Head CT hemorrhage detection — זיהוי דימום בסריקות CT ראש

Machine Learning - Final Project — **למידת מכונה- פרויקט גמר**

By:
Roi Abramovitch
Chen Asaraf

מגישים:
רועי אברמוביץ'
חן אסרף

הקדמה: בפרויקט זה השתמשנו במאגר תמונות CT ראש במטרה לזהות דימום ראש על כל סוגיו.
עשינו שימוש במספר כלים מעולם למידת המכונה ולמידה עמוקה כדי לסווג את התמונות ובמטרה למצוא את הכלי היעיל ביותר מבחינת אחוזי דיוק וזמני חישוב. בין הכלים בהם השתמשנו היו ה K-nearest-neighbors , ה Support Vector Machine וה Convolutional Neural Network שהתגלו כמודלים המתאימים ביותר לבעיה שלנו.
את הקוד ניתן לראות בקישור הבא: Our Github Repository

תיאור הפרויקט:

מאגר המידע: את המאגר לקחנו מאתר Kaggle והוא הכיל 200 תמונות של סריקות CT ראש, כאשר 100 תמונות CT היו עם דימום בראש ו 100 תמונות היו בלי דימום בראש (ללא הבחנה בין סוגי הדימומים).
כמו כן המאגר הכיל קובץ CSV שבו לכל תמונה במאגר נתן תיוג של 0 או 1 כאשר 1 אומר שיש דימום ו 0 אין.

שלב ראשון- חילוץ וקטורי פיצ'רים מהתמונות במאגר: לפני שיכולנו להשתמש במודלים ללמידת מכונה על המאגר, היה עלינו להבין איך לייצג את המידע בצורה שתאפשר עיבוד של מודלים מתמטיים. שלב זה משמעותי ביותר לתוצאות המודלים שכן המידע שיישאר על כל תמונה זה המידע שהמודלים יקבלו ולפיהם הם יסווגו את התמונות.
לכן השתמשנו בספריית OpenCV המאפשרת ייצוג של תמונה כמטריצה של גווני הפיקסלים. המרנו את תמונות הקלט של המאגר לתמונות grayscale - בגווני אפור, שכן כך ניתן לייצג את התמונה במרחב דו מימדי.
עם זאת על מנת להשתמש במודלים היה עלינו להמיר את מטריצת הפיקסלים לוקטור בודד לכל תמונה
כאן נקטנו בשתי גישות:
הדרך הפשוטה והישירה הייתה להפוך כל מטריצה למערך שטוח על ידי חיבור כל השורות במטריצה. אך מימד התמונות המקורי היה (957, 821) והשטחה של תמונה לוקטור תהפוך כל תמונה במאגר להיות וקטור עם 957 * 821 פיקסלים ~ וקטור במימד 785,697.

במידה והיינו משתמשים בוקטור כזה לייצוג כל תמונה היינו נתקלים בשתי בעיות:

1. כמות עצומה של מידע לכל דגימה: עלות חישובית גבוהה ובעיית מקום בזיכרון.
2. בעיית overfitting ו"קללת המימדים" (curse of dimensionality)- ככל שהמימד של וקטור הפיצ'רים עולה כך הוא נעשה דליל יותר, והלמידה פחות טובה.

בשל כך עלה הצורך להעביר את התמונות צמצום מימדים כלשהו לפני הכנסתם למודלים.
לתהליך צמצום מימדים זה יש השפעה רבה על תוצאות המודלים, ולכן היה עלינו לנקוט במשנה זהירות.

- <u>המתודה הישירה - 'SIMPLE'</u>: כל תמונה מוקטנת לגודל קטן יותר על ידי אינטרפולציה של כל סביבת פיקסלים לפיקסל אחד (אופציית interpolation=cv2.INTER_CUBIC בפונקציה resize של OpenCV).
בדקנו מספר אופציות לגודל התמונה החדש - מ (20,20) עד (110,110) בקפיצות של 10.

- <u>המתודה השניה - 'HISTOGRAM'</u>: כל תמונה מומרת לוקטור של היסטוגרמת הצבעים שלה - וקטור בעל 256 מימדים המייצגים את גווני התמונה האפשריים, והערך בהם את מספר הפיקסלים השייכים לגוון זה בתמונה.

- <u>מתודת תיאור צורות בתמונה - 'HUMOMENTS'</u>: כל תמונה מומרת לוקטור בעל 7 מימדים, המתאר, מאפיין ומכמת את הצורה שבתמונה, באמצעות האלגוריתם 'Hu Moments' של הספרייה OpenCV (לעיבוד תמונה).

- <u>מתודה רביעית - 'PCA'</u>: או בשם האלגוריתם - Principal Component Analysis. לשם כך השתמשנו בספריית Scikit-Learn ובאובייקט PCA. דבר ראשון - העברנו את התמונות לממד אחיד (היות וכל תמונה בגודל מעט שונה עלה צורך זה). לאחר מכן שיטחנו כל תמונה לוקטור ארוך (בשימוש הפונקציה ששימשה למתודה 'SIMPLE'), היות וה PCA במימוש זה מצפה לקבל אובייקט דו מימדי (מימד אחד היה מספר הדגימות, מספר שני מספר הפיקסלים בכל תמונה). הPCA מקבל את התמונות משוטחות ומחלץ את האלמנטים החשובים ביותר בכל אחת.
עשינו בדיקה לממד האופטימלי של התמונות - מ (20,20) עד (110,110) בקפיצות של 10.

- <u>מתודה חמישית ואחרונה - 'EDGES'</u>: עשינו שימוש באלגוריתם Canny Edge (בשימוש בספריית OpenCV) למציאת קווי מתאר בתמונה. כל תמונת קווי מתאר מומרת לוקטור על ידי שיטוחה.
הרעיון היה, בשונה מהמתודה הישירה של שיטוח התמונה כמו שהיא, הוא שמציאת קווי המתאר תנקה "רעש" בתמונה ותבליט את הפיקסלים הרלוונטיים לניתוח התמונות.
גם כאן עשינו ניסוי למצוא את הממד האופטימלי של התמונות.

<u>שלב שני- למידת מכונה:</u>

לאחר שלב חילוץ הפיצ'רים (קובץ *Extract*) יש לנו 'X' - מטריצה שבה כל שורה מייצגת תמונה ו 'Y' - וקטור של התיוגים 1/0 התואם לשורות התמונות ב 'X'.
על מנת שאימון המודלים יעשה בצורה נכונה, עלינו לדגום דוגמאות אימון בצורה אחידה ממקבץ התמונות שיש ברשותנו. לכן הוספנו את הפונקציה לפיצול הדגימות לקבוצת אימון וקבוצת ביקורת (*splitTestTrain*) ואחראית גם על ערבול הדגימות בצורה רנדומלית (*shuffle*).

בשלב זה ברשותנו trainX, trainY, testX, testY - קבוצות הלמידה והביקורת עם התיוגים שלהן בהתאמה, אותם שלחנו למכונות הבאות:

- **K-nearest-neighbor** - כאשר מימשנו פעם אחת עם פונקצית מרחק אוקלידי כלומר p=2 ופעם נוספת השתמשנו בפונקצית מרחק של earth mover, הפונקציה *wasserstein_distance* של *scipy*.
- **Support Vector Machine** - כאשר בדקנו מספר kernels במטרה למצוא את האחד שיניב את התוצאה הכי טובה. הקרנלים שהשתמשנו היו: Linear , Polynomial, RBF(Gaussian), Sigmoid.
- **Decision Tree** - כאשר השתמשנו באלגוריתם CART עם מודל GINI (דיפולטיבי) -מדד זה מודד כמה פעמים אלמנט שנבחר באופן אקראי מהסט יתויג בצורה שגויה אם הוא תויג באופן אקראי בהתאם לחלוקת התוויות בקבוצות המשנה.
- **Adaboost** - המודל בספרייה Scikit-learn משתמש דיפולטיבית בעצי-החלטה כמסווג, השארנו את זה כך כי זהו המודל הנפוץ והיעיל ביותר בשימוש AdaBoost. מספר המסווגים: 50 (n_estimatorsint default=50).
- **Random Forest** - המודל בספרייה Scikit-learn משתמש דיפולטיבית ב 100 עצי בחירה.

❖ בכל הטכניקות האלה השתמשנו בספריית **sklearn** של scikit-learn .

<u>שלב שלישי - למידה עמוקה:</u>

בנפרד מכל המודלים המפורטים לעיל, בחרנו בנוסף לאמן רשת Cnn היות ואנו יודעים כי היא מותאמת בצורה הטובה ביותר לסיווג תמונות. בקובץ *Cnn* אנו עושים חלוקה מחדש לקבוצת אימון (80%) וקבוצת ביקורת (20%) באמצעות הפונקציה *train_test_split* של sklearn. את קבוצת הביקורת אנו מחלקים שוב לקבוצת ולידציה (10% מסך כל הדגימות) וקבוצת ביקורת (10% מסך כל הדגימות).

- **Convolutional Neural Network** - בניית הרשת:
  - ברשת Cnn סטנדרטית השכבות החבויות הן שכבות קונבולוציה ולאחריהן pooling.
    שימוש בשכבות ה pooling בין שכבות הקונבולוציה על מנת:
    1. להקל על הרשת ללמוד את הפיצ'רים החשובים בתמונה
    2. להקטין את מספר הפרמטרים שיש לאמן, ולכן לבסוף מביא להפחתת זמן האימון
    3. מניעת overfitting
  - בניית השכבה ראשונה - שכבת קונבולוציה: לשכבה הראשונה מספר פרמטרים שמשפיעים על המודל כולו והיה עלינו לקבוע אותם.
    <u>גודל ה kernel</u>: גודל זה צריך להיות סימטרי ואי זוגי - על מנת שכל הפיקסלים בשכבות הראשונות יהיו סימטריים סביב הפיקסל בפלט. בחרנו את המימד (3x3) , מכיוון שזוהי בחירה פופולרית ברוב הרשתות שראינו וככל הנראה משיגה את התוצאות הטובות ביותר. (ניסינו גם גודל 5x5)
    <u>מספר ה kernels</u>: הבחירה הייתה בין 16, 32, 64 לשכבה הראשונה ולהכפיל פי 2 כל פעם כשמפת הפיצ'רים ברשת הצטמצמה ב2 (maxpooling). כשבחרנו ב 16 kernels התוצאות היו מעט פחות טובות מב32 (94 אחוזי דיוק לעומת 100 אחוזי דיוק). לכן בחרנו 32 כגודל ממוצע.
  - מספר השכבות במודל: האידיאלי בין 5-100 לשכבות תלוי בכמה מורכבת המשימה. מכיוון שהמשימה שבחרנו יחסית פשוטה, נשארנו עם 3 שכבות של קונבולוציה, כשביניהן שכבות max pooling, ועוד שתי שכבות fully-connected בסוף.

אתגרים בפרויקט:

- האתגר הראשון בו נתקלנו קרה בניסיון לעבוד על dataset מורכב יותר המכיל תיקיות של מספר סריקות CT לכל פציינט, עם אבחון מפורט של מספר רופאים. לצערנו קבצים רפואיים מסוג זה נמצאים תחת פורמט dicom, ועל מנת להמיר אותם לקבצי numpy (מטריצות מספריות המהוות את האובייקט הבסיסי בכל הספריות ללמידת מכונה), היה עלינו להשתמש בספרייה מיוחדת. היחידה שמצאנו מתאימה לצרכים שלנו הייתה ספריית Pydicom, אך לספרייה זו קיימת בעיית התקנה שלא הצלחנו לפתור. לאחר כמה ימי עבודה ללא התקדמות החלטנו לעבור ל dataset המורכב רק מתמונות בפורמט רגיל.

- למרות שעברנו לעבוד עם קבצים מסוג מוכר, עדיין היה עלינו להבין איך לעבוד עם התמונות. הבנו שעלינו להציג את התמונה בצורה מספרית וקטורית עבור האלגוריתמים, האתגר היה להבין איך לעשות זאת. כפי שציינו לעיל, השתמשנו בשתי טכניקות שונות לחילוץ הפיצ'רים ובדקנו מה היעילה ביותר, בהתחשב בשיקולי איכות ומהירות.

- אתגר נוסף היה למצוא את הגודל המתאים ביותר של וקטור הפיצ'רים - לא רצינו לאבד מידע חשוב מהתמונות אבל ראינו שיש מודלים, knn למשל, שנותנים תוצאות פחות טובות ככל שגדל הוקטור. כלומר כשיש יותר פיצ'רים (curse of dimensionality).

- אתגרים נוספים היו העבודה עם ספריות רבות בפייתון שלא יהיו מאוד מוכרות לנו(tensorFlow ו-sklearn), שלמרות שהקלו מאוד על העבודה גרמו לנו לחקור לעומק כל ספרייה בשביל להבין כיצד להשתמש.

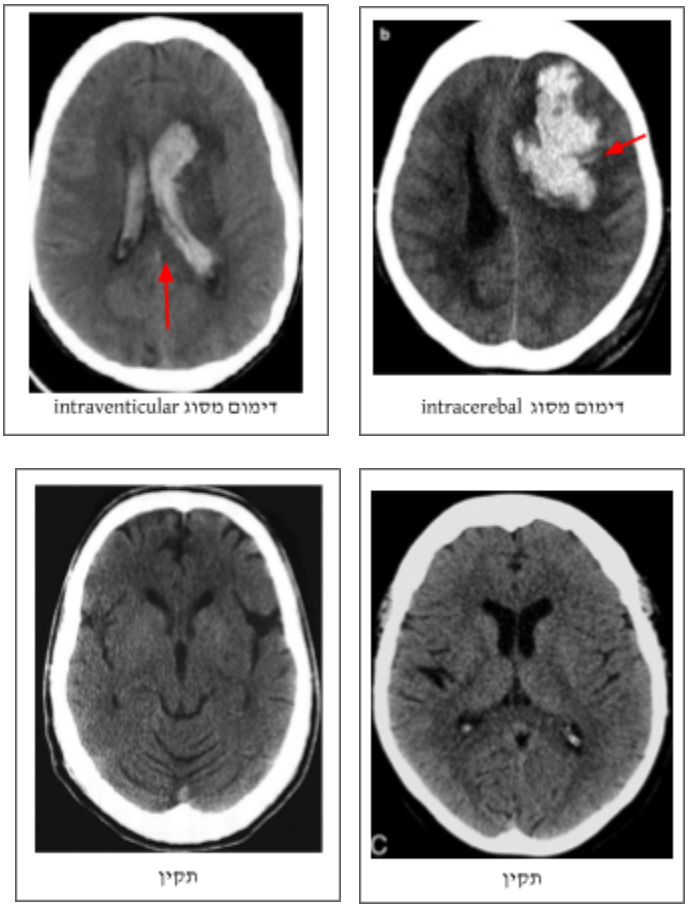
התוצאות וניתוחן:

נחלק את התוצאות שקיבלנו לפי אופן הוצאת הפיצ'רים מהתמונות, שכן יש השפעה רבה לתהליך זה על יעילותם של המודלים ונרצה להבדיל בין המקרים.(ראה קובץ *resultSimple.txt* ו *resultHistogram.txt*) לפני שנסביר את התוצאות יש לציין דבר אחד חשוב על התמונות ששווגו כתמונות עם דימום- לא היה הבדל בין סוגי הדימומים.

ממחקר קטן סביב נושא זה הבנו שיש מספר סוגי דימומים -

- דימום תוך-מוחי (פנים-אקסיאלי)
- דימום חוץ-מוחי (חוץ-אקסיאלי) שגם הוא מתחלק לשלושה תת סוגים - דימום תת-עכבישי, דימום תת-דוראלי ודימום אפידורלי.

לכל דימום יש מאפיינים שונים וניתן לראותם ויזואלית בסריקת CT.
לכן רק התמונות של סריקת CT תקינה היו דומות אחת לשניה ואילו סריקות עם דימום יהיו שונות זו מזו בהתאם לסוג הפגיעה.
לכן יש להניח כי ההפרדה בין התמונות עם הדימום לתמונות בלי הדימום היא הפרדה בין התמונות בהן לא היו כלל צורות/צבעים המעידים על הפגיעה ותמונות עם מגוון שונה של פגיעות.



דוגמא להבדל בין שתי תמונות המסווגות כתמונות עם דימום (שתי התמונות העליונות) לעומת הדמיון בין שתי תמונות המסווגות כתמונות ללא דימום.

<u>באופן כללי:</u> אחוזי הדיוק של המודלים בשימוש וקטור הפיצ'רים מהמתודה '**SIMPLE**' היו גבוהים יותר מכל מתודה אחרת, עם אחוזי דיוק של 90% במודל Svm ובמודל Random-Forest.
אפשר להגיד שהמתודה '**PCA**' הייתה השניה בגובה אחוזי הדיוק, עם אחוזי דיוק של 80%-87.50%, אבל רק עבור מודל הSvm לכל סוגיו (בשימוש עם kernels שונים).
'**HISTOGRAM**' הייתה הבאה, וגם המהירה ביותר שהניבה דיוק יחסית טוב יש לציין, עם אחוזי דיוק של מעל 70% לרוב המודלים.
זוהי תוצאה הגיונית, שכן וקטור הפיצ'רים בשני המקרים הראשונים הוא וקטור בגודל של החל מ 400 פיקסלים (20x20) ועד 12,100 (110x110), ואילו וקטור הפיצ'רים במקרה השני הוא וקטור של 256. כאשר יש יותר פיצ'רים לכל תמונה יש למודל יותר "מידע" לעבוד איתו.
המתודה '**EDGES**' התבררה כמפחיתת עוצמת המודלים, עם אחוזי דיוק של למטה מ-50% להרבה מהמודלים החזקים ביותר (Knn, Random-Forest).
אחרונה היא המתודה '**HUMOMENTS**' עם אחוזי דיוק גרועים ביותר, כשהגבוה ביניהם 57.50%.

ניתוח התוצאות לפי כל מתודה להוצאת פיצ'רים:

1. כאשר השתמשנו במתודה 'SIMPLE'- תמונה מושטחת:

בניסויים הראשונים כאשר חיפשנו את גודל וקטור-הפיצ'רים האידיאלי, אפילו כשהקטנו את התמונה להיות בגודל של 20x20, שזה 2/100 מגודל התמונה המקורי, knn, random-forest וה svm הצליחו לסווג ברמת דיוק גבוהה, כאשר ה knn עמד על רמת דיוק של 80%.
מעניין לשים לב לדבר מעניין שקרה - אחוזי דיוק של ה knn ירדו ככל שהגדלנו את גודל וקטור הפיצ'רים. דבר זה תואם את הציפיות שלנו ולמה שנקרא "קללת המימדים", אך כאשר המשכנו להגדיל את וקטור הפיצ'רים, אחוזי הדיוק התחילו להתנדנד- עולים ויורדים בלי תבניתיות.

Image size:	20x20	30x30	40x40	50x50	60x60	70x70	80x80	90x90	100x100	110x110
Knn-EMD:	57.50%	65%	77.50%	67.50%	70%	65%	77.50%	67.50%	57.50%	75%
Knn:	80%	75%	70%	70%	87.50%	75%	85%	57.50%	77.50%	87.50%
Svm-linear:	72.50%	87.50%	75%	75%	77.50%	85%	82.50%	72.50%	80%	72.50%
Svm-poly:	72.50%	**90%**	72.50%	77.50%	87.50%	80%	80%	67.50%	85%	75%
Svm-RBF:	62.50%	77.50%	75%	72.50%	82.50%	90%	85%	67.50%	77.50%	75%
Svm-sigmoid:	50%	37.50%	32.50%	37.50%	30%	25%	27.50%	60%	35%	27.50%
Random forest:	71.50%	88.50%	77%	85%	**90%**	86.50%	82.50%	73.50%	82%	85%
Decision tree:	60.50%	85%	73%	73.50%	66%	75%	59.50%	60.50%	77%	78.50%
AdaBoost:	65%	75%	67.50%	82.50%	77.50%	72.50%	79%	72.50%	85%	70%

כפי שניתן לראות לאורך כל הניסוי המתואר לעיל בטבלה - המודלים בעלי אחוזי הדיוק הגבוהים ביותר הם ה-Knn, ה -Svm וה Random forest.

- **Knn**: נראה שהמודל רגיש לשינויים בגווני הצבע בין התמונות המסווגות שונה.
- **Svm**: ידוע כמודל המסוגל להתמודד עם דגימות במימד גבוה, אפילו כאשר מימד הפיצ'רים גדול ממספר הדגימות (כמו במקרה שלנו).
- **Random- forest:** ידוע כמודל בין החזקים ביותר שיש היום בעולם למידת מכונה, לכן הניסוי שלנו עומד בקנה אחד עם הציפיות מאלגוריתם זה.

2. כאשר השתמשנו במתודה של 'HISTOGRAM' - היסטוגרמת תמונה:

model:	Knn-EMD	Knn	Svm-linear	Svm-poly	Svm-RBF	Svm-sigmoid	Random forest	Decision tree	AdaBoost
accuracy:	77.50%	72.50%	67.50%	65.00%	80.00%	30.00%	74.75%	76.75%	78.75%

אנו מניחים כי לסריקות CT התקינות תהיה התפלגות יחסית קבועה בגווני האפור שכן דימום או פגיעה יבלטו בצבע מסוים, ובסריקות הכוללות דימום תהיה סטייה כלשהי בגוון/מספר גווני אפור מההתפלגות הזו.
ניתן לראות שהמודל שהניב את התוצאות הכי מדויקות היה ה **svm-RBF** עם 80% אחוזי דיוק.
ראויים לציון גם **ה-AdaBoost**, מודל ה- **Knn** המבוסס על פונקצית המרחק earth-mover ומודל ה - **Decision Tree** שהניבו שלושתם אחוזי דיוק של 78.75% , 77.5% , 76.75% בהתאמה.
נשים לב שה- Random-Forest לא רחוק מאחוזי הדיוק של המודלים הללו עם 74.75% וכך ציפינו. אך הופתענו מאוד שגם ה Adaboost נתן לנו תוצאות טובות.

בהשוואה לתוצאות עם המתודה '*SIMPLE*' הבנו שהשיטה לייצג את התמונות באמצעות ההיסטוגרמה שלהן היא לא השיטה הטובה ביותר, והיא מאבדת יותר מדי מידע קריטי לסיווג.

3. כאשר השתמשנו ב'HUMOMENTS' -תיאור הצורה שבתמונה באמצעות האלגוריתם 'Hu Moments':

לאורך כל הניסויים מתודה זו הניסה את התוצאות הנמוכות ביותר, כפי שניתן לראות בטבלה הבאה.
מהתחושה שלנו - הסיבה היא שהאלגוריתם הזה מחלץ את המאפיינים של הצורה מהצללית שיש בתמונה, אך מפספס פרטים עדינים שנעלמים במעבר לוקטור בעל 7 ממדים בלבד.

model:	Knn-EMD	Knn	Svm-linear	Svm-poly	Svm-RBF	Svm-sigmoid	Random forest	Decision tree	AdaBoost
accuracy:	50%	52.50%	52.50%	55%	55%	50%	53%	57.50%	47.50%

4. <u>כאשר השתמשנו ב'PCA' -הורדת מימד התמונה באמצעות Principal Component Analysis:</u>

הכלים שהניבו את התוצאות הטובות ביותר היו ה- **SVM** למיניהם.
ראוי לציין שבהשוואה למתודות הקודמות שסקרנו, ה **svm-sigmoid** התגלה ככלי יעיל עם שיטה זו. בניגוד למתודה 'SIMPLE' וכן למתודה 'HISTOGRAM', בהן ה svm-sigmoid הניב תוצאה ממוצעת של 36.25%, כאן כלי זה הניב ממוצע תוצאות של 75.75%.

Image size:	20x20	30x30	40x40	50x50	60x60	70x70	80x80	90x90	100x100	110x110
Knn-EMD:	55%	65%	77.50%	72.50%	67.50%	62.50%	57.50%	60%	77.50%	67.50%
Knn:	62.5%	60%	57.50%	60%	55%	70%	65%	62.50%	65%	75%
Svm-linear:	70%	67.50%	70%	65%	70%	62.50%	70%	80%	77.50%	80%
Svm-poly:	82.50%	80%	80%	70%	70%	65%	62.50%	77.50%	65%	55%
Svm-RBF:	77.50%	77.50%	75%	77.50%	67.50%	80%	**87.50%**	75%	77.50%	82.50%
Svm-sigmoid:	77.50%	75%	75%	67.50%	67.50%	80%	80%	82.50%	70%	82.50%
Random forest:	74%	66.25%	72.50%	70.75%	68.75%	70.50%	65.50%	70.50%	71.25%	64.50%
Decision tree:	58.50%	60.75%	59.50%	47.50%	59%	61.50%	65%	66.50%	54%	64.50%
AdaBoost:	77.50%	55%	65%	55%	65%	57.50%	75%	72.50%	75%	65%

<u>כאשר השתמשנו ב'EDGES'- חילוץ קווי המתאר באמצעות האלגוריתם 'Canny Edges':</u>

כמה דברים מעניינים ששמנו לב אליהם - בהשוואה לתוצאות של המתודה 'SIMPLE':
המתודה svm sigmoid קפצה בביצועים שלה בצורה משמעותית (לדוגמא מ 37.50% במקרה של תמונה בגודל 30x30 במתודה 'SIMPLE'- לאחוזי דיוק של 67.50%, כמעט פי 2).
לעומת זאת, היו כלים כמו ה Desicion tree שאחוזי הדיוק שלו צנחו בצורה דרמטית (מ85% בגודל תמונה של 30x30 לאחוזי דיוק של 49% - פחות מניחוש רנדומלי!).
התוצאות של הKNN למיניהם הפכו להיות ממוצעות.

טבלת התוצאות של 'EDGES':

Image size:	20x20	30x30	40x40	50x50	60x60	70x70	80x80	90x90	100x100	110x110
Knn-EMD:	62.50%	60%	57.50%	55%	60%	50%	55%	50%	52.50%	30%
Knn:	67.50%	60%	65%	60%	65%	52.50%	47.50%	45%	42.50%	50%
Svm-linear:	57.50%	65%	67.50%	62.50%	62.50%	57.50%	70%	70%	62.50%	50%
Svm-poly:	65%	75%	65%	70%	62.50%	50%	57.50%	45%	42.50%	70%
Svm-RBF:	67.50%	75%	62.50%	60%	70%	52.50%	65%	57.50%	52.50%	47.50%
Svm-sigmoid:	57.50%	67.50%	62.50%	62.50%	70%	50%	62.50%	65%	55%	45%
Random forest:	67%	64.25%	57.50%	61%	66.50%	61.75%	64.50%	60.50%	59.50%	51.50%
Decision tree:	55%	49%	48.75%	55%	43.75%	61.25%	59%	70%	58.75%	54.25%
AdaBoost:	57.50%	50%	50%	50%	60%	52.50%	62.50%	45.00%	52.50%	57.50%

תוצאות convolutional neural network:

כמו שציפינו - המודל שסיווג את ה תמונות ברמה הטובה ביותר היה ה cnn.
זאת בזכות טבעה של הרשת - קונבולוציה, העוזרת להתחשב בסביבה של כל פיקסל ובכך לתת משמעות להקשר הסביבתי של כל מאפיין בתמונה. רשת כזו מחלצת בצורה אוטומטית את הפיצ'רים החשובים ביותר בתמונה ובכל שכבה ברשת קטן מימד הקלט אך נהיה מתומצת יותר.

image-size:	30x30	**40x40**	**50x50**	60x60	**70x70**	**80x80**	**90x90**	100X100	110x110	120x120	**130x130**	140x140
Cnn accuracy:	89.99%	100%	100%	94.99%	100%	100%	100%	94.99%	89.99%	94.99%	100%	94.99%

לסיכום,

ניתן להגיד כי התוצאות לא הפתיעו אותנו יותר מדי,והמכונה שביצעה את משימת הסיווג בצורה הטובה ביותר הייתה רשת הקונבולוציה -Cnn. אך בכל זאת, ראינו כי גם מכונות פשוטות יותר ועם הרבה פחות מידע הצליחו לחזות את הסיווג הנכון, במיוחד הSvm וה Random-Forest שעם גודל תמונה של 30x30 בשימוש בשיטה *SIMPLE* הניבו תוצאות כמעט זהות למודל הCnn (תוצאות של 88.5% ל- Random-Forest , תוצאות של 89.99% ל-Cnn ו 90% לSvm).