**למידה עמוקה וראייה ממוחשבת ברכבים אוטונומיים**

סתיו כהן

בהנחיית ד"ר מעינה הרמן

אוגוסט 2020

[](https://www.here.com/platform/automotive-services/roahttps:/www.here.com/platform/automotive-services/road-sign-recognitiond-sign-recognition)

**תוכן עניינים**

תוכן עניינים...................................................................................2

מבוא............................................................................................3

הנוירון הביולוגי...............................................................................7

נוירונים מלאכותיים..........................................................................9

מבנה רשת נוירונים.........................................................................13

אימון רשתות נוירונים.......................................................................14

רשתות קונבולוציה..........................................................................24

ארכיטקטורת רשת קונבולוציה...........................................................26

התאמת יתר ובחירת נתונים..............................................................31

כיצד עובדים רכבים אוטונומיים.........................................................34

זיהוי אובייקטים...............................................................................36

מעקב אחרי אובייקטים.....................................................................44

סיגמנטציה של תמונה......................................................................48

בעיית זיהוי התמרורים ברכבים אוטונומיים...........................................53

סיכום ומסקנות................................................................................62

ביבליוגרפיה....................................................................................64

**מבוא**

בינה מלאכותית – אחת המילים המושמעות ביותר בשנת 2020, בינה מלאכותית הינה תחום חדש יחסית בעולם מדעי המחשב אשר תחילת מחקרו התחיל בשנות ה50.

מתחילת מחקרה בינה מלאכותית שולבה באין ספור מערכות ברחבי העולם וכיום ניתן למצוא שימושים של בינה מלאכותית בשלל תחומים לדוגמא: רפואה, זיהוי ועיבוד שפה, רובוטיקה, ראייה ממוחשבת, עיבוד נתונים ועוד.

אז מה זה בעצם בינה מלאכותית?

בינה מלאכותית היא "לגרום למכונה להתנהג בדרך שהייתה נחשבת לאינטילגנטית לו אדם התנהג כך" – מרווין לי מינסקי- אחד מאבות תחום הבינה המלאכותית.

תחום הבינה מלאכותית מתחלק לשני קטגוריות מרכזיות:

* בינה מלאכותית "רחבה" : בינה מלאכותית המתמקדת בבניית בינה אינטיליגנטית, חכמה ברמת האינטיליגנציה האנושית אשר מסוגלת לפתור בעיות באופן כללי.

גישה זו מאתגרת מאוד ולכן בינה מלאכותית זו לא תושלם בעשורים הקרובים[6].

* בינה מלאכותית "מצוצמת" : בינה מלאכותית המתרכזת בביצוע מצויין של משימה אחת מצומצמת. בינה מלאכותית זו נראית חכמה אך היא פועלת תחת הנחות והגבלות מקלות מאוד.

גישה זו היא הגישה השולטת כרגע, בשיטה זו נראית התקדמות משמעותית במחקר ופיתוח (לעומת הגישה הרחבה).

גישה זו מיושמת בתחומים רבים אך אתמקד דווקא באחת המשימות הגדולות של העשור והיא **רכבים אוטונומים ומערכות ראייה ממוחשבת**.

**רכבים אוטונומיים**

רכב אוטונומי הוא רכב אשר יכול לנסוע, לנווט בכבישים תוך כדי התמודדות עם תנאי סביבה אשר משתנים ללא צורך בהתערבות אקטיבית של נהג אנושי.

רכב אוטונומי צריך לדעת לראות, לנתח את הסביבה שבה הוא נמצא ולדעת לקבל החלטות בזמן אמת לפי הנתונים שהוא מקבל מהסביבה.

על מנת לאפשר לרכב אוטונומי לחוות את סביבתו צריכים לצייד את הרכב בחיישנים רבים כמו מצלמות סורקים ועוד. לאחר מכן הרכב צריך לקבל החלטות כיצד לפעול בהתאם למידע שקיבל מהחיישנים, בשלב זה מחשב הרכב נעזר באלגוריתמים רבים אשר חלקם הם אלגוריתמים מבוססים למידת מכונה וראייה ממוחשבת.

זה לא סוד שבשנים האחרונות תחום הרכבים האוטונומיים קיבל תשומת לב רבה ומספר לא קטן של חברות גדולות (גוגל,אובר,טסלה וכו) החליטו להשקיע משאבים וזמן בפיתוח תחום זה[5].

האתגר בפיתוח רכבים אוטונומיים הוא לא רק עניין של פיתוח טכנולוגיה אלא אתגר בעל פן בטיחותי ואנושי נרחב.

כיום תחום המכוניות אוטונומיות הוא אחד התחומים החמים הנעזר בטכנולוגית הבינה מלאכותית ונמצא בשלבי פיתוח מתמיד.

תחום המכוניות האוטונומיות התקדם במהלך השנים האחרונות מזיהוי תמרורי תנועה ועד הטכנולוגיה של מכוניות המסוגלות לנסוע בכוחות עצמן.

בעוד מכוניות אוטונומיות באות לפתור לנו בעיות רבות בחברה הן מביאות איתן אתגרים טכנולוגים, אתים, בטיחותיים ועוד.

[](https://rotter.net/forum/scoops1/422783.shtml)

**האתגרים המרכזיים ברכבים אוטונומיים**

* ביטחון: אחד האתגרים המרכזיים ביותר בפיתוח הרכבים האוטונומיים הינו רמת הבטחון ליושבים ברכב ולסביב [11].

לפי מחקר של World Health Organization [11] הסיבה המרכזית למוות בקרב בני 15-29 הינה תאונות דרכים. בעוד שמערכות הביטחון ברכבים נהיות יותר ויותר חכמות, הרכבים האוטונומיים עדיין צריכים לדאוג לשלום הולכי הרגל שאינם מוגנים באמצעות כריות אוויר ואמצעים משוכללים אשר נמצאים בתוך הרכב.

מכונית אוטונומית תצטרך לספק מידת ביטחון ושליטה על הרכב בדומה לרמת הביטחון בנהיגה של בני אדם ואפילו לעקוף ולהתעלות על רמת ביטחון זו.

* נוחות: בני אדם אוהבים נוחות, האם תעדיפו נהג פרטי או לנהוג בעצמך ?

כיום אנשים יעדיפו לנוח, לדפדף בטלפון, לראות סרט, במקום לחפש חניה באזור עמוס. אתגר יהיה לאפשר לבן האדם לבצע פעולות נוספות בצורה נוחה במקום להתרכז במשימת הנהיגה וכמובן למנוע את הצורך בביצוע פעולות במקביל לנהיגה.

* אימוץ ואמונה: במידה וקיבלתם מכונית אוטונומית חדשה האם תאפשרו לה לנהוג במקומכם ?

כיצד מכונית אוטונומית תתקבל בסביבה? אנשים יפחדו ממנה?

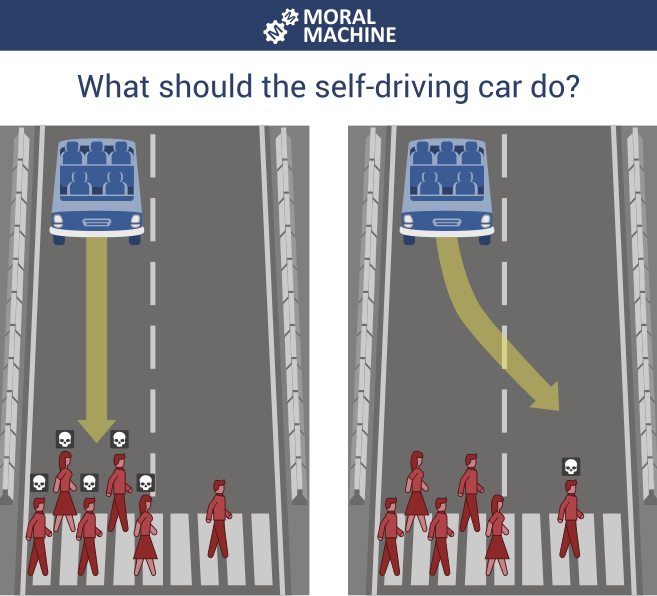
אחד האתגרים יהיה לגרום לבני האדם לסמוך על מכונית אוטונומית ולתת בה אמונה יקרת ערך שהיא שליטה מלאה על חייהם.

* בעיות מוסר: אחת הבעיות המרכזיות שעולות הינה הבעיה בשם:

"who dies and who lives?" ניסוי המוכר הנערך [בMIT בנושא רכבים אוטונומים](https://www.technologyreview.com/2018/10/24/139313/a-global-ethics-study-aims-to-help-ai-solve-the-self-driving-trolley-problem/)[7]:

אחד התרחישים מציג בעיה אשר מכונית נוסעת בכביש וצריכה לבחור בין שני מסלולים באחד המכונית תדרוס אדם אחד מול דריסה של 5 בני אדם, באיזה נתיב המכונית תבחר?

הבעיה מתגברת כאשר מציגים אותו תרחיש עם בחירה בין אדם מול חיה, גבר מול אישה, מבוגר מול תינוק. ניתן לראות תיאור של דילמה זאת בתמונה מספר 1.

[](https://www.technologyreview.com/2018/10/24/139313/a-global-ethics-study-aims-to-help-ai-solve-the-self-driving-trolley-problem/)

תמונה מספר 1: המציגה את בעיית המוסר[7].

* רגולציה: אתגר מדיני יהיה לקבוע היכן עומד הרכב אוטונומי כחוק, האם אדם הנמצא ברכב אוטונומי לחלוטין שדרס ילד יעמוד לדין? האם החברה שיצרה את הרכב תעמוד לדין? זה בהחלט אתגר מעניין אשר ימתוח את גבולות הרגולציה בכל מדינה.
* יעילות: בעיית הפקקים הולכת וגוברת בכל רחבי העולם, תשתיות הכבישים לא מצליחות לעמוד בקצב הגדילה של האוכלוסייה ובמקביל גידול בשימוש ברכבים.

האם מכוניות אוטונומיות יוכלו לעזור בפתרון הבעיה?

מכוניות אוטונומיות יוכלו לייעל את תהליך מציאת החנייה אם יוכלו לתקשר בין עצמן, נוכל לייצר רמזורים חכמים שידעו לווסת צמתים לפי כמות הרכבים המתקרבת, נוכל למנוע את בעיית ה [Phantom Traffic Jam](https://www.youtube.com/watch?v=7wm-pZp_mi0)?

אתגרים רבים עומדים בתחום היעילות של הרכבים האוטונומיים.

* טכנולוגיה: איזה מערכות-סנסורים צריכות להיכלל ברכב אוטונומי, איך מאבטחים רכב אוטונומי שיהיה מאובטח מבחינת סייבר, תמחור רכבים לקהל הרחב.

לבסוף נשארת השאלה כיצד נצליח לתכנן מכונית אוטונומית אשר תוכל לעמוד במשימה המורכבת של נהיגה טובה, רגועה, איכותית, בטוחה והיכן בינה מלאכותית תיכנס כאן לעזרתנו.

בסמינר זה אתמקד במערכת ראייה ממוחשבת ברכבים אוטונומיים, אציג כיצד טכנולוגית הבינה מלאכותית עובדת בתחום הראייה ממוחשבת וכיצד היא עוזרת לנו לפתור משימות מורכבות בראייה ממוחשבת.

לבסוף אתמקד במימוש מערכת לזיהוי תמרורי דרך הנמצאת בכל רכב אוטונומי כיום.

**הנוירון- מביולוגיה באנו לטכנולוגיה נגיע**

המוח האנושי הוא האיבר החשוב ביותר בגוף האדם ומטרתו לקבל מידע ממערכות החושים של האדם, לעבד אותו ולאחר מכן להשתמש במידע זה על מנת לקבל החלטות.

המוח שולט בכל הפעולות שהאדם - האורגניזם עושה ולכן זו אינה הפתעה שהמדע החליט לנסות להתחקות אחרי מערכת מרכזית זו. המוח שלנו שוקל כ1.5 קילו בלבד ובתוכו נמצאים כ86 ביליון נוירונים [[1].](https://search-ebscohost-com.elib.openu.ac.il/login.aspx?direct=true&db=nlebk&AN=1607842&site=eds-live&ebv=EB&ppid=pp_7)

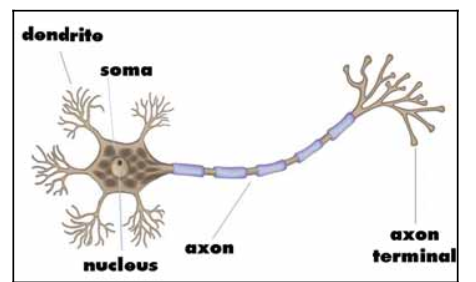
נוירון הוא הגדרה לתא המקבל ושולח פולסים עצביים ואותות אלקטרוכימיקלים. מוחנו בנוי מרשתות נרחבות של נוירונים מחוברים אשר בעזרתם הוא מעבד אינפורמציה רבה וללא הפסקה.

בשל מורכבותו הגדולה של המוח, עדיין איננו בטוחים כיצד הוא בדיוק עובד ונושא זה עדיין מהווה נושא מחקר משמעותי בכל מרחב הגלובוס, אך תודות להתקדמות הטכנולוגיה כיום אנו יכולים להצליח לחקות חלק מהפעולות שהמוח עושה וביניהן פעולת רשת הנוירונים.

תחום המחקר של Artificial Neural Networks החל בשאיפה להתחקות אחר פעולותיו של המוח ולנסות לפתור בעיות בצורה שבה מוחנו פותר אותן. רשתות הנוירונים המלאכותיות השתפרו משמעותית במהלך שנות המחקר אך עדיין חיקוי מלא של פעולות המוח האנושי נראה כשאיפה ארוכת טווח.

**ההשראה ליצירת רשתות נוירונים**

ניתן להסתכל על גוף האדם כמכונה בעלת יחידת עיבוד – המוח, ובעלת קולטני סביבה רבים – חושי האדם. מוחנו מעבד כמויות עצומות של אינפורמציה המתקבלת באמצעות חושי האדם (לדוגמא חוש הראייה מאפשר לנו לדעת כיצד נראית סביבתנו). תהליך עיבוד המידע מבוצע על ידי הנוירונים אשר מעבירים ביניהם אותות אלקטרונים המנוהלים לפי לוגיקה של Flip-flop כמו פתיחה וסגירה של חיבורי הנוירונים לשם העברת מידע[1].

[](https://eds-a-ebscohost-com.elib.openu.ac.il/eds/ebookviewer/ebook?sid=f196c952-10ac-4105-b8ff-ce4e44fce642@sessionmgr4007&ppid=pp_Cover&vid=0&format=EB)

תמונה מספר 2: מבנה הנוירון[1].

תמונה מספר 2 מתארת את מבנה הנוירון. החלקים המרכזים בנוירון הם:

Dendrite- נקודת הכניסה של כל נוירון אשר דרכה הוא מקבל קלט משאר הנוירונים ברשת, אשר ממיר את המידע לאות חשמלי ומעביר אותו לגוף התא.

Cell Body- גוף התא מקבל את המידע החשמלי מה-Dendrite וקובע איזה פעולה לבצע.

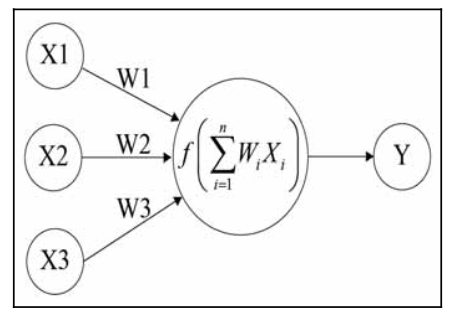
Axon terminal – מתפקד בתור המוליך של המידע מגוף התא עד העברת המידע לנוירונים הבאים בצורת אות חשמלי.

חשוב לציין כי נוירונים יכולים להעביר מידע בצורה של אות בינארי בלבד משמע יש בינהם בחירה לשדר מידע (1) או לא לשדר דבר (0(.

**הנוירון המלאכותי הבסיסי – perceptron**

Perceptron – הינו נוירון מלאכותי אשר פותח בשנים 1950-1960 על ידי המדען [Frank Rosenblatt](https://en.wikipedia.org/wiki/Frank_Rosenblatt) במטרה להתחקות אחר פעולת הנוירונים במוח האנושי[[8](http://neuralnetworksanddeeplearning.com/chap1.html)].

נוירון זה הוא הבסיסי ביותר.



תמונה מספר 3: המתארת מבנה נוירון מלאכותי [1]

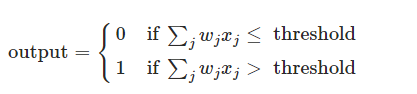
נוירון זה כמתואר בתמונה מספר 3, מקבל מידע בצורה בינארית - 0 או 1 מכמה מקומות שונים (x1, x2, …) בצורת וקטור, מעבד את המידע ומוציא פלט יחיד המיוצג בצורה בינארית - 0 או 1.

עיבוד המידע מתבצע בעזרת וקטור ערכים הנקרא משקולות Weights (w1, w2,…) אשר מהווה וקטור של סקלרים המתואמים מול וקטור הקלט, וקטור המשקלים מייצג את חשיבות הקלט התואם שלו.

הנוירון לא עושה דבר חוץ מהכפלת איברי הוקטורים (ללא פונקציית אקטיבציה):



ומחזיר תוצאה סופית אשר נבדקת מול ערך סף Threshold הידוע מראש אשר קובע האם הנוירון ישדר 0 או 1 לנוירונים הבאים[7]:

[](http://neuralnetworksanddeeplearning.com/chap1.html)

לצורך הדגמה פשוטה נוכל להסתכל על אותו נוירון ולשאול את השאלה הפשוטה: " האם לצאת לריצה " ?

נוכל להציב את הערכים הבאים בוקטורים

(x1=Sunny, x2= Healthy, x3= Energized) , (w1=0.30, w2=0.50, w3=0.20)

Threshold=0.65

נניח קיבלנו את הוקטור הבא כקלט מהנוירונים הקודמים:

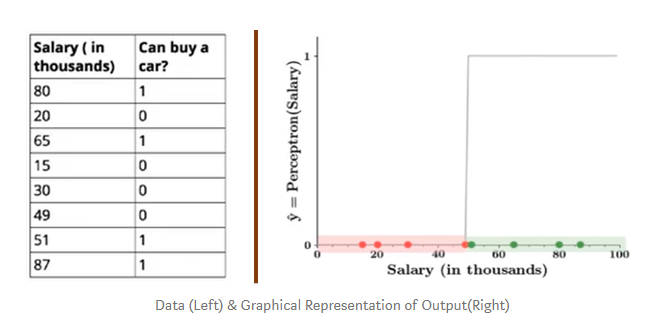
)x1=1, x2=0, x3=1( \* (0.30,0.50,0.20)= 0.50

לכן קיבלנו כי 0.50<0.65 אז נבחר לא ללכת לריצה.

אך אם הThreshold שלנו היה נקבע להיות 0.40 היינו בוחרים כן ללכת לריצה.

**Sigmoid Neuron**

נוירון זה בא לפתור כמה בעיות שהתגלו בנוירון הבסיסי, המרכזית ביניהן הינה סף ה Threshold אשר גרם להתנהגות תמוהה[[9](https://towardsdatascience.com/sigmoid-neuron-deep-neural-networks-a4cd35b629d7)]:

[](https://towardsdatascience.com/sigmoid-neuron-deep-neural-networks-a4cd35b629d7)

תמונה מספר 4: המתארת גרף החלטה [9]

ניתן לראות באיור מספר 4, המדגים קבלת החלטה האם ניתן לקנות מכונית ע"פ גובה השכר, כיצד מתקבלת החלטה בהתאמה ל Threshold . Threshold העומד על סה"כ 50 אלף הקובע האם ניתן לקנות מכונית או לא, דבר זה יוצר החלטה תמוהה אשר בנאדם יכול לקנות מכונית עם 50.1K משכורת אך אינו יכול עם 49.9K. הבדל קטן כמו זה יכול לגרום לנוירון לשנות את החלטתו מ0 ל1 או להפך. נוכל להתגבר על בעיה זו באמצעות תכנון שונה של הנוירון, נוירון כזה נקרא Sigmoid Neuron .

השינוי הראשון שמתבצע בנוירון זה הוא הפיכת הThreshold לפרמטר bias בצורה הבאה:

Bias=- Threshold

נשנה את פונקציית הפלט לצורה הבאה:

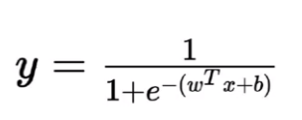
W \* X + bias = Output

פרמטר ההטיה Bias הינו סוג של מדד אשר מטרתו היא לאפשר לפונקציית האקטיבציה תזוזה חיובית או שלילית לשם התאמה טובה יותר לנתונים – במושגים ביולוגים כמה קל לגרום לנוירון זה להעביר זרם חשמלי.

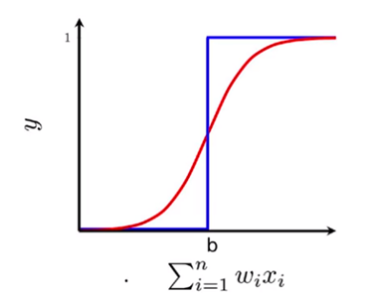
ככל שה Bias הוא מספר גדול יותר כך הנוירון יקבל ערכים גדולים בתדירות גבוהה יותר.

השינוי השני הינו שינוי פונקציית האקטיבציה של הנוירון שהיא הפונקציה הפועלת על הקלט שהתקבל לאחר עיבודו עם המשקלים וה Bias- בתמונה 2 ניתן לראות אותה כמסומנת באות F():

כעת נשתמש בפונקציה המוכרת לכולנו והיא הפונקציה הלוגיסטית. כאשר Y יהיה ה-Output של הנוירון.



פונקציית הSigmoid כעת תראה יותר "חלקה" כמתואר באיור מספר 4, באדום אל מול הפונקציה הקודמת בה השתמשנו בכחול.

[](https://towardsdatascience.com/sigmoid-neuron-deep-neural-networks-a4cd35b629d7)

תמונה מספר 5: המתארת השוואה בין הגרפים [9]

ניתן לראות באיור 5, הפלט של נוירון הSigmoid שלנו יהיה מספר בתחום [0,1] אשר יכול להתפרש כהסתברות ולא כהחלטה בינארית.

למראית-עין פלט זה יכול להיראות בעייתי בסיטואציות בהן מנסים לקבוע מצב במפורש לדוגמא : האם קלט של תמונה מסוימת מייצג את המספר 5 ? לפעמים נרצה תשובה חד משמעית, ונוכל לפתור זאת בנקל באמצעות קביעה שאם הפלט גדול מ0.5 התמונה תקבע כ-1 ואם הפלט קטן מ0.5 אז התמונה תקבע כ-0 .

כעת אם נסתכל על ההחלטה שתוארה קודם, "האם לצאת לריצה?", החישוב ישתנה.

(x1=Sunny, x2= Healthy, x3= Energized) , (w1=0.30, w2=0.50, w3=0.20)

Bias =0.05

כעת נוכל לקבל וקטור "מעניין" יותר מאשר וקטור בינארי:

) x1=0.65, x2=0.83, x3=0.14( \* (0.30,0.50,0.20) + 0.05 = 0.688

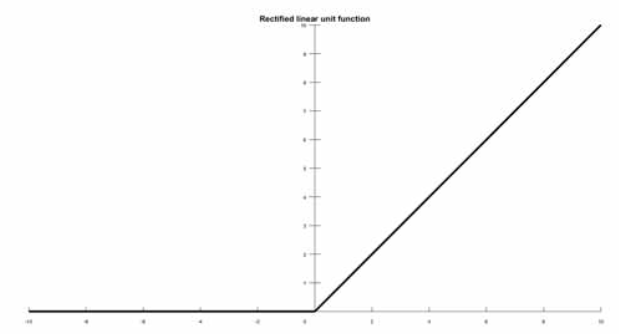
ונראה כי הפעם יש סבירות גבוה יותר שכן נצא לרוץ אך זו עדיין אינה החלטה סופית (אלא אם כן נגדיר קביעת Threshold לפלט(.

**Rectified Linear Unit** (ReLu)

ReLu הינה פונקציית אקטביציית הנוירון הפופולרית ביותר כיום[1]

כאשר בפונקציה זו האות Z מייצגת את המספר המתקבל לאחר עיבוד הקלט עם ה Weights- וה-Bias

F(Z) = Max(0,Z)

[](https://search-ebscohost-com.elib.openu.ac.il/login.aspx?direct=true&db=nlebk&AN=1607842&site=eds-live&ebv=EB&ppid=pp_18)

תמונה מספר 6: המתארת את פונקציית ReLu [1]

פונקציה זו בתמונה מספר 6 באה לפתור בעיות נוספות אשר התגלו בנוירון ה – Sigmoid לדוגמא:

* בעיית החישובים – כיוון שהמודל הלוגיסטי מבצע חישובים אקספוניציאלים, סיבוכיות זמן הריצה גבוהה לעומת פונקציית ReLu המשתמשת בחישוב פשוט יותר.
* בעיית ה True Zero – פונקציית ה- Sigmoid לא יכולה להוציא לנו 0 אמיתי אלא רק ערך קרוב מאוד ל0, לעומת זאת ReLu יכולה להוציא 0 אמיתי אשר יגרום

לרשת נוירונים "מפוזרת" יותר, תכונה אשר יכולה לפשט ולהאיץ את אימון המודל.

לסיכום ReLu הינה פונקציית אקטיבציה פשוטה, מהירה יותר ולכן הראתה שיפורים ניכרים לעומת פונקצית ה – Sigmoid[1] דבר זה הוביל אותה להיות פונקציית האקטיבציה הפופולרית ביותר כיום.

חשוב לציין כי לפונקציית הReLu ישנן גירסאות נוספות ומתקדמות יותר אשר לא אסקור כאן בפירוט.

**מבנה רשת נוירונים**

בדומה לרשתות הנוירונים במוח האנושי גם כאן נתחקה על המבנה ונבנה רשת נוירונים מלאכותית הבנויה משכבות של נוירונים המחוברות אחת לשנייה.[[8](http://neuralnetworksanddeeplearning.com/chap1.html)]

[](http://neuralnetworksanddeeplearning.com/chap1.html)

תמונה מספר 7: המתארת רשת נוירונים [8]

**Input Layer**- הינה השכבה הראשונה ברשת אשר מקבלת את וקטור הקלט ומתחילה את תהליך העיבוד של הרשת.

**Output Layer** – הינה השכבה האחרונה ברשת אשר מקבלת את התוצאה הסופית של עיבוד הנתונים ברשת, שכבה זו מוציאה את הפלט הסופי של הרשת.

**Hidden Layers** – הינן שכבות של נוירונים אשר נמצאות במרכז הרשת ואינן שכבות Output או Input, מספר שכבות אלה יכולות להשתנות בהתאם לארכיטקטורת הרשת.

**נקודות עניין באיור מספר 7:**

* שוני בשכבות הנוירונים - בדרך כלל מספר הנוירונים בשכבת הInput יהיה שווה למספר הפרמטרים בוקטור הקלט שהרשת מקבלת, מספר הנוירונים בשכבת ה- Output נקבע לפי כמות הסיווגים השונים אשר נרצה להציג, אך מספר הנוירונים ומספר השכבות בשכבות הHidden יוכל להשתנות מארכיטקטורה לארכיטקטורה.
* ניתן לראות באיור שכל נוירון מחובר לכל נוירון בשכבה הבאה ברשת ומספר הנוירונים הולך וקטן עד סוף הרשת, גם כאן מבנה זה תלוי בארכיטקטורת הרשת.
* ישנן אין ספור ארכיטקטורות שונות של רשתות נוירונים ואסקור בהמשך כמה מהן.

**אימון רשתות נוירונים**

תהליך אימון רשתות הנוירונים הוא תהליך רפטטיבי אשר מבוצע באמצעות מסד נתוני אימון מתויג אשר גורם לרשת ללמוד דפוסים חוזרים במסד הנתונים באמצעות התאמת הWeights ו- Bias של כל הנוירונים ברשת[1].

מסד נתונים מתויג הינו מסד נתונים אשר כל שורת מידע בו מתויגת לפי תווית מסוימת אשר קובעת את סיווג השורה. בחזרה לדוגמא הקודמת, שורת מידע מתויגת יכולה להיראות כך:

|  |  |  |  |
| --- | --- | --- | --- |
| Sunny | Healthy | Energized | Went for a Run? |
| 0.65 | 0.83 | 0.40 | YES |

שורת מידע זו מעידה על כך שאדם יצא לריצה כאשר Sunny, Healthy, Energized היו הנתונים המתועדים לפני הריצה.

Forward Propagation- הינו התהליך אשר רשת הנוירונים מקבלת נתונים כקלט לשכבת הקבלה ומעבירה אותם לשכבות הנסתרות עד הגעתם לשכבת הפלט, כמתואר בתמונה מספר 8.

כל נוירון מבצע חישוב של Sum(Input \* Weights) +Bias לפי פונקציית האקטיבציה שלו ולאחר מכן מעביר את התוצאה לשכבה הבאה בתור.

ברגע שהמידע מגיע לשכבת האחרונה (שכבת הפלט) נוכל לחשב את גודל השגיאה באמצעות חישוב פשוט: פלט הרשת - סיווג השורה = Error Size .

אנו משתמשים בError Size לשם תיקון ה-Weights ו- Biases בהם אנו משתמשים לחיזוי התגיות.



תמונה מספר 8: תהליך למידת הרשת [1]

Backpropagation – הינו התהליך השני בלמידת הרשת אשר מתבצע לאחר שהסתיים המעבר הקדמי וחישבנו כבר את ה Error Size.

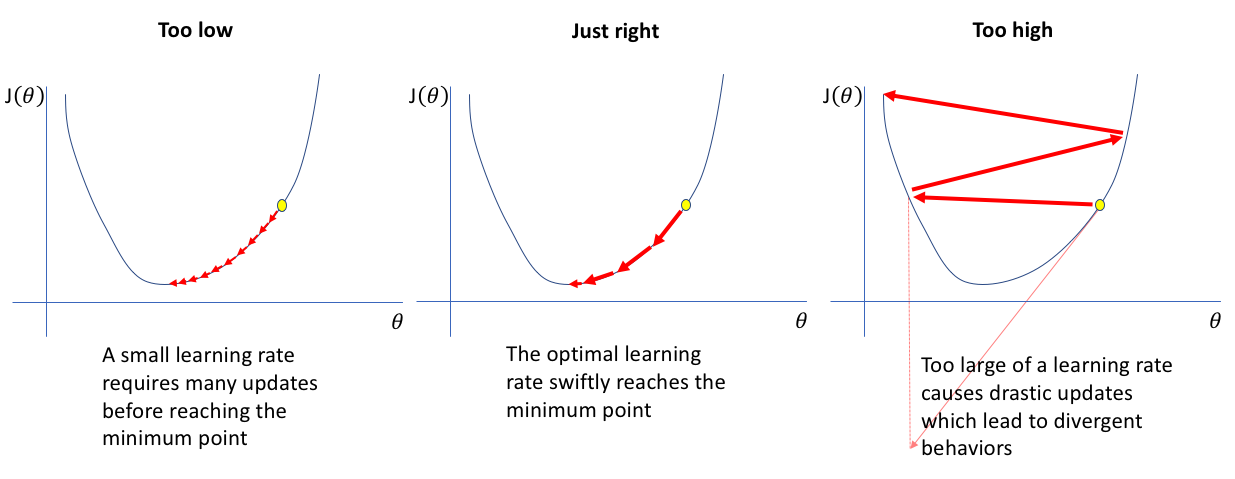
תהליך המעבר האחורי משתמש בנגזרת החלקית של פונקציית אקטיבציה של כל נוירון על מנת לזהות את השיפוע ( או ה- Gradient) בכיוון של כל אחד מהמשקלים הנכנסים.

ה –Gradient גרדיאנט מציג כמה גודל השגיאה Error Size יגדל או יקטן כפונקציה של שינוי המשקולות weights בנוירון.

תהליך הפעפוע אחורה Backpropagation ממשיך לשנות את המשקולות של כל נוירון עד אשר מגיעים להפחתה גדולה בגודל השגיאה וזאת בהתאם לקצב הלמידה.

Learning Rate הינו פרמטר סקלרי אשר קובע את גודל הצעד אשר האלגוריתם מבצע על וקטור ה – Weights לשם הקטנת גודל השגיאה Error Size - .

Learning Rate גבוה יגביר את קצב האלגוריתם וזמן האימון יקטן, אך אם נקבע פרמטר זה כגבוה מדי, יכול להיות שנעשה צעדים גבוהים מדי בכיוון מסוים ונפגע דווקא באימון המודל.

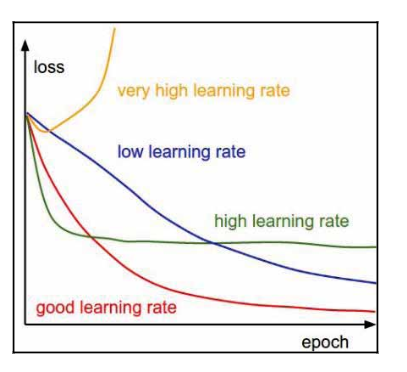


תמונה מספר 9: ערכים שונים לקצב לפרמטר קצב הלמידה [https://bit.ly/3ou5Qu8]

ניתן לראות בתמונה מספר 9 את ההשלכות של קביעת פרמטר קצב הלמידה לערכים שונים, מטרתנו היא הקטנת גדול השגיאה – Error Size ולהגיע לנקודת המינימום של פונקציית ה- Error Size :

* אם נבחר פרמטר למידה נמוך מדיי יקחו לנו הרבה צעדים להגעה לנקודת המינימום וזמן האימון שלנו יתארך.
* אם נבחר פרמטר למידה גבוה מדיי אז כל צעד שלנו יוכל להיות גדול מדיי, כתוצאה מכך נוכל לדלג על נקודת המינימום ולא להצליח להתכנס לנקודת המינימום של הפונקציה.

לכן הבחירה של פרמטר למידה הינה קריטית ליכולת האימון של המודל.



תמונה מספר 10: המתארת את תהליך אימון הרשת לפי ערכים שונים של ה Learning Rate [1]

בתמונה מספר 10 ניתן לראות גרפים המקושרים לערכים שונים של הLearning Rate,

ציר ה – X של הגרף מציג את המושג Epoch.

* Epoch הוא פרמטר אשר מגדיר את מספר הפעמים בו אלגוריתם הלמידה יעבור במלואו על כל מסד נתוני האימון, משמע שכל אחת מהרשומות הנמצאות במסד הנתונים תעבור סבב אימון שלם של הרשת (הכולל ForwardPropagation וגם Backpropagation ) וכתוצאה מכך תשפיע על המשקולות ברשת.

בדרך כלל משתמשים בכמה מעברים כאלו בשלב אימון הרשת.

ניתן לראות כי בחירת הסקלר המתאים לפרמטר Learning Rate יכולה להשפיע רבות על תוצאות אימון המודל.

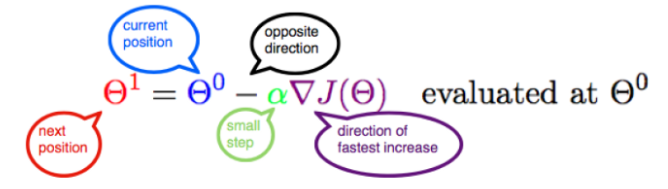
**Gradient Descent**

אלגוריתם זה הוא בעל גישה איטרטיבית לתיקון שגיאות המתאים לכל מודל למידה.

אלגוריתם זה בא לשימוש בשלב אימון רשתות הנוירונים בזמן מעבר

ה– Backpropagation, בו האלגוריתם מעדכן את ה Weights ו-Biases.

המטרה המרכזית של האלגוריתם היא להגיע למינימום של פונקציית השגיאה[10].

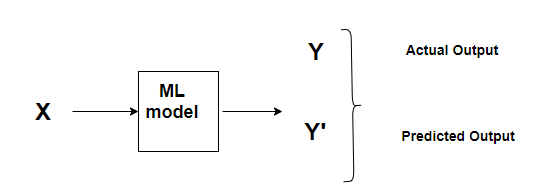


תמונה מספר 11: אלגוריתם ה Gradient Descent [10]

בתמונה 11 ניתן לראות את הנוסחה הסופית של הGradient Descent. אך כיצד הגענו אל נוסחה זו? נתחיל מההתחלה.

לאחר בניית מודל למידת המכונה מתחילים לבצע את שלב האימון של המודל:

מכניסים למודל מידע מתויג מראש כקלט ובודקים איזה תיוג המודל פלט כמתואר בתמונה מספר 12.



תמונה מספר 12: תהליך הלמידה [10]

לאחר מכן מחשבים את השגיאה של המודל באמצעות הנוסחה:

**Error=Loss Function= Y’(predicted) – Y(Actual)**

לפונקציה זו נקרא Loss Function כיוון שהיא מחשבת את השגיאה עבור קלט אחד ממסד נתוני האימון, לעומת הפונקציה Cost Function אשר מחשבת את ממוצע השגיאה עבור כל הקלטים ממסד נתוני האימון. נניח שיש לנו N קלטים-רשומות במסד האימון אז נוכל לקבוע את פונקציית המחיר Cost Function להיות:

Cost=

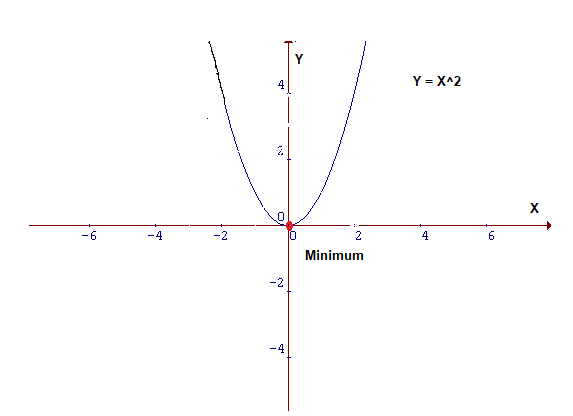
ה Cost Function בעצם אומרת לנו כמה "טוב" המודל שלנו בחיזוי הנתונים.

המטרה של כל אלגוריתם למידת מכונה הוא להקטין כמה שאפשר את הCost- Function כיוון שככל שהשגיאה נמוכה יותר אז המודל שלנו למד טוב יותר את הנתונים.

אז איך בעצם אנו מוצאים את המינימום של כל פונקציה?

נסתכל על דוגמא פשוטה של Cost Function שהיא מהצורה Y=X^2

משוואה זו היא פרבולה וקל להציגה באופן גרפי הבא, בתמונה מספר 14:

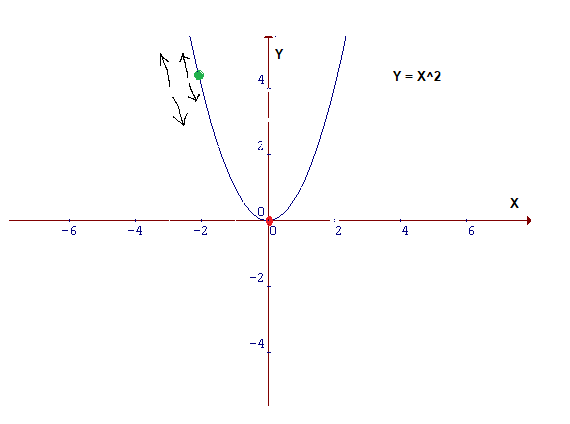


תמונה מספר 13: הפרבולה Y=X^2 [10]

כדי להגיע לנקודה הקטנה ביותר בפונקציה זו צריך למצוא את הערך של X אשר יתן לנו את הערך הנמוך ביותר של Y, קל לראות כי ערך זה הוא 0 המסומן בנקודה אדומה בתמונה מספר 13.

משימה זו דיי פשוטה לביצוע כיוון שזה גרף דו ממדי אבל זה לא תמיד המצב.

משימה זו נהפכת להיות מורכבת יותר כאשר אנו מתעסקים במרחבים עם מימד גבוה יותר ולכן אנו זקוקים לאלגוריתם מיוחד למציאת נקודת המינימום וכמו שכבר ניחשתם אלגוריתם זה הוא ה – Gradient Descent .

[](https://blog.paperspace.com/content/images/size/w2000/2019/09/F1-02.large.jpg)

תמונה מספר 14: הפרבולה Y=X^2[10] , [תמונה מספר 15: מרחב תלת מימדי](https://blog.paperspace.com/part-2-generic-python-implementation-of-gradient-descent-for-nn-optimization/)[https://bit.ly/2RpSOOM].

ניתן להסביר את האינטואיציה מאחורי אלגוריתם זה באמצעות תמונה מספר 14: דמיינו שאתם הנקודה הירוקה אשר מטיילת על הגרף הדו ממדי. מטרתכם היא להגיע לנקודה המינימלית ביותר -הנקודה האדומה אך מהיכן שאתם נמצאים אין לכם אפשרות לראות את הנקודה האדומה.

האופציות היחידות שלכם הם:

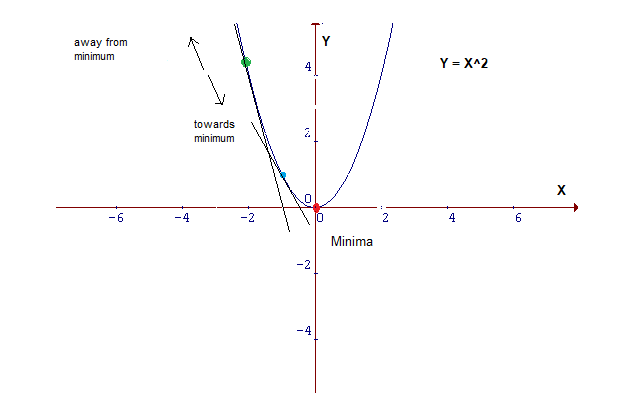
* ללכת למעלה או למטה לפי גרף הפונקציה.
* אם בחרתם ללכת לאחד הכיוונים אז האם אתם רוצים לקחת

צעדים גדולים או קטנים – Learning Rate.

תמונה מספר 15 מציגה את אותה הגדרה של הבעיה אך כעת הסביבה היא תלת מימדית וניתן ללכת גם לצדדים ולא רק לצעוד למעלה או למטה.

אלגוריתם ה-Gradient Descent עוזר לנו לקבוע איזה צעד נעשה ובאיזה גודל באמצעות שימוש של נגזרת, מושג זה מוכר לנו מחדו"א ומשמעו הוא לחשב את המדרון של הגרף בנקודה מסוימת.

המדרון מתואר באמצעות ציור של קו משיק לגרף בנקודה שאותה אנו בוחנים. אם נוכל לחשב את הקו המשיק נוכל לגלות את הכיוון אשר נצטרך לקחת כדי להגיע אל נקודת המינימום.



תמונה מספר 16: הקווים המשיקים לנקודות בגרף Y=X^2 [10]

ניתן לראות בתמונה מספר 16 שלאחר מיקום המשיק לנקודה הירוקה אנו יודעים שאם נתקדם למעלה נתרחק מנקודת המינימום ואם נתקדם למטה נתקרב אל נקודת המינימום.

הנגזרת – הנגזרת משמשת את אלגוריתם ה- Gradient Descent בעת שינוי ה- Weights לשם הקטנת ה-.Cost Function כעת נצלול עמוק אל עולם הנגזרות כדי להבין כיצד כל התהליך הזה באמת מתרחש[8].

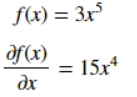
לשם פישוט המתמטיקה והבהרת האלגוריתם המרכזי נשתמש בפונקציה ליניארית

**y=Mx + b**  אשר נחפש את ה-Gradient שלה.

ראשית נגדיר כמה קונספטים חשובים להמשך הדרך:

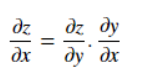
* Power Rule – כלל החזקה מחשבת את הנגזרת של פרמטר בעל חזקה:

פונקציה מהצורה -  אשר נבצע עליה את פעולת הגזירה תהיה שווה ל- 

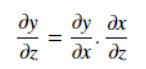
לדוגמא אפשר לקחת את הפונקציה הבאה ולגזור אותה: 

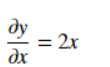
* Chain Rule – כלל השרשרת משומש לחישוב נגזרת של פונקציות מורכבות:

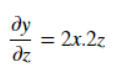
אם יש לנו משתנה Z אשר תלוי במשתנה Y, כאשר המשתנה Y תלוי במשתנה X אז המשתנים Z ו Y תלויים במשתנה X, כלל זה נקרא כלל השרשרת והוא נכתב כך:



לדוגמא אם יהיו לנו הפרמטרים הבאים: Y=X^2 , X=Z^2 נוכל לחשב את הנגזרת של Y לפי Z באמצעות כלל השרשרת:



נוכל לגזור כל משתנה לחוד : , 

ונקבל סה"כ כי : 

* Partial Derivatives – נגזרת חלקית, במידה ויש לנו פונקציה עם 2 משתנים אז נוכל לחשב את הנגזרת החלקית של הפונקציה לפי אחד המשתנים. נעשה זאת באמצעות התייחסות למשתנה השני כסקלר קבוע.

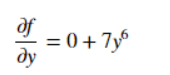
לדוגמא ניקח את הפונקציה:

**F(x,y)=x^4 +y^7**

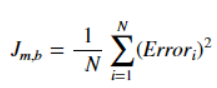
מציאת נגזרת חלקית של הפונקציה F לפי x תהיה:

 כאשר התייחסנו ל y כסקלר קבוע.

מציאת נגזרת חלקית של הפונקציה F לפי y תהיה:

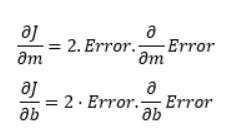
 כאשר התייחסנו ל x כסקלר קבוע.

כעת נוכל לחזור לחיפוש ה-Gradient Descent של פונקציית ה- Cost Function שלנו אשר תקרא כאן J לפי M ולפי b (המוצגים בנוסחה y=Mx+b)[11] **:**

****

לשם הפשטות נוותר כעת על סימן ה"סיגמא" ונסתכל אך ורק על כל שגיאה לחוד.

כעת נקבל שתי משוואות של J לפי M ולפי b :



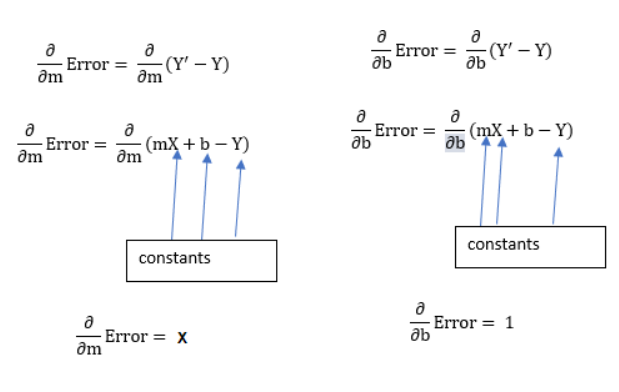
ניזכר בהגדרה שלנו למושג השגיאה:

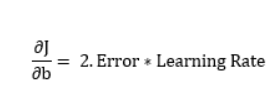
**Error=Loss Function= Y’(predicted) – Y(Actual)**

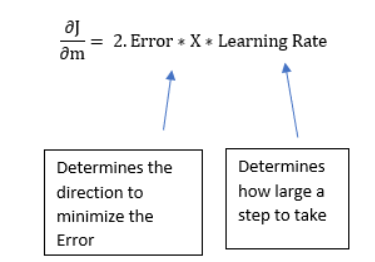
ונבחר להציגו כעת בצורה מצומצמת יותר:

**Error=(Y’-Y)** .

נחשב את ה Gradient- של השגיאה לפי M ולפי b:

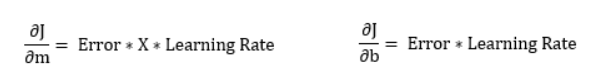


 כעת נחזור אחורה אל פונקציית הCost Function ונכפיל אותה ב Learning Rate אשר יקבע מה גודל הצעד שנבחר לעשות בכיוון הנבחר.



נוכל להיפטר גם מהסקלר 2 שבסה"כ יכול ליצור לנו אותו אקט ש-Learning Rate גדול פי 2 יעשה, לכן ניפטר ממנו ונוכל לשנות בעתיד את ה-Learning Rate בהתאם.

לבסוף נוכל לסכם את כל החישובים לשתי משוואות פשוטות המגדירות את ה-Gradient Descent הנ"ל:



לאחר חישוב הגרדיאנט סוף סוף נוכל לבצע את הצעד הרצוי, בהנחה שאנו נמצאים עם הערכים M0,b0 נוכל לעדכנם להיות M1,b1 כך:

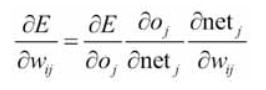
**M1=M0 – Error \* X \* Learning Rate b1=b0 – Error \* Learning Rate**

ברשתות נוירונים תהליך הGradient Descent- הוא יותר מסובך אך אתאר אותו בכלליות כחלק מהתהליך השלם של אימון רשת נוירונים [1]:

1. אתחול רשת הנוירונים עם Weights , Biases רנדומליים לכל נוירון ברשתות הנסתרות והתחלת המעבר Forward propagation.
2. ביצוע חישוב של פונקציית האקטיבציה בכל נוירון.
3. העברת פלט הנוירון לשכבה הבאה.
4. חישוב פונקציית האקטיבציה בשכבה האחרונה וקביעת הפלט של הרשת.
5. כעת באמצעות מסד נתוני האימון המתויג נוכל לזהות את השגיאה בכל נוירון באמצעות חיסור של הפלט המקורי מהפלט שהרשת קבעה.
6. חישוב השגיאה של קלט אחד ממסד נתוני האימון באמצעות

**** כאשר t זה התיוג המקורי של הקלט וy - זה הפלט של הרשת.

השבר בא להיפטר מהכפל המיותר בסקלר-2 שראינו בדוגמא הקודמת.

1. חישוב הנגזרת החלקית של השגיאה לפי כל אחד מה- Weights של הרשת באמצעות שימוש בכלל השרשרת: 

כאשר Wij מתייחס ל משקל מסוים,

Oj מתייחס לנוירון מסוים j ,

Netj מתייחס לקלט הנכנס לנוירון Oj.

1. לאחר חישוב זה מעדכנים כל משקל Wij בכל נוירון באמצעות הנגזרת החלקית בשילוב ה- Learning Rate בדומה למה שראינו בדוגמא הקודמת.
2. לאחר סיום אימון המודל נוכל להשמיש אותו ולבצע באמצעותו תחזיות על נתונים עתידים.

כעת לאחר שבידנו הבסיס לאיך עובדת רשת נוירונים מלאכותית נעבור לנושא המרכזי בו נדבר על רשתות נוירונים מסוג CNN – Convolutional Neural Networks

**Convolutional Neural Networks -CNN**

רשתות נוירונים מסוג קונבולוציה (ConvNet) דיי דומות לרשת הנוירונים הקודמת שדיברנו עליה: שתי הרשתות בנויות מנוירונים אשר תפקידם הוא ללמוד את ה-Weights וה-Biases המתאימים למשימה.

מבנה הרשת נשאר דומה ועדיין כולל שכבת קלט ושכבת פלט אחרונה, אז מה בעצם השתנה ומדוע אנו צריכים ארכיטקטורה חדשה?

ארכיטקטורת רשת זו יודעת שהקלט שהיא מקבלת הוא קלט מסוג תמונה ולכן אנו יכולים לנצל תכונות ספציפיות של תמונות לשם ייעול הארכיטקטורה.

**ייצוג תמונה דיגיטלית**

תמונה דיגיטלית ניתנת לייצוג על ידי שלוש שכבות של פיקסלים המסודרים לפי גודל מוגדר, כל שכבה מייצגת את החוזק של צבע אחד מבין הצבעים: אדום, ירוק וכחול.

כל פיקסל מייצג מספר בין 0 - 255 אשר מעיד על חוזק הצבע.

הצבע המוצג בכל קורדינטה בתמונה מיוצג על ידי חיבור הערכים של שלושת הפיקסלים המתאימים וכך נוצר הצבע המוצג בתמונה.

ניתן לייצג עד 256\*256\*256=16,777,216 צבעים שונים באמצעות שיטה זו.

ניתן להדגים את הפיצול בצורה הבאה:

רשתות נוירונים רגילות לא מסתדרות טוב עם הגודל של תמונה מלאה. לדוגמא אם ניקח תמונה בעלת 3 ערוצי צבעים (RGB) בגודל של 32 פיקסלים על 32 פיקסלים נקבל תצורה של 32X32X3, כתוצאה מכך כל נוירון בשכבת הקלט של הרשת יהיה בעל 32\*32\*3=3072 Weights- המתאימים לכל פיקסל הנמצא בתמונה!

כמות המשקלים כאן לא נראית גדולה מדיי אבל מה אם ניקח תמונה בגודל 600X600X3 נקבל 1,080,000 משקלים, כמובן שנרצה שיהיה לנו יותר מנוירון אחד בשכבת הקלט וכך כמות המשקלים שנצטרך לאמן תגדל במהירות רבה.

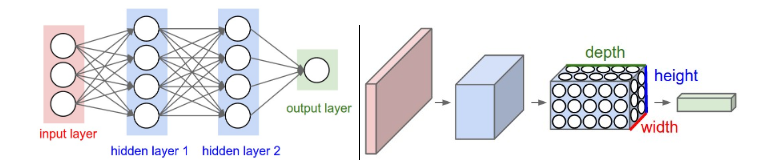
בבירור ניתן לראות כי גישה זו היא בעייתית מבחינת Scale, בזבזנית מבחינת כמות המשקלים, איטית לאימון ולכן נצטרך למצוא ארכיטקטורה חכמה יותר.

ConvNet מנצלת את הידע שהקלט של הרשת הוא תמונה ובכך מסדרת את רשת הנוירונים בצורה שונה.

בפועל כל שכבה מסודרים נוירונים ב-3 ממדים : אורך, גובה, עומק. לדוגמא אם נתכנן רשת שתקלוט תמונות בגודל 32X32X3 אז השכבה שלנו תהיה בממדים של קלטי התמונות.

הנוירונים שבשכבה יהיו מחוברים רק לאזור קטן של השכבה הקודמת במקום להיות מחוברים לכל הנוירונים אשר נמצאים בשכבה הקודמת – נראה זאת בקרוב.

לבסוף בשכבת הפלט נבנה וקטור בגודל 1X1YX, כאשר Y יהיה כמות הקטגוריות מבניהן נרצה לסווג אליה את התמונה אשר קלטנו לרשת.



תמונה מספר 17: המציגה השוואה בין סוגי הרשתות]16]

ניתן לראות בתמונה מספר 17 בצד שמאל את רשת הנוירונים הרגילה עליה דיברנו בעבר, בצד ימין ניתן לראות את רשת ה ConvNet אשר באדום מוצג התמונה הנקלטת לרשת, בכחול ייצוג התמונה כ3 מטריצות של פיקסלי התמונה לאחר מכן שכבת הנוירונים המסודרת ב-3 ממדים ולבסוף בירוק וקטור הפלט.

ארכיטקטורת ה- ConvNets בנויה משלוש שכבות ראשיות:

* Convolutional Layer
* Pooling Layer
* Fully-Connected Layer

שכבות אלו מהוות את הסיבה להצלחה הארכיטקטורה והסיבה אשר בגללה רשתות ה- ConvNets נהיו פופולריות מאוד לזיהוי וסיווג תמונות.

**Convolutional Layer**

שכבה זו היא אבן הבניין המרכזית ברשתות ConvNets והיא מבצעת את רוב החישובים המתמטיים הכבדים[16].

שכבה זו כוללת קבוצה של פילטרים הניתנים ללמידה, כל פילטר הוא מרחב קטן המוגדר לפי גובה, רוחב ומתאים לגודל העומק של קלטי התמונות.

לדוגמא פילטר סטנדרטי בשכבה הראשונה של הרשת יכול להיות בגודל 5X5X3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

(אורך -5 גובה -5 עומק -3 מכיוון שלתמונה יש 3 מרחבי צבע)

ניתן להדגים את הפילטר כך שכל תא בפילטר ימולא עם סקלר יחיד אשר ילמד במהלך אימון הרשת.

במהלך המעבר הקדמי השכבה מבצעת פעולת Slide-גלישה של הפילטר על התמונה שנקלטה ברשת ומבצעים מכפלה סקלרית בין ערכי התאים של הפילטר לבין ערכי התמונה בכל מקום. כאשר אנו "מגלישים" את הפילטר על גבי כל האורך והרוחב של התמונה אנו יוצרים סוג של מפת אקטיבציה דו ממדית המיוצרת על ידי הפילטר המועבר בכל מקום במרחב התמונה.

מטרת השכבה היא ללמוד פילטרים אשר יופעלו כאשר נראה בתמונה פיצ'ר ויזואלי מסוים כמו : פינה של עצם כלשהו, כתם גדול של צבע, או אפילו צורה של גלגל בשכבות המתקדמות.

כעת לשכבה יהיה סט שלם של פילטרים וכל אחד מהם ייצר מפת אקטיבציה דו ממדית שבעזרתם השכבה תייצר את הפלט שלה שהיא תעביר הלאה.

**חיבור מקומי** – כמו שצוין קודם כאשר אנו מתעסקים עם קלט שהוא תמונה זה לא פרקטי לחבר את כל הנוירונים לכל הנוירונים בשכבה הקודמת, במקום זאת אנו נחבר כל נוירון לאזור כלשהו מוגדר בתמונה. אזור זה נקרא ה- Receptive field של הנוירון והוא שווה לגודל הפילטר.

לדוגמא אם קלט התמונה שלנו הוא בגודל 32X32X3 וגודל אזור ה- Receptive field הוא 5X5 אז בכל נוירון בשכבת הConv יהיו Weights המתאימים לאזור הקליטה לכן יהיו בסה"כ

5\*5\*3=75 משקלים (ועוד 1 ל Bias). קל לראות כי ישנו הפרש גדול בכמות המשקלים כאן לעומת כמות המשקלים ברשת נוירונים רגילה.

**סידור במרחב** – ישנם שלושה פרמטרים חשובים אשר קובעים את סידור הנוירונים במרחב:

* Depth – העומק של הרשת הוא זה שקובע את כמות הפילטרים שאנו נרצה להשתמש בהם, כשכל אחד מהם לומד לחפש עבור משהו מסוים בתמונה.
* Stride – המגדיר את גודל הצעד אשר עושים כאשר מחליקים את הפילטר על גבי הקלט. כאשר הצעד הוא 1 אז אנו מזיזים את הפילטר פיקסל אחד הצידה בכל תנועה, כאשר הצעד הוא 2 אז אנו מזיזים את הפילטר 2 פיקסלים הצידה בתנועה אחת. ככל שנגדיל את גודל הצעד ככה נקטין את הקלט שקיבלנו ונוציא פלטים קטנים יותר.
* Zero-Padding – ריפוד באפסים: תכונה זו קובעת כמה שכבות של הסקלר 0 נרפד סביב מסגרת הקלט, הפיצ'ר הזה יאפשר לנו לשלוט בגודל הפלט ולקבוע כי גודל הקלט והפלט יישארו זהים- נראה זאת בהמשך.

**שיתוף פרמטרים[8,16]** – הארכיטקטורה של שיתוף פרמטרים באה לידי ביטוי בשכבות קונבולוציה בשביל לשלוט על מספר הפרמטרים השונים ברשת בכך שכל נוירון בשכבה מסוימת יהיה בעל פרמטר של Bias ופילטר משקלים בגודל 5X5 לדוגמא המחובר ל- Receptive Field שלו.

כל הנוירונים בשכבה הראשונה של הרשת מפענחים בדיוק את אותו פיצ'ר (פינה, צבע), רק במקומות שונים של התמונה.

תכונה זו שימושית מכיוון שנגיד אם יהיו משקלים ו- Bias אשר נועדו לפיענוח של קו אנכי באזור מסוים של Receptive Field. אז היכולת הזאת יכולה להיות שימושית לפיענוח קווים אנכיים בעוד מקומות בתמונה ולכן זה שימושי להשתמש במזהה זה בעוד מקומות בתמונה.

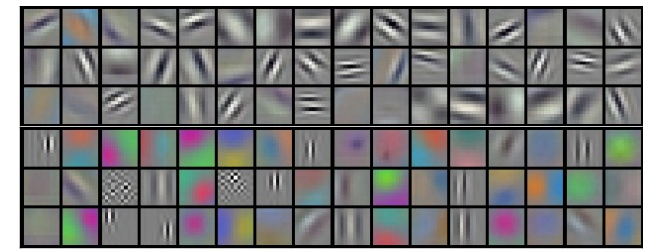
בצורה קצת יותר מופשטת תכונה זו מעניקה לרשתות קונבולוציה את היכולת להיות מותאמת יותר טוב לתזוזות ושינוי מקומות. לדוגמא אם ניקח תמונה של חתול ונעביר את החתול למקום אחר בתמונה אז עדיין נוכל לזהות את החתול.

אך לזיהוי תמונה נדרש יותר מאשר מזהה של פיצ'ר אחד, אראה דוגמא קונקרטית יותר אך מסובכת יותר:

הארכיטקטורה של : [Krizhevsky et al](http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks) היא ארכיטקטורה של רשת אשר מקבלת תמונות בגודל 227X277X3. בשכבת הקונבולוציה הראשונה היא משתמשת ב Receptive field בגודל של Field Size=11, Stride=4, Zero-Padding=0, Conv-Depth=96 ובשכבת הקונבולוציה הראשונה ישנם 290,400=55\*55\*96 נוירונים ולכל אחד מהם יש 11\*11\*3 =363 משקלים ו Bias אחד לכן יש בסה"כ 105,705,600 פרמטרים רק בשכבה הראשונה של הרשת, בברור זה מספר גבוה מדי לכן אם ננקוט בשיטת שיתוף הפרמטרים נוכל לקבל תוצאות טובות יותר:

נוכל להקטין בצורה דרסטית את מספר הפרמטרים בכך שנגדיר פילטר ספציפי דו-מימדי הנקרא Depth Slice אשר כמותו תהיה שווה לConv-Depth, לכן נגיד 96 Depth Slices כל אחד בגודל 55X55 ונגביל את הנוירונים בכל Depth Slice להשתמש באותם משקלים וBias.

באמצעות שיטה זו של שיתוף הפרמטרים לשכבה הראשונה בדוגמא שלנו יהיו רק 96 סטים שונים של משקלים אשר יסכמו ל: 96\*11\*11\*3=34,848 משקלים שונים או 34,944 פרמטרים שונים (+96 Bias), כעת כל ה55\*55 נוירונים בכל Depth Slice ישתמשו באותם פרמטרים. בפועל במעבר האחורי של הרשת כל נוירון יחשב את הגרדיאנט שלו עבור המשקלים, אבל כל הגרדיאנטים יתווספו לאורך כל ה Depth Slice לחוד ויעדכנו אך ורק סט אחד של משקלים לכל אחד מה-Slices .



תמונה מספר 18 :דוגמא לפילטרים הנלמדו ברשת .[16]

ניתן לראות בתמונה מספר 18 את הפילטרים שנלמדו ברשת של Krizhevsky et al. כל אחד מה96 פילטרים האלה הם בגודל 11X11X3 וכל אחד מהם משותף ל 55\*55 נוירונים בעומק אחד של Depth Slice.

ניתן לשים לב כי עקרון שיתוף הפרמטרים הוא הגיוני מכיוון שאם פיענוח של קו אנכי הוא חשוב במקום מסוים של תמונה אז מבחינה אינטואיטיבית פיענוח זה יכול להיות חשוב גם במקום אחר בתמונה ולכן אין טעם ללמוד מחדש לפענח קו אנכי בכל אחד מ55\*55 הנוירונים השונים בשכבת הקונבולוציה.

**Pooling Layer**

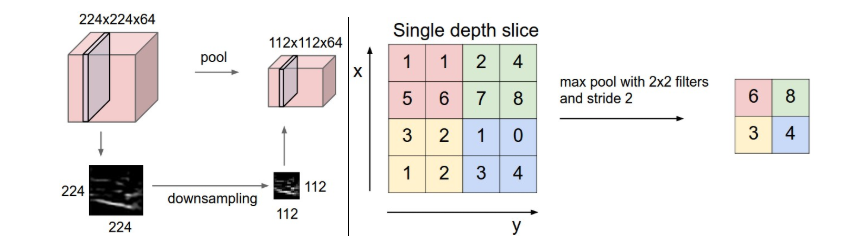
שכבת Pooling הינה שכבה שמטרתה היא להקטין- DownSampling ולפשט את הקלט המועבר בין השכבות. באמצעות הקטנתו נוכל להקטין את מספר הפרמטרים שנצטרך לאמן, החישובים שנצטרך לעשות וגם לצמצם את מידת ה Overfitting אשר תוסבר בהמשך[16].

בדרך כלל שמים שכבת Pooling ישר אחרי שכבת קונבולוציה לשם פישוט האינפורמציה לאחר עיבוד הקונבולוציה.

שכבת ה- Polling עובד בצורה אינדיבידואלית על כל Depth Slice ומשנה את המימדים שלו באמצעות שימוש בפונקציה לבחירה כמו ,Average Max, Min אך הנפוצה ביותר היא פונקציית ה Max- ולכן ניתן לקרוא לשכבה זו Max Pooling Layer. השימוש הנפוץ ביותר בשכבה זו הוא עם פילטרים בגודל 2X2 אשר מופעלים עם Stride=2 ובכך מקטינים ב75% את גודל הפלט.

לסיכום כל פעולת Max תתבצע מעל 4 מספרים (2X2) באזור מסוים ב Depth Slice.

ניתן לחשוב על שכבה זו כדרך לבדוק האם פיצ'ר מסוים נמצא באיזה מקום כלשהו בתמונה ואם כן זה מספיק טוב עבורנו לדעת את האזור בו הוא נמצא, האינטואיציה היא שברגע שנמצא פיצ'ר כזה המיקום הספציפי שלו לא בדיוק חשוב כמו האזור בו הוא ממוקם ביחס לפיצ'רים אחרים[9].



תמונה מספר 19 :שכבת Max Pooling Layer .[16]

ניתן לראות בתמונה 19 כיצד שכבת Max Pooling עובדת עם פילטר בגודל 2X2 וStride =2 כאשר בצד ימין של התמונה מבצעים פעולת Max על אזור בגודל 2X2 ומבצעים החלקה הצידה ב2 פיקסלים בכל פעם.

בצד שמאל של התמונה ניתן לראות דוגמא לתוצאה של שכבת ה Max Pooling וכיצד נראה הDownSampling לתמונה עצמה. כאשר השכבה מקבלת קלט בגודל 224X224X64 ומקטינה אותו בחצי ל112X122X64.

**ReLu Layer**

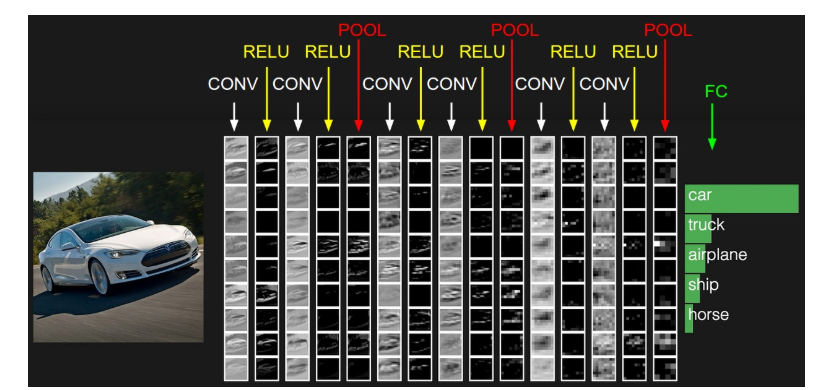
שכבה נוספת אשר מתווספת ישר אחרי שכבת הקונבולוציה אשר מבצע את פונקציית האקטיבציה ReLu Max(0,Z) על כל אלמנט המתקבל כקלט משכבת הקונבולוציה.

**Fully-connected layer**

שכבה זו היא בעצם השכבה האחרונה ברשת, שכבה זו מחברת כל נוירון מהשכבה האחת לפני האחרונה לכל אחד מהנוירון בשכבה הזו.

מספר הנוירונים בשכבה האחרונה יקבע לפי מספר הקטגוריות שאנו מאמנים את הרשת שלנו לסווג. לדוגמא, אם נרצה לסווג תמונות בין האפשרויות: חתול, כלב ובן אדם נצטרך 3 נוירונים בשכבה האחרונה.

הנוירונים בשכבה זו יקבעו לאיזה קטגוריה שייכת התמונה שעובדה ברשת.



תמונה מספר 20 :דוגמא לארכיטקטורת CNN . [16]

ניתן לראות בתמונה מספר 20 דוגמא לארכיטקטורת רשת קונבולציה אשר כוללת 6 שכבות קונבולוציה שלאחר כל אחת מהן יש שכבת ReLu, 3 שכבות Max-Pooling ולבסוף שכבה אחת מסוג Fully-connected אשר מסווגת את התמונה המתקבלת לרשת בתור מכונית.

**Overfitting and Data Selection**

לפני שאנו מתחילים לבנות את המודל שלנו אנו צריכים לקבוע כיצד אנו מחלקים את המסד הנתונים שלנו שעליו אנו נעבוד, לכן נבחר לחלק אותו לשתי קבוצות[1]:

* Training Data – סט נתונים אשר כולל את הנתונים שלנו שבעזרתם אנו מאמנים את הרשת שלנו.
* Validation Data- סט נתונים מצומצם יותר אשר בעזרתו אנו בודקים את דיוק המודל שלנו בחיזוי או סיווג נתונים שהוא לא ראה מעולם בזמן האימון.
* Test Data- סט נתונים נוסף המשתמש לבדיקה סופית של המודל לאחר סיום האימון.

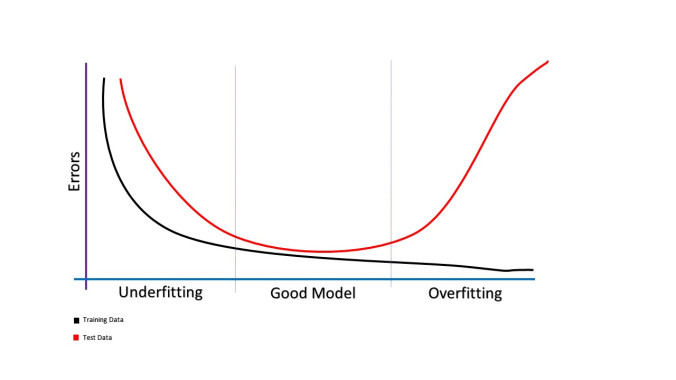
בדרך כלל אנו בוחרים לחלק את מסד הנתונים שלנו באופן שרוב הנתונים יהיו בסט הנתונים המשמשים לאימון הרשת בשביל שהרשת שלנו תראה כמה שיותר מקרים שונים וכך תיחשף ליותר אפשרויות, חלוקה אפשרית לדוגמא היא

60% מהנתונים בTraining Data ,20% ב Validation Data ו20% ב Test Data.

לאחר שחילקנו את הנתונים נבנה ארכיטקטורה של מודל למידה כמו רשת נוירונים, אנו נתחיל לאמן את המודל על הData- שיש ברשותנו, בזמן אימון המודל הנתונים שבסט האימון קובעים את ה-Weights וה-Bias בכל הנוירונים ברשת.

אחת מהטכניקות אימון הנפוצות היא מספר מעברים על בסיס הנתונים - שימוש בחלוקת לEpochs- [1] .

לכן בטכניקה זו נחלק את את שלבי האימון שלנו לכמה Epochs שונים. לדוגמא נוכל לקבוע לעבור על מסד הנתונים 5 פעמים אז נקבע את פרמטר ה- Epochs להיות 5. לאחר כל Epoch נוכל לבדוק את דיוק המודל שלנו על סט נתוני האימון וגם סט נתוני הבדיקה[14]. לפעמים נוכל לשים לב לתופעה דיי מוזרה אשר הדיוק של אחוזי האימון עולה אך הדיוק של נתוני הבדיקה יורד מצב זה נקרא Overfit.



תמונה מספר 21 :גרף Overfit [[](https://meditationsonbianddatascience.com/2017/05/11/overfitting-underfitting-how-well-does-your-model-fit/)https://bit.ly/35A767K] .

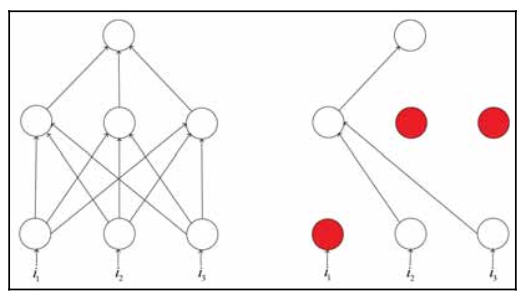
ניתן לראות בתמונה מספר 21 גרף המציג את אחוזי השגיאה על סט האימונים בשחור ועל סט הבדיקה באדום לאורך מספר Epochs.

נראה שתחילה אנו במצב הנקרא **Underfitting** אשר אומר כי המודל שלנו עדיין אינו מכיר מספיק את הנתונים וכדאי להמשיך לאמנו. Good Model הינו המקום בו אחוז השגיאות על סט האימונים ואחוז השגיאות על סט הבדיקות הוא נמוך וזה מקום טוב מבחינת דיוק המודל. **Overfitting** התאמת יתר הינו המצב שבו המודל למד את נתוני האימון טוב מידי עד לרמה שאפשר להגיד שהוא "שינן" אותם לכן הוא מדויק מאוד לגביהם אך לגבי נתונים חדשים שאינו ראה מעולם הוא טועה ובגדול.

זהו הוא מצב רע שנרצה להימנע ממנו בכל מחיר[1]..

אז איך נמנעים מ Overfit - ?

* איזון הסטים של הנתונים- חשוב להקפיד על כך שסט האימון וסט הבדיקה יהיו מגוונים ככל שאפשר וחלוקת הנתונים השונים מביניהם תתבצע בצורה שווה יחסית. מומלץ לבצע את חילוק מסד הנתונים לשני הסטים בצורה רנדומלית.
* Dropouts- טכניקה זו באה לידי מימוש בזמן אימון רשת הנוירונים בכך שהיא מציגה לנו משתנה הסתברותי חדש P אשר מוצמד לכל נוירון ברשת אשר קובע הסתברות מסוימת שנוירון כלשהו לא יפעל במהלך מעבר אחד על קלט מסוים ברשת. הרשת לא נפגעת מכך ואפילו נהיית יותר מדויקת אפילו כאשר חלק מהאינפורמציה בה נעלם. טכניקה זו מונעת מהרשת להסתמך יותר מדי על נוירון אחד או יותר וכתוצאה מכך להגדיל את הדיוק של המודל ולהימנע יותר מOverfitting.



תמונה מספר 22 :המציגה את תהליך ה- Dropout בארכיטקטורה בסיסית[1].

לדוגמא ניתן לראות בתמונה מספר 22 כי הנוירונים האדומים הם נוירונים אשר לא "נפלו" במעבר אחד ברשת.

ישנן טכניקות רבות נוספות להימנע מOverFit אך כעת אתמקד בטכניקה פופולרית מאוד ואף יגידו הכרחית ברשתות CNN הנקראת [14]Data Augmention.

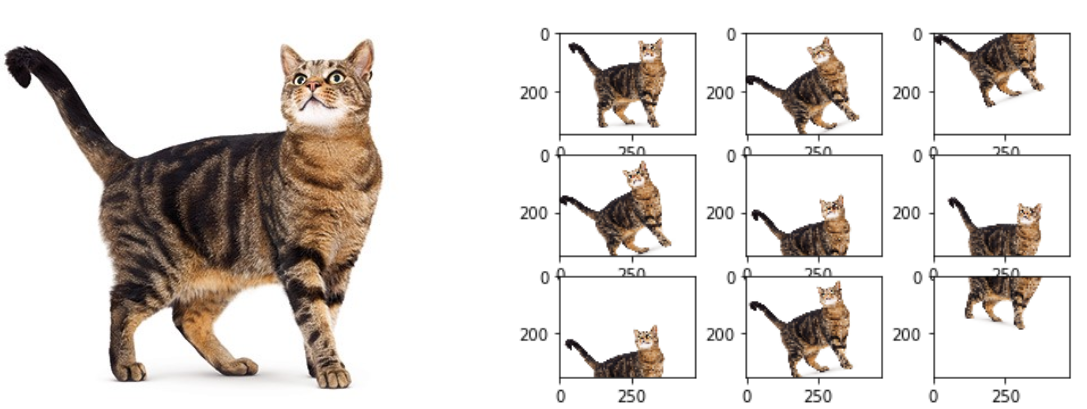
טכניקה זו נותנת לנו דרכים נוספות להגדיל את מסד נתוני התמונות שלנו מבלי להשיג תמונות חדשות.

טכניקה זו יכולה לעזור מאוד כאשר מסד נתוני התמונות שלנו מצומצם ואין לנו מספיק תמונות שונות בשביל לאמן את הרשת. אופציה אחת היא להשיג עוד תמונות אך מה נעשה אם התמונות שעליהן אנו מאמנים את הרשת הן נדירות?

לדוגמא אם אנו בונים מודל אשר מחפש לזהות את החיה נמר השלג שנחשבת אחת החיות הנדירות והחמקניות בעולם, בשביל לבסס מודל מדויק מספיק שידע לזהות את החיה במדויק נצטרך הרבה תמונות של החיה מכל מיני זווית. לצערנו החיה הזאת כמעט נכחדה ואין מספיק תמונות טובות שלה בטבע.

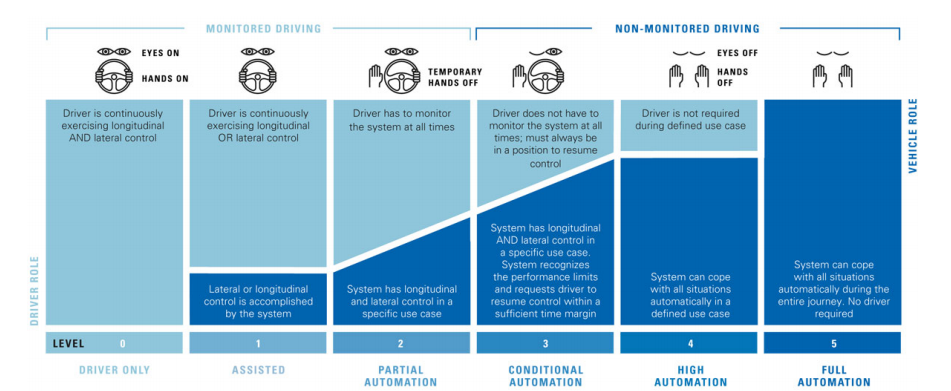
בעזרת טכניקת Data Augmention נוכל לקחת תמונות קיימות שיש לנו ולעשות עליהן עריכה מסוימת אשר תוכל להיראות למודל כתמונה שונה מהקיימת.

חלק מטכניקות האוגמנטציה הן: הפיכת התמונה, חיתוך רנדומלי של חלקי התמונה, זום אין לתוך התמונה, שינוי גווני צבע התמונה, סיבובים ועוד.



תמונה מספר 23 :דוגמא לאוגמנטציה של תמונה [[](https://mc.ai/image-data-augmentation-image-processing-in-tensorflow-part-2/)https://bit.ly/35A7R0A].

ניתן לראות בתמונה מספר 23 כיצד טכניקת אוגמנטציה יכולה לקחת תמונה אחת לדוגמא ולבצע עליה אוגמנטציה בשביל לקבל כמה תמונות שונות מאותו מקור.

**כיצד רכבים אוטונומיים עובדים**

תמונה מספר 24: דרגות האוטומציה ברכבים[4].

בתמונה מספר 24 ניתן לראות את דרגות האוטומציה ברכבים מסודרים משמאל לימין לפי רמת האוטומציה של הרכב ללא תלות בנהג.

כדי להגיע לרמות אוטומציה גבוהות רכב אוטונומי יזדקק לשילוב טכנולוגי בין חומרה לתוכנה, ראשית נצטרך לתת לרכב אפשרות "לחוש את הסביבה", לדעת היכן הוא נמצא ומה קורה סביבו. לשם כך נצטרך כמה מערכות חומרתיות[5]:

* Lidar System – מערכת לידאר היא מערכת לייזרים אשר מודדת מרחק אל אובייקט מסוים באמצעות שימוש בפולסי לייזר אשר היא מקרינה אל האובייקט אליו היא רוצה למדוד את המרחק, בעזרת מערכת זו המכונית יכולה לקבל אינפורמציה על הסביבה הקרובה אליה.
* Radar System – מערכת ראדר המשמשת גם היא לקבלת אינפורמציה על סביבת המכונית. מערכת ראדר משתמש בגלים אלקטרומגנטים. היתרון במערכת זאת הוא הטווח שידור הארוך של גלים אלקטרומגנטים ובכך היא תורמת להשלמת התמונה הרחוקה של הסביבה[4].
* Sonar System – מערכת נוספת אשר עוזרת לקבל אינפורמציה על סביבת המכונית, מערכת סונאר משתמשת בגלי קול ומחשבת את הזמן שלוקח לגלי הקול להגיע לאובייקטים קרובים. המרחק לאובייקטים מחושב לפי זמן ההגעה של גלי הקול לאובייקט.
* GPS System - מערכת GPS המשמשת לניווט באמצעות שימוש במיקום גיאוגרפי לפי לווינים.
* Controlling System – המערכת המרכזית אשר שולטת על הרכב. מטרת מערכת זו היא להיות המוח מאחורי המערכות ולהפעיל את כל המערכות המכאניות בעת הצורך (מערכת הברקסים, הזזת ההגה ועוד).
* Camera System – מערכת מצלמות אשר מטרתה לקבל אינפורמציה איכותית יותר על הסביבה הקרובה לרכב. בדרך כלל מרשתים את הרכב במצלמות 360, מצלמות לטווח קרוב ומצלמות עם עדשה לטווח רחוק. בעזרת האינפורמציה המתקבלת על ידי מצלמות ניתן לבנות אלגוריתמים של למידת מכונה לזיהוי הולכי רגל, פסי נתיבי ואפילו זיהוי תמרורים שונים. על מערכות אלו ארחיב בהמשך.



תמונה מספר 25: מערכת סנסורים הנדרשים למערכת Tesla Autopilot[4].

ניתן לראות בתמונה מספר 25 את מערכת הסנסורים של המערכת [Tesla AutoPilot](https://www.tesla.com/autopilot), המשלבת מערכת מצלמות אשר כוללת מצלמות למרחקים שונים וזוויות שונות, מערכת Sonar באמצעות שימוש בגלי Ultrasonic לרדיוס של עד כ8 מטרים סביב המכונית, מערכת Radar קדמי עד למרחק של 160 מטרים.

כעת נצלול עמוק אל תוך כמה מערכות הנמצאות ברכבים אוטונומים.

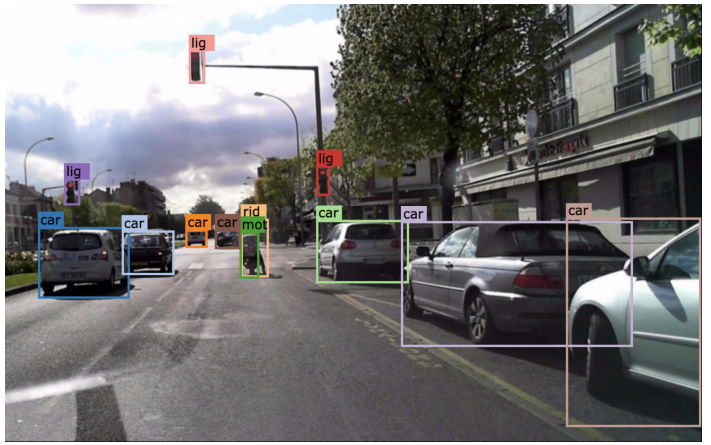
**Object Detection**

זיהוי אובייקטים הינו אחד הצרכים הקריטיים לשם בניית מכונית אוטונומית[5].

המכונית האוטונומית חולקת את הכביש עם עוד אובייקטים תנועתיים, במיוחד באזורים עירונים ובינעירוניים. המודעות לאובייקטים תנועתיים נוספים או מכשולים היא קריטית על מנת להימנע מתאונות שיכולות לסכן את חיי הנוסעים במכונית וחיי הסובבים את המכונית. זיהוי אובייקטים בסביבות עירוניות, עמוסות, יכול להיות משימה קשה במיוחד בגלל המבחר הגדול של צורות האובייקטים בסביבות עמוסות וגם אי היכולת לראות אובייקט במלואו בגלל הסתרה חלקית של שדה הראייה על ידי אובייקט אחר. מעבר לכך ישנם גורמים פיזיקליים נוספים אשר מקשים על משימת הזיהוי כגון : השתקפויות אור, צללים, ערפל ועוד.

מערכת לזיהוי הולכי רגל היא מערכת אשר משתמשת בזיהוי אובייקטים בתמונה וחייבת להיות מושלמת מבחינת עמידות מול תנאי מזג אוויר, יעילות בזיהוי אנשים בזמן אמת , אין מקום לטעויות בבניית מערכת כזו כל טעות קטנה יכולה להיות פטאלית. בניית אלגוריתם ראייה ממוחשבת שידע לזהות אנשים נחשב למשימה מסובכת בגלל שוני גדול בצורה שאדם יכול להיראות, אם זה בבגדים שהוא בחר ללבוש או בפוזיציית הגוף שבה הוא נמצא כרגע (עמידה, ישיבה, חיבור פיזי עם אדם נוסף ועוד).

הסנסורים הנפוצים אשר משתמשים במערכת זיהוי אובייקטים הם המצלמות המוצבות באזורים השונים במכונית. במהלך היום משתמשים בראייה מסוג Visible spectrum ובלילה נעזרים בטכנולוגית הInfraRed אשר מועילה יותר לראייה בשעות לילה. בנוסף למצלמות אלו משתמשים גם במצלמת Thermal InfraRed אשר קולטת את הטמפ' היחסית בתמונה, בעזרת מצלמה זאת יהיה קל יותר להבדיל בין אובייקטים חמים כמו הולכי רגל לבין אובייקטים קרים כגון ארגז ירקות או הכביש עצמו. ישנו גם שימוש בסנסורים נוספים אשר משדרים סיגנל ומקבלים קלט ממנו כגון סורקי לייזר אשר יכולים לעזור בקבלת מידע על המרחק בין אובייקטים ובכך לעזור בהבנת העולם התלת ממדי מסביב למכונית.



תמונה מספר 26: הדגמה של זיהוי אובייקטים בתמונה[5].

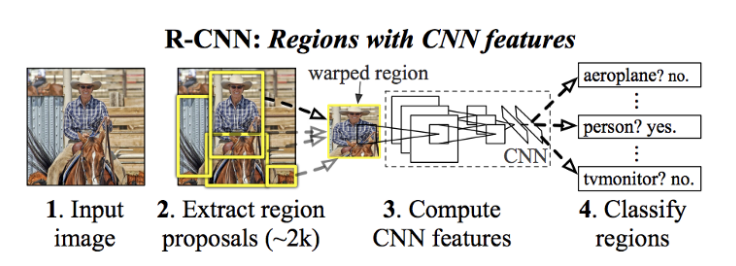
ניתן לראות בתמונה מספר 26 זיהוי של אובייקטים שונים המסווגים לאובייקטים נייחים כגון: רמזור, מכונית חונה ואובייקטים ניידים כגון: מכונית, אופנוע, רוכב אופנוע אלה הם אובייקטים אשר לאחר זיהויים המכונית האוטונומית תצטרך לעקוב אחרי תנועתם ואף לדעת להבדיל בין מכונית אחת למכונית שניה (הרחבה על כך בהמשך)

תנאי השטח, מזג האוויר והשעה ביום משתנים כל הזמן לכן הסתמכות על סוג אחד בלבד של סנסורים הוא בעייתי. מצלמות וסורקי לייזר מושפעים על ידי השתקפויות או משטחים שקופים, אובייקטים חמים כמו מנועים או טמפ' חמה יכולה להשפיע על המצלמות הטרמיות. לכן שימוש בכמה מערכות סנסורים שונות לבניית אלגוריתם לזיהוי אובייקטים זו הדרך הבטוחה והטובה ביותר.

ישנם אלגוריתמים רבים לזיהוי אובייקטים בתמונה אך שימוש בטכנולוגית ה – Convolutional Neural Networks תרם לביצועים גבוהים יותר בזיהוי האובייקטים .

במשך השנים התפתחו כמה רשתות אשר ניסו לפתור את בעיית זיהוי האובייקטים בהיבטים שונים אציין כמה מהן:[,14,135]

* **CNN** – רשת זו הינה רשת בסיסית אשר באה לזהות אובייקט בתמונה ולסווגו לקטגוריה, אך היא אינה הגישה המתאימה לזיהוי אובייקטים רבים בתמונה אחת מכיוון שכמות האובייקטים המופיעים בתמונה בכל רגע היא אינה פרמטר קבוע אלא פרמטר משתנה. פתרון נאיבי לבעיה זו היא לחלק את התמונה לכמה חלקים -לוקליזציה, ולסווג כל חלק בעצמו אך גישה זו תתקל בבעיית Scale רצינית אם נרצה לחלק את התמונה להרבה אזורים, לכן נהיה חייבים לחפש פתרון נוסף יצירתי יותר.
* **R-CNNS** - רשת זו באה לפתור את בעיית הלוקליזציה (חלוקת התמונה לחלקים) של רשת ה- CNN הרגילה בכך שבמקום לחלק את התמונה למספר ענק של חלקים, האלגוריתם יבצע חיפוש סלקטיבי בתמונה ויחלק את התמונה ל- 2000 חלקים בלבד שבהם נחפש אובייקטים, ניתן לראות בתמונה 27 דוגמא לרשת זו.

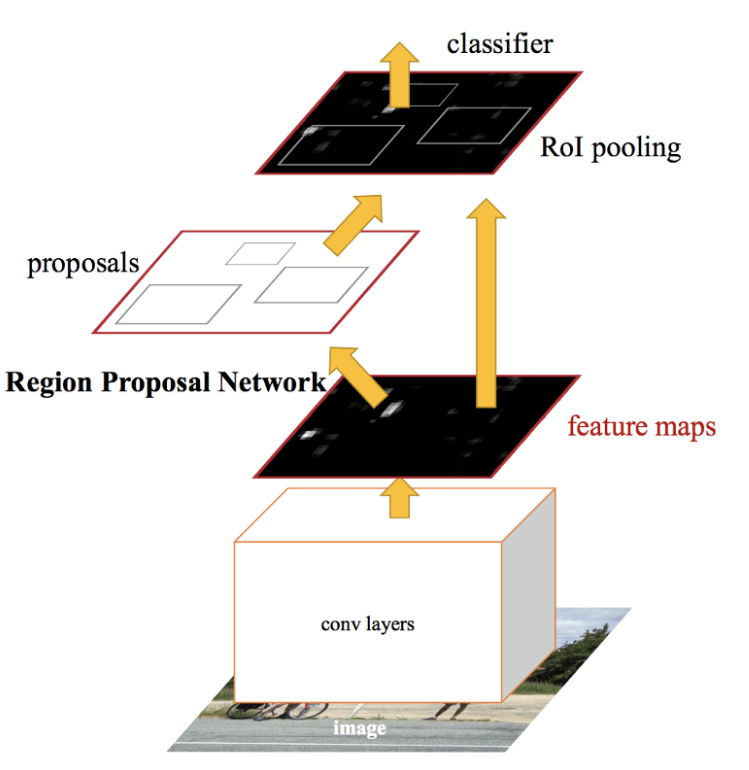


תמונה מספר 27: רשת R-CNN[[13.

* **Fast R-CNN -**  רשת זו באה לשדרג את הרשת הקודמת ולהפוך אותה למהירה יותר, כעת במקום להכניס לרשת ה- CNN כאלפיים חלקים שבהם היא תבצע חיפוש לאובייקטים ננקוט בגישה מעט שונה.

מכניסים את התמונה לרשת ה- CNN על מנת ליצור מפת פיצ'רים של התמונה ובעזרתה מזהים את האזורים שבהם נרצה לבצע חיפוש אובייקטים.

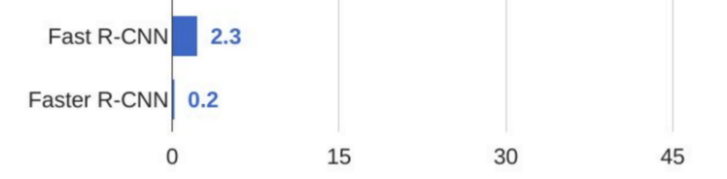
* **Region-based network Faster-R-CNN –**  רשת זו באה לשדרג את הרשת הקודמת בכך שבמקום לבצע חיפוש סלקטיבי איטי, הוסיפו רשת למידה מקדימה אשר תלמד לחזות את האזורים בתמונה שבהם נרצה לחפש את האובייקטים ובכך לייעל יותר את הרשת.

****

תמונה מספר 28: רשת Faster- R-CNN[[13.

ניתן לראות בתמונה מספר 28 שימוש בשתי רשתות שונות לביצוע המשימה של זיהוי האובייקטים. הרשת הראשונה הינה Region Proposal Network והרשת השנייה מוצגת כ-Classifier אשר מסווגת את האובייקט אשר נמצא באזורים המוצעים לחיפוש. ניתן לראות בתמונה 29 השוואה במהירויות של רשתות אלו.





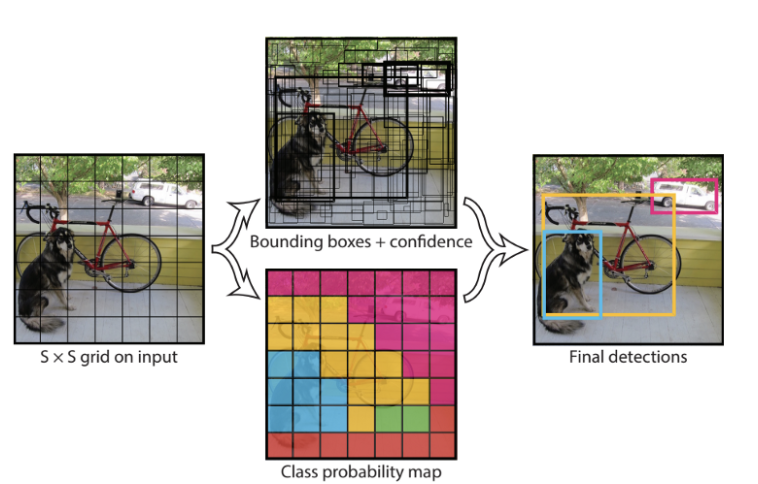
תמונה מספר 29: השוואה במהירות בין הרשתות שהוצגו לעיל[[13.

* **Yolo -You Only Look Once –** האלגוריתמים הקודמים פרקו את התמונה לחלקים בשביל לאתר את האובייקט בתמונה. רשת זו לא מסתכלת על התמונה השלמה, היא מסתכלת על חלקים בתמונה שיש להם הסתברות גבוהה שיש בהם אובייקטים, כמתואר בתמונה מספר 30.

אלגוריתם זה משתמש ברשת – CNN לחיזוי Bounding boxes ואת ההסתברות לאובייקט מקטגוריה מסוימת בכל אחת מהקופסאות האלו.

האלגוריתם מקבל תמונה ומפצל אותה לגריד בגודל SxS -, לאחר מכן בכל אחד מהתאים בגריד אנו לוקחים - M Bounding Boxes (כאשר M הוא פרמטר) .

לכל אחד מה- Bounding Boxes הרשת תחזיר פלט המייצג הסתברות שבמקום זה נמצא אובייקט מסוג מסוים, ה- Bounding Boxes עם הסתברות גדולה יותר מ- Threshold כלשהו נבחר ואז משומש לאיתור האובייקט בתוך התמונה.



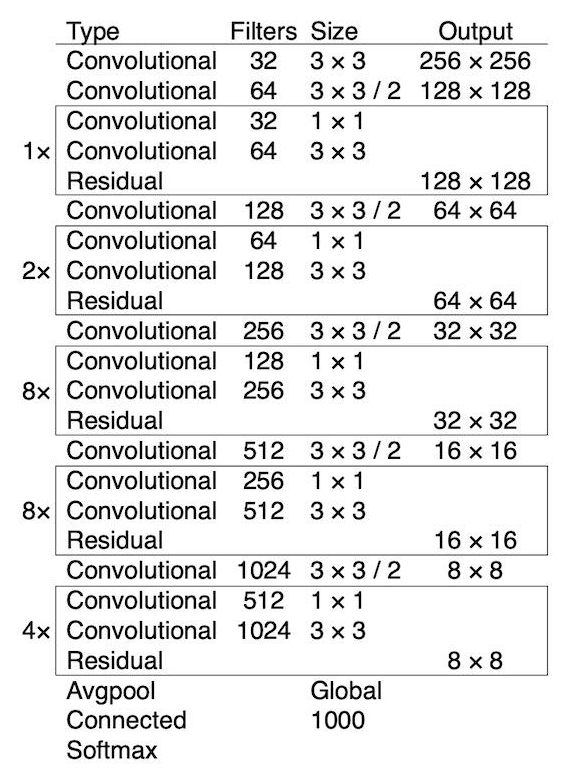
תמונה מספר 30: המציגה את תהליך האלגוריתם YOLO [[13.

**YoloV3**

רשת נפוצה ביותר מסוג Yolo אשר בנויה משני חלקים עיקריים[3]:

1. Feature Extractor -אשר יזהה 3 או יותר פיצרים מרכזיים בתמונה כאשר כל אחד מהם בגדלי תמונה הולכים וקטנים.
2. Detector- אשר יקבל את ה3 או יותר פיצרים ובאמצעות עיבוד של כולם יקבע את ה-Bounding Box ולאיזה קטגוריה האובייקט שזוהה שייך.

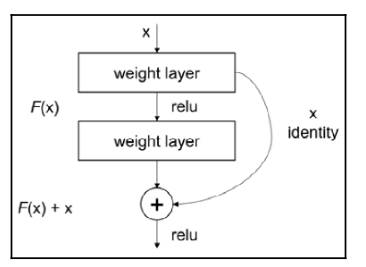
מנתח הפיצ'רים של YOLOv3 נקרא Darknet-53 אשר בנוי בצורה הבאה:



תמונה מספר 31: Darknet-53 [3]

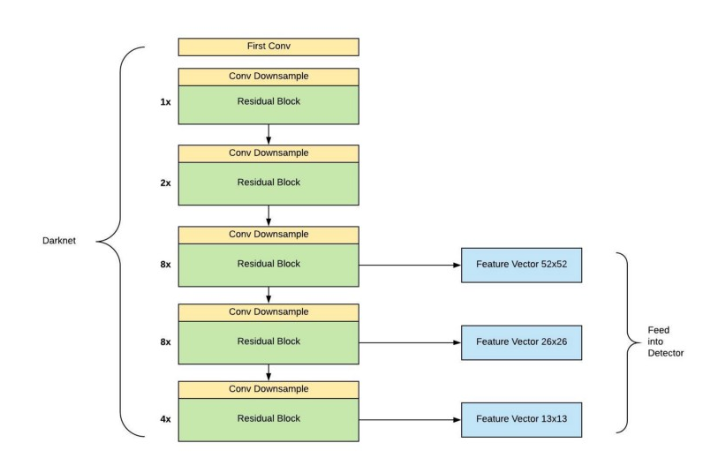
ניתן להשתמש בארכיטקטורה זאת גם לזיהוי וסיווג של תמונה רגילה אז נוסיף לארכיטקטורת הרשת את שכבת הAvgpool ,Connected , Softmax לשם זיהוי קטגוריית תמונה מתוך 1000 קטגוריות שונות.

* Residual Block – בבלוק זה עדיין לא נתקלנו אך נתאר אותה בקלות- מטרתה היא לקחת את האינפורמציה מהשכבה הקודמת ולהעביר אותה קדימה, בעזרת שכבה זאת ניתן לאמן רשתות עמוקות יותר מהרגיל, ניתן לראות ההעברת מידע זה בתמונה מספר 32.



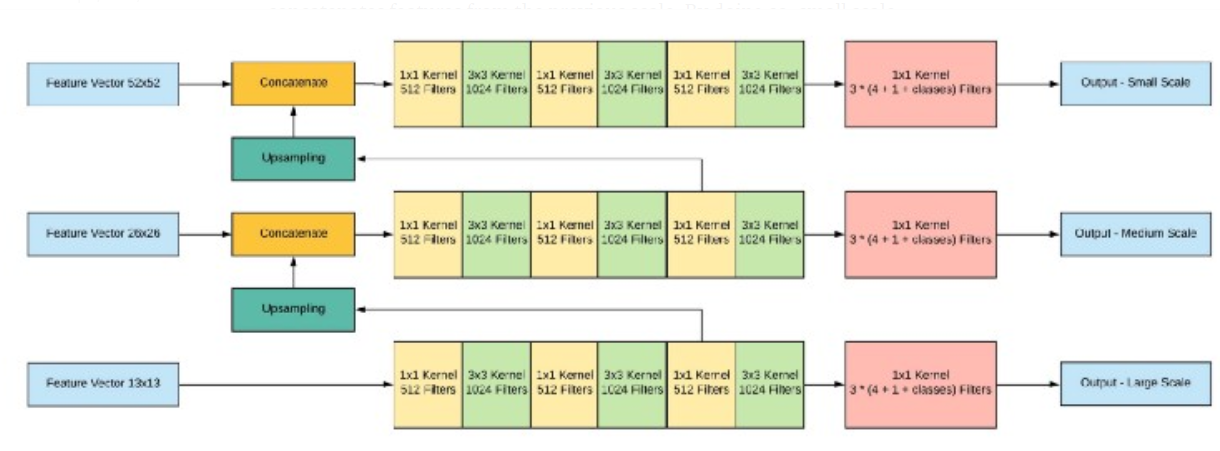
תמונה מספר 32: Residual block [14].

במקרה של זיהוי אובייקטים לא נשתמש בחלק הסופי בארכיטקטורת DarkNet-53 אלא נחבר את הפלטים של בלוקי ה Residual לרשת ה-Detector בצורה המתוארת בתמונה 33.



תמונה מספר 33: DarkNet-53 לזיהוי אובייקטים [3].

בתמונה מספר 34 ניתן לראות שה- Detector בנוי מ3 רשתות קונבולוציה אשר כל אחת עובדת על גודל שונה של Feature Vector, ניתן לראות גם כי לפני סוף הרשת השכבה עם הוקטור הקטן ביותר מעבירה מידע לשכבה מעל בשביל לעזור לה לזהות את האובייקט בצורה טובה יותר.



תמונה מספר 34: DarkNet-53 – Detector[3].

המטרה הבאה של האלגוריתם הוא לאתר את ה- Bounding Box של האובייקט, בדרך כלל נציג את ה- Bounding Box על ידי 4 קורדינטות (X,Y) בצורת ריבוע סביב האובייקט אך ברשת זו נקטו בגישה אחרת בשם Anchor Box.

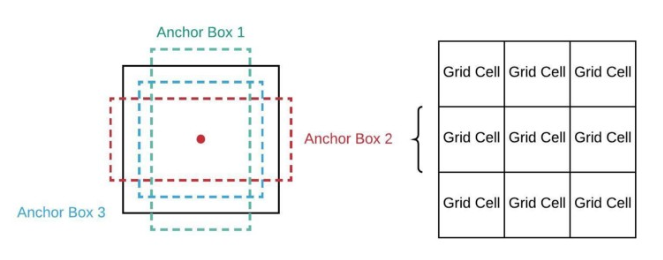
Anchor Box היא קופסא המיוצגת על ידי 4 קורדינטות בצורת ריבוע אשר גודלה הוא הייפר-פרמטר אשר נקבע לפני שלב אימון הרשת, שכבות הקונבולוציה מוציאות פלט בצורת מטריצה ריבועית אשר אנו יכולים להתייחס אליו בתור סוג של רשת של תאים ולשים Anchor Box בכל אחד מתאי הרשת( ניתן לשים כמה קופסאות כאלו אחד על השני ולגרום לחפיפה בינהם).

כעת בשלב האימון במקום לחזות קורדינטות של אובייקט מאפס אנו רק נצטרך לחזות הזזות של קופסאות אלו בשביל לאתר אובייקטים.

באלגוריתם YOLOv3 משתמשים ב 3 Anchor Boxes לכל אחד מתאי הרשת, בנוסף יש לנו 3 רשתות בגדלים שונים עליהם אנו מבצעים את האלגוריתם לכן אם גדלי

הרשתות מטריצה שלנו הם:

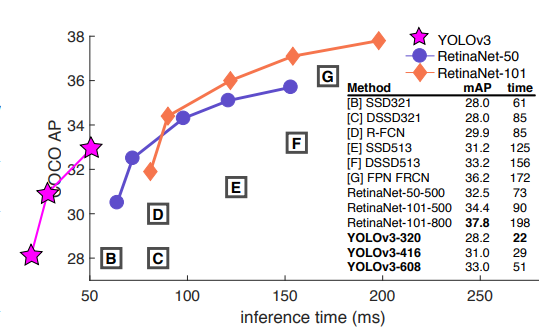
52X52,26X26,13X13 אז יהיו לנו 52x52x3,26x26x3,13x13x3 Anchor Box שונים כפי שניתן לראות בתמונה מספר 35.



תמונה מספר 35: Anchor Boxes לכל תא[3].

לכל Anchor Box נצטרך לחזות 3 דברים:

1. כמה צריך להזיז את קופסא בשביל למצוא אובייקט.
2. פרמטר הסתברותי המציג מה ההסתברות שבקופסא זאת נמצא אובייקט.
3. וקטור הסתברויות אשר מציג מה ההסתברות שקופסא זו שייכת לכל אובייקט.



תמונה מספר 36: השוואת מהירויות הסקה בין רשתות שונות [https://arxiv.org/pdf/1804.02767.pdf].

לסיכום ניתן לראות בתמונה מספר 36 כי YOLOv3 ניצחה במהירות אלגוריתמים רבים לזיהוי אובייקטים בפער רב, זו אחת הסיבות שבגללה YOLO כל כך פופולרית בתחום זיהוי האובייקטים ונפוצה בשימוש רשתות אשר זקוקות לזיהוי אובייקט מהיר כמו מערכת לזיהוי הולכי רגל לדוגמא.

**Object Tracking**

במעקב המטרה המרכזית היא לחזות את מיקומו ומצבו של אובייקט אחד או יותר במשך תקופת זמן מסוימת באמצעות קלטים מסנסורים (לדוגמא ממצלמות הרכב)[5].

משימה זו שונה ממשימת זיהוי האובייקט מכיוון שבמשימה הקודמת כל פריים של תמונה עובד בצורה בלתי תלויה בשני, אך כעת אנו מבצעים קישורים בין כל פריים ופריים בשביל לעקוב ולחזות תנועות ומצבים של אובייקטים רבים.

מצבו של אובייקט בדרך כלל מיוצג לפי מיקומו, המהירות שלו ויכולת ההאצה שלו בזמן מסוים. מעקב אחרי אובייקטים תנועתיים זו משימה מאוד חשובה עבור נהיגה אוטונומית. לדוגמא נסיעה אוטונומית בכביש מצריכה חישוב מרחק בלימה של הרכב האוטונומי מול הרכב אשר נמצא מולו בכביש. מרחק בלימה זה מחושב לפי מצב הרכב ומהירותו.

חישוב מרחק בלימה זה קריטי לשם מניעת תאונות והתנגשויות שיכולות לקרות מאי שמירת מרחק מספיק בין מכונית למכונית. במקרה של הולכי רגל, אופניים או אופנוע, חיזוי התנועה הבאה שלהם נהיה קשה יותר מכיוון שהם יכולים לשנות את כיוון תנועתם ומהירותם בזריזות וללא סימני מוקדמים. זו אחת הסיבות שבגללה אנשים נוהגים לנהוג לאט ובזהירות יותר באזור של אנשים ורוכבים.

זיהוי ומעקב אחרי אובייקטים תנועתיים מאפשרים למכונית האוטונומית להתאים את מהירותה לפי הסביבה שבה היא נמצאת.

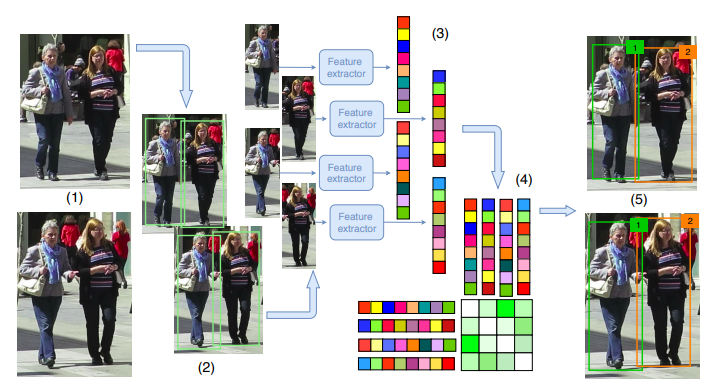
מערכות מעקב נתקלות באתגרים רבים שעליהם הן חייבות להתגבר לדוגמא: מרחב עמוס אובייקטים, השונות והמורכבות של התנועה, חסימת מרחב הראייה ועוד.

המשימה של מעקב אחר אובייקט ספציפי היא גם אספקט מאתגר מעצם כך שאובייקטים מאותו סוג (לדוגמא מכונית) יכולים להיראות ממש דומה (ישנה אפשרות שבשדה הראייה יהיו בדיוק אותן 2 מכוניות מאותו סוג ודגם). אלגוריתם למעקב יהיה חייב ללמוד להבדיל אפילו בין אובייקטים דומים וכאן רשתות נוירונים יכולות לעזור לשם סיווג קטגורית האובייקט ולמידת המראה של כל אובייקט על מנת שהאלגוריתם יוכל להבדיל ביניהם.

אתגרים פיזיקליים נוספים הם תנאי מזג האוויר, שעת היום, צללים והשתקפויות אשר גם משפיעים על היכולת של אלגוריתם לעקוב אחרי אובייקט מסוים.

ישנם אלגוריתמים רבים שפותחו לשם מעקב אחרי אובייקטים אך רובם עובדים לפי עקרון מנחה מסוים, הכינוי לאלגוריתמים אלו הוא MOT - Multiple Object Tracking .[15]

1. שימוש באלגוריתם לזיהוי אובייקטים המנתח כל פריים של הסרטון בשביל לסווג אובייקטים לקטגוריות שונות באמצעות Bounding Boxes.
2. שימוש באלגוריתמי Feature Extraction/Motion אשר ינתחו כל פיצ'ר של האובייקט שזוהה ויבנו לו סוג של פרופיל נראות. בנוסף האלגוריתמים ינסו לחזות את הצעד הבא של כל אובייקט.
3. שימוש במידע שנותח בשלב 2 בשביל לחשב מטריצת דמיון/מרחק בין כל אחד מהאובייקטים שזיהינו.
4. שימוש במטריצת המרחק/דמיון בשביל לקשר כל אובייקט שנלמד בשלבים הקודמים לאובייקט המתאים הנמצא בפריים הבא בסרטון.



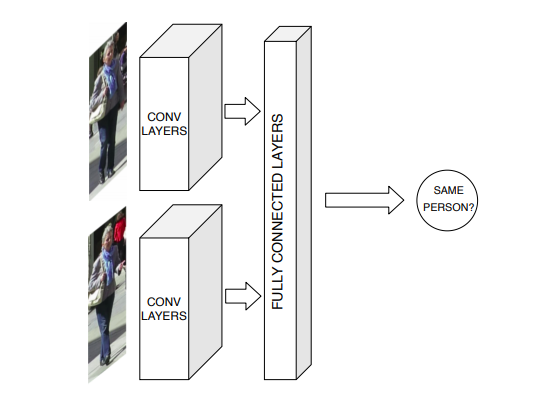
תמונה מספר 37: שלבי אלגוריתם MOT [15].

ניתן לראות בתמונה מספר 37 את שלבי האלגוריתם -MOT העיקריים, חלק מאלגוריתמי -MOT משנים חלק מהשלבים או מאחדים כמה מהם אך העיקרון נשאר זהה: זיהוי האובייקט, ניתוח ולמידת המראה שלו, תיוג האובייקט ומעקב אחריו.

**Siamese networks**

גישה נוספת לפתרון בעיית המעקב אחרי אובייקטים היא השימוש ברשת – CNN עם פונקציית שגיאה המשלבת אינפורמציה משתי תמונות שונות, בשביל ללמוד את סט הפיצ'רים הטוב ביותר שמבדיל בין אובייקט לאובייקט[15].

השם של רשת זו בא מעצם העבודה שאנו מאמנים שני רשתות תואמות עם פרמטרים ומשקלים משותפים בין שתי הרשתות על מנת שנוכל להבדיל בין שני קלטים שונים, ניתן לראות זאת בתמונה מספר 38.



תמונה מספר 38: דוגמא לרשת Siamese [15].

* Kim et al הציע ארכיטקטורת רשת אשר אומנה עם פונקציה המציגה את השוני בין שני תמונות. הרשת קיבלה שני תמונות של אובייקט ,רטיו החיתוך שלהם והאזור שלהם בתמונה.

הרשת חישבה פונקציה המציגה את השוני בין שני התמונות כפלט.

לאחר אימון הרשת הסירו את השכבה שחישבה את פונקציית השוני וכעת השכבה האחרונה ברשת מוציאה וקטור פיצ'רים של התמונה.

לאחר מכן בשלב הסיווג יחשבו את פונקציית השוני בין 2 תמונות באמצעות המרחק בין פיצ'רי הווקטור, רטיו החיתוך שלהם והאזור שלהם בתמונה.

* Zhang et al הציע ארכיטקטורה אשר משתמשת בפונקציית שגיאה בשם SymTriplet loss.

בזמן אימון הרשת CNNs השתמשו במשקלים משותפים ופונקציית השגיאה השתמשה בפונקציית מרחק שנותחה מתוך 2 תמונות השייכות לאותו אובייקט (זוג חיובי ) וגם פונקציית מרחק שנותחה בעזרת תמונה נוספת שאינה שייכת לאותו אובייקט ( 2 זוגות שלילים). פונקציית השגיאה קטנה כאשר המרחק בין שני וקטורי הפיצ'רים של אותו אובייקט היה קטן, לעומת זאת פונקציית השגיאה גדלה כאשר המרחק בין שני וקטורים של אובייקט שונה היה קטן.

* Son et al הציע ארכיטקטורת CNN חדשה בשם Quad-CNN. מודל זה קיבל ארבע תמונות עוקבות כאשר שלוש התמונות הראשונות שייכות לאותו אדם בסדר פריימים עולה, התמונה הרביעית והאחרונה בסדרה שייכת לאדם אחר.

הרשת אומנה עם פונקציית שגיאה הכוללת את האינפורמציה על המרחק בין התמונות, פיצ'רים ויזואליים ומיקום Bounding box. בשלב בדיקת הרשת, הרשת הסתכלה על 2 תמונות של אנשים וחזתה את ההסתברות שב2 התמונות נמצא אותו אדם.

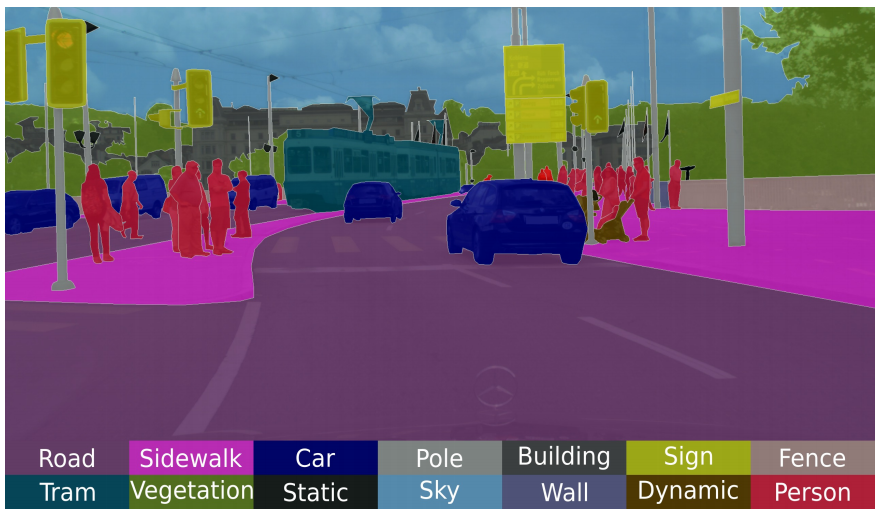
אלו רק כמה שימושים אשר נעזרים בארכיטקטורת ה Siamese Network, ניתן למצוא שימושים לאלגוריתם זה בנושאים רבים אחרים כמו מערכות אבטחה אשר לומדות לזהות דפוסי פנים של אדם ספציפי בשביל להעניק לו גישה לרכב, לטלפון, לבית שלו.

**Semantic Segmentation**

אחת המשימות של ראייה ממוחשבת במכוניות אוטונומיות היא המשימה לסרוק ולהבין את הסביבה בצורה איכותית כדי שהמכונית האוטונומית תוכל לבצע פעולות מורכבות התלויות בסביבה שלה כמו שאדם מתבונן- קולט את הסביבה ופועל לפיה[5].

מטרת הסגמנטציה היא לתייג כל פיקסל בתמונה באמצעות תוויות סמנטיות הנקבעות מראש. סגמנטציה של תמונות עירוניות לחלקים סמנטיים בדרך כלל מציגות תוויות כמו: מכוניות, הולכי רגל, כביש, מדרכה, בניינים, רמזורים, תמרורים ועוד. בעזרת תיוגים אלו אנו מקבלים אינפורמציה מקיפה על האזור בו המכונית נמצאת.

משימה זו גם איננה פשוטה בגלל המורכבות של תמונות עירוניות המלאה בפרטים, חסימה חלקית של אובייקטים, אובייקטים קטנים יחסית וגם הכמות הגדולה של התוויות שאותן נצטרך לקבוע מראש בשביל לבצע את הסגמנטציה.



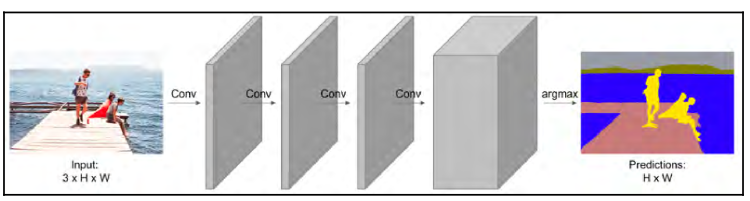
תמונה מספר 39: סגמנטציה של תמונה עירונית [5].

ניתן לראות בתמונה מספר 39 כיצד מבוצעת הסגמנטציה לתמונה עירונית, ניתן לראות את מבחר התוויות בתחתית התמונה אשר הוגדרו מראש לשם הבנה מעמיקה יותר של מרחב התמונה העירונית, ניתן לראות כי כל פיקסל בתמונה מתויג כחלק מתוויות וזו בדיוק המטרה של אלגוריתם זה.

ישנם כמה אלגוריתמים המשתמשים בסוגים של רשתות CNN אציין כמה מהם[14,4]:

* **CNN** – רשת קונבולוציה רגילה אשר נסקרה בעבר המשתמשת בשכבות Sub – Sampling לדוגמא:Max pooling אשר מקטינות את רזולוציית התמונה, אך למשימה הסמנטיקה אנו רוצים לבצע חיזוי על הגודל המלא של התמונה כדי שנוכל באמת להבין את הסביבה שנסרקה .
* **FCN – Fully Convolutional Netwo**r**k** רשת זו מציעה להשתמש בשכבות קונבולוציה בלבד ללא שימוש בשכבות Sub-Sampling ועיבוי שכבת ה Softmax האחרונה ברשת לשכבה בעומק של מספר הקטגוריות- תוויות אותן אנו רוצים לסווג, ניתן לראות דוגמא לרשת זו בתמונה מספר 40.

החיסרון בשכבה זו היא עלות החישוביות שהיא גבוהה מאוד.



תמונה מספר 40: סגמנטציה של תמונה עירונית [14].

* **The SegNet** – גישה זו עובדת בשיטה קצת שונה, שיטת ה- Encoder-Decoder.

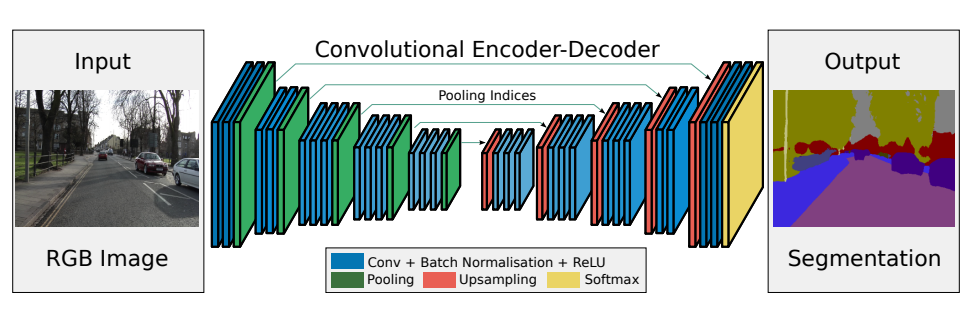
רשת זו משתמש בשכבות קונבולוציה ובשכבות Sub-Sampling שנחשבות

ה-Encoder אשר מקטינות את הרזולוציה ובשכבות Up-Sampling שנחשבות

ה-Decoder אשר הופכות חזרה את פעולת ה-Encoder ומחזירות את הקלט לרזולוציה הנקלטת בשכבה הראשונה ברשת.

כך יצא מצב שאנו נמנעים מחישובים רבים כמו ברשת ה-FCN ועדיין שומרים על הרזולוציה של התמונה.

ניתן לראות בתמונה מספר 41 כיצד רזולוציית התמונה קטנה בהתחלת הרשת ולבסוף גדלה חזרה לגודלה המקורי.



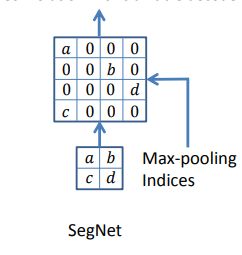
תמונה מספר 41: הדגמה לרשת SegNet [17].

ארכיטקטורת רשת זו בנויה מ13 שכבות קונבולוציה אשר מתאימות ל13 השכבות הראשונות בארכיטקטורת הרשת [VGG16](https://arxiv.org/pdf/1505.06798.pdf) אשר מטרתם היא סיווג אובייקטים.

לפני כל שכבת 2X2 MaxPooling שומרים את המיקום שבו נמצא הערך הגדול ביותר בכל החלקה של הפילטר על התמונה זהו שלב הEncoder.

בשלב הDecoder מבוצע התהליך ההפוך – Upsampling ואז עוברים לשלב הקונבולוציה.

בשלב הDecoder משתמשים במקומות ששמרנו אשר בשביל להגדיל את ממדי התמונה כמתואר בתמונה מספר 42.



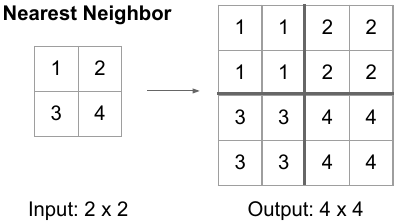
תמונה מספר 42: Upsampling ברשת SegNet [17].

ניתן לראות בתמונה מספר 42 כי נשמרו המיקומים של הערכים הגדולים ביותר בשלב הMaxPooling ובאמצעות שמירת ערכים אלה נוכל לבצע "חזרה לאחור" לממדי התמונה המקורית אשר התקבלה בשכבה הראשונה.

לבסוף השכבה האחרונה ברשת זו היא שכבת Softmax המאפשרת לחזות קטגוריה לכל פיקסל בתמונה.

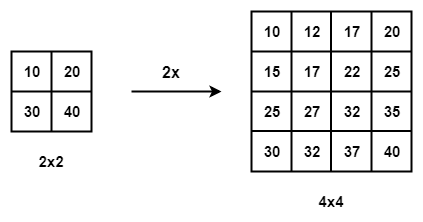
ישנן עוד טכניקות UpSampling פופולריות אתאר כמה מהן[2]:

* **Nearest Neighbors**: טכניקה יחסית פשוטה הלוקחת ערך של פיקסל מסוים ומעתיקה אותו ל X פיקסלים שכנים שלו כמתואר בתמונה מספר 43.



תמונה מספר 43: טכניקת Nearest Neighbors [2].

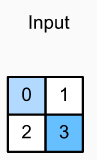
* **Bi-Linear Interpolation**: בטכניקת אינטרפולציה זו המטרה שלנו היא לשמור על הממוצע המשוקלל של מרחקי הפיקסלים הקיימים ובמקביל להגדיל את רזולוציית התמונה, ניתן לראות בתמונה מספר 44 כיצד מתבצעת הפעולה של הגדלת מטריצת הפיקסלים בגודל 2X2 לגודל של 4X4 באמצעות שמירת ממוצע המרחקים כתלות בגודל ההגדלה.



* תמונה מספר 44: טכניקת Bi-Linear Interpolation [2].

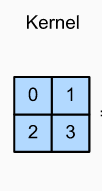
**Transposed Convolutions:**  טכניקה זו היא פופולרית במיוחד בתחום ה- Semantic Segmentation, טכניקה זו פופולרית מכיוון שניתן לאפשר לרשת ללמוד את הפרמטרים שבעזרתם היא מבצעת את פעולת ה- UpSampling. טכניקה זו פועלת בצורה הבאה:

ניקח מטריצת פיקסלים בגודל 2X2 כמתואר בתמונה 45 ונרצה להגדילה ל3X3 .



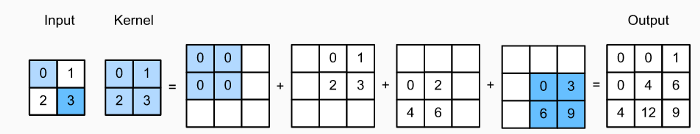
* תמונה מספר 45: מטריצת פיקסלים לפני הגדלה [2].

נוכל לקחת פילטר בגודל 2X2 עם פרמטרים אשר ניתנים ללמידה, בדוגמא זאת ניקח פילטר זהה לתמונה שאותה נרצה להגדיל כמתואר בתמונה מספר 41.



* תמונה מספר 41: פילטר הזהה לתמונה [2].

כעת נוכל לקחת את הפילטר ולבצע הכפלה שלו בכל אחד מערכי הפיקסלים בתמונה שנרצה להגדיל, נבצע פעולה זאת על גבי כל אחד מהפיקסלים ובסוף נסכום את כולם למטריצת הפלט בגודל 3X3 כמתואר בתמונה 42



* תמונה מספר 42: פעולת הTransposed Convolution [2].

**יישום : זיהוי תמרורים ברכבים אוטונומיים**

**הגדרת הבעיה:**

תמרורי דרך, להתגבר על קשיים כמו: תנאי ראות שונים (חושך, גשם ועוד) , תמרורים דהויים וה כרה מלאה של כל סוגי התמרוריםוכבעת זיהוי תמרור מסוים2

ן

**גישת הפתרון :**

החלטתי לבנות מודל – CNN לזיהוי, סיווג תמונות של תמרורים באמצעות הטכנולוגיה אשר סקרתי במהלך סמינר זה.

בחרתי להשתמש בשפה Python ובספרייה TensorFlow לשם בניית המודל.

**בסיס הנתונים:**

השתמשתי בבסיס הנתונים :

GTSRB - German Traffic Sign Recognition Benchmark

אשר משומש גם במאמר 10 ברשימת המקורות.

בסיס הנתונים הזה פורסם לשם שימוש בתחרות -international Joint Conference on Neural Networks

אשר נערכה בשנת 2011 ומטרתה היא להצליח לבנות מודל אשר יצליח לדייק בזיהוי 43 סוגים שונים של תמרורים מתוך כ50 אלף תמונות שונות של תמרורי תנועה מגרמניה.

43

43

ניתן לראות גם כי חלק מהתמרורים מטושטשים, חשוכים, לא ברורים.

בבסיס נתונים זה ישנם 39,209 תמונות שונות לשם אימון המודל, 12,630 תמונות לשם בחינת המודל לערך סיום האימון.

**מימוש:**

CNN מתוך

ראשית במסד הנתונים המיועדים לאימון המודל

. לאחר בחינת התמונות נראה כי ישנן תמונות מטושטשות ולא ברורות אשר יקשו על המודל בזיהויים בחרתיל

לאחר מכן בדקתי את גדלי התמונות, גיליתי שהגובה הנפוץ ביותר בקרב תמונות אלו הוא 33, האורך הנפוץ ביותר בקרב תמונות אלו הוא 34 אך גם 33 די נפוץ לכן בחרתי לקבוע שכל תמונה תהיה בגודל 33.

ולהלן תיאורה

1. X.

בשביל למנוע Overfit השתמשתי בשתי טכניקות:

1. השתמשתי בפונקציית CallBack אשר מפסיקה את אימון הרשת כאשר אחוזי הדיוק על סט האימון מגיע לפרמטר מסוים, בחרתי פרמטר זה להיות 98.5.
2. בחרתי להשתמש בשכבת Dropout עם הסתברות 0.5.

השתמשתי בכמה Hyperparameters חשובים והם:

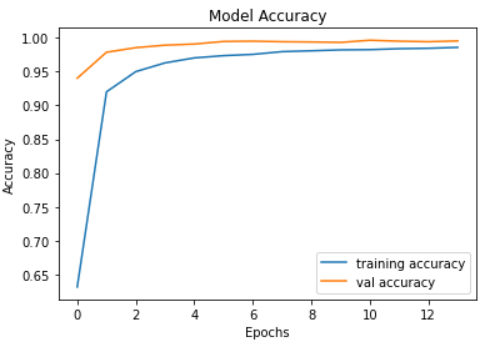
* Learning Rate – החלטתי להשאיר את פרמטר זה להיות הערך הדיפולטי של TensorFlow אשר ערכו הוא 0.001.
* Number of Epochs – בחרתי לאמן את המודל עד 20 מעברים או עד אשר אגיע לדיוק של 98.5% על סט האימון.
* Batch Size – פרמטר זה קובע אחרי כמה קלטים שעברו ברשת צריך לכוון את משקלי הרשת לפי תהליך ה – Gradient Descent, בחרתי במספר 32 שהוא די נפוץ ולפי-כך כל Epoch חולק ל981 מעברים שלמים על הרשת.
* Hidden Layers- השתמשתי ב3 שכבות קונבולוציה, 3 שכבות MaxPooling, שכבת Dropout, שכבת Dense- שכבת נוירונים רגילה, שכבת Dense אחרונה לסיווג הקלט ל43 קטגוריות התמרורים.
* Activation Functions – בחרתי להשתמש בפונקציית האקטיבציה -Relu לשכבות הפנימיות ובפונקצית האקטיבציה – Softmax אשר תסווג את הפלט שהועבר ברשת לקטגוריות בשכבה האחרונה של הרשת.

**תוצאות:**

המודל אומן במשך 14 Epochs ונעצר ברגע שעבר את אחוזי הדיוק על סט נתוני האימון שהגדרתי לו.

לשמחתי המודל השיג תוצאות טובות מאוד יחסית לארכיטקטורה פשוטה שכזו וזמן אימון מועט:

* 98.5% דיוק על סט האימון.
* 99.5% דיוק על סט הבדיקה.



תמונה מספר 44: המציגה גרף של דיוק המודל לפי Epochs .

ניתן לראות בתמונה מספר 44 את אחוזי הדיוק של המודל במהלך אימונו.

תוצאות אלו מפתיעות למדי ולכן החלטתי לבצע בדיקה נוספת אל מול הTest Set אשר נמצא במסד נתונים זה, בבדיקה זאת בחנתי את המודל על תמונות שאינו ראה בעבר והמודל הגיע לדיוק של %96.2 אחוזים, מרשים ביותר.

אך האם מודל זה באמת כזה מושלם?

החלטתי לנסות לחקור את המודל ובסיס הנתונים הזה קצת יותר לעומק:

בחרתי שתי תמונות של השלט Stop אשר תוויתו היא מספר 14 במודל שלנו ורציתי לחקור האם המודל ידע לזהות שני סוגים של תמרור עצור.



תמונה מספר 45-46: תמרורי עצור שונים[https://en.wikipedia.org/wiki/Stop\_sign][ https://www.safetysupplywarehouse.com/STOP\_Sign\_Floor\_Decals\_p/wfs3.htm] .

ניתן לראות בתמונות מספר 45-46 שני סוגים של תמרורי עצור כאשר התמרור הימני מקורו באוסטרליה.

למרבה ההפתעה המודל זיהה את התמרור הימני כתמרור עצור וסיווג אותו בקטגוריה הנכונה -14 אך המודל זיהה את התמרור השמאלי כתמרור מקטגוריה 13.

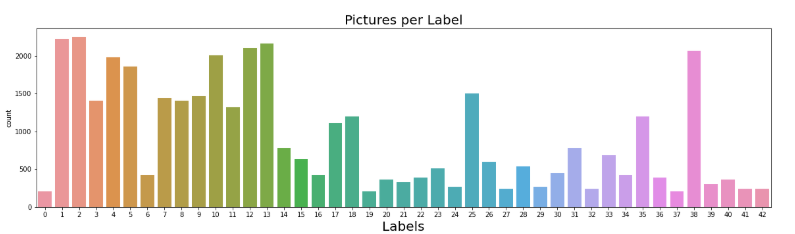
קטגוריה מספר 13 מציגה תמרור יחסית דומה:



תמונה מספר 47 : תמרור מקטגוריה 13 במסד הנתונים.

זיהוי שגוי זה במקביל לאחוזי דיוק גבוהים מאוד על בסיס הנתונים יכול לרמז על אפשרות ל – OverFit.

החלטתי לבדוק את כמות התמונות שיש לכל קטגוריה במסד הנתונים וגיליתי נתון מפתיע:



תמונה מספר 48: מספר התמונות מפולחות לפי כל אחת מ43 הקטגוריות.

נראה בתמונה 48 כי ישנו חוסר איזון גדול בין מספר התמונות ששייכות לכל תווית במסד הנתונים, ניתן לראות כי ישנם תמרורים עם מספר תמונות גדול מאוד כמו תמרור מספר 13 שהוצג לעיל לעומת תמרורים כמו 14,0,19 ועוד אשר כמות התמונות שלהם קטנה משמעותית.

**מסקנות וסיכום:**

בניתי מודל – CNN פשוט למדיי מבחינת ארכיטקטורה אשר הציג ביצועים מצוינים על סט הנתונים שאיתו עבדתי.

* 98.5% דיוק על סט האימון.
* 99.5% דיוק על סט הבדיקה.
* 96.2% דיוק על סט הבחינה.

לאחר בדיקה מעמיקה של מסד הנתונים ניתן לראות שהוא אינו מאוזן מבחינת כמות התמונות השייכות לכל אחת מה-43 קטגוריות לסיווג, כתוצאה מכך נראה כי משקלי המודל נוטים לכיוון הקטגוריות שבהן קיימות יותר תמונות.

ניתן להסיק מהבדיקה המעמיקה של קטגוריות 13,14 שתי אפשרויות:

1. אין במסד הנתונים של תמרורי התנועה בגרמניה תמרורי תנועה אשר מציגים תנועת יד בנוסף לטקסט "Stop" ולכן נסיק כי כל רכב אוטונומי יצטרך ללמוד תמרורים מארצות אחרות ולא רק בארץ בו הוא נקנה למקרה בו נרצה לנסוע עם הרכב למחוץ לארץ.
2. במסד הנתונים ישנו מצב של Overfit אשר קטגוריה 14 למדה לזהות אך ורק תמרורי עצור עם המילה "Stop" בלבד. ניתן לראות דמיון בין הקטגוריות 13,14 בצבעי התמונה וניתן לשער שהסיבה לכך שהמודל חשב שתמונה מספר 38 שייכת לקטגוריה 13 היא השטח הלבן באמצע השלט המציג את תנועת היד.

לאחר בדיקה נוספת של מסד הנתונים התברר כי ההשערה הראשונה נכונה, כל תמונות התמרורים בקטגוריה 14 הם תמרורים אשר מציגים אך ורק את המילה "Stop" ללא תנועת יד נוספת, ניתן לראות זאת בתמונה 49.



תמונה מספר 49 : תמרורי עצור מקטגוריה 13 במסד הנתונים.

כעת נשאלת השאלה האם אנו רוצים לבנות מודל לזיהוי תמרורים לפי נראות התמרורים בגרמניה או האם אנו רוצים להרחיב מודל זה לזיהוי תמרורים במדינות נוספות?

אם נרצה לקחת את מודל זה ולהרחיב את יכולותיו מעבר לתחומי גרמניה בוודאי שנצטרך לטפל בבעיה של זיהוי תמרורים אשר נראים שונה בכל מדינה.

ניקח לדוגמא את בעיית זיהוי התמרור עצור הנדונה לעיל ותמרור עצור בישראל אשר מוצג בתמונה 50.



תמונה מספר 50 : תמרור עצור במדינת ישראל[https://bit.ly/31d3428] .

על מנת לפתור בעיה זו נצטרך לאסוף עוד הרבה תמונות של תמרור העצור המוצג בישראל כדי שנוכל לאמן את המודל לזהות גם תמרורי עצור מהצורה בישראל.

אציין כי על מנת למנוע Overfit לתמרורי העצור בגרמניה נצטרך לאזן את מספר תמונות תמרורי העצור בגרמניה מול מספר תמונות תמרורי העצור בישראל, נוכל להשתמש גם בטכניקת Data Augmentation על מנת להגדיל את מספר תמונות תמרורי העצור בישראל שאספנו.

באמצעות הטכניקה שתוארה לעיל נוכל לטפל בבעיות זיהוי של תמרורים נוספים אשר נראים בצורה שונה במדינות שבהם מודל זה יפעל.

המסקנה המרכזית שאני מסיק מבניית מודל זה היא שמשימת בניית מערכת אשר תזהה תמרורי דרך (בלי תלות במדינה) אינה משימה פשוטה כמו שהיא נשמעת.

השימוש בארכיטקטורת CNN די פשוטה הביא אותנו לדיוק גבוה למדי אך בסיס נתונים זה רלוונטי לתמרורי הדרך הנפוצים בגרמניה, אם נרצה לבנות מודל שידע לזהות, לסווג תמרורי דרך בלי תלות במדינה נצטרך להשיג עוד הרבה Data של תמרורים מרחבי העולם. במידה ונאמן את המודל הזה עם יותר קטגוריות ויותר תמרורים אני יכול להניח שאחוז דיוקו ירד ונזדקק למודל מתוחכם יותר.

נקודה אחרונה שאני רוצה להעלות היא הצורך בדיוק מושלם במערכת כזו במכונית אוטונומיית, מודל שמדייק ב99% עלול לפספס תמרור עצור וכתוצאה מכך לגרום לתוצאה פאטלית.

ניתן לצפות במודל זה במלואו באתר Kaggle באמצעות לחיצה [כאן](https://www.kaggle.com/stavcohen/predicting-traffic-signs)

**סיכום ומסקנות**

בסמינר זה הצגתי את האתגרים בתכנון מכוניות אוטונומיות, את עולם למידת המכונה, הלמידה העמוקה, ולבסוף סקרתי כמה מערכות ראייה ממוחשבת במכוניות אוטונומיות.

התחלתי מהסבר בסיסי על מאין הגיע וכיצד התפתח הרעיון של רשתות נוירונים, הצגתי כיצד כמה נוירונים עובדים ואיך עובדת פונקציות האקטיביציה שלהם.

לאחר מכן פיתחתי את הרעיון של חיבור כמה נוירונים לרשת אחת שלמה המדמה רשת נוירונים ביולוגית. הראיתי איזה סוגי למידת מכונה קיימים, תוך כניסה לעומק המתמטי של כיצד ניתן לאמן רשת נוירונים באמצעות אלגוריתם ה- Gradient Descent ואיך באמצעות אימון רשתות נוירונים המחשב לומד לבצע משימות מורכבות. פרק זה נתן בסיס בשביל להמשיך לסקור רשת נוירונים נוספת אשר הינן בעלת ארכיטקטורה שונה המותאמת לעיבוד תמונות.

התחלתי בלסקור כיצד מיוצגת תמונה דיגיטלית במחשב והמשכתי לארכיטקטורת ה- CNN שבעזרתה נוכל לאפשר למחשב לסווג תמונות דיגיטליות. סקרתי את השכבות המיוחדות המרכיבות רשת זו, הצגתי כמה שיפורים שנעשו על מנת לייעל רשת זאת.

סקרתי את בעיית חילוק הנתונים לשם אימון המודל, התקדמתי להצגת בעיית ה- OverFit וכיצד ניתן להימנע ממנה ברשתות קונבולוציה באמצעות כיבוי נוירונים ואוגמנטציית תמונה פשוטה.

הצגתי את הסנסורים הנפוצים הנמצאים ברכבים אוטונומיים וסקרתי שלוש מערכות ראייה ממוחשבת הנמצאות בכל רכב אוטונומי המעניקות לו את היכולת "לראות" את סביבתו.

לאחר סקירת שלוש המערכות שציינתי שהן רק חלק מהמערכות ראייה ממוחשבת שנדרשות למכוניות אוטונומיות, ניתן להבין את הצורך במערכות ראייה במכוניות אוטונומיות.

בניתי מודל CNN המשתמש בטכנולוגית הנסקרות בסמינר ומציג דוגמא מצומצמת מאחת מערכות הראייה הממוחשבות שנמצאות בכל מכונית אוטונומית.

מערכות הראייה הממוחשבת הן האמצעי דרכו המכונית האוטונומית קולטת את הסביבה שלה ומגיבה על פיה בצורה דומה לצורה שבה אנחנו כבני אדם מגיבים לסביבה.

מערכות אלה הן צעד ראשוני בלבד בתכנון מכוניות אוטונומיות. לאחר קליטת המידע ועיבודו מגיע החלק שבו המכונית צריכה להחליט כיצד להגיב לשלל הסיטואציות היכולות לקרות בסביבתה. משימה זו אינה פשוטה בכלל ויכולה להציג גם דילמות מוסריות כמו בעיית "Who dies and Who lives?" אשר סקרתי בתחילת הסמינר.

בעוד שטכנולוגית המכוניות האוטונומיות משתפרת כל הזמן עדיין קשה לחזות האם ומתי מכוניות אוטונומיות יהפכו להיות חלק משגרת חיינו. אנו בעצם מנסים לבנות מכונה שתצליח להשתוות לביצועים שלנו כבני אדם ולהתחקות אחרי "המחשב" המסובך ביותר בעולם – המוח שלנו, משימה זו בהחלט מסובכת יותר ממה שהיא נשמעת.

ישנם המון אתגרים שעומדים לפנינו, לדוגמא: הדיוק המושלם בראיית הסביבה שמכונית אוטונומית צריכה להשיג בשביל להימנע מפגיעה בסביבה או בעצמה, העמידות של המערכות מול תופעות טבע משתנות כמו: גשם, שלג, ערפל, חושך, מגנטיות חזקה ועוד אשר יכולות להקשות על הסנסורים של המכונית. אתגר גדול שלא סקרתי בסמינר הינו היכולת של המכונית לבצע החלטות קריטיות בזמן קצר מאוד, האם מכונית תוכל להגיב במהירות כמו מערכת האינסטינקטים של האדם?

אתגר מעניין נוסף שאינו נסקר במהלך הסמינר הוא השאלה אם נניח שמכוניות אוטונומיות היו קיימות, האם היית מוכן "להסתכן" בלאפשר למכונית כזו לנהוג עבורך?

ובנימה אישית, נהנתי מאוד לחקור כיצד ראייה ממוחשבת עובדת במכוניות אוטונומיות ולכתוב את הסמינר. ניסיתי להנגיש את החומר כך שגם מי שאינו בקיא בתחום של למידת מכונה יוכל להנות לקרוא את הסמינר ולקבל הצצה אל עולם הבינה המלאכותית, למידת מכונה וראייה במכוניות אוטונומיות.

סמינר זה גרם לי לרצות להמשיך לחקור את עולם הבינה מלאכותית ולבחור לעשות את התואר השני שלי בתחום זה.

**ביבליוגרפיה**

1. [Ciaburro, Giuseppe, Venkateswaran, Balaji (2017):Neural Networks with R](https://search-ebscohost-com.elib.openu.ac.il/login.aspx?direct=true&db=nlebk&AN=1607842&site=eds-live&ebv=EB&ppid=pp_Cover)
2. [Divyanshu Mishra: Transposed Convolution Demystified](https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba)
3. [Ethan Yanjia Li: Dive Really Deep into YOLO v3: A Beginner’s Guide](https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e)
4. [Fatma Güney, Aseem Behl, Andreas Geiger (2020): Computer Vision for Autonomous Vehicles : Problems, Datasets and State of the Art](https://search-ebscohost-com.elib.openu.ac.il/login.aspx?direct=true&db=edseee&AN=edseee.9134692&site=eds-live)
5. [Farzeen Munir, Shoaib Azam, Muhammad Ishfaq Hussain, Ahmed Muqeem Sheri (2018): Autonomous Vehicle: The Architecture Aspect of Self Driving Car](https://dl-acm-org.elib.openu.ac.il/doi/10.1145/3290589.3290599) (ACM)
6. [John P. Holdren Megan Smith U.S. Chief Technology Officer (2016): Preparing for the future of ai](https://obamawhitehouse.archives.gov/sites/default/files/whitehouse_files/microsites/ostp/NSTC/preparing_for_the_future_of_ai.pdf)
7. [Karen Hao(2018):Should a self-driving car kill the baby or the grandma? Depends on where you’re from](https://www.technologyreview.com/2018/10/24/139313/a-global-ethics-study-aims-to-help-ai-solve-the-self-driving-trolley-problem/)
8. [Michael Nielsen (2019):Neural Networks and Deep Learning](http://neuralnetworksanddeeplearning.com/about.html)
9. [Niranjan Kumar (2019) : Sigmoid Neuron](https://towardsdatascience.com/sigmoid-neuron-deep-neural-networks-a4cd35b629d7)
10. [Parul Pandey (2019) :Understanding the Mathematics behind Gradient Descent.](https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e)
11. [Patrizio Pelliccione , Avenir Kobetski , Tony Larsson , Maytheewat Aramrattana (2016): Architecting cars as constituents of a system of systems](https://dl-acm-org.elib.openu.ac.il/doi/10.1145/3175731.3175733) (ACM)
12. [Qiang Chen, Shaoyu Wang\*, Lirong Yi (2019) : Traffic signs recognition using dynamic-scale CNN](https://dl-acm-org.elib.openu.ac.il/doi/10.1145/3371425.3371444) (ACM)
13. [Rohith Gandhi (2018) R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms](https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)
14. [Shanmugamani, Rajalingappaa (2018) : Deep Learning for Computer Vision : Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras](https://search-ebscohost-com.elib.openu.ac.il/login.aspx?direct=true&db=nlebk&AN=1699215&site=eds-live&ebv=EB&ppid=pp_Cover)
15. [Siham Tabik , Luigi Troiano , Roberto Tagliaferri , Francisco Herrera, Gioele Ciaparrone, Francisco Luque Sánchez (2019): DEEP LEARNING IN VIDEO MULTI-OBJECT TRACKING: A SURVEY](https://arxiv.org/pdf/1907.12740.pdf)
16. [Stanford University - CS231n Convolutional Neural Networks for Visual Recognition](https://cs231n.github.io/convolutional-networks/)
17. [Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, Senior Member, IEEE(2016)\_: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation](https://arxiv.org/pdf/1511.00561.pdf)