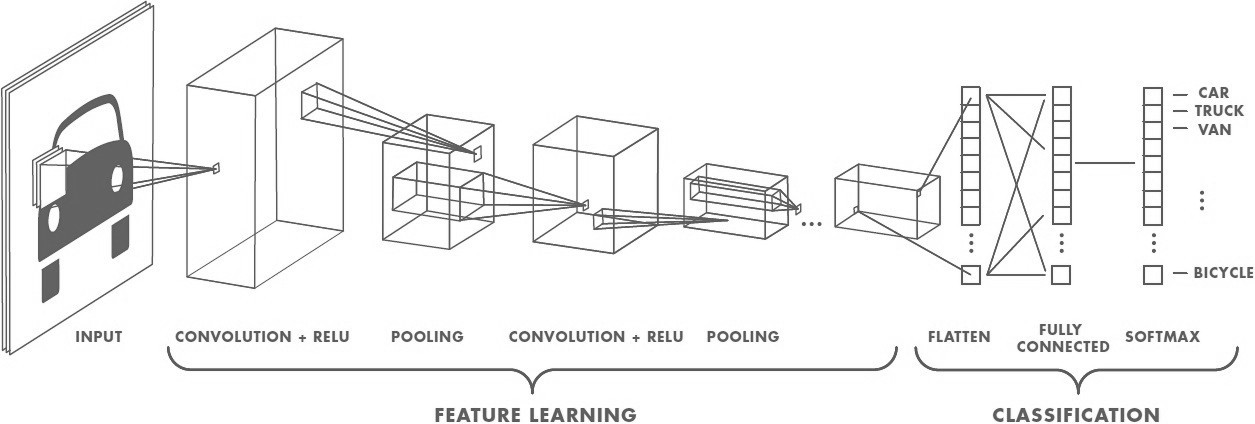
****

**רשת נוירונים עמוקה לזיהוי עוברי אורח במערכת נהיגה אוטונומית**

**Deep Neural Network for Pedestrian detection for autonomous driving**



**מנחה:**

**פרופסור יוסף בן-עזרא**

**מגיש:**

**אסף קסלסי , ת.ז 301467098**

תוכן עניינים

[רשימת איורים: 4](#_Toc41421438)

[1. **מבוא** 5](#_Toc41421439)

[אבני דרך בציר הזמן של רשתות נוירונים ולמידת מכונה 5](#_Toc41421440)

[רקע לרשתות נוירונים 6](#_Toc41421441)

[הגדרת רשת נוירונים 6](#_Toc41421442)

[כיצד רשת נוירונים דומה למוח ? 6](#_Toc41421443)

[רשתות נוירונים מלאכותיות 7](#_Toc41421444)

[ייצוג 7](#_Toc41421445)

[סוגים של פונקציות הפעלה 8](#_Toc41421446)

[1. פונקצית סף function) threshold ) 8](#_Toc41421447)

[2. פונקצית סיגמוייד ( function (sigmoid 8](#_Toc41421448)

[3. פונקצית המיישר (ReLU function) 9](#_Toc41421449)

[4. פונקצית מיישר דולפת (leaky ReLU function) 9](#_Toc41421450)

[5. פונקצית מיישר פרמטרית (Parametric ReLU function) 10](#_Toc41421451)

[ארכיטקטורות רשת 11](#_Toc41421452)

[רשתות חד-שכבתיות ללא משוב 11](#_Toc41421453)

[רשתות רב-שכבתיות ללא משוב 11](#_Toc41421454)

[.3 רשתות חוזרות (Recurrent) 12](#_Toc41421455)

[סוגי בעיות 13](#_Toc41421456)

[למידה 14](#_Toc41421457)

[סגמנטציה 16](#_Toc41421458)

[**2. ארכיטקטורה ואימון:** 18](#_Toc41421459)

[אימון הרשת – זמן כולל ופלטופרמת המערכת. 19](#_Toc41421460)

[כמות פרמטרים הכוללת ברשת 20](file:///C:\Users\Assaf\Desktop\פרוייקט_גמר_אסף\AsafKaslassy_DeepLearning\Project%20book_firstDraft_26-5.docx#_Toc41421461)

[ארכיטקטורת הרשת: 21](#_Toc41421462)

[מניעת Over Fitting 22](#_Toc41421463)

[1. אתחול משקולות 22](#_Toc41421464)

[Batch normalization.2 22](#_Toc41421465)

[Dropout.3 23](#_Toc41421466)

[MaxPooling2d.4 23](#_Toc41421467)

[Back-propagation 25](#_Toc41421468)

[תאור הארכיטקטורה של הרשת: 28](#_Toc41421469)

[3. סיכום 30](#_Toc41421470)

[דרכים לשיפור הרשת: 31](#_Toc41421471)

[**ביבליוגרפיה** 32](#_Toc41421472)

# רשימת איורים:

איור 1 - אבני דרך בציר הזמן של רשתות נוירונים ולמידת מכונה

איור 2 - השוואה בין רשת בגוף האדם לרשת נוירונים מלאכותית

איור 3 - מודל של נוירון

איור 4 -

איור 5 -

איור 6 -

איור 7 -

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

איור 8 - Parametric ReLU function

# 1. מבוא

למידה חישובית (Machine Learning ) הינה תחום במדע החוקר מצב בו אלגוריתמים יכולים "ללמוד" מה נכון לעשות ומה לא נכון לעשות מתוך אוסף נתון שלdata - מידע. למידה חישובית היא כלי חשוב בתחום הAI . אלגוריתמים אלו משתמשים במודל המבוסס על input אשר לפיו הם חוזים או מחליטים. זה כמובן שונה מאלגוריתמים שלהם אין את האפשרות ללמוד מנתונים אלא רק עובדים על ידי ביצוע של סט פקודות ברורות ומוגדרות מלכתחילה.

טום מ. מיטשל בסיפרו *Machine Learning* הגדיר את למידת המכונה באופן הבא:

*"תוכנית מחשב תיקרא לומדת מניסיון E ביחס למחלקת משימות T ומדד ביצועים P , אם הביצועים של משימות ב- T בהתאם למדד P ,משתפרים עם הניסיון E. [1]*

## אבני דרך בציר הזמן של רשתות נוירונים ולמידת מכונה

***איור 1***

# 2. רקע לרשתות נוירונים

## הגדרת רשת נוירונים

מחקרים ומאמרים רבים אודות שימוש ברשתות ניורונים לחישוב ופתרון בעיות בעזרת מחשב נבעו מההבנה שהמוח האנושי מבצע חישובים בצורה שונה מהדרך בה מתבצעים חישובים ע"י מחשב דיגיטלי קונבנציונלי. המוח מאוד מורכב , עובד בצורה אי-ליניארית ומקבילית. יכול לארגן את הנוירונים , בצורה ייחודית המאפשרת זיהוי תבניות, שליפה מזיכרון, יכולות מוטוריות וכו' במהירות גדולה בכמה סדרי גדול מהמחשב המהיר ביותר כיום.

artificial neural networks בנויות כך שהן יכולות "ללמוד" מניסיון העבר, בכל רשת יש הרבה מאוד קשרים בין נוירונים מלאכותיים (או יחידות עיבוד) שמשתנים לאורך חיי הרשת כדי להפיק ביצועים טובים וחישוב יעיל.

## כיצד רשת נוירונים דומה למוח ?

רשת נוירונים מלאכותית דומה למוח בשני אופנים עיקריים:

רכישת ידע באמצעות תהליך למידה ואחסון מידע באמצעות חיבורים בין נוירונים (משקלים סינפטיים).

באמצעות אלגוריתם למידה משתנים המשקלי הרשת בצורה איטרטיבית בכדי להגיע למטרה שדרשנו מהרשת.

הקשרים עצמם קבועים אך המשקל של כל קשר, או החוזק שלו, משתנה (תהליך דומה למוח ).

קיימות גם רשתות שיכולות לשנות את הארכיטקטורה והמבנה של עצמן על ידי יצירת קשרים חדשים

)תהליך נוסף הדומה למוח)**.**[***[4]***](#_ביבליוגרפיה)

# רשתות נוירונים מלאכותיות

***איור* 2**

### ייצוג

הנוירון הוא יחידת עיבוד המידע הבסיסית הנחוצה לתפקוד רשת הנוירונים. באיור ניתן לראות מודל של נוירון, היוצר את הבסיס לעיצוב רשת נוירונים מלאכותית. ניתן לזהות 4 אלמנטים בסיסיים במודל

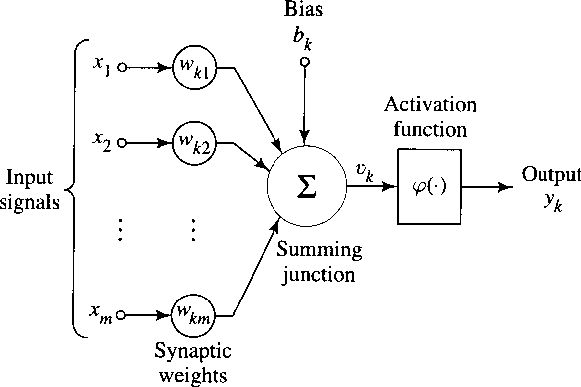
המוצג:

1. סט של סינפסות אשר כל אחת מהן מאופיינת במשקל משלה. כלומר אות כלשהו ,xm המגיע בתור קלט לסינפסה j המחוברת לנוירון ,k מוכפל במשקל הסינפטי wkm

2. מחבר (adder) של אותות הinput

.3 פונקצית הפעלה function activation הגורמת שפונקצית הinput של הנוירון יכולה להיות אי-ליניארית.

4.אות הטיה (bias) מסומן בסימון bk מטרת אות זה היא הגברה או הנמכה של האות המסוכם טרם כניסתו לפונקצית ההפעלה.



***איור* 3**

## סוגים של פונקציות הפעלה

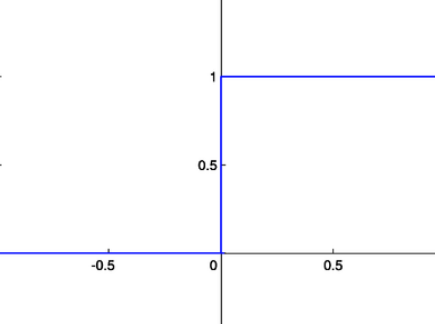
פונקציות ההפעלה מגדירה את הפלט של הנוירון כתלות בפוטנציאל ההפעלה שלו . ישנם סוגים

רבים של פונקציות הפעלה, אפרט על הבסיסים ועל פונקצית ההפעלה שבחרתי בפרויקט.

### 1. פונקצית סף function) threshold )

פונקציה זו עוזרת בהחלטה האם יש להפעיל את הנוירון או לא על סמך הערך מהטרנספורמציה הליניארית. במילים אחרות, אם הקלט לפונקצית ההפעלה גדול מסף מסוים, אז הנוירון מופעל, אחרת הוא מבוטל, כלומר הפלט שלו לא נחשב לשכבה הסמויה הבאה.

מוגדרת בצורה הבאה:



def threshold \_function(x):

if x<0:

return 0

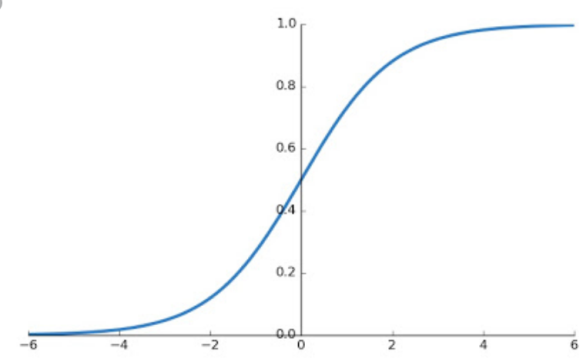
else:

return 1

***איור* 4**

### 2. פונקצית סיגמוייד ( function (sigmoid

הגרף שלה דומה וויזואלית לאות ,S אחת הפונקציות הנפוצות ביותר בבניית רשתות. מוגדרת כפונקציה עולה-ממש, ואחת התכונות המיוחדות שלה שבתחום מוגדר אפשר להגיד שמתפקדת כפונקציה ליניארית ובאזורים אחרים כפונקציה אי-ליניארית. (אחד ההבדלים בין פונקציית סיגמוייד לבין פונקצית אקטיבציה מסוג "threshold" היא תכונת הגזירות שכן סיגמוייד גזירה ופונקצית סף לא גזירה.(

****

def sigmoid\_function(x):

z = (1/(1 + np.exp(-x)))

return z

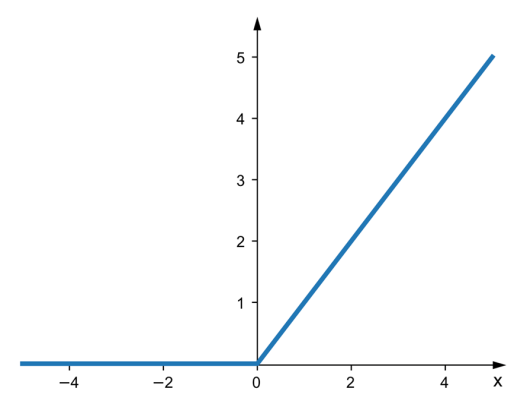
***איור* 5**

### 3. פונקצית המיישר (ReLU function)

### 

פונקצית הפעלה המוגדרת כהחלק החיובי בארגומנט שלה , כאשר x הוא הקלט לנוירון, ידועה גם כפונקצית רמפה (Ramp function) והיא מקבילה לתיקון חצי גל בעולם הנדסת החשמל.

אחד היתרונות של פונקציה זו הוא היכולת להעלים את כל החלק השלילי ביעילות על ידי הגדרתו כאפס - מה שמגביר את תכונות האי-ליניאריות שפירטתי בפרק הקודם.



def relu\_function(x):

if x<0:

return 0

else:

return x

***איור 6***

למרות מספר חסרונות שאפרט בהמשך, ReLU (נכון לשנת 2020) היא פונקצית ההפעלה הפופולארית ביותר עבור רשתות עצביות עמוקות.

חסרונות ReLU והסיבה שרבים משתמשים בגרסאות שונות שלה  
לא ניתנת להבחנה באפס, אך עם זאת, ניתן להבחין בה בכל מקום אחר, וניתן לבחור באופן שרירותי את הערך של הנגזרת באפס להיות 0 או 1, ללא גבול,

בעיית ReLU גוססת: לעיתים ניתן לדחוף נוירונים של ReLU למצבים בהם הם הופכים לא פעילים. במצב זה הנוירון נתקע במצב בלתי פעיל כל הזמן ו"מת ". זוהי בעיית Vanishing gradient problem . בחלק מהמקרים, מספר גדול של נוירונים ברשת יכול להיתקע במצבים מתים, ובכך להפחית למעשה את יכולת המודל. בעיה זו מתעוררת בדרך כלל כאשר קצב הלמידה מוגדר גבוה מדי. במקום זאת ניתן להקל על ידי שימוש ב- ReLUs דולפים,

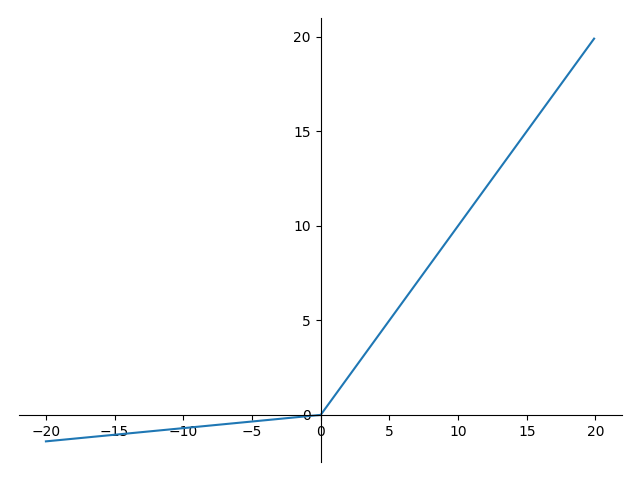
המקצים שיפוע חיובי קטן עבור x < 0

### 4. פונקצית מיישר דולפת (leaky ReLU function)

פונקצית הפעלה מאוד דומה לReLU הרגילה אך הבדל משמעותי קורה בערכים השליליים:

היא מאפשרת 0.01 של גרדיאנט חיובי כשהיחידה לא אקטיבית *.*[*[11]*](#_ביבליוגרפיה)

אחד היתרונות של פונקציה זו הוא היכולת להעלים את כל החלק השלילי ביעילות על ידי הגדרתו כאפס - מה שמגביר את תכונות האי-ליניאריות שפירטתי בפרק הקודם.



***איור 7***

*def leaky\_relu\_function(x):*

*if x<0:*

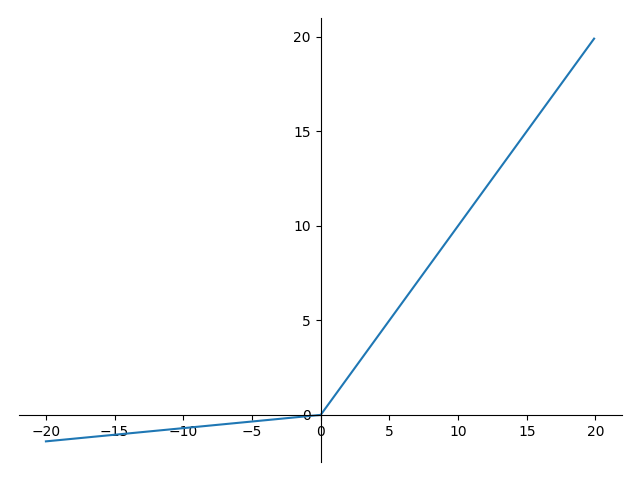
*return 0.01\*x*

*else:*

*return x*

### 5. פונקצית מיישר פרמטרית (Parametric ReLU function)

*פונקצית הפעלה מאוד דומה לReLU הרגילה אך לוקחת את הזליגה צעד אחד קדימה על ידי הגדרת כמות הזליגה בפרמטר שנלמד כמו שאר הפרמטרים של רשת הנוירונים .*[*[12]*](#_ביבליוגרפיה)

**

*def PReLU\_function(x):*

*if x<0:*

*return a\*x*

*else:*

*return x*

***איור 8***

*a\*v*

*v*

**6. פונקצית סופטמקס (Softmax function)**

*פונקצית Softmax מתוארת לעתים קרובות כשילוב של מספר sigmoids מרובים. אנו יודעים כי sigmoid מחזיר ערכים בין 0 ל -1, שניתן להתייחס אליהם כאל הסתברות לנתונים השייכים למחלקה מסוימת. לכן משתמשים בסיגמואיד באופן נרחב בבעיות סיווג בינאריות. ניתן להשתמש בפונקציה softmax לבעיות סיווג רב-כיתתיות. (multiclass classification problems) פונקציה זו מחזירה את ההסתברות לדאטא מסויים השייך לכל מחלקה בנפרד . בעת בניית רשת לבעיה רב-כיתתית, בשכבת הפלט יהיו מספר נוירונים רבים כמספר הכיתות ביעד(target) לדוגמה, אם יש לך שלוש כיתות, יהיו שלושה נוירונים בשכבת הפלט*

*הביטוי*המתמטי של סופטמקס נראה כך:

*def softmax\_function(x):*

*z = np.exp(x)*

*z\_**soft = z/z.sum()*

*return z\_soft*

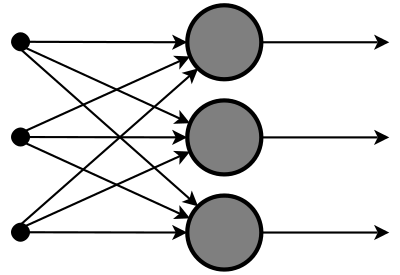
## ארכיטקטורות רשת

### צורת הבנייה והגדרת הקשר בין נוירון לנוירון תלוי במספר גורמים- אחד מהם הוא האלגוריתם הלמידה בו הוחלט להשתמש לאימון הרשת.

באופן כללי, ישנם 3 סוגים בסיסיים של ארכיטקטורות רשת:

### 1. רשתות חד-שכבתיות ללא משוב

רשת בעלת שכבות היא רשת שבה הנוירונים מסודרים בצורת שכבות קשורות. אם אנסה לפשט את התהליך בשביל ההבנה הצורה הכי פשוטה של רשת שכבות מכילה רק שכבה אחת, שכבת-קלט של מקור , שכבה זו מעבירה מידע לפלט של נוירונים , אבל חשוב לשים לב שברשתות מהסוג הזה המידע לא יכול לזרום לצד השני כלומר אין משוב. ובאנגלית נקראת גם רשת מזינה קדימה ( Feed-Forward )[***[5]***](#_ביבליוגרפיה) באיור 7 מוצגת רשת בעלת 3 מקורות ו3 נוירונים רשת כזאת נקראת גם "חד-שכבתית" מכוון שבנויה משכבת חישוב אחת בלבד– שכבת הפלט.



***איור 7***

***input***

***output***

***output***

***output***

***input***

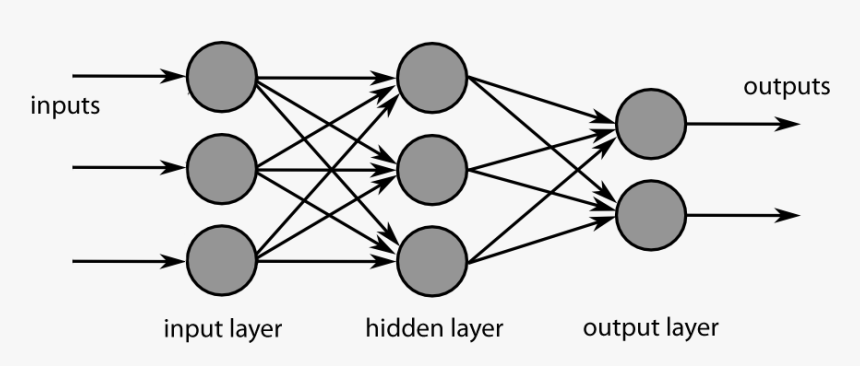
***input***

***output layer***

### 2. רשתות רב-שכבתיות ללא משוב

רשתות מהסוג הזה מכילות שכבות חישובנוספות הנקראות שכבות חבויות או נסתרות ( layers hidden) לכן גם הנוירונים בשכבה הזו נקראים נוירונים חבויים neurons) (hidden . תפקיד השכבות החבויות הוא "להתערב" בין שכבת הקלט ושכבת הפלט. בעזרת הוספת שכבות חבויות הרשת מסוגלת להפיק סטטיסטיקות מסדר גבוה יותר. זו כמובן תכונה חשובה מאוד כששכבת הקלט מכילה הרבה נוירונים. אותות הפלט של הנוירונים בשכבה השנייה הופכים לאותות הפלט של השכבה השלישית, וכך הלאה. בדרך כלל, הנוירונים בכל שכבה מקבלים את אותות הקלט שלהם רק מהשכבה הקודמת להם ולכן זוהי רשת ללא משוב.   
אותות הפלט של השכבה האחרונה ,מהווים את התגובה הכוללת של הרשת לאותות הקלט אשר התקבלו בעזרת השכבה הראשונה. איור מדגים את הפריסה של רשת רב-שכבתית בעלת שכבה חבויה אחת. רשת זו מחוברת בצורה מלאה (fully connected ) כלומר כל נוירון מחובר לכל אחד מהנוירונים בשכבה הבאה. אם לרשת אין את

התכונה הזו, היא נקראת רשת מחוברת-חלקית.



***output***

***input***

***input***

***input***

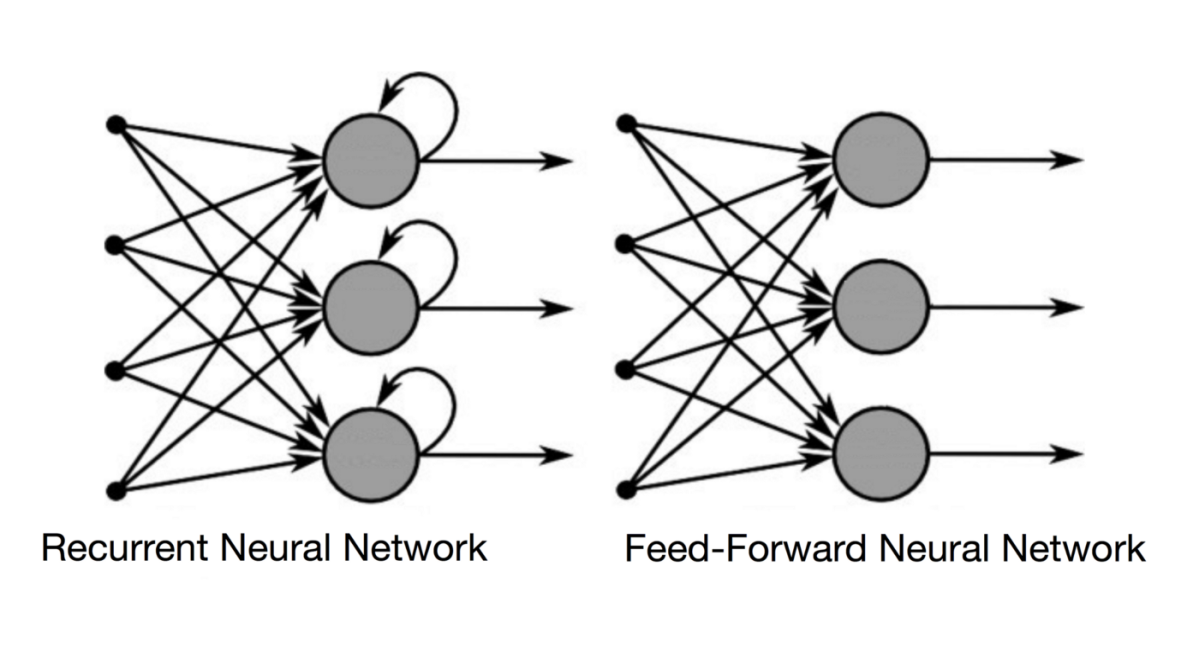
***output***

***input***

***איור 7***

### .3 רשתות חוזרות (Recurrent)

רשות חוזרות שונות במיוחד מהרשתות חסרות המשוב בגלל העובדה שלהן יש לולאות משוב   
( loops .( feedback באיור 7 ניתן לראות רשת חוזרת. ברשת זו יש משוב-עצמי: הפלט של כל נוירון מועבר בתור קלט לעצמו. ניתן גם לתכנן רשת ללא משוב עצמי בה הפלט של כל נוירון מועבר לנוירון אחר באותה שכבה. לולאות משוב משפיעות מאוד על יכולות הלמידה של הרשת ועל ביצועיה. בלולאות משוב ניתן למצוא ענפים (קשרים סינפטיים ) מיוחדים המכילים יחידות מעכבות מידע אשר מטרתן לעכב את האות לפני שהוא מגיע לנוירון הבא בד"כ נסמן אותן כמו ב , יחידות אלה תורמות לאופי האי-ליניארי של רשתות הנוירונים.



***input***

***input***

***input***

***output***

***output***

***output***

***input***

***איור 7***

***output layer***

## סוגי בעיות

רשתות נוירונים יכולות לפתור בקלות יחסית מספר רב של בעיות אנו נסווג את סוגי הבעיות ל3 סוגים עיקריים:

1. בעיות התאמה

בעיות שבהן יש להחליט, בהינתן הקלט, מהו הערך המספרי אשר הכי "מתאים" לקלט המשמעות של ערך זה כמובן תלויה בבעיה המדוברת.) לדוגמה, בעיית מחירי הבתים: נתונים מאפיינים של בית (שטח, מספר חדרים, גודל חצר וכדומה) והפלט הוא המחיר שבו יימכר הבית הזה. במקרה זה 𝑅 ∈ .𝑦 (נזכיר כי y הוא הפלט של הרשת.

### 2. בעיות סיווג

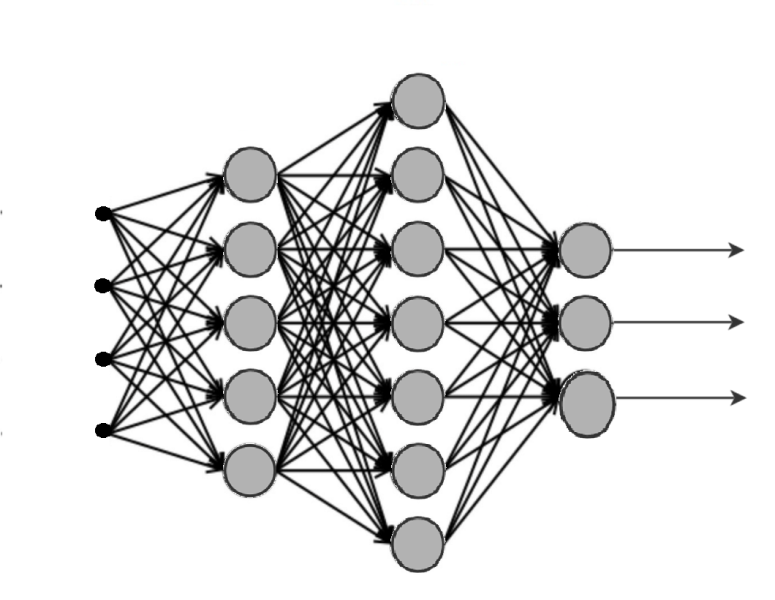
בעיות שבהן יש להחליט האם הקלט נופל תחת קטגוריה (מחלקה) מסוימת או לא.

לדוגמה, בעיית זיהוי תמונה: נתונה תמונה ויש לקבוע האם התמונה הינה תמונה של מכונית (לצורך

העניין) או לא. במקרה זה {0,1} ∈ .

### 3. בעיות סיווג מרובה-מחלקות:

בעיות שבהן יש להחליט לאיזו קטגוריה (מחלקה( מבין הקיימות, שייךהקלט. לדוגמה, גרסה קצת יותר מסובכת של בעיית זיהוי תמונה: נתונה תמונה ויש לקבוע האם התמונה הינה תמונה של מכונית, של הולך-רגל, של אופנוע או של משאית. להלן רשת פוטנציאלית לפתרון

הבעיה הזו:

***איור 7***

***output layer***

***input***

***output***

***output***

***input***

***input***

***input***

***output***

***Hidden layer***

***Hidden layer***

***הולך-רגל***

*מכונית*

*מעבר חצייה*

ברשת זו ישנם 2 נוירוני-פלט אשר פלטם הוא 1 או 0 (שכן זוהי בעיית סיווג.) ליד כל נוירון פלט מצוינת

המחלקה אותה "מייצג" הנוירון. במקרה זה .

סט האימון יורכב מזוגות y) (x, כאשר x הוא תמונה (מטריצת פיקסלים) וy הוא וקטור בינארי בגודל ,3

כאשר:

*מייצג*

*מכונית*

*מייצג*

*מעבר חצייה*

*מייצג*

***הולך-רגל***

ללא קשר לסוג הבעיה, תהליך הפתרון שאמצעות רשת נוירונים הוא זהה: יש למצוא כמות (גדולה כמה שיותר) של זוגות (פלט, קלט) אשר ישמשו כסט אימון. ככל שיהיו יותר זוגות בסט האימון כך הרשת תוכל לפתור את הבעיה בצורה טובה יותר. לאחר מכן יש להעריך את רמת הדיוק (ולתקן במידת הנדרש) של הרשת האמצעות זוגות (פלט, קלט) אשר יהוו את סט הבדיקה. לאחר מכן ניתן להשתמש

ברשת על הבעיה האמתית שברצוננו לפתור.

למידה

כוחן הגדול של רשתות נוירונים מקורו ביכולת הלמידה וההסתגלות שלהן לסביבה. בפרק זה נבחן את

הדרך בה רשתות נוירונים יכולות לעשות זו.

הפלט של רשת הנוירון תלוי (מלבד בקלט כמובן) במשקלים הסינפטיים שלה (המטריצות .)𝛩(𝑗) הרעיון הכללי הוא להזין לרשת כמה שיותר דוגמאות קונקרטיות (דוגמאות אלה מרכיבות את סט האימון) כאשר בכל פעם שנותנים לרשת "ללמוד" דוגמה, המשקלים הסינפטיים משתנים על מנת שהפלט של הרשת יתאים כמה שיותר לכל הדוגמאות שנלמדו עד עכשיו. הדרך לעשות זו הינה להגדיר "פונקציית עלות"

אשר נמצאת ביחס ישר עם השגיאה של הרשת (קרי, כמה הרשת עובדת "רע)" ולמצוא את המינימום

שלה (וע"י כך לגרום לרשת לעבוד "טוב" במידת האפשר.)

פונקציית העלות

(2.1) 𝑒𝑗(𝑛) = 𝑑𝑗(𝑛) − 𝑦𝑗(𝑛)

אות השגיאה של הפלט של נוירון j באיטרציה n מוגדר ע"י הנוסחה:

אנו מגדירים את ערך אנרגיית השגיאה של נוירון j ע"י הביטוי 𝑒2(𝑛) .1 בהתאם לכך, הערך 𝐽(𝑛) (סכום

2 𝑗

אנרגיית השגיאה באיטרציה )n מחושב ע"י סכימת 𝑒2(𝑛) 1 על פני כל נוירוני הפלט. אלה הם הנוירונים

2 𝑗

( ) ( ) 1 2

ה"נראים" (לא חבויים) ולכן ניתן לחשב את גודל שגיאתם בצורה ישירה:

2.2

𝐽 𝑛

= 2 ∑ 𝑒𝑗 (𝑛)

𝑗 ∈𝐶

כאשר C מסמן את קבוצת כל הנוירונים בשכבת הפלט. נסמן בN את מספר כל הדוגמאות בסט האימון

(גודל סט האימון.) אנרגיית ריבועי השגיאות הממוצעת מחושבת ע"י סכימת כל 𝐽(𝑛) ונרמול ביחס ל:N

𝑁

(2.3) 𝐽𝑎𝑣

= 1 ∑ 𝐽(𝑛)

𝑁

𝑛=1

הערך ,𝐽(𝑛) וכך גם הערך ,𝐽𝑎𝑣 הוא פונקציה של הפרמטרים החופשיים (המשקלים הסינפטיים ומשקלי

ה)bias של הרשת. עבור סט אימון מסוים, 𝐽𝑎𝑣 מייצג את פונקציית העלות כאמת מידה לביצועי הרשת. המטרה של תהליך הלמידה היא לכוונן את הפרמטרים החופשיים על מנת להגיע למינימום של הערך

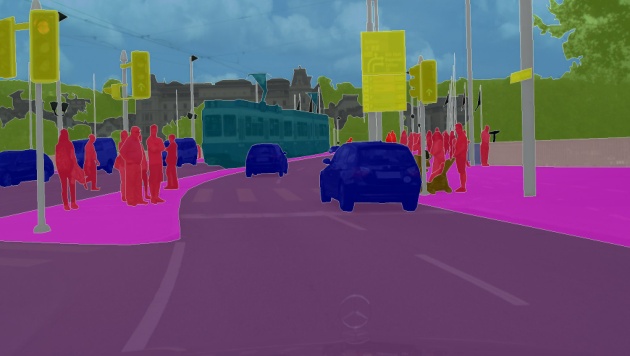
.𝐽𝑎𝑣

## סגמנטציה

בעולם הראייה הממוחשבת לאחרונה ישנה התקדמות מהירה בנושא הסגמנטציה, כאשר המשימה היא להקצות כל פיקסל תיוג מסויים וכך לקבל הבנה של מה נמצא בתמונה. עד כה נעזרו לרוב בשיטות בסיסיות פשוטות אך חזקות, כולל רשתות Fully Convolutional Networks (FCN)

וMASK\_R-CNN

שיטות אלה פשוטות באופן רעיוני, מהירות וגמישות. בעבודה אציג ארכיטקטורה משלי למשימת זיהוי אובייקטים בעזרת רשת ניורנים מבוססת CNN המשתמשת כdatabase במאגר תמונות COCO בתצורת זיהוי

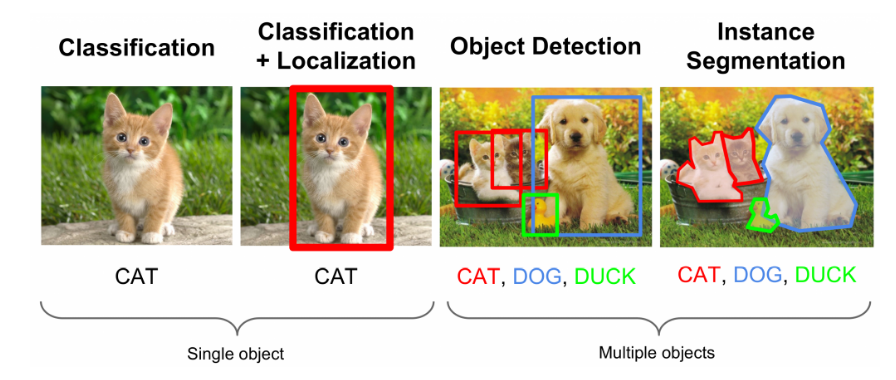
**Segmentation** או בעברית פילוח פונה לזיהוי קלאסים שונים של אובייקטים ומאחד את המשימות הפילוח המופיעות בדרך כלל. המטרה היא לייצר קיטועים או סגמנטציה מגובשת ועשירה, שימוש מרכזי של המערכות הוא בעיקר בדוגמאות כמו בנהיגה אוטונומית או augmented reality

בעזרת מערכת לומדת ניתן לזהות אובייקטים בתמונות במידה ויש מאגר תמונות רבות שמסווגות בצורה טובה.

מערכת לומדת זאת אפשרות מאוד טובה לזיהוי אובייקטים ואנחנו יכולים לראות מהתעשייה שיש הרבה יישומים שמשתמשים במערכות לומדות לזיהוי אובייקטים

תחילה אסביר את ההבדל בין קלאסיפיקציה, זיהוי אובייקט, ופילוח סמנטי:

***ד' ג' ב' א'***



בתמונה **א**' ניתן לראות שישנו אובקייט אחד והוא זוהה כחתול

בתמונה **ב**' ניתן לראות שישנו אובקייט שזוהה כחתול וסביבו ממורכז bounding-box שנותן לוקאליזציה של האובייקט במרחב התמונה.

בתמונה **ג**' ניתן לראות שישנם מספר אובקייטים מקאלסים שונים ולכולם boxes ותיוגים

בתמונה **ד'** ניתן לראות שישנם מספר אובקייטים מקאלסים שונים כשלכל אובייקט סגמנטציה הדוקה, צבע אחר המתאר קלאס שונה.

בפרוייקט זה בחרתי את מאגר הנתונים **COCO** המכיל תמונות RGB בגדלים שונים

המאגר מכיל **118**,**287** תמונות לאימון, **5000** תמונות לוולידציה ו-**40**,**670** תמונות לבחינה.

את המאגר ניתן להוריד מהאתר של קהילת COCO : <http://cocodataset.org/#download>

במאגר זה משתמשים בקבצי json לתאור האנוטציות בתמונות.



# **2. ארכיטקטורה ואימון:**

לאחר ייבוא כל הספריות הנדרשות , טענתי את מאגר התמונות למחשב,

השתמשתי בקובץ Instances כדי לקבל את המסכות של האובייקט שאני מנסה לזהות בתמונות.

לאחר מכן יצרתי מסיכות בינאריות לכל התמונות

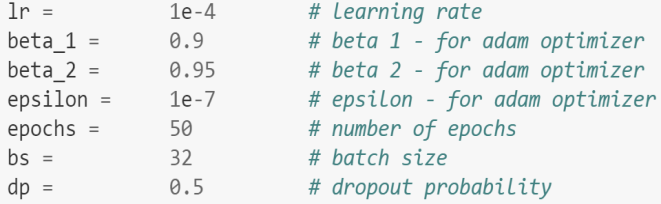
במידה וקיימת אנוטציה של האובייקט שמעניין אותי בתמונה אז יצרתי מסיכה עם ערך המאפיין את האובייקט

לדוגמה אם אני מנסה לזהות אנשים וחתולים אז לכל אובייקט יש מספר סידורי שמתאר את הקלאס , למשל הערך שמאפיין אנשים הוא 1 והערך שמאפיין חתולים הוא 17

אז יצרתי מסכה שבכל מקום שיש בן אדם שם יהיה ערך 1 ובמקום שיש חתול יהיה ערך 2 כך אני יכול לסווג את המידע שלי בצורה נוחה למודל.

בגלל מגבלות של זכרון נאלצתי לשנות את הרזולוציה של התמונות ל64x64 כך יכולתי לאמן את המודל על כל המאגר

הגדרתי את hyper-parameters שלאחר הרבה איטרציות של ניסוי וטעייה מצאתי שהכי מתאימים למודל ולארכיטקטורה שבניתי:



מאגר התמונות COCO שאיתו אני עובד מחולק לתיקיות מסודרות:

* החלק המרכזי שעליו אימנתי את המודל (train)
* החלק שעליו אעשה test וכך אוכל לבחון את הביצועים של המודל עבור מידע שעדיין הוא לא ראה ואומן עליו.

## אימון הרשת – זמן כולל ופלטופרמת המערכת.

***Train time***

2ms per sample

173s per Epoch   
2.5 hours of training (total of 50 epochs)

***PC platform:***

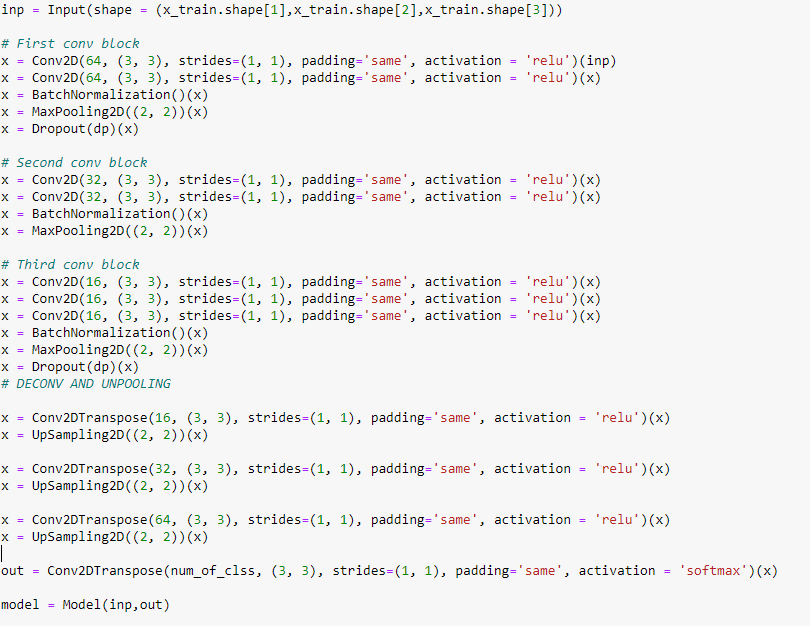
CPU: Intel i7 9700k

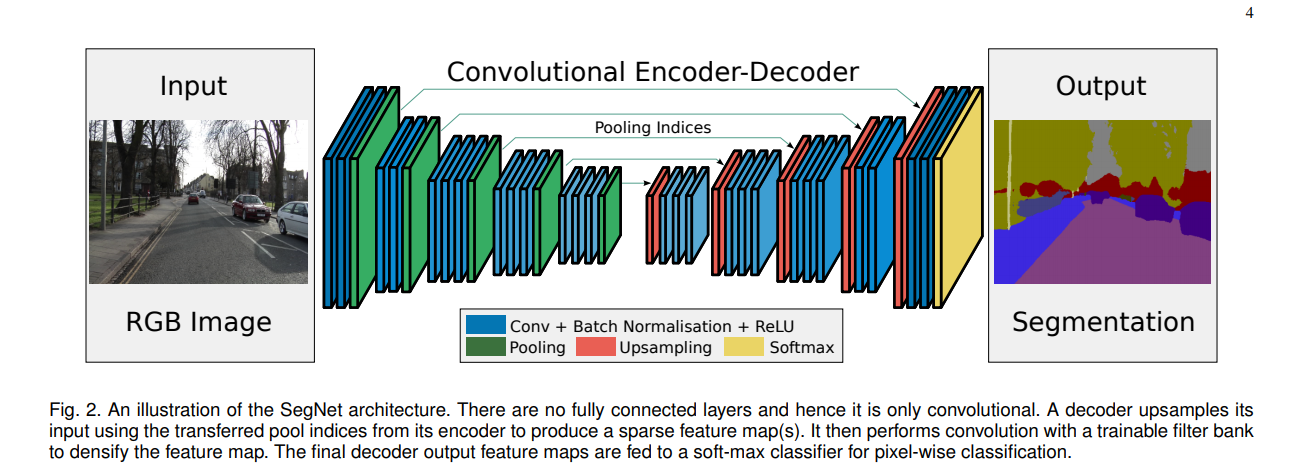
GPU: nvidia geforce 1050ti

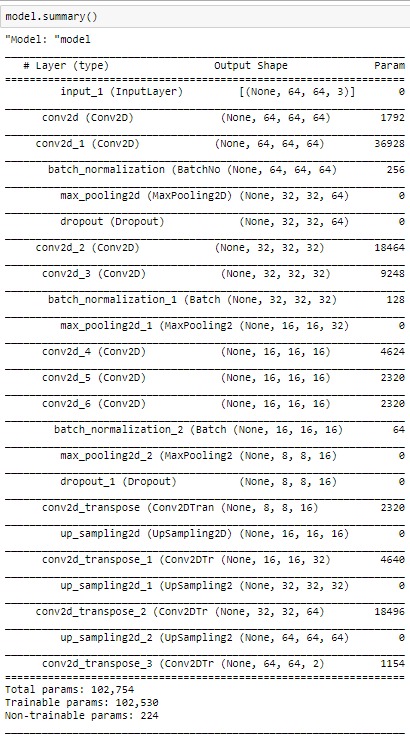
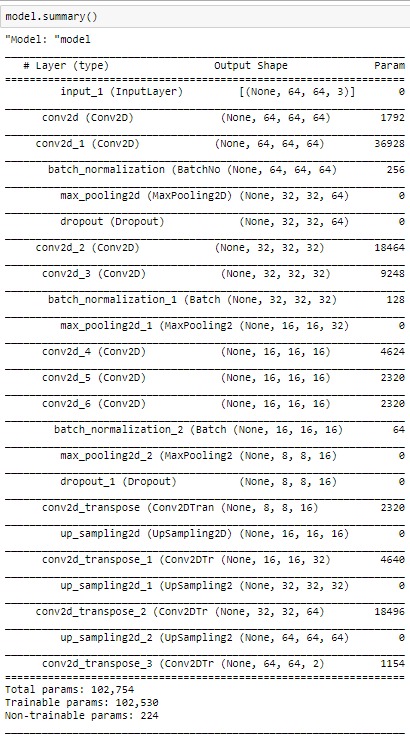
RAM: 32GB DDR4

Disk: M2 SSD kingston

בניתי מודל בעל השכבות הבאות:







### כמות פרמטרים הכוללת ברשת

### ארכיטקטורת הרשת:

שרשור של שכבות קונבולוציה עם ReLU ובסוף פולי-קונקטד מאפשר טיפול במקרים אי-לינארים ומשמר את הdimensions וכך משפר את הrobustness של הרשת.

ניתן לראות שברשת שלי בחרתי בארכיטקטורת encoder-decoder או בעברית מפענח-מקודד   
ארכיטקטורה זו היא תבנית ידועה של רשתות שנוח להשתמש בה בעיקר לsegmentation.

הארכיטקטורה מחולקת לשני חלקים, המקודד והמפענח. תפקיד המקודד הוא לקודד את הכניסות למצב.

ואז המצב מועבר למפענח כדי לייצר את היציאות.

המפענח הוא בלוק ברשת (בדרך כלל אותה מבנה רשת כמו המקודד) שלוקח את וקטור הפיצ'רים מהמקודד ומעניק את ההתאמה הטובה ביותר לקלט או לפלט המיועד.

לאחר שאימנתי את המודל נותר כעת לבחון אותו על מידע חדש שהוא לא ראה מימיו כדי לראות את היעילות שלו בפועל הלכה למעשה.

# מניעת Over Fitting

תחילה אסביר את המושג אובר-פיטינג:

בסטטיסטיקה, התאמת יתר היא תוצר של ניתוח שנותן תוצאות קרוב מדי או מדויק עם מערך נתונים מסוים, ולכן עשוי שלא להתאים לנתונים נוספים או לחזות תצפיות עתידיות באופן אמין". מודל עם התאמת-יתר הוא מודל סטטיסטי המכיל יותר פרמטרים שיכולים להיות מוצדקים על ידי הנתונים.

בתור דוגמה קיצונית להתאמת יתר, אם מספר הפרמטרים המשמשים להגדרת המודל זהה או גדול יותר מאשר מספר התצפיות, אזי המודל יוכל לחזות באופן מדויק את התצפיות פשוט על ידי זכירתן במלואן.

התאמת יתר מתרחשת במקרים בהם חסרה תאוריה להנחיית האנליזה של הנתונים. במצב כזה יש צורך לבחון מספר רב של מודלים אפשריים ונדרשים יותר פרמטרים כדי להגדירם[13].

ישנן כמה שיטות מקובלות להפחתת הסיכון בהתאמת יתר, בכולם השתמשתי בפרוייקט שלי: רגולריזציה (regularization), עצירה מוקדמת (early stopping) , עיוות הדאטה (Data augmentation וכן drop-out

הבסיס לשיטות אלה הוא:

* מחיר מפורש למורכבות המודל (למשל ברגולריזציה) ובכך לתת לתהליך האופטימיזציה להעדיף פתרונות פשוטים. מורכבות המודל אינה נבחנת רק לפי מספר הפרמטרים אלא גם לפי מידת הביטוי של הפרמטרים.
* עיוות המידע והזנתו למאגר כמידע נוסף , ושימוש במספר שינויים קטנים ברשת כדי ליצור אנסמבל של רשתות אקראיות וזאת כדי למנוע תאימות יתר

## 1. אתחול משקולות

Glorot uniform initializer, also called Xavier uniform initializer

<http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>

כשעובדים עם רשתות עצביות עמוקות, אתחול הרשת עם המשקולות הנכונות יכול להיות קשה מכיוון שרשתות עצביות עמוקות סובלות מבעיות שנקראות Vanishing/Exploding Gradients.

לכן אנחנו צריכים שהאות יזרום כראוי לשני הכיוונים: בכיוון קדימה בעת ביצוע תחזיות, ובכיוון ההפוך עבור BackPropagating gradients

אנחנו לא רוצים שהאות ימות, ואנחנו גם לא רוצים שהוא יתפוצץ או יגיע לסאטורציה. לכן אנחנו צריכים שהמדרגות יהיו שונות שווה לפני ואחרי שזורמים בשכבה בכיוון ההפוך.

אתחולים מגדירים את הדרך להגדרת המשקולות האקראיות הראשוניות של שכבות keras.

ברשת שלנו אנו השתמשנו באיתחול משקולות מסוג XAVIER ידוע גם בשם GLOROT

## Batch normalization.2

טכניקה שמאפשרת לנו לזרז את ולייצב את תהליך הלמידה זה מאפשר לנו להשתמש במדד למידה גבוה יותר כך גם הלמידה מהירה יותר, אחרי פונקצית אקטיבציה אנחנו מנרמלים את המידע לשיכבה הבאה

לדוגמה כאשר יש לנו שיכבה חבוייה שמורכבת מ

*נחשב את המיצוע והשונות*

*כאשר נוסחאת הנרמול נראית כך:*

*מטרת האפסילון היא למקרה שסיגמה מתאפסת ואחרי הנרמול עכשיו לכל יחדה של השכבה יש עכשיו מיצוע סביב האפס ושונות השווה לאחד ובגלל שאנחנו נעדיף שיהיה לכל נוירון ערכים שונים אז אפשר לכתוב את זה בצורה כזאת כך שהמידע יהיה מנורמל בצורה אחרת*

*במידה ו*

*כך שאפשר לשחק עם הפרמטרים של גמא ובטא ואז המיצוע והשונות יכולים להיות מנורמלים סביב איזה אזור שנרצה*

*במודל שלנו כאשר לא ספקנו פרמטרים אז הוא נרמל את המיצוע קרוב לאפס ואת השונות קרוב ל1*

שיטות נוספת שבחרנו להשתמש בה למניעתoverfitting היא

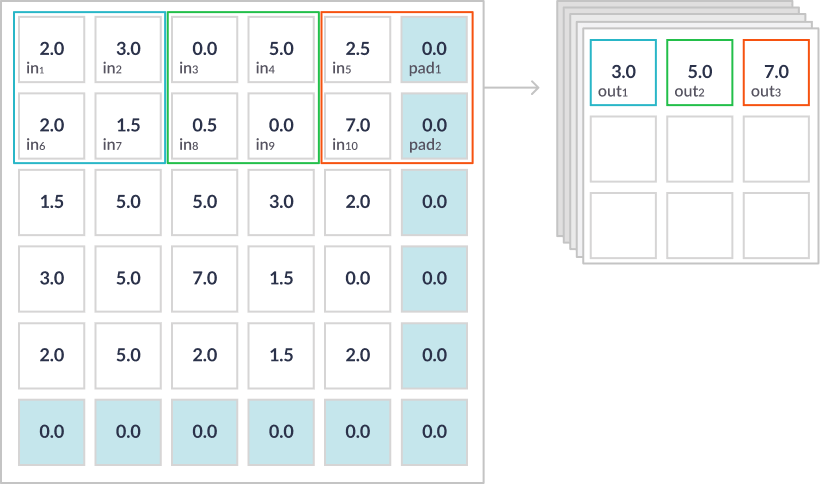
## Dropout.3

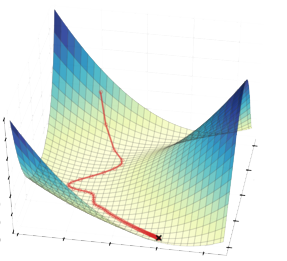
טכניקת רגולריזציה שמאפשרת לנו להתגבר על הנושא של תאימות יתר (over fitting) של המודל שלנו

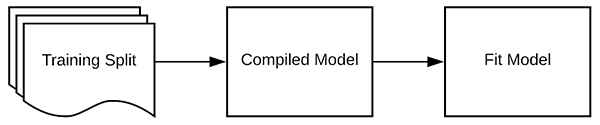
שבזמן הלימוד בהסתברות מסויימת מתעלמים מחלק הנויירונים בשיכבה, בצורה זו אנו מכניסים אקראיות למודל - שכן כל ריצה רצף ניורונים אחרים "מתים" וכך המודל חייב להתאים את עצמו לניורונים אחרים.

## MaxPooling2d.4

מטרת שכבת הפולינג היא להוריד את גודל הכניסה שכבה זו מרדדת את הרעש ומרדדת שינויים קלים כמו סיבוב או הזזה של התמונה גם שכבה זו מבוססת על פילטור של חלון רץ השכבה הזו לוקחת את התוצאה הגבוהה ביותר בכל חלון דוגמה בתמונה: (כפי שהשתמשנו במודל שלנו חלון 2X2) בכל חלון ערך המקסימום נלקח למפת הפיצ'רים שלנו במילים אחרות כל ערך מקסימום הוא מייצג את החלון של 2X2 במפת הפיצ'רים שלנו שכבה זו בתמונות קטנות יכולה לעזור בפתרון העובר פיתינג







השתמשתי לאחר מכן ב adam optimizer

**Conv2d**

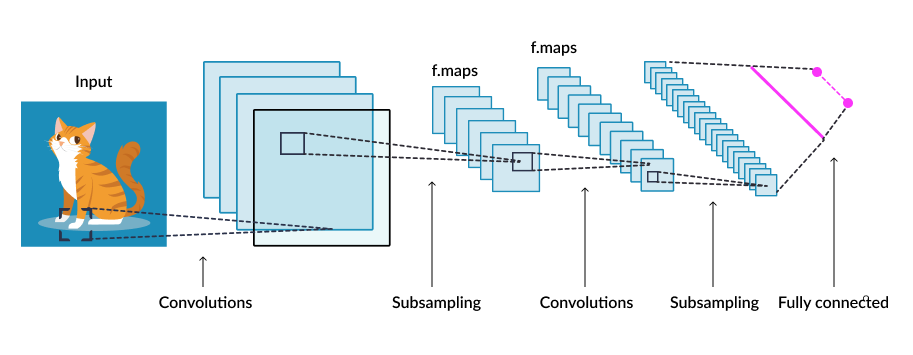
*שכבת קונבולוציה דו ממדית מקבלת בכניסה מערך תלת מימדי בדרך כלל תמונה עם שלושה ערוצי צבע,*

*השכבה רצה עם חלון (קרנל) קורולציה קטן (בדרך כלל 3x3) על כל התמונה כך למעשה הוא מוצא פיצ'רים (אזורי עניין) בתמונה החלון מאותחל עם משקלים שמתעדכנים בכל איטרציה בזמן הלימוד עד שמגיעים לתוצאות אופטימאליות למודל.*

*החלון נקרא 2D בגלל שתזוזת החלון היא למעשה על התמונה שהיא דו ממדית שלכל פיקסל יש שלושה ערכי צבע אדום ירוק וכחול*

*הפילטר רץ על התמונה שלוש פעמים על כל ערוץ צבע.*

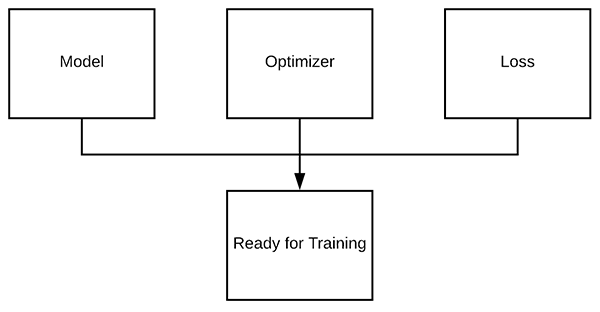
*אפשר להתייחס לזה בצורה כזאת ששכבת הקונבולוציה הראשונה תופסת פיצ'רים בתמונה המקורית אם נקח לדוגמה חתולים היא מזהה פיצ'רים כגון עיניים , אוזניים, ראש, זנב וכדומה לאחר מכאן היא מזהה פיצ'רים באזורי העניין של הראש של האוזניים, של העיניים וכדומה בסופו של דבר המטרה היא לנסות לזהות את כל הפיצ'רים שאיתם אנחנו יכולים לסווג את האובייקט שלנו ולשייך אותו לקבוצה מסוימת*



# Back-propagation

הרעיון הוא לעדכן את המשקלים, דוגמה אחר דוגמה, עד סוף תקופה מסוימת (EPOCH)

תקופה זו מייצגת את מספר המעברים הנדרש להגיע להצגה מלאה של כל סט האימון לרשת הנוירונים. עדכון המשקלים נעשה ביחס לשגיאות המחושבות עבור כל אחת מהדוגמאות בסט האימון. הממוצע של שינויי המשקל תוך כדי מעבר על סט האימון הוא הערכה של השינוי האמתי שיקרה כתוצאה משינוי משקלים



לאחר מכן הוצאנו גרף למידה כדי לוודא התכנסות,

והצגנו את התוצאות בעזרת מטריצת מבוכה

לבסוף הוצאנו עקומת ROC

עקומת ROC, היא גרף המציג את הביצועים של מסווג, לאור סף ההחלטה שנקבע לו. העקומה נוצרת על ידי התוויית שיעור החיוביים האמיתיים (TPR) מול שיעור החיוביים הכוזבים (FPR) תחת ספי קבלה שונים. שיעור החיוביים האמיתיים ידוע גם ככיסוי בלמידת מכונה





אחת הדרכים למדוד את טיב הבחירה היא להשתמש במדד F-score , מדד סטטיסטי לבדיקת דיוק שמתחשב

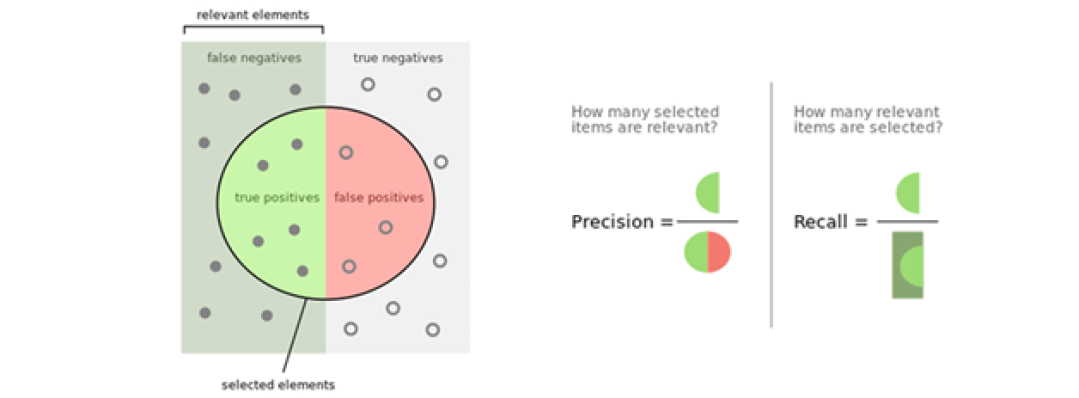
ב- Recall = TP/(TP+FN)

 וב- Precision = TP/(TP+FP)

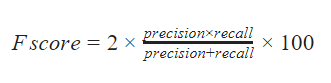
כאשר TP=True Positive זה כשהאלגוריתם חזה שהקטע מעניין וצדק

ו FN=False Negative זה כשהאלגוריתם חזה שהקטע לא מעניין וטעה

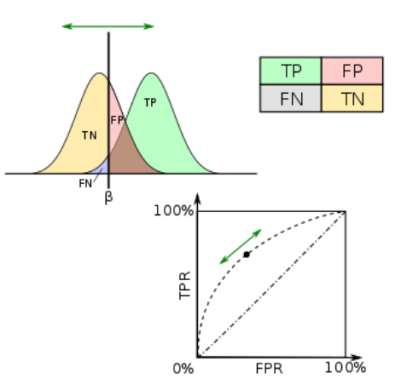
ו FP=False Positive זה כשהאלגוריתם חזה שהקטע מעניין וטעה.

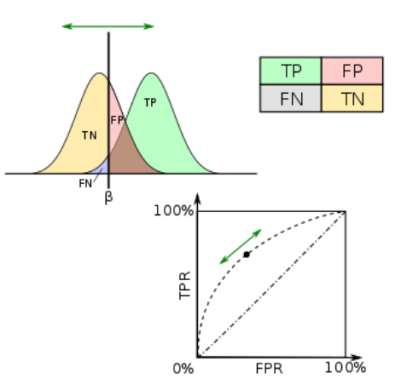


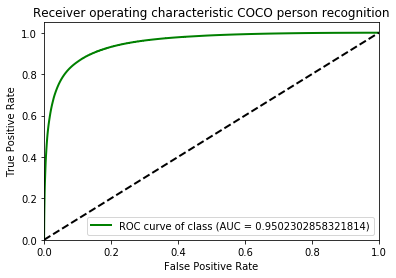
 מדד  F-score  מוגדר באופן הבא:



ניתן לראות שהגענו למודל טוב שמצליח לזהות את הקטגוריות השונות בצורה דיי טובה, לפי מטריצת המבוכה ולפי גרף הROC ,



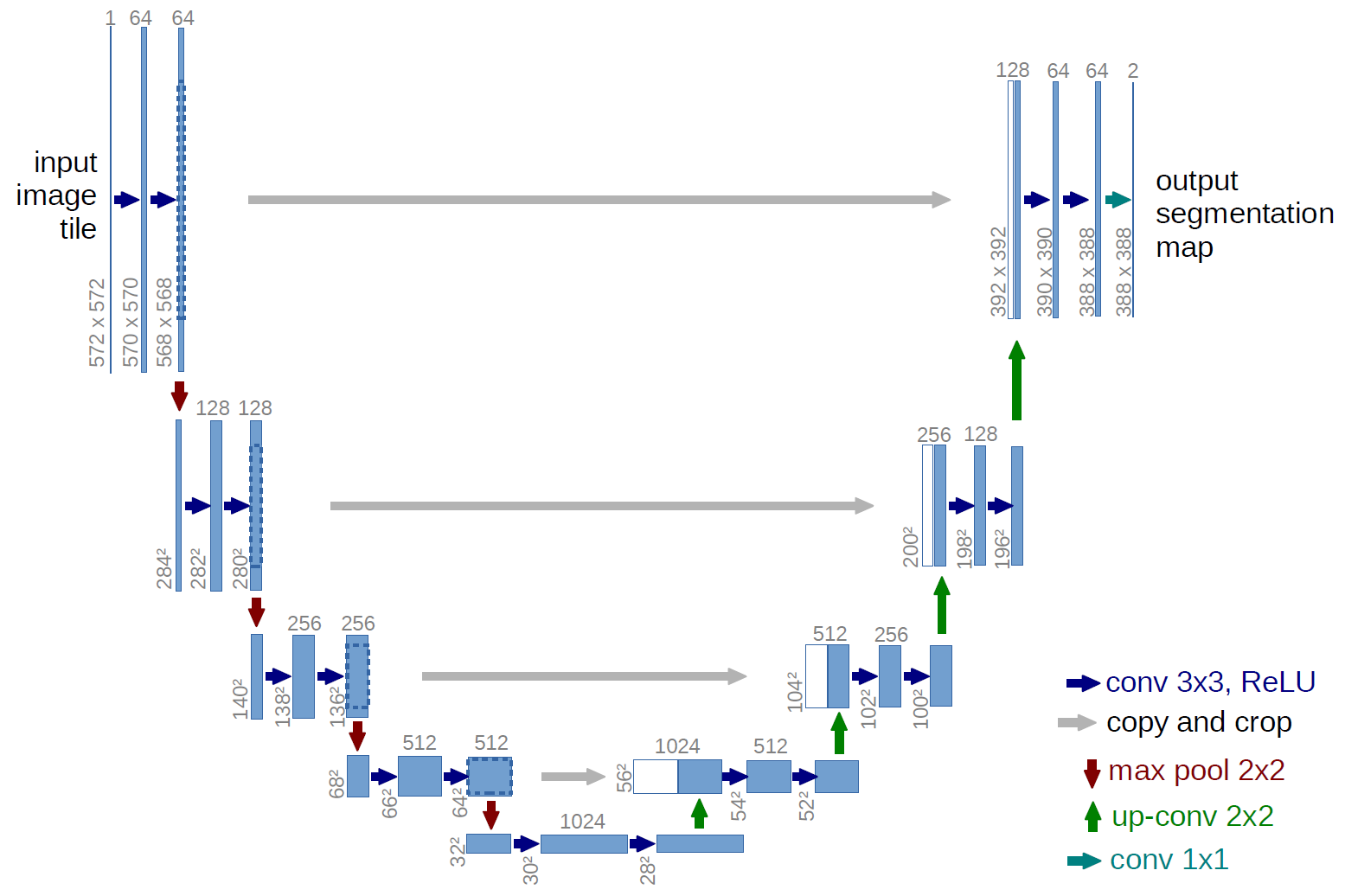




# תאור הארכיטקטורה של הרשת:

הרעיון המרכזי בשימוש בשכבות קונבולוציה זה לזהות פיצרים בתמונה ולמפות אותם וזה עובד ממש טוב בפתרון בעיות קטלוג לקטגוריות,למעשה מתמונה מייצרים ווקטור שבוא משתמשים לקטלוג אבל כאשר מנסים לייצר מערכת שממפה תמונה למסכה של אובייקט בתמונה אנחנו לא רק צריכים לייצר ווקטור אנחנו גם צריכים לייצר תמונה שזאת משימה מורכבת מאוד, מהרעיון הזה למעשה נוצר הרעיון להשתמש במודל Unet לאחר שלקחנו תמונה ויצרנו ממנה ווקטור, מיפינו את הפיצרים בתמונה למעשה זה המתכון למערכת מפה אנחנו יכולים לעשות את התהליך בחזרה ולייצר תמונה

דוגמה לארכיטקטורת Unet

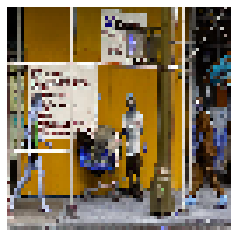


הארכיטקטורה נראת בצורת האות U מפה השם

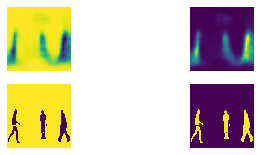
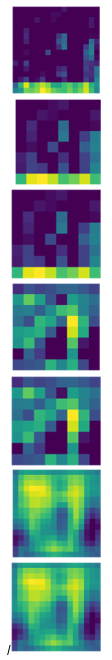
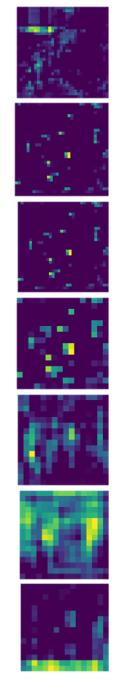
לשם הבנה יותר טובה של **מה הרשת למדה** ניתן להתבונן במשקולות בשכבה הראשונה, להכניס

תמונות בוחן ולראות מה קורה ברשת

הכנסנו את התמונה הזאת:

******

וכאן אנחנו רואים וויזואליזציה של איך הרשת שלנו טיפלה בתמונה כל שכבה בתמונה:



# 3. סיכום

כמו שציינו בהתחלה, המטרה שלנו הייתה ליצור רשת CNN שמסווגת בצורה מיטבית ואכן הצלחנו ולמדנו מכך המון.

בעיקר למדנו ש deployment של מודל הוא הרבה יותר פשוט ממה שחשבנו בהתחלה, אך אימון מודל הוא עדיין תהליך מאתגר – שדורש תכנון, הבנה, תיעוד ותחקור משמעותי.

**ביצועי הרשת** שלנו דיי טובים יחסית למידע שסיפקנו למודל למרות חוסר זמן וכוח מחשוב חזק מספיק הרשת הצליחה במשימה בסופו של דבר.

**האתגרים** שהיו לנו הם בעיקר כח עיבוד וחוסר זמן לכן הורדנו את הרזולוציה של התמונות לרזולוציה נמוכה ((64\*64 וחיפשנו מודל שיעבד את המידע בצורה המהירה ביותר כך שבתחילת הדרך המודל "למד" במשך יממה וקצרנו את זה לשעתיים וחצי אימון והתוצאות כמעט ולא נפגעו.

אתגר נוסף היה סידור המידע שהיינו צריכים ללמוד איך עובדים עם מאגר המידע COCO ולטעון את המידע בצורה מסודרת לזכרון כך שיהיה מסודר בצורה הגיונית (מבחינת פורמט, רזולוציה, כמות ערוצים וכו) גם בכניסה למודל.

# דרכים לשיפור הרשת:

את הרשת ניתן לשפר המון וזה דורש הרבה זמן מחקר:

אפשר לנסות כלי אופטימיזציה שונים אנחנו השתמשנו ב"אדם" אבל לפי מה שקראנו באינטרנט יש עוד אלגוריתמים שניתן להשתמש בהם,

כמו כן הרשת שלנו דיי קטנה מבחינת כמות שכבות - אפשר להוסיף עוד שכבות וכמובן להגדיל את רזולוציית התמונות.

אפשר להתגבר על over Fitting בעזרת data Augmentation או early Stopping שתי שיטות שלמדנו בהרצאות אך בחרנו שלא השתמשנו בהם, כמו כן, ראינו שיש צורך באיזון בין ניסוי כיוונים חדשים לבין מיצוי המיטב מגישות קיימות

# **ביבליוגרפיה**

1. Mitchell, T. (1997). *Machine Learning*, McGraw Hill. [ISBN](https://he.wikipedia.org/wiki/%D7%9E%D7%A1%D7%AA%22%D7%91) [0-07-042807-7](https://he.wikipedia.org/wiki/%D7%9E%D7%99%D7%95%D7%97%D7%93:%D7%9E%D7%A9%D7%90%D7%91%D7%99_%D7%A1%D7%A4%D7%A8%D7%95%D7%AA/0-07-042807-7), p.2.
2. [*"About"*](https://www.kaggle.com/docs/competitions)*. Kaggle. Kaggle Inc.*
3. [*"Google achieves AI 'breakthrough' by beating Go champion"*](https://www.bbc.com/news/technology-35420579)*. BBC News. BBC. 27 January 2016.*
4. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* 521, 436 (2015).
5. ["Deep learning in neural networks: An overview"](https://www.sciencedirect.com/science/article/pii/S0893608014002135). *Neural Networks* **61**: 85–117. 1 בינואר 2015. [ISSN](https://he.wikipedia.org/wiki/International_Standard_Serial_Number) [0893-6080](https://www.worldcat.org/issn/0893-6080). [arXiv](https://he.wikipedia.org/wiki/ArXiv):[1404.7828](https://arxiv.org/abs/1404.7828). [doi](https://he.wikipedia.org/wiki/Digital_object_identifier):[10.1016/j.neunet.2014.09.003](https://dx.doi.org/10.1016%2Fj.neunet.2014.09.003)
6. Le Cun Y., Boser B., Denker J. S., Henderson D., Howard R.E., Hubbard W., and Jackel L.D. (1990). Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in neural information processing systems,* volume II, pages 396-404. Morgan Kaufmann Publishers Inc. San Francisco, CA.
7. Pomerleau, D.A. (1990) Rapidly adapting artificial neural networks for autonomous navigation. In *Advances in Neural Information Processing Systems 3*
8. Hadsell R., Sermanet P., Scoffier M., Erkan A., Kavackuoglu K., Muller U., and LeCun Y. (2009) Learning Long-Range Vision for Autonomous Off-Road Driving. *Journal of Field Robotics, 26(2)* pages 120-144.
9. Krizhevsky, A., Sutskever, I., and Hinton, G. (2012) ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*
10. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11)* pages 2278– 2324.
11. [Haykin](http://www.goodreads.com/author/show/83659.Simon_Haykin) S. (1998) Neural Networks: A Comprehensive Foundation. Prentice Hall New Jersey, NY.
12. Pomerleau, D.A. (1993) Knowledge-based training of artificial neural networks for autonomous robot driving. In *J.H. Connell & S. Mahadevan (Eds.), Robot learning* pages 19–43. Kluwer Academic New York, NY.
13. Russell, S.J., and Norvig P. (1995) Artificial Intelligence: A Modern Approach. Prentice Hall New Jersey, NY.
14. Ng, A. (2011) Machine Learning. <https://www.coursera.org/course/ml>
15. Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng (2014). Rectifier Nonlinearities Improve Neural Network Acoustic Models
16. He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Image *Net* Classification". [arXiv](https://en.wikipedia.org/wiki/ArXiv_(identifier)):[1502.01852](https://arxiv.org/abs/1502.01852) [[cs.CV](https://arxiv.org/archive/cs.CV)].
17. Claeskens, G.; Hjort, N.L. (2008), *Model Selection and Model Averaging*, [Cambridge University Press](https://he.wikipedia.org/wiki/Cambridge_University_Press).

