Abstract

Object Detection has been an important topic in the advancement of computer vision systems. With the advent of Neural Network techniques, the accuracy for object detection has increased drastically.

The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance. A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the Neural Network learning based approach, which leads to slow and non-optimal performance.

In this project, we use a You Only Look Once (YOLO) learning based approach to solve the problem of object detection in an end-to-end fashion. The network is trained on the most challenging publicly available dataset (PASCAL VOC,MS COCO), on which a object detection challenge is conducted annually. The resulting system is fast and accurate, thus aiding those applications which require object detection.

ACKNOWLEDGEMENT

It is our pleasure to acknowledge the assistance of a number of people without whose help this project would not have been possible. First and foremost ,We would like to express our gratitude to **Mr. Sharad Gupta** , Assistant professor of department CEA ,our project guide for providing invaluable encouragement, guidance and assistance. We would like to thank the group member for the operation extended to us throughout the project. After doing this project we can confidently say that this experience has not only enriched us with technical knowledge but also has unparsed the maturity of thought and vision. The attributes required being a successful professional.

Sakshi Gupta(171500283)

Gouri Shrivastava(171500113)

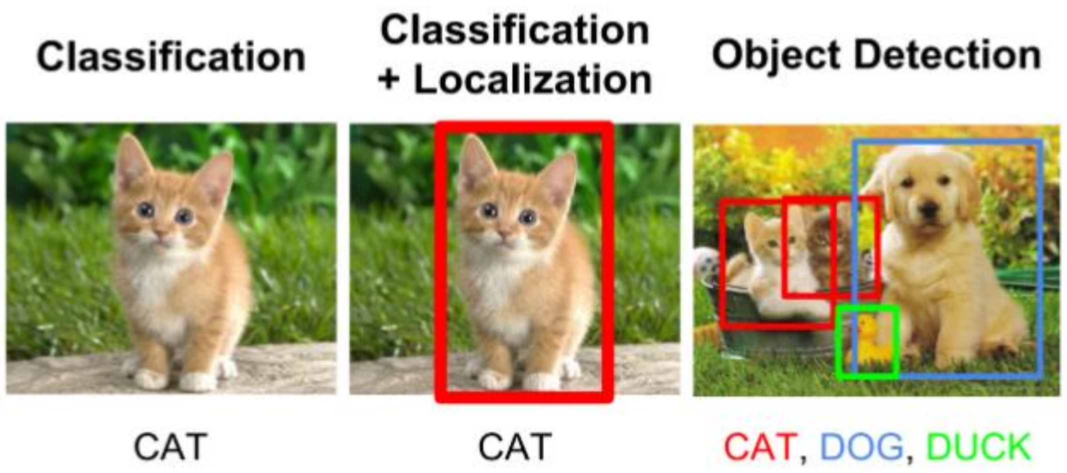
Surbhi Sharma(171500348)

Shivam Singh(171500323)

1. INTRODUCTION

Object Detection is a software developed for real-time tracking object in videos and images. In computer vision were saturating on their accuracy before a decade. However, with the rise of Neural Network learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the image.

A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization. In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box.



1.1 PROBLEM AND MOTIVATION

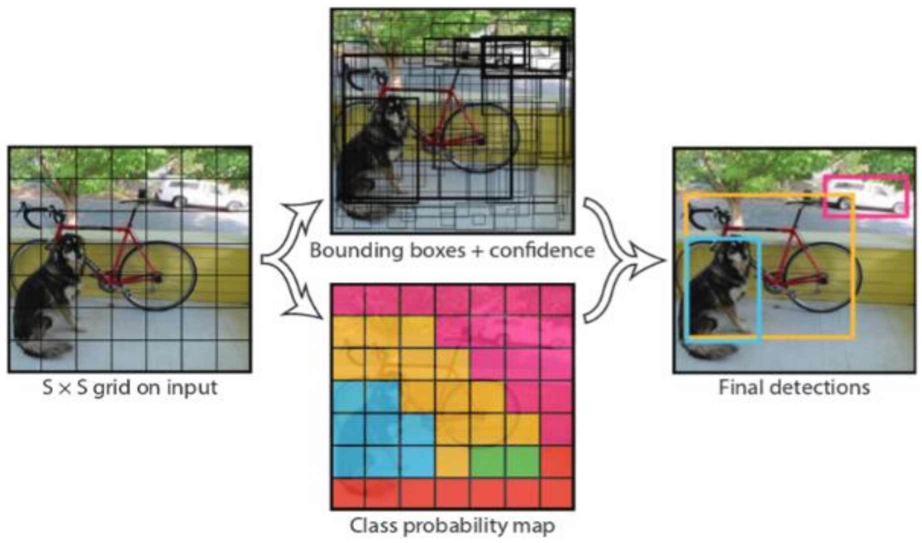
- Object detection is a technologically challenging and practically useful problem in the field of computer vision. Object detection deals with identifying the presence of various individual objects in an image.

- object detection is face detection, that is used in almost all the mobile cameras. A more generalized (multi-class) application can be used in autonomous driving where a variety of objects need to be detected
- Also it has a important role to play in surveillance systems. These systems can be integrated with other tasks such as pose estimation where the first stage in the pipeline is to detect the object, and then the second stage will be to estimate pose in the detected region.
- The motive of object detection is to recognize and locate all known objects in a scene. Preferably in 3D space, recovering pose of objects in 3D is very important for robotic control systems. The information from the object detector can be used for obstacle avoidance and other interactions with the environment.

1.2 OVERVIEW

Our Object Detection basically has one main module You Only Look Once(YOLO) it base on Convolutional Neural Network. YOLO is an important framework that has led to improved speed and accuracy for object-detection tasks.

- First we divide the image into a grid of cells. Say it is 7x7. Now ground truth objects which have their object centers in their respective grid cells turns out to be +ve cells.



Yolo grid cell

- Each cell predicts 2 bounding boxes and each bounding box consists of 5 predictions :x,y,w,h, and confidence. The (x,y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image.
- Finally the confidence prediction represents the IOU between the predicted box and any ground truth box. So for VOC with 20 classes the output would be ((4+1)x2+20) = 30. If the grid size is 7x7, we will have 7x7x30 outputs.

1.3 OBJECTIVE

- The project objective was to produce a working system for tracking objects in 3 dimensional space.
- The objective of the project was to begin from this spec, and design a solution to the problem. After a satisfactory solution was designed, the task came to implement the solution
- Also it has a important role to play in surveillance systems
- Objects detection is very important for robotic control systems.

Throughout the project, many problems arose. So objective of this project was creating Object detection software Results of this project have been applied to a range of products such as video surveillance, robotic vision and autonomous flight. In robotics, tracking is frequently used to provide a means for localization and mapping of an unknown environment .

1.4 CHALLENGES

The major challenge in this problem is that of the variable dimension of the output which is caused due to the variable number of objects that can be present in any given input image. Any general machine learning task requires a fixed dimension of input and output for the model to be trained.

Another important obstacle for widespread adoption of object detection systems is the requirement of real-time (30fps) while being accurate in detection. The more complex the model is, the more time it requires for inference; and the less complex the model is, the less is the accuracy. This trade-off between accuracy and performance needs to be chosen as per the application. The problem involves classification as well as regression .

1.5 REQUIREMENT ANALYSIS

1.5.1 Hardware Requirement

1) Computer system with minimum 4GB RAM
2) 4 GB of available disk space minimum
3) Scanner :
    I. 2D scanner
    II. 3D scanner
4) Camera
5) 1280x800 minimum screen resolution

1.5.2 Software Requirement

1) 1)Window Jupyter Notebook
2) Tensorflow
3) Tensorboard
4) pip installment

i) Operating System: Windows 10

TOOLS AND TECHNOLOGY :

- **PYTHON :** Python has features like high-level built-in data structure , dynamic typing , and binding, which makes it attractive for rapid application development. It is an open-source software and easy for debugging .
- **USES OF PYTHON :**
  1) Python can be used to develop different applications like web applications, graphic user interface based application, software development applications, scientific and numeric applications, network programming, Games and 3D applications and other business application. It makes an interactive interface and easy developmentof applications.
  2) There are other applications for which python is used that are Robotics, web scraping, scripting, artificial intelligence , data analysis , machine learning , face detection ,video-based apllications and applications for images etc.

LIBRARIES USED :

- **NUMPY :** NumPy stands for 'Numerical Python' or 'Numeric Python'. It is an open source module of Python which provides fast mathematical computation on arrays and matrices. Since, arrays and matrices are an essential part of the Machine Learning ecosystem. NumPy provides the essential multi-dimensional array-oriented computing functionalities designed for high-level mathematical functions and scientific computation. Numpy can be imported into the notebook using.

  ```
  >>> import numpy as np
  ```

- **PANDAS :** Similar to NumPy, Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Dataframe. It is like a spreadsheet with column names and row labels.

  Hence, with 2d tables, pandas is capable of providing many additional functionalities like creating pivot tables , computing columns based on other columns and plotting graphs . Pandas can be important into Python using:

```
>>> import pandas as pd
```

- **MATPLOTLIB :** Matplotlib is an amazing visualization library in Python for 2D and 3D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

  Matplotlib allows you to create reproducible figures programmatically.

  Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

```
>>> from matplotlib import pyplot as plt

>>> import matplotlib.pyplot as plt
```

- **SCIPY :** scipy is a python library that is useful in solving many mathematical equations and algorithms. It is designed on the top of Numpy library that gives more extension of finding scientific mathematical formulae like Matrix Rank, Inverse, polynomial equations, LU Decomposition, etc. Using its high level functions will significantly reduce the complexity of the code and helps in better analyzing the data. SciPy is an interactive Python session used as a data-processing library that is made to compete with its rivalries such as MATLAB, Octave, R-Lab,etc. It has many user-friendly, efficient and easy-to-use functions that helps to solve problems like numerical integration, interpolation, optimization, linear algebra and statistics.

```
<<< from scipy import sparse
```

- **TENSORFLOW : TensorFlow** is an open-source software library. **TensorFlow** was originally developed by researchers and engineers working on the Google Brain Team within Google’s Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well!

```
<<<import tensorflow as tf
```

- **OPENCV :** OpenCV (Open Source Computer Vision Library) is an open source

computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
It supports a couple of programming languages namely: python, java, c and C++. On the other hand, it supports Windows, Linux, Mac Os and even the Android operating systems.

- **SCIKIT – IMAGE :** scikit-image (formerly scikits.image) is an open-source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more. It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
  <<< from skimage.transform impoer resize

- **KERAS :** Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML.[1][2] Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. \

ALGORITHMS :

- **NEURAL NETWORK :** Neural networks are one of the learning algorithms used within machine learning. They consist of different layers for analyzing and learning data. Every hidden layer tries to detect patterns on the picture. When a pattern is detected the next hidden layer is activated and so on. The first layer detects edges. Then the following layers combine other edges found in the data, ultimately a specified layer attempts to detect a wheel pattern or a window pattern. Depending on the amount of layers, it will be or not be able to define what is on the picture, in this case a car.The more layers in a neural network, the more is learned and the more accurate the pattern detection is.

- **BOUNDING BOX :** In object detection, we usually use a bounding box to describe the target location. The bounding box is a rectangular box that can be determined by the xx and yy axis coordinates in the upper-left corner and the xx and yy axis coordinates in the lower-right corner of the rectangle. We will define the bounding boxes of the dog and the cat in the image based on the coordinate information in the above image. The origin of the coordinates in the above image is the upper left corner of the image, and to the right and down are the positive directions of the xx axis and the yy axis, respectivelyPA

- **YOLO :** YOLO is an extremely fast real time multi object detection algorithm. YOLO stands for **"You Only Look Once".**
  The algorithm applies a neural network to an entire image. The network divides the image into an S x S grid and comes up with bounding boxes, which are boxes drawn around images and predicted probabilities for each of these regions.
  The method used to come up with these probabilities is logistic regression. The bounding boxes are weighted by the associated probabilities. For class prediction, independent logistic classifiers are used.

DATASET :

COCO:

1) COCO is a large image dataset designed for object detection, segmentation, person keypoints detection, stuff segmentation, and caption generation. This package provides Matlab, Python, and Lua APIs that assists in loading, parsing, and visualizing the annotations in COCO.

2) The Common Objects in Context (COCO) dataset has 200,000 images with more than 500,000 object annotations in 80 categories. It is the most extensive publicly available object detection database. The following image has the list of objects present in the dataset:

3) The average number of objects is 7.2 per image. These are the famous datasets for the object detection challenge. Next, we will learn how to evaluate the algorithms against these datasets.

Implementation

5.1. Preamble:

The goal of this chapter is to inform the reader of the actual implementation of the code. It details the steps taken by the user to know how the optical tracking component works, and the overall integration of all the components that make up the system. Brief analysis of the project's code is made and discussed, along with the technologies and API's used in implementing the project. Unified Modeling Language (UML) diagrams are presented which illustrate the system and the interactions between components. References are made to parts of the code which perform key operations in the system.

5.2. System Overview(setup):

5.2.1. YOLO method use in implementation:

What is YOLO and Why is it Useful?

The YOLO framework (You Only Look Once) deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation.This is one of the best algorithms for object detection and has shown a comparatively similar performance to the R-CNN algorithms.

5.2.2. How does the YOLO Framework Function?

In this section, I have mentioned the steps followed by YOLO for detecting objects in a given image.YOLO first takes an input image:

(1)The YOLO first takes an input image:

(2)The framework then divides the input image into grids (say a 3 X 3 grid):

(3)Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects (if any are found, of course).

5.2.3. How to Encode Bounding Boxes?

The bounding box usually consists of four parameters. Intuitively they could be the center coordinates of the bounding box, width, and heigh of the bounding box.

Bounding Box Encoding:

The above bounding box representations are usually encoded as the final representation of the bounding box.

(1) Centroids Representation Encoding:

The encoded representation of a ground-truth bounding box [xgt, ygt, wgt, hgt] with the corresponding anchor box [xanchor, yanchor, wanchor, hanchor] is [x′, y′, w′, h′], where

x′=(xgt−xanchor)/wanchor

y′=(ygt−yanchor)/hanchor

w′=ln[wgt/wanchor]

h′=ln[hgt/hanchor]

(2) Corners Representation Encoding:

The encoded representation of a ground-truth bounding box [xmin, gt, ymin, gt, xmax, gt, ymax, gt] with the corresponding anchor box [xmin, anchor, ymin, anchor, xmax, anchor, ymax, anchor] is [x′min, y′min, x′max, y′max], where

x′min=(xmin,gt−xmin, anchor)/wanchor

y′min=(ymin,gt−ymin, anchor)/hanchor

x′max=(xmax,gt−xmax, anchor)/wanchor

y′max=(ymax,gt−ymax, anchor)/hanchor

(3) MinMax Representation Encoding:

Similarly, the encoded representation of a ground-truth bounding box [xmin, gt, xmax, gt, ymin, gt, ymax, gt] with the corresponding anchor box [xmin, anchor, xmax, anchor, ymin, anchor, ymax, anchor] is [x′min, x′max, y′min, y′max], where

x′min=(xmin,gt−xmin,anchor)/wanchor

x′max=(xmax,gt−xmax,anchor)/wanchor

y′min=(ymin,gt−ymin,anchor)/hanchor

y′max=(ymax, gt−ymax,anchor)/hanchor

5.2.4. Intersection over Union and Non-Max Suppression:

How can we decide whether the predicted bounding box is giving us a good outcome (or a bad one)? This is where Intersection over Union comes into the picture. It calculates the intersection over union of the actual bounding box and the predicted bonding box.

IoU = Area of the intersection / Area of the union, i.e.

IoU = Area of yellow box / Area of green box

This is what Non-Max Suppression is all about. We are taking the boxes with maximum probability and suppressing the close-by boxes with non-max probabilities. Let’s quickly summarize the points which we’ve seen in this section about the Non-Max suppression algorithm:

(1) Discard all the boxes having probabilities less than or equal to a pre-defined threshold (say, 0.5).
(2) For the remaining boxes:
    - Pick the box with the highest probability and take that as the output prediction.
    - Discard any other box which has IoU greater than the threshold with the output box from the above step.

(3) Repeat step 2 until all the boxes are either taken as the output prediction or discarded.

5.2.5 Combining the Ideas:

In this section, we will first see how a YOLO model is trained and then how the predictions can be made for a new and previously unseen image.