# Computer Vision - YOLO Algorithm

a report for Artificial Intelligence 220621-D course by prof. dr. hab. Michał Bernardelli

Bartosz Bogucki 82801, Mateusz Baczewski 82341

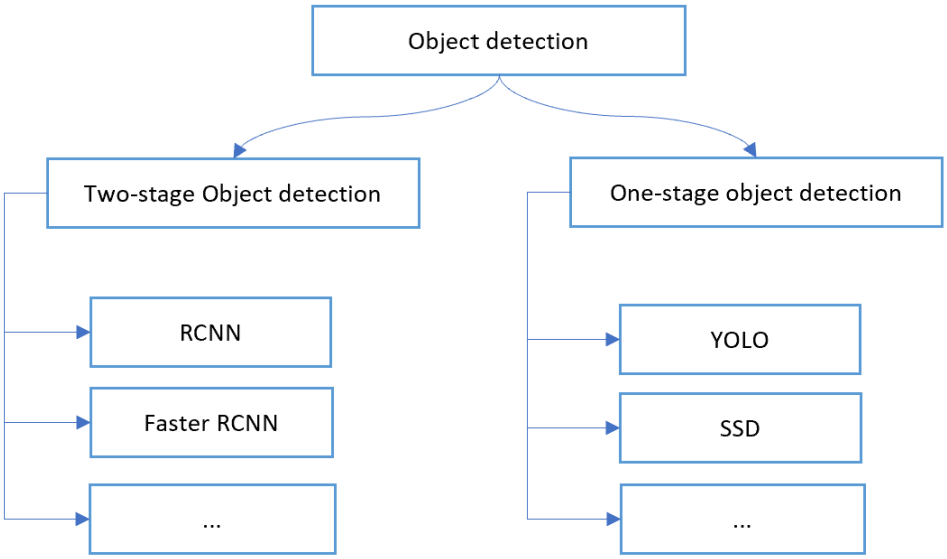

05.06.2023

Table of contents

## 1. Definition of object detection

Object detection is a crucial task in computer vision that involves identifying and locating objects in images or videos. It has various applications, including surveillance, self-driving cars, and robotics.

Object detection algorithms can be categorized into single-shot detectors and two-stage detectors:




Traditional object detection algorithms involve two steps: first, they generate a set of region proposals that are likely to contain objects, and then they classify those regions. This two-step process can be computationally expensive and time-consuming.

In contrast, single-shot object detection, such as the YOLO (You Only Look Once) algorithm, uses a single pass of the input image to make predictions about the presence and location of objects in the image. It processes an entire image in a single pass, making them computationally efficient.

## 2. YOLO algorithm

### 2.1. Introduction

The YOLO (You Only Look Once) algorithm is a real-time object detection algorithm that was introduced by Joseph Redmon, et al. in 2015. YOLO revolutionized the field of

computer vision by introducing a single-stage object detection approach that achieves good application in real-time problems.

Yolo is:

- possible to be used with many objects of different classes
- so fast that can be used in real-time problems
- a leading approach for computer vision and object detection
- a family of algorithms that continuously come out in better version (newest: v8)

## 2.2. How does it work?

The YOLO algorithm operates by dividing the input image into a grid and making predictions for object bounding boxes, class probabilities, and confidence scores for each grid cell. The key steps involved in YOLO's operation are as follows:

1. Divide the input image into a grid of cells.
2. Define anchor boxes of different sizes and aspect ratios.
3. Label the data by creating a vector for each grid cell.
4. Train the model using the labeled data.
5. Predict bounding box coordinates and class probabilities for each grid cell.
6. Remove low probability bounding boxes.
7. Apply non-maximum suppression to remove overlapping boxes.
8. Output the remaining bounding boxes with their class labels and confidence scores as the detected objects.

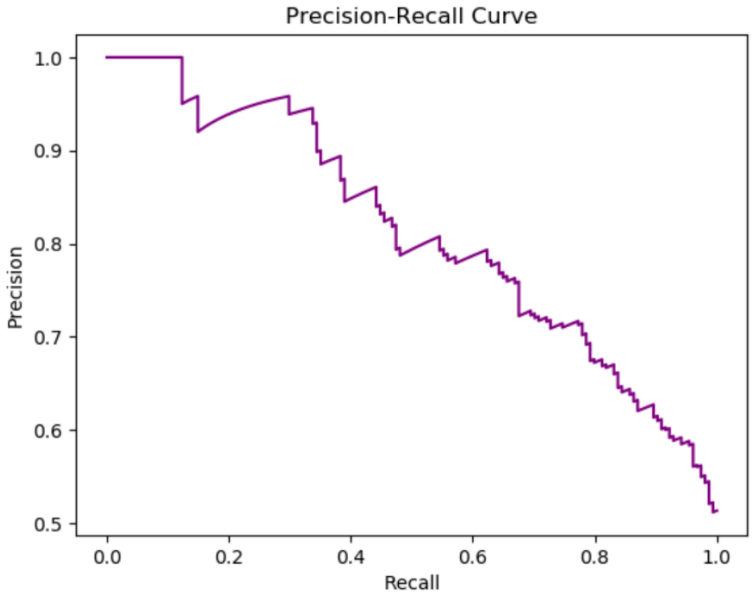
## 2.3. Training and Testing

During training, the backbone convolutional neural network in YOLO extracts high-level features from the input image, while the detection layers predict bounding boxes and class probabilities based on these features extracted by the backbone network. The loss function used in YOLO guides the training process by quantifying the difference between the predicted bounding boxes and the ground truth annotations consisting of multiple components: objectness loss, localization loss, and class prediction loss. The activation

function used in YOLO is typically the Leaky ReLU (Rectified Linear Unit) activation function.

Traditional image classification metrics cannot be directly applied to object detection, as the task involves both object classification and localization. The evaluation metric for object detection needs to consider both the class and location of the objects. Mean Average Precision (mAP) and F1-score are commonly used metrics in object detection. Therefore, the ultimate goal is to achieve the maximum scores for both precision and recall.

The plotting of a precision-recall curve depicts the trade-off between precision and recall based on different thresholds (probability that a box contains an object):

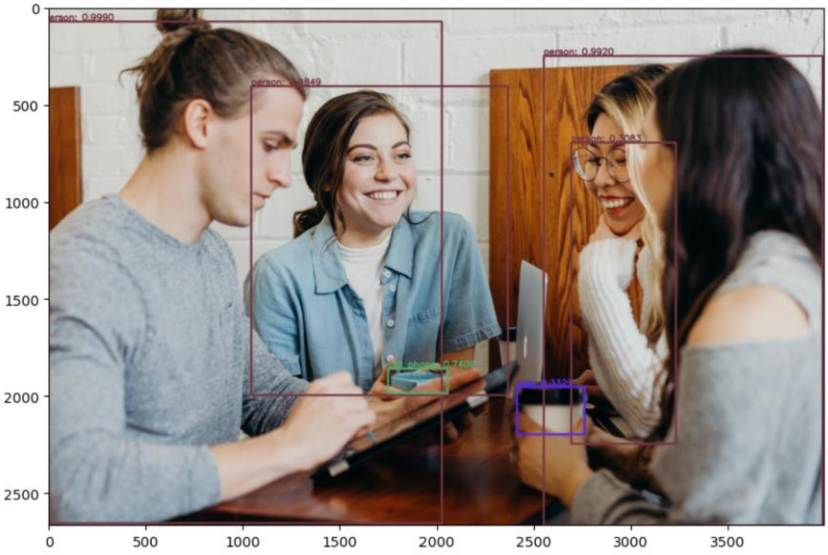



Therefore, it helps to select the best threshold that maximizes both precision and recall. To find out the best precision and recall, we use F1-score to combine precision and recall into a single metric by taking their harmonic mean.

Average Precision (AP) finds the area under the precision-recall curve. At each recall level, we replace each precision value with the maximum precision value to the right of that recall level. We can calculate the mAP by simply taking the mean over all the class APs and use this measure to evaluate the algorithm overall.

## 3. Code Implementation

The provided Python code demonstrates a YOLO implementation for object detection. First, a pre-trained model is loaded, obtaining pre-trained weights and architecture configuration files for the YOLO model. These files contain the pre-trained parameters and network architecture that enable the model to detect objects in images. A list of class labels that the model has been trained to recognize is then loaded. This list contains the names of the different classes of objects that the model can detect, such as "cup," "person," "cell-phone," etc. We then load an input image to perform object detection on it. We also preprocess the image, resizing it to the size required by the YOLO model and normalizing the pixel values to the range of 0 to 1. We then set the preprocessed image as input to the YOLO model and perform a forward pass through the network. This will generate predictions for bounding boxes, class labels and confidence scores for each detected object. We then process the output of the YOLO model to extract relevant information, such as significant vectors. A confidence threshold of 0.3 is also applied to remove detections with low confidence scores. To eliminate overlapping envelope detections, we also apply non-maximum suppression (NMS) to the filtered detections, setting the NMS threshold to 0.3. NMS selects the most confident bounding box among the overlapping boxes for each object instance. Finally, we draw the remaining bounding boxes on the original image and display the labels and confidence scores associated with each detected object. This step helps visualize the results of the object detection algorithm:



## 4. Conclusion

The YOLO algorithm is a popular and efficient approach for real-time object detection in images. It revolutionized object detection by introducing a unified framework that predicts bounding boxes and class probabilities directly from raw pixel data, in a single pass. YOLO offers several advantages over traditional object detection approaches. It is extremely fast, capable of processing images in real-time due to its single-pass architecture. It can detect multiple objects in a single image, even if they belong to different classes. YOLO also generalizes well to objects of different sizes and aspect ratios, thanks to the anchor box mechanism. However, YOLO does have some limitations. It may struggle with detecting small objects due to the coarser grid and relatively large anchor boxes. It can also have difficulty with closely packed objects or objects with significant overlap. These limitations can be mitigated to some extent by using higher-resolution inputs or implementing multi-scale inference. Despite its limitations, YOLO has become a popular choice for a wide range of computer vision applications, including autonomous driving, surveillance, and robotics. It strikes a good balance between accuracy and speed, making it suitable for real-time object detection tasks. Overall, YOLO is a powerful and efficient algorithm that has significantly advanced the field of object detection. Its ability to achieve real-time performance and high accuracy makes it a valuable tool for various practical applications.