YOLO Algorithm for Object Detection

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

How the YOLO algorithm works

YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

Residual blocks

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.

In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

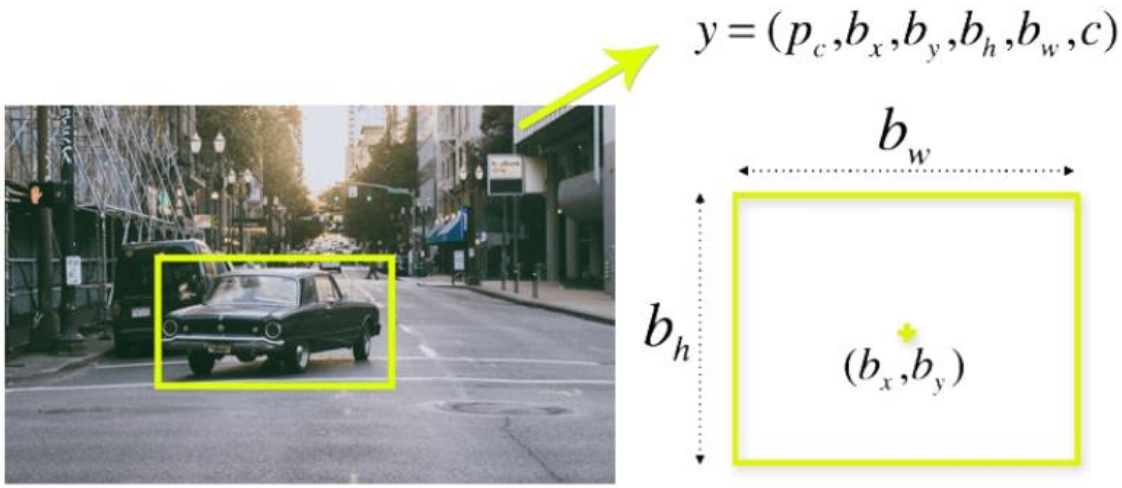
Bounding box regression

A bounding box is an outline that highlights an object in an image.

Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
- Bounding box center (bx,by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.

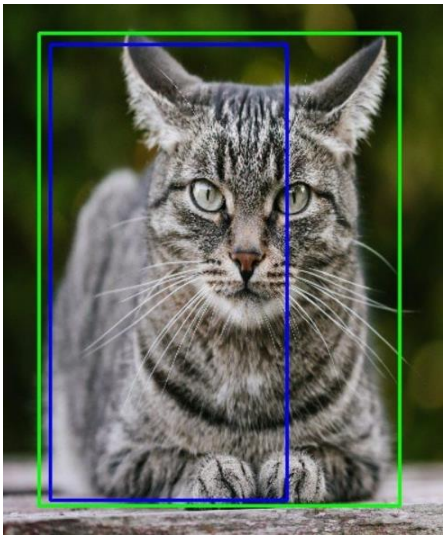


Intersection over union (IOU)

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

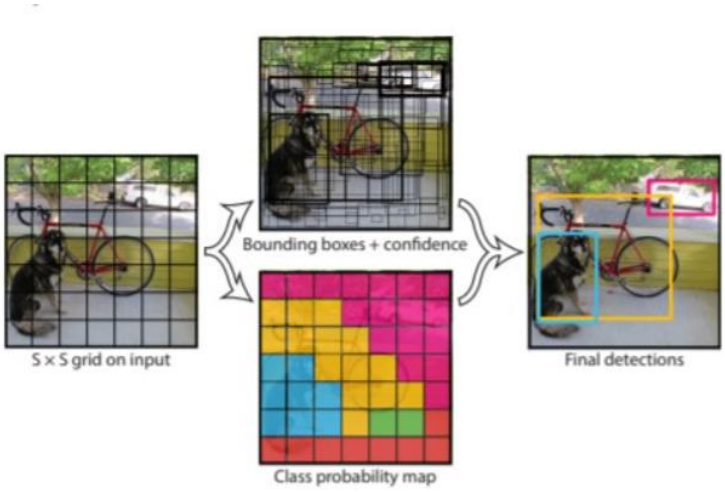
The following image provides a simple example of how IOU works.



Combination of the three techniques

The following image shows how the three techniques are applied to produce the final detection results.

How YOLO Algorithm Works



Applications of YOLO

- Autonomous driving
- Wildlife
- Security
- Object Detection

Implementation In Python:

- Python 3
- Numpy
- OpenCV Python bindings

Python 3
If you are on Ubuntu, it's most likely that Python 3 is already installed. Run python3 in terminal to check whether its installed. If its not installed use

sudo apt-get install python3

For macOS please refer my earlier post on deep learning setup for macOS.

I highly recommend using Python virtualenvironment. Have a look at my earlier post if you need a starting point.

Numpy
pip install numpy

This should install numpy. Make sure pip is linked to Python 3.x (pip -V will show this info)

If needed use pip3. Use sudo apt-get install python3-pip to get pip3 if not already installed.

OpenCV-Python
You need to compile OpenCV from source from the master branch on github to get the Python bindings. (recommended)

Adrian Rosebrock has written a good blog post on PyImageSearch on this. (Download the source from master branch instead of from archive)

If you are overwhelmed by the instructions to get OpenCV Python bindings from source, you can get the unofficial Python package using

pip install opencv-python

This is not maintained officially by OpenCV.org. It's a community maintained one. Thanks to the efforts of Olli-Pekka Heinisuo.

Command line arguments
The script requires four input arguments.

input image
YOLO config file
pre-trained YOLO weights
text file containing class names

All of these files are available on the github repository I have put together. (link to download pre-trained weights is available in readme.)

You can also download the pre-trained weights in Terminal by typing

wget https://pjreddie.com/media/files/yolov3.weights

This particular model is trained on COCO dataset (common objects in context) from Microsoft. It is capable of detecting 80 common objects. See the full list here.

Input image can be of your choice. Sample input is available in the repo.

Run the script by typing

$ python yolo_opencv.py --image dog.jpg --config yolov3.cfg --weights yolov3.weights --classes yolov3.txt