Shelly Gamlielly
Aqiva Hassan

CV Ex4

Part 3.5 - Binary digit learning

Use VGG16 in order to get features from mnist images of 0/1

Use nearest neighbor to classify

קוד - ניתן לצפות במחברת של colab :

https://colab.research.google.com/drive/1Lpax85uIe19-KdS6bCdPybyiY-EE_iSG#scrollTo=E0JoK8EmoPcf

הסברים :

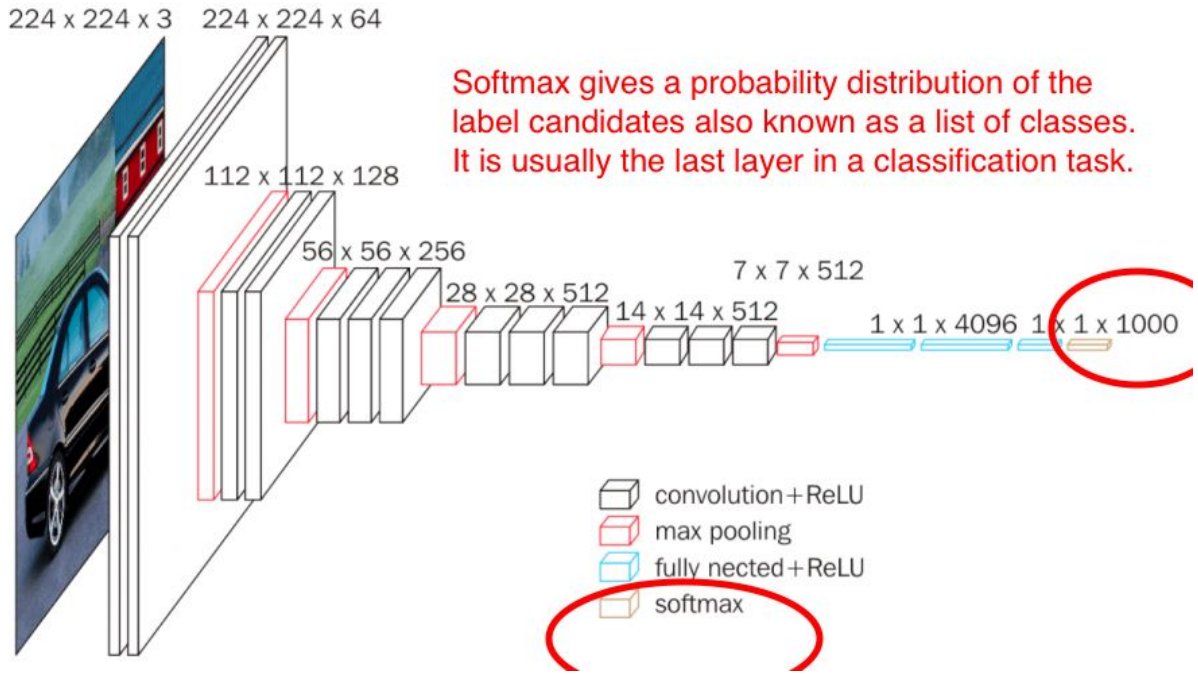
- עיבוד הדאטא - מכיוון שעלינו לבצע קלסיפיקציה רק על הספרות 0 ו- 1 נוכל לחלץ מהדאטא של mnist את כל התמונות והתיוגים של הספרות 0 ו-1 בלבד. זה מתבצע בפונקציה get_2_labels.

  כמו כן, היה עלינו לבצע preprocess על מנת לודא שהקלט יתאים לvgg16:

  i. Resize the images 48*48 as required by VGG16
  ii. Reshape images as per the tensor format required by TensorFlow
  iii. Convert the images into 3 channels
  iv. Normalize the data and change the data type to float32
  v. Preprocessing the input using preprocess_input:
  from keras.applications.vgg16 import preprocess_input

- עשינו **שימוש בvgg16 בתור feature extractor** באמצעות הסרת השכבה האחרונה שמבצעת את הקלסיפיקציה. נצפה כי הפיצ'רים שנבחרו יכילו מידע רלוונטי מהתמונות שהכנסנו כקלט, כך שניתן יהיה לבצע את המשימה של זיהוי ספרות על ידי רפרזנטציה שמכילה את המידע הרלוונטי ולא את כל המידע ההתחלתי.

  כפי שניתן לראות בתמונה שכבת האקטיבציה של softmax מחזירה וקטור הסתברויות בו מוצגת ההסתברות לכל אחת מהמחלקות ובאמצעותה מבצעים קלסיפיקציה, לכן על ידי הסרת שכבה זו נקבל כפלט את הפיצ'רים.



<u>Using a pre-trained model in Keras to extract the feature of a given image</u>

VGG is a convolutional neural network model for image recognition proposed by the Visual Geometry Group in the University of Oxford, where VGG16 refers to a VGG model with 16 weight layers.

The architecture of VGG16: the input layer takes an image, and the output layer is a softmax prediction on 1000 classes. From the input layer to the last max pooling layer (labeled by 7 x 7 x 512) is regarded as the feature extraction part of the model, while the rest of the network is regarded as the classification part of the model.

אז בעצם על ידי הסרת השכבה האחרונה נקבל :

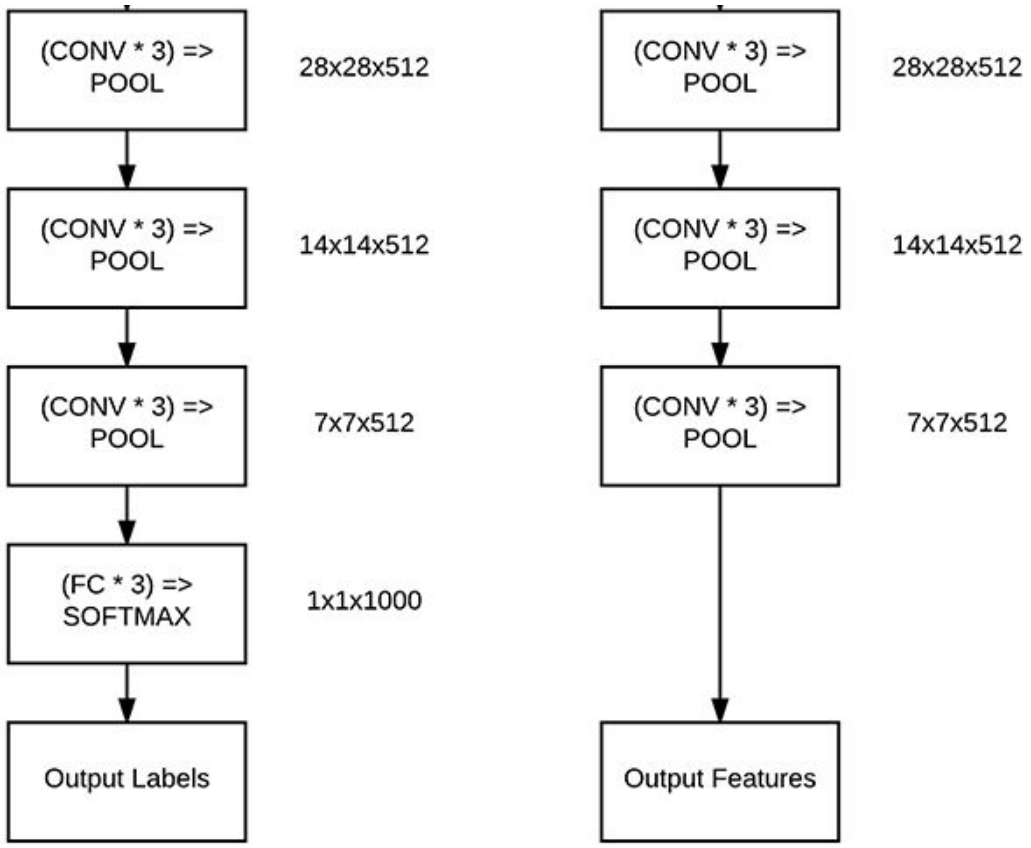


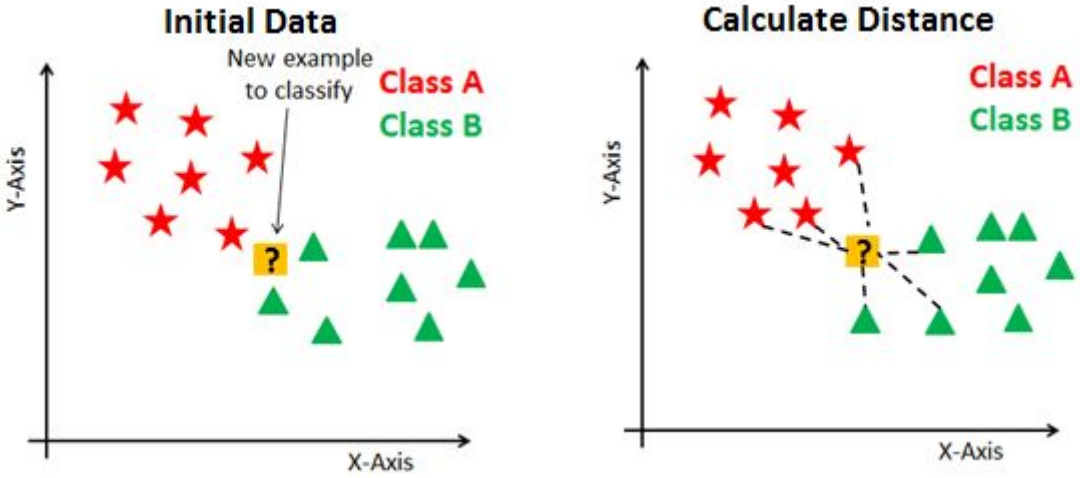
Figure 1: *Left*: The original VGG16 network architecture that outputs probabilities for each of the 1,000 ImageNet class labels. *Right*: Removing the FC layers from VGG16 and instead returning the final POOL layer. This output will serve as our extracted features.

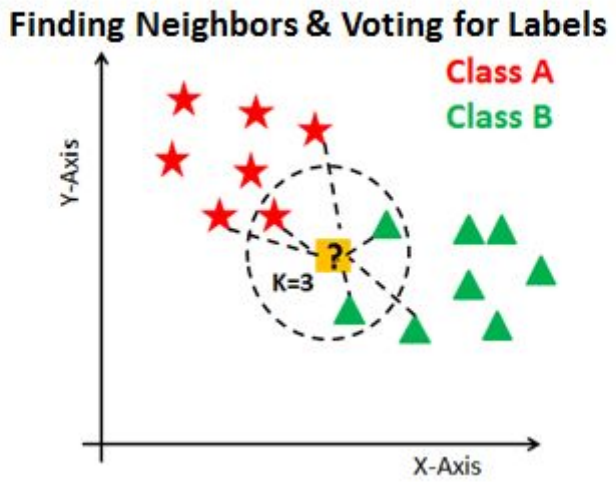
- עשינו **שימוש ב- Nearest Neighbors Classification מהספריה scikit-learn**
הכנסו את מס' השכנים שלפיו נבצע את החישוב בתור k=5 שנתן לנו את הדיוק הגבוה היותר.
איך זה עובד:

<u>KNN classifier model</u>

1. Import the KNeighborsClassifier module and create KNN classifier object by passing the argument number of neighbors in KNeighborsClassifier() function.
2. Fit your model on the train set using fit() and perform prediction on the test set using predict().

נתבונן באיור הבא שממחיש כיצד האלגוריתם פועל :



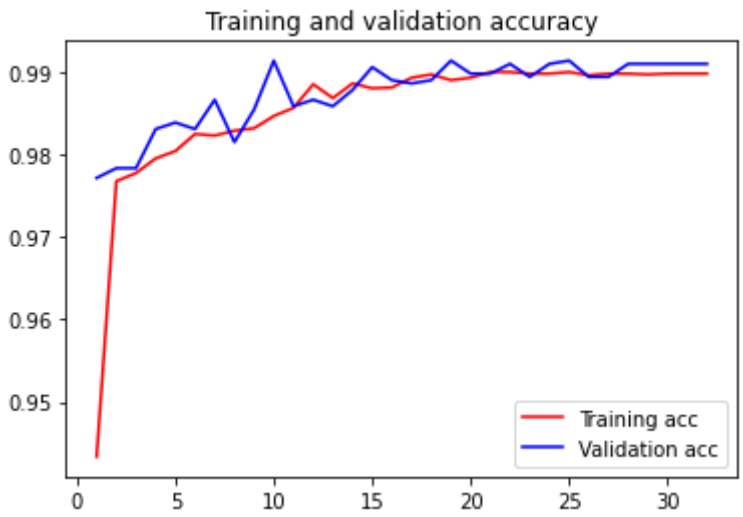


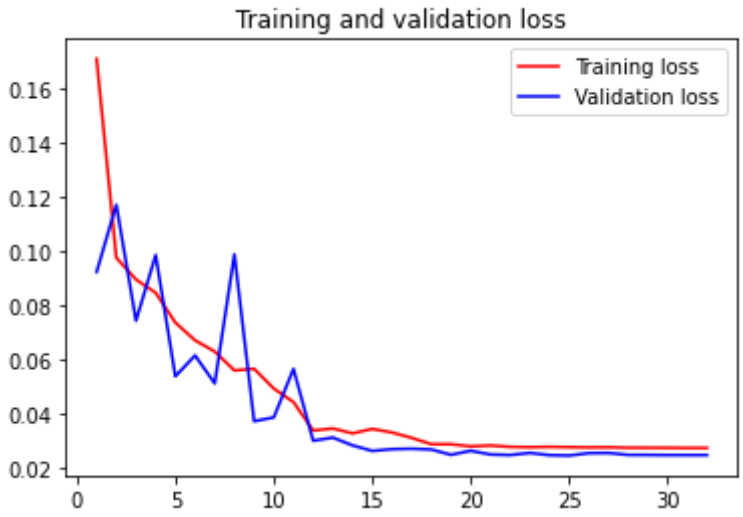
In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor.
First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN has the following basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels

- בדיקה נוספת שבצענו לשם השוואה עם KNN היא **שימוש בקלסיפייר בו הוספנו למודל שכבת אקטיבציה של relu ולבסוף softmax**. הקלסיפייר קיבל כקלט את הפיצ'רים שחילצנו לאחר שעשינו להם flatten והוציא כפלט פרדיקציה לאיזה מחלקה התמונה שייכת.
  במקרה זה : Train on 10132 samples, validate on 2533 samples
  התוצאות שהתקבלו :

loss: 0.0274 - acc: 0.9897 - val_loss: 0.0247 - val_acc: 0.9909





תוצאות VGG16 + KNN:

- **Test Accuracy: 0.9913146466640348**

ניתן לראות כי בבדיקת הדיוק עבור הTest set הדיוק עומד על 99.13%.

- **Confusion Matrix :**

כמו כן ניתן ללמוד מהconfusion matrix כי עבור 2533 דוגמאות סכ"ה הפרדקיציה נכונה עבור 2511 דוגמאות. התוצאה מתקבלת על ידי סכימת איברי האלכסון המייצגים את מידת ההסכמה בין הפרדקציה לבין המחלקה האמיתית אליה שייכת הספרה בתמונה.
1353 מהדוגמאות הן של הספרה 0, כאשר הפרדיקציה שנתנו היא נכונה עבור 1333.
1200 מהדוגמאות הן של הספרה 1, כאשר הפרדיקציה שנתנו היא נכונה עבור 1178.

Part 4 -

Input: short video sequence + yolo lite output for a few classes

Output : your choice of something not learning based

שלב א' - שימוש בyolo

מחברת של colab שם בצענו זיהוי של אובייקטים בעזרת : yolo
https://colab.research.google.com/drive/1cmTd2u0hdGc--Oz31kq237C81Z76aHUk#scrollTo=nRHSWu5lebas

נתבונן בפלט של יולו על קליפ מיוטיוב :
האלגוריתם זיהה 5 אובייקטים :

- בתמונה הראשונה : אדם , מכונית וכלב
- בתמונה השנייה אדם וכוס יין
- בתמונה השלישית אדם ועוגה.

ליד כל אובייקט מופיעה ההסתברות לכך שמדובר באובייקט זה והיא הגבוהה ביותר מתוך כלל המחלקות המוצעות.

פלט זה של האלגוריתם מדגים מצבים בהם הוא טועה, אך יש היגיון בטעות.

במקרה זה הוא זיהה את הכוסות כעוגה בהסתברות של 83%, עם זאת נבחין כי מבחינת צורה ומיקום הגיוני שיהיה מדובר בעוגה.

שלב ב' - שימוש בyolov3 על מנת לאכוף את הנחיות הריחוק החברתי

הקלט של התוכנית הוא קטע וידאו בו נראים אנשים שהולכים ברחובות פריז.
אנו עושים שימוש ב yolov3 על מנת לזהות אנשים במרחב באמצעות bounding box , לאחר מכן אנו בודקים האם יש יותר מאדם אחד בפריים, אם כן נבדוק מהו המרחק בינהם.
במידה והמרחק מנוגד למגבלות נסמן את box של אותם אנשים באדום, ואילו במידה והם פועלים על פי ההנחיות נסמן את הboxבירוק.

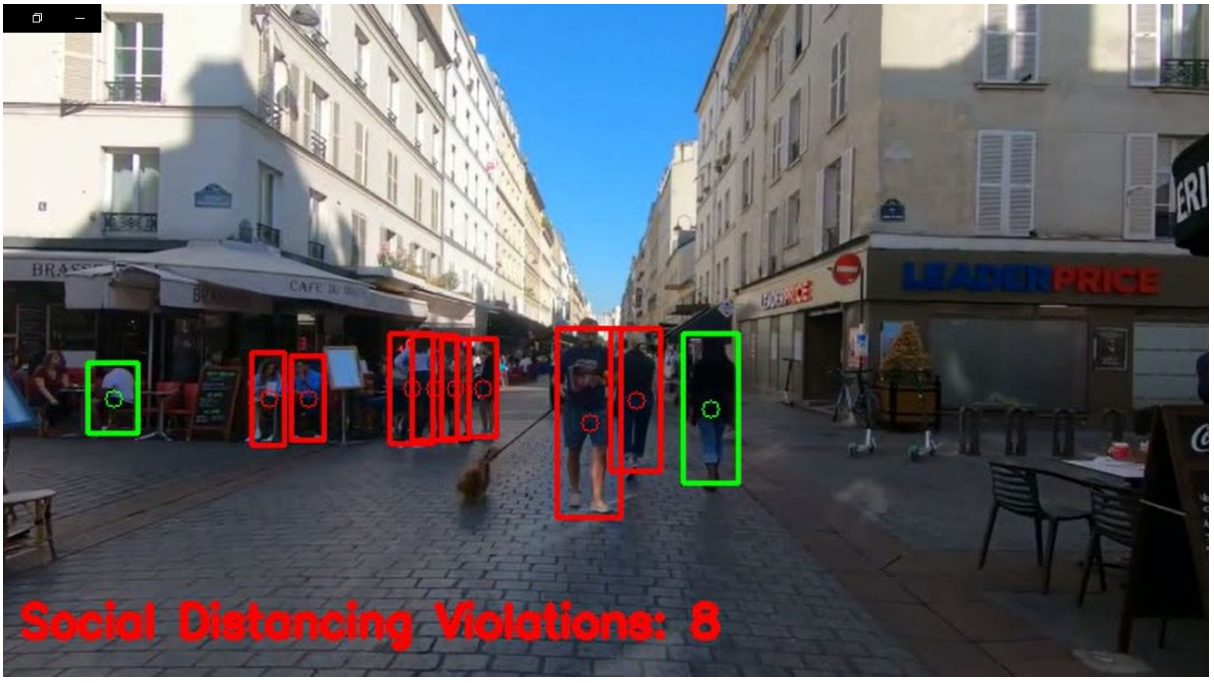
הפלט של התוכנית הוא קטע וידאו בו מסומנים האנשים ב boxes כאשר הצבע נקבע על פי שמירה על התנאים, כמו כן מופיע טקסט המתאר את מספר ההפרות בפריים.

איך זה עובד?

- זיהוי אוביקטים בעזרת YOLOV3 וספציפית זיהוי אנשים.
- נוודא שיש לפחות שני אנשים שזוהו על מנת לחשב מרחקים.
- נשים לב כי YOLO מחזיר גם את הקורדינטות של המרכזים של הbounding box. נשתמש בהם כדי לחשב מרחק אוקלידי בין כל זוג של מרכזים.
- אם המרחק שחושב קטן מהמרחק שהוגדר כמרחק המינימלי המותר נכניס למערך הפרות שיש אותנו כדי להדפיס למסך את מס ההפרות וכמו כן את מיקום ההפרה כדי לסמן את הbounding box בצבע המתאים.

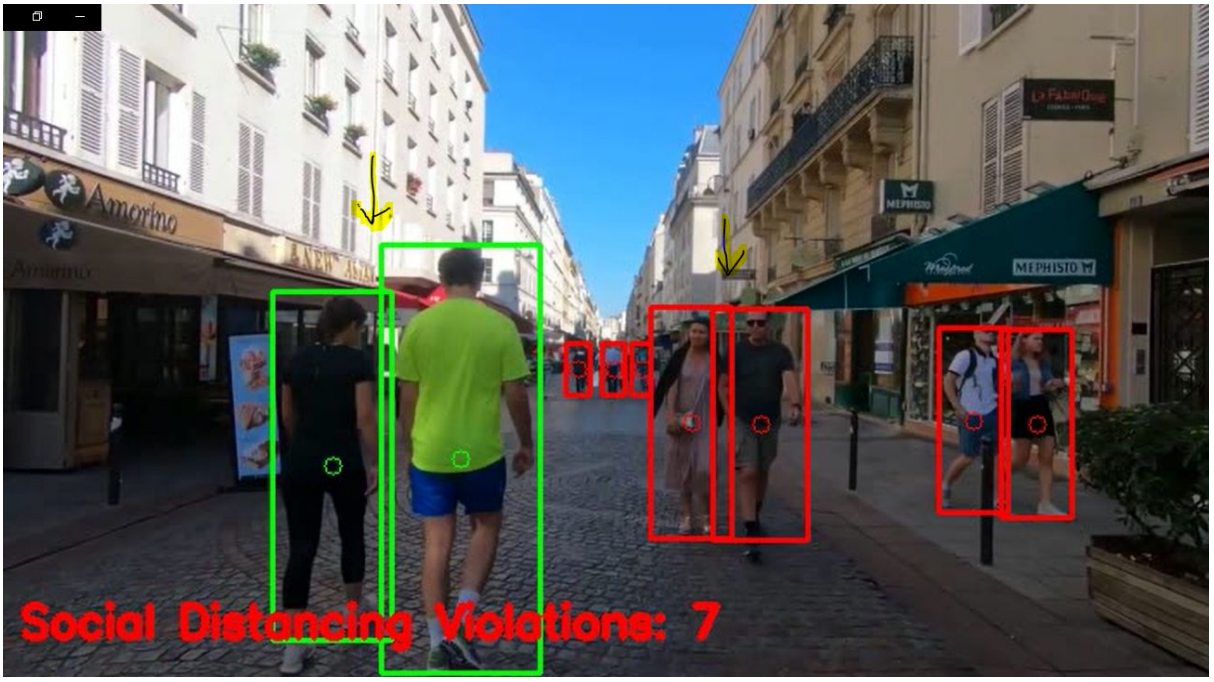
כיצד נראה פלט של התוכנית :

1. חתכנו את הוידאו הבא : https://www.youtube.com/watch?v=osf8KSaGiv0
והכנסנו כקלט לתוכנית.
צילום מסך מתוך הפלט:



2. הקלט - https://www.youtube.com/watch?v=pk96gqasGBQ
צילום מסך מתוך הפלט:

נדגיש כי הקבוע המתאר את המרחק המינימלי המותר הוא עבור **מרחק בין פיקסלים** ולכן במקרים בהם המצלמה קולטת אנשים במרחקים שונים ממנה נקבל טעויות.
עבור אנשים הנמצאים קרוב למצלמה נצטרך לתת ערך קטן יותר בתור המרחק המינמלי המותר, ועבור אנשים שנמצאים רחוק מהמצלמה נצטרך לתת ערך גדול יותר.
בתמונה הבאה ניתן לראות כיצד מזהים זוג שהולך יד ביד כהפרה של ריחוק חברתי כנדרש, אך זוג נוסף שקרוב יותר למצלמה מזוהה כאילו המרחק תקין.



<u>כיצד נשפר זאת?</u>

אנו מסתמכים על מרחק בין פיקסלים על מנת לקבוע הפרה של ריחוק חברתי, שאינו מספיק מדוייק ובמקרים מסויימים נותן תוצאות שגויות.
נצטרך למעשה לשנות את הערך הקבוע המתאר את המרחק המינימלי המותר בהתאם למיקום האובייקט בתמונה ביחס למצלמה, ולהתאים אותו למרחק במציאות.

1. עבור תוצאות מדוייקות יותר נוכל לבצע **calibaration** למצלמה עם פרמטרים שהם intrinsic/extrinsic, באופן זה נוכל למפות פיקסלים בתמונה ליחידות מידה שמתאימות למרחק 2 מטר.

Camera calibration is the process of estimating intrinsic and/or extrinsic parameters. Intrinsic parameters deal with the camera's internal characteristics, such as its focal length, skew, distortion, and image center. Extrinsic parameters describe their position and orientation in the world. Knowing intrinsic parameters is an essential first step for 3D computer vision, as it allows you to estimate the scene's structure in Euclidean space and removes lens distortion, which degrades accuracy.
we first need to compute the “pixels-per-metric” ratio, used to determine how many pixels “fit” into a given unit of measurement.
Once we have this ratio, computing the distance between objects is almost trivially easy.

2. דרך נוספת שתניב תוצאות פחות מדוייקות הינה - **triangle similarity calibration**.