# 9. Computer Vision

## 9.1 Object Detection

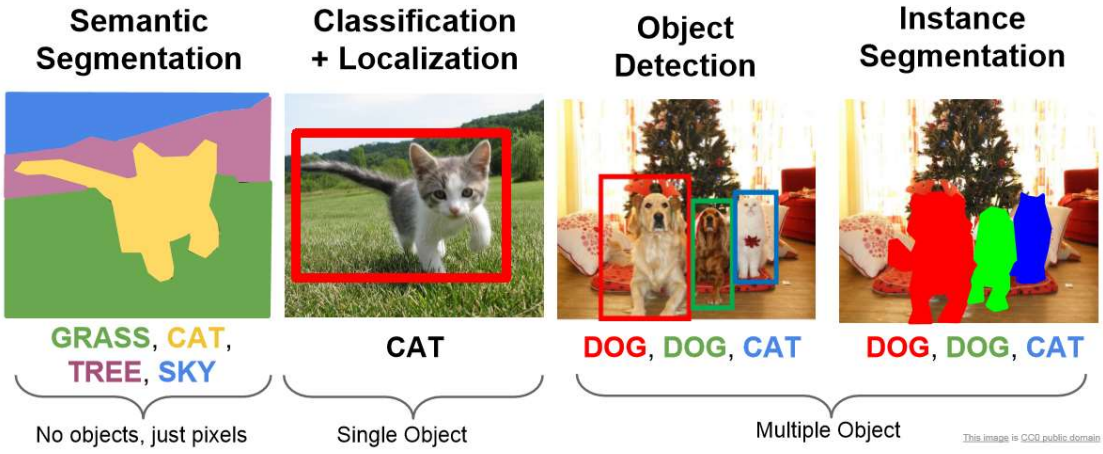
### 9.1.1 Introduction to Object Detection

זיהוי אובייקטים היא משימה מאוד נפוצה בעולם של ראייה ממוחשבת, ויש לה המון יישומים מגוונים. ניקח למשל מכונית אוטונומית, שבכל רגע צריכה לזהות את האובייקטים שסביבה ולקבל תמונת מצב עדכנית על המתרחש, או לחילופין מצלמה של טלפון נייד שיודעת לזהות פנים של בנאדם בכדי לבצע עליהם פוקוס או לתקן רעשי רקע, ועוד המון יישומים נרחבים בכל מיני תחומים.

בשלב ראשון יש להגדיר באופן מדויק את המשימה – מה הכוונה לזהות אובייקט בתמונה? יש כמה רמות שונות של זיהוי. המשימה הקלאסית של סיווג (classification) מניחה שיש אובייקט יחיד בתמונה, והמטרה היא לסווג אותו בצורה נכונה, כלומר לקבוע מה ה-class שלו. משימה יותר מתקדמת היא לא רק לומר איזה אובייקט נמצא בתמונה, אלא גם לומר איפה בדיוק הוא נמצא. גם פה יש שתי רמות – ניתן לסמן את האזור בו הוא נמצא בעזרת מלבן (bounding box) שמקיף את השוליים שלו, וניתן לסווג כל פיקסל בפני עצמו האם הוא שייך לאובייקט או לרקע. הזיהוי מהסוג הראשון נכנס תחת התחום של Object Detection ואילו זיהוי סמנטי של הפיקסלים נכנס תחת התחום של Segmentation.

ניתן להכליל את משימת הזיהוי גם ל-multiple object. כלומר, יש מספר לא ידוע של אובייקטים בתמונה, והמטרה היא למצוא היכן הם נמצאים ולסווג כל אחד מהם ל-class המתאים. בעצם משימה זו מורכבת משתי תתי משימות – מציאת המיקום של האובייקט (Localization/Regression) וסיווג האובייקט ל-class הנכון (Classification). נשים לב שעבור משימת Object Detection, ההכללה מאובייקט יחיד למספר אובייקטים הינה ישירה – על המודל לספק מספר bounding boxes ועבור כל אחד מהם להתאים class. במשימת Segmentation לעומת זאת, ההכללה למספר אובייקטים יכולה להיעשות בשתי דרכים: 1) Semantic Segmentation – שיוך כל פיקסל ל-class מסוים ללא הבחנה בין פיקסלים השייכים לאובייקטים שונים בעלי אותו class. במקרה זה אם יהיו שני כלבים בתמונה, אז במפת הסגמנטציה נראה הרבה פיקסלים המשויכים ל-class של כלב, אך לא נוכל להבין בין הכלבים השונים, ואפילו לא נדע שיש יותר מכלב אחד. 2) Instance Segmentation – שיוך כל פיקסל ל-class מסוים תוך הבחנה בין פיקסלים המשתייכים ל-class ספציפי אך שייכים לאובייקטים נפרדים.

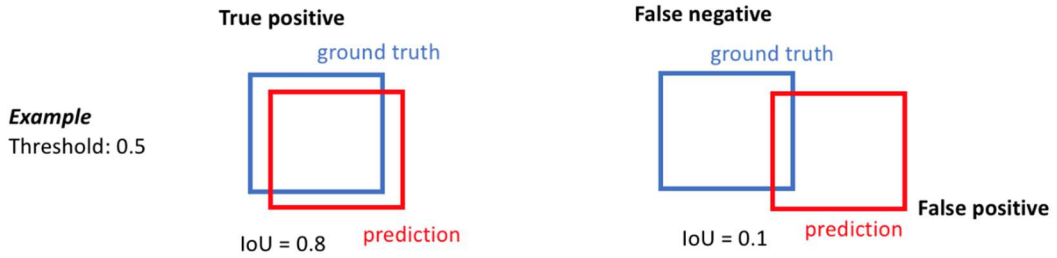
בפרק זה נדון במשימת Object Detection ובפרק הבא ב-Segmentation. בשביל להגדיר את המשימה ולקבוע מה נחשבת הצלחה, ראשית יש להגדיר מה נחשב השטח בו נמצא האובייקט, ובהתאם מה אנו מצפים מהמודל שיספק לנו כפלט. בתחום של Object Detection, המוסכמה היא לסמן אובייקט באמצעות מלבן, שנקרא bounding box (או בקיצור bbox), כאשר קצוות האובייקט משיקים ל-bbox. נניח ויש תמונה ובה בנאדם, אז ה-bbox יהיה מלבן המכיל את כל הגוף שלו, והמלבן ישיק לקצוות הגוף – החלק העליון של המלבן ישיק לנקודה הגבוהה ביותר של הראש, החלק התחתון ישיק לכפות הרגליים, ובאופן דומה בשני הצדדים. אם יש כמה אובייקטים, אז לכל אובייקט ישויך bbox בנפרד.




איור 9.1 זיהויים של אובייקטים בתמונה: Classification, Object Detection and Segmentation (semantic and instance). במשימת Object Detection המטרה היא לסמן כל אובייקט באמצעות bounding box – מלבן המקיף אותו ומשיק לקצוות שלו מכל כיוון.

לאחר שהגדרנו בצורה מדויקת מה נחשב השטח בו נמצא האובייקט – נוכל לקחת תמונה ולסמן בה את האובייקטים בהתאם למוסכמה זו, כאשר נהוג לכנות כל אובייקט מתויג כ-groundtruth. אם נאמן גלאי (detector) לפי המוסכמה הזו, אז נרצה שהפלט שלו יהיה אוסף של דטקציות (detections) – מלבנים מסביב לאובייקטים יחד עם סיווג תואם לכל אחד מהאובייקטים. כאשר נריץ את הגלאי המאומן על תמונה מסוימת, נוכל להשוות בין כל האובייקטים שתויגו (groundtruths), לבין הדטקציות של המודל. מקובל לבצע את ההשוואה בעזרת מדד שנקרא Intersection Over Union, או בקיצור IoU. מדד זה בוחן עבור על אובייקט את היחס של גודל החפיפה בין ה-groundtruth לבין הדטקציה התואמת עבורו. אם החפיפה גדולה מסף מסוים, אז נוכל לומר שהגלאי זיהה בהצלחה שיש פה אובייקט. סף זה נקרא IoU threshold, וערך טיפוסי עבור הינו 50%, כלומר חפיפה של מעל 50% בין מיקומו האמיתי של האובייקט לבין הדטקציה של המודל נחשבת כמציאה נכונה של האובייקט. אם גם הסיווג של אותו אובייקט על פי המודל זהה ל-class האמיתי (בהתאם למה שתויג מראש), אז יש פה התאמה מלאה ואותו אובייקט הוא בעצם true positive – דוגמה חיובית שזוהתה באופן נכון.

אם יש אובייקט שאין אף דטקציה שחופפת לאזור בו הוא נמצא, אז אותו אובייקט הינו **False Negative (FN)** (groundtruth שלא זוהה). אובייקט יהיה false negative גם במצבים בהם יש חפיפה אך היא אינה עוברת את הסף שנקבע, או לחילופין יש חפיפה מספיקה אך הסיווג שגוי. נשים לב שבשני המקרים האחרונים יש גם **False Positive (FP)** – הדטקציה אינה מתאימה לאף groundtruth, ולכן היא למעשה זיהוי חיובי של אובייקט שבאמת לא קיים. הגדרות אלו מאפשרות להשתמש במשימת Object Detection במטריקות מקובלות כמו Precision ו-Recall באופן דומה לשימוש הקלאסי שלהם במשימות classification, כפי שיוסבר בהרחבה בהמשך.




איור 9.2 Intersection Over Union (IoU) – מדד חפיפה בין groundtruth לבין דטקציה. עבור IoU threshold=0.5, דטקציה שחופפת בלפחות 50% ל-groundtruth הינה true positive, ואילו דטקציה שאינה עוברת את סף החפיפה הינה false negative ובנוסף ה-groundtruth הינו false negative.

לאחר שהגדרנו בצורה מדויקת ומפורטת מהי משימת זיהוי אובייקטים נסביר בקצרה על הגישות השונות בתחום והתפתחותן לאורך השנים. אמנם בשנים האחרונות כל המודלים הינם מבוססי Deep Learning, אך כמובן שמשימת זיהוי אובייקטים הייתה קיימת עוד לפני התפתחות הענף והיו אלגוריתמים קלאסיים לצורך כך, כמו למשל אלגוריתם Scale-Invariant Feature Transform (SIFT) שהיו פופולרי מאוד בעשור הראשון של המאה. לרוב, גם האלגוריתמים הקלאסיים וגם אלו שמבוססי רשתות נוירונים פועלים על פי אותו רעיון, שהינו פשוט ואינטואיטיבי. כיוון שמשימת זיהוי אובייקטים כלולה משני חלקים – Localization and Classification, נרצה שהמודל שלנו גם הוא יפעל בשני חלקים – בשלב הראשון נמצא אזורי עניין בתמונה, כלומר אזורים החשודים ככאלה שיש בהם אובייקטים (Region Of Interest – ROI), ולאחר מכן נבצע שני דברים במקביל – ננסה לחזות את ה-bbox המדויק של אובייקט הנמצא באותו אזור, ובנוסף ננסה לסווג אותו לקלאס מסוים.

למרות שגישה זו פשוטה יחסית, יש בה שתי בעיות מובנות. ראשית כל – יתכנו אזורים שהמודל יחשוד שיש בהם אובייקט אך בפועל באמת אין בהם אובייקט אלא הם מכילים רק רקע. שנית – הרבה אלגוריתמים מתוכננים באופן כזה שהקלט שלהם הוא בגודל קבוע וידוע מראש, והם גם מוציאים פלט בגודל קובע. כיוון שמספר האובייקטים הוא משתנה, פלט בגודל קבוע יכול להיות בעייתי. למעשה יש פתרון פשוט לשתי הבעיות – נדרוש מהמודל לצרף לכל אחת מהדטקציות מספר הסתברותי בין 0 ל-1 שמשמעותו היא "עד כמה אני בטוח שבאמת יש פה אובייקט". במקביל לכך, על המשתמש להחליט על סף מסוים (confidence threshold), כך שרק אובייקטים בעלי הסתברות הגבוהה מסף זה יחשבו כזיהויים של המודל, ומכל יתר הזיהויים פשוט נתעלם. באופן הזה נוכל לקבוע מראש שהמודל יוציא מספר קבוע של זיהויים, למשל 100, אך נתעלם מכל אלה שלא עברו את הסף. מודל טוב ייתן הסתברות גבוהה רק לאזורים בהם באמת יש אובייקטים, ואילו כל יתר האזורים החשודים יהיו בעלי הסתברות נמוכה. בנוסף, מודל כזה ידע להתעלם מאזורי עניין שנמצאו כחשודים בשלב הראשון שלו ולתת להם הסתברות נמוכה. (בהמשך יוסבר על אלגוריתם NMS שגם מתקשר לעניין של מספר הדטקציות).

כאמור לעיל, בשנים האחרונות כל הגישות המובילות בתחום זה הן מבוססות Deep Learning, כאשר הן פועלות על פי אותו עיקרון – חילוץ ROIs בשלב הראשון, ולאחר מכן ביצוע רגרסיה ל-bbox של האובייקט יחד עם סיווג שלו ל-class ספציפי. באופן כללי ניתן לחלק את הגלאים לשלוש קבוצות:

גלאי דו-שלבי (Two-stage detector):

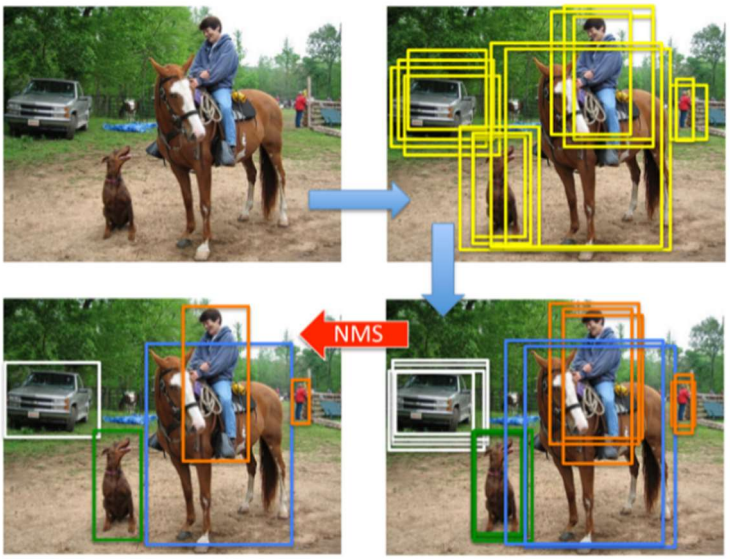
גישה אחת, הנקראת two-stage detection, אכן פועלת לפי שני השלבים שתיארנו קודם: בהתחלה מופעל אלגוריתם לזיהוי ROIs, כאשר אלגוריתם זה הוא רשת נוירונים שמחלצת פיצ'רים בתמונה נתונה ומזהה בה אזורי עניין, ולאחר מכן אזורים אלו נכנסים לרשת נוספת המבצעת רגרסיה ל-bbox וסיווג ל-class. המוצא של הרשת השנייה בנוי בצורה של two head detection, כאשר לרוב ראש הרגרסיה מנסה ללמוד בתהליך האימון (ולחזות ב-inference) ארבעה פרמטרים – את מרכז האובייקט $(x, y)$ והאורך והרוחב שלו $(h, w)$.

גישה זו היא בעצם השיטה הכי אינטואיטיבית, ומשפחת הגלאים הראשונה ואולי הכי מפורסמת הפועלת לפי עיקרון זה של two-stage היא R-CNN Family. משפחה זו הינה סדרה של גלאים שפותחו בשנים 2014-2017, כאשר כל גלאי מתבסס על קודמו ומשפר אותו, וכולם יחד בנויים בצורה של two-stage. עם הזמן התפתחו עוד גלאים הפועלים באופן דומה, כמו למשל SSP ו-FPN שנדון בהם בהרחבה בהמשך הפרק.

גלאי חד-שלבי (One-stage detector):

גישה נוספת, דומה בקונספט אך שונה בדרך הביצוע, נקראת one-stage detection. גם בגישה זו הרעיון הוא למצוא אזורי עניין ועבורם לבצע רגרסיה של bbox וסיווג ל-class, אך הפעם אזורי העניין אינם נקבעים באמצעות אלגוריתם כלשהו, אלא אנו **קובעים אותם מראש**. אנו למעשה אומרים מראש למודל איפה בדיוק לחפש אובייקטים, כאשר אותם אזורים מסומנים מראש נקראים anchors. בדרך כלל קובעים מראש הרבה anchors, כאשר יש anchors בגדלים ובצורות שונות, על מנת שנוכל לזהות אובייקטים במגוון גדלים וצורות. למשל – נחלק את התמונה לגרידים בגדלים שונים וכל מלבן יהיה anchor, אך נרצה להוסיף גם anchors בצורת מלבנים רוחביים בכדי לזהות מכוניות, וכן נרצה anchors בצורת מלבנים אופקיים בכדי לזהות אנשים עומדים, וכך הלאה. העיקרון הוא שנקבע מראש הרבה anchors כך שכל אזור בתמונה יכוסה במספר anchors בגדלים שונים, בכדי לאפשר זיהוי של אובייקטים בעלי אופי שונה. גלאי מפורסם שעובד לפי העיקרון של one-stage הינו YOLO – You Only Look Once, וכשמו – הוא עובר על התמונה רק פעם אחת (ולא כמו ב-two-stage, שם יש צורך לעבור פעמיים על התמונה – פעם אחת לחילוץ ROIs ופעם שנייה לרגרסיה וסיווג). גלאים נוספים הפועלים לפי גישה זו הינם SSD ו-RetinaNet, שגם בהם נדון בהרחבה בהמשך.

בשונה מגלאי דו-שלבי המחלץ בעצמו את אזורי העניין, גלאי המשתמש ב-anchors קובע מראש את מספר אזורי העניין. עובדה זה יכולה לגרום לכך שיהיו כמה אזורי עניין שונים המכילים את אותו אובייקט. אם למשל חילקנו את התמונה לגריד של $10 \times 10$ ויש אובייקט שתופס רבע מהתמונה, אז יהיו כ-25 anchors שיכילו בתוכם את האובייקט. כדי לאחד את כל הזיהויים של ה-anchors האלה מקובל להשתמש באלגוריתם קלאסי שנקרא Non Maximum Suppression, או בקיצור NMS. הרעיון של האלגוריתם הוא לעבור על כל הדטקציות, ואם יש דטקציות השייכות לאותו קלאס ויש ביניהן חפיפה מסוימת, אז הן יתאחדו לכדי דטקציה אחת. השימוש באלגוריתם זה נפוץ גם בגלאי דו-שלבי – אמנם הגלאים מחלצים לבד את אזורי העניין, אך ישנם גלאים בהם **מספר** אזורי העניין קבוע מראש, ולכן יתכנו הרבה דטקציות של אותו אובייקט, ויש צורך לאחד אותן.




איור 9.3 Non-Maximum-Suppression (NMS) – איחוד דטקציות השייכות לאותו אובייקט. בשלב הראשוני מחפשים אזורי עניין, לאחר מכן מסווגים כל אחד מהם, ולבסוף מאחדים זיהויים השייכים לאותו class ויש ביניהם חפיפה גבוהה.

גלאי מבוסס טרנספורמר (Transformer-based detector)

כניסתם של הטרנספורמרים לעולם ה-vision לא דילגה על Object Detection. ב-2020 יצא מאמר שנקרא DETR (קיצור של DEtection Transformers) שהראה כיצד ניתן להשתמש במנגנון של self-attention בשביל למצוא אובייקטים בתמונה ולאחר מכן להשתמש במנגנון של attention בשביל לסווג אותם, כפי שיפורט בהרחבה בהמשך הפרק. מאז יצאו הרבה מאמרים המבוססים על הרעיונות שהונחו ב-DETR, כאשר מאמרים אלו מנסים לשפר את התוצאות תוך התייחסות למספר בעיות מובנות שעולות כאשר משתמשים בטרנספורמרים במשימות של ראייה ממוחשבת. בפרט, יש מספר אתגרים בסיסים: א. הסיבוכיות של טרנספורמר היא ריבועית בגודל הקלט. כאשר הקלט הוא משפט, או אפילו פסקה, זה לא מהווה בעיה, אך כאשר מדובר בתמונות ברזולוציה גבוהה (למשל 4K), אז שימוש בטרנספורמרים נהיה לא יעיל, ועבור משימות זמן אמת אפילו בלתי אפשרי. ב. טרנספורמרים תוכננו במקור לקלט חד ממדי, כמו למשל משפט שהוא סדרה של מילים. בשביל להשתמש בהם עבור תמונות, יש להפוך את התמונות מייצוג דו-ממדי לייצוג חד ממדי. מעבר זה יכול לגרום לאיבוד קשרים מרחביים הקיימים בתמונה. ג. מנגנון ה-attention יכול לזהות תבניות מרחביות, אך הוא לא מתייחס לממד הצבעים, שכמובן יש לו משמעות בעיבוד התמונה.

לכל אחת מהגישות יש יתרונות וחסרונות וכמובן שיש כל מיני עבודות שמנסות לשלב בין היתרונות של הגישות השונות. באופן פשוט ניתן לומר ביחס לשתי הגישות הראשונות, שהן מקיימות trade-off בין דיוק לבין מהירות – גלאי חד-שלבי הוא כמובן יותר מהיר, אך עם זאת הוא פחות מדויק. עבור מכשירי קצה שרצים בזמן אמת ואין להם כוח חישוב רב, כמו למשל פלאפונים, הצרכים העיקריים של המשתמש הם מהירות חישוב וחיסכון בסוללה, ולכן כנראה שתהיה עדיפות לגלאי חד-שלבי. אם לעומת זאת המשתמש צריך דיוק גבוה ויש לו משאבים חישוב חזקים, כמו למשל רופא שמתעסק בפענוח בדיקות רפואיות, אז תהיה לו עדיפות לגלאי דו-שלבי או לגלאי מבוסס טרנספורמר, המספקים תוצאות מעט יותר טובות.

כאמור לעיל, כל הגלאים בשנים האחרונות הם מבוססי רשתות נוירונים – רשתות עבור זיהוי ROIs, רשתות רגרסיה וסיווג, רשתות של טרנספורמרים ועוד. בשביל לאמן את הרשתות האלה יש צורך בדאטה, וכמובן שצריך שהוא יהיה מתויג. בשונה מתיוג תמונות למשימות סיווג, שם כל שנדרש זה רק לתת label לכל תמונה, תיוג של תמונות עבור משימות Object Detection היא משימה מורכבת פי כמה – גם יש כמה אובייקטים בתמונה, וגם צריך לסמן לכל אחד מהם bbox בצורה מדויקת. עקב המורכבות של תיוג כזה, התפתחו אלגוריתמים של semi-supervised, המשלבים למידה מבוססת דאטה מתויג יחד עם למידה שאינה דורשת תיוג.

נציין בקצרה רעיון אחד של semi-supervised learning עבור משימת זיהוי אובייקטים, המאפשר לאמן מודל בלי הרבה דאטה מתויג. הרעיון הוא לאמן את המודל בשני שלבים – בשלב הראשון משתמשים בדאטה המתויג ומאמנים מודל באחת הדרכים שראינו עד כה. לאחר מכן משפרים את הלוקליזציה באופן הבא – מריצים אלגוריתם unsupervised שיודע לזהות אובייקטים (הוא בעצם עושה סוג של clustering לתמונה ומבחין בין אובייקטים לבין אזורים שאינם אובייקטים), ומשווים בין הפלט שלו לפלט של המודל המקורי שאימנו. כיוון שהאלגוריתם השני מאומן רק לבצע לוקליזציה והוא אינו נדרש להבחין בין אובייקטים שונים, ניתן להשתמש עבורו בהרבה דאטה, ולכן הנחה סבירה היא שהוא מזהה אובייקטים בצורה טובה. אם כן, ניתן לבצע אימון שינסה לעשות fine tune ללוקליזציה של המודל הראשון כך שה-bounding boxes שהוא מספק יהיו דומה עד כמה שניתן לפלט של האלגוריתם השני.

בדומה לכך, ישנם אלגוריתמים של self-supervised שאינם מצריכים בכלל דאטה מתויג. אלגוריתמים אלה לומדים לזהות אובייקטים ולהבחין בין אובייקטים שונים, אך הם אינם יכולים לדעת מה ה-label של כל קבוצה, כיוון שאין להם שום דאטה מתויג. זה יכול להיות שימושי למשל עבור רופאים המסתכלים על תוצאות של צילומים רפואיים – האלגוריתם יזהה את האובייקטים ויסווג כל אובייקט לקבוצה מסוימת, ואז הרופא יסתכל על התוצאות וידע לתת את ה-label המתאים לכל אחת מהקבוצות.

### 9.1.2 OD Metrics

על מנת להעריך ביצועי אלגוריתמים של Object Detection, יש צורך במדד כמותי מוסכם שיעיד על טיב הזיהוי ויאפשר השוואה בין אלגוריתמים שונים. נציג מספר מושגים ומדדי איכות נפוצים בסיווג, ונראה איך משתמשים בהם לניתוח והשוואה של ביצועי אלגוריתמי Object Detection. ראשית, נסמן את מצבי הזיהוי האפשריים:

**True Positive (TP)** – מצב בו הרשת מזהה אובייקט קיים.

**False Positive (FP)** – מצב בו הרשת מזהה אובייקט שלא קיים.

**False Negative (FN)** – מצב בו הרשת לא מזהה אובייקט קיים.

**True Negative (TN)** – מצב בו הרשת לא מזהה אובייקט לא קיים.

ניתן לשים לב ש-**TP** ו-**TN** הם מצבים רצויים, כיוון שהרשת צודקת בחיזוי שלה, ואילו **FP** ו-**FN** אינם רצויים, כיוון שהם שגיאות של הרשת. סימון מצבים אלה מאפשר להגדיר את המדדים הבאים:

מדד ה-**Precision** בוחן כמה מתוך האובייקטים שהרשת מזהה אכן קיימים בפועל:

$$P = \frac{TP}{TP + FP}$$

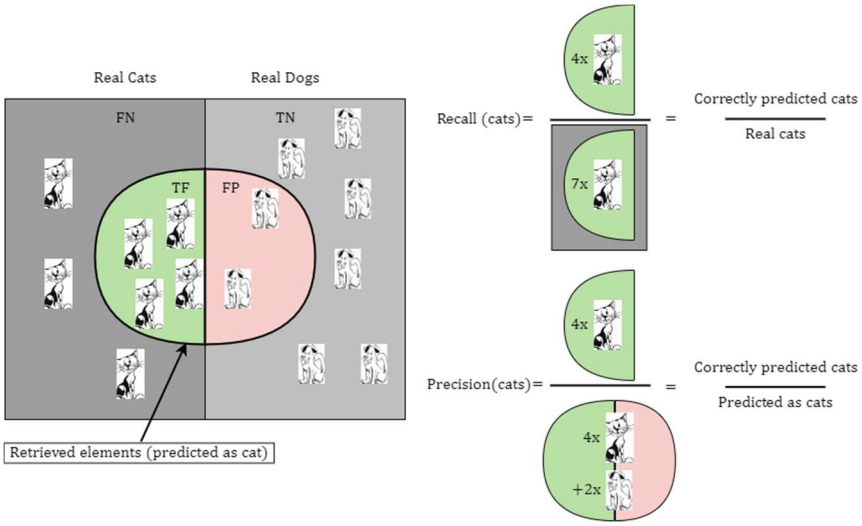
מדד ה-**Recall** בוחן כמה מתוך האובייקטים הקיימים הרשת מצליחה לזהות:

$$R = \frac{TP}{TP + FN}$$

מדד זה לעיתים נקרא גם **True Positive Rate (TPR)**.

מדד ה-**False Positive Rate (FPR)** בוחן כמה מתוך האובייקטים שלא קיימים הרשת מזהה, כלומר, כמות החיזויים החיוביים השגויים מתוך כלל המקרים בהם לא קיים אובייקט:

$$FPR = \frac{FP}{FP + TN}$$




איור 9.4 Precision ו-Recall.

עבור רשתות של זיהוי אובייקטים, מוצא הרשת עבור כל אובייקט כולל גם קלסיפיקציה לסוג האובייקט וגם מיקום של האובייקט (Bounding Box). בשונה מרכיב הקלסיפיקציה, עבור מיקום האובייקט ההשוואה ל-ground truth אינה טריוויאלית ולשם כך מגדירים את מדד ה-Intersection Over Union (IoU). מדד זה הוא למעשה חלוקה בין חיתוך התיבות לאיחוד התיבות (ועל כן ערכו בין 0 ל-1, כאשר ערך גבוהה יותר מצביע על התאמה טובה יותר), כפי שמופיע באיור 9.2.

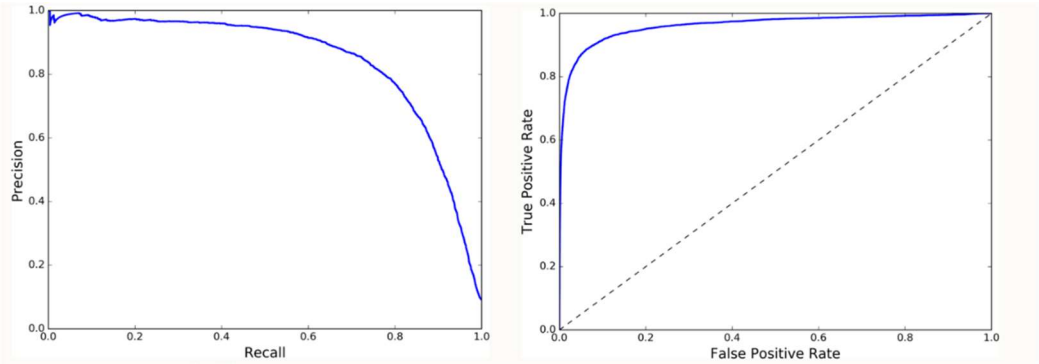
בעזרת ה-IoU מקבלים את מידת ההתאמה בין האובייקטים שהרשת מצאה לבין האובייקטים הידועים (ground truth). עבור כל detection שהרשת מספקת, בוחנים עד כמה יש חפיפה בינו לבין כל אחד מהאובייקטים ב-ground truth. כעת לוקחים את האובייקט בעל החפיפה הגדולה ביותר, ובוחנים האם רמת החפיפה (IoU) עוברת סף מסוים שנקבע מראש (ונקרא IoU threshold). אם רמת החפיפה גדולה מסף זה, הזיהוי נחשב נכון, אחרת מתעלמים מה-detection הזה ומתייחסים אליו כאילו הוא לא זוהה על ידי הרשת. כמובן שבחירת ערך הסף משפיעה על ביצועי הרשת: אם נבחר ערך סף גבוה, נקבל רק אובייקטים עם התאמה מאוד טובה אך גם נפספס אובייקטים רבים (Precision גבוה, Recall נמוך). מנגד, אם נבחר ערך סף נמוך, לא נפספס זיהויים נכונים אך נקבל הרבה זיהויי שווא (Precision נמוך, Recall גבוה). מכאן ניתן להבין שיש tradeoff בבחירת סף הזיהוי – פספוס של זיהויים עבור ערך גבוה וזיהויי שווא עבור ערך נמוך.

כאמור, ראשית יש להגדיר ערך סף של IoU, והוא מתייחס לשאלה מהו נחשב זיהוי (detection) של הרשת. ערך זה קובע גם את מדדי ה-Precision וה-Recall, המתייחסים לשאלת הסיווג (classification) של הרשת. כעת,

באמצעות מדדים אלו, ניתן לאפיין את איכות ביצועי הסיווג של האלגוריתם. עבור ערכי סף שונים ניתן לשרטט גרפית שני סוגי עקומות, המסייעות לבחון את ההצלחה הכללית של הרשת, הן בזיהוי והן בסיווג:

**עקומת Precision-Recall**. בהינתן Recall מינימלי שהאלגוריתם מוכרח לעמוד בו (=אחוז מינימלי של זיהוי וסיווג מדויק), נרצה שה-Precision יהיה מקסימלי. עקומה זו (Recall בציר האופקי ו-Precision בציר האנכי) היא מונוטונית-יורדת, שכן הגדלת ה-Recall כרוכה בהגדלת הסיכוי ל-False Positive ולכן לירידה ב-Precision. מסווג מושלם מתאים לנקודה $(1,1)$ במרחב ה-Recall-Precision שכן הוא לא מפספס אף אובייקט אמיתי (Recall=1) וכל החיזויים החיוביים שלו נכונים (Precision=1). בפועל העקומה של מסווג מציאותי מתחילה בנקודה $(0,1)$ (על ידי סיווג Negative באופן דטרמיניסטי ניתן להשיג Precision=1), וערך ה-Precision יורד בהדרגה עם הגדלת סף ה-Recall. ככל שהעקומה קרובה יותר לישר Precision=1, ביצועי המסווג נחשבים טובים יותר.

**עקומת Receiver Operating Characteristic (ROC)**. בהינתן FPR מקסימלי שאותו אסור לאלגוריתם לעבור (אחוז מקסימלי של זיהוי אובייקטים שאינם קיימים), נרצה שערך ה-TPR יהיה המקסימלי. עקומה זו (FPR בציר האופקי ו-TPR בציר האנכי) היא מונוטונית עולה, שכן הגדלת ה-FPR מאפשרת "לסבול" יותר False Positive, לכן ניתן לסווג Positive בצורה רווחת יותר ובכך להגדיל את ה-TPR. מסווג מושלם מתאים לנקודה $(0,1)$ במרחב ה-FPR-TPR, שכן הוא אינו שוגה באף סיווג חיובי (FPR=0) ואינו מפספס אף אובייקט (TPR=1). בקצה השני של מדרג הביצועים נמצא מסווג אקראי (מכונה "No-skill"), אשר מיוצג ע"י עקומת ROC אלכסונית, שקצותיה $(0,0)$ ו-$(1,1)$. בפועל, עקומת ה-ROC של מסווג מציאותי מתחילה בנקודה $(0,0)$ (מתאים לסיווג Negative באופן דטרמיניסטי), וערך ה-TPR עולה בהדרגה עם הגדלת סף ה-FPR עד לנקודה $(1,1)$ (מתאים לסיווג Positive באופן דטרמיניסטי). ככל שהעקומה קרובה יותר לישר TPR=1 וערכיה גבוהים ביחס לעקומת ה-"No-skill" ביצועי המסווג נחשבים טובים יותר.




איור 9.5 עקומת Precision-Recall ועקומת ROC. עקומות אלה מאפשרות לבחון את טיב המודל.

לאחר מציאת עקומת ה-Precision-Recall עבור המסווג, ניתן לחשב את השטח מתחת לעקומה בכדי לקבל מדד מספרי למידת ה-Precision הממוצעת שהאלגוריתם משיג. מדד זה נקרא Average Precision (AP), ונתון מפורשות על ידי:

$$AP = \int_0^1 Precision(Recall)d(Recall)$$

עבור בעיית סיווג רב-מחלקתית עם סט מחלקות אפשרית $\mathcal{C}$, מקובל לחשב AP לכל מחלקה $c \in \mathcal{C}$ בנפרד, ולאחר מכן למצע על פני כל המחלקות על מנת לקבל את מדד ה-mean AP (mAP):

$$mAP = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} AP(c)$$

מדד ה-mAP הינו המדד הנפוץ ביותר לאפיון ביצועי אלגוריתמי Object Detection.

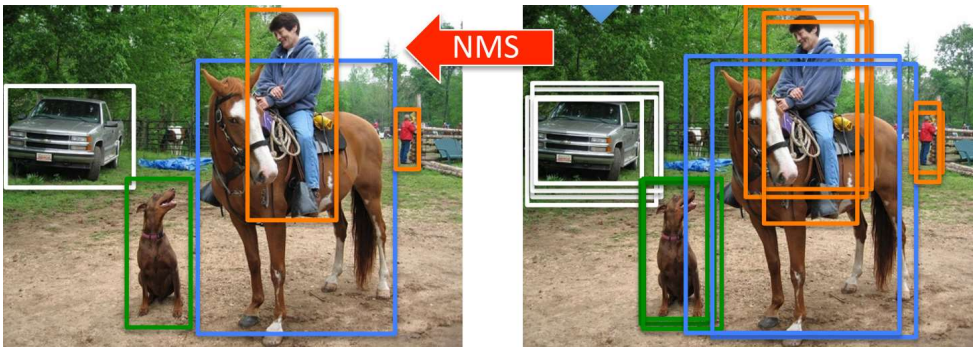
### 9.1.4 You Only Look Once (YOLO)

עד שנת 2016 כל הגלאים (detectors) המבוססים על למידה עמוקה (כמו למשל R-CNN family) היו גלאים דו-שלביים – השלב הראשון יצר אלפי הצעות (proposals) למסגרות מלבניות החשודות כמכילות אובייקטים, ואלו הוזנו בזו אחר זו לשלב השני אשר דייק את המסגרות וביצע סיווג לאובייקטים המוכלים בהן.

ארכיטקטורת YOLO, הנגזרת מראשי התיבות של YOU ONLY LOOK ONCE, הוצגה על ידי ג'וזף רדמון ב-2016, והייתה הגלאי הראשון שמורכב משלב יחיד, ובו הרשת מנבאת את המסגרות וגם מסווגת את האובייקטים שבתוכן בבת אחת. בנוסף, ארכיטקטורת YOLO מתאפיינת במיעוט פרמטרים וכמות פעולות אריתמטיות. היא אמנם משלמת על המבנה הרזה והאלגנטי שלה בדיוק נמוך יותר, אך המהירות הגבוהה (שנובעת בעיקר מהיעדר האילוץ לעבד מסגרת יחידה בשלב השני בכל מעבר של הלולאה) הפכה את גישת השלב היחיד לאטרקטיבית מאוד, במיוחד למעבדים קטנים כדוגמת מכשירי mobile. בעקבות עבודה זו פותחו גלאים רבים על בסיס שלב יחיד, כולל גרסאות מתקדמות יותר של YOLO (הגרסה המתקדמת ביותר כיום היא v5).

### NMS (Non-Maximum Suppression)

כמעט כל אלגוריתם של זיהוי אובייקטים מייצר מספר רב של מסגרות חשודות, כאשר רובן מיותרות ויש צורך לדלל את מספרן. הסיבה ליצירת מספר גדול של מסגרות נובע מאופי פעולת הגלאים. בזכות התכונות הלוקאליות של פעולת הקונבולוציה, מפת הפיצ'רים במוצא הגלאי ניתנת לתיאור כמטריצת משבצות כאשר כל משבצת שקולה לריבוע של הרבה פיקסלים בתמונה המקורית. רוב הגלאים פועלים בשיטת עוגנים (anchors), כאשר כל משבצת במוצא הגלאי מנבאת מספר קבוע של מסגרות שעשויות להכיל אובייקט (למשל, ב-YOLOv2 המספר הוא 5, ובגרסאות המתקדמות יותר המספר הוא 3). השיטה הזו יוצרת אלפי מסגרות שרק מעטות מהן הן משמעותיות. בנוסף – יש ריבוי של מסגרות דומות בסביבת כל אובייקט (למשל – YOLOv3 מנבאת יותר מ-7000 מסגרות לכל תמונה). אחת הדרכים הפופולריות לסנן את אלפי המסגרות ולהשאיר רק את המשמעותיות נקראת NMS. בשיטה זו מתבצעת השוואה בין זוגות של קופסאות מאותה המחלקה (למשל – חתול), ובמידה שיש ביניהן חפיפה גבוהה – מוחקים את המסגרת בעלת הוודאות הנמוכה היותר ונשארים רק עם המסגרת בעלת רמת הוודאות הגבוהה. שיטה זו בזבזנית בחישוב (סיבוכיות פרופורציונלית לריבוע מספר המסגרות) ואינה חלק מהמודל המתאמן, אך עם זאת הינה אינטואיטיבית יחסית למימוש, ומשום כך נמצאת בשימוש נפוץ בגלאים, כולל ב-YOLO.

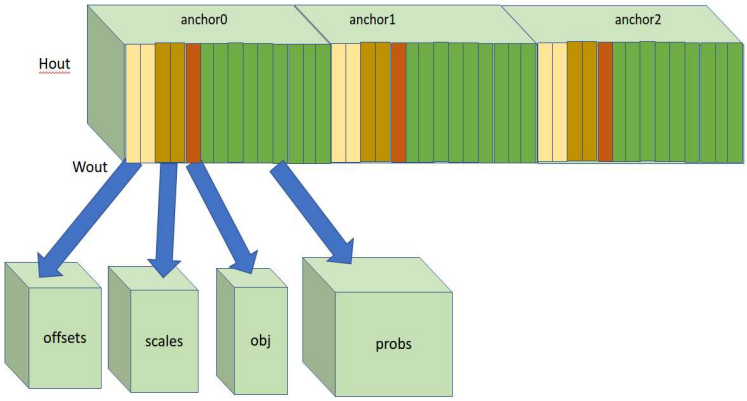



איור 9.6 אלגוריתם NMS.

### YOLO Head

כמו רוב הגלאים, YOLO הינו מבוסס-עוגנים (anchor-based), שהינן תיבות מלבניות קבועות ושונות זו מזו בצורתן. לכל עוגן מוקצה מקטע של פיצ'רים במפת המוצא של הרשת וכל הניבויים במקטע הזה מקודדים כסטיות (offsets) ביחס לממדי העוגן. כפי שניתן לראות באיור 9.2, הפיצ'רים של כל תא מרחבי במפת המוצא מחולקים למקטעים על פי העוגנים (שלושה עוגנים במקרה הזה).

ניבוי הוא מסגרת שמרכזה נמצא בנקודה כלשהי בשטח התא (השקול לריבוע של מספר פיקסלים בתמונה המקורית). ההסטה המדויקת של מרכז המסגרת ביחס לתא ניתנת על ידי שני הפיצ'רים הראשונים ברצף. לוג הממדים של הקופסא (ביחס לממדי העוגן) ניתן באמצעות שני הפיצ'רים הבאים ברצף. הפיצ'ר החמישי לומד את מידת ה-objectness, כפי שהוסברה לעיל. שאר הפיצ'רים ברצף של העוגן הנ"ל הם ההסתברויות המותנות לכל מחלקה (אם אוסף הנתונים מכיל 80 מחלקות, יהיו 80 פיצ'רים כאלה). על מנת לקבל רמת ודאות סופית, יש להכפיל את מדד ה-objectness במדד ההסתברות המותנה לכל מחלקה.

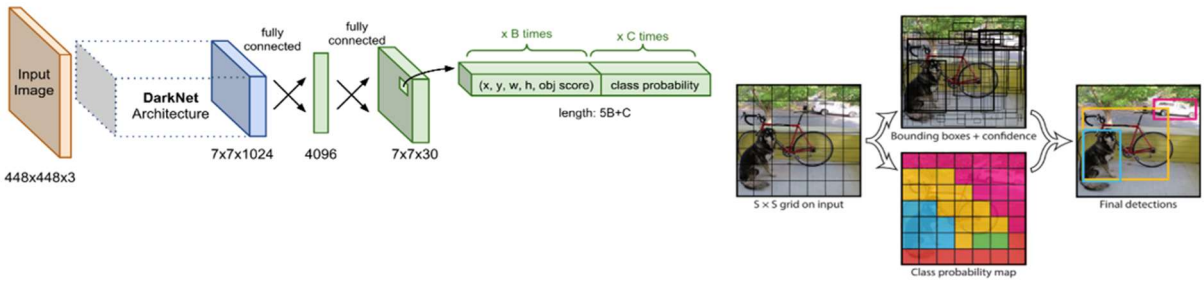



איור 9.7 ראש YOLO.

## YOLOv1

מודלי YOLO מבוססים על גרסאות Backbone הנקראות Darknet ומשמשות לעיבוד פיצ'רים מתוך התמונה, ומראש detection המקבל את הפיצ'רים האלה ומתאמן לייצר מהם ניבויים למסגרות סביב אובייקטים.

המודל מחלק את התמונה לרשת בעלת $S \times S$ משבצות, כאשר כל משבצת מנבאת $N$ מסגרות של אובייקטים בשיטת העוגנים, כאמור לעיל. כל ניבוי כולל את מספר ערכים: הסטת הקואורדינטות $x, y$ של מרכז המסגרת ביחס למשבצת, הגובה והרוחב של המסגרת, ורמת ה-objectness, כפי שהוסברה לעיל. בנוסף, כל מסגרת מבצעת גם סיווג, כלומר מנבאת את רמת הוודאות של השתייכות האובייקט לכל אחת מהמחלקות האפשרויות. החידוש באלגוריתם נעוץ בעובדה שחיזוי המסגרות וסיווגן לאובייקטים נעשה במקביל, ולא באופן דו-שלבי. הרעיון הוא להתייחס לסוג האובייקט כעוד פיצ'ר שהרשת מנסה לחזות בנוסף למיקום וגודל של המסגרת.



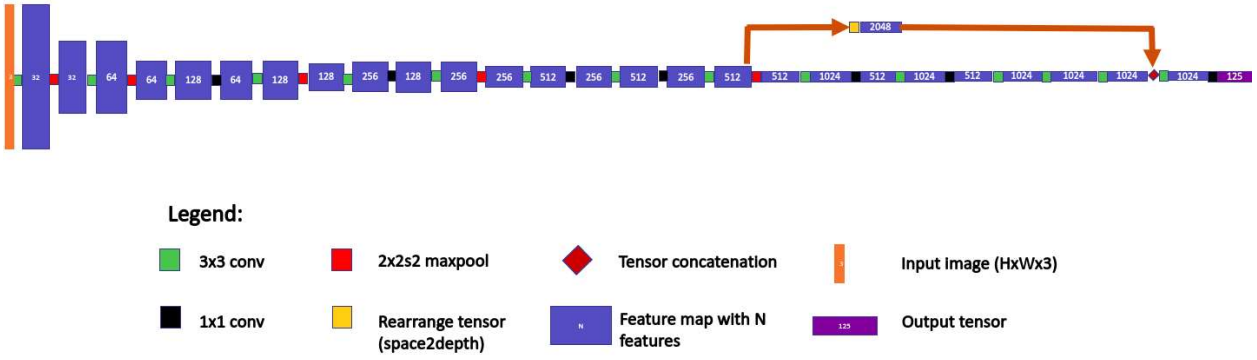

איור 9.8 ארכיטקטורת YOLO.

הגרסה הראשונה של ארכיטקטורת YOLO כללה שכבת fully connected שהוסרה בגרסאות הבאות. בנוסף, פונקציית ה-LOSS וסדר התכונות של המסגרות במוצא הרשת השתנו, אבל הרעיון נותר זהה.

## YOLOv2

גרסה זו משתמשת ברשת Backbone הנקראת Darknet19 ובה 19 קונבולוציות. המודל כולו כולל 22 קונבולוציות (מלבד 5 שכבות ה-MAXPOOL המקובלות על מנת למצוא את הפיצ'רים), ועוד מסלול עוקף בסמוך לסוף הרשת המחזק את יכולת העיבוד. הכותב של המאמר המקורי, רדמון, נהג לפתח גם גרסאות "Tiny" לכל מודל. הווריאנט YOLOv2 Tiny מכיל רשת Backbone קצרה יותר וללא מסלול עוקף ומבנהו לינארי ופשוט מאוד. ביצועיו אמנם נמוכים יותר אך הוא מהיר מאוד (207 תמונות לשנייה לעומת 67 של מודל YOLOv2, על מעבד TitanX).

מעניין לציין ש-YOLOv2 הוא המודל הראשון שאומן על תמונות בממדים משתנים, תהליך המשפר את דיוק המודל.



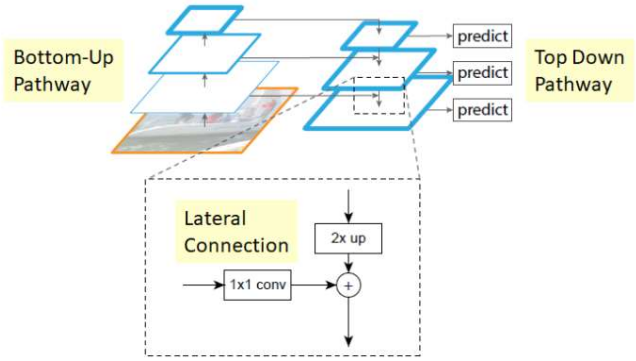

איור 9.9 ארכיטקטורת YOLOv2.

## YOLOv3

כל דור נוסף של YOLO הציג חידושים ארכיטקטוניים שהגדילו את מורכבות החישוב וגודל המודל ושיפרו את ביצועיו. גרסה מספר 3 מבוססת על רשת Backbone גדולה בהרבה שנקראת Darknet53 ובה, כפי שעולה משמה, 53 קונבולוציות. כמו כן הרשת מכילה צוואר של ארכיטקטורת Feature Pyramid Network (FPN) בעלת שלושה ראשי גילוי, כאשר כל אחד מהם הוא בעל רזולוציה שונה ($38 \times 38$, $19 \times 19$ ו-$76 \times 76$).

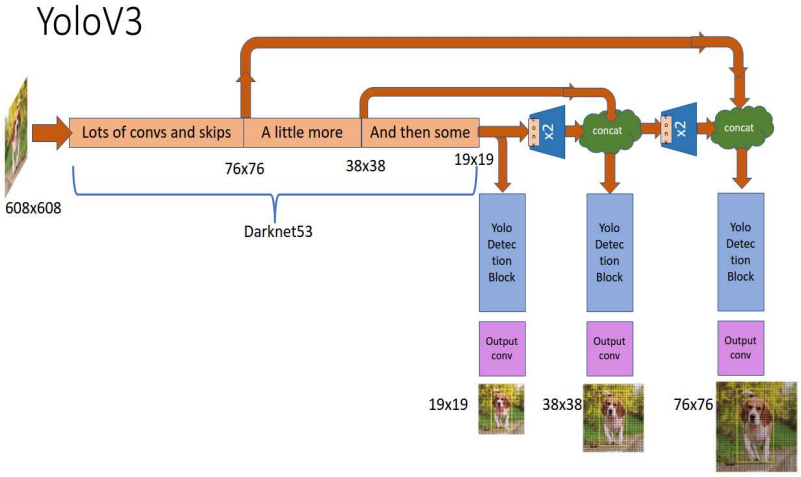
ארכיטקטורת FPN היא תוספת בקצה ה-Backbone, אשר מגדילה חזרה באופן הדרגתי את מפת הפיצ'רים. תוספת זו נועדה ליצור מסלול Top Down במקביל למסלול ה-Feed Forward ברשת ה-Backbone, שבנוי למעשה בצורת Bottom Up, בו ככל שמתקדמים בשכבות ה-Backbone מתבצע עיבוד ברמה גבוהה יותר והרזולוציה המרחבית של מפת הפיצ'רים הולכת וקטנה. בעזרת השימוש ברשת ה-FPN המודל לומד לנצל את המיטב משני עולמות: הוא משתמש במידע שטמון במפת הפיצ'רים הגדולה, שהיא אמנם פחות מעובדת אך בעלת פיצ'רים ברזולוציה מרחבית גבוהה, ובנוסף הוא מנצל גם את המידע ממפת הפיצ'רים הקטנה, שהיא אמנם בעלת פיצ'רים ברזולוציה מרחבית נמוכה, אך עם זאת היא מעובדת יותר.

לאחר כל הגדלה של מפת הפיצ'רים מתבצע חיבור בין התוצאה לבין מפת פיצ'רים קדומה יותר בממדים זהים (מתוך ה-Backbone). זאת בדומה לחיבורים העקיפים ברשת ResNet המסייעים להתכנסות האימון. השכבות השונות של רשת FPN מאפשרות לגלאי למצוא מיקום מדויק יותר של האובייקט ברזולוציות השונות, מה שמעניק לרשת זו את היכולת להבחין באובייקטים קטנים בתמונה גדולה.




איור 9.10 ארכיטקטורת Feature Pyramid Network (FPN), המשלבת מסלול top down לאחר ה-bottom up.

לראש המודל של YOLOv3 יש מספר ענפי detection, אשר כל אחד מהם פועל על מפת פיצ'רים ברזולוציה שונה ובאופן טבעי מתמחה בגילוי אובייקטים בגודל שונה (הענף בעל הרזולוציה הנמוכה מתמחה בגילוי אובייקטים גדולים, והענף בעל הרזולוציה הגבוהה מתמחה בגילוי אובייקטים קטנים).




איור 9.11 ארכיטקטורת YOLOv3.

**YOLOv4**

רשת YOLOv4 היא בעלת ראש זהה לזה של שתי הגרסאות הקודמות, אך ה-Backbone שונה ומורכב יותר. הוא נקרא CSPDarknet53 כאשר CSPNET היא קיצור של Cross-Stage Partial Network. רשת זו מפצלת מפות פיצ'רים לטובת קונבולוציה בחלקים ואיחוד מחדש. פיצול זה מאפשר, כמוזכר במאמר המקורי, חלחול טוב יותר של הגרדיאנטים בשלב האימון. בנוסף, נעשה ברשת זו שימוש בפונקציית אקטיבציה הנקראת Mish (ולא Leaky ReLU כמו בגרסאות הקודמות).

אנקדוטה מעניינת – החל מגרסה זו התצורות אינן של היוצר המקורי ג'וזף רדמון, שהחליט לפרוש ממחקר ראייה ממוחשבת בגלל שיקולים אתיים של שימושים צבאיים או שימושים הפוגעים בפרטיות.

**YOLOv5**

רשת YOLOv5 מוסיפה עוד שכלולים על רשת ה-Backbone על פני זו של הדור הקודם, ומציגה אופרטור חדש המארגן פיקסלים סמוכים בתמונת בממד הפיצ'רים. אופרטור זה דואג לכך שהכניסה לרשת היא לא בעומק 3 פיצ'רים כמקובל (RGB), אלא 12 פיצ'רים, תוך הקטנת הממד המרחבי. באופן זה הרשת מתאמנת לעבד תמונות ברזולוציה גבוהה, ואף לזהות אובייקטים גדולים בקלות רבה יותר, שכן שדה התמך (receptive field) של הקונבולוציות מכיל מידע משטח תמונה גדול יותר.

### 9.1.8 DE:TR: Object Detection with Transformers

https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

/https://machinelearningmastery.com/object-recognition-with-deep-learning

## 9.2 Segmentation

### 9.2.1 Introduction to Segmentation

אחד האתגרים הכי משמעותיים בעולם הראייה הממוחשבת הוא זיהוי אובייקטים בתמונה והבנת המתרחש בה. אפשרות אחת, שדנו עליה באריכות בפרק הקודם היא Object Detection, כאשר המטרה היא ליצור מודל שיסמן מלבנים (bounding boxes) מסביב לאובייקטים השונים בתמונה. אחת הבעיות בגישה זו היא חוסר ה"טבעיות" שבה – אנחנו כבני אדם לא תופסים אובייקטים באופן זה של מלבנים, אלא מסתכלים על אובייקט בהתאם לקווי המתאר שלו. המשמעות של זה היא כפולה: ראשית מבחינה מעשית, יתכנו מצבים בהם נרצה לדעת את הגבולות המדויקים של כל אובייקט ולא רק מה המלבן המקיף אותו (למשל – בניתוחים של צילומים רפואיים). שנית, מבחינה אלגוריתמית יתכן ואימון של מודל ביחס ל-bounding box יכול להטעות אותו – אנחנו דורשים מהמודל להקיף אובייקט באופן כזה שהמלבן לרוב יכיל גם רקע, וזה יכול להיות מבלבל.

אפשרות אחרת לזהות אובייקטים הינה ביצוע סגמנטציה, כלומר, התאמת label לכל פיקסל בתמונה. בתהליך הסגמנטציה מבצעים חלוקה/בידול בין עצמים שונים בתמונה המצולמת באמצעות סיווג ברמת הפיקסל, כלומר כל

פיקסל בתמונה יסווג וישויך למחלקה מסוימת. מודל שמבצע סגמנטציה יכול לספק הבנה יותר טובה של התמונה מאשר Object Detection, אך יש לכך גם חסרונות – המודל צריך להיות יותר גדול ומורכב, יותר קשה לאמן אותו (וממילא יכולות להיות יותר שגיאות) וליצור דאטה מתויג ברמת הפיקסל זו משימה מאוד יקרה.

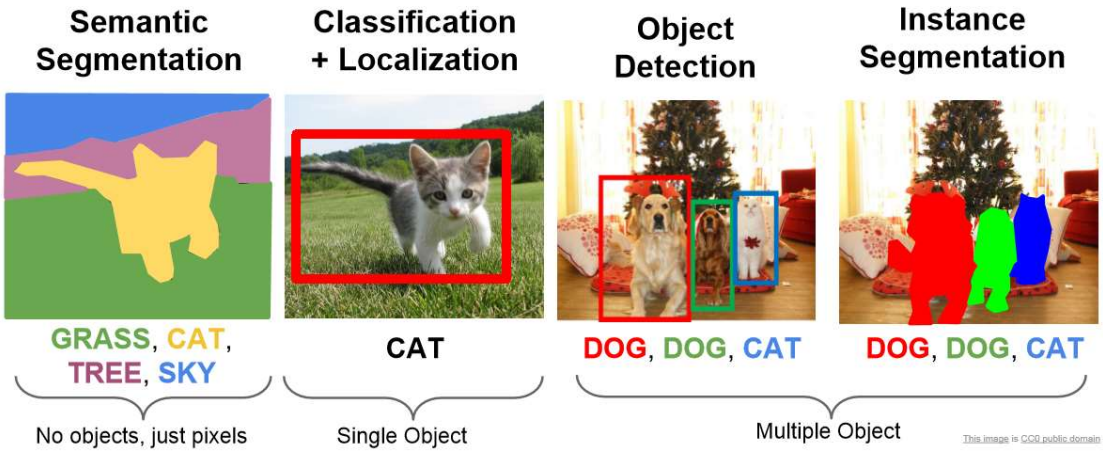
ישנם שימושים מגוונים באלגוריתמים של סגמנטציה – הפרדה של עצמים מסוימים מהרקע שמאחוריהם, מציאת קשרים בין עצמים ועוד. לדוגמא, תוכנות של שיחות ועידה, כמו zoom, skype, teams וכדו', מאפשרות בחירת רקעים שונים עבור המשתמש, כאשר מלבד הרקע הנבחר רק הגוף של המשתתף מוצג בווידאו. הפרדת גוף האדם מהרקע והטמעת רקע אחר מתבצעות באמצעות אלגוריתמים של סגמנטציה. דוגמא נוספת – ניתן לזהות בתמונה אדם, כלב, וביניהם רצועה, ומכך ניתן להסיק שתוכן התמונה הוא אדם מחזיק כלב בעזרת רצועה. במקרה זה, הסגמנטציה נועדה למצוא קשר בין עצמים ולהבין את המתרחש.

קיימים שני סוגים עיקריים של סגמנטציה:

Semantic segmentation (חלוקה סמנטית) - חלוקה של כל פיקסל בתמונה למחלקה אליה העצם אותו הוא מייצג שייך. למשל, פיקסל יכול להיות משויך לכלי רכב, בן אנוש, מבנה וכדו'.

Instance segmentation (חלוקה מופעית) - חלוקה של פיקסל בתמונה למופע של אותה מחלקה אליה העצם אותו הוא מייצג שייך. במקרה זה, בתמונה בה מופיעים מספר כלי רכב, תבוצע חלוקה של כל פיקסל לאיזה כלי רכב אותו פיקסל מייצג – מכונית 1, מכונית 2, אופנוע 1, משאית 1 וכדו'.

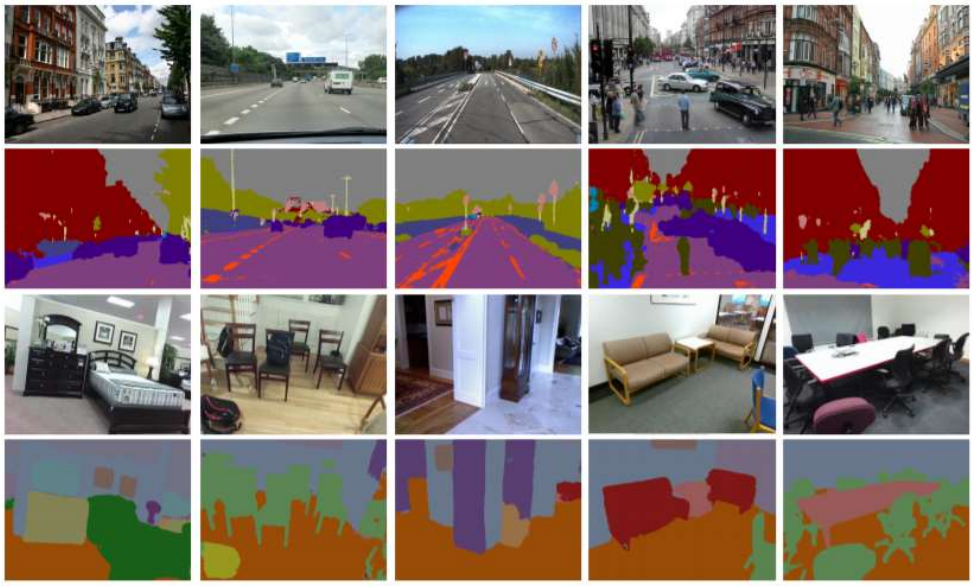
ההבדל העיקרי בין שני סוגים אלו הוא ברמת עומק המיפוי של פיקסל - המיפוי עשוי לסווג את הפיקסל למחלקה כלשהי, או לעצם ספציפי בתמונה. עומק המיפוי משליך גם על עלות המיפוי. החלוקה הסמנטית מבוצעת ישירות, בעוד שהחלוקה המופעית דורשת בנוסף ביצוע של זיהוי אובייקטים כדי לסווג מופעים שונים של המחלקות.




איור 9.15 זיהויים של אובייקטים בתמונה: Classification, Object Detection and Segmentation (semantic and instance). במשימת Semantic Segmentation המטרה היא לסווג כל פיקסל לאיזה class הוא שייך. גם במשימת Instance Segmentation המטרה היא לסווג כל פיקסל ל-class מסוים, אך בנוסף יש גם להתאים בין כל פיקסל לבין אובייקט ספציפי.

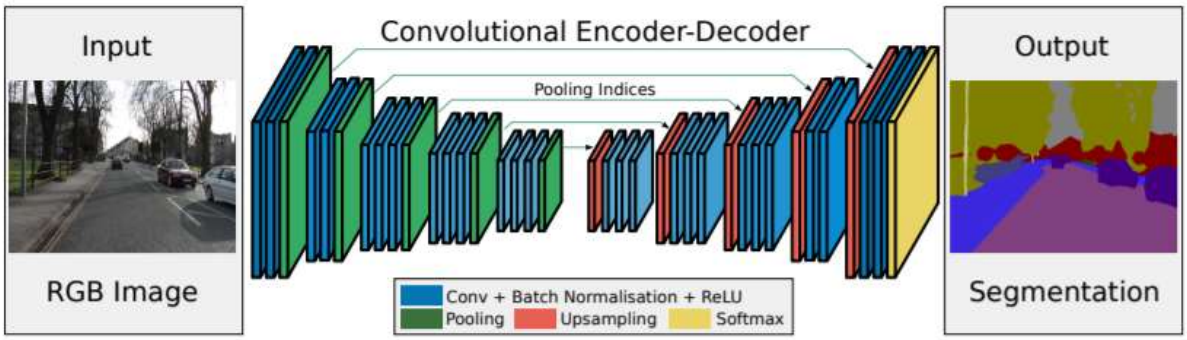
### 9.2.2 SegNet neural network

רשת SegNet הינה רשת קונבולוציה עמוקה, שמטרתה לבצע חלוקה סמנטית (Semantic segmentation) לתמונת הקלט. בתחילה הרשת פותחה להבנה של תמונות חוץ (למשל כביש עם מכוניות ובצדדים בתים והולכי רגל) ותמונות פנים (למשל חדר עם מיטה וכיסאות). הרשת נבנתה מתוך מטרה להיות יעילה בהיבטי זיכרון וזמן חישוב, תוך שמירה על דיוק מעשי.



איור 9.16 סיווג סמנטי בעזרת רשת SegNet עבור תמונות פנים וחוץ.

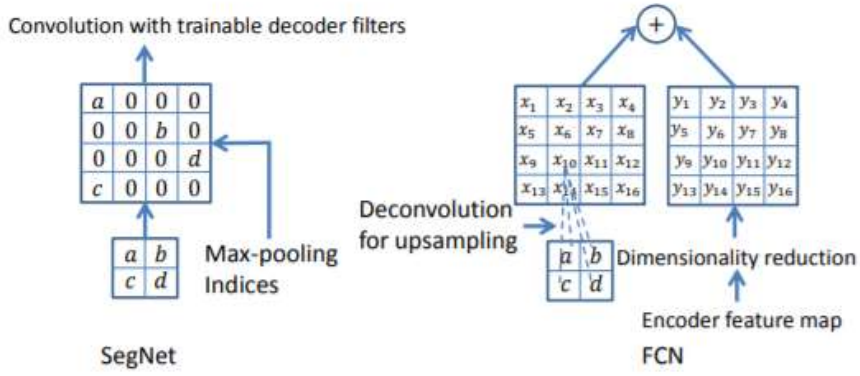
הרשת בנויה כארכיטקטורת מקודד-מפענח (encoder-decoder). המקודד מורכב מ-13 השכבות הראשונות של רשת VGG16, ומטרתו לחלץ את מפת המאפיינים (feature maps). לכל שכבת קידוד יש שכבת פיענוח תואמת ומכאן שגם רשת המפענח מכילה 13 שכבות. מטרת המפענח היא לבצע up-sampling ולייצר תמונת מאפיינים בגודל המקורי. את מוצא המפענח מעבירים דרך מסווג SoftMax, המתאים את המחלקה בעלת ההסתברות הגבוהה ביותר לכל פיקסל בנפרד.




איור 9.17 ארכיטקטורת רשת SegNet.

המקודד מורכב משכבות קונבולוציה, אחריהן שכבות נרמול (batch normalization) ושכבות אקטיבציה מסוג ReLU. לאחר שכבות אלו, ישנה שכבת max-pooling המבצעת subsampling, בעלת גודל חלון $2 \times 2$ ומרווח (stride) בגודל 2.

החידוש ברשת זו הוא אופן הפעולה של המפענח. בשונה מרשתות אחרות, בהן תהליך ה-up-sampling גורר ביצוע חישובים עבור הפיענוח, כמתואר באיור הבא, הרעיון ברשת זו הוא שמירת מיקומי ערכי המקסימום הנבחרים מכל רביעייה. רק הערכים שנבחרו כמקסימום ישוחזרו בתהליך ושאר הערכים יתאפסו.




איור 9.18 שכבת פענוח ברשת SegNet לעומת רשת FCN.

ארכיטקטורה זו מביאה את הרשת לביצועים טובים בהיבטי זמן חישוב, על חשבון פגיעה מסוימת בדיוק הרשת. למרות זאת, ביצועי הרשת מתאימים לשימושים פרקטיים והפגיעה בדיוק קטנה מאוד.

בדומה למקודד, לאחר כל שכבת up-sampling במפענח, יופיעו שכבות קונבולוציה, שכבות נרמול ושכבות אקטיבציה מסוג ReLU. את מוצא המפענח מעבירים דרך שכבת SoftMax המבצעת סיווג ברמת הפיקסל. מוצא הרשת, שהוא גם מוצא שכבת ה-SoftMax, הינו מטריצת הסתברויות, כאשר עבור כל פיקסל יש וקטור באורך K, כאשר K הוא מספר המחלקות לסיווג. כמובן שהסיווג מתבצע בהתאמה להסתברות הגבוהה ביותר המתאימה לכל פיקסל.

אימון הרשת בוצע על בסיס מידע קטן יחסית של 600 תמונות דרך צבעוניות בגודל $360 \times 480$, שנלקחו מבסיס המידע CamVid road scene dataset. סט האימון הכיל 367 תמונות וסט הבדיקה הכיל את 233 התמונות הנותרות. המטרה הייתה לזהות בתמונות אלה 11 מחלקות (דרך, בניין, מכונית, הולכי רגל וכדומה). כל תמונה עברה נרמול מקומי לערך הניגודיות של תמונת הקלט לפני הכניסה לרשת. האימון בוצע בשיטת ה-SGD, עם קצב למידה קבוע שערכו 0.1 ומומנטום שערכו 0.9. האימון נמשך עד ששגיאת האימון התכנסה. לפני כל epoch סט האימון עורבב וחולק ל-mini-batch של 12 תמונות. פונקציית המחיר הינה cross-entropy.

לעיתים, נדרש לבצע איזון-מחלקות (class balancing). מונח זה מתאר מצב בו קיים שוני גדול בין כמות הפיקסלים המשויכים לכל מחלקה, למשל כאשר קיימת הטיה מסוימת - סצינה שברובה מכילה בניינים / דרכים. במצב זה, יבוצע משקול מחודש לפונקציית השגיאה, באמצעות תהליך "איזון התדירות החציונית" (median frequency balancing). התהליך ממשקל מחדש את המחלקות בפונקציית המחיר, באופן יחסי לחציון של תדירות הופעות המחלקות בכל סט האימון, תוך חלוקה בתדירות הופעת המחלקה:

$$\alpha_c = \frac{\text{median freq}}{\text{freq(c)}}$$

משקול זה משנה את היחסים בפונקציית המחיר כך שהתרומה של כל המחלקות לפונקציית המחיר תהיה שווה. לכן, הוא מעניק למחלקות הגדולות יותר משקל נמוך יותר ולמחלקות הקטנות משקל גבוה יותר.

### 9.2.3 Atrous Convolutions (Dilated Convolutions)

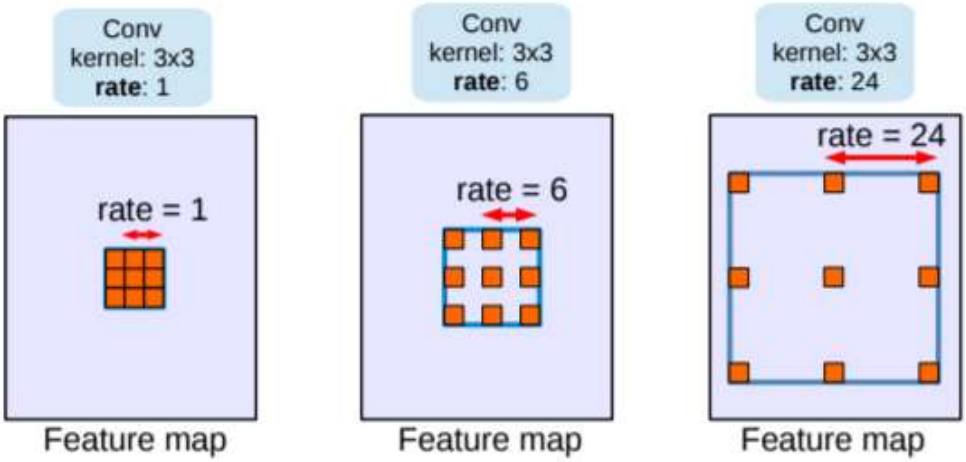
המונח Atrous מקורו בשפה הצרפתית – "à trous", שמשמעותו "עם חורים". לכן, ניתן לתרגם את המונח Atrous convolution כ-"קונבולוציה מחוררת", ובמשמעות מעט יותר מתאימה – קונבולוציה מרווחת (או בשם אחר – Dilated convolution – קונבולוציה מורחבת).

בטכניקת קונבולוציה זו, יש שימוש בפרמטר נוסף בנוסחת הקונבולוציה – dilation rate. פרמטר זה מסמן את המרווח בין כל איבר בגרעין הקונבולוציה (הרחבה על פרמטר זה – בפרק 5.1.2). נוסחת הקונבולוציה עבור המקרה החד ממדי יחד עם פרמטר ההתרחבות $r$ ניתנת לתיאור באופן הבא:

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k] w[k]$$

עבור $r = 1$ מתקבלת הקונבלוציה הרגילה, וכאשר ישנו dilation של $r > 1$, מתקבלת קונבולוציה מרווחת בפקטור $r$. יתרונה של קונבולוציה נעוץ בכך שעבור אותו קרנל ועבור אותה כמות חישובים, מרחיבים את ה- field of view (FoV) של הקונבולוציה.

ניתן לראות את הרחבת ה-field of view באיור הבא:




איור 9.19 קונבולוציה מרווחת דו-ממדית, עם קרנל בגודל $3 \times 3$ ופרמטר התרחבות $r = 1,6,24$. בהתאם לכל פרמטר מתקבל field-of-view בגודל שונה – $3 \times 3, 16 \times 16, 49 \times 49$ בהתאמה.

### 9.2.5 Deeplab V3+

בכדי להבין את המוטיבציה של המאמר, נתבונן על הבעיות המובנות שיש בשימוש ברשתות קונבולוציה עבור משימות של Semantic Segmentation:

1. **צמצום ברזולוציית הפיצ'רים:** בכדי לבצע סיווג לכל פיקסל, יש לאמן מודל שידע להסתכל על תמונה ברמת הפיקסל. דבר זה יכול להיות בעייתי מכמה סיבות כשמשתמשים ברשתות קונבולוציה:
   - ראשית, רשתות אלו מנסות ללמוד את הפיצ'רים השונים שבתמונה. הפיצ'רים הם מאפיינים מרחביים, וכאשר מנסים לחלץ אותם מתמונה, נוצרת באופן אינהרנטי ירידה ברזולוציית הפיצ'רים.
   - בעיה נוספת נובעת מכך שאימון של רשת סגמנטציה הוא דבר מורכב ויקר בהרבה מאשר מודל של קלסיפיקציה, ולכן כמעט תמיד משתמשים בטכניקה של Transfer learning, כלומר לוקחים רשת שאומנה על משימת קלסיפיקציה ועושים לה fine tune עבור סגמנטציה. כמובן שהיתרון העיקרי בטכניקה זו הוא ההפחתה המשמעותית בעלויות. עם זאת, רוב הרשתות אומנו על ImageNet – דאטה שמכיל תמונות ואובייקטים רבים, אך התמונות בו בעלות רזולוציה נמוכה יחסית, מה שעלול להוביל לרזולוציית פיצ'רים נמוכה.

2. **הימצאות של אובייקטים בגדלים שונים:** הופעתם של אובייקטים בגדלים שונים השייכים לאותה מחלקה יכולים להקשות על הרשת. ברשתות של סיווג למשל זה פחות מפריע, כיוון שכל מה שמעניין זה רק איזה אובייקט קיים, ומכל הרקע אפשר פשוט להתעלם. ברשת שצריכה לבצע סגמנטציה לכל הפיקסלים בתמונה לא ניתן להתעלם מאזורים מסוימים, ולכן במקרים בהם יש אובייקטים שונים השייכים לאותה מחלקה הרשת יכולה להתקשות לשייך אותם למחלקה זהה.

3. **הפחתה בדיוק המקומי בגלל השונות של רשתות קונבולוציה עמוקות:** רשתות קונבולוציה יודעות לזהות דפוסים ולחלץ פיצ'רים בצורה טובה מאוד, אך בו בזמן הן מאבדות את המיקום המדויק של כל אובייקט בתמונה המקורית.

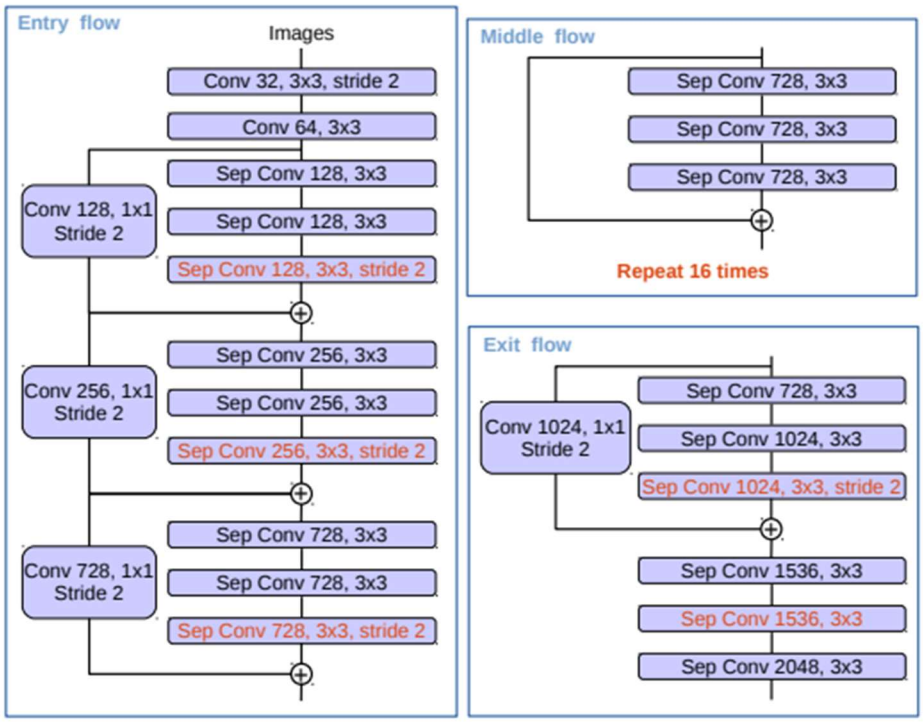
אלגוריתם Deeplab V3+ בא להתמודד עם בעיות אלה, כאשר הוא מבוסס על חיבור של שתי ארכיטקטורות מרכזיות, יחד עם מספרים שיפורים. שני החלקים של האלגוריתם הם Modified Aligned Xception ו- Encoder-Decoder with Atrous Convolution: החלק הראשון, המשמש ב-backbone, הוא שיפור של רשת קלסיפיקציה מהירה הנקראת Aligned Xception. לאחר מכן יש חלק הבנוי מ-encoder-decoder המבוססים על הרעיון של Atrous convolution (המוזכר בפרקים קודמים). נרחיב כעת על כל אחד מהחלקים:

**Modified Aligned Xception**

הכותבים השתמשו ברשת Xception כ-backbone משתי סיבות עיקריות: א. היא ידועה בביצועים מהירים ויחסית טובים (בנוסף לעלויות מאוד נמוכות). ב. היא הוכיחה את עצמה כ-backbone טוב במשימת Object Detection. כותבי המאמר ביצעו מספר שינויים קלים ברשת שיתאימו ספציפית למשימת סגמנטציה:

- במקום max pooling, מבצעים atrous separable convolution עם stride 2.

- לאחר כל ביצוע של Depthwise conv3x3, מבצעים נרמול ולאחר מכן מעבירים את המוצא באקטיבציה.




איור 9.20 Modified Aligned Xception המשמש כ-backbone עבור משימת סגמנטציה.

### Encoder-Decoder with Atrous Convolution

לאחר ה-backbone מגיע החלק השני של הארכיטקטורה, המבוססת על קונבולוציה מרווחת (Atrous convolution), וכפי שנכתב לעיל, מדובר בכלי מאוד חזק. כלי זה מאפשר ללכוד מידע קונטקסטואלי עשיר ביותר על ידי איגוד של פיצ'רים בגדלים שונים. כאשר משתמשים במרווחים שונים ניתן לקבל ראייה מרחבית משתנה ולא קבועה, ולמעשה איגוד התמונות לתמונה אחת מספק לרשת יכולה גבוהה מאוד של הבנת המתרחש. באמצעות זאת, הרשת מסוגלת ליצור קשרים בין אובייקטים שונים ואף בין אותו אובייקט שמופיע בגדלים שונים בתמונה. בפרט, כותבי המאמר התבססו על הרעיון של (ASPP) **Atrous Spatial Pyramid Polling**, תוך שהם משלבים בינו לבין פעולה שנקראת **Separable Convolution**, וכל זה יחד עם מנגנון של **encoder-decoder**, כפי שיוסבר בהרחבה.

נרחיב מעט על האינטואיציה שעומדת מאחורי ASPP: קונבולוציה רגילה מתרכזת בחלק מאוד ספציפי (בגודל קבוע) בתמונה, מה שעוזר לחלץ פיצ'רים, אך ההקשר המרחבי הרחב יותר נותר מוגבל. זאת לעומת שימוש ב-ASPP אשר מקנה הקשר מרחבי שונה וגדול יותר (בגלל הפרמטר של המרווח המשתנה). כלומר, כאשר מאגדים את התמונות לתמונה אחת, מתחזקת ההבנה הגלובלית שיש לרשת על התמונה. במילים אחרות – ב-ASPP כבר לא מדובר בהסתכלות על תמונה אחת אלא על הרבה זוויות שונות של אותו דבר, מה שעוזר לרשת להבין קשרים והקשרים בצורה יותר ברורה. נסתכל על דוגמה פשוטה:

איור 9.21.

אם הרשת "תסתכל" רק על הכלי המופיע בריבוע הצהוב, היא עלולה לחשוב בטעות שמדובר במכונית. אך ברגע שהרשת תרחיב את המבט ו"תתפוס" את המים מסביב ואת התמונה הרחבה יותר -- היא תקבל הבנה עמוקה יותר,

ותבין שמדובר בסירה ולא במכונית. דבר דומה יתרחש כאשר יהיה אובייקט שיופיע יותר מפעם אחת בתמונה אך בגדלים שונים (למשל כלב גדול וכלב קטן): האיגוד של התמונות עם המרווחים השונים עוזר לרשת להבין שמדובר באובייקטים שונים השייכים לאותו class.

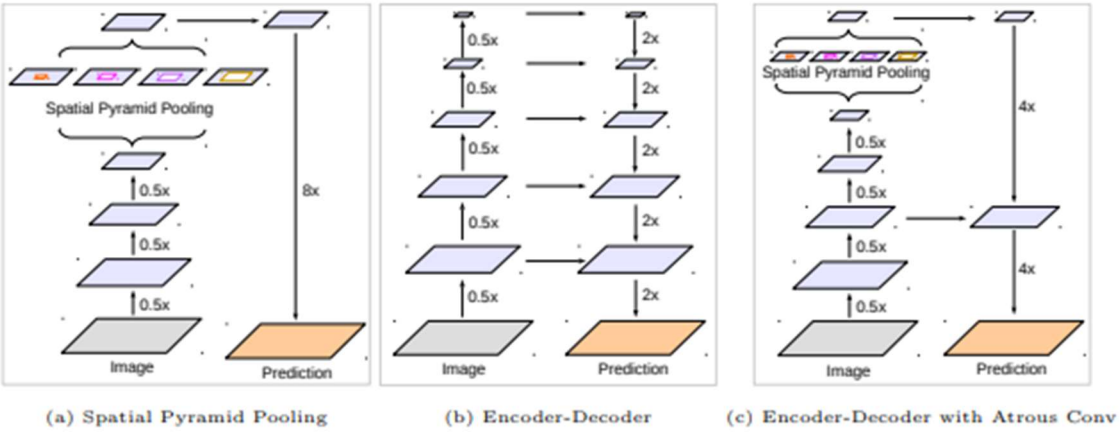
מבחינה טכנית, הפעולות שאנו מבצעים ב- ASPP על התמונה מורכבות מ- 4 קונבולוציות מרווחות, כאשר בכל אחת מהן מתקבלת תמונה באותו הגודל, אבל בעלת ראייה מרחבית שונה. לאחר כל קונבולוציה מתבצע נרמול ולאחר מכן המוצא עובר בפונקציית אקטיבציה לא לינארית (ReLU). בהמשך לביצוע הקונבולוציות, התמונה עוברת Pooling, כאשר מימושים שונים למאמר השתמשו ב-Adaptive Average Pooling. לאחר פעולת ה-Pooling, מתבצעת פעולת bilinear interpolation, המאפשרת להחזיר את התמונה לרזולוציה המקורית ובכך לבצע שרשור בין כל התמונות.

בנוסף לרעיון של ASPP, הרשת מציעה גם שימוש ב-Separable convolution, המכילה בתוכה שתי טכניקות עיקריות – **Depthwise convolution** ו-**Pointwise convolution**. בעוד ש-Depthwise convolution הינה פעולה המתבצעת על המרחב הדו-ממדי של התמונה (האורך והרוחב שלה), Pointwise convolution הינה פעולה המתבצעת על הערוצים של התמונה (ממד העומק – RGB).

מבחינה מתמטית, נניח שה-input feature maps הוא בגודל $F \in D_F \times D_F \times M$ ושה-output feature maps בגודל $G \in D_F \times D_F \times N$. במידה והיינו משתמשים בקונבולוציה רגילה, ה-convolution kernel צריך להיות בגודל של $K \in D_F \times D_F \times M \times N$, ולכן עלות החישוב תהיה $D_F \times D_F \times D_F \times D_F \times M \times N$.

אם לעומת זאת נבחר להשתמש ב-separable convolution, ראשית נבצע Depthwise conv, בה פילטר הקונבולוציה הוא מממד $K \in D_K \times D_K \times M$. כלומר, במצב זה אנחנו פועלים רק על ממד התמונה ולא על ממד הערוץ. לכן העלות תהיה $D_F \times D_F \times D_K \times D_K \times M$. לאחר מכן נבצע pointwise conv בה אנחנו פועלים בממד הערוץ. פילטר הקונבולוציה במקרה זה יהיה בגודל $K \in 1 \times 1 \times M$, ולכן עלות החישוב תהיה $D_F \times D_F \times M \times N$. אם נחבר את שניהם עלות החישוב תהיה: $D_F \times D_F \times D_K \times D_K \times M + D_F \times D_F \times M \times N$, וניתן לראות שבניגוד לקונבולוציה הרגילה, נוצר חיסכון משמעותי בעלויות החישוב ובמספר הפרמטרים.

הארכיטקטורה של הרשת מבוססת על שימוש ב-Atrous **יחד** עם Separable, וכפי שהמנגנון נקרא במאמר – Atrous separable convolution. קונבולוציות אלו נכנסות לתוך ארכיטקטורת **encoder- decoder**, כמתואר באיור הבא:




איור 9.22. ארכיטקטורת DeepLab v3.

ארכיטקטורת ה-encoder דומה מאוד לזו שהייתה בשימוש ברשת Deeplab V3, בה הוא היה מחולק ל-3 חלקים:

1. רשת backbone שמטרתה לחלץ פיצ'רים מהתמונה, וכאמור בחרו להשתמש ב-Aligned Xception. בנוסף לכך, רשת ה-backbone מחזירה גם low level feature שאותו היא מקבלת לאחר ה-block הראשון של ה- entry flow.
2. לאחר מכן מתבצע ASPP על התמונה.
3. לבסוף מבצעים קונבולוציה 1x1 על מנת לעבור למספר הערוצים הרצוי.

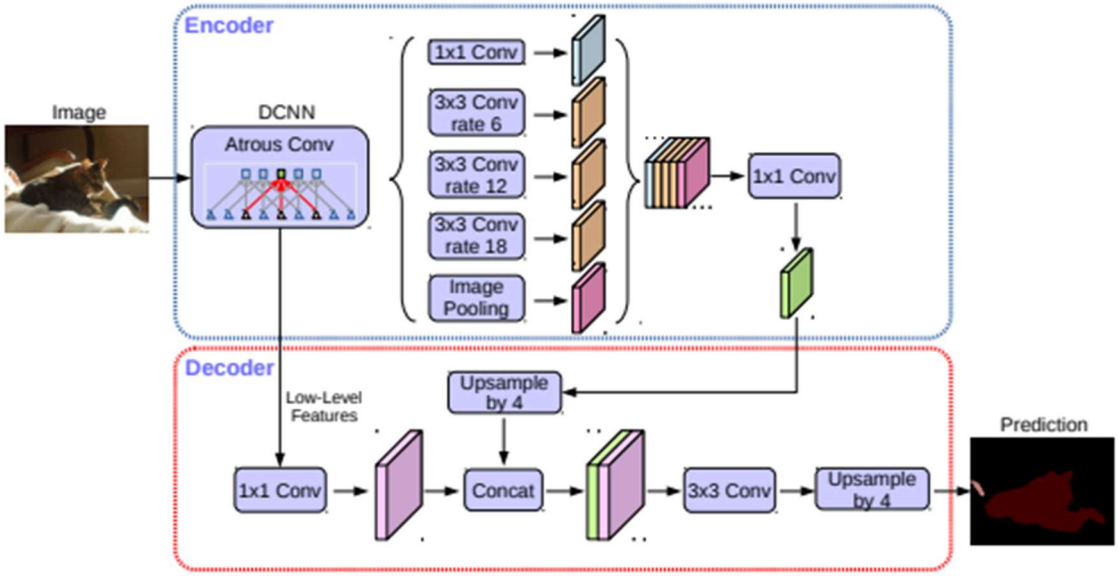
לעומת ה-encoder, ה-decoder עבר שינוי דרסטי. כותבי המאמר ראו שהשחזור של התמונה לגודל המקורי לוקה בחסר ופוגע ביכולת הדיוק של הרשת, ולכן הם ביצעו מספר שינויים, המתבססים על שלוש טכניקות:

1. שימוש ב- low level feature לביצוע השחזור.
2. שימוש ב-up sampling בהדרגה.
3. הפעלת קונבולוציה 3x3 על מנת לקבל אובייקטים חדים וברורים יותר.

באופן יותר מפורט, המעבר של הדאטה ב-decoder נראה כך:

- מתקבלת תמונה מה-encoder, ועליה ניתן לבצע up sampling פי 4 (לדוגמה, אם התמונה היא בגודל 32x32, במוצא היא תהיה בגודל של 128x128).
- לאחר מכן, יש לבצע קונבולוציה 1x1 על הקלט שהתקבל מה-low level feature, על מנת להקטין את גודל ממד הערוץ, והפלט משורשר יחד עם התמונה עצמה.
- על הפלט המשורשר מתבצעת קונבולוציה 3x3 כדי לחזק את האובייקטים שבתמונה, ובנוסף מספר הערוצים שבתמונה משתנה למספר הקטגוריות לנו ב-dataset.
- לבסוף יש לבצע up sampling נוסף לתמונה בגודל פי 4 על מנת לחזור לגודל המקורי של התמונה.

בסף הכל, הארכיטקטורה מקצה לקצה נראית כך:




איור 9.23. ארכיטקטורת DeepLab v3.

נציין שמבחינת ביצועים Deeplab V3+ היה למעשה state of the art חדש, עם Mean IoU 89 עבור PASCAL VOC 2012 Test Set ו-Mean IoU 82.1 עבור Cityscapes dataset.

## 9.3 Face Recognition and Pose Estimation

### 9.3.1 Face Recognition

אחד מהיישומים החשובים בראייה ממוחשבת הינו זיהוי פנים, כאשר ניתן לחלק משימה זו לשלושה שלבים:

1. Detection – מציאת הפרצופים בתמונה.
2. Embedding – מיפוי כל פרצוף למרחב חדש, בו המאפיינים שאינם קשורים לתיאור הפנים (למשל: זווית, מיקום, תאורה וכדו') אינם משפיעים על הייצוג.
3. Searching – חיפוש במאגר של תמונות למציאת תמונת פנים הקרובה לתמונת הפנים שחולצה מהתמונה המקורית.

גישה פשטנית, כמו למשל בניית מסווג המכיל מספר יציאות כמספר הפנים אותם רוצים לזהות, הינה בעייתית משתי סיבות עיקריות: ראשית יש צורך באלפי דוגמאות לכל אדם (שלא ניתן בהכרח להשיג). כמו כן, נצטרך ללמד את המערכת מחדש בכל פעם שרוצים להוסיף מישהו חדש. כדי להתגבר על בעיות אלו מבצעים "למידת מטריקה" (metric learning) בה מזקקים מאפיינים של פנים ויוצרים וקטור יחסית קצר, למשל באורך 128, המכיל את האלמנטים המרכזיים בתמונת הפנים. כעת נפרט את שלושת השלבים:

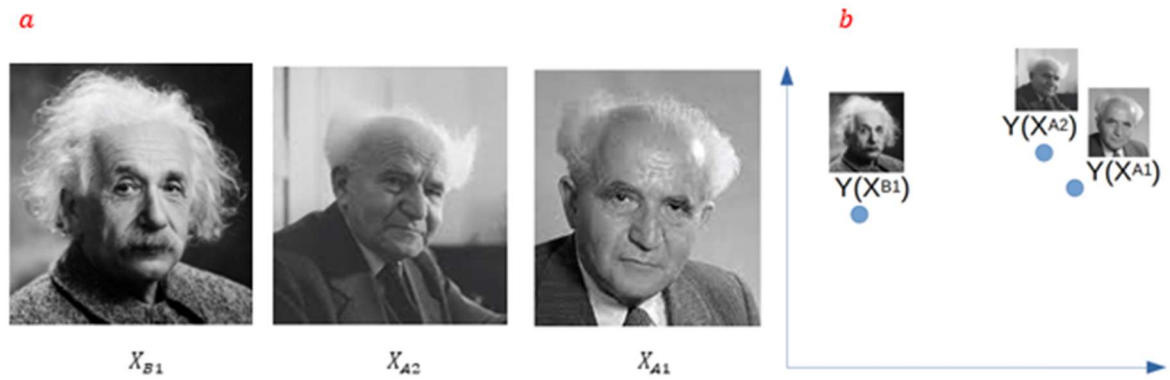
1. מציאת פנים.

כדי למצוא פרצופים בתמונה ניתן להשתמש ברשתות המבצעות detection, כפי שתואר בפרק 9.1. שיטה מקובלת למשימה זו הינה Yolo, המבוססת על חלוקת התמונה למשבצות, כאשר עבור כל משבצת בוחנים האם יש בה אובייקט מסוים, מהו אותו אובייקט, ומה ה-bounding box שלו.

2. תיאור פנים.

כאמור, המשימה בתיאור פנים נעשית בעזרת metric learning, כאשר הרעיון הוא לזקק פנים לוקטור שאינו מושפע ממאפיינים שלא שייכים באופן מהותי לפנים הספציפיות האלה, כגון זווית צילום, רמת תאורה וכדו'. בכדי לעשות זאת יש לבנות רשת המקבלת פנים של בנאדם ומחזירה וקטור, כאשר הדרישה היא שעבור שתי תמונות של אותו אדם יתקבלו וקטורים מאוד דומים, ועבור פרצופים של אנשים שונים יתקבלו וקטורים שונים. למעשה, פונקציית ה-loss תקבל בכל פעם minibatch, ותעניש בהתאם לקרבה בין וקטורים של אנשים שונים וריחוק בין וקטורים של אותו אדם.

כעת נניח שיש לנו קלט $X$, המכיל אוסף פרצופים. כל איש יסומן באות אחרת – A,B,C, ותמונות שונות של אותו אדם יסומנו על ידי אות ומספר, כך שלמשל $X_{A1}$ זוהי התמונה הראשונה של אדם $A$ בסט הקלט $X$, וכמובן ש-$X_{A1}$ ו-$X_{A2}$ הן שתי תמונות של אותו אדם. באופן גרפי, בדו-ממד ניתן לתאר זאת כך (בפועל הווקטורים המייצגים פנים יהיו בממד גבוה יותר):




איור 9.24 a) דוגמאות מסט הפרצופים $X$. b) איך נרצה שהדאטה ימופה לממד חדש $Y$.

כאמור, נרצה לבנות פונקציית loss שמעודדת קירבה בין $X_{A1}$ ו-$X_{A2}$, וריחוק בין $X_{A1}$ ו-$X_{B1}$. פונקציית ה-loss מורכבת משני איברים, המודדים מרחק אוקלידי בין וקטורים שונים:

$$L = \sum_{X} \left\| Y(X^{Ai}) - Y(X^{Aj}) \right\| - \left\| Y(X^{Ai}) - Y(X^{Bj}) \right\|$$

כאשר האיבר הראשון ינסה להביא למינימום וקטורים של אותו אדם, והאיבר השני ינסה להביא למקסימום וקטורים של פרצופים שאינם שייכים לאותו אדם. כיוון שנרצה להימנע מקבלת ערכים שליליים, נוסיף פונקציית מקסימום. בנוסף, ניתן 'להרחיק' תוצאות של פרצופים שונים על ידי הוספת קבוע $k$, כך שהפרש בין המרחק של פרצופים של אנשים שונים לבין המרחק של פרצופים של אותו איש יהיה לפחות $k$:

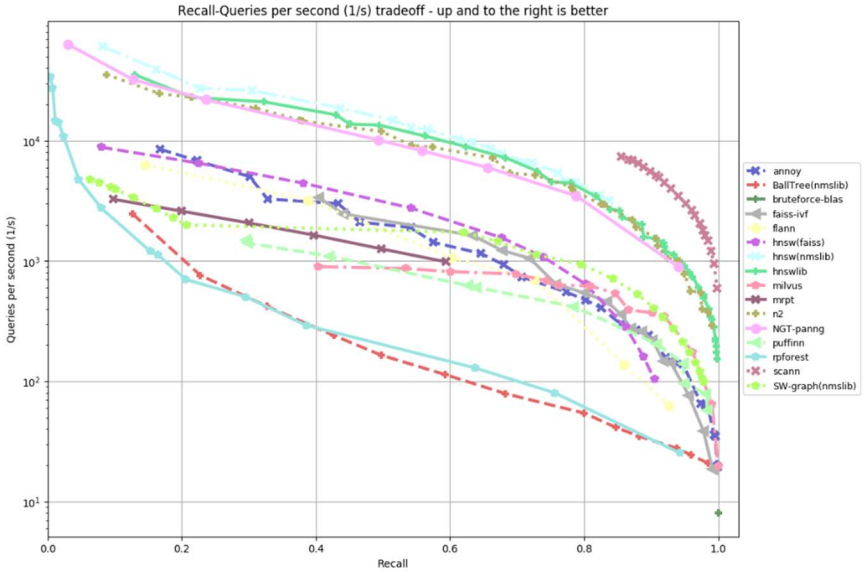
$$L = \sum_{X} max\left(\left\| Y(X^{Ai}) - Y(X^{Aj}) \right\| - \left\| Y(X^{Ai}) - Y(X^{Bj}) \right\| + k, 0\right)$$

loss כזה נקרא triplet loss, כיוון שיש לו שלושה איברי קלט – שתי תמונות של אותו אדם ואחת של מישהו אחר. כאמור, הפלט של הרשת הנלמדת צריך להיות וקטור המאפיין פנים של אדם, ומטרת הרשת היא למפות פרצופים

שונים של אותו אדם לווקטורים דומים עד כמה שניתן, ואילו פרצופים של אנשים שונים יקבלו וקטורים רחוקים זה מזה.

3. מציאת האדם

בשלב הקודם, בו התבצע האימון, יצרנו למעשה מאגר של פרצופים במרחב חדש. כעת כשיגיע פרצוף חדש, כל שנותר זה למפות אותו למרחב החדש, ולחפש במרחב זה את הווקטור הקרוב ביותר לוקטור המייצג את הפנים החדשות. בכדי לעשות זאת ניתן להשתמש בשיטות קלאסיות של machine learning, כמו למשל חיפוש שכן קרוב (כפי שהוסבר בחלק 2.1.3). שיטות אלו יכולות להיות איטיות עבור מאגרים המכילים מיליוני וקטורים, וישנן שיטות חיפוש מהירות יותר (ובדרך כלל המהירות באה על חשבון הדיוק). בעזרת השיטה המובילה כרגע (SCANN) ניתן להגיע לכמה מאות חיפושים שלמים בשנייה (החיפוש ב-100 ממדים מתוך מאגר של 10000 דוגמות).




איור 9.24 השוואת ביצועים של שיטות חיפוש שונות. עבור פרצוף נתון, מחפשים עבורו וקטור תואם בממד החדש המכיל ייצוג וקטורי של הפרצופים הידועים. בכל שיטה יש טרייד-אוף בין מהירות החיפוש לבין הדיוק.

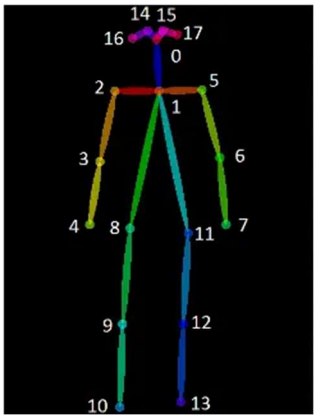
מלבד זיהוי וסיווג פנים, יש גם שיטות של מציאת אלמנטים של פנים הכוללות אף, עיניים וכו'. אחת השיטות המקובלות משתמשת בשערוך הצורה של פנים אנושיות, וניסיון למצוא את איברי הפנים לפי הצורה הסטנדרטית. בשיטה זו ראשית מבצעים יישור של הפנים והתאמה לסקאלה אנושית (על פי מרחק בין האיברים השונים בפנים), ולאחר מכן מטילים 68 נקודות עניין מרכזיות על התמונה המיושרת, מתוך ניסיון להתאים בין הצורה הידועה לבין התמונה המבוקשת.

איור 9.25 זיהוי אזורים בפנים של אדם על ידי התאמת פנים לסקאלה אנושית והשוואה למבנה של פנים המכיל 68 נקודות מרכזיות.

### 9.3.2 Pose Estimation

יישום פופולרי נוסף של אלגוריתמים השייכים לראייה ממוחשבת הינו קביעת תנוחה של אדם – האם הוא עומד או יושב, מה התנוחה שלו, באיזה זווית האיברים נמצאים וכו'. ניתן להשתמש בניתוח התנוחה עבור מגוון תחומים – ספורט, פיזיותרפיה, משחקים שונים ועוד. לרוב, תנוחה מיוצגת על ידי המיקומים של חלקי גוף עיקריים כגון ראש,

כתפיים, מרפקים וכו'. ישנם כמה סטנדרטים מקובלים, למשל COCO, posenet ועוד. ב-COCO התנוחה מיוצגת בעזרת מערך של 17 נקודות (בדו-מימד):




איור 9.26 מיפוי תנוחה ל-17 נקודות מרכזיות.

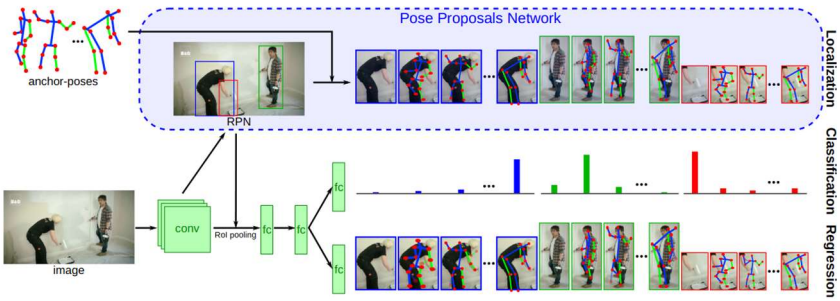
שאר הפורמטים דומים; מוסיפים עוד מידע (למשל מיקום הפה), משתמשים בתלת ממד במקום בדו ממד, משתמשים בסידור אחר וכדו'. כאשר רוצים לאסוף נתונים על מנחי גוף שונים, ניתן לשים חיישנים על אותן נקודות מרכזיות וככה לקבל מידע על מנח הגוף לאורך זמן.

רשת לצורך קביעת תנוחה יכולה לטפל במקרה כללי, בו יש מספר אנשים בתמונה, או במקרה הפרטי של גוף יחיד. המקרה השני כמובן יותר פשוט, מכיוון שישנו פלט יחיד, אותו ניתן לחזות באמצעות רגרסיה (למשל, 34 מספרים שמתארים את המיקומים של 17 הנקודות בפורמט COCO בדו-ממד). במקרה זה ניתן לעשות למידה סטנדרטית לחלוטין של בעיית רגרסיה בעלת 34 יציאות.

ישנן מספר גישות כיצד להכליל את הרשת כך שתוכל לטפל גם במקרה הכללי בו יש יותר מגוף אחד בתמונה. באופן נאיבי ניתן לבצע תהליך מקדים של מציאת כל האנשים בתמונה, ואז להפעיל על כל אחד מהם בנפרד את הרשת שמבצעת רגרסיה, כפי שתואר לעיל. שיטה נוספת פועלת בכיוון הפוך – ראשית כל הרשת מוצאת את כל האיברים בתמונה נתונה, ולאחר מכן משייכת אותם לאנשים שונים. השיטה השנייה נקראת "מלמטה למעלה" (bottom-up), כיוון שקודם כל היא מוצאת את הפרטים ולאחר מכן מכלילה אותם. גישה זו יעילה למקרה בו יש הרבה חפיפה בין האנשים בתמונה, כיוון שאין לה צורך לבצע תהליך מקדים של הפרדת האנשים. השיטה הראשונה, הנקראת "מלמעלה למטה" (top-down), תהיה פשוטה יותר עבור מקרים בהם אין חפיפה בין האנשים בתמונה וכל אחד מהם נמצא באזור שונה בתמונה, כיוון שבמקרה זה אין צורך לשייך איברים לאנשים.

רשת פופולרית לקביעת תנוחה נקראת Multi-person pose estimation, המורכבת למעשה משתי תת-רשתות ופועלת בשיטת bottom-up. הרשת שאחראית על שיוך חלקי גוף לאדם מסוים, נקראת part affinity fields (PAF), והרעיון שלה הוא לייצג כל איבר כשדה וקטורי. בייצוג זה הווקטורים השונים מצביעים לכיוון איבר הגוף ה'בא בתור' (למשל זרוע מצביעה ליד), וככה ניתן לשייך איברים שונים אחד לשני, ואת כל יחד לגוף מסוים.

רשת פופולרית אחרת, הפועלת בגישת top-down, נקראת LCR-NET, והיא מבוססת על רעיון של 'מיקום-סיווג-רגרסיה' (Localization-Classification-Regression). בשלב הראשון יש תת-רשת המייצרת עוגנים עבור אנשים, כלומר, אזורים בהם הרשת חושבת שנמצא בן אדם, ולאחר מכן הרשת משערכת את התנוחות שלהם. בשלב השני מתבצע clustering לכל העוגנים, כלומר כל עוגן מקבל ציון המייצג את טיב השערוך של העוגן והתנוחה של האדם הנמצא בתוכו. השלב השלישי מלטש את העוגנים ומשקלל את השערוך הסופי בעזרת מיצוע של הרבה עוגנים. שלושת השלבים משתמשים ברשת קונבולוציה משותפת, כמתואר באיור.



איור 9.27 Localization-Classification-Regression.

## 9.4 Few-Shot Learning

יכולת הצלחתם של אלגוריתמי למידה עמוקה נשענת על כמות ואיכות הדאטה לאימון. עבור משימת סיווג תמונות (Image Classification), נדרש שעבור כל קטגוריית סיווג תהיה כמות גדולה של תמונות מגוונות (עם הבדלי רקעים, בהירות, זוויות וכו'), ובנוסף יש צורך בכמות דומה של דוגמאות בכל קטגוריות הסיווג. חוסר איזון בין כמות התמונות בקטגוריות השונות משפיע על יכולת הלמידה של האלגוריתם את הקטגוריות השונות ועל כן עלול ליצור הטיה בתוצאות הסיווג לטובת הקטגוריות להן יש יותר דוגמאות. בפרק זה נעסוק בשיטות כיצד ניתן להתמודד עם מצבים בהם הדאטה אינו מאוזן.

### 9.4.1 The Problem

התחום של למידה ממיעוט דוגמאות (Few-Shot Learning) נוצר על מנת להתמודד עם מצב של חוסר איזון קיצוני בין כמות הדוגמאות של כל קטגוריה לאימון הרשת. באופן פורמלי, קיימות קטגוריות הנקראות קטגוריות בסיס (base classes), עבורן יש כמות גדולה של דוגמאות, ובנוסף ישנן קטגוריות חדשניות (novel classes), עבורן יש כמות קטנה מאוד של דוגמאות. בכדי להגדיר את היחס, משתמשים בשני פרמטרים: פרמטר k המייצג את מספר ה-shots, כלומר מספר הדוגמאות הקיימות בסט האימון מכל קטגוריה חדשנית, ופרמטר n שמייצג את מספר הקטגוריות החדשניות הקיימות סך הכל. כל בעיה מוגדרת על ידי "k-shot n-way", ולמשל "one-shot 5-way learning" מתאר מצב בו יש חמש קטגוריות חדשניות, ומכל אחת מהן יש רק דוגמא אחת לאימון הרשת. ככלל, בקטגוריות הבסיס תהיה כמות גדולה של דוגמאות. למשל בסט התמונות האופייני לבעיות אלו, mini-ImageNet, יש 600 דוגמאות לכל קטגוריית בסיס ולרוב 1-5 דוגמאות עבור הקטגוריות החדשניות.

האתגר בלמידה ממיעוט דוגמאות נובע מהצורך להכניס לרשת כמות ידע קטנה נוספת על הידע הנרחב הקיים, תוך הימנעות מ-overfitting כתוצאה מכמות הפרמטרים הגדולה של הרשת לעומת הכמות המועטה של הדאטה. לכן, גישה נאיבית כמו אימון מחדש של רשת על מעט דוגמאות נוספות עלולה ליצור הטיה בתוצאות.

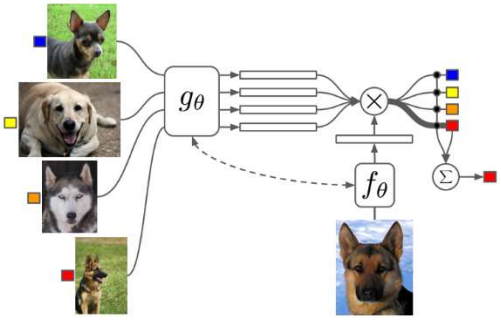
יש לציין כי בכל בעיות הלמידה ממיעוט דוגמאות, אמנם חסר דאטה עבור האימון, אך השאיפה היא להצליח באופן זהה בזיהוי כל הקטגוריות בשלב המבחן, בו לא יהיה חוסר איזון. לכן בעיות אלו רלוונטיות לשימושים רבים כמו: זיהוי חיות נדירות באופן זהה לחיות יותר נפוצות, מערכת זיהוי טילים שצריכה להתמודד גם עם איומים נדירים יותר (ניתן לחשוב למשל על פצצת אטום), מערכות זיהוי פנים שצריכות לעבוד טוב עבור כל אדם ללא תלות בדאטה שהיה קיים באימון הרשת.

בפרק זה נתאר את שלוש הגישות העיקריות לפתרון בעיות למידה ממיעוט דוגמאות. עבור כל גישה נציג את האלגוריתמים המשמעותיים ביותר שנקטו בגישה זו. לאחרונה, מפותחים יותר ויותר אלגוריתמי למידה ממיעוט דוגמאות שמשלבים יחד רעיונות השאובים ממספר גישות יחד אך נשענים על האלגוריתמיים המשמעותיים מהעבר. לבסוף, נציג את התחום של Zero-Shot Learning, כלומר יכולת למידה של קטגוריה חדשה כאשר לא קיימת אף דוגמא שלה לאימון.

### 9.4.2 Metric Learning

שיטות להתמודדות עם למידה ממיעוט דוגמאות הנוקטות בגישת למידת מטריקה, שואפות לייצג את הדוגמאות כווקטורים של מאפיינים במרחב רב-ממדי, כך שניתן יהיה למצוא בקלות את השיוך הקטגורי של דוגמא חדשה, גם אם היא תהיה מקטגוריה חדשנית. שיטות אלו מבוססות על עיקרון הגדלת המרחק בין ייצוגים וקטורים של דוגמאות מקטגוריות שונות (inter-class dissimilarity), בד בבד שמירה על מרחק קטן בין הייצוג הווקטורי של דוגמאות מאותה הקטגוריה (intra-class similarity).

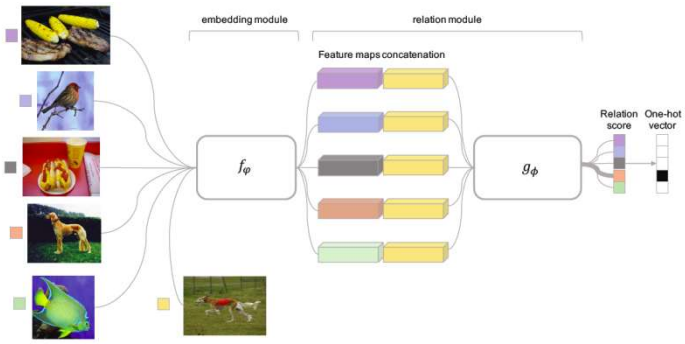
התקדמות משמעותית של שיטות אלו הוצגה במאמר Matching Networks for One Shot Learning בשנת 2016. שיטה זו משתמשת בזיכרון שהגישה אליו נעשית באמצעות מנגנון Attention, על מנת לחשב את ההסתברות של דוגמא להיות שייכת לכל קטגוריה, בדומה לשיטות השכן הקרוב (Nearest Neighbors).




איור 9.28 אילוסטרציה של שיטת Matching Networks.

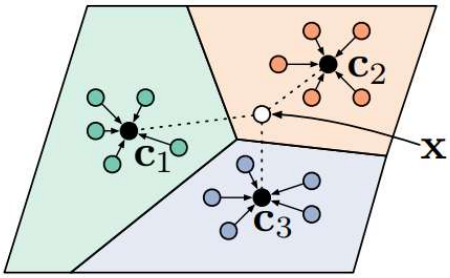
החידוש המשמעותי בשיטת Matching Networks נעוץ בשיטת האימון המבוצעת באפיזודות (Episodes). בשיטה זו האימון מכיל כמות של משימות, כאשר כל משימה היא למעשה מדגם של הדאטה שבו יש קטגוריות מסוימות שהן חדשניות ואחרות שהן קטגוריות בסיס. על ידי דגימות רבות ויצירת משימות מלאכותיות כאלו, בהן בכל פעם נלקחות קטגוריות אחרות לייצג את החדשניות, מתבצע אימון המתאים לבעיה של מיעוט דוגמאות. שיטת אימון באפיזודות הפכה לנפוצה ביותר בלמידה ממיעוט דוגמאות, גם בגישת המטריקה וגם בגישות שנראה בהמשך.

שיטות רבות מתבססות על הרעיונות של מאמר זה. למשל שיטת Relation Network משרשרת וקטורי מאפיינים של דוגמת מבחן לבין כל דוגמא של קטגוריות האימון. אלו נכנסים למודל המשערך מדד דמיון בעזרתו ניתן לסווג את דוגמת המבחן.




איור 9.29 Learning to Compare: Relation Network for Few-Shot Learning.

שיטה משמעותית נוספת הנוקטת בגישת למידת מטריקה נקראת Prototypical Networks. בגישה זו כל קבוצת דוגמאות של קטגוריה מסוימת במרחב וקטורי המאפיינים מקבלת נקודת אב-טיפוס אופיינית המחושבת על ידי הממוצע של הדוגמאות בקטגוריה זו. בכך מחשבים מסווג לינארי המפריד בין הקטגוריות. בעת המבחן נסווג דוגמא חדשה על סמך מרחק אוקלידי מנקודות האב-טיפוס.




איור 9.30 Prototypical Networks for Few-Shot Learning.

בטבלה הבאה ניתן לראות השוואת ביצועים של שיטות למידת המטריקה שהוזכרו על הקטגוריות החדשניות. יש להדגיש כי כל השיטות מגיעות לאחוזי דיוק נמוכים משמעותית מאחוזי הדיוק המדווחים במקרים של איזון בין כמות הדוגמאות בקטגוריות השונות (לרוב מעל 90% דיוק).

| Method | 5-way 1-Shot | 5-way 5-Shot |
|---|---|---|
| Matching Networks | 46.6% | 60.0% |
| Prototypical Networks | 49.42% | **68.20%** |
| Learning To Compare | **50.44%** | 65.32% |

איור 9.31 השוואת ביצעי דיוק של שיטות למידת מטריקה על קטגוריות חדשניות עבור mini-ImageNet.

### 9.4.3 Meta-Learning (Learning-to-Learn)

גישה שניה להתמודדות עם מיעוט דוגמאות וחוסר איזון בין הקטגוריות נקראת מטא-למידה (או: ללמוד איך ללמוד). באופן כללי בלמידת מכונה, כאשר מדובר על מטא-למידה, מתכוונים לרשת שלומדת על סמך התוצאות של רשת אחרת. בלמידה ממיעוט דוגמאות הרעיון הוא שהרשת תלמד בעצמה איך להתמודד עם מיעוט הדאטה על ידי עדכון הפרמטרים שלה לאופטימיזציה של בעיה של סיווג ממיעוט דוגמאות. לשם כך משתמשים באפיזודות של משימות למידה ממיעוט דוגמאות.

שיטה חשובה בגישה זו היא Model-Agnostic Meta-Learning (MAML). בשיטה זו, שאינה מיועדת ספציפית לסיווג תמונות ממיעוט דוגמאות, בעזרת מספר צעדים מעטים בכיוון הגרדיאנט ניתן ללמד את הרשת התאמה מהירה (fast adaptation) למשימה חדשה. כזכור, כל משימה באימון היא אפיזודה שבה קטגוריות מסוימות נבחרות רנדומלית לדמות את הקטגוריות החדשניות. בכל משימה כזו נלמדים פרמטרים של המודל האגנוסטי כך שעדכונם בכיוון הגרדיאנט יוביל להתאמה למשימה החדשה. הכותבים מציינים שמנקודת מבט של מערכות דינאמיות, ניתן להתבונן על תהליך הלמידה שלהם ככזה שממקסם את רגישות פונקציית המחיר של משימות חדשות ביחס לפרמטרים. כאשר הרגישות גבוהה, שינויי פרמטרים קטנים יכולים להוביל לשיפור משמעותי במחיר של המשימה. מתמטית, פרמטרי המודל, המיוצגים על ידי $\theta$, משתנים עבור כל משימה $T_i$ להיות $\theta_i'$, כאשר:

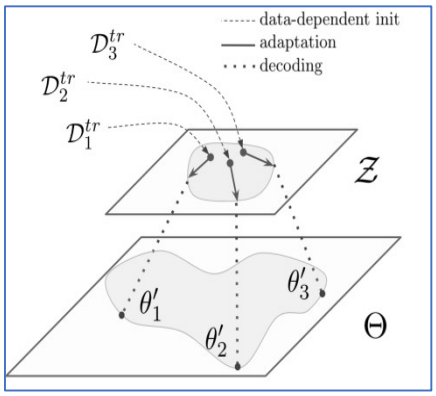
$$\theta_i' = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta)$$

עבור פונקציית מחיר $L$ והיפרפרמטר $\alpha$. כאשר מבצעים מטא-למידה לעדכון הפרמטרים, מחשבים למעשה SGD:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \sim p(T)} \left( \boldsymbol{L}_{T_i}(f_\theta) \right)$$

כאשר המשימות נדגמות מתוך $p(T)$ ו-$\beta$ הוא גודל הצעד של המטא-למידה.

שיטה מעניינת נוספת בשם Latent Embedding Classification (LEO) מיישמת את הגישה של מטא-למידה במרחב ייצוג לטנטי בעל ממדים נמוכים, ולאחר מכן עובר בחזרה למרחב המאפיינים הרב ממדי.




איור 9.32 שיטת Latent Embedding Classification (LEO).

השימוש במרחב מאפיינים בעל ממדים נמוכים המשמרים את המאפיינים החשובים לייצוג הקטגוריות, שיפר באופן ניכר את תוצאות הסיווג, כפי שניתן לראות בטבלה הבאה:

| Method | 5-way 1-Shot | 5-way 5-Shot |
|---|---|---|
| MAML | 48.7% | 63.11% |
| LEO | **61.76%** | **77.59%** |

איור 9.33 השוואת ביצעי דיוק של שיטות מטא-למידה על קטגוריות חדשניות עבור mini-ImageNet.

### 9.4.4 Data Augmentation

גישה שונה להתמודדות עם מיעוט דוגמאות נוקטת ביצירת דוגמאות כדי להימנע מהטיה. שיטות אוגמנטציה למעשה יוצרות דאטה חדש על סמך הדאטה הקיים. השיטות הפשוטות יותר מייצרות מהתמונות הקיימות תמונות ראי, שינויי תאורה וקונטרסט, שינויי סקאלה, שינוי זוויות, ואף הוספת רעש רנדומלי. כל אלו הראו שיפורים ביכולות הרשתות ללמוד קטגוריות שהיו במצב של חוסר איזון. דרך נוספת היא שימוש ברשתות גנרטיביות (GANs) על מנת לייצר דוגמאות רלוונטיות, למשל דוגמאות של אותו האובייקט מזוויות שונות. שיטה מעניינת של אוגמנטציות היא CutMix, בה פאצ'ים של תמונות נחתכים ומודבקים בתמונות האימון וגם התיוגים מעורבבים בהתאם. שיטה זו הגיעה לביצועים מרשימים בסיווג תמונות וגם בזיהוי אובייקטים, ככל הנראה בגלל שהיא מאפשרת למודל להיות גנרי יותר בהתייחסות לחלקים שונים מהתמונה המשפיעים על הסיווג לקטגוריה.

## 9. References

Detection:

https://arxiv.org/pdf/1406.4729.pdf

Segmentation:

https://arxiv.org/ftp/arxiv/papers/2007/2007.00047.pdf

SegNet:

https://arxiv.org/pdf/1511.00561.pdf

https://mi.eng.cam.ac.uk/projects/segnet/#demo

https://arxiv.org/pdf/1409.1556.pdf

https://arxiv.org/pdf/1502.01852.pdf

Face recognition:

https://docs.opencv.org/master/d2/d42/tutorial_face_landmark_detection_in_an_image.html

http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html