DOI 10.5281/zenodo.2554189

## Citation

This work was accepted to be presented at IWSSIP 2020. If you use this code for your research, please consider citing:

```
@article{padillaCITE2020,
title = {Survey on Performance Metrics for Object-Detection Algorithms},
author = {Rafael Padilla, Sergio Lima Netto and Eduardo A. B. da Silva},
booktitle  = {International Conference on Systems, Signals and Image
Processing (IWSSIP)},
year = {2020}
}
```

# Metrics for object detection

The motivation of this project is the lack of consensus used by different works and implementations concerning the **evaluation metrics of the object detection problem**. Although on-line competitions use their own metrics to evaluate the task of object detection, just some of them offer reference code snippets to calculate the accuracy of the detected objects.
Researchers who want to evaluate their work using different datasets than those offered by the competitions, need to implement their own version of the metrics. Sometimes a wrong or different implementation can create different and biased results. Ideally, in order to have trustworthy benchmarking among different approaches, it is necessary to have a flexible implementation that can be used by everyone regardless the dataset used.

**This project provides easy-to-use functions implementing the same metrics used by the the most popular competitions of object detection**. Our implementation does not require modifications of your detection model to complicated input formats, avoiding conversions to XML or JSON files. We simplified the input data (ground truth bounding boxes and detected bounding boxes) and gathered in a single project the main metrics used by the academia and challenges. Our implementation was carefully compared against the official implementations and our results are exactly the same.

In the topics below you can find an overview of the most popular metrics used in different competitions and works, as well as samples showing how to use our code.

## Table of contents

- Metrics for object detection
    - Table of contents
    - Different competitions, different metrics
    - Important definitions
        - Intersection Over Union (IOU)
        - True Positive, False Positive, False Negative and True Negative
        - Precision

        - Recall
    - Metrics
        - Precision x Recall curve
        - Average Precision
            - 11-point interpolation
            - Interpolating all points
            - An ilustrated example
            - Calculating the 11-point interpolation
            - Calculating the interpolation performed in all points
    - How to use this project
        - Create the ground truth files
        - Create your detection files
        - Optional arguments
    - References

## Different competitions, different metrics

- **PASCAL VOC Challenge** offers a Matlab script in order to evaluate the quality of the detected objects. Participants of the competition can use the provided Matlab script to measure the accuracy of their detections before submitting their results. The official documentation explaining their criteria for object detection metrics can be accessed here. The current metrics used by the current PASCAL VOC object detection challenge are the **Precision x Recall curve** and **Average Precision**. The PASCAL VOC Matlab evaluation code reads the ground truth bounding boxes from XML files, requiring changes in the code if you want to apply it to other datasets or to your speficic cases. Even though projects such as Faster-RCNN implement PASCAL VOC evaluation metrics, it is also necessary to convert the detected bounding boxes into their specific format. Tensorflow framework also has their PASCAL VOC metrics implementation.

- **COCO Detection Challenge** uses different metrics to evaluate the accuracy of object detection of different algorithms. Here you can find a documentation explaining the 12 metrics used for characterizing the performance of an object detector on COCO. This competition offers Python and Matlab codes so users can verify their scores before submitting the results. It is also necessary to convert the results to a format required by the competition.

- **Google Open Images Dataset V4 Competition** also uses mean Average Precision (mAP) over the 500 classes to evaluate the object detection task.

- **ImageNet Object Localization Challenge** defines an error for each image considering the class and the overlapping region between ground truth and detected boxes. The total error is computed as the average of all min errors among all test dataset images. Here are more details about their evaluation method.

## Important definitions

### Intersection Over Union (IOU)

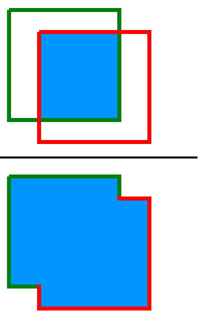
Intersection Over Union (IOU) is measure based on Jaccard Index that evaluates the overlap between two bounding boxes. It requires a ground truth bounding box $B_{gt}$ and a predicted bounding box $B_p$. By

applying the IOU we can tell if a detection is valid (True Positive) or not (False Positive).
IOU is given by the overlapping area between the predicted bounding box and the ground truth bounding box divided by the area of union between them:

$$\text{IOU} = \frac{\text{area}\left(B_p \cap B_{gt}\right)}{\text{area}\left(B_p \cup B_{gt}\right)}$$

The image below illustrates the IOU between a ground truth bounding box (in green) and a detected bounding box (in red).

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$



### True Positive, False Positive, False Negative and True Negative

Some basic concepts used by the metrics:

- **True Positive (TP)**: A correct detection. Detection with IOU ≥ *threshold*
- **False Positive (FP)**: A wrong detection. Detection with IOU < *threshold*
- **False Negative (FN)**: A ground truth not detected
- **True Negative (TN)**: Does not apply. It would represent a corrected misdetection. In the object detection task there are many possible bounding boxes that should not be detected within an image. Thus, TN would be all possible bounding boxes that were corrrectly not detected (so many possible boxes within an image). That's why it is not used by the metrics.

*threshold*: depending on the metric, it is usually set to 50%, 75% or 95%.

### Precision

Precision is the ability of a model to identify **only** the relevant objects. It is the percentage of correct positive predictions and is given by:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

### Recall

Recall is the ability of a model to find all the relevant cases (all ground truth bounding boxes). It is the percentage of true positive detected among all relevant ground truths and is given by:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$

## Metrics

In the topics below there are some comments on the most popular metrics used for object detection.

### Precision x Recall curve

The Precision x Recall curve is a good way to evaluate the performance of an object detector as the confidence is changed by plotting a curve for each object class. An object detector of a particular class is considered good if its precision stays high as recall increases, which means that if you vary the confidence threshold, the precision and recall will still be high. Another way to identify a good object detector is to look for a detector that can identify only relevant objects (0 False Positives = high precision), finding all ground truth objects (0 False Negatives = high recall).

A poor object detector needs to increase the number of detected objects (increasing False Positives = lower precision) in order to retrieve all ground truth objects (high recall). That's why the Precision x Recall curve usually starts with high precision values, decreasing as recall increases. You can see an example of the Prevision x Recall curve in the next topic (Average Precision). This kind of curve is used by the PASCAL VOC 2012 challenge and is available in our implementation.

### Average Precision

Another way to compare the performance of object detectors is to calculate the area under the curve (AUC) of the Precision x Recall curve. As AP curves are often zigzag curves going up and down, comparing different curves (different detectors) in the same plot usually is not an easy task - because the curves tend to cross each other much frequently. That's why Average Precision (AP), a numerical metric, can also help us compare different detectors. In practice AP is the precision averaged across all recall values between 0 and 1.

From 2010 on, the method of computing AP by the PASCAL VOC challenge has changed. Currently, **the interpolation performed by PASCAL VOC challenge uses all data points, rather than interpolating only 11 equally spaced points as stated in their paper**. As we want to reproduce their default implementation, our default code (as seen further) follows their most recent application (interpolating all data points). However, we also offer the 11-point interpolation approach.

#### 11-point interpolation

The 11-point interpolation tries to summarize the shape of the Precision x Recall curve by averaging the precision at a set of eleven equally spaced recall levels [0, 0.1, 0.2, ... , 1]:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{\text{interp}(r)}$$

with

$$\rho_{\text{interp}(r)} = \max_{\tilde{r}:\tilde{r} \geqslant r} \rho\left(\tilde{r}\right)$$

where $\rho\left(\tilde{r}\right)$ is the measured precision at recall $\tilde{r}$.

Instead of using the precision observed at each point, the AP is obtained by interpolating the precision only at the 11 levels $r$ taking the **maximum precision whose recall value is greater than** $r$.

**Interpolating all points**

Instead of interpolating only in the 11 equally spaced points, you could interpolate through all points in such way that:

$$\sum_{r=0}^{1} \left(r_{n+1} - r_n\right) \rho_{interp}\left(r_{n+1}\right)$$

with

$$\rho_{interp}\left(r_{n+1}\right) = \max_{\tilde{r}:\tilde{r} \geq r_{n+1}} \rho\left(\tilde{r}\right)$$

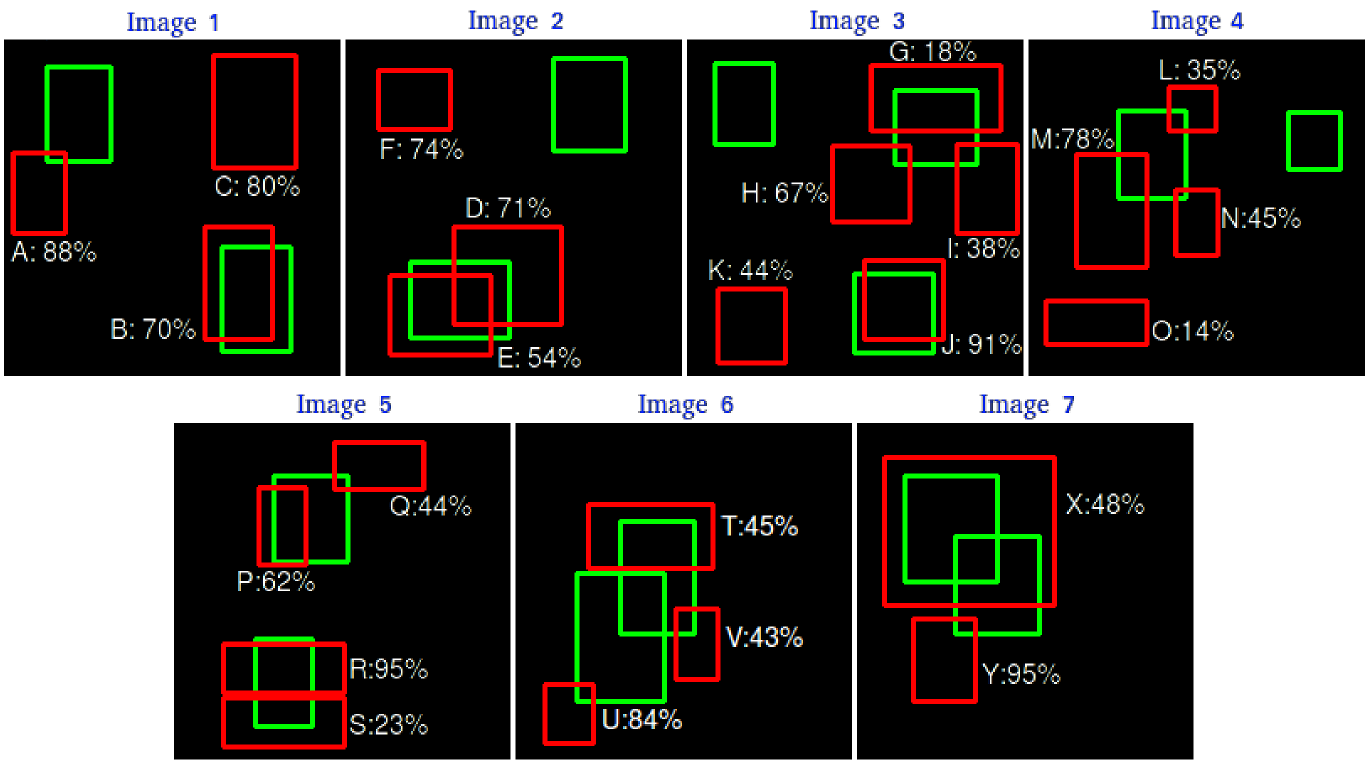
where $\rho\left(\tilde{r}\right)$ is the measured precision at recall $\tilde{r}$.

In this case, instead of using the precision observed at only few points, the AP is now obtained by interpolating the precision at **each level**, $r$ taking the **maximum precision whose recall value is greater or equal than** $r+1$. This way we calculate the estimated area under the curve.

To make things more clear, we provided an example comparing both interpolations.

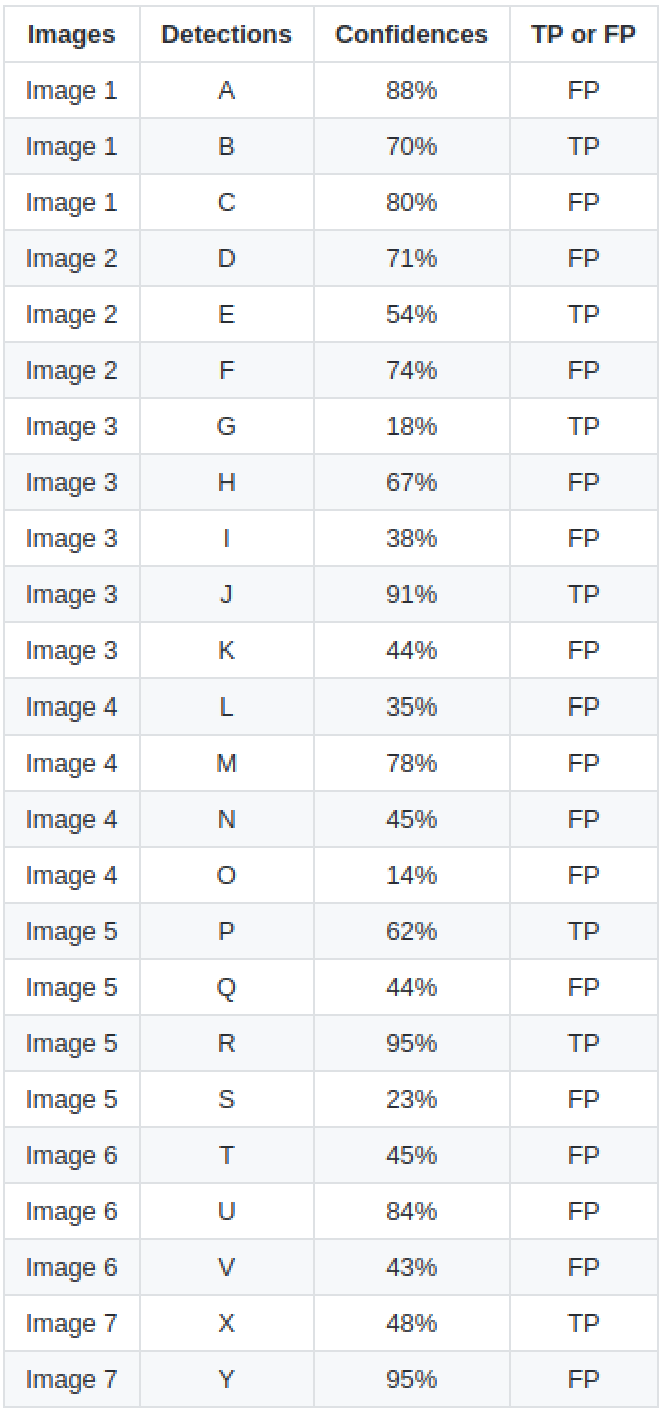
**An ilustrated example**

An example helps us understand better the concept of the interpolated average precision. Consider the detections below:



There are 7 images with 15 ground truth objects representented by the green bounding boxes and 24 detected objects represented by the red bounding boxes. Each detected object has a confidence level and is identified by a letter (A,B,...,Y).

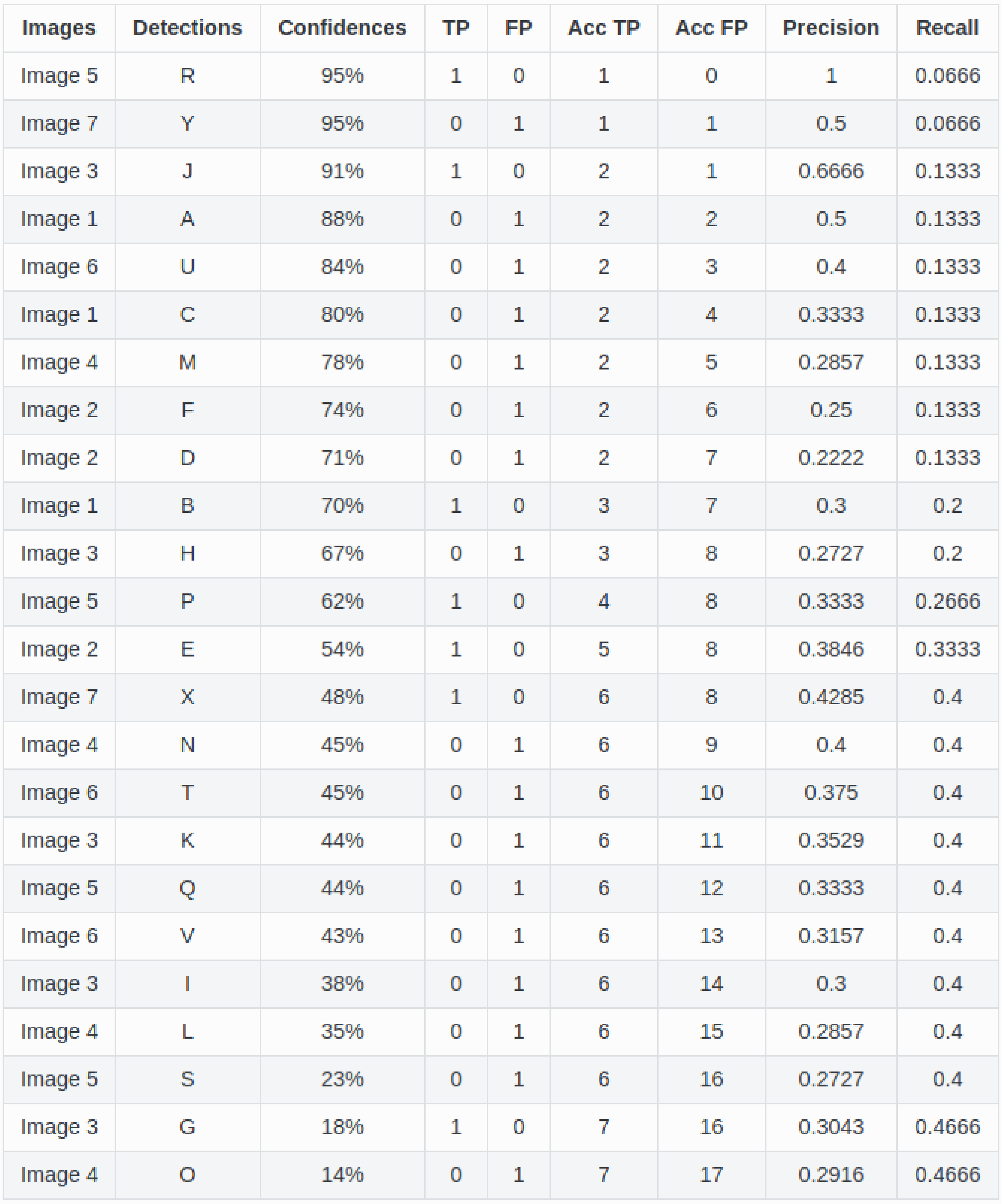
The following table shows the bounding boxes with their corresponding confidences. The last column identifies the detections as TP or FP. In this example a TP is considered if IOU $\geq$ 30%, otherwise it is a FP. By looking at the images above we can roughly tell if the detections are TP or FP.



| Images | Detections | Confidences | TP or FP |
|---|---|---|---|
| Image 1 | A | 88% | FP |
| Image 1 | B | 70% | TP |
| Image 1 | C | 80% | FP |
| Image 2 | D | 71% | FP |
| Image 2 | E | 54% | TP |
| Image 2 | F | 74% | FP |
| Image 3 | G | 18% | TP |
| Image 3 | H | 67% | FP |
| Image 3 | I | 38% | FP |
| Image 3 | J | 91% | TP |
| Image 3 | K | 44% | FP |
| Image 4 | L | 35% | FP |
| Image 4 | M | 78% | FP |
| Image 4 | N | 45% | FP |
| Image 4 | O | 14% | FP |
| Image 5 | P | 62% | TP |
| Image 5 | Q | 44% | FP |
| Image 5 | R | 95% | TP |
| Image 5 | S | 23% | FP |
| Image 6 | T | 45% | FP |
| Image 6 | U | 84% | FP |
| Image 6 | V | 43% | FP |
| Image 7 | X | 48% | TP |
| Image 7 | Y | 95% | FP |

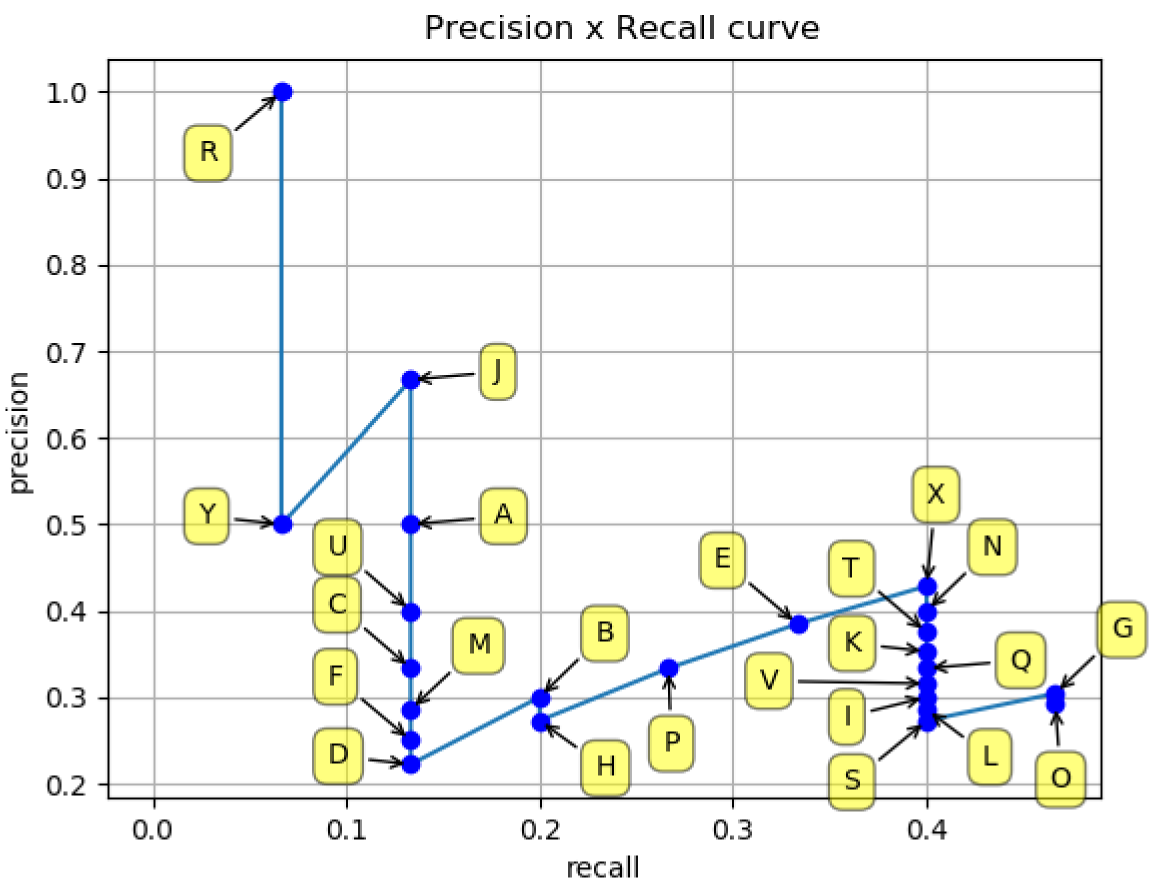
In some images there are more than one detection overlapping a ground truth (Images 2, 3, 4, 5, 6 and 7). For those cases the detection with the highest IOU is considered TP and the others are considered FP. This rule is applied by the PASCAL VOC 2012 metric: "e.g. 5 detections (TP) of a single object is counted as 1 correct detection and 4 false detections”.

The Precision x Recall curve is plotted by calculating the precision and recall values of the accumulated TP or FP detections. For this, first we need to order the detections by their confidences, then we calculate the precision and recall for each accumulated detection as shown in the table below:



| Images | Detections | Confidences | TP | FP | Acc TP | Acc FP | Precision | Recall |
|---|---|---|---|---|---|---|---|---|
| Image 5 | R | 95% | 1 | 0 | 1 | 0 | 1 | 0.0666 |
| Image 7 | Y | 95% | 0 | 1 | 1 | 1 | 0.5 | 0.0666 |
| Image 3 | J | 91% | 1 | 0 | 2 | 1 | 0.6666 | 0.1333 |
| Image 1 | A | 88% | 0 | 1 | 2 | 2 | 0.5 | 0.1333 |
| Image 6 | U | 84% | 0 | 1 | 2 | 3 | 0.4 | 0.1333 |
| Image 1 | C | 80% | 0 | 1 | 2 | 4 | 0.3333 | 0.1333 |
| Image 4 | M | 78% | 0 | 1 | 2 | 5 | 0.2857 | 0.1333 |
| Image 2 | F | 74% | 0 | 1 | 2 | 6 | 0.25 | 0.1333 |
| Image 2 | D | 71% | 0 | 1 | 2 | 7 | 0.2222 | 0.1333 |
| Image 1 | B | 70% | 1 | 0 | 3 | 7 | 0.3 | 0.2 |
| Image 3 | H | 67% | 0 | 1 | 3 | 8 | 0.2727 | 0.2 |
| Image 5 | P | 62% | 1 | 0 | 4 | 8 | 0.3333 | 0.2666 |
| Image 2 | E | 54% | 1 | 0 | 5 | 8 | 0.3846 | 0.3333 |
| Image 7 | X | 48% | 1 | 0 | 6 | 8 | 0.4285 | 0.4 |
| Image 4 | N | 45% | 0 | 1 | 6 | 9 | 0.4 | 0.4 |
| Image 6 | T | 45% | 0 | 1 | 6 | 10 | 0.375 | 0.4 |
| Image 3 | K | 44% | 0 | 1 | 6 | 11 | 0.3529 | 0.4 |
| Image 5 | Q | 44% | 0 | 1 | 6 | 12 | 0.3333 | 0.4 |
| Image 6 | V | 43% | 0 | 1 | 6 | 13 | 0.3157 | 0.4 |
| Image 3 | I | 38% | 0 | 1 | 6 | 14 | 0.3 | 0.4 |
| Image 4 | L | 35% | 0 | 1 | 6 | 15 | 0.2857 | 0.4 |
| Image 5 | S | 23% | 0 | 1 | 6 | 16 | 0.2727 | 0.4 |
| Image 3 | G | 18% | 1 | 0 | 7 | 16 | 0.3043 | 0.4666 |
| Image 4 | O | 14% | 0 | 1 | 7 | 17 | 0.2916 | 0.4666 |

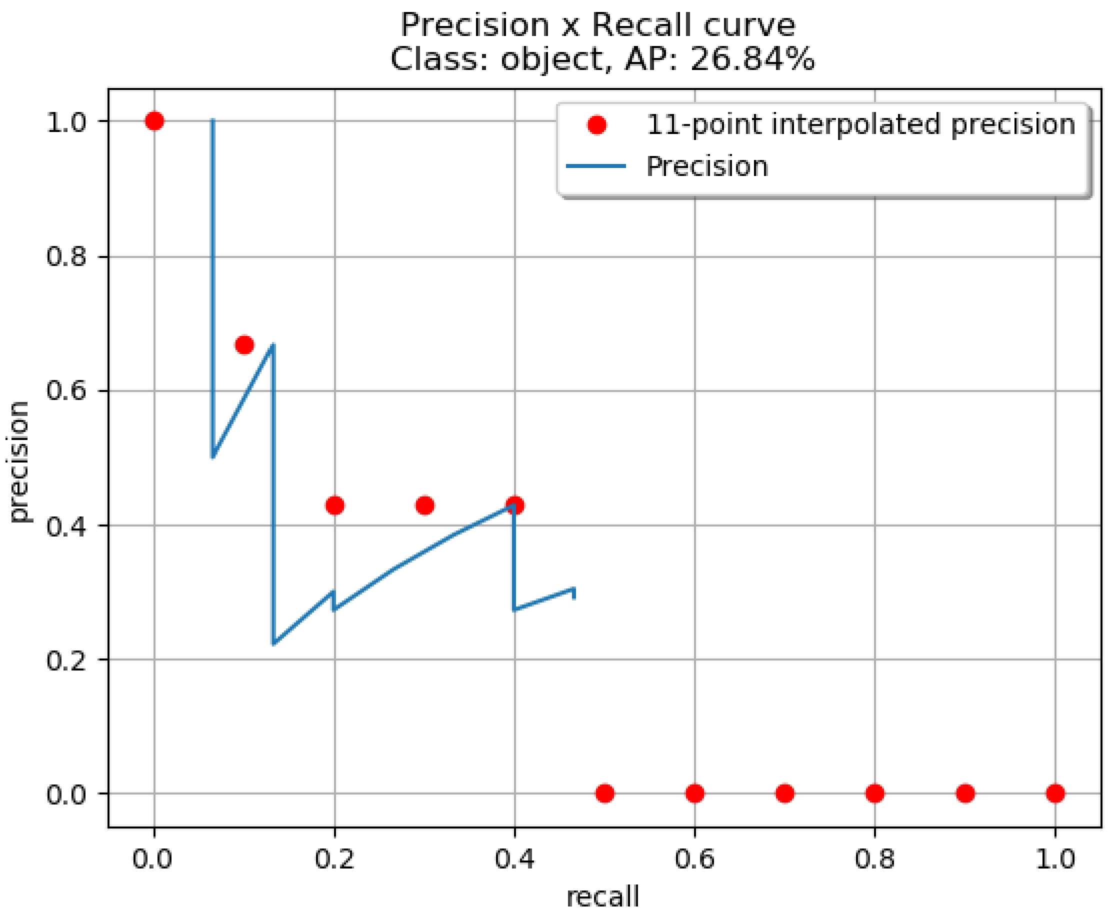
Plotting the precision and recall values we have the following *Precision x Recall curve*:



As mentioned before, there are two different ways to measure the interpolted average precision: **11-point interpolation** and **interpolating all points**. Below we make a comparisson between them:

**Calculating the 11-point interpolation**

The idea of the 11-point interpolated average precision is to average the precisions at a set of 11 recall levels (0,0.1,...,1). The interpolated precision values are obtained by taking the maximum precision whose recall value is greater than its current recall value as follows:



By applying the 11-point interpolation, we have:

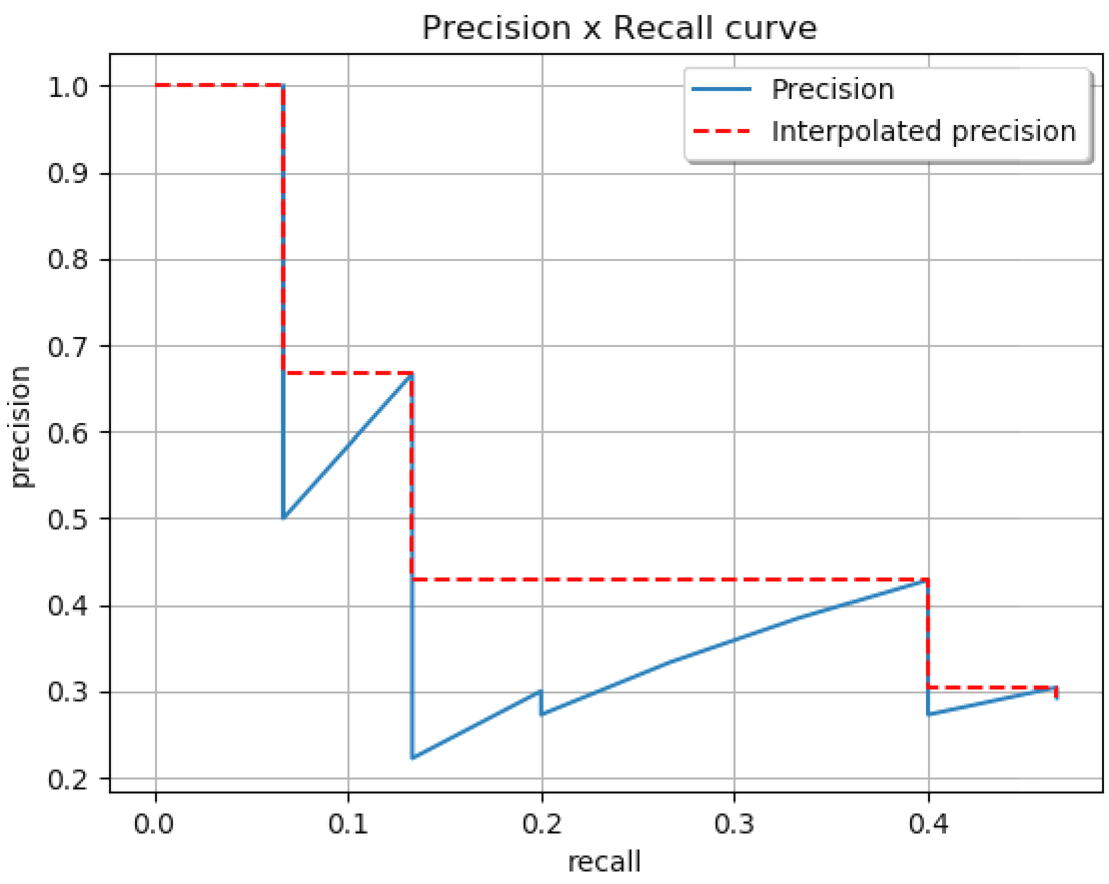
$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{\text{interp}(r)}$$

$$AP = \frac{1}{11}(1 + 0.6666 + 0.4285 + 0.4285 + 0.4285 + 0 + 0 + 0 + 0 + 0 + 0)$$

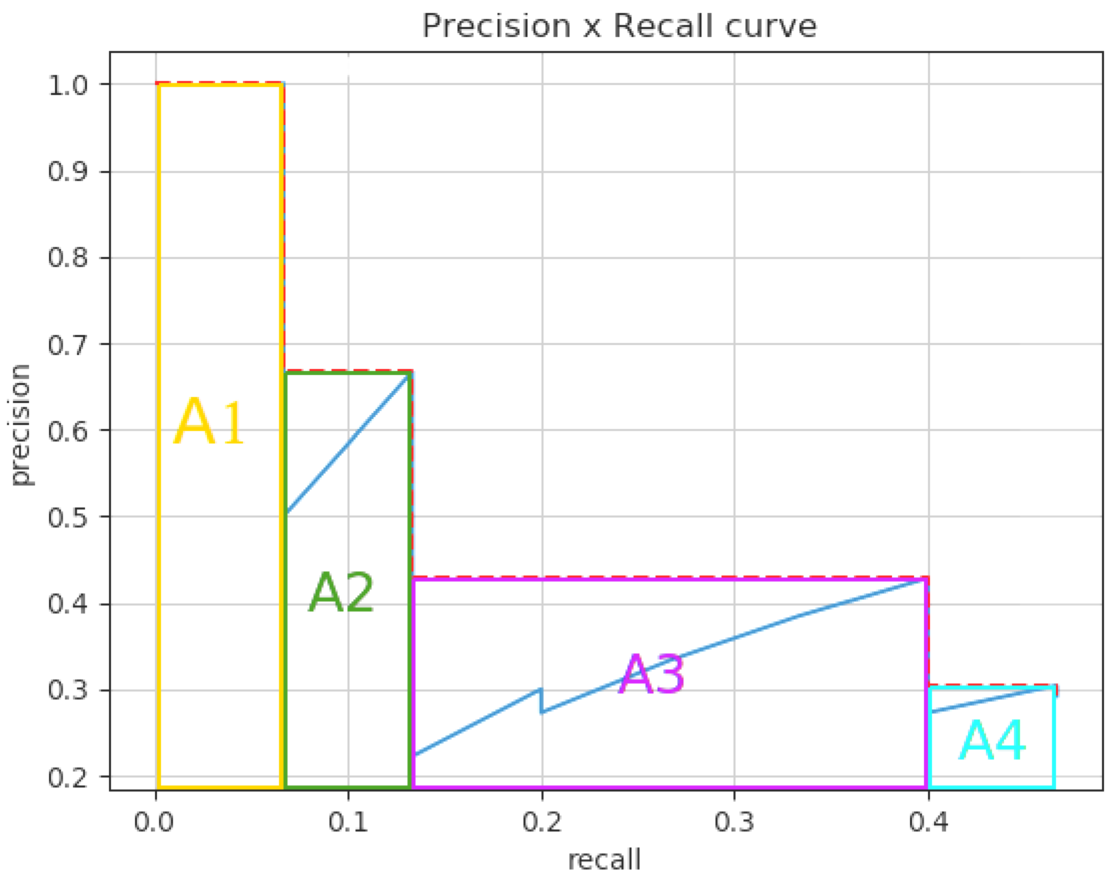
$$AP = 26.84\%$$

**Calculating the interpolation performed in all points**

By interpolating all points, the Average Precision (AP) can be interpreted as an approximated AUC of the Precision x Recall curve. The intention is to reduce the impact of the wiggles in the curve. By applying the equations presented before, we can obtain the areas as it will be demostrated here. We could also visually have the interpolated precision points by looking at the recalls starting from the highest (0.4666) to 0 (looking at the plot from right to left) and, as we decrease the recall, we collect the precision values that are the highest as shown in the image below:



Looking at the plot above, we can divide the AUC into 4 areas (A1, A2, A3 and A4):



Calculating the total area, we have the AP:

$$AP = A1 + A2 + A3 + A4$$

With

$$A1 = (0.0666 - 0) \times 1 = \mathbf{0.0666}$$
$$A2 = (0.1333 - 0.0666) \times 0.6666 = \mathbf{0.04446222}$$
$$A3 = (0.4 - 0.1333) \times 0.4285 = \mathbf{0.11428095}$$
$$A4 = (0.4666 - 0.4) \times 0.3043 = \mathbf{0.02026638}$$

$$AP = 0.0666 + 0.04446222 + 0.11428095 + 0.02026638$$
$$AP = 0.24560955$$
$$AP = \mathbf{24.56\%}$$

The results between the two different interpolation methods are a little different: 24.56% and 26.84% by the every point interpolation and the 11-point interpolation respectively.

Our default implementation is the same as VOC PASCAL: every point interpolation. If you want to use the 11-point interpolation, change the functions that use the argument `method=MethodAveragePrecision.EveryPointInterpolation` to `method=MethodAveragePrecision.ElevenPointInterpolation`.

If you want to reproduce these results, see the **Sample 2**.

## How to use this project

This project was created to evaluate your detections in a very easy way. If you want to evaluate your algorithm with the most used object detection metrics, you are in the right place.

Sample_1 and sample_2 are practical examples demonstrating how to access directly the core functions of this project, providing more flexibility on the usage of the metrics. But if you don't want to spend your time understanding our code, see the instructions below to easily evaluate your detections:

Follow the steps below to start evaluating your detections:

1. Create the ground truth files
2. Create your detection files
3. For **Pascal VOC metrics**, run the command: `python pascalvoc.py`
   If you want to reproduce the example above, run the command: `python pascalvoc.py -t 0.3`
4. (Optional) You can use arguments to control the IOU threshold, bounding boxes format, etc.

### Create the ground truth files

- Create a separate ground truth text file for each image in the folder **groundtruths/**.
- In these files each line should be in the format: `<class_name> <left> <top> <right> <bottom>`.
- E.g. The ground truth bounding boxes of the image "2008_000034.jpg" are represented in the file "2008_000034.txt":

```
bottle 6 234 45 362
person 1 156 103 336
person 36 111 198 416
person 91 42 338 500
```

If you prefer, you can also have your bounding boxes in the format: `<class_name> <left> <top> <width> <height>` (see here * how to use it). In this case, your "2008_000034.txt" would be represented as:

```
bottle 6 234 39 128
person 1 156 102 180
person 36 111 162 305
person 91 42 247 458
```

## Create your detection files

- Create a separate detection text file for each image in the folder **detections/**.
- The names of the detection files must match their correspond ground truth (e.g. "detections/2008_000182.txt" represents the detections of the ground truth: "groundtruths/2008_000182.txt").
- In these files each line should be in the following format: `<class_name> <confidence> <left> <top> <right> <bottom>` (see here * how to use it).
- E.g. "2008_000034.txt":

  ```
  bottle 0.14981 80 1 295 500
  bus 0.12601 36 13 404 316
  horse 0.12526 430 117 500 307
  pottedplant 0.14585 212 78 292 118
  tvmonitor 0.070565 388 89 500 196
  ```

Also if you prefer, you could have your bounding boxes in the format: `<class_name> <left> <top> <width> <height>`.

## Optional arguments

Optional arguments:

| Argument | Description | Example | Default |
|---|---|---|---|
| `-h`, `--help` | show help message | `python pascalvoc.py -h` | |
| `-v`, `--version` | check version | `python pascalvoc.py -v` | |

| Argument | Description | Example | Default |
|---|---|---|---|
| `-gt`, `--gtfolder` | folder that contains the ground truth bounding boxes files | `python pascalvoc.py -gt /home/whatever/my_groundtruths/` | `/Object-Detection-Metrics/groundtruths` |
| `-det`, `--detfolder` | folder that contains your detected bounding boxes files | `python pascalvoc.py -det /home/whatever/my_detections/` | `/Object-Detection-Metrics/detections/` |
| `-t`, `--threshold` | IOU thershold that tells if a detection is TP or FP | `python pascalvoc.py -t 0.75` | `0.50` |
| `-gtformat` | format of the coordinates of the ground truth bounding boxes * | `python pascalvoc.py -gtformat xyrb` | `xywh` |
| `-detformat` | format of the coordinates of the detected bounding boxes * | `python pascalvoc.py -detformat xyrb` | `xywh` |

| Argument | Description | Example | Default |
|---|---|---|---|
| `-gtcoords` | reference of the ground truth bounding bounding box coordinates. If the annotated coordinates are relative to the image size (as used in YOLO), set it to `rel`. If the coordinates are absolute values, not depending to the image size, set it to `abs` | `python pascalvoc.py -gtcoords rel` | `abs` |
| `-detcoords` | reference of the detected bounding bounding box coordinates. If the coordinates are relative to the image size (as used in YOLO), set it to `rel`. If the coordinates are absolute values, not depending to the image size, set it to `abs` | `python pascalvoc.py -detcoords rel` | `abs` |

| Argument | Description | Example | Default |
|---|---|---|---|
| `-imgsize` | image size in the format `width,height` <int,int>. Required if `-gtcoords` or `-detcoords` is set to `rel` | `python pascalvoc.py -imgsize 600,400` | |
| `-sp`, `--savepath` | folder where the plots are saved | `python pascalvoc.py -sp /home/whatever/my_results/` | `Object-Detection-Metrics/results/` |
| `-np`, `--noplot` | if present no plot is shown during execution | `python pascalvoc.py -np` | not presented. Therefore, plots are shown |

(*****) set `-gtformat xywh` and/or `-detformat xywh` if format is `<left> <top> <width> <height>`. Set to `-gtformat xyrb` and/or `-detformat xyrb` if format is `<left> <top> <right> <bottom>`.

## References

- The Relationship Between Precision-Recall and ROC Curves (Jesse Davis and Mark Goadrich) Department of Computer Sciences and Department of Biostatistics and Medical Informatics, University of Wisconsin
  http://pages.cs.wisc.edu/~jdavis/davisgoadrichcamera2.pdf
- The PASCAL Visual Object Classes (VOC) Challenge
  http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.5766&rep=rep1&type=pdf
- Evaluation of ranked retrieval results (Salton and Mcgill 1986)
  https://www.amazon.com/Introduction-Information-Retrieval-COMPUTER-SCIENCE/dp/0070544840
  https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ranked-retrieval-results-1.html