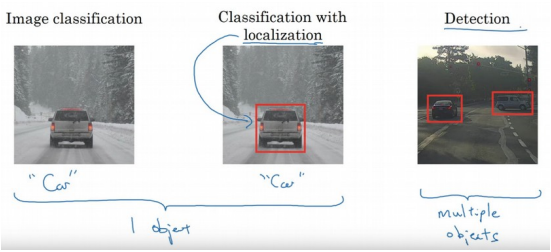
Object Localization

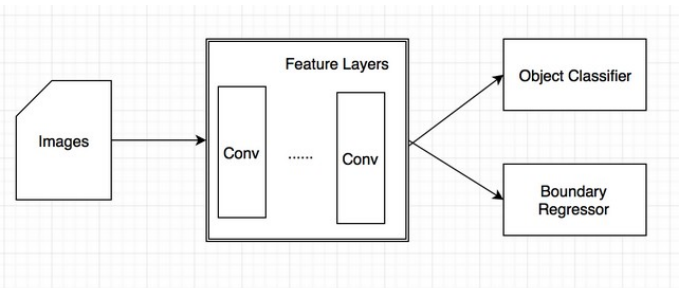
מבוסס על הקורס של Deeplearning.ai

- Image Classification: חיזוי סוג של אובייקט בתמונה.
  - קלט: תמונה עם אובייקט בודד.
  - פלט: איזה אובייקט מופיע בתמונה.
- Object Localization: איתור את נוכחותם של אובייקטים, זיהוי מיקומם.
  - קלט: תמונה עם אובייקט אחד.
  - פלט: המיקום של האובייקט בתמונה.
- Object Detection: מציאת מיקום האובייקטים ,וסיווגם.
  - קלט: תמונה עם אובייקט אחד או יותר.
  - פלט: איכן האובייקטים מופיעים בתמונה, וסיווגם.



Object classification + Localization

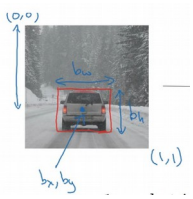
עד עכשיו פתרנו בעיות של סיווג, בהינתן תמונה סיווג האובייקט שבתמונה. לוקליזציה של עצמים מתייחסת לזיהוי המיקום של אובייקט בתמונה.



נניח שיש לנו סט של תמונות אימונים עם תיבת גבול המסומנת ידנית. עם תמונות אלה כמערכת האימונים שלך, נגדיר את וקטור היעד שלך Y או וקטור התווית עבור הרשת כ:

$$Y = \begin{bmatrix} p_1 \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- p1: אינדיקציה עבור קיום אובייקט בתמונה. 1–קיים אובייקט. 0– לא קיים אובייקט.
- B_x: הוא קואורדינטת ה– x של מרכז תיבת הגבול.
- B_y: הוא קואורדינטת ה– y של מרכז תיבת הגבול.
- B_h: גובה התיבה.
- b_w:רוחב התיבה.
- C1, c2, c3: תוויות הסוג של האובייקט שנמצא בחלון.
  - כלומר, אם קבענו כי c1= dog, c2= cat, c3= car. ובתמונה יש מכונית אז c3= 1 ו c1, c2= 0.



Object Detection

מבוסס על הקורס של Deeplearning.ai

Sliding window

גישה פשוטה לאיתור אובייקטים. נניח ויש לנו תמונה עם אופניים בתוכה:

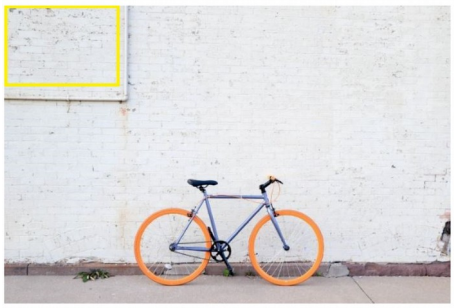
Photo by Tiffany Nutt on Unsplash

אנחנו רוצים לסווג את האובייקט שבתמונה (אופניים), ולמצוא את מיקומו. נעשה זאת באמצעות sliding windows detection.

לצורך sliding windows detection, ראשית, נבנה את מערך האימונים שלנו עם תמונות קצוצות של האובייקט.

https://medium.com/x8-the-ai-community/object-classification-detection-and-localization-72d51b0fd6bc

לכל אחת מהתמונות הללו תווית שתאמר שאם יש אופניים או לא. אנו נשתמש בתמונות הקצוצות הללו בכדי לאמן רשת להזהות אופניים.כעת בתמונה המקורית אנו נבחר בגודל חלון (nxn) ונקח כל פעם חלק אחר מהתמונה, בדומה לאופן בו אנו מבצעים קונבולוציה באמצעות קרנל ב-CNN. בכל מיקום בחלון נשתמש ב- CNN המאומן שלנו על התמונות החתוכות כדי לזהות האם בחלק זה של התמונה קיים אופניים.



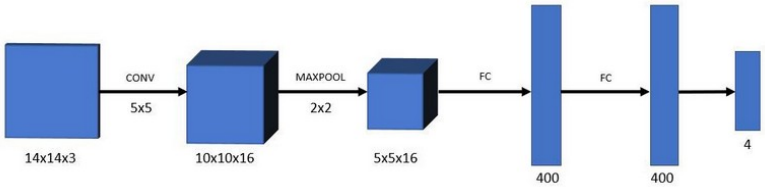
We start with a randomly sized window (top left) and slide it across the image and send each image inside the window to our CNN trained on closely cropped images.

https://medium.com/x8-the-ai-community/object-classification-detection-and-localization-72d51b0fd6bc

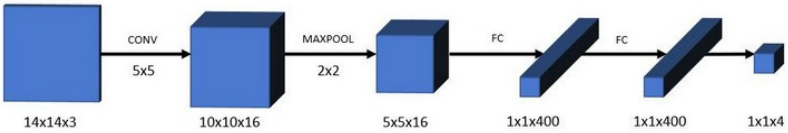
יתכן כי גודל החלון קטן מידי עבור האופניים, ולכן CNN לא מצליחה לסווג את האופניים בתמונה כולה. במקרה כזה נגדיל את החלון ונחזור על התהליך. כפי שניתן לראות לשיטה זו יש חסרונות רבים. ישנם חישובים רבים מה שהופך את השיטה הזאת ללא יעילה.

כדי לטפל בבעיית היעילות אנחנו יכולים ליישם את איתור חלון ההזזה באמצעות CNN.

Convolutional Implementation of sliding windows



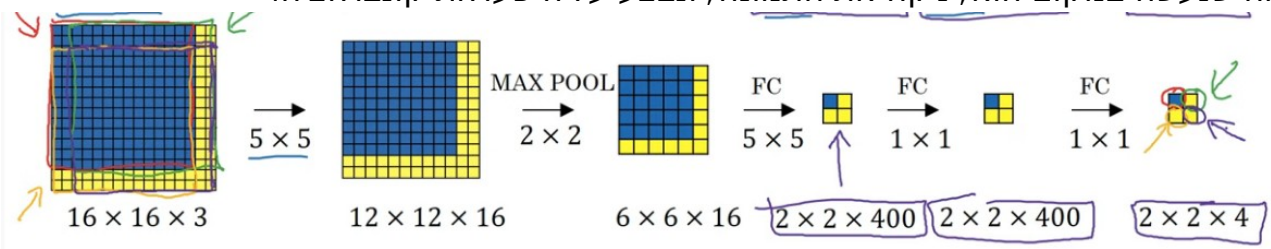
בצד שמאל יש לנו תמונה 14X14X3, אשר עוברת קונבולוציה עם 16 פילטרים 5X5X3 כדי לקבל מערך של 10X10X16 מערך זה מועל ל maxpooling שהפלט שלו הוא 5X5X16. אם נשטח מערך זה נקבל 400 ערכים המחוברים לשכבת ה FC. נשים לב שבמקום שכבת ה FC ניתן לממש שכבה זו באמצעות 400 פילטרים של 1X1 convolution.



כך נוכל להמיר את שכבות ה FC בשכבות קונבולוציה. כעת נראה כיצד ניתן לממש כך את ה sliding windows.

נניח ויש לנו תמונת קלט בגודל 16X16X3 ואנחנו בוחרים חלון בגודל 14X14. בשיטה הישנה, הקלט ל CNN היה תמונה מקוצצת בגודל של 14X14 במידה והקפיצות בתזוזת החלון היו של 2 אז הינו שולחים לרשת 4 תמונות של 14X14, כפי שכבר ראינו זה יקר מאוד חישובית.

מה שנעשה במקום הוא, ניקח את התמונה, ונבצע עליה פעולות קונבולוציה:



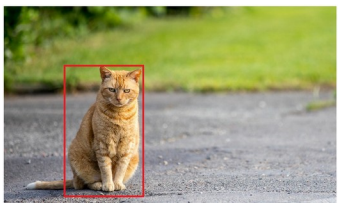
מסתבר שהערך השמאלי העליון בשכבה הסופית מעניק לנו את הערך השווה לפלט ה CNN בשיטה הקודמת עבור המלבן בצד שמאל למעלה בגודל 14x14 פיקסלים של תמונת הקלט, הערך הימני העליון בשכבה הסופית הוא הערך המתאים לימין למעלה 14x14 פיקסלים של תמונת הקלט וכן הלאה. אז על ידי שינוי ממדי הסינון, נוכל לבנות CNN שעושה פעולת חלון הזזה על תמונת הקלט עם גדלי חלונות וצעדים כרצוי. שיטה זו הוצאה על ידי Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun במאמר שכתבו.

Intersection Over Union (IOU)

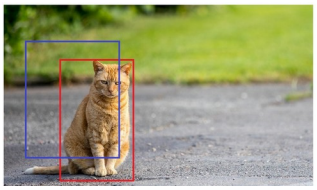
(fron:**https://medium.com/koderunners/intersection-over-union-516a3950269c**)

נניח ואנחנו מכניסים ל detector שלנו את התמונה הבאה:

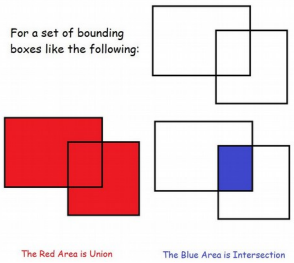
במקרה האידיאלי הפלט של האלגוריתם לזיהוי אובייקטים שלנו צריך להראות כך:



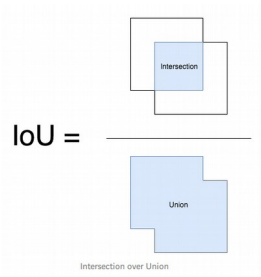
אבל אם האלגוריתם שלנו פולט את התיבה הכחולה. האם זה נחשב חיזוי טוב?



במקרה כזה, נחשב IOU עבור התיבות.



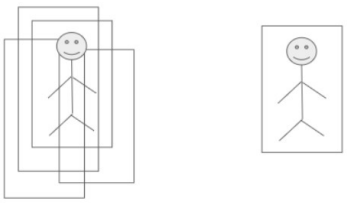
IOU(Box1, Box2) = Intersection_Size(Box1, Box2) / Union_Size(Box1, Box2)



ככל שה IOU גבוה יותר החפיפה גדולה יותר ולכן החיזוי טוב יותר.

Non-maximum Suppression (NMS)

NMS היא טכניקה המשמשת אלגוריתמים רבים בראייה ממוחשבת. זוהי מחלקה של אלגוריתמים שמטרתה לבחור ישות אחת (למשל bbox במקרה שלנו) מתוך ישויות רבות חופפות. בדרך כלל הקריטריונים לסיווג הם סוג כלשהו של מספר הסתברות יחד עם איזושהי מידה של חפיפה (למשל IOU).



Before NMS and after NMS

https://medium.com/@whatdhack/reflections-on-non-maximum-suppression-nms-d2fce148ef0a

NMS:

Input: A list of Proposal boxes B, corresponding confidence scores S and overlap threshold N.

Output: A list of filtered proposals D.

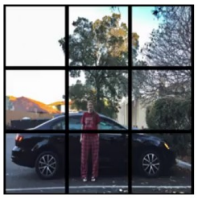
Algorithm:

1. Select the proposal with highest confidence score, remove it from B and add it to the final proposal list D. (Initially D is empty).
2. Now compare this proposal with all the proposals — calculate the IOU (Intersection over Union) of this proposal with every other proposal. If the IOU is greater than the threshold N, remove that proposal from B.
3. Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.
4. Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have high IOU than threshold.
5. This process is repeated until there are no more proposals left in B.

https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

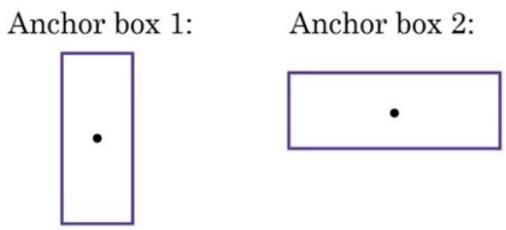
Anchor Boxes

לעתים קרובות במציאות יש מספר עצמים, שהמרכז שלהם נופל באותו מקום. וזה מוביל אותנו למושג anchor boxes. נתבונן בתמונה הבאה המחולקת לתאים 3X3:



איך אנחנו מקצים אובייקט לתא? לוקחים את נקודת האמצע של האובייקט ובהתבסס על מיקום האמצע, נקצה את האובייקט לתא. בדוגמה הנ"ל נקודת האמצע של שני העצמים נמצאת באותו תא. כך יראו ה bbox עבור האובייקטים:

אם נשתמש רק בשיטות שלמדנו עד עכשיו אזי נקבל רק אחת משתי התיבות, או לרכב או לאדם. אבל אם נשתמש ב anchor boxes יתכן כי נצליח לחזות את שני האובייקטים. איך נוכל לעשות זאת? ראשית, אנו מגדירים מראש שתי צורות שונות הנקראות anchor box. כעת, עבור כל תא בתמונה, במקום שיהיה לנו פלט אחד, יהיו לנו שני פלטים. אנו תמיד יכולים להגדיל את מספר anchor boxes גם כן (2 anchor boxes רק לשם הדוגמה):



כך נראה תווית y ללא anchor boxes:

$$Y = \begin{bmatrix} p_1 \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

וכך Y נראה עם 2 anchor boxes:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \text{Anchor box 1} \\ \text{Pedestrian} \\ \\ \\ \\ \\ \\ \\ \text{Anchor box 2} \\ \text{Car} \\ \\ \\ \\ \\ \\ \\ \end{matrix}$$

8 השורות הראשונות שייכות ל anchor box 1 והשמונה הנותרות שייכות ל anchor box 2.
הפלט של הרשת במקום 3X3X8 כפי שהיה לפי להוספת ה anchor box ל 3X3X16.

כעת מה שנותר הוא לשלב את כל הרעיונות הללו כדי לתאר את אלגוריתם YOLO.

YOLO (YOU Only Look Once)

מערכת זיהוי אובייקטים בזמן אמת. מאמר: https://arxiv.org/pdf/1506.02640.pdf.

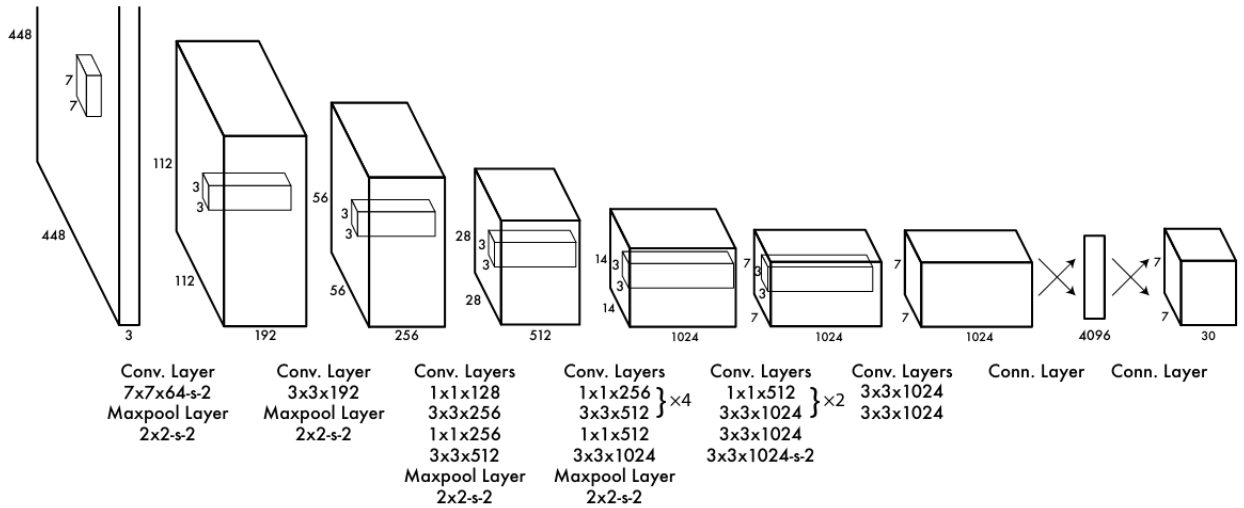
YOLO מחלק את תמונת הקלט ל grid בגודל S × S. כאשר כל תא מנבא B boundary box יחד עם ה confidence score שלהם, כל תא יכול לחזות אובייקט אחד בלבד ללא קשר למספר התיבות B, ובנוסף כל תא מנבא הסתברות C כמספר האובייקט שהוא מנסה לחזות.
במאמר כדי לנבא את PASCAL VOC הם משתמשים בגריד בגודל של 7X7, כשאר כל תא מנבא 2 boundary box (B=2) ו 20 מחלקות שונות (C=20).
כל boundary box B מכיל 5 אלמנטים:

- confidence score: מה הסיכוי שמופיע בתיבה אובייקט.
- B_x: הוא קואורדינטת ה- x של מרכז תיבת הגבול. (מנורמל לתא המתאים)
- B_y: הוא קואורדינטת ה- y של מרכז תיבת הגבול. (מנורמל לתא המתאים)
- B_h: גובה התיבה.
- b_w:רוחב התיבה.
  - מכאן ש- B_x, B_y, b_w ו- B_h נמצאים בין 0 ל 1.

ולכל תא יש 20 הסתברויות. אזי לחיזוי של YOLO יש צורה של:

$$S \times S \times (B * 5 + C) = (7,\ 7,\ 2 \times 5 + 20) = (7,7,30)$$

ולכן הארכיטקטורה נראית כך:



ל YOLO יש 24 שכבות קונבולוציה ולאחריהן 2 שכבות FC.

אימון הרשת:

- תחילה, אנו מאמנים את הרשת לבצע סיווג עם במונות ImageNet ברזולוציה 224x224 , תוך שימוש רק ב –20 השכבות הראשונות ואחריהם average pooling ושכבת FC.
- שנית, אנו מאמנים את הרשת ל detection על ידי הוספת ארבע שכבות קונבולוציה ושתי שכבות FC. לצורך detection מאמנים את הרשת עם תמונות ברזולוציה של 448x448.

Loss function:

YOLO מנבא מספר bbox עבור כל תא. נבחר את האחד עם ה IOU הכי גבוה.
YOLO מפעיל Non–maximal suppression על מנת להימנע מאיתור כפול של אובייקטים.
פונקציית ה LOSS מורכבת מ 3 חלקים:

- classification loss.
  - אם מתגלה אובייקט, פונקציית הלוס היא sum squared error:

$$+\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)$$

    - $1_i^{obj}=1$ אם קיים אובייקט בתא i, אחרת 0.

- localization loss.
  - מחשב את הטעות בין ה bbox שהמודל חזה לבין ה bbox הצפוי שלנו.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

    - $1_{ij}^{obj}=1$ אם ה boundary box בתא ה i אחראית לגילוי האובייקט, אחרת 0.
    - $\lambda_{coord} = 5$ בברירת מחדל

- על מנת לטפל בזה שאנו לא רוצים שהשגיאות בתיבות גדולות ובתיבות קטנות יהיו באותה מידה, YOLO מנבא את השורש הריבועי של הרוחב והגובה. בנוסף, כדי לשים דגש רב יותר על דיוק תיבת הגבול, אנו מכפילים את ההפסד ב- λ_{coord}.

- confidence loss. האם קיים אובייקט בתיבה:

$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left(C_i-\hat{C}_i\right)^2$$

  - $1_{ij}^{obj}=1$ אם ה boundary box בתא ה i אחראית לגילוי האובייקט, אחרת 0.

- אם לא קיים אובייקט בתיבה:

$$\lambda_{\text{noobj}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{noobj}}\left(C_i-\hat{C}_i\right)^2$$

  - 1_{ij}^{noobj} הוא המשלים של 1_{ij}^{obj}
  - λ_{noobj} באופן דיפולטיבי מוגדר להיות 0.5.
  - מרבית התיבות אינן מכילות אובייקטים. אנו מאמנים את המודל לאתר רקע בתדירות גבוהה יותר מאשר גילוי אובייקטים. כדי לתקן זאת, נוסיף את λ_{noobj}.

לסיכום פונקציית ההפסד היא:

$$\begin{aligned}
&\lambda_{\text{coord}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]\\
&+\lambda_{\text{coord}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]\\
&+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left(C_i-\hat{C}_i\right)^2\\
&+\lambda_{\text{noobj}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{noobj}}\left(C_i-\hat{C}_i\right)^2\\
&+\sum_{i=0}^{S^2}\mathbb{1}_{i}^{\text{obj}}\sum_{c\in\text{classes}}(p_i(c)-\hat{p}_i(c))^2 \qquad (3)
\end{aligned}$$

מגבלות של YOLO

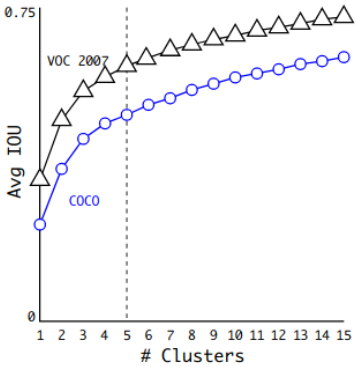
1. מאחר שכל תא מנבא שתי תיבות בלבד, ויכול לזהות אובייקט אחד בלבד, ישנה הגבלה במספר העצמים הסמוכים אותם הוא יכול לחזות.
2. YOLO יכול לאתר רק 49 עצמים.
3. שגיאת לוקליזציה גבוהה יחסית.

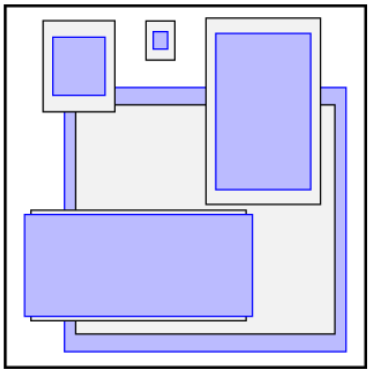
YOLOv2

מאמר: https://arxiv.org/pdf/1612.08242.pdf

YOLO מבצע שגיאות לוקליזציה גבוהות יחסית. בנוסף, ל YOLO היה recall נמוך יחסית. ולכן בגרסה השניה הם התמקדו בשיפור הלוקליזציה וה recall תוך שמירה על דיוק בסיווג.
במאמר Joseph Redmon, Ali Farhadi מציעים מספר פתרונות:

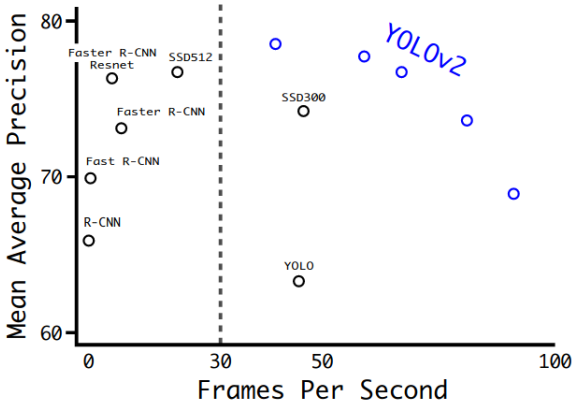
- BatchNormalization: הוביל לשיפורים משמעותיים בהתכנסות הרשת תוך ביטול הצורך בסוגים אחרים של נרמולים.
- High Resolution Classifier: ב YOLO הם אימנו את המסווג על תמונות בגודל 224X224, ותמונות בגודל 448X448 עבור ה detection. מה שגרם לכך שבמעבר ל detection הרשת הייתה צריכה להסתגל גם למעבר עצמו וגם לשינוי הגדול ברזולוציות. ב YOLOv2 המסווג אומן על תמונות מ imagenet ברזולוציה 448X448 במשך 10 epochs (תחילה אומן על תמונות 224X224).
- Convolutional With Anchor Boxes: המחברים ניסו לפתור את הבעיה שב YOLO כל תא יכול לאתר אובייקט אחד בלבד. YOLOv2 הם מציעים את השימוש ב Anchor boxes, וכך לאפשר לתא לאתר יותר מאובייקט אחד עד ידי שימוש ב k bounding box.
- Dimension Clusters: במקום לבחור את ה Anchor boxes ביד, הכותבים הפעילו k-means clustering על ה training set bounding boxes. הם בחרו ב- k = 5 שמאזן בצורה טובה בין מורכבות המודל לבין ה high recall.





- Multi-Scale Training: על מנת לאפשר ל YOLOv2 להתמודד גם עם תמונות ברזולוציות שונות, המודל אומן עם תמונות בגדלים שונים. בכל 10 batches הם שינו את מימדי התמונה לאחד המימדים שנקבעו מראש: $\{320, 352, 354, \ldots, 608\}$.

השוואה למודלים אחרים:



Network Architecture:

המחברים הציעו מודל סיווג חדש הנקרא Darknet-19 שישמש את YOLOv2.
ל Darknet-19 יש 19 שכבות קונבולוציה ו 5 maxpooling layers.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Training:

המודל אומן תחילה לסיווג ולאחר מכן ל detection.

- Classification: הם אימנו את הרשת על ImageNet 1000 class classification dataset עם קלט של תמונות בגודל 224X224 במשך 160 epoch. לאחר מכן אימנו את הרשת עם תמונות ברזולוציה של 448X448 למשך 10 epoch.
- Detection: לאחר אימון המסווג, הם הסירו את שכבת הקונבולוציה האחרונה ובמקומם הוסיפו שלוש שכבות קונבולוציה ו 1x1 convolutional layer על מנת להגיע לפלט המתאים (13X13X125)

YOLOv3: An Incremental Improvement

מאמר: https://pjreddie.com/media/files/papers/YOLOv3.pdf

It's a little bigger than last time but more accurate

Bounding Box Prediction:

זהה ל YOLOv2, גם YOLOv3 חוזה 4 קואורדינטות עבור כל תיבה. גם חוזה objectness score לכל תיבת הגבלה באמצעות רגרסיה לוגיסטית. כלומר האם בתיבה יש אובייקט או לא.

Multi labels prediction:

שימוש ב Softmax לשם סיווג המחלקה של האובייקט יוצא מנקודת הנחה שבכל תיבה יש אובייקט ממחלקה אחת בלבד אך לעיתים זהו לא המקרה. מסיבה זו, YOLOv3 לא משתמש ב softmax במקום זאת מנבאים כל מחלקה באמצעות רגרסיה לוגיסטית.

Small objects detection:

YOLOv3 משפר את החיזויים שלו עבור עצמים קטנים, שיפור זה נובע בעקבות שימוש ב -skip connections. באמצעות שיטה זו אנו מקבלים מידע לא רק מהשכבה הקודמת אלא גם מהשכבות הקודמות לה.

Feature Extractor Network:

YOLOv3 משתמש ברשת חדשה לביצוע feature extraction. לרשם החדשה יש 53 שכבות קונבולוציה ועל כן שמה Darknet-53.

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

ניתן לראות כי בהשוואה ל YOLOv2 הוא פחות מהיר, אך לעומת ResNet הוא הרבה יותר מהיר.
מבחינת התוצאות רואים שיפור ו YOLOv3 יותר מדיוק מקודמו.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

לאחר אימון המסווג מוחקים את השכבה האחרונה מהמודל.

Predictions Across Scales:

בניגוד לקודמיו שנבאים את הפלט בשכבה האחרונה בלבד, YOLOv3 מנבא boxes at 3 different scales.

Performance: