ML/DL Assignment EagleView
Object Detection Using Detectron2

Chetan V. Deshmukh

Nashik, India

chetandeshmukh497@gmail.com

ABSTRACT

Object Detection has always been a vital problem and have always been looked upon with a great importance. Significance of this problem has increased even more with the usecases and applications like tracking objects, detecting vehicles, self-driving cars, counting the crowd and many more that demandprecise detection and localization of various objects. On the other hand, neural networks have taken the world by storm. And no surprises that the area of Computer Vision and the problem of facial expressions recognitions hasn't remained untouched. This is just a first-hand implementation and can be improved if put in more timeand resources.

INTRODUCTION

Object detection is a technique of the AI subset computer vision that is concerned with identifying objects and definingthose by placing into distinct categories such as humans, cars,animals etc. It is a computer vision technique that allows us toidentify and locate objects in an image or video. With this kindof identification and localization, object detection can be usedto count objects in a scene and determine and track their precise locations, all while accurately labeling them. Object detection allows us to at once classify the types of things foundwhile also locating instances of them within the image. It combines machine learning and deep learning to enable machines to identify different objects.

However, image recognition and object detection these terms are often used interchangeably but, both techniques aredifferent. Object detection could detect multiple objects in animage or, in a video. Image recognition assigns a label to an image. A picture of a dog receives the label "dog". A pictureof two dogs, still receives the label "dog". Object detection, on the other hand, draws a box around each dog and labels thebox "dog". The model predicts where each object is and whatlabel should be applied. In that way, object detection providesmore information about an image than recognition. The demand for trained experts in this field is pretty high and having a background in deep learning for computer vision with python can prove extremely vital in today's world.

Why is Object Detection Important?

Object detection is inextricably linked to other similar computer vision techniques like image recognition and image segmentation, in that it helps us understand and analyse scenes in images or video. But there are important differences. Image recognition only outputs a class label for an identified object, and image segmentation creates a pixel-level understanding of a scene's elements. What separates object detection from these other tasks is its unique ability tolocate objects within an image or video. This then allows us to count and then track those objects. Given these key distinctions and object detection's unique capabilities, we can see how it can be applied in a number of ways:

- Crowd counting
- Self-driving cars
- Video surveillance
- Face detection

Of course, this isn't an exhaustive list, but it includes someof the primary ways in which object detection is shaping ourfuture.

COCO DATASET

The MS COCO dataset is a large-scale object detection, segmentation, and captioning dataset published by Microsoft. Machine Learning and Computer Vision engineers popularlyuse the COCO dataset for various computer vision projects.

Understanding visual scenes is a primary goal of computervision; it involves recognizing what objects are present, localizing the objects in 2D and 3D, determining the object'sattributes, and characterizing the relationship between objects. Therefore, algorithms for object detection and objectclassification can be trained using the dataset.

COCO stands for Common Objects in Context, as the image dataset was created with the goal of advancing image recognition. The COCO dataset contains challenging, high- quality visual datasets for computer vision, mostly state-of- the-art neural networks. For example, COCO is often used tobenchmark algorithms to compare the performance of real- time object detection. The format of the COCO dataset is automatically interpreted by advanced neural network libraries. It contains over

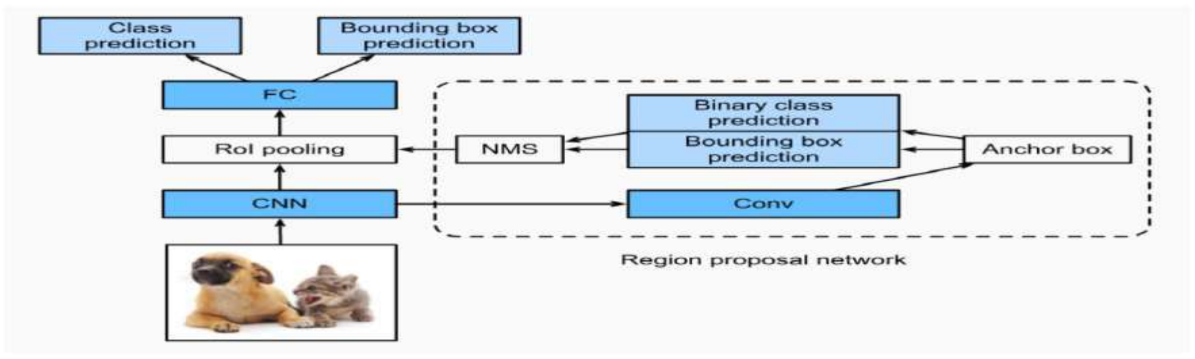
200’000 images of the total 330’000 images are labeled with 80 object categories which include “things” for which individual instances may be easily labeled(person, car, chair, etc.). However, here we will be working with just two categories, Persons and Cars.

APPROACH

A number of popular object detection models belong to theR-CNN family. Short for region convolutional neural network. Over the years, they’ve become both more accurate and more computationally efficient. There are also a number of models that belong to the R-CNN family i.e. RCNN, Fast RCNN and Faster RCNN.

Here I have used Faster RCNN. In general, object detectors have three main components: a backbone that extracts features from the given image; a feature network thattakes multiple levels of features from the backbone as input and outputs a list of fused features that represent salient characteristic of the image; and the final class/box network that uses the fused features to predict the class and location ofeach object.

→ Understanding Faster R-CNN



Faster R-CNN – Object Detection Algorithm | Source

The Faster R-CNN model is one of the best versions of the R-CNN family and improves the speed of performance tremendously from its predecessors. While the R-CNN and Fast R-CNN model make use of a selective search algorithm to compute the region proposals, the Faster R-CNN method replaces this existing method with a superior region proposal network. The region proposal network (RPN) computes images from a wide range and different scales to produce effective outputs.

The regional proposal network reduces the margin computation time, usually 10 ms per image. This network consists of the convolutional layer from which we can obtain the essential feature maps of each pixel. For each feature map, we have multiple anchor boxes which have varying scales, different sizes, and aspect ratios. For each anchor box, we make a prediction of the particular binary class and generate a bounding box for the same.

The following information is then passed through the non-maximum suppression to remove any unnecessary data since many overlaps are produced while creating the feature maps. The output from the non-maximum suppression is passed through the region of interest, and the rest of the process and computation is similar to the working of Fast R-CNN.

→ Points to consider

1. **Limitations** – One of the main limitations of the Faster R-CNN method is the amount of time delay in the proposition of different objects. Sometimes, the speed depends on the type of system being used.
2. **When to Use Faster R-CNN?** – The time for prediction is faster compared to other CNN methods. While R-CNN usually takes around 40-50 seconds for the prediction of objects in an image, the Fast R-CNN takes around 2 seconds, but the Faster R-CNN returns the optimal result in just about 0.2 seconds.
3. **Example use cases** – The examples of use cases for Faster R-CNN are similar to the ones described in the R-CNN methodology. However, with Faster R-CNN, we can perform these tasks optimally and achieve results more effectively.

Faster R-CNN models available in Detectron2, with different backbones and learning schedules.

What is Detectron2?

Facebook AI Research (FAIR) came up with this advanced library, which gave amazing results on object detection and segmentation problems known as Detectron2. Its implementation is in PyTorch. With a new, more modular design, Detectron2 is flexible and extensible, and able to provide fast training on single or multiple GPU servers. Detectron2 includes high-quality implementations of state-of-the-art object detection algorithms.

The Detectron2 library also allows the users to train custom models and datasets with ease. The installation procedure for the following is quite simple. The only dependencies that you require for the following is PyTorch and the COCO API. Once you have the following requirements, you can proceed to install the Detectron2 model and train a multitude of models with ease.

ASSUMPTION

As the training data consists of only two classes, namely,Persons and Cars, I have assumed that the model in being built only for detection of these two object categories and thus, the model won’t be able to detect other object categories. I have also assumed that we do not need real-timeobject detection and localization, thus, I haven't taken any such extra care for making it work into real-time. As the givendata consists of training images, I have assumed that we cannot use any extra training data, thus, haven't used any other data or haven't increased the size &/or variety of datasetby performing image augmentation.

1) Environment and Hardware:

I have used Google Colaboratory for this task and have used Nvidia Tesla T4 Single GPU system.

2) Overview of steps followed:

- Creating Initial Setup:

I uploaded the trainval.zip file through connecting the Google drive to Colab which contain both the images and their annotations. The given coco dataset contains 2239 images and 16772 annotations in those images with 2 classes, Person and Car.

I have installed Pytorch version = = 1.9 which supports current version of Detectron2, pyyaml version = = 5.1 for creating configuration file, and Detectron with matches with Pytorch version.

- Loading Given Dataset:

In this step I unzipped the trainval.zip folder which contain both Images and Annotation which I registered with COCO instances with the name ObjectDetection. I have done visualization to verify whether our dataset is registered properly or not.

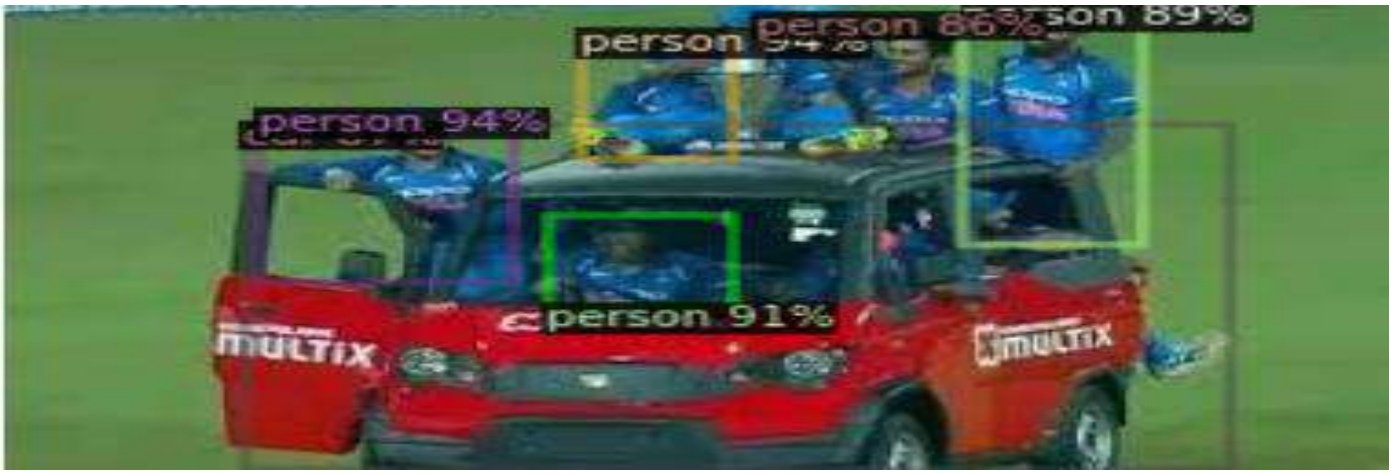
Which shows that our our dataset registered correctly.

- Training:

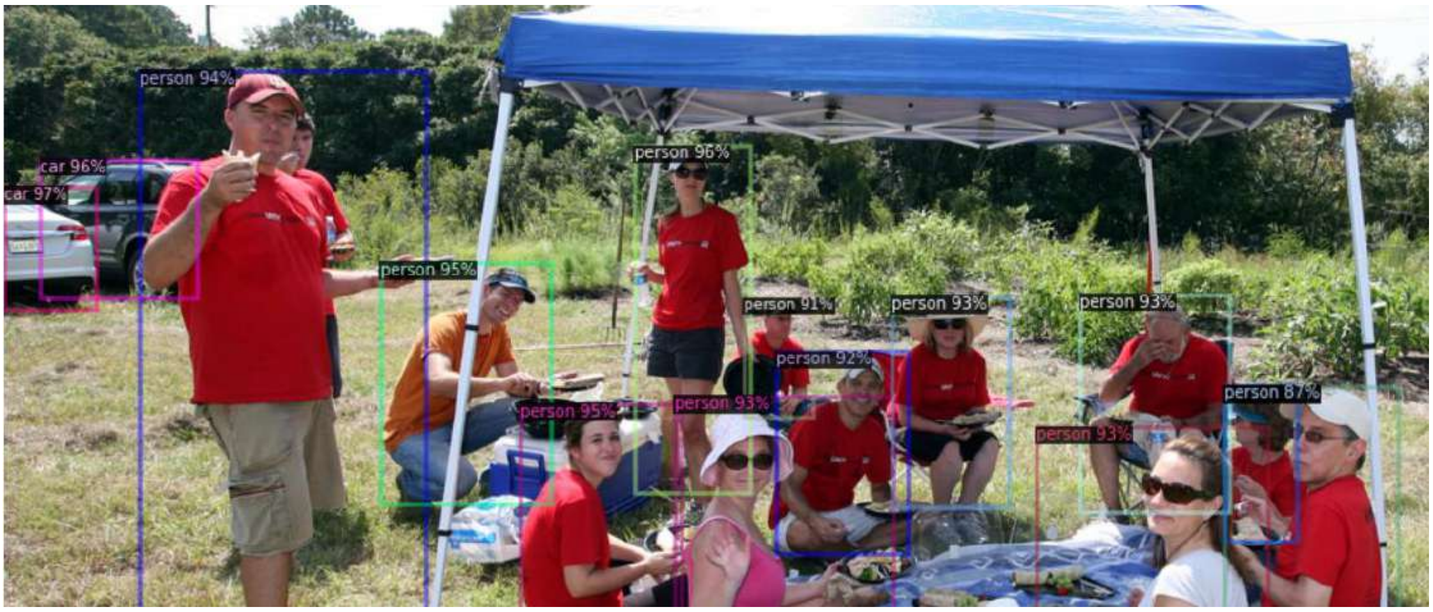
In the Training part, to train our model for Object Detection I have use Faster RCNN. I access this model from code using detectron2.model_zoo APIs. Using IMAGES_PER_BATCH = 2, LEARNING RATE = 0.00025, MAXIMUM ITERATION = 1000, I trained the model to detect Person and car for a given dataset.

- Testing:

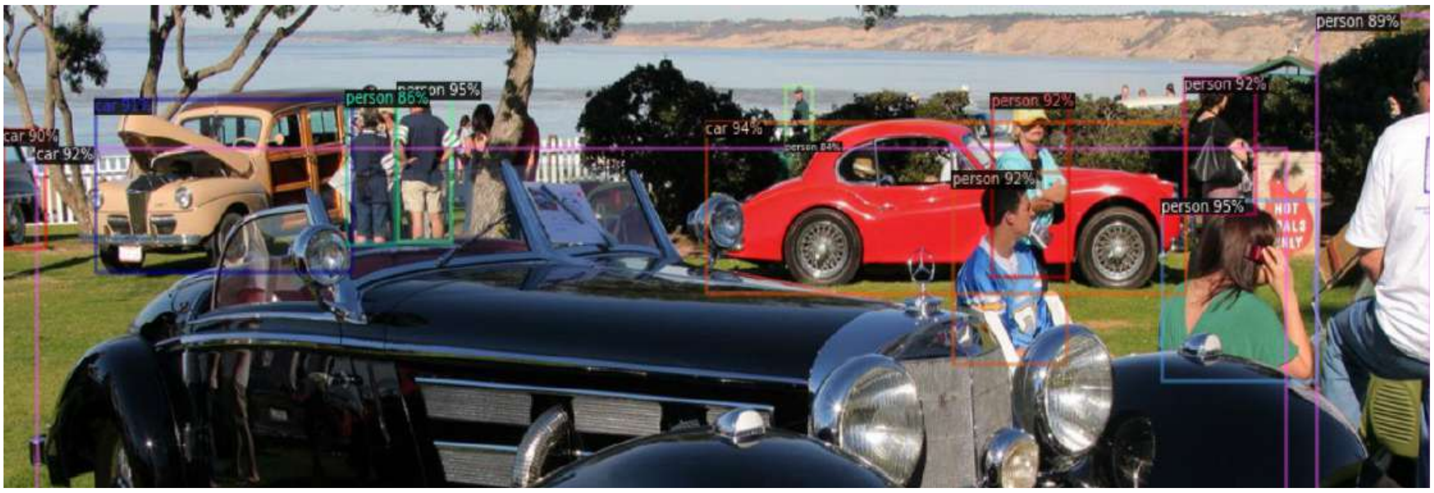
Our task here is to develop and train a model on this dataset to detect person and cars in an image. Sample result images are shown below.



Sample Result Image 1



Sample Result Image 2



Sample Result Image 3

CONCLUSION

The problem of object detection is an extremely importantone and can be solved in various ways by adopting various deep learning techniques. Here, I have used Faster RCNN and Detectron2 for performing this task and have achieved decent first-hand results. These results can be easily interpreted by looking at the result images. These results can be further improved by tuning the model hyperparameters or even by choosing some other technique.

FUTURE SCOPE

This is just a first-hand implementation of the task and following things can be tried to generate better results:

A. Deployment:

The built system can be deployed to be used either as a web application or can be embedded in some target device. Necessary optimizations need to be performed in the code andthe way be implement it as per the target device of deployment and the computational resources available. Easiest way to deploy for having a first-and demo will be making use of Streamlit and GitHub for building a web application.

B. Trying other techniques & more aggressive hyperparameter tuning:

I have used the Faster RCNN and Detectron2, but we can always give a shot by trying out some other state-of-the-art techniques like the YOLO. Furthermore, more time and resources can be put in tuning the model hyperparameters in order to get the best results.

C. Image Augmentation:

I haven't applied any augmentation techniques assuming that only the give data has to be used to training without any further modifications or additions. Performing augmentation will surely generate better results. Some operations that mightprove significant in getting better results are zoom-in and zoom-out, increase and decrease in brightness or illumination, horizontal flipping and sheer operations, slight degree of rotation, etc.

REFERENCES

Following resources have certainly helped me in getting a first-hand solution for this task:

[1] COCO Dataset: https://cocodataset.org/#home

[2] Your Guide to Object Detection with Detectron2 in PyTorch: https://tinyurl.com/39p9b3d5

[3] Facebookreasearch Detectron2 Model Zoo and Baselines: https://github.com/facebookresearch/detectron2

[4] Understanding Region-Based Convolutional Neural Networks: https://www.analyticsvidhya.com/blog/2021/08/your-guide-to-object-detection-with-detectron2-in-pytorch/

[5] Object detection: https://www.fritz.ai/object-detection