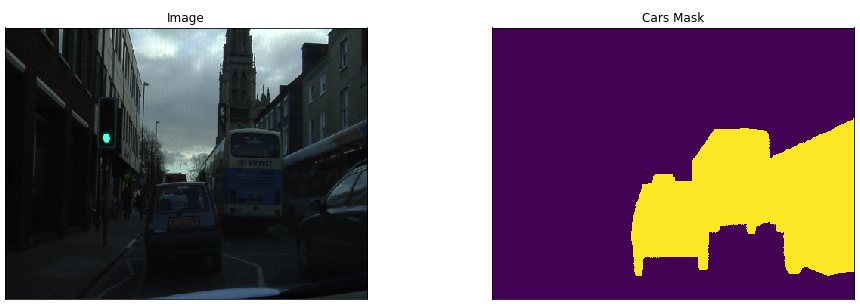
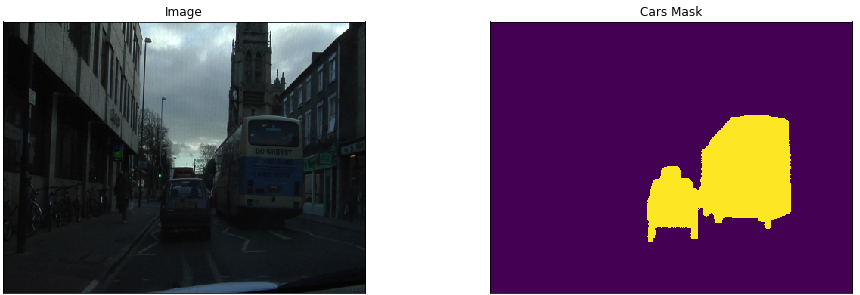
Digital Image Processing – Final Project

Aviad Shurgi 311125371 Avraham Raviv 204355390

### Introduction

בעבודה זו בחרנו להתמודד עם זיהוי אובייקטים בתוך תמונה (Semantic Segmentation) והדאטא שבחרנו לקוח מתוך CamVid – מאגר תמונות של אוניברסיטת קיימברידג'. מטרת המאגר הינה לקחת סרטוני וידאו קצרים ולהפוך אותם לסטים בדידים של תמונות המהוות אירוע רציף המתועד כסט של תמונות במקום כוידאו. מתוך מאגר התמונות הזה, בחרנו להתמקד ברכבים, ויותר במדויק – בזיהוי רכבים מתוך תמונה נתונה. לשם המחשה, נציג מספר תמונות שנלקחו מתוך מקטע וידאו, ואת ה-masking שמנתח את הימצאות כלי התחבורה בתמונה:







איור 1. המחשה של מכוניות בתנועה וה-mask שלהם.

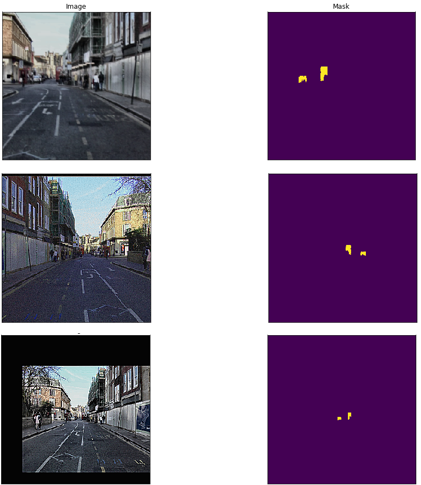
ניתן לראות כיצד מתוך סרט וידאו של חציית צומת מרומזר נלקחו מספר פריימים, והתבצע ניתוח מיקום של האובייקטים שהינם כלי תחבורה.

### Augmentation and perlin noise

גודל סט האימון שלנו הוא 367 תמונות. זהו מאגר יחסית לא גדול. כדי להגדיל ולגוון את מבחר התמונות, השתמשנו בספריה albumentations המאפשרת לקחת תמונה ולעשות עליה כל מיני וריאציות. הרחבה על מגוון הפיצ'רים של ספריה זו ניתן למצוא כאן:

<https://github.com/albumentations-team/albumentations>

אם מבצעים 9 פרמוטציות לכל תמונה, סה"כ מקבלים 10 וריאציות עבור כל תמונה, והמאגר גדל פי עשרה, וזהו גודל כבר יותר סטנדרטי שאפשר לעבוד איתו יותר בנוחות. ניתן דוגמא להמחשה על פרמוטציות שניתן לבצע לתמונה:



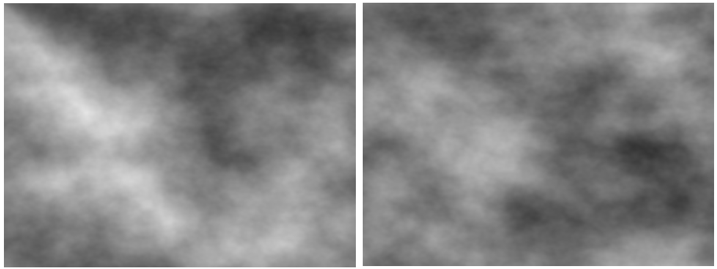
איור 2. ביצוע פרמוטציות לתמונה נתונה.

בשורה העליונה ניתן לראות תמונה סטנדרטית עם ה-mask שלה, ובשורות הנוספות ניתן לראות כיצד יצרנו עותקים שלה עם שינויים מסוימים, במקרה זה שיקוף והקטנה, וכמובן שגם ה-mask עבר את אותו שינוי. בכך אפשרנו הגדלה של סט התמונות שלנו פי עשרה (ואפשר אפילו יותר, כמספר השינויים שאנו מאפשרים. בספריה המקורית יש כ-30 אפשרויות).

את הרעש הדרוש, Perlin noise, חוללנו על בסיס הקוד המופיע כאן:

https://stackoverflow.com/questions/60350598/perlin-noise-in-pythons-noise-library

למעשה כתבנו שתי פונקציות – אחת המחוללת את הרעש הזה ושניה הלוקחת תמונה מתוך מאגר התמונות שלנו ובעזרת הפונקציה הראשונה מוסיפה לתמונה את הרעש. כתבנו שתי פונקציות ולא אחת בשביל שגם יהיה נח וגם יהיה ברור – מצד אחד יש פונקציה שכל ייעודה הוא לחולל את הרעש, ומצד שני יש פונקציה שמקבלת כקלט תמונה ומוסיפה לה רעש, בלי שהמשתמש יצטרך להתעסק בעצמו עם הפונקציה שמחוללת את הרעש. הקלט של הפונקציה שמחוללת את הרעש הוא תמונה כל שהיא, וזה נועד לכך שמימדי תמונת הרעש יהיו כמימדי התמונה לה אנו רוצים להוסיף את הרעש. שתי דוגמאות סטנדרטיות לרעש מהסוג הדרוש:



איור 3. דוגמאות ל-Perlin noise.

כפי שציינו, לפני שביצענו את תהליף הלמידה הוספנו אפשרות של augmentation, בעזרת הספריה שציינו לעיל. לאחר הוספת אפשרות זו, הוספנו לכל התמונות רעש, כאשר חוללנו לכל תמונה רעש אחר, כפי הנדרש. נראה שתי דוגמאות לתמונות לאחר תוספת הרעש:



איור 4. תמונות מהמאגר לאחר הוספת הרעש.

שיחקנו עם הפרמטרים של הרעש, ומצאנו שני דברים עיקריים: א. הפרמטרים שלבסוף בחרנו מייצרים רעש שבאמת נראה כשכבת אבק ועננים, כפי הדרוש. ב. הפרמטרים שבחרנו טובים מבחינת השילוב בין רעש אמיתי לבין הרצון שהתמונה עדיין תהיינה ניתנת לזיהוי בעיניים אנושיות.

התמונות שלנו צבעוניות, והדרך שבחרנו לשלב בין הרעש לבין התמונה הינה לשים את הרעש כשכבה נוספת מעל התמונה המקורית (ולא למשל לעדכן את הפיקסלים לשכבה יחידה המשלבת בין הרעש לבין התמונה כמו שהתנסינו בתרגיל 3). באופן הזה הן הרעש והן התמונה המקורית באו לידי ביטוי בצורה מיטבית.

### Train and validation noising datasets

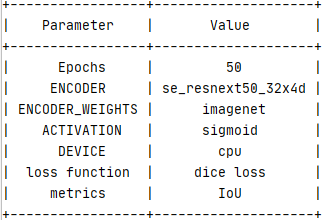
בעבודה שלנו ביצענו את תהליך האימון פעמיים – פעם אחת ללא pre-trained ופעם אחת עם (המחברת המצורפת כוללת את השלב הזה. בכדי לבטל אותו יש למחוק את קטעי הקוד מספר 15 ו-16, המסננים את הרעש). כעת נציג את השלב הבסיסי והתוצאות ללא ה-pre-trained, ולאחר מכן נרחיב עליו ונשווה בין הביצועים.

בכדי לבצע את האימון, נעזרנו בדוגמא המופיע ב-Git של הספריה segmentation\_models.pytorch:

<https://github.com/qubvel/segmentation_models.pytorch>

כפי שניתן להבין מהשם של הספריה, השתמשנו ב-pytorch במהלך העבודה לצורך הקטעים של הלמידה.

המאגר כולל כבר חלוקה לסטים של אימון, ולידציה וטסט, כך שמהבחינה הזו לא היינו צריכים לחלק בעצמנו את הדאטא. פרמטרי הלמידה שלנו:



איור 5. הפרמטרים של הלמידה.

באופן סטנדרטי הרצנו 50 epochs, וכאשר הזנו את הפקודה:

train\_loader **=** DataLoader**(**train\_noise\_dataset**,** batch\_size**=**8**,** shuffle**=True)**

ממילא קיבלנו שהמשקולות מתעדכנות בכל 46 איטרציות . כיוון שמערכת ההפעלה שהרצנו עליה את הפרוייקט הינה ווינדווס ולא לינוקס, לא יכולנו לשנות את הפרמטר num\_workers בפקודה זו (ברירת המחדל היא 0, אבל אם מגדילים את הפרמטר ניתן לקבל חישוב מקבילי הרבה יותר מהיר ויתכן שגם מספק ביצועים יותר טובים, אך זה אפשרי רק בלינוקס). מסיבה דומה השתמשנו ב-cpu ולא ב-cuda (אין לנו מעבד של nvidia אלא core של intel).

כיוון שאנו עוסקים בתמונה, הפרמטר encoder\_weights כמובן יהיה imagenet, ובהתאמה איתחלנו את encoder שיתאים לזיהוי אובייקטים בתמונה.

בחרנו בפונקציית הפעלה סיגמואיד, המוגדרת על ידי: .

כעת לשני הפרמטרים שיעסיקו אותנו גם בהמשך – loss function and metrics. פונקציית ההפסד שבחרנו במקורה נועדה לבחון דמיון בין שתי דגימות, אך בשנים האחרונות היא נמצאת בשימוש רב בתחום של CV. פונקצייה זו טובה מאוד למקרה שלנו, כיוון שאנו רוצים להשוות בין תמונה לבין ה-mask שלה, וזה נעשה בצורה טובה בעזרת הקורלציה שמציעה ה-dice loss (ככל שערך הפונקציה יותר קטן, כך הקורלציה גדלה). כמובן שנצפה לירידה בערך פונקציית ההפסד ככל שנתקדם בתהליך הלמידה. לגבי הפרמטר השני – metrics – בחרנו להשתמש במבחן IoU. מבחן זה בא לבדוק עד כמה יש התאמה בין הגבולות של אובייקט שהוצעו בעזרת אלגוריתם הלמידה ובין הגבולות האמיתיים, ולכן מבחן זה מתאים למקרה אותו אנו רוצים לבחון. המחשה טובה לכך ניתן לראות באיור הבא:



איור 6. IoU – יחס בין גבולות משוערכים (אדום) לבין גבולות אמיתיים (ירוק).

באיור ניתן לראות דוגמא מספרית למדד IoU, כאשר ככל שיש יותר חפיפה בין המסגרת האדומה לירוקה, כך הציון יותר טוב. ואצלנו – ככל שנקבל ערך IoU גדול יותר, כך ההתאמה של ה-masking שנקבל הינה טובה יותר. המקורות מהם למדנו על שני פרמטרים אלו:

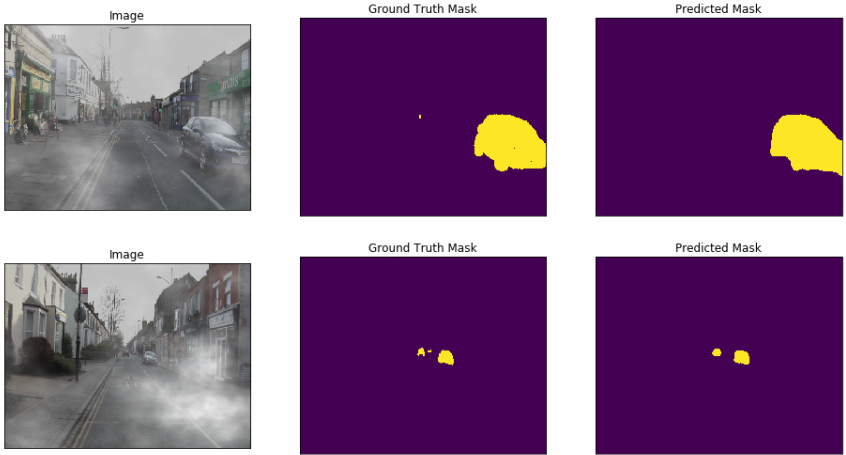
<https://medium.com/ai-salon/understanding-dice-loss-for-crisp-boundary-detection-bb30c2e5f62b>

<https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

### Results

#### A. visually

דבר ראשון ובסיסי, כדי לקבל תחושה על איכות התוצאות, הדפסנו עשר תמונות עם ה-mask האמיתי וה-mask המשוערך לאחר הלמידה. ניתן לראות שהתוצאות שהתקבלו טובות מאוד – איפה שיש אזורים של רעש חזק הפיקסלים לפעמים לא מזוהים ככלי תחבורה, אך במקומות עם רעש בינוני-חלש – הזיהוי מתבצע כראוי. אלו תוצאות מעודדות כיוון שבסופו של דבר למעט מקרים קיצוניים, לאחר הלמידה אנו מסוגלים לזהות את מיקום האובייקטים שהינם כלי תחבורה, גם אם ישנם פיקסלים מסויימים שלא מזוהים כראוי. נראה דוגמה להמחשה:

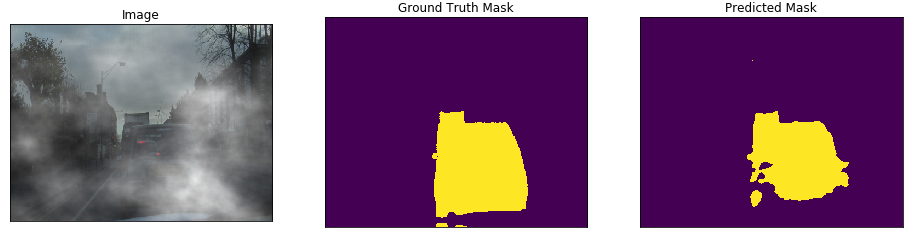


איור 7. השוואה בין mask אמיתי לבין mask משוערך.

בשורה העליונה ניתן לראות כי התמונה הימנית זהה כמעט לחלוטין לתמונה האמצעית, למרות שניתן לראות בתמונה השמאלית רעש מסוים באזור של כלי התחבורה. בשורה התחתונה הרעש שנוצר נראה הרבה יותר משמעותי, אך עדיין הזיהוי מתבצע בצורה איכותית כיוון שעיקר הרעש לא על כלי התחבורה עצמם אלא לידם ומשפיע עליהם באופן חלקי.

#### B. Improve

ישנם מקרים בהם פיקסלים רועשים לא מזוהים ככלי תחבורה, אך כל הסביבה שלהם כן תזוהה נכון, למשל:



איור 8. פיקסלים שאינם מזוהים נכון, בתוך סביבה המזוהה נכון ככלי תחבורה (תמונה 30 במאגר).

בתמונה הימנית ניתן לראות שהצורה של הרכב מזוהה נכון, למרות רעש כבד מאוד, אך ישנם אזורים קטנים סגולים בתוך כל המסגרת הצהובה. כדי להתגבר על שגיאות מסוג זה נוכל להציע שיפור על ידי הוספת postprocessing step – ניתן לשער שברוב המקרים אם יש פיקסלים בודדים בתוך mask שאינם מזוהים כחלק מה-mask, נוכל לשערך אותם להיות כמו ה-mask. ניתן לעשות זאת בבכמה דרכים:

1. שימוש בהתמרה מורפולוגית, כפי שלמדנו בקורס.
2. בעזרת אלגורתים k-means, המתקן פיסקלים בודדים ששונים משמעותית מסביבתם.
3. על ידי הוספת שכבה נוספת של cnn לתהליך הלמידה. השכבה בעצם בודקת תת-חלקים של התמונה החלקים, ומתאימה את הפיקסלים בהם לפיקסלים השכנים.

נרחיב על השיטות. (הערה: על שיטות ב,ג חשבנו לבד. הפניה לשיטה הראשונה קיבלנו בפייסבוק, בפוסט שפרסמנו בקבוצה [Machine and deep learning Israel](https://www.facebook.com/groups/MDLI1). הציעו לנו שם עוד גישות אך כיוון שכבר חשבנו בעצמנו על שתיים, הסתפקנו לקחת משם רק אחת, ובחרנו בכזו שלמדנו עליה בקורס).

1. התמרה מורפולוגית – יש כמה פונקציות מובנות בספריה opencv2. בחרנו לעשות closing על ידי שימוש בפונקציה [cv.morphologyEx](https://docs.opencv.org/trunk/d4/d86/group__imgproc__filter.html#ga67493776e3ad1a3df63883829375201f). קיבלנו את השיפור הבא:



איור 9. תיקון אזורים בתמונה שהתקבלה על לאחר תהליך הלמידה בעזרת התמרה מורפולוגית.

ניתן לראות כיצד ההתמרה שיפרה את התמונה, וכעת התמונה הסופית הרבה יותר דומה לתמונה המקורית, ובעיקר "נסתמו" בה החורים הבעייתיים שנבעו מהרעש. בחרנו להשתמש דווקא בהתמרה זו, שהרי: " It is useful in closing small holes inside the foreground objects, or small black points on the object." (מתוך תיעוד הספריה: <https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html>).

1. K-means. קראנו הרבה על שימוש באלגוריתם k-means לצורך segmentation, ומצאנו את המאמר הבא שימושי עבורנו: <https://towardsdatascience.com/introduction-to-image-segmentation-with-k-means-clustering-83fd0a9e2fc3>

בעזרת הקוד המופיע במאמר (המתבסס גם על [Canny Edge detection](https://en.wikipedia.org/wiki/Canny_edge_detector)), לקחנו את התמונה המשוערכת וחיפשנו אזורים בהם יש פיקסלים השונים מהותית מסביבתם. קיבלנו אחרי הרצת הקוד:



איור 10. זיהוי אזורים בתמונה שהתקבלה על לאחר תהליך הלמידה.

בתמונה ניתן לראות שני איזורים קטנים יחסית הבלועים בתוך אזור בצבע אחר. ניתן להוסיף לאחר תהליך הלמידה לולאה הלוקחת את האיזורים הבלועים, ואם הם קטנים מגודל מסוים היכול להבחר על ידינו, הם יצבעו בצבע השני. בדוגמא שלנו, הנקודה ליד החץ העליון תצבע בסגול, שזהו הצבע של הסביבה שלה, ואילו האזור המסומן על ידי החץ השני יחליף צבע לצהוב ויזוהה כחלק מה-mask. בדרך זו נוכל לשפר את זיהוי האובייקט למרות הרעש.

1. Cnn – השיטה השלישית שהצענו לשיפור הינה להוסיף שכבת קונבולוציה, הבוחנת תת-תמונות מתוך התמונה הכללית. לשם המחשת התוצאה נעזר באתר אינטראקטיבי המאפשר לחזות בכל שלבי רשת הקונבולוציה: <https://poloclub.github.io/cnn-explainer/>.

הזנו את התמונה שלנו (מאיור 8, עם ה-mask המשוערך), ולאחר מספר שכבות קיבלנו את הצורה הבאה:

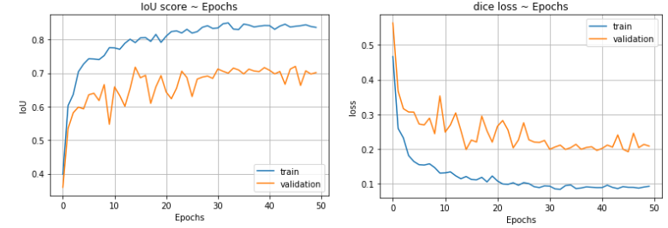


איור 11. Mask לאחר העברת התמונה המשוערכת ברשת קונבולוציה.

ניתן לראות כי כעת שני האזורים הבעייתיים שהצבענו עליהם קודם "נבלעו" בסביבה שלהם ונקבל mask קרוב מאוד לזה האמיתי. (עבור השיטות שהשתמשנו בהן בקוד פייתון – מצורף קובץ פייתון בשם detect\_wrong\_areas.py, ותמונה בשם im\_to\_fix).

#### C. Graphically

כעת נתבונן במספרים שיצאו לנו, שהדפסנו אותם כגרפים. הדפסנו בגרף אחד את העקומות של פונקציית ההפסד עבור סט האימון וסט הולידציה (מימין), ובגרף נוסף הדפסנו את העקומות של ה-IoU עבור אותם סטים:



איור 12. גרפים של dice loss ו-IoU כפונקצייה של ה-epochs.

המגמות נראות בהחלט טובות – פונקציית ההפסד הולכת וקטנה, ולבסוף יורדת מ-0.1 ובמקביל הציון של ה-IoU הולך ועולה, ומגיע כמעט עד 0.9.

כזכור, פונקציית ההפסד שבחרנו בוחנת את הקורלציה בין מראה את ההפרש בין ה-mask המשוערך לאמיתי, וכפי שציינו ככל שהערך יותר קטן כך נהיה יותר שבעי רצון מהתוצאה, כיוון שזה מראה על קורלציה טובה יותר. מהגרף הימני ניתן להתרשם שמתקבלת קורלציה טובה, כיוון שערך פונקציית ההפסד מגיע לפחות מ-0.1.

פרמטר ה-metrics שבחרנו היה IoU, וכפי שכתבנו הוא בוחן את היחס בין ה-mask האמיתי לבין המשוערך. בגרף השמאלי ניתן לראות שערכי ה-IoU גדלים ומגיעים לאזור ה0.85-0.9, כלומר יש כמעט 90% חפיפה בין ה-mask שהתקבל לאחר הלמידה לבין האמיתי.

שני פרמטרים אלו הולכים ומשתפרים ככל שמעדכנים את המשקולות, וזה סימן שתהליך הלמידה עובד כראוי. אמנם קיבלנו ערכים די טובים בגרפים, אך היינו שמחים לשפר יותר את התוצאות שלנו. כפי שכתבנו קודם באריכות ניתן היה לעשות זאת בעזרת שיפור הפיקסלים השונים מסביבתם, ופה אנו יכולים להציע משהו נוסף שיכול לשפר את הביצועים. כפי שציינו בהתחלה, סט האימון שלנו אינו ענקי – לשם השוואה, MNIST הידוע מכיל 55,000 ספרות בכתב יד, שזה יותר מפי עשרה מסט האימון שלנו. לכן, אם היינו יכולים להרחיב את מאגר התמונות שלנו, ניתן לשער שהיינו מצליחים לקבל תוצאות אף יותר טובות.

### Pre-trained

שיפור נוסף שניתן להציע, הוא להוסיף שלב מקדים לתהליך האימון. בשלב זה יש לנו את הסט המורעש, ולפי שנאמן אותו על מנת שנוכל לזהות את ה-mask, נוכל להוסיף שכבה המפחיתה את הרעש. אלגוריתם ייעודי לכך הינו [dark channel prior](http://kaiminghe.com/publications/cvpr09.pdf). השתמשנו בקוד קיים עבור אלגוריתם זה, וקיבלנו תוצאות מעניינות. את הקוד העתקנו מכאן: <https://github.com/Utkarsh-Deshmukh/Single-Image-Dehazing-Python>.

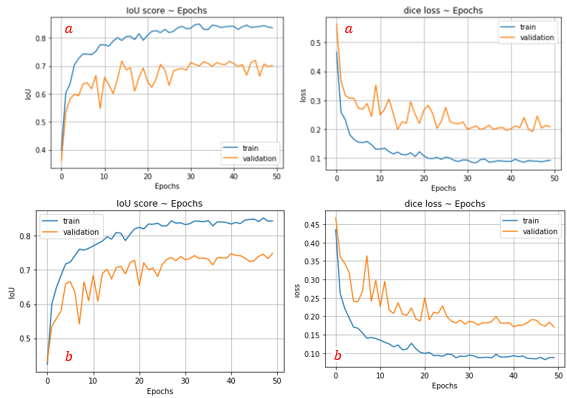
נראה שתי דוגמאות לפלט על מנת להמחיש מדוע שימוש באלגוריתם זה יכול להיות יעיל למקרה שלנו:





איור 13. הפחתת רעש בעזרת אלגוריתם dark channel prior.

ניתן לראות שבתמונות מימין כלל האובייקטים יותר ברורים לעין. אמנם התמונה עצמה נהייתה כהה, אך ה-SNR (יחס אות לרעש) פחת. כעת נאמן את המודל על התמונות לאחר הפחתת הרעש, כפי שפירטנו לעיל. התוצאות שקיבלנו מעט יותר טובות מאלו שקיבלנו ללא הצעד המקדים של הורדת הרעש. נראה זאת בעזרת הגרפים:



איור 14. השוואת הגרפים ללא pre-trained (a) וגרפים עם pre-trained (b).

ההבדלים לא דרמטיים, אבל עדיין ניתן לראות שהגרפים עבור הדאטא שעבר ניקוי רעש מגיעים לביצועים מעט יותר טובים: ב-dice loss, העקומה של ה-validation יורדת עד 0.15 עם ה-pre-trained בעוד שללא pre-trained היא לא יורדת מתחת ל0.2. סדר גודל דומה של יתרון לדאטא עם ה-pre-trained ניתן לראות גם בעקומות הכתומות עבור מדד ה-IoU.

נתייחס גם לתמונה שהרחבנו עליה לעיל. כעת קיבלנו:



איור 15. תמונה 30 לאחר pre-trained.

פה ניתן לראות באופן בולט יותר את היתרון של ניקוי הרעש. אותו חור שנוצר בעקבות הרעש וניסינו לסתום, לא נוצר בכלל כאשר ביצענו ניקוי רעשים מקדים. הפעם הצורה של הרכב ברורה ונכונה יותר.

לסיכום, הוספת ה-pre-trained באופן שבו בחרנו (dark channel prior) מציגה ביצועים מעט טובים יותר מאשר הביצועים הראשונים שהצגנו, אך ההבדל הוא לא גדול מאוד מבחינת המספרים. לגבי התמונה עליה הרחבנו בחלק הקודם – הפעם ראינו שהבעיה שניסינו לפתור כלל לא התחילה, כיוון שסינון הרעש מנע אותה.

### Summary

בעבודה התמודדנו עם בעיית זיהוי אובייקטים בתמונה תחת רעש מסוים (planet Vulcan). יותר בפירוט – ניסינו לזהות תבניות של כלי רכב בתמונה נתונה. הדאטא שלנו נלקח מ-CamVid – סרטונים שהפכו אותם לרצף תמונות. בהתחלה הסברנו כיצד הגדלנו את המאגר בעזרת albumentations, לאחר מכן הסברנו על הרעש וכיצד הוספנו אותו לתמונה – רעש שונה עבור כל תמונה, כשכבה נוספת מעל התמונה המקורית. בשלב זה עברנו לעיקר התהליך – הלמידה והאימון – ופירטנו את הפרמטרים בהם השתמשנו. לבסוף ראינו את התוצאות, הן ויזואלית והן מספרית/גרפית ביחס לפונקציות שהגדרנו בתהליך האימון. כמו כן, הצענו דרכים לשיפור הארכיטקטורה על מנת לספק ביצועים טובים אף יותר ולגשר על הפערים המובנים הנולדים כתוצאה ישירה של הרעש, ואף הראנו מימוש שלהם בפועל. ההצעה שלנו היתה להוסיף postprocessing step שנועד לזהות נכון פיקסלים שזוהו בצורה שגויה בעקבות הרעש, וזאת בעזרת מציאת אזורים חשודים ותיקונם כפי שפירטנו.

בשלב זה הוספנו pre-trained שנועד לנקות רעשים בשיטת dark channel prior, וראינו שהשגנו תוצאות קצת יותר טובות.

נגענו גם בנקודה משמעותית בעינינו – גודל סט האימון, ועמדנו על כך שבאופן יחסי הוא לא גדול, ואם היתה לנו האפשרות להרחיבו בסדר גודל, כנראה שהתוצאות היו אף משופרות יותר ממה שהצגנו בעבודה.

בסך הכל העבודה היתה מאוד מלמדת ומהנה, ואף שמחנו לקבל תוצאות טובות וזיהוי יחסית טוב של האובייקטים הרצויים תחת הרעש הנתון. כמו כן, היה מאוד מרתק לקרוא המון חומר באינטרנט ולגלות שיטות רבות בתחום של CV. הקורס היה עבורנו דלת כניסה מצוינת לתחום הזה, ונתן לנו את הכלים הבסיסיים להתקדם בעצמנו, כפי שנדרשנו לבצע בעבודה זו. תודה רבה.