ספר פרוייקט

Improved object detection

מגישים:

* עדי מזוז - 305769036
* דמיטרי קולטונוב – 320645401

מנחה:

* רומן קפלן

סמסטר:

* אביב 2020
* תוכן עניינים

[פרק א - הקדמה .3](#_Toc54026245)

[**רקע** 3](#_Toc54026246)

[**מטרת הפרויקט** 3](#_Toc54026247)

[פרק ב - כלים 4](#_Toc54026248)

[**IoU (Intersection over union)** 4](#_Toc54026249)

[**Map** 5](#_Toc54026250)

[פרק ג - אלגוריתמים לזיהוי אובייקטים 6](#_Toc54026251)

[**YOLO – you only look once** 6](#_Toc54026252)

[**Masked R-CNN** 8](#_Toc54026253)

[**עצי החלטה** 10](#_Toc54026254)

[**ResNet50** 13](#_Toc54026255)

[פרק ד – כללי החלטה 15](#_Toc54026256)

[**כללי החלטה פשוטים** 15](#_Toc54026257)

[**כללי החלטה בעזרת ML** 16](#_Toc54026258)

[**עצי החלטה** 17](#_Toc54026259)

[**עצי החלטה ורשת נוירונים** 17](#_Toc54026260)

[**עצי החלטה וResnet50** 17](#_Toc54026261)

[פרק ה - סיכום 18](#_Toc54026262)

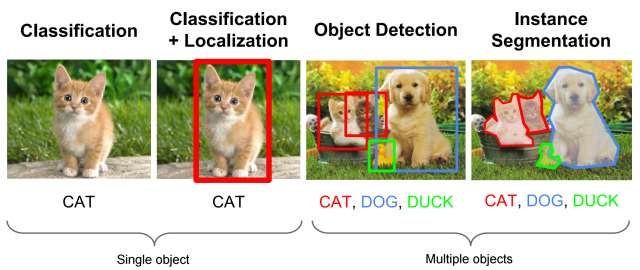
[פרק ו - מקורות 20](#_Toc54026263)

# פרק א - הקדמה

## **רקע**

המשימה המוגדרת כזיהוי אובייקטים בתמונה בעצם משלבת בתוכה מספר תתי משימות

1. זיהוי האובייקט בתמונה
2. מציאת קופסה המקיפה את אותו אובייקט הנותנת את מיקום האובייקט (לוקליזציה)
3. יש גם אלגוריתמים אשר משייכים כל פיקסל בתמונה האם הוא שייך לאובייקט מסויים או לא. (סגמנטציה)



## **מטרת הפרויקט**

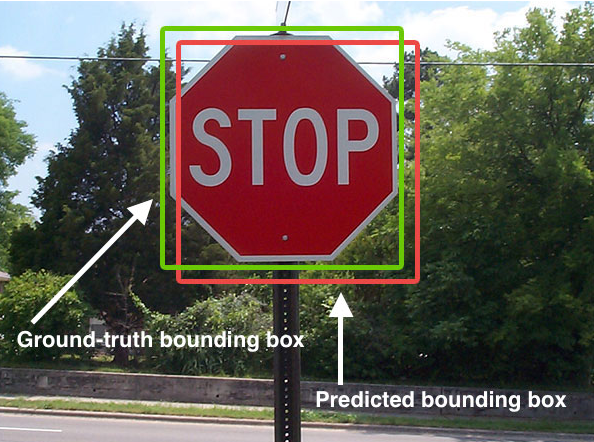
בפרוייקט אנו בוחנים מספר גישות לזיהוי אובייקטים בתמונה על ידי אלגוריתמים המשתמשים בCNN , המטרה שלנו היא לשלב בין שני האלגוריתמים הנלמדים במסגרת הפרוייקט על מנת ליצור מערכת משופרת הנותנת ביצועים טובים יותר

תחילה ע"י כללי החלטה פשוטים ובהמשך בעזרת כללי החלטה יותר מסובכים בעזרת ML

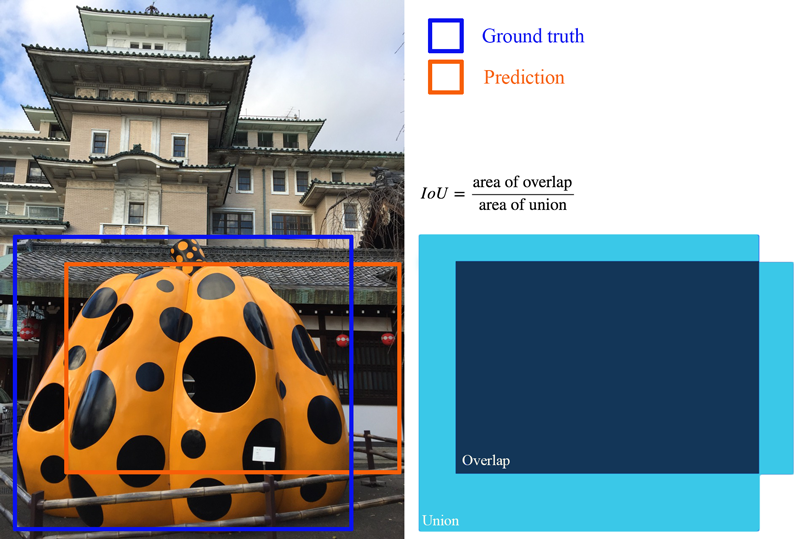
# פרק ב - כלים

## **IoU (Intersection over union)**

IOU - הינו מדד לחפיפה בין שני קופסאות, כלומר מספק מדד לדיוק המציאת האובייקט במאגר נתונים כלשהו. בהינתן קופסת הground-truth וקופסה ממודל כלשהו אנחנו נקבל מדד על מיקום הקופסה. (מדד לobject detection).



אנו נשתמש בו על מנת לסנן תוצאות כפולות, ה IOU בין שני קופסאות הינו מספר בין 0 ל 1, בדר"כ נהוג להגדיר את הסף לחפיפה (שני תוצאות חופפות) הינו 0.5



## **Map**

הMap הינו מדד מקובל אשר משמש על מנת לאמוד את רמת הדיוק של האלגוריתם

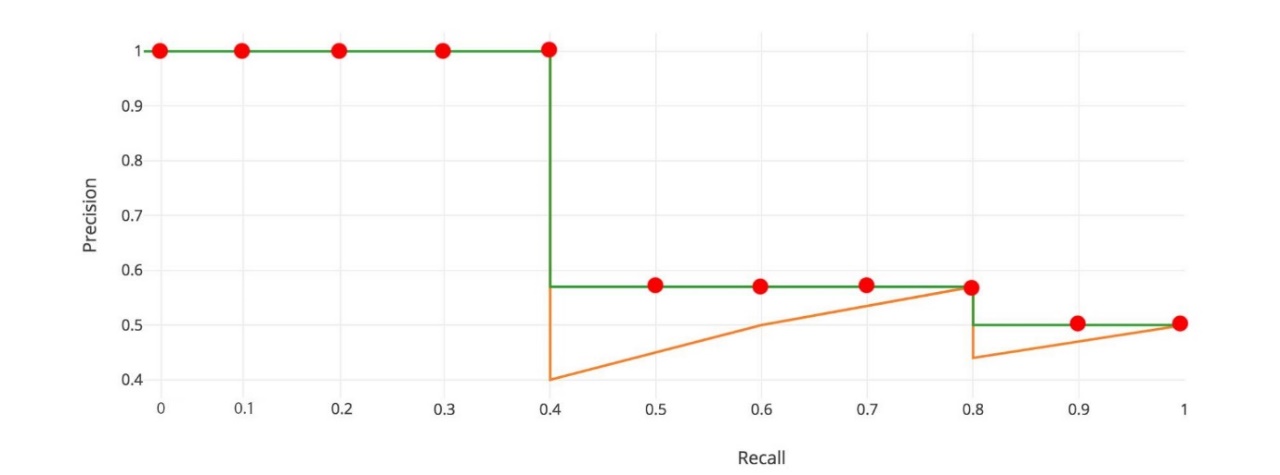
(מדד ל object classification).

ראשית נגדיר את

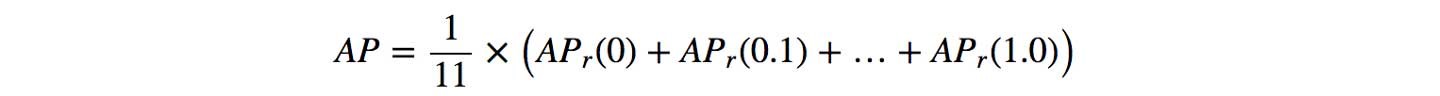
**Precision** - מודד מה האחוז מהתחזיות שלנו שהוא צודק

**Recall**  - מודד כמה מתוך התוצאות האפשריות מצאנו

אם נסתכל על גרף מסויים המראה precision מעל recall



נוכל לחשב עבורו את מדד הMAP באופן הבא:

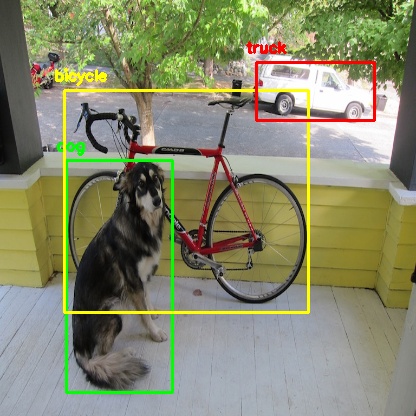


# פרק ג - אלגוריתמים לזיהוי אובייקטים

בתחילת הפרוייקט למדנו על שני אלגורתמים מוכרים אשר משמשים לזיהוי אובייקטים בעזרת רשתות נוירונים.

למדנו על אופן הפעולה של האלגוריתמים הקלט והפלט, וכיצד להשתמש בהם למטרותינו

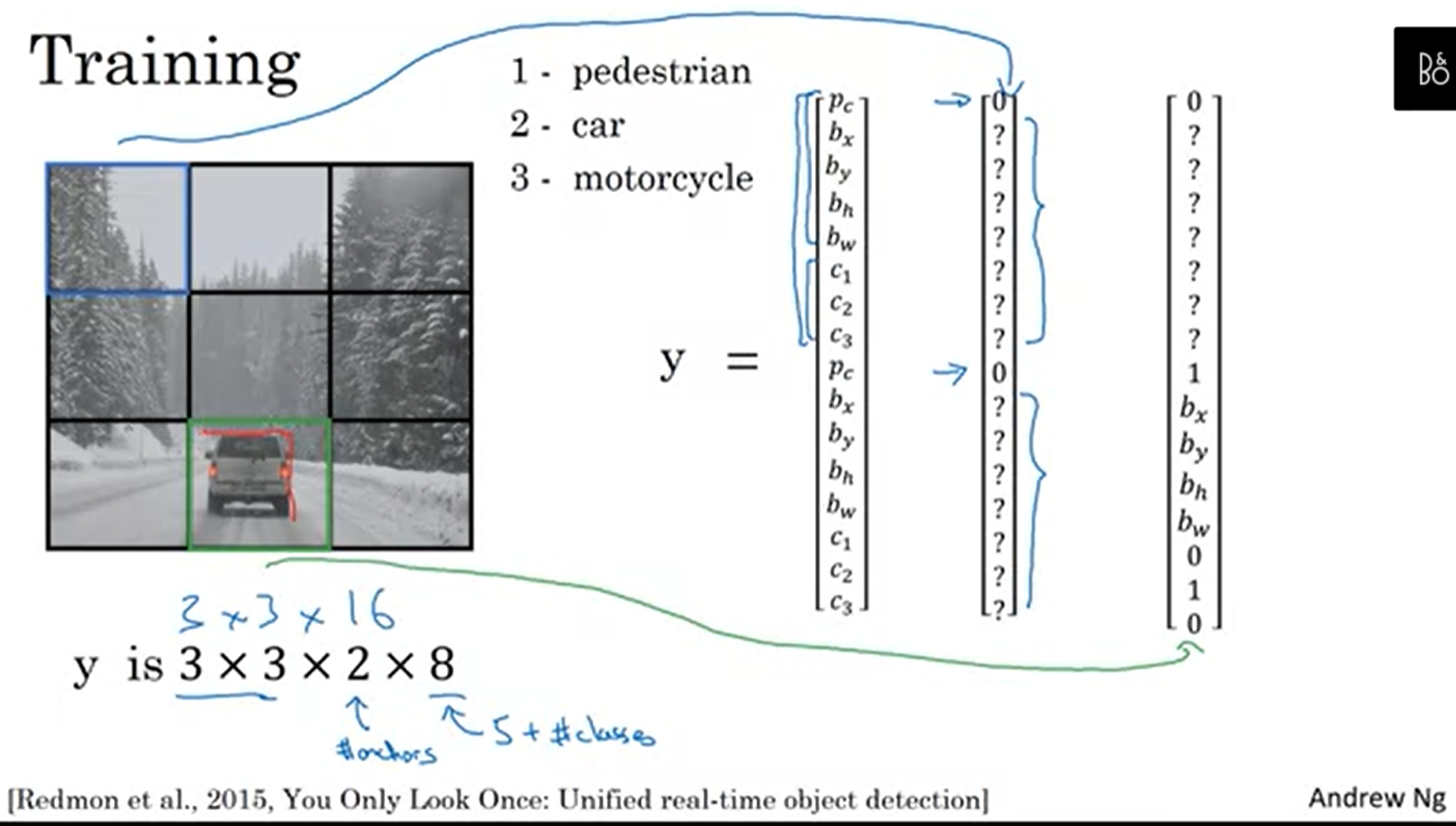
## **YOLO – you only look once**



YOLO הינו אלגוריתם מהיר המבצע זיהוי אובייקטים בתמונה בזמן אמת, הגרסה הרשמית של האלגוריתם נמצאת ב <https://pjreddie.com/darknet/> והיא זמינה לשימוש בgithub

אלגוריתמים קודמים המבצעים זיהוי אובייקטים בעצם הסתכלו על חלקים שונים של התמונה מספר פעמים בעזרת חלונות בגדלים שונים במטרה למצוא אובייקטים, הגישה הזו הייתה יקרה חישובית ובדר"כ איטית4

לעומת זאת אלגוריתם YOLO עובר בפעם אחת על התמונה (כאשר בעצם אנו מחלקים את התמונה למספר מסויים של חלונות) ומנסים לזהות בכל חלון אובייקט מסויים



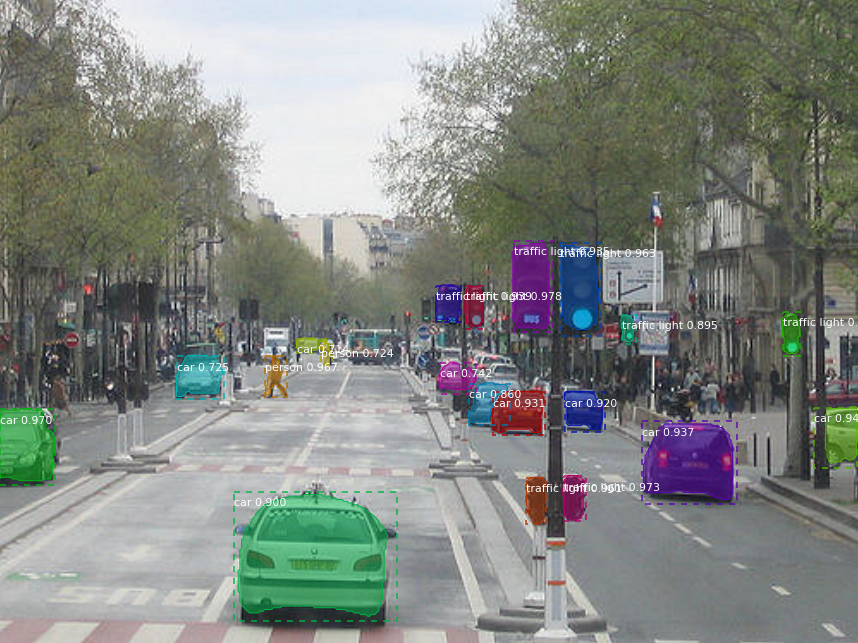
לכל "חלון" בעצם יש לנו שכבת פלט שמוציא לנו את המידע – האם זיהינו אובייקט, איזה, מה החלון שמקיף אותו ומה הביטחון של הרשת בכך.

כיוון שאובייקט מסויים יכול להיות מוכל בכמה חלונות בו זמנית אנו נבצע סינון על התוצאות של הרשת , אנו נמצא אובייקטים חופיים בעזרת IOU ונקח את התחזית עם הבטחון הגדול ביותר

אנו שילבנו לתוך הפרוייקט שלנו את הרשת המאומנת הבאה :

<https://github.com/arunponnusamy/object-detection-opencv>

## **Masked R-CNN**



אלגוריתם זה הוא בעצם חלק ממשפחה של אלגוריתמים :

Region-Based Convolutional Neural Network algorithm family

הכוללים את :

* **R-CNN**

**רשת לומדת עמוקה המשתמשת ב selective search כדי להציע קופסאות כפלט**

* **Fast R-CNN**

**נוספה שכבה שאמורה לנבא אזורים "מעניינים" בתמונה**

* **Faster R-CNN**

**נוספה רשת הקודמת לרשת המנבאת אשר תפקידה הוא להציע מקומות "מעניינים" בהם אנו חושדים שיהיו אובייקטים ורק שם לבצע את החישוב**

* **Mask R-CNN**

בעצם מהווה שיפור של המודל הקודם בכך שאנו מוסיפים חיזוי של כל פיקסל בתמונה

הפלט של האלגוריתם הינו תחזית המכילה :

1. קטגוריה
2. קופסא מקיפה
3. ביטחון בתחזית

ובנוסף רשימה של פיקסלים כאשר כל פיקסל משתייך לאובייקט מסויים או לשום אובייקט.

השתשמנו בגיט : <https://github.com/matterport/Mask_RCNN>

שימוש באלגוריתמים

בנינו מערכת המשלבת את שני הרשתות האלו אשר מוציאה לנו כפלט את הפלט של שני האלגוריתמים .

כיוון שלשני האלגוריתמים יש פלט קצת שונה היינו צריכים ליצור משניהם תחזית אחידה הנראית כך:

<class> <confidence> <x1> <x2> <y1> <y2> <algorithm>

שמרנו את התחזיות בקבצים על מנת לבצע עליהם חישוב עתידי

## **עצי החלטה**

שיטת עצי החלטה משמשת להסקה אינדוקטיבית, כלומר הסקה מתוך data נתון נוכל ללמוד ולחזות החלטה עבור data עתידי. שיטה זו שימושים מאוד במערכות למידה וגילוי ידע, כאשר תחום העיסוק העיקרי הוא סיווג (classification).

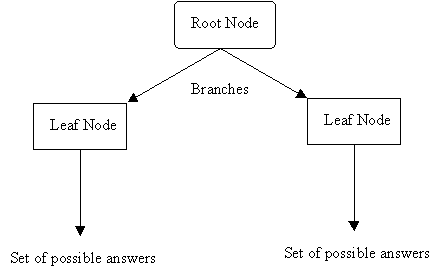
ביישום זה, נתונה קבוצה של דוגמאות (training set) כאשר לכל דוגמה יש אותו מבנה, והיא מכילה מספר זוגות של תכונות וערכים. אחת מהתכונות מייצגת את הסיווג של הדוגמה.

בהינתן קבוצה נוספת (test set) שעבורה הסיווג אינו ידוע, מטרת הלמידה היא ליצור עץ החלטה, בשימוש בדוגמאות מה- training set, אשר יקבע סיווג לכל רשומה בtest-set.

עצי החלטה מסווגים את הדוגמאות ע"י מיונם לעומק העץ מהשורש לאיזשהו צומת עלה, שמספק את הסיווג של הדוגמה. כל צומת בעץ מציין מבחן של תכונה מסוימת של הדוגמה, וכל ענף שיוצא מהצומת מתאים לאחד הערכים האפשריים לתכונה זו.

כל דוגמה מסווגת כאשר מתחילים בשורש של העץ, בוחנים את התכונה המצוינת ע"י צומת זה, ואז מתקדמים למטה בענף המתאים לערך של התכונה. תהליך זה חוזר על עצמו בצומת הבא בהסתעפות זו וכן הלאה עד שמגיעים לצומת עלה.

נתאר את עץ ההחלטה באמצעות הדיאגראמה הבאה:



* כל צומת שאינו עלה מתייחס למבחן שמפריד את קבוצת התשובות האפשריות לתתי קבוצות המתאימות לתוצאות שונות של המבחן. קבוצת התשובות האפשריות מתייחסת לערכי התכונה (המבחן) האפשריים.
* כל ענף נושא תת קבוצה של תוצאה ספציפית במבחן לצומת נוסף.
* כל צומת מתייחס לסט של תשובות אפשריות.

העיקרון המנחה אלגוריתמים לבניית עצי החלטה:

ההסקה של עצי החלטה הינו תחום פעיל מאוד במערכות למידה, וגישות וטכניקות רבות פותחו לבניית עצים עם יכולת סיווג גבוהה.

עבור סט דוגמאות נתון, ניתן לבנות עצי החלטה רבים שיסווגו נכון את כל הדוגמאות.

מבין כל העצים, אנו מעוניינים בעץ הפשוט ביותר. החיפוש הזה מודרך ע"י עקרון הידוע בשם Occam’s Razor. לפי עקרון זה, עץ ההחלטה הקטן ביותר שעקבי עם הדוגמאות הוא העץ בעל הסיכוי הטוב ביותר לסיווג דוגמאות חדשות בצורה נכונה.

בהינתן m תכונות, לעץ ההחלטה יכול להיות גובה מקסימאלי של m.

כלומר, נוכל לבנות את כל העצים האפשריים, למדוד את הגודל של כל אחד מהם, ולבחור את העץ הקטן ביותר שמתאים הכי טוב ל- data. מכיוון שזה לא מעשי מבחינה חישובית למצוא את עץ ההחלטה הקטן ביותר, אזי משתמשים בפרוצדורה שנוטה לבנות עצים קטנים.

מאחר ועץ ההחלטה מורכב מצמתים שמיצגים מבחן המבוסס על תכונה, פישוט העץ משמעותו הקטנה של מספר המבחנים. נוכל להשיג זאת ע"י בחירת המבחנים המתאימים בכל שלב בעץ.

למעשה, הבעיה העיקרית בלמידה של עצי החלטה לתכונות עם ערכים בינאריים היא בחירת הסדר של המבחנים. עבור ערכים קטגוריים או מספריים ( labeled data ) עלינו להחליט בנוסף איזה מבחנים צריכים להיות.

כלומר, בחירת המבחן הינה קריטית לפשטות עץ ההחלטה.

השיטה לבחירה של התכונה למבחן בכל צומת החלטה בעץ נקראת קריטריון בחירה.

Overfitting בעצי החלטה

תהליך בניית עץ ההחלטה מורכב למעשה משני שלבים:

* בנית עץ התחלתי מה- training data.
* הקטנת העץ להגדלת הדיוק על ה- train set.

כאשר שלב זה חיוני בעקבות תופעת ה- Overfitting.

ה- training set עלולים להכיל רעש, וע"י כך נקבל עץ למידה פחות מדויק.

למעשה בגלל שה- training set הם בד"כ רק מדגם קטן של כל הדוגמאות האפשריות, אזי קיימת האפשרות להוסיף ענפים לעץ שמשפרים ביצועים ( דיוק ) על ה- training set בעוד שמקטינים ביצועים על דוגמאות אחרות מחוץ לסט זה. תופעה זו נקראת overfitting.

למעשה overfitting אפשרי אפילו כאשר ה- training examples נקיים מרעש, במיוחד כשמספר קטן של דוגמאות מקושרות עם צמתי עלה.

במקרה זה תכונה מסוימת יכולה לחלק את הדוגמאות בצורה טובה למרות שהיא לא קשורה ישירות לפונקציית המטרה.

באופן כללי overfitting הינה בעיה מעשית ומשמעותית בלמידה של עצי החלטה ואף בשיטות למידה אחרות. Overfitting מקטין את הדיוק של עצי החלטה בדרך כלל ב10-25 אחוז.

ישנן כמה גישות למניעת overfitting בלמידה של עצי החלטה, המתחלקות לשני סוגים עיקריים:

גישות שעוצרות את גידול העץ בשלב מוקדם, לפני שהוא מגיע לנקודה שבה הוא מסווג את ה- training set באופן מושלם.

גישות שמאפשרות לעץ להגיע למצב של overfitting, ואז לבצע מחיקה של צמתים על העץ.

הגישה השנייה נמצאה יעילה יותר מעשית. זאת מכיוון שבגישה הראשונה יש קושי בקביעה מדויקת לגבי הרגע המתאים לעצירה של גידול העץ.

השיטה הנפוצה ביותר היא להשתמש בסט נפרד של דוגמאות, השונה מה- training setעל מנת לבצע את המחיקת צמתים בעץ. לסט הוא ה validation set (כלומר מחלקים את הdata לשני סטים training +validation.

הרעיון הוא שאפילו אם הלומד הוטעה ע"י שגיאות אקראיות בסט ה- training, לא סביר שיהיו אותן שגיאות אקראיות בסט של ה- validation.

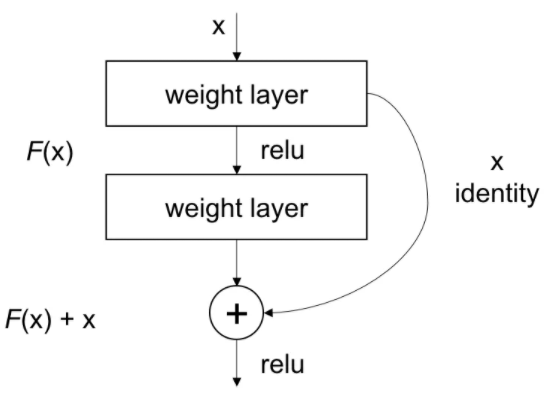
## **ResNet50**

כל המודלים הקודמים (YOLO, MRCNN) השתמשו ברשתות עצביות עמוקות שבהן הם ערמו שכבות קונבלוציה רבות אחת אחרי השנייה. אך ישנם כמה בעיות ברשתות עמוקות:

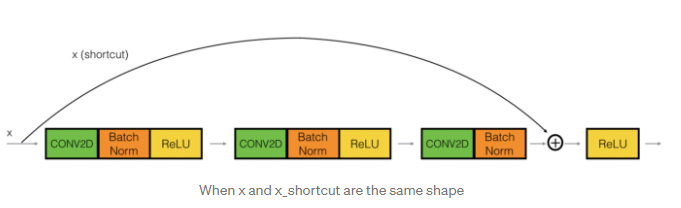
* קשה לגרום לרשת להתכנס לפתרון (בשלב האימון).
* עלולה להיווצר בעיה של היעלמות / פיצול של גרדיאנטים.
* הדיוק בהתחלה עולה עד שמגיע לרוויה ואז נופל.

כידי לטפל בבעיות עלו resnet "מדלגת על קשרים" בהנחה כי השכבות העמוקות צריכות להיות מסוגלות ללמות כמו השכבות הרדודות.

פתרון אפשרי הינו העתקת האקטיבציות מהשכבות הרדודות וקביעת שכבות נוספות למיפוי הזהויות.



תפקידם של קשרים אלו הוא לבצע פונקצית זהות על האקטיבציה של השכבת הרדודה (הראשונית), אשר בתורה תייצר את אותה אקטיבציה. לאחר מכן, הפלט מתווסף לאקטיבציה של השכבה הבאה. כדי לאפשר קשרים אלו, ובכלל לאפשר את האופרציות הנוספות הללו, יש צורך להבטיח את אותן מימדים של קונבלוציות לאורך כל הרשת. לכן למודלי Resnets יש את אותן קונבלוציות של  3 על 3 לאורך כל הדרך.



הרעיון מאחורי אותם מעקפים הוא שבמקרה “הגרוע ביותר”, השכבות הנוספות לא למדו כלום והרשת תיתן את הפרגיקציה הטובה ביותר בהתבסס על המעקפים, ובמקרה הפחות גרוע יכול להתברר כי השכבות הנוספות למדו משהו שימושי ובמקרה זה, הרי שביצועי הרשת השתפרו.

בפרויקט שלנו אנו משתמשים בResNet50 על תמונות שרק אלגוריתם אחד זיהה קודם (YOLO או MRCNN), כידי לאמת את הזיהוי. מפורט בהמשך אופן השימוש.

# פרק ד – כללי החלטה

## **כללי החלטה פשוטים**

לאחר שהרצנו את האלגוריתמים על 1000 תמונות וניתחנו את התוצאות עם כללי החלטה שונים קיבלנו את מדדי הmap הבאים:

בשימוש בתוצאות של רק אחד מן האלגוריתמים קיבלנו את המדדים הבאים:

MAP of: YOLO is: 50.67%

MAP of: MRCNN is: 62.88%

לאחר מכן ניסנו להשוות בין הפלטים של שני האלגוריתמים, ובמידה ויש חפיפה מסויימת בIOU אנו ניקח את התוצאה בעלת הConfidence הגדול יותר, ניסינו מספרים שונים עבור IOU ,ועבור קופסאות שזיהה רק אובייקט אחד תמיד לקחנו אותם ,וקיבלנו את התוצאות הבאות:

MAP of Filter: Highest Score, Take All, IOU = 0.80 is: 64.03%

MAP of Filter: Highest Score, Take All, IOU = 0.70 is: 65.65%

MAP of Filter: Highest Score, Take All, IOU = 0.60 is: 65.93%

MAP of Filter: Highest Score, Take All, IOU = 0.50 is: 65.95%

הסבר אפשרי לתוצאות : כאשר אנו מגדילים את הIOU אנו בעצם "משאירים" יותר תוצאות שנחשבים כזיהוי של אלגוריתם בודד, ואז יכול להיות שאחד האלגוריתמים טעה בזיהוי האובייקט והשני צדק ואז ירד לנו מדד ה precision ובהתאם ירד לנו הMAP

נסיון נוסף שביצענו הוא לא תמיד לקחת את כל הקופסאות אלה אם יש לנו קופסא שזיהה רק אלגוריתם אחד אנו נקח אותה רק בתנאי שהבטחון של האלגוריתם גדול מסף מסויים.

MAP of Filter: Highest Score, IOU = 0.5, Take with Confidence > 90 is: 54.90%

MAP of Filter: Highest Score, IOU = 0.5, Take with Confidence > 80 is: 61.43%

MAP of Filter: Highest Score, IOU = 0.5, Take with Confidence > 70 is: 64.79%

הסבר אפשרי לתוצאות : אנו רואים שככל שאנו מורידים את סף הבטחון אנו מקבלים תוצאות יותר טובות, אנו מסיקים מכך שכאשר אנו לוקחים כמה שיותר קופסאות מדד הMAP גדל.

כלומר נראה בסה"כ התוצאה הטובה ביותר התקבלה כאשר אנו תמיד לוקחים את כל האובייקטים ומסננים לפי IOU של 50~

וכך קיבלנו שיפור של כ 3% לעומת שימוש בכל אלגוריתם בנפרד

## **כללי החלטה בעזרת ML**

**עיבוד מקדים של המידע:**

הפלט של שני האלגוריתמים מתחלק לשני סוגים :

1. קופסאות שזוהוו על ידי שני האלגוריתמים – זוגות
2. קופסאות שזוהוו על ידי אלגוריתם אחד – בודדים

עבור כל תמונה חילקנו את התחזיות לשני הקטגוריות הנ"ל , כאשר אנו מסווגים קופסאות זהות ע"י IOU=0.5 .

החלטנו על הפרמטרים הבאים כקלט למערכת הלומדת שלנו:

<class name> <confidence> < width > <Hight> <algorithm>

כעת על מנת לאמן את עצי ההחלטה שלנו היינו צריכים ליצור Dataset המתאים לפרמטרים שלנו

יצרנו שני datasets שונים , אחד עבור הזוגות ואחד עבור הבודדים:

**Data זוגות :**

ווקטור X : מכיל את הפרטמטרים הנ"ל עבור זוגות

ווקטור Y : מכיל {0,1} 0 מסמל YOLO ו1 מסמל MRCNN

יצירת ווקטור Y : עבור זוג קופסאות השוונו את הIOU עם הקופסא של הGT ובחרנו את האלגוריתם שנותן לנו את הIOU הגבוה ביותר.

**Data בודדים :**

ווקטור X : מכיל את הפרטמטרים הנ"ל

ווקטור Y : מכיל {0,1}, 1 מסמל שהקופסא אכן קיימת בGT .

יצירת ווקטור Y: עבור כל קופסא השוונו מול הקופסאות בGT ובדקנו אם היא קיימת באותה קטגוריה

## **עצי החלטה**

השתמשנו בשני מערכות של עצי החלטה כאשר אחת היא עבור הזוגות והשניה היא עבור הבודדים.

לאחר האימון קיבלנו את מדד הMAP הבא:

MAP of Decision trees is: 56.16%

הסבר אפשרי לתוצאות : אנו רואים שעדיין קיבלנו שיפור ביחס לYOLO בלבד אך אנחנו עדיין זורקים קופסאות שאנו צריכים או לוקחים קופסאות שאנחנו לא צריכים.

יכול להיות שעבור אימון על Dataset גדול יותר היינו יכולים לשפר עוד יותר.

## **עצי החלטה ורשת נוירונים**

השתמשנו במערכת של עץ החלטה עבור הזוגות ומערכת של רשת נוירונים עבור הקופסאות הבודדים.

לאחר האימון קיבלנו את מדד הMAP הבא:

MAP of Decision tree AND NN is: 48.47%

הסבר אפשרי לתוצאות :כאן קיבלנו את התוצאות הגרועות ביותר, ואפילו פחות טוב משימוש בYOLO לבד, זה קורה כיוון שיש לנו ממש קצת פרמטרים המשמשים כקלט לרשת נוירונים שבדר"כ מקבל הרבה יותר, בנוסף גם למבנה של הרשת עצמה יש השפעה ( כמות השכבות הנסתרות וכמות הנוירונים בכל רשת) אותה יכלנו רק לנחש.

## **עצי החלטה וResnet50**

השתמשנו במערכת של עץ החלטה עבור הזוגות

ועבור קופסאות בודדות : הכנסנו את הקופסא לתוך רשת Resnet50 ובמידה והרשת זיהתה אובייקט בבטחון שגבוה מ 0.5 לקחנו את הקופסא , אחרת זרקנו

קיבלנו את מדד הMAP הבא:

MAP of Decision tree AND Resnet50 is: 51.99%

הסבר אפשרי לתוצאות : יש לנו שיפור קל מאשר שימוש רק בYOLO אך עדיין פחות טוב מMRCNN , לעיתים הקופסאות שנשלחו את תוך Resnet היו בגודל של עשרות פיקסלים בודדים ולכן Resnet זיהה פחות אובייקטים משרצינו.

שמנו לב שכאשר הקופסאות שנשלחו לResnet היו יחסית גדולות (לפחות כ 150X150) הבטחון של האלגוריתם שהוא זיהה אובייקט גדל משמעותית.

# פרק ה - סיכום

טבלאת תוצאות:

|  |  |  |
| --- | --- | --- |
|  | YOLO | MRCNN |
| Result | 5.67 | 62.88 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ALL + IOU = 0.8 | ALL + IOU = 0.7 | ALL + IOU = 0.6 | ALL + IOU = 0.5 | Confidance > 0.9  + IOU = 0.5 | Confidance > 0.8  + IOU = 0.5 | Confidance > 0.7  + IOU = 0.5 |
| Result | 64.03 | 65.65 | 65.93 | 65.95 | 54.9 | 61.43 | 64.79 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Decision Trees | Decision Tree + NN | Decision Tree + Resnet |
| Result | 56.16 | 48.47 | 51.99 |

בשילוב התוצאות של שני האלגוריתמים הגענו אכן הצלחנו לשפר ביחס לתוצאות של אלגוריתם בודד בעזרת כללי החלטה פשוטים, אך לא הצלחנו להשתמש לשפר עוד יותר את התוצאות בעזרת מערכות מתוחכמות יותר.

האתגר הגדול היה לנסות ולזהות תוצאות false positive של האלגוריתמים וזה הדבר שניסינו לשפר בעזרת המערכות המתוחכמות יותר, ייתכן שבעזרת אימון על DATASET גדול יותר ובעזרת פרמטרים מתוחכמים יותר היינו יכולים לשפר עוד את התוצאות.

# פרק ו - מקורות

* ML introduction : [https://www.coursera.org/learn/convolutional-neural-networks#syllabus](https://eur01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.coursera.org%2Flearn%2Fconvolutional-neural-networks%23syllabus&data=02%7C01%7Cadimazuz%40campus.technion.ac.il%7Cd94cd53065f542318d5608d7c452feed%7Cf1502c4cee2e411c9715c855f6753b84%7C1%7C0%7C637193733069654281&sdata=mmazIQ1D7ZcnS7EEq9SQIwXvCzkQKw5HXxsq8n6dtdY%3D&reserved=0)
* COCO dataset: https://cocodataset.org/
* YOLO: <https://github.com/arunponnusamy/object-detection-opencv>
* MRCNN: : <https://github.com/matterport/Mask_RCNN>
* MAP: <https://github.com/Cartucho/mAP>
* Sckit algorithms: <https://scikit-learn.org/>
* Restnet Tutorial : <https://medium.com/@venkinarayanan/tutorial-image-classifier-using-resnet50-deep-learning-model-python-flask-in-azure-4c2b129af6d2>