NETZWERK ACADEMY

Deep Learning Computer Vision

1. Computer Vision

1.1. Introduction

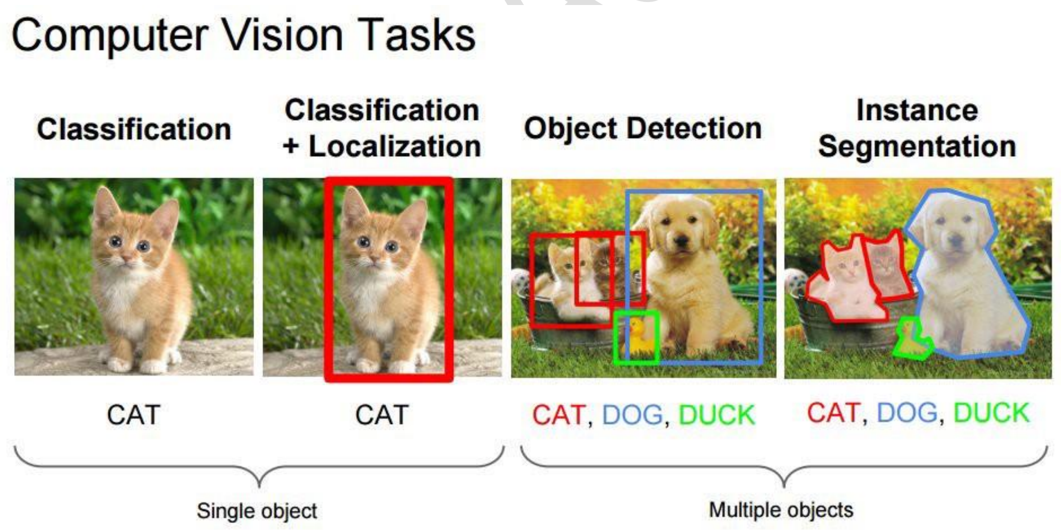
Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields –

- **Image Processing** – It focuses on image manipulation.
- **Pattern Recognition** – It explains various techniques to classify patterns.
- **Photogrammetry** – It is concerned with obtaining accurate measurements from images.

1.2. Computer Vision Tasks

Object Classification: What broad category of object is in this photograph?
Object Identification: Which type of a given object is in this photograph?
Object Verification: Is the object in the photograph?
Object Detection: Where are the objects in the photograph?
Object Landmark Detection: What are the key points for the object in the photograph?
Object Segmentation: What pixels belong to the object in the image?
Object Recognition: What objects are in this photograph and where are they?



1.3. Hierarchy

Computer vision is divided into three basic categories as following –
Low-level vision – It includes process image for feature extraction.
Intermediate-level vision – It includes object recognition and 3D scene interpretation
High-level vision – It includes conceptual description of a scene like activity, intention and behavior.

2. OpenCV

2.1. Introduction

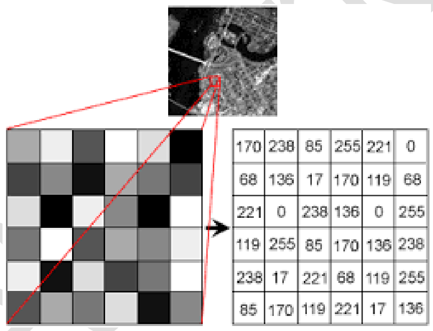
OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

- OpenCV is a Python library which is designed to solve computer vision problems. OpenCV was originally developed in 1999 by Intel but later it was supported by Willow Garage.
- OpenCV supports a wide variety of programming languages such as C++, Python, Java etc. Support for multiple platforms including Windows, Linux, and MacOS.
- OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays.
- This makes it easier to integrate it with other libraries which use NumPy. For example, libraries such as SciPy and Matplotlib.
- Open Source and free
- Easy to use and install

2.2. Operations

- **Reading, Writing and Displaying Images**

  Machines see and process everything using numbers, including images and text. How do you convert images to numbers – I can hear you wondering. Two words – pixel values:

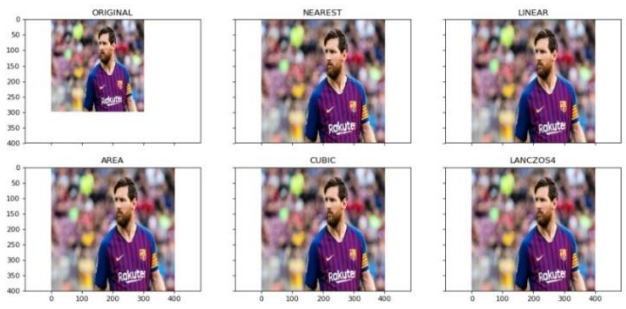


170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

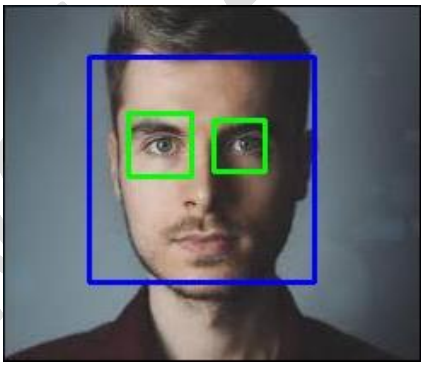
- **Changing Color Spaces**

  A color space is a protocol for representing colors in a way that makes them easily reproducible. We know that grayscale images have single pixel values and color images contain 3 values for each pixel – the intensities of the Red, Green and Blue channels.

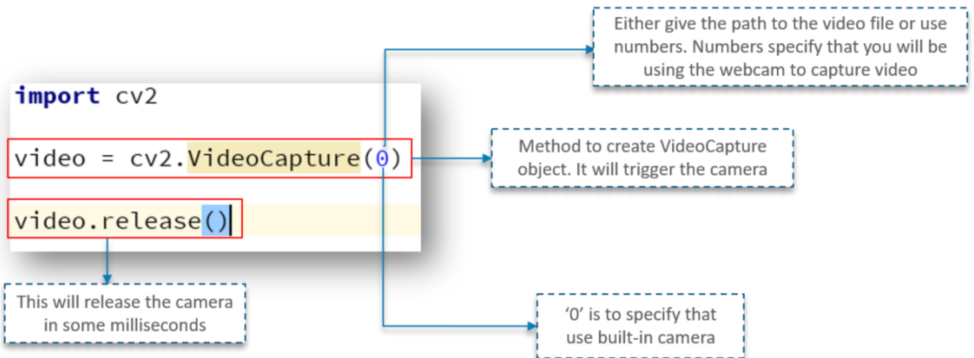
- **Resizing Images**
  Deep learning models work with a fixed sized input. The same idea applies to computer vision models as well. The images we use for training our model must be of the same size.
  Images can be easily scaled up and down using OpenCV. This operation is useful for training deep learning models when we need to convert images to the model's input shape.



- **Face Detection**
  OpenCV supports haar cascade based object detection. Haar cascades are Deep learning algorithms based classifiers that calculate different features like edges, lines, etc in the image. Then, these classifiers train using multiple positive and negative samples.



- **Video Capture**
  Capturing videos using OpenCV is pretty simple as well. the following loop will give you a better idea.



```
import cv2

video = cv2.VideoCapture(0)

video.release()
```

Either give the path to the video file or use numbers. Numbers specify that you will be using the webcam to capture video

Method to create VideoCapture object. It will trigger the camera

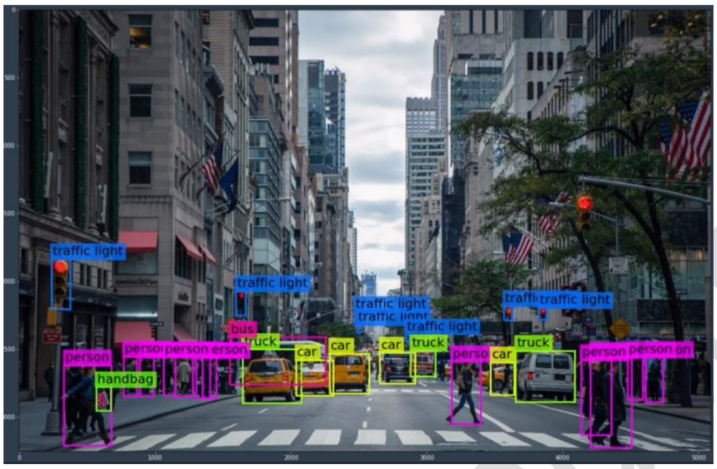
This will release the camera in some milliseconds

'0' is to specify that use built-in camera

3. Yolo

3.1. Introduction

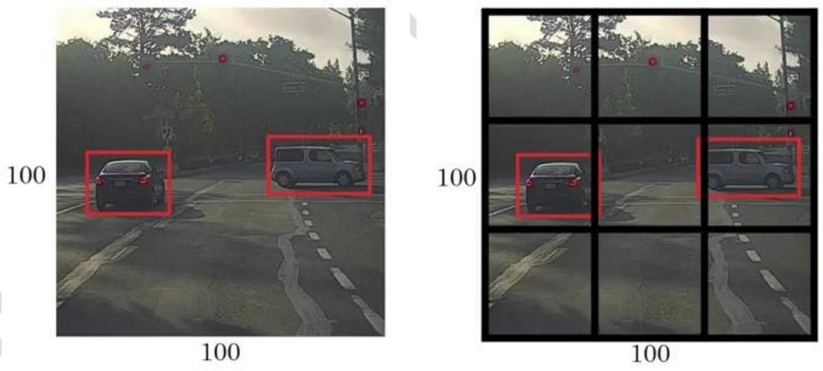
You Only Look Once is a real-time object detection algorithm, that avoids spending too much time on generating region proposals. Instead of locating objects perfectly, it prioritizes speed and recognition



3.2. Functionality

YOLO is such a useful framework; it follows steps for detecting objects in a given image.

- YOLO first takes an input image
- The framework then divides the input image into grids (say a 3 X 3 grid)
- Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects

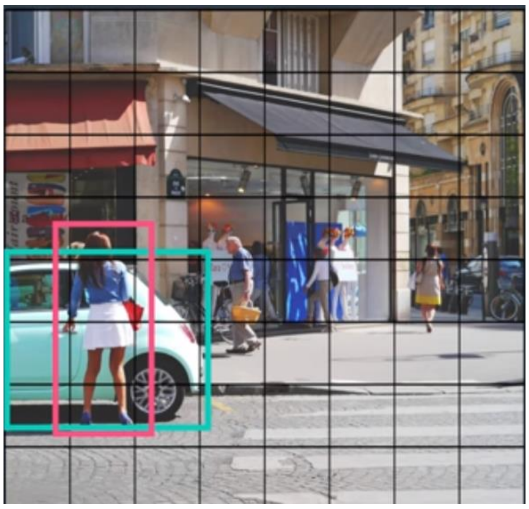


3.3. Anchor Boxes

YOLO can work well for multiple objects where each object is associated with one grid cell. But in the case of overlap, in which one grid cell actually contains the centre points of two different objects, we can use something called anchor boxes to allow one grid cell to detect multiple objects.

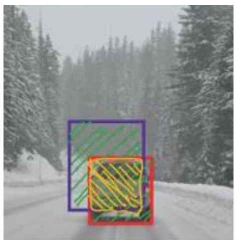
By defining anchor boxes, we can create a longer grid cell vector and associate multiple classes with each grid cell.

Anchor boxes have a defined aspect ratio, and they tried to detect objects that nicely fit into a box with that ratio.



3.4. Non-Maximal Suppression (NMS)

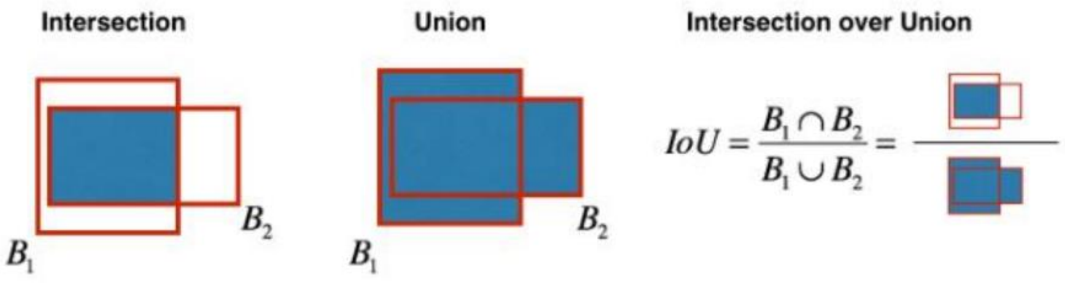
YOLO uses Non-Maximal Suppression (NMS) to only keep the best bounding box. The first step in NMS is to remove all the predicted bounding boxes that have a detection probability that is less than a given NMS threshold. In the code below, we set this NMS threshold to 0.6. This means that all predicted bounding boxes that have a detection probability less than 0.6 will be removed.



3.5. Intersection Over Union Threshold(IOU)

After removing all the predicted bounding boxes that have a low detection probability, the second step in NMS, is to select the bounding boxes with the highest detection probability and eliminate all the bounding boxes whose Intersection Over Union (IOU) value is higher than a given IOU threshold. In the code below, we set this IOU threshold to 0.4. This means that all predicted bounding boxes that have an IOU value greater than 0.4 with respect to the best bounding boxes will be removed.

$$IoU = \frac{Area\ of\ Intersection/Overlap}{Area\ of\ Union}$$



3.6. Making a Prediction

- Center Coordinates
- Dimensions of the Bounding Box
- Objectness Score
- Class Confidences
- Prediction across different scales.
- Output Processing

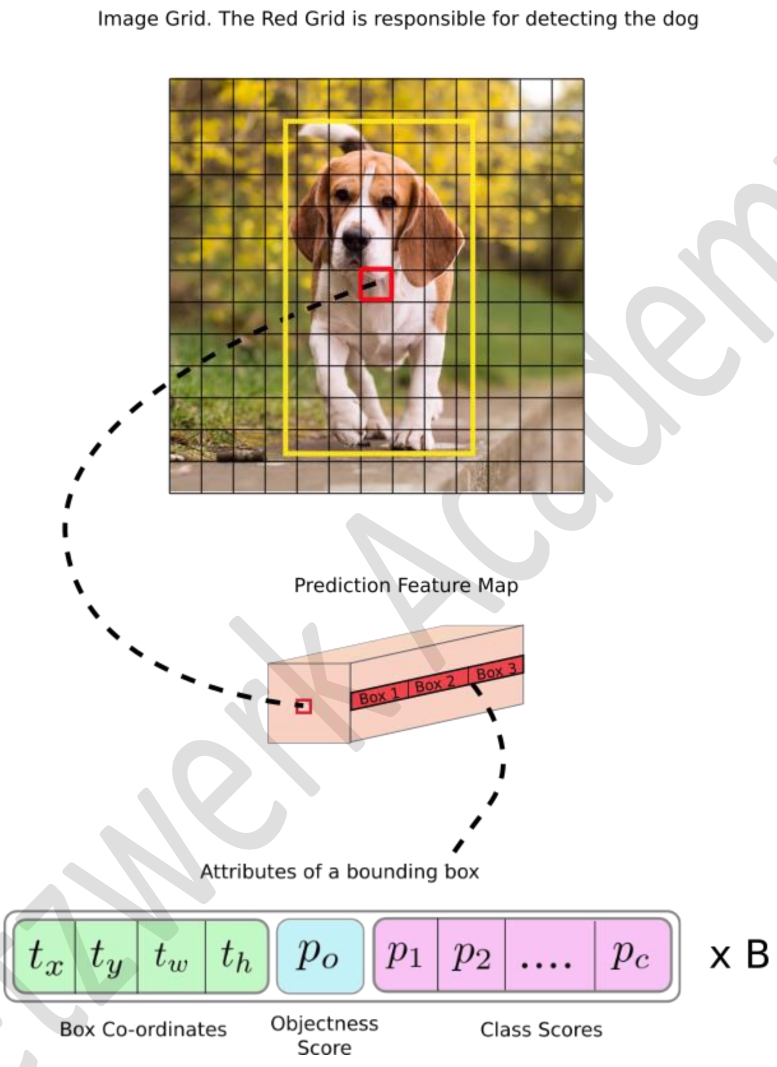


Image Grid. The Red Grid is responsible for detecting the dog
Prediction Feature Map
Box 1
Box 2
Box 3
Attributes of a bounding box
t_x t_y t_w t_h
p_o
p_1 p_2 .... p_c
x B
Box Co-ordinates
Objectness Score
Class Scores