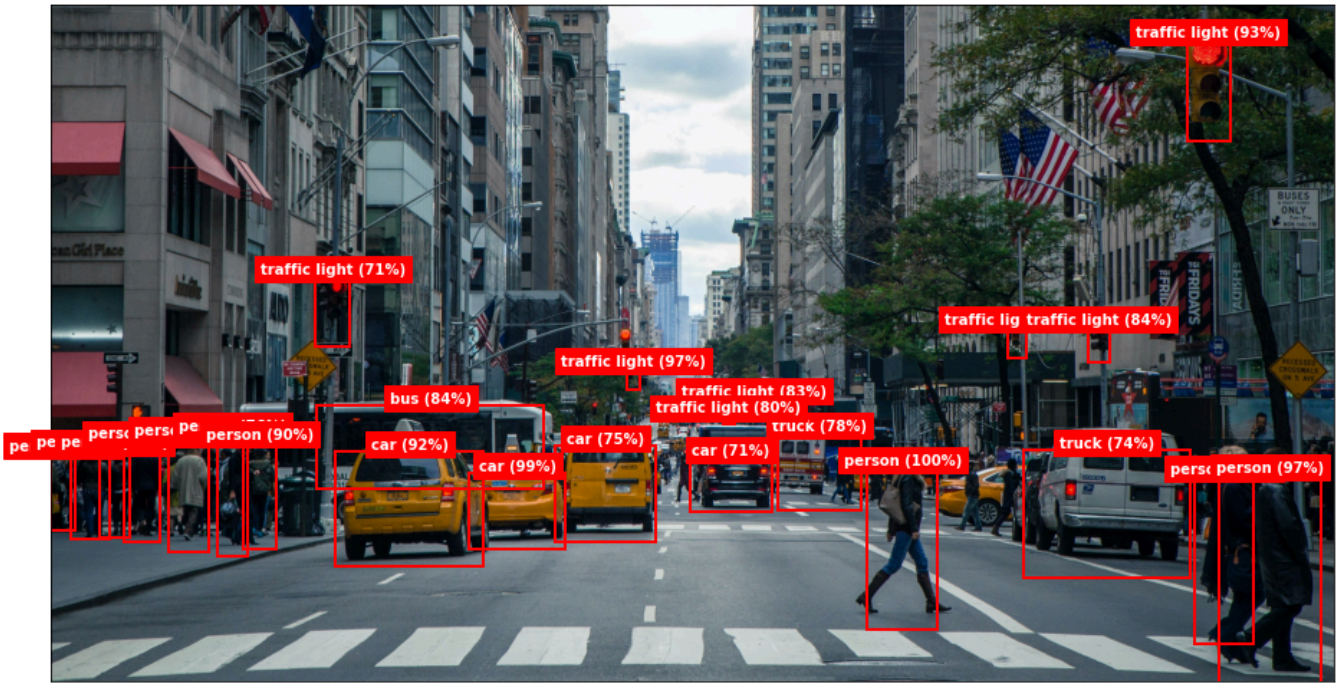
Production Challenge

Computer vision engineer: Algotive



Pablo Quihui

22/01/2024

INTRODUCTION

In the landscape of machine learning, the significance of object detection cannot be overstated. This technology, enabling machines to identify and locate objects within visual data, serves as a linchpin across various industries. In autonomous vehicles, it enhances safety by granting the ability to recognize and respond to the surroundings. In healthcare, it facilitates precise diagnoses through the analysis of medical images, while in security systems, it fortifies surveillance with intelligent identification and tracking capabilities. Object detection also streamlines processes in retail and inventory management, reducing errors in tracking. The efficiency and precision brought about by advanced algorithms and neural networks surpass human capabilities in processing visual data. Looking ahead, object detection is poised to redefine the technological landscape, influencing smart cities, augmented reality, and beyond, ushering in a future where machines seamlessly interpret the visual complexities of our world, unlocking new horizons for intelligent applications.

OBJECT DETECTION MODELS

In the dynamic field of object detection models, several architectures have garnered attention for their efficiency and accuracy. In this task, my focus has centered on exploring the capabilities of the YOLO (You Only Look Once) family of models. Among them, YOLOv5 and YOLOv8 are great contenders, each with distinct features contributing to their efficacy in real-world applications.

YOLOv5:

The YOLOv5 model is a very popular model available in different frameworks. It is characterized by its lightweight design and remarkable speed, and has gained prominence for its ability to maintain high accuracy while ensuring swift inference.

YOLOv8:

The last update of the YOLO family is YOLOv8, which introduces advancements in architecture, refining the model's accuracy and robustness. With an emphasis on handling complex scenes and diverse objects, YOLOv8 showcases improvements in detection precision and adaptability.

COMPARATIVE ANALYSIS

To determine the most suitable model for deployment, a meticulous comparative analysis between YOLOv5 and YOLOv8 has been undertaken. Although there was no ground truth in this exercise, pre-trained weights with COCO dataset was used in both. However, there was not possible to to measure the performance in terms of accuracy, iou, etc. Thus, I used two important metrics on the field of object detection and computer vision with videos: Frame Per Second and Time Inference. The goal is to identify the model that achieves a detection of twelve videos from a highway in the fastest way.

Therefore, after the inference of all the videos by both models, I calculated the mean FPS and Time Inference for the twelve videos. The results were the following:

Table 1. Inference average time per frame for test videos (The lower the better)

MODEL	TIME INFERENCE PER FRAME (SECONDS)
YOLOv5	0.0121
YOLOv8	0.0115

Table 2. FPS for test videos using (The higher the better)

MODEL	FPS
YOLOv5	87.103
YOLOv8	88.989

Those results were obtained using Google Colab and a GPU NVIDIA T4.

However, due to resources limitation, further experiments were realized in a Macbook M1 without GPU.

As seen in Table 1 and 2, YOLOv8 was the fastest during the inference of the test set, and that is the reason it was selected for further optimization.

PRODUCTION

Although the model selected, YOLOv8 was already trained, the limitation of GPU increased the time inference of each video to around **0.51 seconds per frame**, and some videos with more than 1000 frames, it was a lot of time. As for FPS it was reduced to only **1.97**. Therefore, it is clear that the use of the appropriate computational resources can lead to better and faster performance.

However, there exist a lot of new methods to optimize the performance of ML models in CPU, some of them with costs and others with a free tier. For this case, I managed to optimize the model by exporting it as an ONNX model and using the runtime for inference of the videos. Although the parallelization is limited for the absence of GPU, ONNX managed to optimize the time and resources, getting phenomenal results shown in Table 3.

Table 3. Inference average time per frame for test videos (The lower the better)

METRICS	SCORE
Time Inference per frame	0.085 seconds
FPS	12.41

Although it is not the same results as obtained with GColab, it demonstrates a **decrease in inference time of around 84%** and an **increase on FPS of 530%**, which is a significant gain in terms of performance using only CPU.

Having that performance, it could be deployed for production in an online environment using other tools such from cloud services without the need of high demand of computational resources.

QUANTITATIVE RESULTS

Some quantitative examples of the model performance are shown in Figures 1 and 2 using videos of the test set provided. It is important to mention that in a visual inspection, there was a decrease of accuracy in the detection of objects by the implementation of ONNX. However, the weights used during this exercise were the lightest, and therefore, the ones with lower performance in terms of metrics in COCO dataset.

Figure 1. Example of the model performance in test videos.

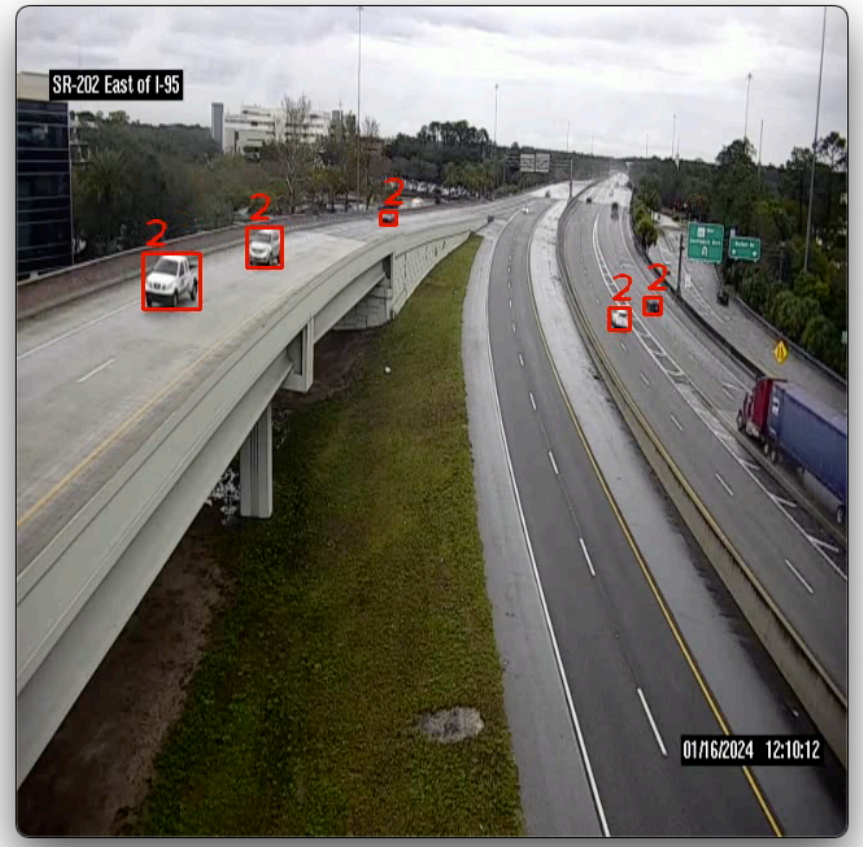


Figure 2. Example of the model performance in test videos with only cars (class #2)

The entire code of the development and implementation can be found in https://github.com/pabloquihui/AlgotiveObjectDetection/