מעבדה בניתוח והצגת נתונים

תרגיל בית 2

מגישים:

משה עבאדי 324658939

רועי רימר 314828732

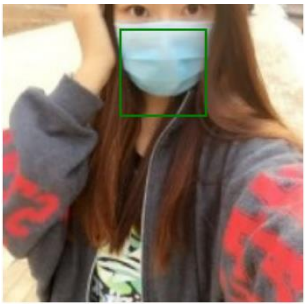
קישור ל-Repository:
https://github.com/moshinhoabadi/Data-Analysis-hw2

Exploratory Data Analysis

א. Visualization of some images

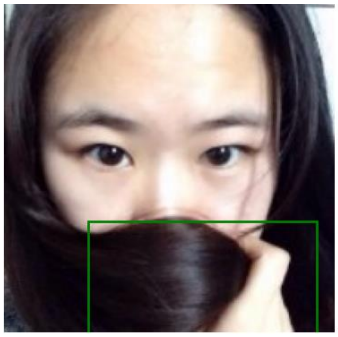
נציג כמה תמונות מן סט הנתונים, שנראה כי תויגו בצורה נכונה:

<u>תמונות שתויגו כ-True:</u>



נראה כי בתמונות אלו האדם השמאלי ביותר שפניו מכוסות לובש מסכה תקנית בצורה שנחשבת לנכונה.

<u>תמונות שתויגו כ-False:</u>

בתמונות אלו, האדם השמאלי ביותר שפניו מכוסות איננו עוטה מסיכה תקנית.

ב. Insights from simply "looking" at the data

נראה כעת כמה תגליות חריגות ומעניינות שמצאנו מסקירה של התמונות בסט הנתונים:

- מצאנו בסט הנתונים תמונות שנראה כי ה-bounding box שלהן ב-ground truth איננו מתייחס לדעתנו לאדם השמאלי ביותר. להלן כמה דוגמאות:

המסגרת מתייחסת לאדם המרכז, אך גם לאדם משמאל יש פנים מכוסות

לאדם עם הג'קט הכחול (מקדימה) יש מסיכה עליו, אך המסגרת מתייחסת לאדם אחר.

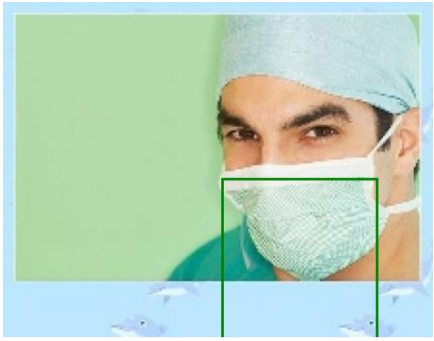
המצלמה מכסה את הפנים של הבחורה משמאל, אך המסגרת מתייחסת לאדם הימני

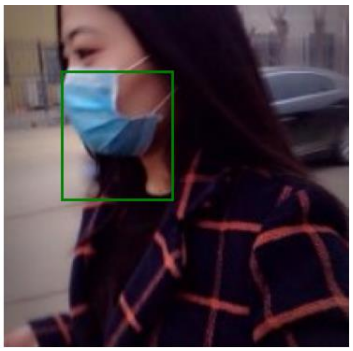
המסגרת מתייחסת לילד ולא לאישה, אף על פי שהיא לשמאלו

- בנוסף, זיהינו כמה תמונות שתויגו כ-True, אך לדעתנו זוהי טעות. לדוגמא:

בתמונות שלעיל, כיסויי הפנים שלובשות הנשים לא נראים כמו מסיכה רגילה תקנית, אלא יותר כמין כיסוי שמשמש למטרות נוי. 3 מתוך התמונות מכילות כיסוי שקוף, כך שגם לא נראה שאלו מסכות תקניות האטומות לחלקיקים.

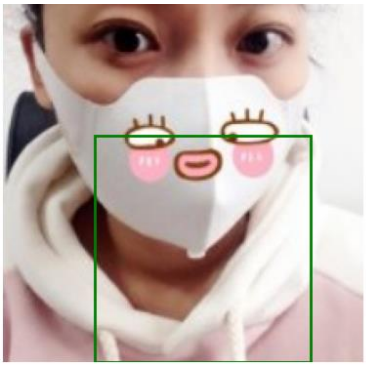
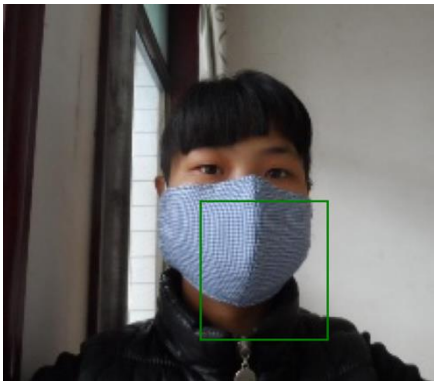
- מצאנו גם תמונות שתויגו כ-False, אך לדעתנו זוהי טעות. לדוגמא:

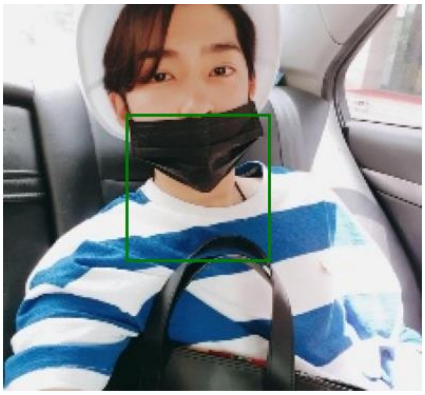


לדעתנו, האנשים בתמונות אלו חובשים מסיכה חלקה תקנית בצורה שמכסה את הפנים כנדרש. אף על פי כן, בסט הנתונים תמונות אלו סווגו כ-False.

- ישנן תמונות בהן נראה כי המסגרת איננה מקיפה את המסיכה בצורה טובה. לדוגמא:



בשתי התמונות העליונות, המסגרת סוטה לדעתנו מן המיקום של המסיכה בצורה משמעותית. בשתי התמונות התחתונות, המסגרת גדולה ורחבה מדי ותופסת שטח גדול יותר מאשר שטח המסיכה בתמונה.

Experiments

<u>גישה ראשונה – מודיפיקציה של Resnet</u>

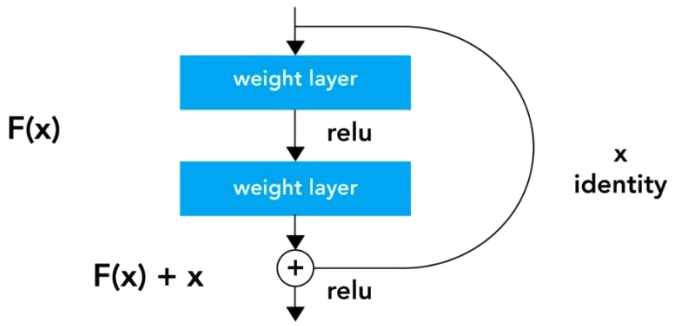
א. Data loading, pre-processing and cleaning

במהלך בניית ה-Dataset, בחרנו לעשות padding לכל התמונות כך שיהיו כולן בעלות שטח של 224X224. בכדי שהתהליך לא יגרום לשינוי בפיקסלים המתאימים ל-bounding box, בחרנו לעשות את ה-padding כך שהפיקסלים (עם ערך 0) שיתווספו בתהליך יהיו מימינה ומתחת לתמונה המקורית. מכיוון שספירת הפיקסלים מתחילה מהפינה השמאלית העליונה, לא יהיה שינוי במיקום היחסי של ה-bounding box.
מלבד זאת, בחרנו גם לנרמל את התמונות ביחס לממוצע ולסטיית התקן שחושבו על פני סט האימון.

ב. Architecture

הארכיטקטורה שבנינו מבוססת על זו שבקישור הבא:
https://towardsdatascience.com/bounding-box-prediction-from-scratch-using-pytorch-a8525da51ddc
את הקלט העברנו דרך רשת Resnet עם 50 שכבות. מהשכבה האחרונה חיברנו את ה-Resnet לשתי שכבות Fully Connected נפרדות: האחת אחראית על חיזוי הערכים הרלוונטיים לחיזוי ה-bounding box, והשנייה חוזה את התיוג של התמונה.

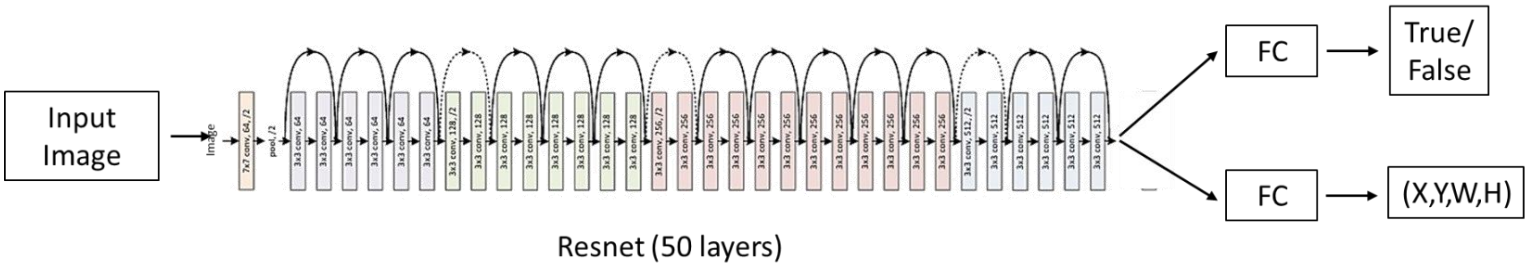
<u>הסבר קצר לגבי Resnet</u>: זוהי רשת המורכבת מ-Residual Blocks, כאשר בלוקים אלו מורכבים משני מעברים לקלט: אחד דרך שתי שכבות קונבולוציה, ואחד דרך פונקציית הזהות. הפלט של הבלוק הוא הפעלת פונקציית האקטיבציה ReLU על חיבור הפלטים של שני המעברים. איור 1 מציג סכמה של Residual Block:



איור 1 – סכמה של Residual Block.

השימוש ב-Residual Blocks מסייע לרשת להתמודד עם בעיית ה-Vanishing Gradients, שכן הגרדיאנטים יכולים לפעפע גם דרך ה"מסלול" של פונקציית הזהות, שהגרדיאנט שלה הוא 1. בנוסף, מבנה זה גורם לכך שניתן להעמיק את הרשת מבלי להסתכן בפגיעה בביצועים, מכיוון שהרשת תוכל ללמוד במקרה כזה לאפס את המשקולות בשכבות החדשות, ולגרום לכך שהקלט יעבור רק דרך פונקציית הזהות.

איור 2 מציג סכמה של ארכיטקטורת המודל בגישה הראשונה:



איור 2 – ארכיטקטורת המודל בגישה הראשונה.

ג. Loss Functions

פונקציית ה-loss בה השתמשנו היא סכום של שתי פונקציות loss, אחת המיועדת לשגיאת classification (בשביל הסיווג הבינארי של התמונה) ואחת נוספת המתאימה יותר לרגרסיה (בשביל הערכים המוחזרים עבור ה-bounding box).
השתמשנו ב-Cross Entropy Loss בשביל שגיאת ה-classification, ו-SmoothL1 Loss (ראו משוואה) עבור הערכים שהוחזרו בשביל ה-bounding box.

$$SmoothL1(y,\hat{y}) = \begin{cases} 0.5 * (y-\hat{y})^2, & |y-\hat{y}| < 1 \\ |y-\hat{y}| - 0.5, & otherwise \end{cases}$$

ד. Optimizers

ה-optimizer בו השתמשנו הוא Adam עם $learning\ rate = 0.0001$. נציין גם כי גודל ה-batch שבחרנו הוא 32. גודל batch גדול מזה גרם לחריגה מן זיכרון המכונה, כנראה משום שמודל Resnet 50 הוא כבד יחסית.

ה. Regularization

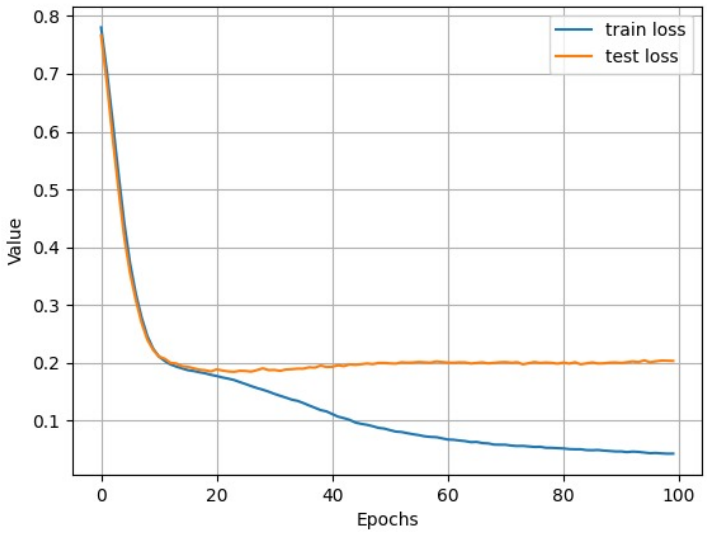
לא הוספנו רגולריזציה למודל. עם זאת, נזכיר כי Resnet עוזר להתמודד עם מקרים של overfitting הנובעים מרשת עמוקה מדי בכך שהוא יכול ללמוד לאפס משקולות בשכבות מיותרות, ובמקום זה להעביר את הקלט דרך פונקציית הזהות בלבד.
ייתכן שרגולריזציה נוספת הייתה משפרת את ביצועי המודל, אך לא ביצענו ניסויים רבים עם גישה זו מכיוון שהעדפנו לנסות ולהתמקד במימוש הגישה השנייה (שתוצג בהמשך).

ו. Hyper parameter tuning

ביצענו ניסויים עם ערכים שונים של ה-learning rate. הערכים שניסינו הם 0.001, 0.0001, 0.00001.
בנוסף, ניסינו גם לשנות את ערך ה-batch size (לצורך כך השתמשנו במודל Resnet עם 18 שכבות במקום 50). הערכים שניסינו הם 32,64,128,256.
הערכים בהם בחרנו להשתמש בסוף הם $learning\ rate = 0.0001$, ו-$batch\ size =$ 32, בעיקר כי גודל batch זה אפשר לנו להשתמש במודל Resnet בעל יותר שכבות.
מספר ה-Epochs שבחרנו לאמן את המודל הוא 20. כפי שנראה בהמשך (איורים 3,4,5), זהו השלב באימון בו המודל נתן את הביצועים הכי טובים על סט המבחן והיה לו loss יחסית נמוך. כאשר המשכנו לאמן את המודל, התחיל תהליך של overfitting וביצועי המודל נפגמו.

התוצאות המיטביות שלנו על סט המבחן:
accuracy: 0.607
IoU: 0.451

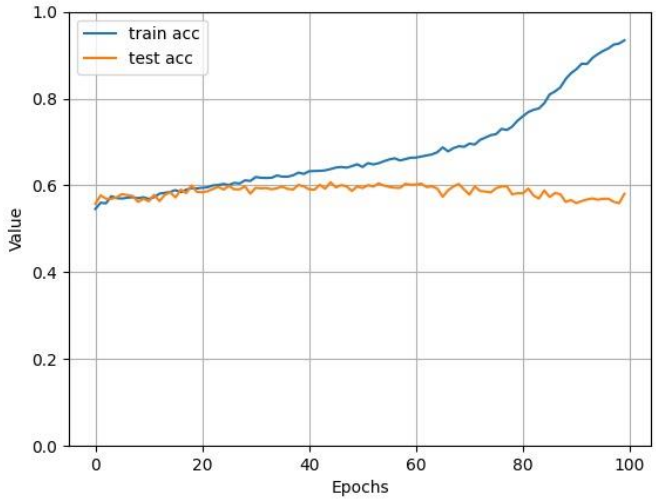
ז. Train + Test loss graph
איור 3 מציג גרף של ה-loss על האימון והמבחן של המודל כתלות במספר ה-epochs:



איור 3 – loss על האימון והמבחן של המודל הראשון כתלות במספר ה-epochs.

ניתן לראות כי יש ירידה בשגיאת המבחן עד ה-epoch ה-20 בערך. שגיאת האימון ממשיכה לרדת, אך בשלב זה נראה כי המודל נמצא כבר בשלב של overfitting ולכן בחרנו בהמשך לאמן את המודל למשך 20 epochs בלבד.

ח. Train + Test average accuracy graph
איור 4 מציג גרף של ה-accuracy של המודל על סט אימון והמבחן כתלות במספר ה-epochs:

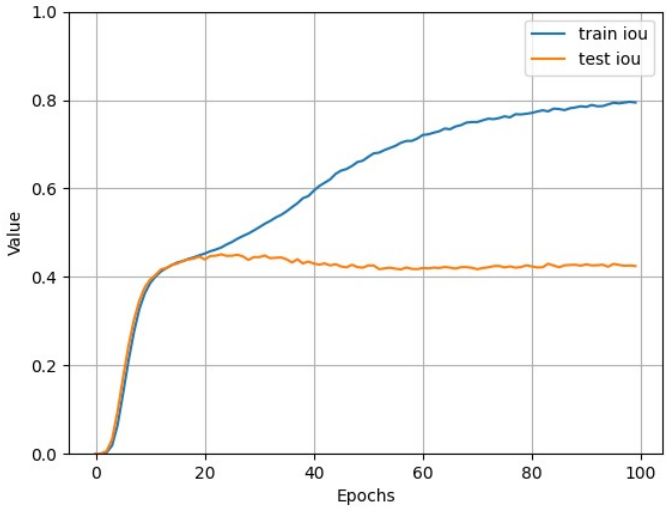


איור 4 - ה-accuracy של המודל הראשון על סט האימון והמבחן כתלות במספר ה-epochs.

בדומה למתרחש באיור 3, גם כאן ניתן לראות שיפור בדיוק על סט המבחן עד ה-epoch ה-20 בערך, בעוד הדיוק הממוצע על סט האימון ממשיך לעלות גם לאחר מכן. זוהי עדות נוספת לכך שנראה כי כדאי לעצור את האימון לאחר ה-epoch ה-20.

ט. Train + Test average IoU graph

איור 5 מציג גרף של ה-IoU של המודל על סט האימון והמבחן כתלות במספר ה-epochs:



איור 5 - ה-IoU של המודל הראשון על סט האימון והמבחן כתלות במספר ה-epochs.

בדומה לאיורים 3 ו-4, גם כאן ניתן לראות שהביצועים המיטביים על סט המבחן מתקבלים בערך לאחר ה-epoch ה-20. לאחר מכן השיפור מתרחש רק על סט האימון (overfitting).

<u>גישה שנייה – Faster R-CNN (זו הגישה בה בחרנו להשתמש בהגשה הסופית)</u>

א. Data loading, pre-processing and cleaning

מכיוון שמודל ה-Faster R-CNN יודע לטפל בתמונות מגדלים שונים ולבצע נרמול של התמונות בתוך המודל, לא נרמלנו את התמונות או ביצענו padding (בניגוד לגישה הראשונה).

עם זאת, לכל תמונה בסט האימון ביצענו בהסתברות של 0.5 Random Horizontal Flip, כלומר הפכנו את התמונה לתמונת מראה של התמונה המקורית. במקרים כאלו הפכנו גם את ה-bounding box בהתאם.

בנוסף, ביצענו Random crop לתמונות בסט האימון תוך כדי תיקון ה-bounding box בהתאם לפלט של ה-crop.

עיבוד נוסף שעשינו לסט האימון הוא הוספת עיוות (Random Distortion) לתמונות, כאשר לכל תמונה ביצענו זאת בהסתברות של 0.5.

לסיום, שינינו את פורמט ה-ground truth של ערכי ה-bounding box מפורמט של קואורדינטות הפינה השמאלית העליונה, אורך התמונה, רוחב התמונה (x,y,w,h) לפורמט של קואורדינטות הפינה השמאלית העליונה והימנית התחתונה של התמונה (x1,y1,x2,y2), וזאת כדי להתאים את הערכים האלו לקלט שמצפה המודל לקבל.

ב. Architecture

הגישה שלנו מבוססת על מודל Faster R-CNN המשתמש במודל Resnet 50 בתור ה-backbone.

המאמר על Faster R-CNN: https://arxiv.org/pdf/1506.01497.pdf

הקוד והמימוש שלנו מבוססים על:

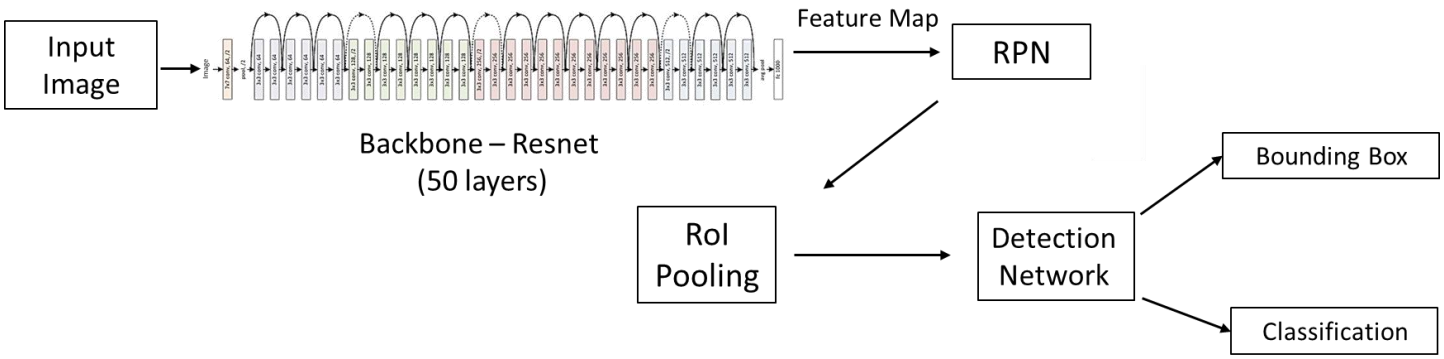
a. המימוש ל-Faster R-CNN בספריית Torchvision.Models:
https://pytorch.org/vision/stable/_modules/torchvision/models/detection/faster_rcnn.html

b. מחברת Tutorial מתוך אתר pytorch למשימות Object Detection המשתמשת במודל Faster R-CNN:
https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

<u>הסבר קצר לגבי Faster R-CNN</u>: זוהי גרסה משופרת של המודל Fast R-CNN, שמהווה בעצמו גרסה משופרת של המודל R-CNN. Faster R-CNN מורכב מכמה רכיבים:

- Backbone - זוהי רשת המקבלת את התמונה כקלט, ומחזירה feature map שמיועד לייצג את הקלט. אנו בחרנו להשתמש במודל Resnet 50 בתור ה-backbone.
- Region Proposal Network (RPN) – רכיב זה אחראי על יצירת אזורים ה"מועמדים" להיות אזורים בהם נמצאים אובייקטים בתמונה. ה-RPN מקבל כקלט את ה- feature map שמחזיר ה-backbone, ומתאים לכל feature בו תשעה anchors, כלומר מסגרות ובגדלים וביחסים שונים. ה-anchors נכנסים כקלט לתוך רשת קונבולוציה שמחזירה לכל anchor חיזוי ל-bounding box ולמחלקה של האובייקט. פלטים אלו מועברים לתוך אלגוריתם בשם Non Maximum Suppression שנועד לסנן מועמדים ל-bounding boxes על סמך החפיפה שיש (במונחי IoU) בין מועמדים שונים. הפלט מאלגוריתם זה מועבר לחלק הבא במודל.
- RoI Pooling – רכיב pooling ברשת שמקבל את ה-feature map של המועמדים שהתקבלו מן ה-RPN, מבצע clipping ל-feature map, מחלק את מה שנותר ל-grid של 7X7, ולכל חלק ב-grid מבצע max pooling בנפרד.
- Detection Network - רשת זו מורכבת משתי שכבות Fully Connected המחוברות לרשת FC נוספת שמיועדת להחזרת ה-bounding box, ולאחת נוספת ונפרדת שמשמשת להחזרת המחלקה של האובייקט. הרשת מקבלת כקלט את הפלט של ה- RoI Pooling.

איור 6 מציג סכמה של ארכיטקטורת המודל בגישה השנייה:



איור 6 - ארכיטקטורת המודל בגישה השנייה.

ג. Loss Functions

ה-Loss Function של המודל היא סכום של ה-Loss Functions של רכיב ה-RPN ושל ה-Detection Network. נסביר על כל אחת מהן בנפרד:

a. RPN – הפלט של ה-RPN הוא מועמדים ל-bounding box (anchors). המועמדים להם IoU גבוה ביחס ל-bounding box של ה-ground truth מקוטלגים כ-positive anchor, ומועמדים להם IoU נמוך ביחס ל-bounding box של ה-ground truth מקוטלגים כ-negative anchor. ה-loss מחושב על סמך כל ה-anchors הללו ועל סמך סכימה לכל anchor של פונקציית ה-loss הנ"ל (המובאת בגרסה עבור anchor יחיד):

$$L_{RPN}(d,p,d^*,p^*) = I_{\{d\ is\ positive\ anchor\}} * L_{reg}(d,d^*) + L_{cls}(p,p^*)$$

כאשר:

- d – הערכים שקובעים את גבולות ה-bounding box.
- p – ההסתברות שהמודל נתן למחלקה הנבחרת.
- d^* - ה-bounding box הנכון של האובייקט.
- p^* - המחלקה הנכונה של האובייקט.
- L_{reg} – פונקציית $Smooth\ L1\ Loss$.
- L_{cls} – פונקציית $Cross\ Entropy$.

b. Detection Network - ה-Loss Function זהה לזה של ה-RPN, למעט ההגדרה של ה-bounding boxes המוצעים (ה-anchors/proposals) כחיוביים או שליליים. במקרה של ה-RPN, anchor נחשב ל-positive אם יש לו IoU של 0.7 לפחות ו/או הוא ה-anchor בעל ה-IoU הגבוה ביותר ביחס ל- bounding box האמיתי. ההגדרה ל-negative anchor הייתה IoU של לכל היותר 0.3 ביחס ל- bounding box האמיתי.
כעת, ההגדרה ל-positive anchor היא IoU של 0.5 לפחות ביחס ל- bounding boxהאמיתי, וההגדרה ל- negative היא IoU של לכל היותר 0.1 ביחס ל- bounding box האמיתי.

ד. Optimizers

ה-optimizer בו השתמשנו הוא SGD עם $learning\ rate = 0.005$ ו-$momentum = 0.9$. נציין. גם כי גודל ה-batch שבחרנו הוא 2. בחרנו גודל batch זה כדי לא לחרוג מזיכרון הGPU הקיים במכונה.

ה. Regularization

כפי שרשמנו לעיל, Resnet עוזר להתמודד עם מקרים של overfitting הנובע מרשת עמוקה מדי. בנוסף, הטרנספורמציות שעשינו לתמונות בסט האימון עוזרים גם הם למנוע overfitting ולאפשר למודל להיות מוכלל בצורה יותר טובה, בכך שהם גורמים לשינויים בקלט שמקבל המודל בכל epoch של האימון. מעבר לכך לא הוספנו רכיבים שמסייעים לרגולריזציה של המודל.

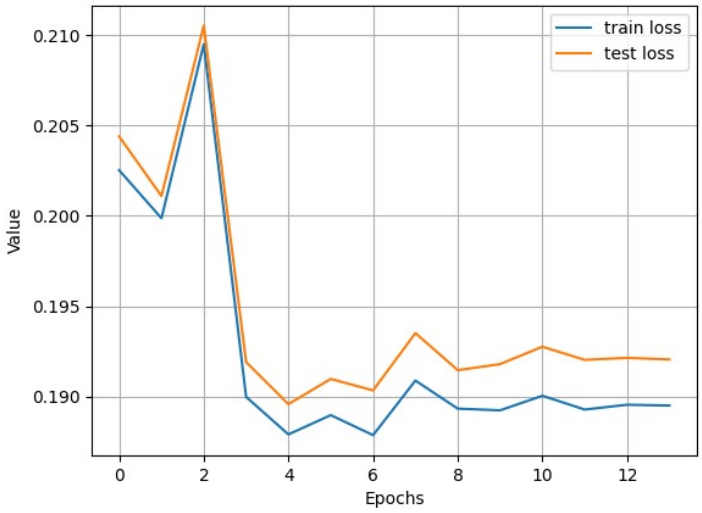
ו. Hyper Parameter Tuning

מפאת קוצר זמן, לא הספקנו לבצע ניסויים עם ערכים שונים של היפרפרמטרים כגון ה-learning rate.
ה-learning rate בו השתמשנו הוא 0.005.

גודל ה-batch שבחרנו הוא 2.
מספר ה-Epochs שבחרנו לאמן את המודל הוא 10. כפי שנראה בהמשך (איורים 7,8,9), זהו השלב באימון בו השיפורים בביצועי המודל עם כל epoch חדש החלו להיות זניחים. לכן, וכדי להימנע מאפשרות של overfitting, בחרנו לעצור את האימון בשלב זה.

התוצאות המיטביות שלנו על סט המבחן:
accuracy: 0.742
IoU: 0.555

ז. Train + Test loss graph
איור 7 מציג גרף של שגיאות האימון והמבחן של המודל כתלות במספר ה-epochs:

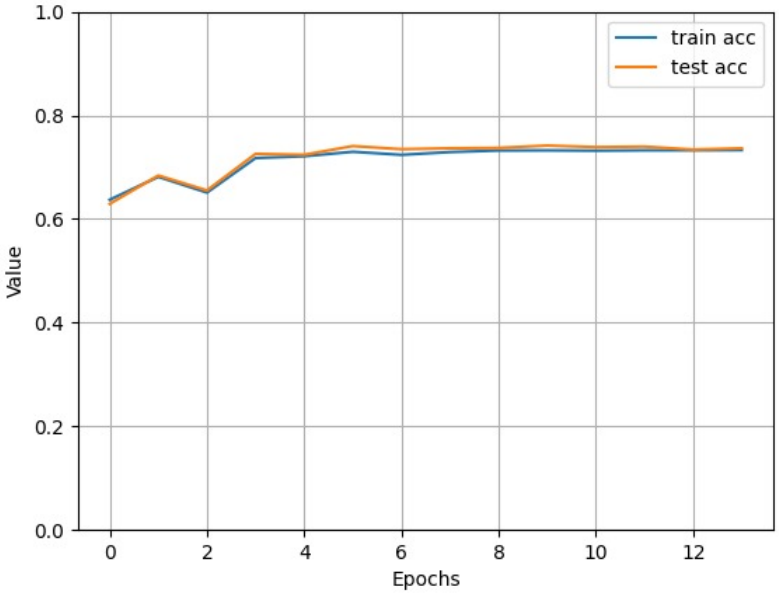


איור 7 – loss על האימון והמבחן של המודל השני כתלות במספר ה-epochs.

ניתן לראות כי ה-loss מעט "מזגזג" בערכים שלו בין ה-epochs השונים, אך המגמה הכללית היא של ירידה, בעיקר ב-epochs הראשונים. לאחר כ-10 epochs, ערך ה-loss מתייצב, ולכן בחרנו להשתמש במודל שהתאמן רק עד שלב זה.

ח. Train + Test average accuracy graph

איור 8 מציג את ה-accuracy על סט האימון והמבחן כתלות במספר ה-epochs:

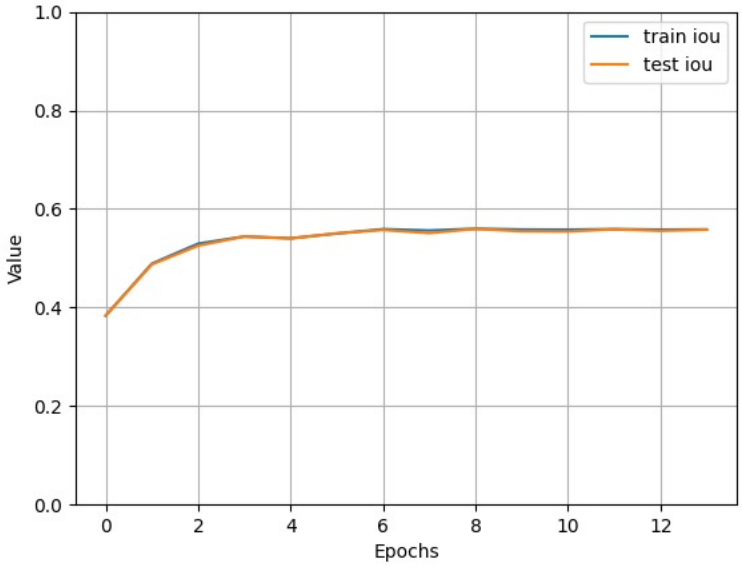


איור 8 – ה-accuracy של המודל השני על סט האימון והמבחן כתלות במספר ה-epochs.

ב-epochs הראשונים ניתן לראות עלייה יחסית חדה ב-accuracy. לאחר מכן, מגמת העלייה מתמתנת אך ממשיכה להתקיים עד אזור ה-epoch העשירי. זוהי סיבה נוספת לבחירה שלנו במודל שהתאמן עשרה epochs.

ט. Train + Test average IoU graph

איור 8 מציג את ה-IoU על סט האימון והמבחן כתלות במספר ה-epochs:



איור 9 - ה-IoU של המודל השני על סט האימון והמבחן כתלות במספר ה-epochs.

בדומה למתרחש עבור accuracy (איור 8), גם כאן ישנה מגמת עלייה שמתמתנת עם הזמן. באזור ה-epoch העשירי השיפור ב-IoU הופך זניח, כך שנראה כי ניתן לעצור את האימון בשלב זה.

Conclusions

נציג כמה מסקנות שעלו מהסקירה של סט הנתונים, ומהעבודה על שתי הגישות שניסינו כדי להתמודד עם המשימה שבתרגיל הבית:

- במהלך המעבר שלנו על תמונות מתוך סט הנתונים, הבחנו שנראה כי חלק מהתמונות אינן מסווגות למחלקה הנכונה, אינן מתייחסות לאדם המכוסה השמאלי ביותר בתמונה, או שה-bounding box שלהן איננו תואם מספיק את כיסוי הפנים הרלוונטי. ייתכן והייתה השפעה לכך על ביצועי המודל, ולכן אנו חושבים ש- ground truth אמין יותר עשוי היה להוביל לשיפור בביצועי המודל שבנינו.
- בבדיקות שעשינו, נראה כי שתי הגישות שניסינו הניבו ביצועים טובים יותר ונטו פחות ל-overfitting כאשר ביצענו transformations (הוספת עיוות, נרמול ועוד) לתמונות שהמודל מקבל כקלט בשלב האימון. נראה כי במשימה זו יש חשיבות ל-pre-processing של התמונות, גם מבחינת ביצועים וגם מבחינת יכולת ההכללה של המודל.
- הגישה השנייה שניסינו הניבה תוצאות טובות יותר מאשר הגישה הראשונה. נסיק מכך על היתרון המשמעותי שיש ל-Faster R-CNN לעומת גישה "נאיבית" יותר לפתרון המשימה.

נעיר בנוסף שניסינו מתוך עניין לבצע את המשימה עם מודל Faster R-CNN שכן אומן מראש, והתקבלו תוצאות הרבה יותר טובות מאלו שמוצגות במסמך זה, הן מבחינת accuracy והן מבחינת IoU. נראה אם כן שלמודל זה יש הרבה כוח בהתמודדות עם משימות של Object Detection.