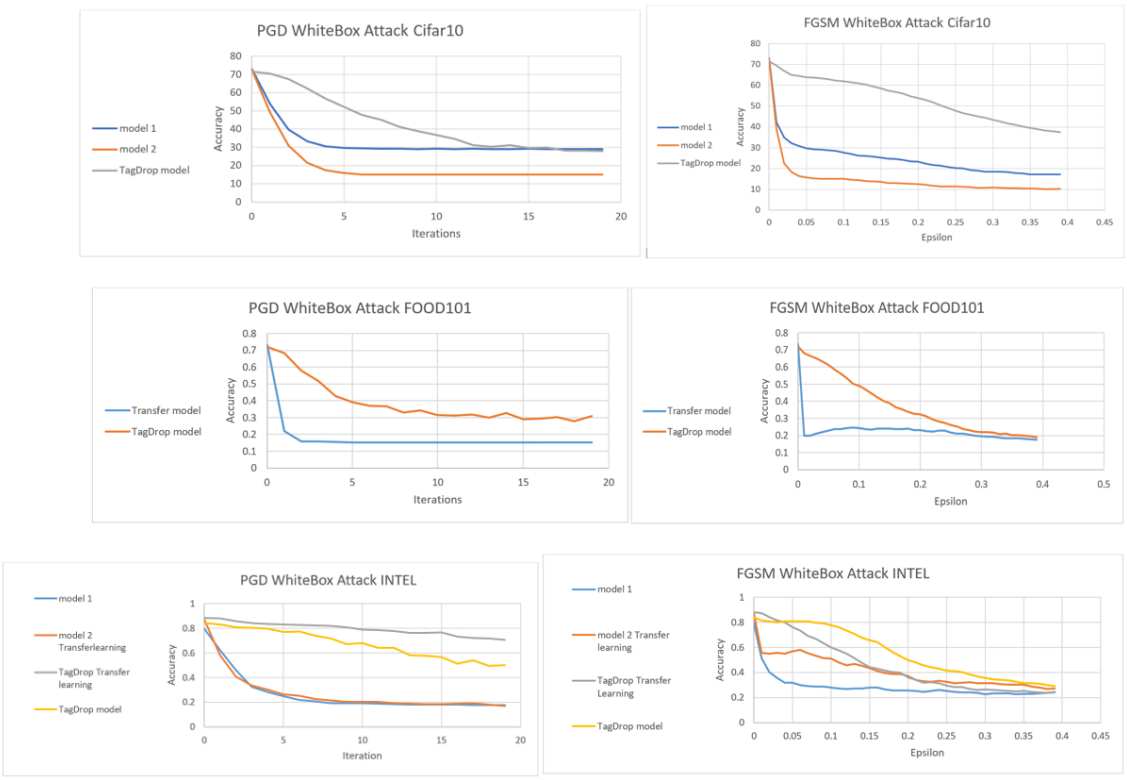
- שכבת תיוג בגודל 500 מיקומים
- ביצוע DropOut של 15 מיקומים לכל תמונה שכבת התיוג

INTEL

- שכבת תיוג בגודל 600 מיקומים
- ביצוע DropOut של 60 מיקומים לכל תמונה שכבת התיוג

FOOD-101

- שכבת תיוג בגודל 600 מיקומים
- ביצוע DropOut של 30 מיקומים לכל תמונה שכבת התיוג



תרשים 3 :השוואת התוצאות של מודלים עליהם הפעלנו את השיטה המוצעת של LabelDropOut כפי שהוצגה אל מול תוצאות המודלים מתרשים 2.

בתרשים 3, אפשר לראות בבירור כי בשיטה שהצענו, אנו מציגים שיפור משמעותי מאוד, ללא פגיעה בביצועי המודלים על תמונות שלא עברו מניפולציה (תמונות מקוריות). השיפור ניכר באופן גורף בכל מסדי הנתונים, הן עבור המודל הבסיסי והן בשימוש ב -InceptionV3 המבוסס על Transfer Learning. העמידות באה לידי ביטוי בצורה מובהקת במיוחד בערכי אפסילון קטנים, הן בתקיפה מבוססת FGSM והן בתקיפה מבוססת PGD.

מסקנות ביניים

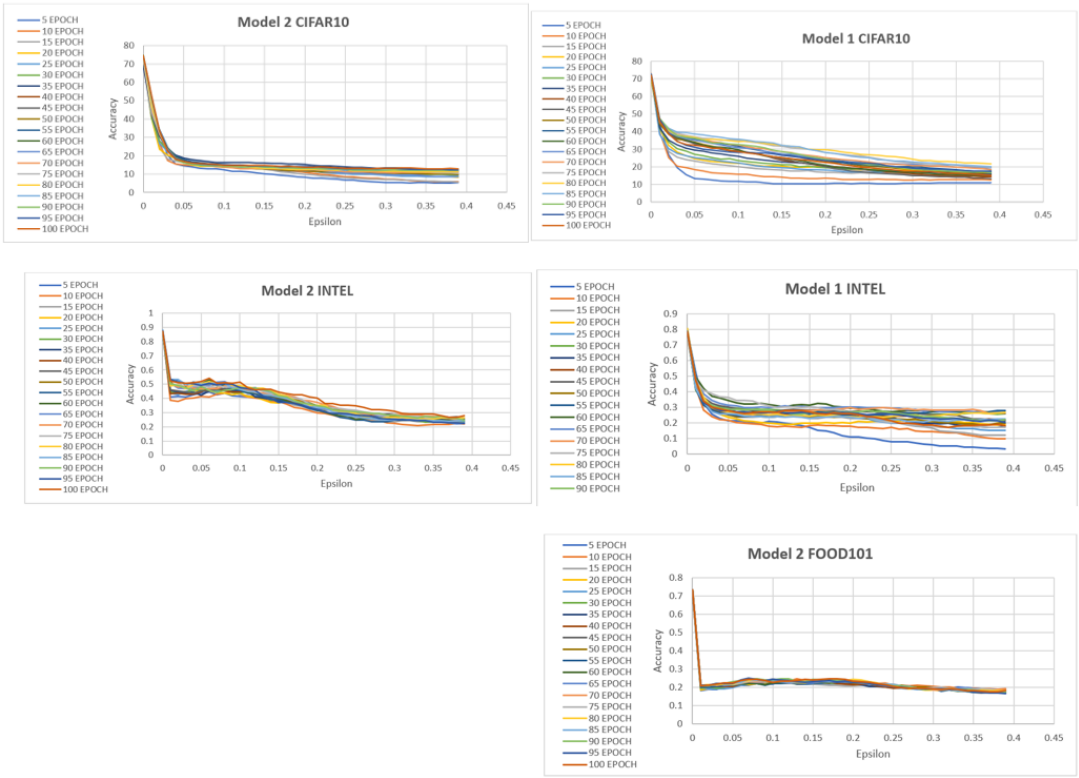
עד כה, הצלחנו להציג שיטה פשוטה ויעילה לשיפור משמעותי בעמידות המודלים, ללא צורך ב -Adversarial Training או במניפולציה של התמונות עצמן. הראינו כי שילוב LabelDropOut בשכבת התיוג לא רק משפר את העמידות לתקיפות מסוג WhiteBox, אלא גם במעט ואינו פוגע בביצועי המודלים ובתוצאותיהם על מסדי נתונים שונים עבור תמונות שלא עברו תקיפה.

בנוסף, הצלחנו להראות שגם בשימוש ב- Transfer Learning על בסיס מודל קיים ומאומן, ניתן לשמר את הביצועים ולהשיג עמידות בפני תקיפות מסוג WhiteBox, זאת באמצעות שימור המבנה של שכבת התיוג, שימוש בפונקציית Loss-Binary ואקטיבציה Sigmoid בשכבת ה- Dense האחרונה. בכך הדגמנו כי ניתן לאמן כל מודל בשכבות התחתונות שלו, ולאחר מכן לשדרג אותו כך שיהפוך עמיד בפני תקיפות מסוג WhiteBox בצורה פשוטה וקלה. **השינוי מאוד קל למימוש.**

השיטה במשפט: שינוי הייצוג של התמונות למספר מיקומים בעלי הערך 1, ואיפוס רנדומלי של ערכים בצורה מאוזנת.

הרחבת הניתוח בביצועי השיטה המוצעת

בחלקים הקודמים הצגנו תוצאות עבור כל מודל באיטרציה ה-100 שלו. כעת, נרצה לבחון לעומק את התפתחות המודלים השונים. כאלה המבוססים על LabelDropOut וכאלה שאינם מבוססים עליו. נתמקד תחילה במודלים שאינם עושים שימוש ב-LabelDropOut. ביצענו אימון למודלים תוך תיעוד הביצועים בכל איטרציה חמישית רק עבור תקיפה מבוססת FGSM.

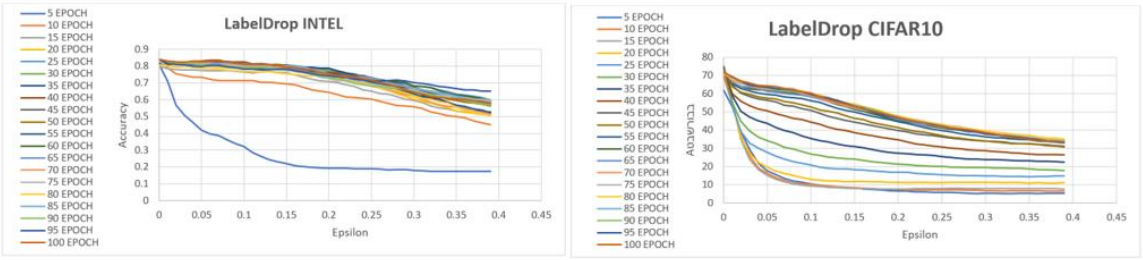


תרשים 4 :תיעוד התפתחות בשלבי האימון של עמידות המודלים סטנדרטים בכל איטרציה חמישית לתקיפת FGSM.

בתרשים 4, רואים בבירור כי גם בשלבי ההתפתחות של המודלים השונים, הביצועים בהיבט העמידות לתקיפות מסוג WhiteBox אינם מספקים. ניכר כי כלל המודלים מושפעים באופן משמעותי מתקיפה כבר באפסילון קטן בטווח של 0.01–0.02.

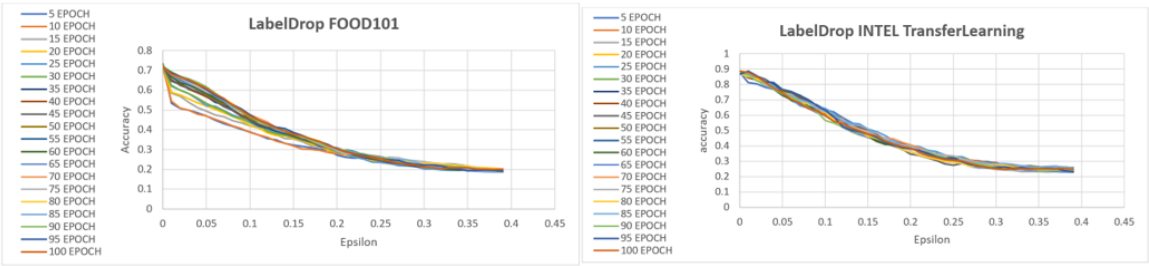
במודל Model CIFAR-10, ניתן להבחין בשיפור מסוים בין קבוצות האיטרציות השונות, אך הפגיעה בביצועי המודל בתקיפה באפסילון קטן עדיין מאוד משמעותיים.

על מנת לבצע השוואה מתאימה ביצענו תיעוד זהה בדגימה ושמירת כל איטרציה חמישית עבור המודלים שבנויים תחת הארכיטקטורה והשיטה המוצעת LabelDropOut.



תרשים 5 :תיעוד התפתחות בשלבי האימון של עמידות המודלים מבוססי LabelDropOut בכל איטרציה חמישית לתקיפת FGSM במיקוד CIFAR-10 ו INTEL.

בתרשים 5, אפשר לראות שעבור שני מסדי הנתונים CIFAR-10 ו-INTEL שנבנו תחת מודלים סטנדרטים ישנו שיפור ביצועים תוך כדי שלבי האימון. כאשר ניכר שעבור INTEL כבר לאחר האיטרציה החמישית ישנה קפיצת ביצועים בעמידות לתקיפה FGSM. עבור CIFAR-10 ניתן לראות שהעמידות מתפתחת בצורה עקבית בין כל חמש איטרציות. עבור שני הגרפים ניתן לראות שככל מתקדמים בשלבי האימון הביצועים מתייצבים ללא שיפור בין דגימה לדגימה בעמידות המודלים.



תרשים 6 :תיעוד התפתחות בשלבי האימון של עמידות המודלים מבוססי LabelDropOut בכל איטרציה חמישית לתקיפת FGSM במיקוד INTEL ו – FOOD-101. מודלים על בסיס InceptionV3 תוך הקפאת משקולות.

בתרשים 6, ניתן לראות שעבור מודלים שהם מבוססים על InceptionV3 עם הקפאת המשקולות אנחנו רואים שמצד אחד כבר באיטרציה חמישית יש שיפור ביצועים בעמידות לתקיפה מסוג whiteBox, אך מצד שני אין התפתחות בעמידות המודלים ככל שהאימון מתקדם. חשוב לציין, בתהליך האימון לא אפשרנו אימון של השכבות הפנימיות כלל. האימון כולו התבסס על הלמידה של המודלים בשכבות הפנימיות על ImageNet ובאימון בשכבה האחרונה אל מול CIFAR-10 ו- FOOD-101.

סיכום ביניים

עד כה הצגנו שיטה חדשה ומאוד פשוטה בשם: LabelDropOut. על פניו הצגנו עמידות והגנה למודלי CNN כנגד תקיפות מסוג WhiteBox שמנצלות את היכולת של התוקף לדעת מה הגרדיאנט ולייצר תקיפה מאוד ממוקדת וחזקה.

תחת סוג התקיפות הללו הראינו כי לשיטה יש השפעה רחבה על סוגים שונים של ארכיטקטורת מודלים, כולל מודלים המתבססים באופן מלא על Transfer Learning. בנוסף, הדגמנו כי היא חוצה-תחומי בעיה באמצעות יישום בשלושה מאגרי מידע שונים לבעיית סיווג : INTEL, CIFAR10 ו-FOOD101.

הראינו שהשיטה אפקטיבית ביצירת עמידות לא רק מול תקיפה מבוססת FGSM, אלא גם מול תקיפה מסוג PDG. יתרה מזאת, הראינו כי שימוש ב- LabelDropOut אינו פוגע בביצועי המודל בזיהוי ובסיווג של תמונות שלא עברו תקיפה כלל.

למעשה, הראינו כי השיטה המוצעת מאפשרת שינוי פשוט שבאמצעותו ניתן לבנות מודל בסיס חדש, לחילופין, לבצע התאמות למודלים קיימים מאומנים באמצעות כוונון מחדש של השכבות האחרונות, ללא צורך באימון מחדש של השכבות הפנימיות ובכך לשמר חוזקות שהצלחנו לייצר בשיטות אחרות.

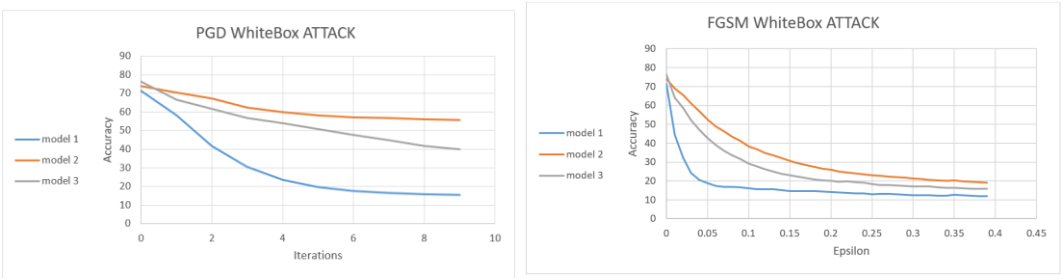
מודל סטנדרטי MultiClassPosition vs LabelDropOut

עד כה הצגנו את LabelDropOut שהיא שילוב MultiPosition כשלב מקדים. בחלק הזה של המחקר אנחנו נראה שמספיק להשתמש ב - MultiPosition על מנת לייצר קפיצת מדרגה משמעותית ביכולת עמידות מתקיפות מסוג WhiteBox, ללא שינוי נוסף או הוספת DropOut לשכבת התיוג. בניסויים מקדימים שבצענו ראינו שמודלים שמבנה התיוג שלהם הוא בתצורה רגילה של מחלקה לפי מיקום ישנה חולשה משמעותית לתקיפות מסוג WhiteBox. בחלק זה של המחקר נציג תוצאות רק עבור מסד הנתונים CIFAR10 עם אותה כמות דוגמאות לאימון ו 100 איטרציות בדומה לפרקים הקודמים במחקר.

מבנה המודלים:

Model 1	Model 2	Model 3
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu
MaxPool 2X2	MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu
MaxPool 2X2	MaxPool 2X2	MaxPool 2X2
FC 256 relu	FC 256 relu	FC 256 relu
FC 256 relu	FC 256 relu	FC 256 relu
FC 10 softmax	FC 500 sigmoid [DropOut]	FC 500 Softmax

מודל 1 הוא מודל הבסיס. כאשר שני המודלים הנוספים הם על בסיס הארכיטקטורה של LabelDropOut ו- MultiPosition בלבד. המטרה היא לתת מענה לשאלה האם אפשר לקחת כל מודל ופשוט להחליף את השכבות האחרונות שלו על בסיס השיטה LabelDropOut ו-MultiPosition בלבד ובכך לייצר להציג שיפור.



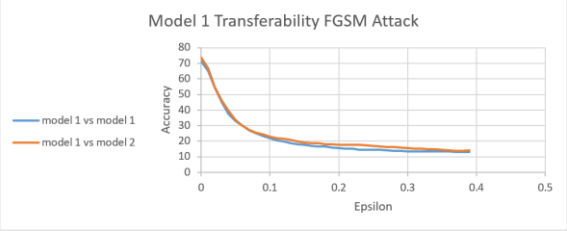
תרשים 7 :תוצאות לתקיפה מסוג FGSM ו PGD מסוג WhiteBox מול מודל סטנדרטי, מודל בשימוש MultiPosition ומודל בשימוש LabelDropOut.

בתרשים 7, ניתן לראות שמספיק לבצע שינוי שהוא מבוסס על MultiPosition על מנת לייצר קפיצת מדרגה בשיפור הביצועים אל מול תקיפה WhiteBox. כאשר מבצעים LabelDropOut שמתבסס כשלב ראשוני על MultiPosition ישנו שיפור בביצועים. על פניו הניסוי הזה מראה איך אפשר לקחת כל מודל בסיסי ורק על ידי החלפת השכבות האחרונות שלו לייצר שיפור בעמידות לתקיפה מבוססת מניפולציה על הקלט. חשוב לציין שאין פגיעה בביצועי המודלים אל מול תמונות שלא עברו מניפולציה.

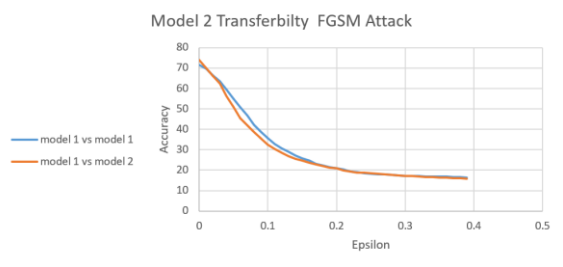
מגבלות השיטה – ניתוח מבוסס Transferability

למרות ש-DropOutLabel מציג שיפור משמעותי בעמידות לתקיפות WhiteBox, בפרק זה נציג ניסויים שמטרתם לבחון את מגבלות השיטה, תוך שימוש בסוג תקיפה מבוססת Transferability. את הבדיקה נבצע עבור המודלים 1 ו-2 שהוצגו בפרק הקודם. בנינו 2 מודלים נוספים בארכיטקטורה של מודלים 1 ו-2 על בסיס אותם פרמטרים ואתם מאפייני אימון. כך אנחנו מדמים תקיפה שבא לתוקף יש הבנה על הארכיטקטורה של המודל, יש את המידע עליו התאמנו, אך אין לו את המשקולות עצמן ובכך אינו יכול לנצל את הגרדיאנט הספציפי. ישנו הבדל יחיד שהוא שהאימון של המודלים התוקפים התבצע על 50 איטרציות.

תקיפה Transferability מבוססת מבנה מודל 1:



תקיפה Transferability מבוססת מבנה מודל 2:



תרשים 8 : תוצאות עבור תקיפה מבוססת Transferability עבור תקיפות מסוג FGSM ו-PGD מול מודל 1 שהוא מודל במבנה רגיל ומודל 2 שהוא מודל בשימוש DropOutLabel עם Sigmoid.

מתרשים 8 ניתן לראות כי אף אחד מהמודלים אינו מציג יתרון עם תקיפות מבוססות Transferability, בין אם מדובר ב-FGSM או ב-PGD. סביר להניח שהדמיון ברכיבי המודל בשכבות הפנימיות מוביל לדמיון גם ביכולת להתמודד עם תקיפות שאינן מבוססות גרדיאנט. עם זאת, לאור העלות הנמוכה של יישום המבנה המוצע המבוסס LabelDropOut, וכן ההבנה כי הוא אינו פוגע בביצועי המודל על תמונות נקיות (ובמקרים מסוימים אף משפר אותם), אין סיבה שלא לשלבו כפתרון בסיסי בכל מודל. שילוב זה יאפשר למקד את המחקרים הבאים בשיטות מתקדמות יותר של תקיפה והגנה שאינן מבוססות על ניצול גרדיאנט.

פוטנציאל בייצוג המידע – יעילות ושיפור ביצועים

במהלך המחקר זיהינו כי בחלק מהניסויים שבצענו על השיטות שמבוססות MultiPosition נרשם שיפור קל בביצועים, בהשוואה למודלים בעלי מבנה וייצוג רגילים.

כמו כן, בניסויים בהם השתמשנו ב-Transfer Learning, מצאנו כי ניתן להשיג עמידות לתקיפות White-Box גם ללא אימון השכבות הפנימיות של המודל. למעשה, בתהליך זה הראנו כי באמצעות שינוי ייצוג המידע שעליו מאומן המודל, ניתן לשפר את העמידות לתקיפת White-Box גם כאשר משקולות המודל כלל לא התעדכנו או הותאמו לבעיה הספציפית שאותה ניסינו לפתור.

הסבר אפשרי לכך הוא שהשכבות הפנימיות של הרשת לומדות תבניות אבסטרקטיות של העולם שעליו הן אומנו. כתוצאה מכך, ניתן להגיע לביצועים טובים גם כאשר המודל אומן על קטגוריות ותמונות השונות מבעיית היעד שאנו מנסים לפתור.

בדרך כלל, בעת אימון מודלים, הגישה הרווחת היא לאמן את המודל כולו בבת אחת. כלומר, נהוג לאמן בו-זמנית הן את השכבות הפנימיות והן את השכבות החיצוניות על אותו המידע, מתוך מטרה להגיע ישירות לביצועיים מיטביים. שיטה זו מתבססת על ההנחה שהמודל ילמד את הייצוגים הפנימיים והחיצוניים באופן משולב ואופטימלי מההתחלה.

במחקר שלנו, ובפרט בפרק הקודם, הראנו כי למרות שהצלחנו להשפיע על השכבות האחרונות של המודל ובכך לשפר את עמידותו לתקיפות White-Box, השכבות הפנימיות לא הושפעו באופן שתורם לשיפור עמידות המודל כולו.

מכאן עולה השאלה: האם ניתן לשפר את ביצועי השכבות הפנימיות בלבד, אך ורק באמצעות שינוי ייצוג המידע ובאופן פשוט?

לדעתנו, התשובה היא כן.
בתוצאות הניסויים הבאים נראה כי באמצעות ייצוג שונה של המידע שעליו מאומן המודל ובאמצעות פירוק תהליך האימון לשלבים, ניתן לשפר את ביצועי המודל כולו.

רקע והסבר לייצוג המוצע:

הגענו להבנה כי ייתכן שניתן לנצל את ייצוג של המידע כדי לפתח שכבות פנימיות המסוגלות ללמוד תבניות מורכבות יותר. השימוש ב MultiPosition בשכבה האחרונה מאפשר מבנה גמיש יותר, ובכך מאפשר למודל ללמוד שילובים מורכבים יותר שלא ניתן ללמוד במבנה רגיל.

כאשר אנו מתמודדים עם בעיה שבה יש 10 מחלקות, ניתן לייצג אותן באמצעות 10 מיקומים, למשל, להקצות 5000 תמונות לכל מיקום. במבנה זה, המודל לומד את כלל השילובים הקיימים בין התמונות שתויגו למיקום מסוים, תוך מקסום היכולת להפריד בינהן לבין המחלקות האחרות.

אך אם המטרה בשלב הראשון של האימון אינה להגיע לדיוק מקסימאלי, אלא קודם כל ללמד את השכבות הפנימיות להבין טוב יותר את עולם הבעיה, ניתן להשתמש בייצוג מורכב יותר.

במקום להגדיר 10 מיקומים בלבד, שכל אחד מהם מייצג מחלקה אחת, נוסיף 45 מיקומים נוספים, שכל אחד מהם מייצג שילוב בין שתי מחלקות שונות. באופן זה, המודל אינו לומד רק כיצד להפריד בין המחלקות, אלא גם לומד מה משותף לכל זוג מחלקות שונות.

כלל המודלים מאומנים על 40,000 דוגמאות סך הכל 100 איטרציות כל אחד. Learning rate של 0.001 וגודל batch של 50.

מבנה המודלים:

Model 0	Model 1	Model 2	Model 3	Model 4	Model 5
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu	Conv2D 3X3 64 relu
MaxPool 2X2	MaxPool 2X2	MaxPool 2X2	MaxPool 2X2	MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu	Conv2D 3X3 128 relu
MaxPool 2X2	MaxPool 2X2	MaxPool 2X2	MaxPool 2X2	MaxPool 2X2	MaxPool 2X2
FC 10 softmax	FC 10 softmax/ FC 10 sigmoid	FC 55 sigmoid/ FC 10 sigmoid	FC 55 sigmoid/ FC 10 softmax	FC 55 Softmax/ FC 10 Softmax	FC 500 sigmoid/ FC 10 sigmoid

אופן אימון המודלים:

- **מודל 0** אומן בשיטה הרגילה במשך 100 איטרציות, כאשר הוקצה בדיוק מיקום אחד לכל מחלקה, סך הכל 10.
- **מודלים 1, 2, 3, ו-4** אומנו במשך 75 איטרציות על שכבה עם 55 מיקומים:
  - 10 מיקומים המייצגים את עשרת המחלקות, עם ערך 1.0.
  - 45 מיקומים נוספים, המייצגים את כל זוגות המחלקות האפשריים, כאשר כל מיקום מתוך ה-45 קיבל ערך 0.5 עבור שתי המחלקות שאותן הוא מייצג.

במודלים שבהם השתמשנו ב Softmax וב Cross-Entropy-Loss, נרמלנו את התיוגים כך שיתאימו לדרישות הפונקציות הללו. לאחר 75 איטרציות, המודלים המשיכו את האימון בתצורה שונה:

- השכבות הפנימיות הוקפאו, כך שלא עברו עדכון נוסף.
- השכבה האחרונה הוחלפה בשכבה חדשה המכילה 10 מיקומים בלבד, בהתאמה למספר המחלקות.

תוצאות:

Model 0	Model 1	Model 2	Model 3	Model 4	Model 5
72.62%	74.7%	79.11%	79.3%	79.26%	71.82%

אפשר לראות שמודלים 2, 3, ו-4 אשר משתמשים בייצוג חדש כפי שתיארנו מספקים את התוצאות הטובות ביותר לסיווג קבוצת הבדיקה.

מבנה המודלים:

Model 0	Model 1
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu
MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu
MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 256 relu	Conv2D 3X3 256 relu
Conv2D 3X3 256 relu	Conv2D 3X3 256 relu
MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 512 relu	Conv2D 3X3 512 relu
Conv2D 3X3 512 relu	Conv2D 3X3 512 relu
MaxPool 2X2	MaxPool 2X2
FC 10 softmax	FC 55 sigmoid/ FC 10 sigmoid

תוצאות:

Model 0	Model 1
76%	83.28%

האם השיטה המוצעת יכולה להפוך לסטנדרט במימוש מודלי CNN ?

ניתן לראות שגם עבור מודל מורכב יותר, השיטה שהוצעה מביאה לשיפור יחסי בעלות נמוכה. הדבר מעלה מספר שאלות:

1. האם השיטה אכן מאפשרת להפיק יותר מכל מודל נתון?
2. אם התשובה חיובית, מדוע לא להפוך את יישומה לחלק אינטגרלי מכל מודל שנבנה, כדי למצות את הפוטנציאל המרבי שלו?

ניתן להביא דוגמה נוספת שמחזקת את ההבנה כי השיטה המוצעת אינה מקרה נקודתי, אלא עשויה להיות בעלת השפעה רחבה ומאפיינת על ביצועי מודלים שונים. אם גם בדוגמה זו מתקבל שיפור עקבי, הדבר מחזק את הסברה שהשיטה אינה תלויה במודל מסוים אלא בעלת השפעה רחבה.

נבחן את התוצאות עבור מסד הנתונים INTEL על בסיס מבנה המודלים הבא:

Model 0	Model 1
Conv2D 3X3 32 relu	Conv2D 3X3 32 relu
MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 64 relu	Conv2D 3X3 64 relu
MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 128 relu	Conv2D 3X3 128 relu
MaxPool 2X2	MaxPool 2X2
Conv2D 3X3 256 relu	Conv2D 3X3 256 relu
MaxPool 2X2	MaxPool 2X2
FC 10 softmax	FC 21 sigmoid / FC 10 sigmoid

תוצאות:

Model 0	Model 1
78.95	85%

אפשר לראות שגם עבור מסד הנתונים INTEL, השיטה המוצעת מובילה לשיפור משמעותי, וזאת ללא מורכבות רבה בשלב המימוש. ראוי לציין שבתהליך ביצוע הניסויים לא בוצעה אופטימיזציה לתהליך, מה שממחיש את הפוטנציאל הגלום בה.

קיימות שיטות ידועות המשתמשות בייצוגים שונים של המידע לשיפור ביצועי מודלים, אך לרוב מדובר בגישות מורכבות הדורשות התאמות משמעותיות או בתהליך הלמידה או על ידי תיוג ידני של המידע עצמו ובניית היררכיות ומורכבות נוספת בהגדרת הצלחת המודל בתיוג נכון.

מטרת פרק זה היא לעורר עניין ולספק תיאבון למחקרים נוספים, אשר יידרשו לבחון לעומק את ההיבטים השונים של שינוי הייצוג והשפעתו על ביצועי המודל. בעוד שהצגנו כאן גישה ראשונית והראנו את הפוטנציאל הגלום בה, יש צורך במחקר מקיף יותר שיחקור את מגבלות השיטה ואת היכולת ליישם אותה במודלים שונים ומורכבים יותר אשר מביאים לתוצאות טובות יותר ללא השיטה המוצעת.

מסקנות והמלצה להמשך מחקר

כבר בתחילת המחקר הבנו שעל מנת להצליח למצוא ארכיטקטורה או שינוי במודל שאינו מתבסס על Adversarial Learning, עלינו להתמקד בלמידה ממקור החיכוך שבבעיה ולא להסתמך רק על מחקרים קיימים. מדובר בבעיה קשה ולא פתורה: קל מאוד לתקוף מודלים, אך קשה מאוד להגן עליהם.

הרכיב שיוצר את החולשה הפנימית של המודלים הוא גם הרכיב שדרכו ניתן יהיה למצוא פתרון. השלבים הראשונים במחקר שלנו התמקדו בזיהוי הרכיב הזה. השלבים המאוחרים יותר התמקדו במציאת הפרמטרים ש"ינטרלו את העוקץ" של הבעיה.

במאמר "Intriguing Properties of Neural Networks (Szegedy et al., 2014) " עלתה תזה מעניינת המשלבת את הגורמים שיוצרים את חולשתם של מודלים לתקיפות מסוג זה:

1. **אזורים דלילים במרחב הקלט** :התופעה מתרחשת לעיתים קרובות באזורים במרחב הקלט שלא נצפו במהלך האימון או באזורים בהם הנתונים דלילים. אזורים אלו עלולים לגרום למודל לספק סיווג שגוי.

2. **רגישות של Softmax ו-Cross-Entropy:** השימוש בשכבה האחרונה במרחב הסתברויות, באמצעות Softmax ו-Cross-Entropy**,** מגביר את הרגישות של המודל. כתוצאה מכך, גם שינויים קטנים בקלט עשויים להוביל לסיווג שגוי.

המחקר שלנו, בסופו של דבר, התמקד ברגישות של השכבות האחרונות. על אף שבמאמר זה ובמחקרים נוספים כבר הוצגו תובנות לגבי הרכיב המשפיע על הבעיה, החלטנו לאמץ גישה שונה. במקום להסתמך על מסקנות ופתרונות קיימים, בחרנו לחקור את הבעיה באופן עצמאי, מתוך למידה על ידי ביצוע ניסויים. גישה זו אפשרה לנו להבין את המנגנון לעומק ולגבש מסקנות מניסיון וחיכוך ישיר עם הבעיה.

במהלך המחקר, הצגנו שיטה פשוטה המשפרת את עמידות המודלים לתקיפות חזקות מסוג WhiteBox, והראנו שהיא אינה מסייעת בהגנה מפני תקיפות מבוססות Transferability.

השילוב בין הפשטות של השיטה, השיפור המשמעותי שהיא מספקת בעמידות המודל, ואף תרומתה לשיפור הביצועים על תמונות נקיות, מעלים שאלה מתבקשת: **אם פתרון כה פשוט מסוגל להתמודד עם סוג מסוים של תקיפות, מדוע שלא יהפוך לסטנדרט בסיסי במימוש מודלים? גם אם הוא אינו מספק הגנה מלאה לכל סוגי התקיפות, השימוש בו מאפשר לצמצם מראש חלק מהאיומים הידועים ולפנות את המאמץ המחקרי לאתגרים המורכבים יותר שעדיין נותרו פתוחים**.

בעולם האמיתי, כל תמונה מורכבת ממגוון רחב של פרטים, ומשמעותה עשויה להשתנות בהתאם להקשר. **השיטה המוצעת מספקת בסיס לייצוג מורכב ועשיר יותר של המידע, אשר עשוי לשפר את יכולת המודלים להתמודד עם סיטואציות מציאותיות ומורכבות יותר.** הראנו את הפוטנציאל הגלום בשינוי ייצוג המידע, וכיצד הוא יכול לשפר את ביצועי המודל באופן יעיל, בעלות נמוכה ובפשטות גבוההץ

לתפיסתנו, הרגישות של מודלים לתקיפות מסוג BlackBox נובעת ממקורות שונים בהשוואה לרגישות לתקיפות WhiteBox. בעוד שתקיפות WhiteBox מנצלות מידע ישיר על גרדיאנט המודל, תקיפות BlackBox מתבססות על תופעות כמו Transferabilty או ניצול של תבניות חבויות במבנה הרשת. לכן,

פתרון מקיף צריך להיות רב-שכבתי ולשלב מספר גישות הגנה, כאשר השיטה שהצענו מהווה נדבך בסיסי שעליו ניתן להוסיף שיטות המותאמות לתקיפות שאינן נשענות על נגישות לגרדיאנט.

מכאן, אנו סבורים כי מחקרי המשך צריכים להתמקד בעיקר בעולמות של תקיפות מסוג BlackBox, ובעיקר באלו שאינן מבוססות גרדיאנט, שכן תקיפות אלו ממשיכות להוות אתגר משמעותי. עם זאת, במקום לנסות למצוא שיטה אחת שתתמודד עם כל סוגי התקיפות, יש לשאוף לפיתוח מספר שיטות שונות, כאשר כל אחת מהן מגנה מפני משפחת איומים מסוימת, וביחד הן יוצרות מערך הגנה רחב ומקיף יותר.

לאור זאת, אנו מאמינים כי אם מחקרי המשך ינסו לזהות את המאפיין שיוצר את הרגישות לתקיפות מסוג BlackBox, סביר להניח כי מאפיין זה יימצא ברמת הארכיטקטורה של המודל עצמו ללא שימוש במניפולציה על המידע עצמו עליו המודל מאומן. באופן דומה לשיטה שהצגנו עבור תקיפות WhiteBox, גם במקרה זה, אנו מעריכים כי הפתרון יהיה פשוט ליישום, ואולי אף יהפוך לסטנדרט בבניית מודלים – ויהווה בסיס בבניית המודל שעליו יהיה אפשר לממש שיטות מתקדמות יותר לשיפור נוסף בביצועים.

רשימת מקורות

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2015. [Online]. Available: https://arxiv.org/pdf/1412.6572

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," *arXiv preprint arXiv:1706.06083v4*, 2019. [Online]. Available: https://arxiv.org/abs/1706.06083

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2014. [Online]. Available: https://arxiv.org/pdf/1312.6199

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2019. [Online]. Available: https://arxiv.org/pdf/1706.06083

[5] M. Goibert and E. Dohmatob, "Adversarial Robustness via Label-Smoothing," *arXiv preprint arXiv:1906.11567*, 2019. [Online]. Available: https://arxiv.org/pdf/1906.11567

[6] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified Robustness to Adversarial Examples with Differential Privacy," *arXiv preprint arXiv:1802.03471v3 [cs.LG]*, 1 Nov 2018. [Online]. Available: https://arxiv.org/pdf/1802.03471v3

[7] I. Goodfellow, Y. Bengio, and A. Courville, "Convolutional Networks," in *Deep Learning*. Cambridge, MA: MIT Press, 2016, pp. 330-370. [Online]. Available: https://www.deeplearningbook.org

[8] National Institute of Standards and Technology, "Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations," NIST AI 100-2e2023, Gaithersburg, MD, USA, 2024. [Online]. Available: https://csrc.nist.gov/pubs/ai/100/2/e2023/final

[9] N. Carlini and D. Wagner, "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods," *arXiv preprint arXiv:1705.07263*, 2017. [Online]. Available: https://arxiv.org/pdf/1705.07263

תקציר

תקיפות מבוססות מניפולציה של תמונות, כמו FGSM (Fast Gradient Sign Method), הן חלק מרכזי בעולם ה-Adversarial Attacks , מטרתן להטעות מודלים של למידת מכונה על ידי יצירת שינויים זעירים בקלט (כמו תמונה) שלא נראים לעין האנושית, אך גורמים למודל לסווג אותו באופן שגוי.
תקיפות אלו מנצלות חולשות במבנה המודל. הבעיה נעוצה בכך שמודלים חזקים לכאורה, כמו רשתות עצביות עמוקות, מגלים רגישות מפתיעה לשינויים זניחים אלו, מה שמעלה אתגרים משמעותיים ביצירת מערכות עמידות ומאובטחות בתחומים קריטיים כמו זיהוי פנים, נהיגה אוטונומית ורפואה.

במחקר שלנו הצגנו שיטה פשוטה בשם LabelDropOut שמבוססת על שינוי ייצוג הקלט, העברת המודל לבעיה מסוג MultiClass וביצוע DropOut על שכתב התיוג. הצלחנו להראות שעל ידי שינוי יחסית פשוט בארכיטקטורה בשכבות האחרונות של המודל אנחנו מצליחים לייצר עמידות לתקיפות כמו FGSM, PGD ותקיפות מבוססות Transferability.

תוצאות מחקר זה מציגות דרך חדשה ליצירת עמידות לעמידות מודלים לתקיפות שיכולה להוות בסיס שעליו יכולות שיטות מתקדמות יותר להתפתח. עלות ומורכבות מימוש השיטה היא נמוכה מאוד ובעלת אימפקט משמעותי על משפחה רחבה של תקיפות.

נספח: ניסויים בוסריים ותובנות ראשוניות

במהלך המחקר נערכו מספר ניסויים בוסריים שאינם חלק מהותי מהמאמר או מתוצאותיו המרכזיות. ניסויים אלה שימשו עבורנו קרקע פוריה ללמידה ולהתנסות. הם התבצעו מתוך רצון להבין טוב יותר את הדינמיקה של הבעיה, לעיתים על בסיס הנחות שאינן מדויקות או גישות נאיביות.

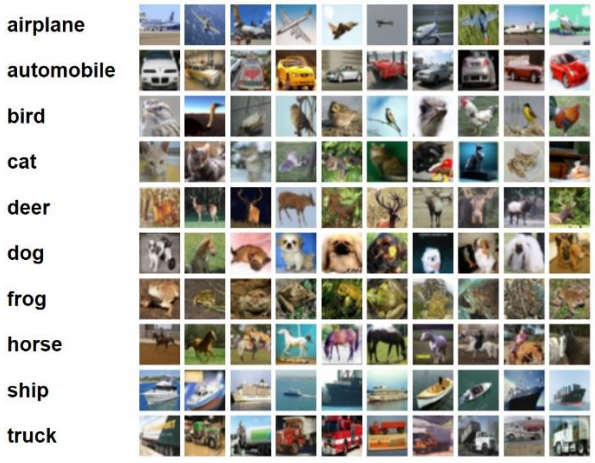
למרות שלא הפכו לחלק מהותי מהמסקנות שהוצגו במאמר, תיעוד ניסויים אלו והפקת התובנות מהם תרמו להבנה מעמיקה יותר של הבעיה. פרק זה נועד לשתף את הקוראים בתהליך המחקרי עצמו, עם כל מורכבויותיו, כדי לשקף את האתגרים והצעדים שנעשו בדרך למציאת הפתרון.

ניסויים ותוצאות עבור CIFAR-10

בפרק זה נציג מספר ניסויים ותוצאות, שבאמצעותם נדגים את התפתחות המחקר שביצענו. נציג את ההבניה החדשה של שכבת התיוג האחרונה במודל, ונראה כיצד היא מאפשרת קפיצת מדרגה משמעותית בעמידות לתקיפות White-Box, גם כאשר נעשה שימוש בפונקציית Loss מסוג Categorical Cross-Entropy.

בשלב האחרון של הצגת הניסויים והתוצאות, נדגים את המעבר למודלים המבוססים על בפונקציית Loss מסוג Binary Cross-Entropy, ונראה שעל מנת להגיע לרמת העמידות כפי שהוצג בפרקים הקודמים נדרש להשתמש ב LabelDropOut.

CIFAR-10 הוא מסד נתונים שמהווה כלי חשוב ומועיל בתחום ראיית ממוחשבת ולמידת המכונה. הוא מציע איזון טוב בין גודל ונוחות השימוש לבין המורכבות והאתגר בסיווג קטגוריות מגוונות. עם זאת, יש לו מגבלות בשל הרזולוציה הנמוכה ומספר התמונות המוגבל. סך כל תמונות במאגר הנתונים הוא 60,000 מתוכם 50,000 לאימון ו 10,000 לבדיקה. כל תמונה היא בגודל 32X32X3 פיקסלים בלבד, מה שמאפשר עבודה מהירה ונוחה מבחינת זמן האימון וצריכת המשאבים. לאור זאת, המאגר מהווה כלי פופולרי בתחום המחקר וישנם כלים ומאמרים רבים שמתבססים על מאגר זה. המאגר מתאים מאוד ללמידת עמוקה בזכות מגוון הקטגוריות והאובייקטים המורכבים. הרזולוציה היחסית נמוכה בתמונה מגבילה את כמות הפרטים שניתן לזהות בתמונה.



ניסוי 1 - פירוק מבנה השכבה האחרונה

כחלק מתהליך הלמידה שהתבצע במחקר, גילינו כי מודלי CNN רגישים באופן משמעותי לתקיפות whitebox. בשלבים הראשוניים של המחקר, התמקדנו במיפוי התנהגותם של מודלים שונים בארכיטקטורות מגוונות. בחנו פונקציות אופטימיזציה שונות, כגון Adam ו SGD, לצד השוואה בין מודלים רחבים לעומת עמוקים. בנוסף, ניתחנו את השפעת רכיבים כמו MaxPooling ושיטות נירמול שונות.

אחת ההשערות שעלו במהלך המחקר היא שהבעיה נובעת מאופן תיוג/ייצוג המידע. ייתכן שניתן לסייע למודל ליצור גבולות חדשים בין המחלקות השונות באמצעות שינוי ייצוג הנתונים. כך, נוכל לאלץ את המודל "לקפוץ" בין מחלקות הנתפסות כשונות מבחינתו מבחינת התיוג, אך למעשה מייצגות את אותה הקטגוריה, ובכך לשפר את עמידותו בפני התקיפות

במקום לייצג כל מחלקה באמצעות מיקום יחיד, ננסה לייצג אותה בשני מיקומים. לשם כך, הערכים באותם מיקומים יהיו 0.5 לכל אחד. במבנה שהצענו, כל מחלקה מיוצגת על פי מודולו של המספר שלה. כלומר, מחלקה 0 מיוצגת על ידי המיקומים 0 ו 10 , מחלקה 1 מיוצגת על ידי המיקומים 1 ו 11 וכך הלאה.

הציפייה היא שהמודל יצטרך לבחור בין שני מיקומים, כך שאם הוא מזהה נכון לפחות אחד מהמיקומים השייכים למחלקה הנכונה, המיקום השני ישמש כמעין ברירת מחדל במקרה של תקיפה.

לקחנו מודל במבנה בסיסי:

```python
def define_model(rate, num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_class, activation='softmax'))
    optimizer = Adam(learning_rate=rate)
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=[custom_accuracy])
    return model
```

על מנת לשפר את היעילות, ביצענו את האימונים על 5,000 דוגמאות בעיקר כדי לקבל תחושה כללית לגבי ביצועי המודל. בנוסף, השתמשנו בפונקציית accuracy ייחודית, השונה מהפונקציה הרגילה. פונקציה זו מסכמת את הערכים בכל המיקומים הרלוונטיים עבור כל מחלקה, ולאחר מכן המחלקה עם הסכום הגבוה ביותר נבחרת כסיווג הסופי של המודל.

```python
def custom_accuracy(y_true, y_pred):
    # Reshape y_pred to (batch_size, total_positions)
    y_pred = tf.reshape(y_pred, (-1, num_classes))  # Assuming 1000 positions in the output layer
    y_true = tf.argmax(y_true, axis=-1)  # Assuming y_true is a one-hot encoded vector

    # Create a list to sum the positions for each class
    class_sums = []

    for class_idx in range(10):  # For each class
        # Sum the values at positions for this class (mod 10 == class_idx)
        positions_for_class = y_pred[:, class_idx::10]
        class_sums.append(tf.reduce_sum(positions_for_class, axis=-1))

    # Stack the class sums and find the predicted class
    class_sums = tf.stack(class_sums, axis=-1)
    y_pred_class = tf.argmax(class_sums, axis=-1)

    # Compute accuracy by comparing predicted class with the true class
    accuracy = tf.reduce_mean(tf.cast(tf.equal(y_pred_class, y_true), tf.float32))

    return accuracy
```

את השכבה האחרונה של המודל אנחנו משנים ע"י 2 פונקציות:

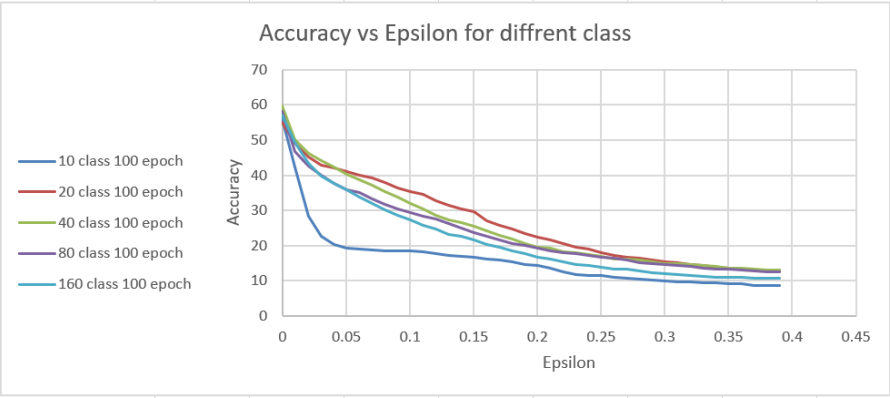
1. פונקציה שהופכת את האובייקט שמחזיק את שכבת התיוגים למבנה one-hot

```
y_1_ = tf.keras.utils.to_categorical(y_cut,num_classes )
```

2. ופונקציה ייעודית שמכניסה במיקומים המודלו את הערך המתאים. בהינתן ומשתמשים softmax חשוב לשמור על ערכים שסכומם הוא בדיוק 1

```
def change_label_by_shift_optimized(labels_cat, labels):
    rows = np.arange(len(labels))
    base_indices = labels.flatten()  # Ensures labels are in a suitable shape for indexing.

    # Generate the column indices based on the pattern described
    # Each label shifts by multiples of 10, starting from the label value, repeated 4000 times.
    column_indices = base_indices[:, np.newaxis] + 10 * np.arange(int(num_classes/10))

    # Flatten row and column indices to use for direct indexing
    row_indices = np.repeat(rows, int(num_classes/10))
    column_indices = column_indices.flatten()

    # Safeguard to prevent indexing errors that go out of bounds
    valid_mask = column_indices < labels_cat.shape[1]
    row_indices = row_indices[valid_mask]
    column_indices = column_indices[valid_mask]

    # Update the array in-place
    labels_cat[row_indices, column_indices] = 0.0625

    return labels_cat
```

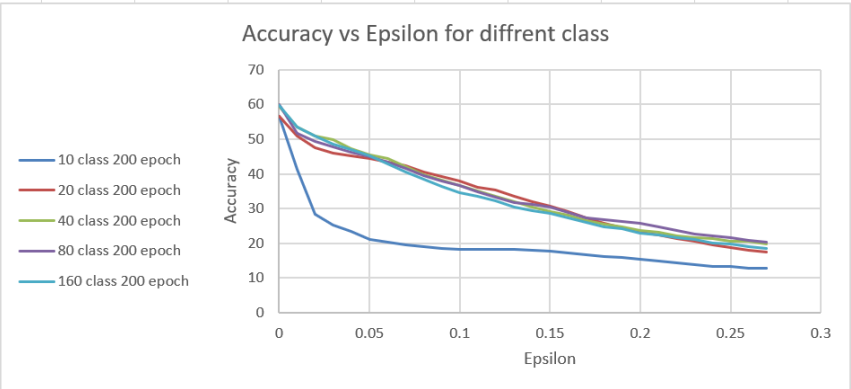
את הניסוי הרצנו עבור מודל עם מיקום 1,2,4,8,16 מיקומים לכל מחלקה. **ואלו התוצאות:**



ניתן לראות כי כבר בפיצול ל-20 מיקומים חלה קפיצת מדרגה משמעותית בעמידות המודל לתקיפות whitebox. כאשר אפסילון קטן מאוד, ההתנהגות של המודלים נותרת יחסית דומה, אך כבר בערך כבר אפסילון של 0.02 ניכר הבדל משמעותי. ככל שהאפסילון גדל ומתקרב לערכים גבוהים מעל 0.15, המודלים מתחילים להתנהג בצורה דומה זה לזה.

בדי לוודא שהבדלים אלה אינם נובעים רק ממספר ה-epochs, החלטנו להמשיך את אימון המודלים למשך 100 epochs נוספים. כך נוכל לבדוק האם משך הלמידה מהווה גורם המשפיע על הביצועים.

אלו התוצאות:



ניתן להבחין בקפיצה משמעותית בביצועים בין המודלים שבהם כל מחלקה חולקה למיקומים נוספים, לבין המודל הבסיסי שבו לכל מחלקה יש 10 מיקומים בלבד.

בנוסף, כאשר משווים את ביצועי המודלים בהתמודדות מול תקיפות WhiteBox, ניתן לראות כי המודלים עם 40, 80 ו-160 מיקומים מציגים יתרון של 3% בדיוק הסיווג של תמונות שלא עברו מניפולציה, בהשוואה למודלים שבהם המחלקות חולקו ל 10-או 20 מיקומים בלבד.

השיטה מציגה שיפור משמעותי בעמידות לתקיפות whitebox, באמצעות שינוי קטן ופשוט, מבלי לפגוע בביצועים הכלליים של המודל.

ניסוי 2 – בדיקת ביצועי עמידות למודלים מורכבים

עבור CIFAR-10, נבחן מספר מודלים תוך ביצוע שינויים ארכיטקטוניים והתאמות בפרמטרים מסוימים. פרמטרים אלו מהווים גורם מכריע ביכולת המודלים להתמודד עם תקיפות WhiteBox.

בשלב ראשון, כל המודלים יתבססו על פונקציית Loss מסוג Categorical Cross Entropy. בנוסף, מתוך רשימת המודלים שנבדקו, נזהה שני מודלים דומיננטיים במיוחד בעמידותם לשני סוגי התקיפות.

באופן חריג, נמצא מודל אחד ספציפי ("מודל 0") שמציג עמידות מוחלטת לתקיפות WhiteBox. נראה כי עמידות זו נובעת מכך שהמודל נכשל ביצירת שינויים בתמונות, ולכן התקיפות מסוג FGSM ו-PGD אינן משפיעות עליו כלל.

מודל 0 - יש בו שכבה אחת של Dense בגודל 512 עם אקטיבציה softplus לפני שכבה נוספת של Softmax

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='softplus'))
    model.add(Dense(num_class, activation='softmax'))

    # Initialize the Adam optimizer with a custom learning rate
    optimizer = Adam(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```

מודל 1- יש בו שתי שכבות Dense עם אקטיבציה ReLU. שכבה אחת בגודל 512 ולאחריה שכבה בגודל 128. שכבה אחרונה היא עם אקטיבציה softmax

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_class, activation='softmax'))

    # Initialize the Adam optimizer with a custom learning rate
    optimizer = Adam(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```

מודל 2- יש בו שתי שכבות Dense עם אקטיבציה ReLU. שכבה אחת בגודל 512 ולאחריה שכבה בגודל 128. שכבה אחרונה היא עם אקטיבציה softmax. שימוש באופטימייזר SGD

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_class, activation='softmax'))

    # Initialize the SGD optimizer with a custom learning rate
    optimizer = SGD(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```

מודל 3- יש בו שתי שכבות Dense עם אקטיבציה ReLU. שכבה אחת בגודל 512 ולאחריה שכבה בגודל 128. שכבה אחרונה היא עם אקטיבציה softmax. שימוש באופטימייזר SGD כאשר השתמשנו רק במשקולות בתהליך הלמידה ולא במודל כולו.

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_class, activation='softmax'))

    # Initialize the SGD optimizer with a custom learning rate
    optimizer = SGD(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```

מודל 4- יש בו שתי שכבות Dense עם אקטיבציה ReLU. שכבה אחת בגודל 512 ולאחריה שכבה בגודל 128. שכבה אחרונה היא עם אקטיבציה softmax. שימוש באופטימייזר SGD בתהליך למידה שמשתמש רק במשקולות. שכבה אחרונה בגודל 100[בניגוד ל 10 כמספר המחלקות]

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_class, activation='softmax'))

    # Initialize the SGD optimizer with a custom learning rate
    optimizer = SGD(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```

מודל 5- שכבת Dense יחידה בגודל 10 עם אקטיבציה softplus

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(num_class,activation='softplus'))

    # Initialize the Adam optimizer with a custom learning rate
    optimizer = Adam(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer,
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True), m
    etrics=['accuracy'])

    return model
```

מודל 6- שכבת Dense יחידה ואחרונה בגודל 100 עם אקטיבציה softplus. שימוש באופטימייזר SGD.

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(num_class,activation='softplus'))

    # Initialize the SGD optimizer with a custom learning rate
    optimizer = SGD(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer,
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])

    return model
```

מודל 7- שכבת Dense יחידה ואחרונה בגודל 100 עם אקטיבציה softplus. שימוש באופטימייזר SGD. בתהליך הלימוד השתמשנו רק במשקולות ולא במודל כולו.

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(num_class,activation='softplus'))

    # Initialize the SGD optimizer with a custom learning rate
    optimizer = SGD(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer,
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'])

    return model
```

מודל 8- יש בו שכבה Dense יחידה ואחרונה עם אקטיבציה eLU. שכבה זו בגודל 100

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(num_class,activation='relu'))

    # Initialize the Adam optimizer with a custom learning rate
    optimizer = Adam(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer,
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    return model
```

מודל 9- יש בו שכבה Dense יחידה ואחרונה עם אקטיבציה ReLU. שכבה זו בגודל 100

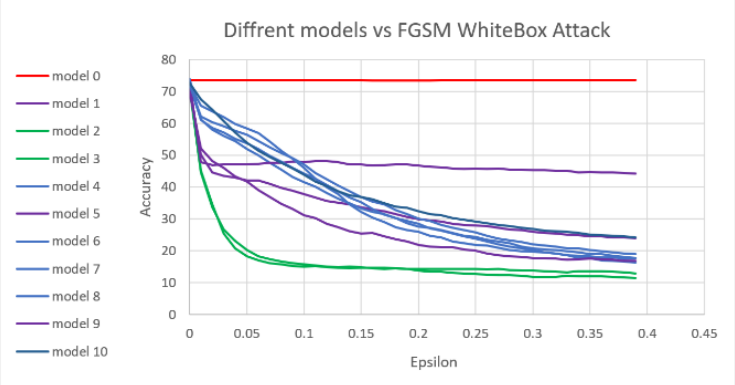
```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(num_class,activation='relu'))

    # Initialize the Adam optimizer with a custom learning rate
    optimizer = Adam(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer,
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

    return model
```

מודל 10- הוספת שכבות Conv בין כל שכבת MaxPooling בהתאמה. אופטימייזר SGD כאשר בתהליך הלמידה השתמשנו רק במשקולות ולא במודל המלא. שכבה אחרונה בגודל 31250. , Learning rate = 0.1

```
def define_model(rate,num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
    input_shape=(32, 32, 3)))
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dense(num_class, activation='softmax'))

    # Initialize the SGD optimizer with a custom learning rate
    optimizer = SGD(learning_rate=rate)

    # compile model
    model.compile(optimizer=optimizer, loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```

בתחילה, ביצענו תקיפת WhiteBox מבוססת FGSM על כלל המודלים, בטווח אפסילון של 0 עד 0.4, עם קפיצות 0.1 בין התקיפות השונות.

תוצאות FGSM:



סימנו בדיאגרמה גרפים כדי בעלי התנהגות דומה בצבעים. נמצא כי מודל 0 מתנהג באופן חריג, שכן תקיפת WhiteBox אינה משפיעה עליו כלל.

- גרפים כחולים מפגינים עמידות גבוהה לתקיפת WhiteBox, בעיקר בערכי אפסילון קטנים. עם זאת, כאשר אפסילון גדל, הם מאבדים את היתרון בהשוואה למודלים אחרים. מודלים
- גרפים סגולים מציגים חולשה באפסילון קטן ,אך באופן מפתיע הם מפגינים עמידות גבוהה באפסילון גדול, מה שעשוי לרמז על התנהגות לא שגרתית.
- גרפים ירוקים הם הפגיעים ביותר, ונחלשים משמעותית כבר באפסילון קטן.

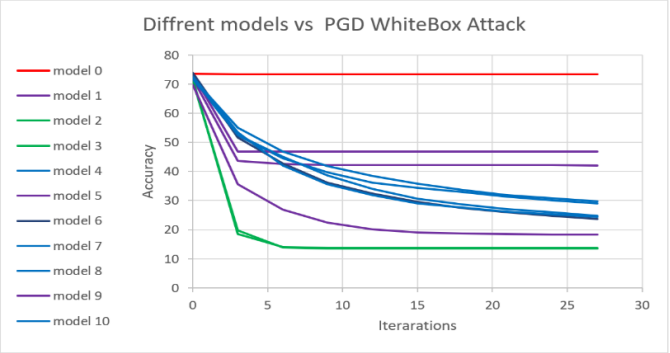
נוסף, ביצענו תקיפה מבוססת PGD על המודלים תחת תקיפת whiteBox, בשתי תצורות:

1. צעדים של 0.0001
2. צעדים של 0.001

במהלך הבדיקות, שמרנו על תיעוד הגרפים לפי הצבעים, על מנת שניתן יהיה לעקוב אחר המגמות בהתנהגות המודלים תחת התקיפות השונות

תוצאות תקיפה PGD 0.0001:

תוצאות תקיפה PGD 0.001:



נראה כי בכל סוגי התקיפות שנבדקו, כלל המודלים שומרים על מגמת התנהגות דומה .בעת, ננסה לזהות את ההבדלים בין הקבוצות השונות ,על מנת לאפיין את הרכיבים שאחראים לתופעה, תוך התמקדות במודלים העמידים במיוחד.

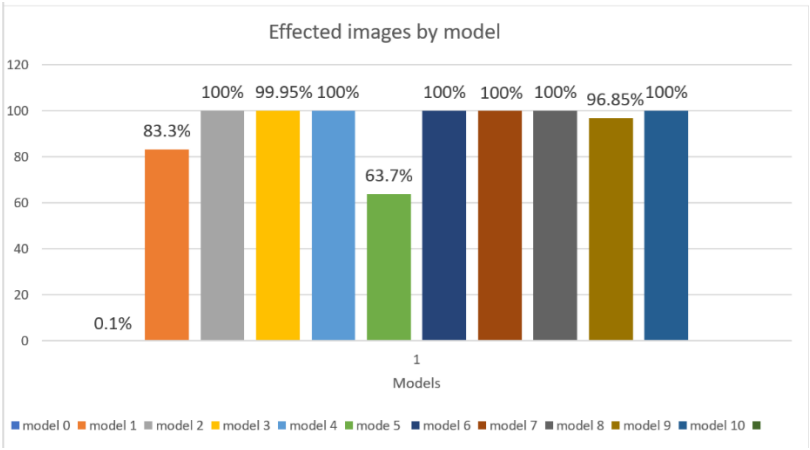
ניתוח קבוצות המודלים:

- מודלים 4, 6, 7, 8, 10 - בכולם השכבה האחרונה היא בגודל 100, למעט מודל 10, שבו השכבה האחרונה היא בגודל 31,250.
- מודלים 1, 5, 0 – מציגים התנהגות חריגה, בכך שמנקודה מסוימת של אפסילון, לא נצפית פגיעה נוספת בביצועים. מודל 0 חריג אף יותר, משום שהוא אינו מושפע כלל מכל ערכי אפסילון.

כדי לבדוק את מקור ההתנהגות הייחודית הזו, נבצע השוואה בין התמונות שנוצרות לאחר התקיפה לבין התמונות המקוריות. לצורך כך, נשתמש בפונקציה הבאה:

```python
def compare_images(original_images, adversarial_images,
    threshold=1e-15):
    unchanged_count = 0
    for orig, adv in zip(original_images, adversarial_images):
        difference = np.abs(orig - adv)
        if np.all(difference < threshold):
            unchanged_count += 1
    return unchanged_count
```

אלו התוצאות:



ניתן להבחין בקלות כי תקיפת WhiteBox על מודלים 1 ו-5 אינה מצליחה לייצר כלל שינויים כלל על חלק מהתמונות, ואילו עבור מודל 0, התקיפה כמעט ואינה מצליחה לשנות אף תמונה.

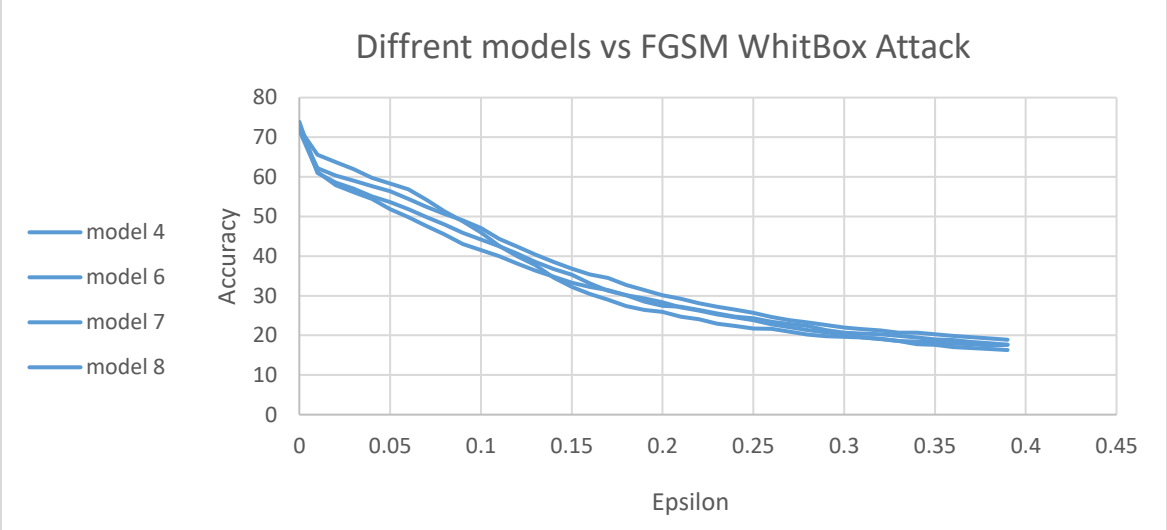
ממצא זה מסביר את השיפוע החריג שנצפה בגרפים של מודלים 1 ו-5 מאפסילון מסוים, וכן את העובדה שמודל 0 "עמיד" כמעט לחלוטין לתקיפה זו.

ניתן לראות כי גם מודל 8 מפגין התנהגות דומה, אם כי בהיקף זניח יותר.

ניסוי 3 – ניסיון לשיפור ביצועים למודלים מניסוי 2

ניסוי 3 נועד לשפר את התוצאות שהתקבלו בניסוי 2 (הגרפים הכחולים). במהלך המחקר, הבנו כי השינוי בייצוג הנתונים הוביל לקפיצת מדרגה משמעותית בעמידות המודלים, אך העובדה שלא הצלחנו לשפר עוד את הביצועים הטרידה אותנו. לכן, המשכנו לחקור כיוונים חדשים ולפתח גישות נוספות שיסייעו בשיפור עמידות המודלים.

בסעיפים הקודמים, זיהינו כי קיימות משפחות של מודלים עם מאפיינים מסוימים אשר מציגים תוצאות טובות יותר תחת תקיפת white-box. מודלים אלו כוללים את מודלים 4, 6, 7, 8 אשר סומנו בגרפים הכחולים.



המאפיין המשותף לכל המודלים שהפגינו ביצועים טובים באפסילון קטן הוא שכל מחלקה פורקה ל-10 מיקומים בתיוג. כאשר כל מיקום קיבל את הערך 0.1. בכל המודלים השתמשנו בפונקציית Categorical-Cross-Entropy.

השיפורים המרכזיים בעמידות לתקיפות:

רוב השיפורים המשמעותיים בהתמודדות עם התקיפות נבעו משינויים ארכיטקטוניים בשכבות האחרונות של המודלים, ובפרט בשכבות ה- Dense:

- הפעלת השכבות האחרונות עם פונקציות אקטיבציה SoftPlus ו- Elu
- שינוי גודל השכבה האחרונה ל-100 ומעלה.

חשיבות ייצוג המחלקות והתאמת ה-Loss:

בכל המודלים הללו, השכבה האחרונה, המייצגת את המחלקות, חולקה ליותר מ-10 מחלקות .מכיוון שהשתמשנו בפונקציית Categorical-Cross-Entropy, היינו צריכים לוודא שהערכים שניתנו למיקומים השונים שמייצגים מחלקה סכומם יהיה בדיוק 1.

פונקציה זו נפוצה בבעיות סיווג מרובות מחלקות ,ובמקרה שלנו, היא גורמת לכך שהמודל מנסה להגיע למצב שבו כל 10 המיקומים של מחלקה מסוימת יקבלו ערך של 0.1, כך שסכומם הכולל יהיה 1.

השוואה בין Categorical-Cross-Entropy ו- Binary-Cross-Entropy

אם נעבור להשתמש בפונקציית Binary-Cross-Entropy , נוכל להתייחס לכל מיקום כתיוג בלתי תלוי של אותה תמונה. במקרה כזה, לכל אחד מהמיקומים ניתן ערך של 1, גם אם סכום המיקומים יהיה גדול מ-1. במודל זה, כל מיקום מטופל בנפרד ואינו תלוי באחרים, פרט לכך שהמודל שואף להוריד את ה-Loss הכולל.

בהשוואה בין שתי הגישות:

- Categorical-Cross-Entropy מאלצת את המודל להגיע לפיזור ערכים מסוים ולשמור על סכום כולל של 1.
- Binary-Cross-Entropy מעניקה גמישות רבה יותר ,שכן כל מיקום מטופל כערך עצמאי ולא תלוי במיקומים אחרים.

שיפור הביצועים בהתמודדות עם תקיפת WhiteBox

ננסה לשפר את עמידות המודלים לתקיפות WhiteBox על ידי בדיקה של מודלים עם מבנה דומה ,תוך שינוי פונקציית ה-Loss והשכבות האחרונות.

ארכיטקטורת המודל:

```
def define_model(rate, num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(num_class, activation='sigmoid'))
    optimizer = Adam(learning_rate=rate)
    model.compile(optimizer=optimizer, loss=custom_loss, metrics=[custom_accuracy])
    return model
```

השינויים העיקריים במודל המוצג:

- השכבה האחרונה משתמשת בפונקציית Sigmoid.
- לכל תמונה מוקצים 100 מיקומים, כאשר המיקומים שמייצגים את המחלקה מקבלים את הערך 1, וכל שאר המיקומים מקבלים את הערך 0.

התאמת פונקציית ה-Loss

- פונקציית ה-Loss הותאמה במיוחד למבנה זה.
- מוגדר כי המידע שמגיע אליה כבר מנורמל ,כך שאין צורך בביצוע נירמול נוסף בתוך הפונקציה.

```
def custom_loss(y_true, y_pred):
    bce = BinaryCrossentropy(from_logits=False)
    loss = bce(y_true, y_pred)
    return tf.reduce_mean(loss)
```

התאמת פונקציית Accuracy

במודל החדש שלנו, כלל המיקומים של המחלקה הנכונה מקבלים את הערך 1. לכן, ישנן שתי גישות אפשריות לחישוב המחלקה הנבחרת:

1. אופציה א' - בחירת המיקום עם הערך הגבוה ביותר ובדיקה האם הוא שייך למחלקה הנכונה.
2. אופציה ב' - חישוב איזו מחלקה מכילה את סך הערכים הגבוה ביותר (על פני כל המיקומים השייכים לה), ולאחר מכן השוואה למחלקה הנכונה.

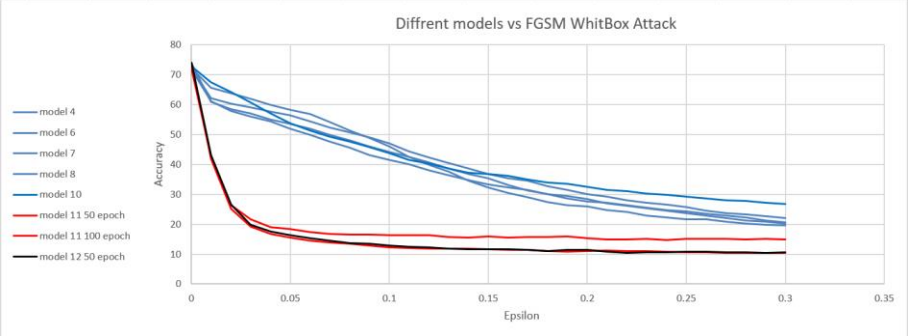
בחרנו ליישם את אופציה ב', שכן היא מאפשרת שקלול רחב יותר של המיקומים המוקצים לכל מחלקה.

```python
def custom_accuracy(y_true, y_pred):
    y_pred = tf.reshape(y_pred, (-1, NUMNUM))
    y_true = tf.argmax(y_true, axis=-1)
    class_sums = []
    for class_idx in range(10):
        positions_for_class = y_pred[:, class_idx::10]
        class_sums.append(tf.reduce_sum(positions_for_class, axis=-1))
    class_sums = tf.stack(class_sums, axis=-1)
    y_pred_class = tf.argmax(class_sums, axis=-1)
    accuracy = tf.reduce_mean(tf.cast(tf.equal(y_pred_class, y_true), tf.float32))
    return accuracy
```

תוצאות הניסוי

בחנו שני מודלים זהים, כאשר ההבדל היחיד ביניהם הוא גודל השכבה האחרונה:

- מודל ראשון: שכבה אחרונה בגודל 100
- מודל שני: שכבה אחרונה בגודל 1000



ניתוח התוצאות

ניתן לראות כי השינוי שביצענו בפונקציית ה-Loss ובאופן הנרמול בשכבה האחרונה לא הביא לתוצאה הרצויה. שני המודלים (מודלים 11 ו-12) מפגינים חולשה משמעותית בהתמודדות עם תקיפת WhiteBox.

מהממצאים עולה כי ייתכן שדווקא הגמישות של מבנה הייצוג החדש והחוזקה של המודל הם אלו שתורמים לחולשה שהתגלתה. מכיוון שהשכבה האחרונה משחקת תפקיד מרכזי בהיווצרות הרגישות לתקיפות מסוג זה, הגענו למסקנה כי עלינו להקשות על המודל באמצעות הכנסת רעש, מה שיאלץ אותו לייצר גבולות חדשים בין המחלקות.

השלב הבא

1. נוסיף רעש רנדומלי לשכבת התיוג למודלים 11 ו-12 על ידי ביצוע איפוס של חלק מהמיקומים בעלי הערך 1.
2. נאמן כל מודל למשך 15 איטרציות נוספות ונבחן האם השינוי שיפר את העמידות לתקיפת WhiteBox.
3. אם יתגלה שיפור, נשמור את המודל המעודכן ונחזור על התהליך עם רעש נוסף ,תוך בדיקה חוזרת, וחוזר חלילה.

נשתמש בפונקציה:

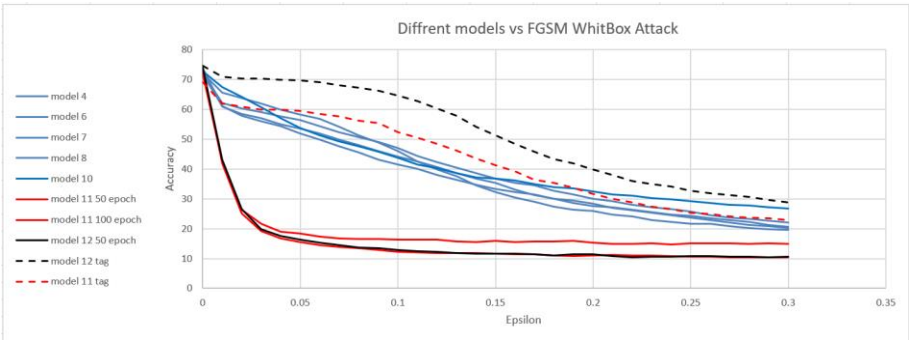
```python
def random_zero_out_positions(labels, amount):
    modified_labels = labels.copy()
    for i in range(labels.shape[0]):
        non_zero_positions = np.where(labels[i] != 0)[0]
        if len(non_zero_positions) < amount:
            raise ValueError(f"Row {i} has fewer non-zero positions ({len(non_zero_positions)}) than the amount ({amount}).")
        positions_to_zero = np.random.choice(non_zero_positions, size=amount, replace=False)
        modified_labels[i, positions_to_zero] = 0
    return modified_labels
```

הפונקציה מקבלת שני משתנים: labels ו-amount.

עבור כל שורה במערך labels, הפונקציה מאפסת באופן רנדומלי ערכים שאינם 0 במקרה שלנו, ערכים שהם 1), בהתאם לכמות שנקבעה על ידי amount.

תהליך זה שומר על איזון בין השורות השונות, כך שהמספר הכולל של השינויים נותר אחיד לכל הדוגמאות.

תוצאות הניסוי



ניתוח התוצאות לאחר הוספת רעש

ניתן לראות כי עבור שני המודלים, מודל 12tag ומודל 11tag, חל שיפור משמעותי מאוד בעמידות לתקיפת whitebox, ללא פגיעה בביצועי המודל המקוריים.

במהלך התהליך שביצענו, זיהינו שיפור ניכר כבר באיטרציה הראשונה, כאשר נקבע amount=1. כלומר, גם הוספה מינימלית של רעש רנדומלי סייעה לחזק את המודלים ולהפוך אותם לעמידים יותר בפני התקיפה.

מסקנות וסיכום

במהלך הניסויים, הראינו כי שינויים ארכיטקטוניים בשכבות האחרונות של המודל יכולים לשפר משמעותית את העמידות לתקיפות WhiteBox.

תובנות עיקריות:

- זיהינו מודלים שאינם מסוגלים לייצר תקיפות כלל (מודל 0), או שמייצרים תקיפות חלשות בלבד.
- "העמידות" של מודלים מסוימים אינה נובעת ממנגנון הגנה אפקטיבי ,אלא מכך שהמודלים עצמם לא מסוגלים לייצר תקיפות משמעותיות.
- ניתן למנוע תקיפת WhiteBox על ידי יצירת מבנה שמגביל את יכולת המודל לייצר תקיפה חלשה.
- הראינו שניתן לשפר עמידות לתקיפת WhiteBox באמצעות שינוי מבני של ה-Labels, כך שההשפעה תהיה רחבה על מגוון ארכיטקטורות שונות ,ולא רק על מודל ספציפי.

ניסוי 4 – בחינת עמידות רחבה יותר

בניסוי זה, נרצה לבחון האם השינויים הארכיטקטוניים שביצענו מספקים עמידות רחבה יותר בפני תקיפות.

היפתוזה מרכזית:

אם נצליח להראות שבתהליך הלמידה עצמו, כבל שמתקרבים למודל היעד, העמידות לתקיפת WhiteBox גדלה, נוכל להדגים הגנה חזקה כנגד תקיפות מסוג זה.

השפעת קרבת התוקף למודל היעד:

- באשר התוקף מנסה לייצר תקיפה חזקה יותר באמצעות יצירת מודל דומה ואף זהה למודל היעד, התקיפה מאבדת מהאפקטיביות לאור התבססות התוקף על הגרדיאנט.
- בכל שהתוקף "מתרחק" מהמודל היעד, כלומר משתמש במודל שונה יותר, האפקטיביות של תקיפת WhiteBox פוחתת, והוא נאלץ להסתמך על תקיפות ממשפחת ה BlackBox שאינם מבוססות על שימוש בגרדיאנט הספציפי של המודל היעד.
- תקיפות BlackBox המבוססות על Transferability חלשות משמעותית בהשוואה לתקיפת WhiteBox, מה שעשוי להוביח את עמידות המודל לאורך טווח רחב יותר של תקיפות אפשריות.

מודל 13

```python
def define_modelGOOD(rate, num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(200, activation='softplus'))
    model.add(Dense(num_class, activation='sigmoid'))
    optimizer = Adam(learning_rate=rate)
    model.compile(optimizer=optimizer, loss=custom_loss_for_10_classes, metrics=[custom_accuracy])

    return model
```

מודל 14

```python
def define_model(rate, num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(200, activation='relu'))
    model.add(Dense(num_class, activation='softmax'))
    optimizer = Adam(learning_rate=rate)
    model.compile(optimizer=optimizer, loss=CategoricalCrossentropy(), metrics=[custom_accuracy])
    return model
```

מודל 15

```python
def define_model(rate, num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(200, activation='relu'))
    model.add(Dense(num_class, activation='sigmoid'))
    optimizer = Adam(learning_rate=rate)
    model.compile(optimizer=optimizer, loss=custom_loss_for_10_classes, metrics=[custom_accuracy])
    return model
```

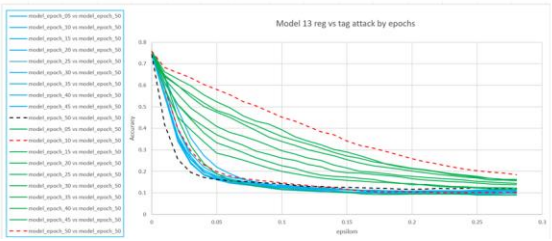
מודל 16

```python
def define_model(rate, num_class):
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(num_class, activation='sigmoid'))
    optimizer = Adam(learning_rate=rate)
    model.compile(optimizer=optimizer, loss=custom_loss_for_10_classes, metrics=[custom_accuracy])
    return model
```

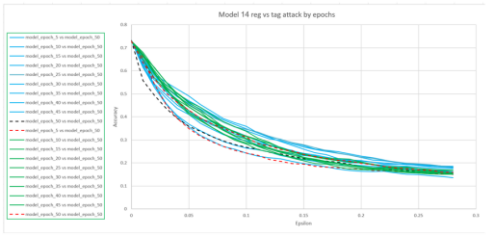
כדי להוביח שהמודלים שומרים על עמידות גם מול גרסאות קודמות שלהם, שמרנו את כלל המודלים שנוצרו לאחר כל epoch עבור מודלים 13-16.

הנתונים המוצגים משקפים את ביצועי כל מודל בהפרש של epochs 5, על מנת לבחון כיצד עמידות המודל לתקיפת whitebox משתנה לאורך זמן.

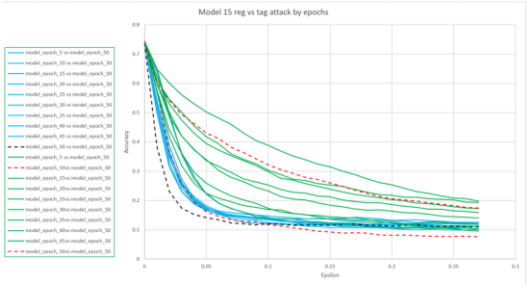
מודל 13



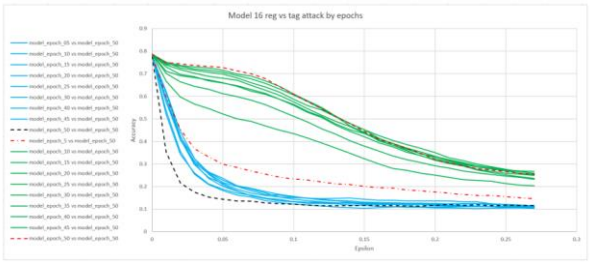
מודל 14



מודל 15



מודל 16



ניתוח היתרון של המודלים עם Labels שעברו שינוי

מהניתוח ניתן לראות בצורה ברורה את היתרון של המודלים שאומנו על Labels שעברו שינוי, וזאת משתי סיבות עיקריות:

1. השפעה הולכת ופוחתת ככל שמתקרבים למודל היעד
   - רוב המודלים שנוצרים במהלך תהליך האימון מראים השפעה קטנה יותר על המודל היעד ככל שהאימון מתקדם.
   - המשמעות היא שהעמידות לתקיפות מתחזקת ככל שהמודל מתכנס למודל היעד.
2. יתרון כמעט אבסולוטי על פני מודלים ללא שינוי ב-Labels
   - גם בשלבי האימון ההתחלתיים ,המודלים שעברו שינוי ב-Labels מציגים יתרון משמעותי כמעט בכל ההשוואות.
   - לעומת זאת, מודלים שנוצרו ללא שינוי ב-Labels נותרו פגיעים יותר לאורך כל שלבי האימון.

מקרה ייחודי – מודל 14 עם Softmax

מודל 14, שבו שכבת ה-Softmax אומנה על Labels שעברו שינוי, מציג תוצאות מעורבות:

- במקרים מסוימים, הוא מראה יתרון קל לעומת מודל זהה שאומן על Labels ללא שינוי.
- במקרים אחרים, הוא מציג חיסרון קל בהשוואה למודל ללא שינוי.

ממצאים אלה מעידים על כך שהשפעת שינוי ה-Labels תלויה גם בארכיטקטורת השכבות האחרונות, ולא רק בעצם שינוי הייצוג

המכללה האקדמית תל-אביב

בית הספר למדעי החשב

השפעת ארכיטקטורת הרשת, תהליך האימון והפרמטרים על תקיפות מבוססת מניפולציה על הקלט

פברואר 2025

חיבור זה הוגש כחלק מהדרישות לקבלת התואר "מוסמך " – M.Sc.

במכללה האקדמית תל-אביב

על ידי

כפיר גרמה

העבודה הוכנה בהדרכתו של

פרופ׳ עדי שרייבמן