פרויקט ראייה ממוחשבת

סמסטר א' 2020

יותם לייבוביץ 204095632

אייל אסולין 300037397

1 משימת הפרויקט

נתון סט של 60 תמונות של אוטובוס צעצוע ב-6 צבעים שונים וקובץ טקסט של תיוגי אמת. יש לבנות אלגוריתם שיבצע לוקליזציה וקלסיפיקציה של האוטובוסים. הקלט של האלגוריתם הוא סט תמונות חדשות של אוטובוסים לזיהוי, והפלט שלו הוא קובץ טקסט המכיל את הקואורדינטות והצבע של כל אוטובוס בתמונה באופן הבא:

<image name>:[$x1_{min}$, $y1_{min}$, width1, height1, color1], [$x2_{min}$, $y2_{min}$, width2, height2, color2],...

כאשר,

x_{min}, y_{min} - הקואורדינטות של הפינה השמאלית העליונה של המלבן התוחם את האוטובוס

width, height – הרוחב והגובה של המלבן התוחם את אוטובוס

color – הצבע של האוטובוס ע"פ המילון [1:green, 2:yellow, 3:white, 4:gray, 5:blue, 6:red]

האלגוריתם ייבחן על סט תמונות חדשות ויוערך ע"פ דיוק הלוקליזציה (IoU > 0.7 ביחס לתיוג אמת), נכונות זיהוי (בינארי), וכמות ה-False Positive ו-Miss Detection.

2 סקירה כללית של הפתרון

הפתרון שלנו מתבסס על כלים של Deep Learning, בפרט רשתות CNN (Convolution Neural Network) שהם הסטנדרט כיום עבור זיהוי עצמים בתמונות. קיימות מספר ארכיטקטורות נפוצות של CNN לזיהוי עצמים, כאשר המוכרות ביותר הן Faster RCNN, YOLO, SSD. אנחנו בחרנו להשתמש בארכיטקטורת SSD כיוון שהיא נחשבת המאוזנת ביותר מבחינת יחס ביצועים לעומת זמן ריצה (ע"פ מדריכים באינטרנט, Faster RCNN בעלת הביצועים הטובים ביותר אך עם זמן ריצה האיטי ביותר, YOLO בעלת זמן ריצה המהיר ביותר אך ביצועים פחותים לעומת שתי האחרות).

SSD (Single Shot MultiBox Detector) הוא ארכיטקטורה לרשת נוירונים לזיהוי עצמים שהוצעה במאמר מ-2015 ע"י Wei Liu et al. (לינק למאמר https://arxiv.org/abs/1512.02325). המבוססת על 16 שכבות בסיס "עמוקות" בארכיטקטורות VGG.

הואיל וכמות התמונות הנתונות לנו עבור האימון היא מאוד קטנה (60 תמונות בלבד), וכמו כן גם משיקולים של זמן, השתמשנו בשיטה של **Transfer Learning**. ע"פ שיטה זו, משתמשים ברשת קיימת שאומנה בעבר על מאגרי תמונות כמו ImageNet, COCO, Pascal VOC, ומאמנים מחדש רק את השכבות האחרונות שאחראיות על הקלסיפיקציה עם הדאטה סט החדש אותו מבקשים לזהות. בפרקטיקה ניתן לקבל בעזרת שיטה זו תוצאות יפות גם עבור דאטה סט קטן וזמני אימון קצרים, בזכות העובדה שהשכבות העמוקות יותר של הרשת, שאחראיות על זיהוי תבניות ויצירת Feature maps, כבר אומנו על מאגרי מידע גדולים.

לאחר אימון הרשת ויצירת מודל ללוקליזציה של האוטובוסים בתמונה הוספנו אלגוריתם לזיהוי הצבע ע"י שימוש בכלים של Computer Vision (באשר ההחלטה על הזיהוי הסופי מתקבלת ע"י השוואה בין הפרדיקציות של שני האלגוריתמים).

3 בניית המודל

3.1 הקטנת התמונות (Resizing)

אנחנו בחרנו להשתמש במימוש של רשת **SSD512 על בסיס Keras** (API של TensorFlow). רשת זו מקבלת כקלט רק תמונות 512X512 בפורמט BGR. ביצענו הקטנה של תמונות המקור מגודל 3648X2736 לגודל של 512X512 ע"י ריפוד באפסים (על מנת שלא לשנות את הaspect ratio של התמונות, מה שיגרום לעיוות לא רצוי).

ניתן למצוא את הסקריפט שמבצע את הקטנת התמונות תחת resizeImgs.py

3.2 יצירת אוגמנטציות

הדאטה סט הנתון מכיל 60 תמונות בלבד עם קובץ תיוגי אמת. זהו סט קטן מדי לביצוע אימון אפקטיבי. כיוון שאין מאגר מידע המכיל תמונות של אוטובוס צעצוע, האופציה היעילה ביותר היא הגדלת הדאטה סט באופן מלאכותי ע"י ביצוע אוגמנטציות שונות על התמונות המקוריות (לאחר הקטנה לגודל 512X512). יצרנו דאטה סט חדש של 3062 תמונות הכולל את התמונות המקוריות + 50 אוגמנטציות שונות לכל תמונה. האוגמנטציות שביצענו כוללות: טרנספורמציה אפינית (scaling, סיבוב, הזזה), היפוך אופקי, היפוך אנכי, הרעשה (Additive Gaussian Noise, Salt & Pepper, Dropout) ושינוי contrast (Gamma contrast, Linear contrast) ועיוות (במקרים בודדים). יצרנו קובץ תיוגי אמת חדש לכל הדאטה סט הכולל את התמונות המקוריות + האוגמנטציות.

ניתן למצוא את הסקריפט שמבצע האוגמנטציות ויוצר קובץ תיוגים חדש תחת dataAugmentation.py

למימוש האוגמנטציות השתמשנו בספריה **imgaug**

3.3 מימוש המודל

השתמשנו במימוש של רשת SSD512 מעל keras. את בניית המודל והאימון ביצענו ב-Google Colab. מספר המחלקות (classes) שהגדרנו למודל הוא 6, בהתאם למספר הצבעים השונים של האוטובוסים (למעשה יש בפלט 7 מחלקות, מחלקה אחת לכל צבע + מחלקת background). טענו למודל משקלים של רשת SSD שעברה אימון על מאגר COCO. כיוון שלמאגר COCO יש 80 מחלקות שונות של עצמים, התאמנו את הקובץ כך שיתאים ל-6 מחלקות בלבד. השתמשנו בפרמטרים הנתונים של המודל ששימשו לאימון המודל על COCO. הפרמטר היחידי ששינינו הוא מספר המחלקות שהוא כאמור 6. קימפלנו את המודל עם Adam optimizer (ע"פ המלצות במדריכים נכתב כי הוא טוב יותר מSGD), ופונקציית loss מותאמת ל-SDD שמחשבת L1 loss על הלוקליזציה ו-log loss על הקלסיפיקציה.

ניתן למצוא את המימוש תחת הריפוזיטורי https://github.com/pierluigiferrari/ssd_keras

3.4 אימון המודל (Training)

אימנו מחדש רק את השכבות האחרונות של הרשת (משכבה 19 בשם 'fc6' עד הסוף) ואת 18 השכבות הראשונות (השכבות ה"עמוקות" של ה-VGG) הותרנו ללא שינוי. סה"כ הפרמטרים בשכבות המאומנות = 10,409,100 פרמטרים. תחילה חילקנו את הדאטה סט לtrain ו-validation ביחס של 80% ו-20% בהתאמה. הפרמטרים של האימון:

- number of epochs = 100
- steps per epoch = train size / batch size

- batch size = 8
- IoU threshold = 0.35 (סף שמנפה זיהויים של מלבנים חופפים)
- confidence threshold = 0.5 (סף שמנפה זיהויים חלשים)

השתמשנו L2 Regularization כדי להימנע מOverfit (שיטה בה מוסיפים לפונקציית loss מחיר שתלוי בגודל של המשקלים, כך שהמודל "יעדיף" משקלים קטנים יותר, מה שיפשט את הרשת). קבענו:

- L2 Regularization = 0.003

מבחינת קצב האימון (Learning rate), השתמשנו בפונקציות callbacks של keras כדי להיטיב את האימון:

- Initial learning rate = 0.001
- Learning Rate Scheduler – מוריד את קצב האימון ל-0.0001 לאחר 80 epochs.
- Reduce Learning Rate On Plateau – מוריד את קצב האימון פי 0.8 לאחר מספר epochs רצופים ללא שיפור ב-val loss.
- Early Stopping – מפסיק את האימון לאחר 20 epochs ללא שיפור ב-val loss.

כדי להימנע מOverfit, דאגנו שה-train loss לא ירד משמעותית ביחס ל-val loss, כלומר נמנענו מ"אימון יתר". עוד דאגנו שב-training set לא תהיה שום אוגמנטציה של תמונה מה-validation set, בכדי שנוכל לבחון את המודל בצורה מהימנה על תמונות שהוא לא התאמן עליהן כלל.

3.5 בחינת המודל (Testing)

לאחר שקיבלנו מספר מודלים מאומנים, ערכנו השוואה ביניהם ובחרנו מתוכם את הטוב ביותר. את המודל המאומן הרצנו במוד inference, ע"י בניית המודל ב-keras וטעינת המשקלים שנשמרו מהאימון. בחנו אותו על הדאטה סט של כל התמונות המקוריות, ובמיוחד התמקדנו בביצועים שקיבלנו עבור התמונות שהיו ב-validation set, כיוון שהן תמונות שהמודל לא התאמן עליהן. בדקנו את המודל גם ידנית וגם ע"פ הסקריפט שמפיק את הF1 score. בחרנו פרמטר **confidence threshold = 0.8** – הוא הסף הביטחון שמתחתיו הפרדיקציה נשמטת. בחרנו בערך זה לאחר שראינו שהוא נותן את התוצאות הטובות ביותר מבחינת הטרייד-אוף בין False Positive ל-Miss Detection. כמו כן ראינו כי זמן הריצה של ה-inference אכן עומד בדרישות (כ-1 שנייה לתמונה בממוצע ב-Colab).

לאחר שראינו כי הקלסיפיקציה (סיווג האוטובוסים לצבעים) של המודל אינה מיטבית, החלטנו לשפר אותה ע"י אלגוריתמים של Computer Vision.

ניתן למצוא את ה-inference של המודל תחת runMe.py

3.6 שיפור קלסיפיקציה לצבעים (Classification)

המודל נותן כפלט את המלבן התוחם של כל אוטובוס בתמונה. עבור כל מלבן בפלט האלגוריתם חותך את התמונה כך שיסתכל על המלבן בלבד, מעביר אותה מפורמט RGB ל-HSV ומבצע סכימה של הפיקסלים במלבן עבור כל צבע, ע"פ סקאלות שהגדרנו. מקבלים score עבור כל אחד מ-6 הצבעים האפשריים: ירוק, צהוב, לבן, אפור, כחול ואדום.
האלגוריתם בוחר את הצבע ע"פ לוגיקה שמתחשבת בצבע החזק ביותר, אך גם משקללת את התרומה של הפרדיקציה של ה-SSD, בהתאם ל-confidence level שלה. ע"י ניסוי וטעיה קבענו את הפרמטרים שנראו לנו הטובים ביותר.

ניתן למצוא את האלגוריתם לקלסיפיקציה לצבעים תחת runMe.py

למימוש האלגוריתם השתמשנו בספריה OpenCV