Realtime Detection using Deep Learning on

Embedded Platforms

Introduction

In this project, we will investigate real time vehicle detection, counting and distance estimation from video streams using wearable and embedded cameras. One possible application is the use of vehicle mounted cameras on Ola Cabs for Delhi-NCR pollution sensing. This is a project undertaken by Prof. Rijurekha Sen with a plan to fit pollution sensors with on-board cameras in collaboration with Ola. The collected data will be used for fine grained correlation analysis of air pollution with road traffic congestion levels. Detecting and counting vehicles from the on-board camera feed in real time will greatly reduce the store costs for raw video footage on resource constrained sensing equipment or communication costs for the raw video footage to be transmitted to a remote server.

Objectives

Our main objective is to make and deploy a model capable of detecting and classifying object of interests in a video. The envisioned computer vision software will process an incoming video feed and do vehicle counting and classification (into types like buses and other public transport, cabs, auto-rickshaws, personal cars, two-wheelers, heavy vehicles like trucks) in near real time. This information can be used in getting real-time information about traffic congestions, incidents and roadwork.

Datasets

1. PASCAL VOC: This dataset consists of realistic daily life images taken from Flickr.

Intra-class variation is high and objects are viewed from multiple viewpoints. This

dataset is composed of 20 object categories organized in four major groups.

2. MS-COCO: The COCO dataset is an excellent object detection dataset. It contains 91

common object categories with 82 of them having more than 5,000 labeled instances.

Overall, the dataset has 2,500,000 labeled instances in 328,000 images.

3. Extensive video data on buses plying on various bus routes in Mumbai. This would

mimic a real deployment setting

Approach to the project:

1. We would focus primarily on recent architectures: SSD (Single Shot Multibox Detector), Faster R-CNN, R-FCN (Region-based Fully Convolutional Networks) and YOLO. While they were presented with a particular feature extractor (e.g., VGG, Resnet, etc), we would experiment with various feature extractors.

2. We would compare our results with standard benchmarks such as Imagenet and COCO with respect to accuracy.

3. In order to satisfy the constraints of the embedded platforms we could experiment with several architectural configurations, such as for Faster R-CNN and R-FCN, we can also choose the number of region proposals to be sent to the box classifier at test time. Typically, this number is 300 in both settings, but an easy way to save computation is to send fewer boxes potentially at the risk of reducing recall.

4. We would then be evaluating all the models based on accuracy/ latency/ energy or hardware metrics on Movidius Stick and NVIDIA Jetson TX2 Module.

5. An intuitive way to perform object detection in videos is to generate bounding box predictions for consecutive frames. We might need to drop frames in order to meet hardware constraints and improve latency.

6. Further Possibilities

(a) We will look for techniques of data augmentation and transfer learning to compensate for the limited amount of training data.

(b) We would look into using impression networks which might improve speed and performance simultaneously.

Challenges we might face:

1. Object detection from wearable cameras is challenging due to difference in view points based on the height of the person wearing the camera, his or her physical orientation and physical movements.

2. Processing high definition videos at powerful GPU servers might give the best performance in road traffic monitoring. But the poor broadband infrastructure in developing countries might prohibit real time streaming of HD videos from roads to servers. In-situ processing might mandate using mobile and embedded platforms, but their processor and battery constraints can conflict with heavy computation and low latency requirements. Our work would be to perform all the detection on embedded platforms like Intel Movidius sticks.

3. While accuracy of the inference task is an important metric to maximize, this might have trade-offs with other metrics on resource constrained embedded platforms. Is the latency of each inference too high to suit a real time mobile application while a user is interacting with it, or to suit a road traffic application to detect/prevent accidents? Is the trained deep-net model used in the inference too large to fit the embedded platform RAM? Does the inference task drain the mobile battery too fast? Our goal would be to see how different research communities are innovating to better handle these trade-offs.

ARCHITECTURE DIAGRAM

HOW WOULD YOU MEASURE THE PERFORMANCE

WHAT DEMO CAN YOU SHOW