Machine Learning Engineer Nanodegree

Capstone Proposal

Mahavir Dwivedi

May 15th 2017

Domain Background

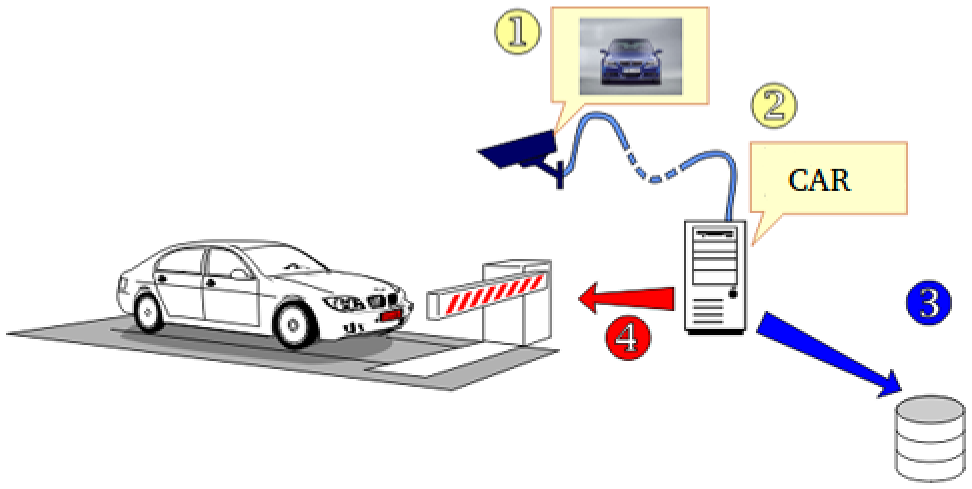
After learning a wide spectrum of ML algorithms and tools through the Udacity Machine Learning NanoDegree, I wanted to get my hands dirty on some problem statement where I can implement the concepts of Machine Learning majorly Deep Learning. As I was deciding on my problem statement for my capstone, I came across a lot of advancements going on in the field of Self-Driving Car. This inspired me to pick up a topic which is related to this theme.

There are a lot of aspects involved in making of a successful Self Driven Car, one of the most important aspects is that the car should be able to differentiate between different vehicles running over the road. A self-driven car should be able to recognize the vehicle around itself. As the recent progress in the field of computer vision majorly object-detection, using these techniques I will try to build a deep learning model which can be used for vehicle detection (specifically **India**).

As I don't have a Self-driving Car environment I will be using our vehicle detection model in another scenario.I will be setting up a smart Vehicle Detection system which will be installed at an entry/exit gate barrier, where it will be used to maintain the database and to monitor the different type of vehicles entering the premises.

Problem Statement

Following is the block diagram for the problem statement:



1. The Internet Protocol Camera at the Entry barrier takes the picture
2. The Picture is send to the CPU.
3. The Picture is being processed at the GPU
4. The decision is made to open the gate by the CPU.

Here I am proposing a deep learning model which can detect vehicles in a video stream.

Following is the list of vehicles which I propose to detect:

1. Car
2. Bus
3. Motor Bike
4. Truck
5. Bicycle
6. Auto Rickshaw

The recognition will be performed on a video stream. I aim to achieve a processing speed where no vehicle is missed.

Datasets and Inputs :

As I am building a system for detecting vehicles, the training will be performed on following publicly available datasets.

1. **VOC Dataset (http://host.robots.ox.ac.uk/pascal/VOC/voc2007/)**

Out of the 20 classes , I will be training the model over following 4 vehicle classes:
Vehicle: Bicycle, Bus, Car, Motorbike

2. **COCO (http://cocodataset.org/#explore)**

In case of COCO dataset, I will be training our model for following 5 classes:

Vehicle: Bicycle,car, motorcycle, bus, truck

3. **Annotated Driving Dataset (By Crowd AI)**

This dataset is made publicly available by udacity.

https://github.com/udacity/self-driving-car/tree/master/annotations

The dataset consists of two vehicle labels: Car, Truck .

4. **Custom Dataset**

As some of Indian vehicles are very constrained to Indian conditions , I will be collecting a sufficient amount of data for major Indian Vehicles which are not a part of the above mentioned dataset. One of such vehicle is Indian Auto-rickshaw.

As I am going for an object detection model, I have chosen publicly available vehicle datasets which comes with annotations.

The annotations are done in Pascal VOC Format.

Solution Statement :

Given the problem statement of vehicle detection in a video stream, I will be going through the various Computer vision models to arrive at the solution.

As the task is to figure out all the vehicles present in a given frame at a given time, I won’t use a classification model, rather I will build our pipeline based on an object detection model.

A classification model gives us the probability of all the classes being present in that picture.And then we output that the class with highest probability in that picture as our target class.

Whereas in an object detection model the model will give us all the objects with their respective classes and their positions respectively.

Benchmark Model :

Not many models are out there which have been just trained for Vehicle Detection for the set of Vehicles which I aim to detect with my model. As I am using VOC Dataset for training, I aim to use the models trained on this Dataset as our benchmark model.

Following are the two best models, which have been trained over VOC dataset:

1. **Faster R-CNN**

Faster R-CNN is a RPN based model.

Faster R-CNN achieves a mAP of 0.7053 on VOC Dataset.

The link to the Faster R-CNN implementation on VOC Dataset:https://github.com/chenyuntc/simple-faster-rcnn-pytorch

2. **YOLO v2**

You only look once (YOLO) is a state-of-the-art, real-time object detection system.

YOLOv2 achieves a mAP of 0.71 on VOC Dataset.

The link to the YOLOv2 implementation on VOC Dataset:
https://github.com/longcw/yolo2-pytorch

Evaluation Metrics :

The evaluation matrix I will be using here is mean average precision or “mAP score”.It has become the accepted way to evaluate object detection competitions, such as for the PASCAL VOC, ImageNet, and COCO challenges.

In object detection, evaluation is non trivial, because there are two distinct tasks to measure:

1. Determining whether an object exists in the image (classification)
2. Determining the location of the object (localization, a regression task).

Furthermore, in a typical data set there will be many classes and their distribution is non-uniform. So a simple accuracy-based metric will introduce biases. It is also important to assess the risk of misclassifications. Thus, there is the need to associate a “confidence score” or **model score** with each bounding box detected and to assess the model at various level of confidence.

In order to address these needs, the Average Precision (AP) was introduced. To understand the AP, it is necessary to understand the precision and recall of a classifier. Briefly, in this context, *precision measures the “false positive rate” or the ratio of true object detections to the total number of objects that the classifier predicted*. If you have a precision score of close to 1.0 then there is a high likelihood that whatever the classifier predicts as a positive detection is in fact a correct prediction. *Recall measures* *the "false negative rate" or the ratio of true object detections to the total number of objects in the data set*. If you have a recall score close to 1.0 then almost all objects that are in your dataset will be positively detected by the model

To calculate the AP, for a specific class (say a "person") the precision-recall curve is computed from the model's detection output, by varying the model score threshold that determines what is counted as a model-predicted positive detection of the class.

The final step to calculating the AP score is to take the average value of the precision across all recall values (see explanation in section 4.2 of the [Pascal Challenge paper] pdf which I outline here). This becomes the single value summarizing the shape of the precision-recall curve. To do this unambiguously, the AP score is defined as the mean precision at the set of 11 equally spaced recall values, *Recall_i* = [0, 0.1, 0.2, …, 1.0]. Thus,

$$AP = \frac{1}{11} \sum_{\text{Recall}_i} \text{Precision}(\text{Recall}_i)$$

The precision at recall *i* is taken to be the maximum precision measured at a recall exceeding *Recall_i*.

Up until now, I have been discussing only the classification task. For the localization component (was the object's **location** correctly predicted?) I must consider the amount of overlap between the part of the image segmented as true by the model vs. that part of the image where the object is actually located.

Localization and Intersection over Union

In order to evaluate the model on the task of object localization, we must first determine how well the model predicted the location of the object. Usually, this is done by drawing a bounding box around the object of interest, but in some cases it is an N-sided polygon or even pixel by pixel segmentation. For all of these cases, the localization task is typically evaluated on the Intersection over Union threshold (IoU). Many good explanations of IoU exist but the basic idea is that *it summarizes how well the ground truth object overlaps the object boundary predicted by the model*.

Putting it all together

Now that I have defined Average Precision (AP) and seen how the IoU threshold affects it, **the mean Average Precision or mAP score is calculated by taking the mean AP over all classes and/or over all IoU thresholds**, depending on the competition. For example:

- PASCAL VOC2007 challenge only 1 IoU threshold was considered: 0.5 so the mAP was averaged over all 20 object classes.
- For the COCO 2017 challenge, the mAP was averaged over all 80 object categories and all 10 IoU thresholds.

Averaging over the 10 IoU thresholds rather than only considering one generous threshold of IoU ≥ 0.5 tends to reward models that are better at precise localization.

Model object detections are determined to be true or false depending upon the IoU threshold. This IoU threshold(s) for each competition vary, but in the COCO challenge, for example, 10 different IoU thresholds are considered, from 0.5 to 0.95 in steps of 0.05. For a specific object (say, ‘person’) this is what the precision-recall curves may look like when calculated at the different IoU thresholds of the COCO challenge:

Project Design :

Data Collection :

The first step in my project will be data collection. I plan to use both IP Camera as well as a Mobile Camera in a variety of different road settings to gain a good mix of Images. I plan to scale down the images (starting at 50% of the original size) to reduce processing time, although if I am successful at training a good model with my approach I may attempt a full-scale run.

Data Preprocessing :

The second step will be to annotate the objects of different class present in the images. I plan to take the images from various angle to make the system perform in real case scenarios.Also I will be collecting data set for various conditions,such as :

1. Sunny, Winter, Rainy
2. Day, Evening,Night

I will be using Labeling software for annotating the objects inside the images.

Data Handling :

Once the Images are labeled I will structure the dataset in a format similar to VOC2007. This means I will have a training set , a validation set and a Test set created out

Model Selection, Model Training, and Model Testing :

I will be using different object detection models to get our task done.Few of them are Faster-RCNN and YOLO.

At this point, after having finished the labelling on a subset of the data, I will test to see whether the model actually appears to be learning anything by plotting various loss matrices. If the model does appear to be minimizing the error , then I will move on to the next step, which is testing the model on completely new data it has never seen before. I will compare both its performance from a visual standpoint and from a speed standpoint. The aim is to get a model which works well on a video stream of an IP Camera.