Introduction

In the world of computer vision, YOLO (You Only Look Once) stands out as a powerful and efficient object detection algorithm. My exploration began with an introduction to YOLO and its various versions, which eventually led me to create a comprehensive dataset and discover the best tools for annotation.

Initial Steps

Discovering YOLO

My journey began with reading about YOLO and its different versions. Understanding the strengths and limitations of each version was crucial. I quickly realised that YOLOv5(based on this article), developed by Ultralytics, was the most suitable for my needs. It strikes a balance between performance and resource consumption, unlike YOLOv3, which relies on the now deprecated TensorFlow 1.x.

Creating a Dataset

With a basic understanding of YOLO, I proceeded to download random images to create an initial dataset. This step was essential to gain practical experience and understand the nuances of preparing data for training an object detection model.

Research Phase

Annotation Tools

To make the dataset useful, annotation was necessary. I delved into the research of various annotation tools. Among the notable ones were VOTT (Visual Object Tagging Tool), CVAT (Computer Vision Annotation Tool) and MakeSense[dot]ai. Each tool had its unique features and user interface, making it important to choose the right one for the job.

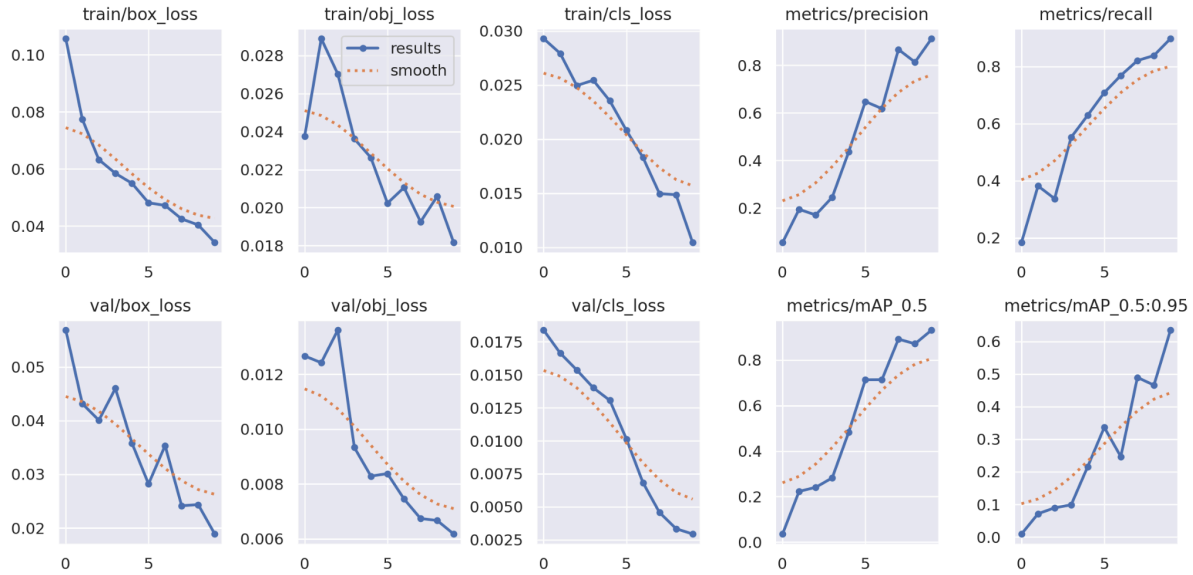
Discovering Annotated Datasets

While researching annotation tools, I stumbled upon an already annotated dataset featuring Pepsi and Coke logos. This discovery was a game-changer as it saved significant time and effort that would have otherwise been spent on manually annotating images.

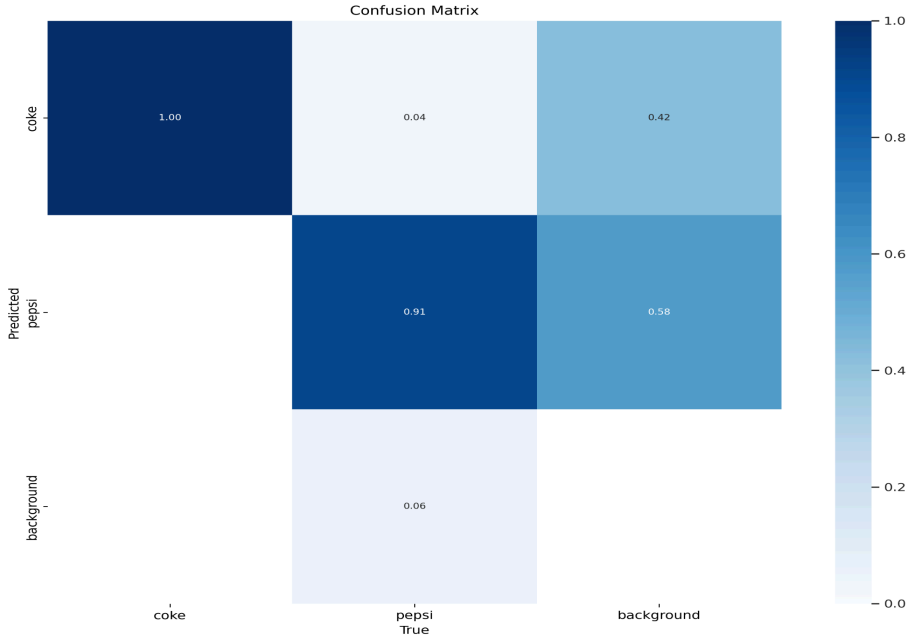
Final Approach

Deciding on YOLOv5

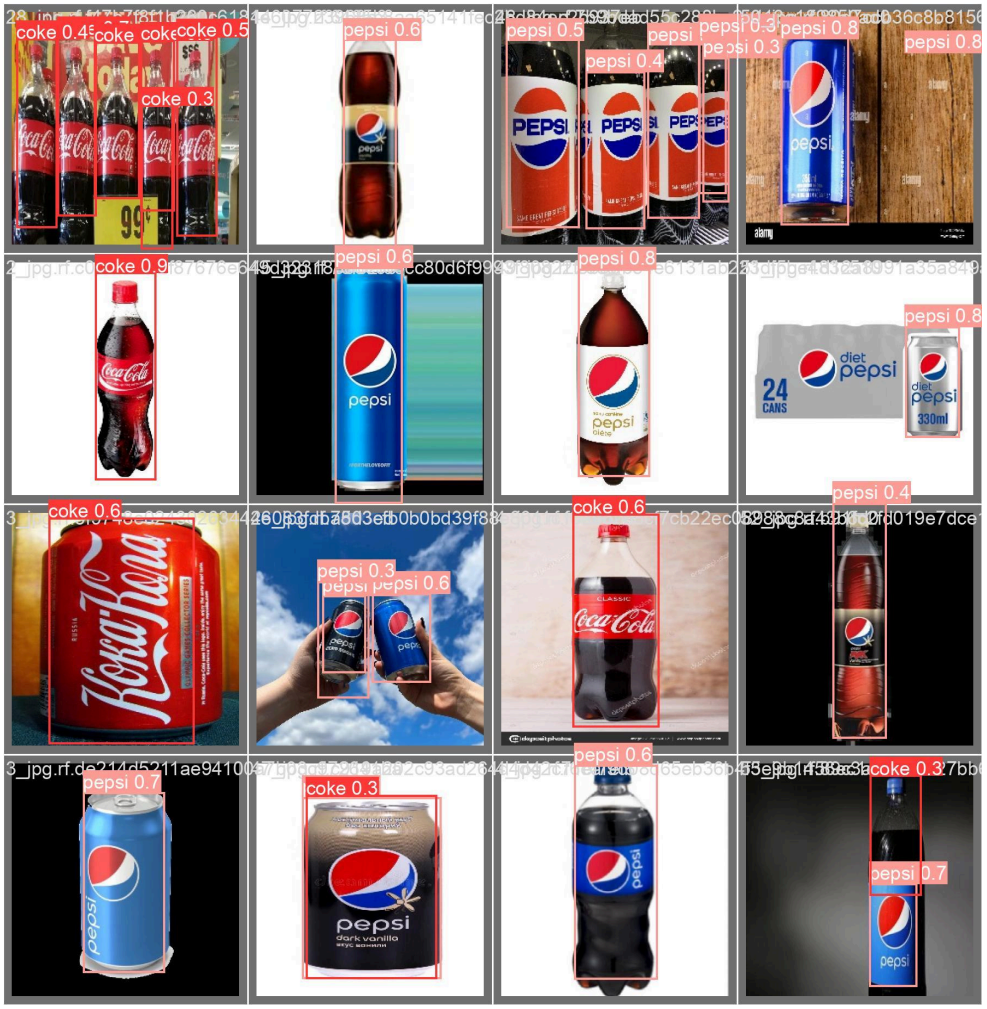
After thorough research, I decided to go with YOLOv5. I trained YOLO as suggested on this blog with 10 epochs and batch size of 16. Below is how training turned out to be:



Below is how results turned out from our test set:



Below are how the predictions look like:



Processing the video using the trained model

Wrote a script that does below steps:
1. Accept the location of the video location
2. Extract the frames out of the video
3. Resize each frame to 416 x 416
4. Save each frame as [timestamp].jpeg in the provided output directory

At the end of the above process, I had a directory containing ~21k images which was hard to process on Colab (on the free plan). Therefore, I broke that directory into multiple subdirectories containing 1k images each.

For each directory, ran the trained YOLOv5 and generated the predictions (and saved it into results.txt). Then, I processed the text file to generate the required JSON file.

(More about this can be found in the repo)