Machine Learning Engineer Nanodegree

Capstone Proposal

Hasan Tuncer

August 13th, 2017

Proposal

Domain Background

Self-driving cars are finally becoming real. Their impact on people's live and economy will be tremendous [1]. There are so many interesting challenges come with self-driving car technology. One of them is detection of surrounding objects including other vehicles. This information is necessary to make a right decision such as turning without causing any safety problem. My goal is to create a machine learning pipeline for detecting vehicle(s) on a road from a video. The video is captured by a forward looking camera mounted on a vehicle.

There are two main data types used for detecting objects in self-driving car domain:

1) Detecting the objects from images captured by camera. Elon Musk and comma.ai are members of the community believing in this approach.

2) Detecting the objects from point clouds captured using Light Detection and Ranging (LIDAR) technology. Majority of the self-driving car startups bets on LiDar technology such as Waymo.

In this project, I will follow the first approach due to extensive data and benchmark resources.

Detecting an object from an image is one of the main reaseach areas of computer vision. Deformable part models (DPMs) and convolutional neural networks (CNNs) are two widely used distinct approaches. DPM are graphical models (Markov random fields) and use an image scanning technique, such as sliding window approach where the classifier such as SVM is run at evenly spaced locations on the image. CNNs are nonlinear classfiers. CNNs are more popular due to their good performance on object detection [2].

Region-based convolutional neural networks (R-CNN) trains CNNs end-to-end to classify the proposed regions into object categories or background. R-CNN deploys region proposal algorithms (such as EdgeBoxes and Selective Search) to select the region in pre-processing step before running the CNN as classifier. Faster R-CNN uses CNN both for the region proposal and also for the prediction. Google Inception uses Faster R-CNN with ResNet. On the other hand, recently published DenseNet outperforms ResNet. YOLO solves region proposal and associated class probabilities with a single neural network. Single Shot Multibox Detector also accomplishes region proposal generation, feature resampling stage and inference with a single deep neural network. SSD's key feature is the use of multi-scale convolutional bounding box outputs attached to multiple feature

sets for prediction. YOLO and Faster R-CNN use single set of feature set for prediction. SSD processing time is shorter than YOLO and Faster R-CNN.

Problem Statement

Self-driving cars need to identify objects around them such as other vehicles on the road. In this problem, the objects are captured by a forward looking camera mounted on a vehicle. Identification of a vehicle will be important factor in deciding the next action that self-driving car will take such as changing lane.

Datasets and Inputs

To train my model, I will use the labeled data for vehicle and non-vehicle which are retrieved by Udacity from GTI vehicle image database, the KITTI vision benchmark suite. If I find the training data set is not enough then I will use the labeled data here.

I will run my pipeline on the video provided by Udacity.

Solution Statement

Additionally, the solution is quantifiable, measurable, and replicable.

Faster R-CNN will be used for object region detection. DenseNet will be used for classifying if the detected object is a vehicle or not. See Project Design section for details. Both of these models are state-of-the art in object detection and classification. These models will be trained on COCO and ImageNet, large datasets conaining thousands of images for hundreds of object types. Therefore, the results will be repeatable and applicable to various problems. For better results on vehicle detection, I will train my model with additional vehicle/non-vehicle data set.

Benchmark Model

I will use KITTI benchmark suit, that includes performance comparison of models in vehicle detection scenario.The result of Faster R-CNN has already been noted in *

Evaluation Metrics

I will use average precision (AP) metric. AP is the area under the precision/recall curve. Precision reflects out of all the items labeled as positive, how many truly belong to the positive class. Precision is ratio of true positive instances to the sum of true postive and false positives. Recall reflects out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset. It is the ratio of true positive instances to the sum of true positives and false negatives.

Project Design

- Load and explore the data: The vehicle/non-vehicle training data mentioned above will be loaded. I will be careful to use the same number of vehicle and non-vehicle images. I will check if the

type/color of the vehicle and the angle the images are taken almost equally distributed. Of course, it is important not to have mis-labeled image.

- Build the model:
  - Region detection: I will use pre-trained faster r-cnn inception resnetv2 for region detection. Pre-trained model will speed my model training because it may take days to train a model from scratch. The reason of selecting faster r-cnn is its high accuracy result [3].
  - Object classification: I will use pre-trained DenseNet on ImageNet for object classification.
- Configure training options: Training options come with the models as they are initially trained. Depending on the performance of the model on test data, I may tune the model.
- Train the model: Vehicle/Non-vehicle data will be divided into train, test and validation groups. The training data will be used to train Faster R-CNN for region detection and DenseNet for object classification.
- Test the model: The model will be evaluated on the test and validation data created out of vehicle/non-vehicle dataset.
- Apply model on the test video:
  - Input: Create images out of video captured from a car. Do necessary editing on the images.
  - Run the model: Run the trained model on the images. Output should be region of car(s) in the image are in rectangle.
  - Output: Create a video by putting toget the frames output of our model.

I plan to use Keras and Tensorflow.