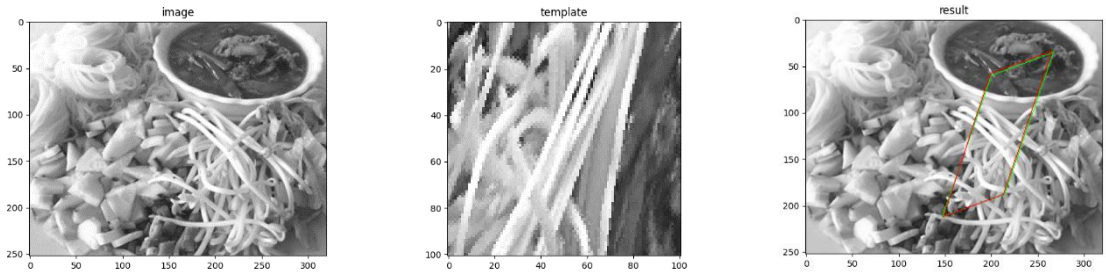
# FAsT-Match

**Research and Improvement**




**אוניברסיטת חיפה, החוג למדעי המחשב**

**אתגר 14**

**אוגוסט 2022**


פרויקט גמר בהנחיית ד"ר סיימון קורמן

אסף סולומיאק, 212585863, asafucho@gmail.com

קארין ספרי, 213883903, karin.sifri@gmail.com

# תוכן

## הגדרת הבעיה

אלגוריתם FAsT-Match הוא אלגוריתם חיפוש שמטרתו היא התאמה של תבנית בתוך תמונה כך שההפרש בין הפיקסלים בתמונה ובתבנית יהיה מינימלי.

התבנית אותה אנו מחפשים בתמונה יכולה להיות בכל מקום בתמונה, מוקטנת/מוגדלת, מסובבת ואפינית (affined, כלומר שינוי מתיחות). מכאן כמות הטרנספורמציות שיש לקחת בחשבון היא עצומה והחסם התחתון בכל תת מרחב פתרונות משתנה בהתאם לפרמטרים ולפונקציית השגיאה.

האלגוריתם משתמש בעיקרון Branch and Bound. אלגוריתם B&B נוצר לבעיות אופטימיזציה ומטרתו למצוא את הפתרון הגלובלי האופטימלי בהסתמך על חסמים עליונים ותחתונים על אזורים של מרחב החיפוש. במילים אחרות, המטרה היא למצוא ערך x הממזער את הערך של הפונקציה $f(x)$ מתוך קבוצת פתרונות אפשריים $S$.

Branch: האלגוריתם מחלק באופן רקורסיבי את מרחב החיפוש למרחבים קטנים יותר, אשר בהם הוא ממזער את $f(x)$.

Bound: כדי לשפר את חיפוש ה- Brute-Force על כל המרחב $S$, האלגוריתם משתמש בחסמים שבעזרתם הוא מדיח תתי מרחבי פתרונות שניתן להוכיח שאינם מכילים את הפתרון האופטימלי.

האלגוריתם מחשב את החסם, אשר מדיח את תתי מרחב הפתרונות שאינם מכילים את הפתרון האופטימלי, באופן ידני על סמך ניסוי וטעיה.

## הגדרת הפתרון לבעיה

מטרת הפרויקט היא מימוש האלגוריתם FAsT-Match ושיפורו באמצעות תכנון והקמה של רשת נוירונים אשר תהיה אחראית לבחירת אותו החסם ובכך תאפשר לשפר את דיוק האלגוריתם.

## פירוט המימוש

המימוש נעשה בשפת Python בסביבות הפיתוח PyCharm ו-Google Colab.

בשלב הראשון היינו צריכים לממש את האלגוריתם המקורי ב-Python. עשינו זאת בעזרת תרגום הקוד המקורי משפת התכנות MATLAB.

בשלב השני היינו צריכים ליצור מסד נתונים בו נוכל להשתמש ביצירת הרשת. אספנו 100 תמונות אקראיות מתוך מסד נתונים שמצאנו באינטרנט. הרצנו את האלגוריתם על 100 תבניות אקראיות על כל תמונה ובכל שלב בהרצה אספנו את המידע על השלב.

הנתונים שאספנו הם:

- מימדי התמונה
- מימדי התבנית
- היחס בין ממוצע הפיקסלים בתבנית לממוצע הפיקסלים בתמונה
- חלקות התבנית (ממוצע הנגזרות בתבנית)
- סטיית התקן של הפיקסלים בתבנית
- delta - הערך שלפיו נקבע החסם באלגוריתם המקורי (ערך המשתנה באופן לינארי בלי תלות בתבנית)
- המרחק המינימלי שהתקבל בין הטרנספורמציות על התבנית הנבדקות באותו השלב לבין התמונה
- ממוצע המרחקים שנמדדו באותו השלב
- סטיית התקן של המרחקים שנמדדו באותו השלב
- האחוזונים 5, 10, 15, ..., 95 של המרחקים שנמדדו באותו השלב
- התחום בו נמצאים המרחקים שנמדדו (ההפרש בין המרחק הגדול ביותר למרחק הקטן ביותר)
- כמות הקונפיגורציות שנבדקו באותו השלב
- ערכי המטריצה של הטרנספורמציה בה מתקבל המרחק המינימלי

סך הכל, שמרנו 37 features כמו גם את החסם האופטימלי המחושב מראש בו נשתמש כ-target במודל.

בנוסף שמרנו בקובץ נפרד לכל שלב את המרחק המינימלי שלו, המרחק המקסימלי שלו ואת היסטוגרמת כל המרחקים באותו השלב בחלוקה ל-100 תחומים. מידע זה נאסף על מנת למדוד את ה-Accuracy של המודל – את כמות הקונפיגורציות (המשוערך) שה-prediction של המודל מעביר לשלב הבא

לבסוף, אספנו kb73.5 של 68854 samples בזמן חישוב כולל של 121.86 שעות.

על מנת להקל את החישוב, חילקנו את התמונות ל-20 תיקיות והרצנו את החישוב על כל תיקייה בנפרד, כך קיבלנו 20 קבצים כך שבכל קובץ יש בממוצע 3442.7 samples.

השלב השני בפרויקט היה לממש מודל בתחום ה-Machine Learning או ה-Deep Learning אשר בשילובו באלגוריתם, הוא יסייע לנו לבחור את החסם העליון המתאים ובך לשפר את ה-Accuracy של האלגוריתם.

בהתחלה, התבקשנו לחקור בתחומים אלו וללמוד כיצד נוכל לבצע זאת ומהו המודל המתאים ביותר לבעיה אותה אנו מנסים לפתור. בסוף, בחרנו להשתמש במודל מסוג Multi-Layer Perceptron for Regression. מודל זה, הוא רשת נוירונים פשוטה המורכבת משכבות ליניאריות.

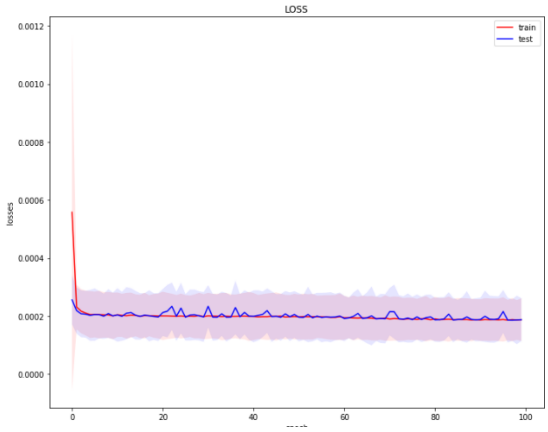
בנוסף, החלטנו להשתמש רק ב10 מה-features אותם אספנו מכיוון שלאחר איסוף ה-samples ראינו שהרבה מהם תלויים אחד בשני מאוד או לא נחוצים למודל. לכן, ה-features בהם אנו משתמשים הם:

- מימדי התבנית
- היחס בין ממוצע הפיקסלים בתבנית לממוצע הפיקסלים בתמונה
- חלקות התבנית (ממוצע הנגזרות בתבנית)
- סטיית התקן של הפיקסלים בתבנית
- delta - הערך שלפיו נקבע החסם באלגוריתם המקורי (ערך המשתנה באופן לינארי בלי תלות בתבנית)
- המרחק המינימלי שהתקבל בין הטרנספורמציות על התבנית הנבדקות באותו השלב לבין התמונה
- ממוצע המרחקים שנמדדו באותו השלב
- סטיית התקן של המרחקים שנמדדו באותו השלב
- התחום בו נמצאים המרחקים שנמדדו (ההפרש בין המרחק הגדול ביותר למרחק הקטן ביותר)
- כמות הקונפיגורציות שנבדקו באותו השלב

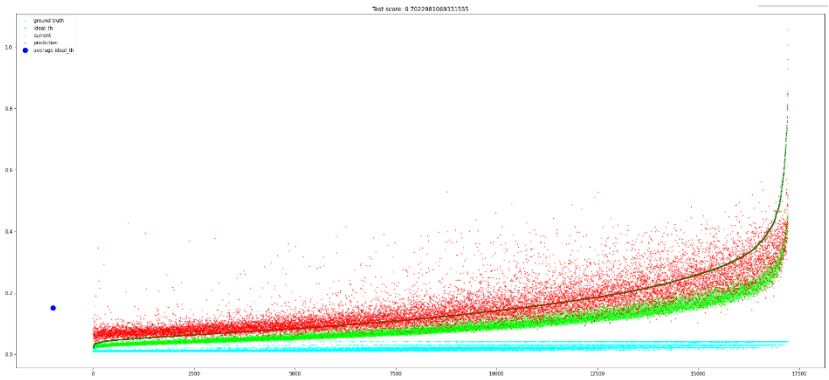
תהליך בחירת הפרמטרים של המודל התבצע בעיקר ע"י ניסוי וטעיה, ובדיקת המודל בעזרת 3 ערכים שונים:

- Loss – השתמשנו ב-MSE (כלומר המרחק בין ה-prediction של המודל ל-target בריבוע) בתור פונקציית ה-Loss של המודל.
- Score – ערך המייצג את היחס בין ה-Loss ב-prediction של המודל, לבין ה-Loss שהיינו מקבלים אם היינו בוחרים בערך הממוצע של ערכי ה-target בכל פעם.
- Accuracy – אחוז הפעמים שהצלחנו לנחש מספר גבוה מספיק אשר מעביר את הקונפיגורציה האמיתית לשלב הבא. כלומר, $prediction > ground\_truth$.

ריצה לדוגמה:



תוצאות על ה-Test-Set:

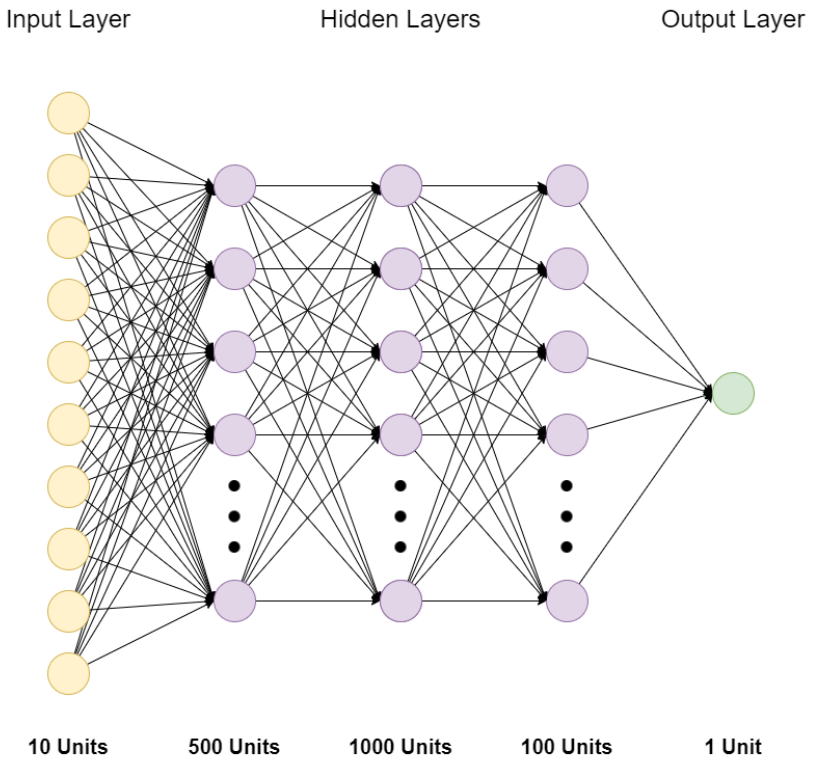


$Loss$: 0.002872562920136345

$Score$: 0.725584916339987

$Accuracy$: 0.9853607528755663

לאחר ניסיון של אפשרויות שונות ובדיקות רבות הגענו למודל הבא:



ול-hyper-parameters הבאים:

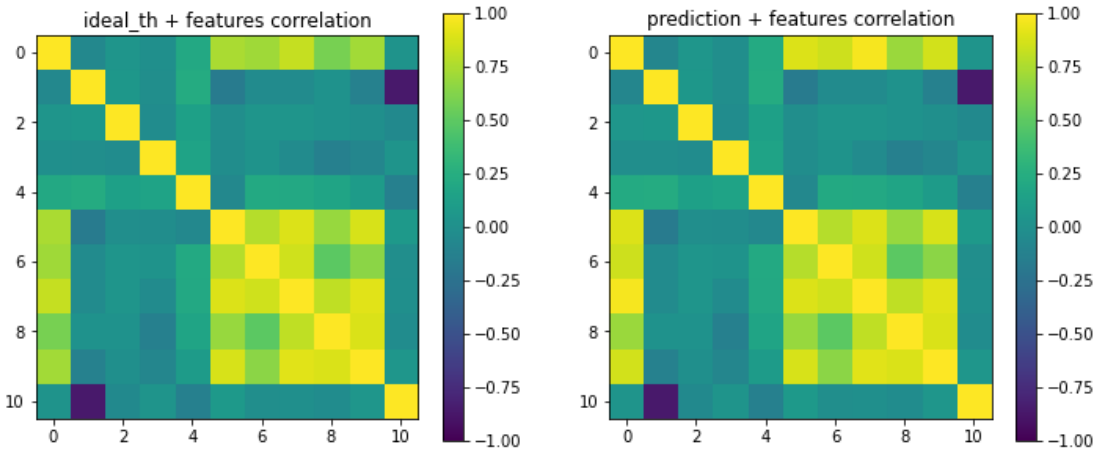
$$number\_epoch = 100$$

$$size\_minibatch = 100$$

$$learning\_rate \ = \ 0.00001$$

$$weight_{decay} = \ 0.000001, (for\ ADAM\ optimizer)$$

תמונת ה-*feature correlation* פלוס ערך ה-*target* (בעמודה והשורה ה-0):




בעזרת המודל הנ"ל אימנו 6 מודלים שאותם שילבנו ובדקנו באלגוריתם עצמו.

ההבדל במודלים הוא בערך ה-*target* שלהם:

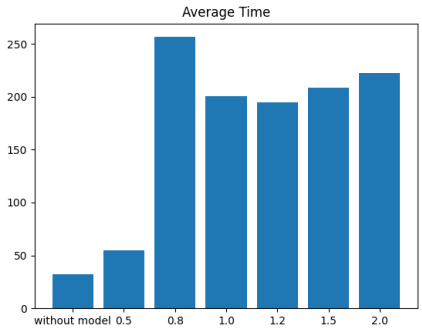
$$\min\_distance + factor * (ground\_truth - \min\_distance)$$

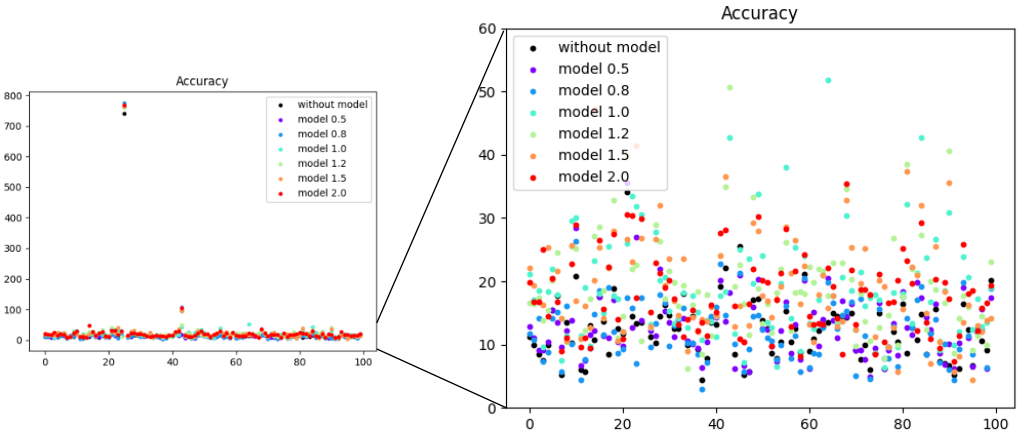
כאשר ערך ה-*factor* משתנה בין המודלים מבין הערכים:

$$\{0.5, 0.8, 1.0, 1.2, 1.5, 2.0\}$$

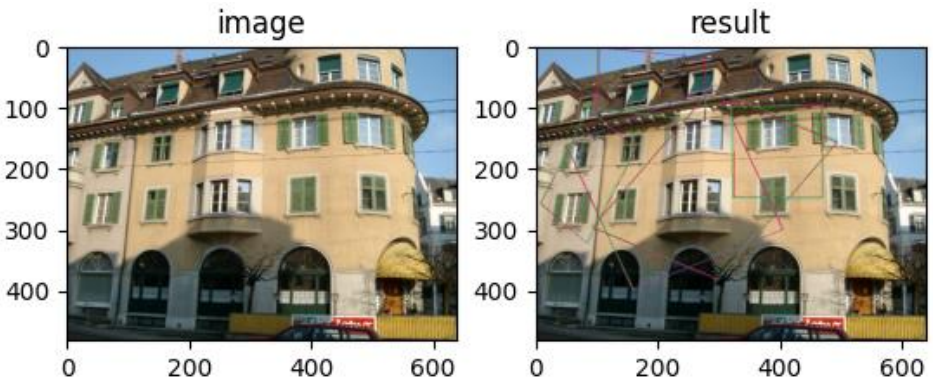
השלב האחרון בפרויקט היה לבדוק איך האלגוריתם *FAsT-Match* מתפקד בשילוב המודלים, והאם הצלחנו לשפר את הדיוק שלו.

הרצנו 100 פעמים על כל מודל (10 תמונות ו-10 תבניות לכל תמונה) ועוד פעם אחת נוספת ללא מודל, ומדדנו את הזמן הממוצע עבור ריצה יחידה לכל מודל וכן את הדיוק של המודל בעזרת המרחק של הפינות שנמצאו מהפינות האמיתיות של התבנית. קיבלנו את הנתונים הבאים:




ריצות לדוגמה עם מודל:



## הרצת האלגוריתם

ניתן להריץ את הקוד מתוך PyCharm, על ידי הרצת ה-main. על מנת להריץ את הקוד יש להוריד את ה packages-הבאים:

- Numpy
- Matplotlib
- Cv2
- Sklearn
- Pandas
- Torch

## פירוט מאפיינים

בהרצת האלגוריתם ניתן לבחור בכמה פרמטרים שונים המגדירים את מרחב החיפוש של האלגוריתם ואת כמות נקודות הדגימה שהאלגוריתם דוגם בתבנית על מנת להעריך את דיוק הקונפיגורציות. בנוסף, ניתן להגדיר שהאלגוריתם ינרמל את רמת הבהירות בתמונה ובתבנית (במידה והתבנית לא נלקחה באופן מלאכותי מהתמונה).

כמו כן, ניתן לבחור מבין המודלים שאימנו את המודל בו האלגוריתם ישתמש, או לבחור שהוא לא ישתמש באחד המודלים.

לבסוף, בהפעלת הריצה עצמה יש לבחור תמונה עליה האלגוריתם ירוץ, ותבנית העשויה להיות מוגדרת מראש או שייבחר תבנית אקראית.

## איך ניתן להמשיך את הפרויקט

ישנם כמה כיוונים שאיתם חשבנו שיש עם מה להתקדם.

ניתן לחשוב על כיוון אחר מבחינת המודל, או לנסות להשיג תוצאות טובות יותר על ידי שינוי ה-target ו/או הפרמטרים. כמו כן, אפשר להשתמש במודל למידה אחר להכרעת פרמטרים נוספים באלגוריתם.

## מה למדנו במהלך הפרויקט

למדנו לקרוא ולהבין את שפת התכנות MATLAB אשר ממנה תרגמנו את האלגוריתם. כמו כן, מכיוון שהפרויקט כולו נכתב בשפת התכנות Python, הצלחנו לשפר את היכולות שלנו בו ולהבין איך לייעל תהליכים בשפה זו.

בנוסף, למדנו רבות על תחום ה-Deep Learning וכיצד לבחור ולאמן רשת נוירונים.

## צוואר הבקבוק

לפי דעתנו, הקושי הגדול ביותר בפרויקט היה לממש את האלגוריתם ב-Python. בהתחלה אף ניסינו לתרגם את האלגוריתם מ-C++ אך לא הצלחנו, ואז עברנו לנסות לתרגם מ-MATLAB ואפילו כשבסוף הצלחנו לממש, נתקלנו בבעיה שהמימוש שלנו לקח זמן רב באופן משמעותי יותר מהמימוש ב-MATLAB.

בשביל לתקן זאת היינו צריכים לחפש פתרונות לאיך לייעל את הריצה בכמה דרכים.