Documentation

This project includes functions to perform the task of detecting a car in a specific parking lot marked in the assignment description. The following are explanations about the structure of the project and its usage. Hamid Shojanazeri has developed this code, and I have used some code for running the yolov3 model from online sources developed by Adrian Rosebrock.

Structure

This project includes a main module with the name of cars.py. This module provides three different functions: detect, compare and analyze.

This project also includes a docker file along with the requirement.txt used for running through docker.

It has a separate folder (yolo-coco) that keeps the yolov3 config file along with the coco dataset labels. The cars.py downloads the Yolov3 weights and save it in this folder as well.

There is a cache folder that will be produced during the runtime, which includes the verkada sample video index file and the first frame of each of the sample videos, which passed to the functions for processing.

Usage

Three functions provided in the cars.py can be used as follows:

• python3 cars.py detect 1538076003

This function accepts a timestamp and run the yolov3 model to check if a car is parked in the defined parking spot.

• python3 cars.py Compare 1538076003 1538076083

This function accepts two timestamps and compare if two images are the same in the defined spot. If they are not the same then it will call detect function to check if the parking spot is occupied with a new car or it is free.

• python3 cars.py analyze 1538076003 1538076083

This function accepts two timestamps and checks the defined parking spot between the two timestamps. Whenever it detects a new car parked in the spot, it returns the time that the car has been parked there until it left. It will continue calculating each new car parking time until it reaches to the second timestamp passed to the function.

The output shows that at which frame (timestamp) a car has been detected and that car has been parked until what timestamp along with the time between them in minutes. It also prints a message if parking is not occupied.

All the above functions can save the output in a json file if the –o or –output along with a file name be passed to the function.

- python3 cars.py analyze 1538076003 1538076083 –o results.json

Engineering Practices

- The idea behind the **compare function** is to check if the portion of the two images that cover the defined parking spot are the same or not. In this case we do not run the yolov3 model for detection initially. We run the model only if the images are not the same to determine the parking has changed to which of the following states: a new car has parked there or the parking is free. The compare function will be called during the analyze function process. In analyze, first the model will be run to detect if the parking for the first passed timestamp is occupied or free, then for the consequent timestamps we just compare the images to determine if they are the same or not until compare function return false that means the parking state has changed. This saves us a lot of computation by not running the yolov3 model each time during the compare function.

- **analyze function** : as per advised for the sampling of the video each 4 seconds and considering that each video in the sample data are almost 4 seconds (so mostly timestamps in the dataset are 4 seconds different: 1538076003.ts - 1538076007.ts) , I decided to loop over the range between the first and second timestamps passed to the analyze function with step of 4, instead of accessing the index file. In this case, I did not have to save the index file in data structure and loop over that to access its elements. I have written this code as well. However, I found some of the data do not follow that 4 seconds routine. For example, data from 1538077459.ts jump 3 seconds to 1538077462.ts and 5 seconds to 1538077467.ts. or from 1538076751.ts to almost 200 seconds later at 1538076916.ts. So, I had to save index file and access its elements to loop over the passed timestamps to the analyze function.

- Another concern that we spot is when a car that is leaving the parking pauses in the mid way for few sampling rates, then it would be detected as a new car. We handle this concern by ignoring the cars with parking time of less than one minute.

Separation of logic from output

Defining different functions to do different tasks implements the logic and minimizes their interference with output. So, to get the desired output we will use different functions with minimum overlap.

Human friendly and readable code

I tried to use meaningful names for functions and variables in the code to make it more readable along with considering documentations where it was needed. Also, tried to output meaningful phrases and sentences to make the app more human and user friendly to guide the user through the process.

when do you think this algorithm would work well? when would it not?

One of the concerns is about the used bounding box.

To identify the marked parking spot, I defined a box that bounds this spot. I found the bounding box through running yolov3 model when this spot was occupied. The way that I defined this box, it tolerates some variation in size. I noticed that yolov3 model sometimes output slightly different box size when detects the same object in the same spot. I tried to define a safe margin for this variation in box size according to a reasonable amount of sample data from yolov3 model runs. If by any chance this variation exceeds the defined margin, this may effect the detection in that frame.

Along with the slight variation in box sizes, I tried to avoid some extra operations such as resizing and matching the boxes; instead I reduced the threshold for the similarity between two images. The logic behind this, goes back to the point that we assumed compare function is defined to be used mostly inside the analyze function to compare images within the range of two timestamps. For example if we detect a parked car, the similarity between two frames is usually higher than defined threshold and it does not effect our detection. And if the state of the parking changes from occupied to free (or vice versa) the difference is higher than threshold so again it works for us in this scenario. It just helps us to save some more computations. The only concern is that if a similar car parks instead of a previous parked car in less than 4 seconds (our sampling rate) that logically rarely happens.

what would you suggest for future exploration?

I suggest exploring other deep learning object detection models for comparison. I explored Mask RCNN for this assignment, too, that it has been said have a good accuracy. My results show that the run time using google colab with GPU was comparable with yolov3.

As mentioned in **Engineering Practices**, if we could keep the data within a specific routine like save every 4 seconds then we do not have to save the index file in a data structure for processing.