Machine Learning Engineer Nanodegree

Capstone Proposal

Rico Meinl

April 15, 2018

Contents

1 Domain Background

In this proposal I am trying to touch on the background of the problem I am trying to solve and give some personal motivation as well as perspectives about why it is relevant. This project is classified as a Computer Vision problem and the task is to build an algorithm that can recognize traffic lights.
I worked with the LISA Traffic Light Dataset [1] and for the task I will be referencing on two papers [2][3] that discuss this issue.
Traffic Light Detection is a crucial element of driver assistance systems (DAS) used in autonomous vehicles to safely navigate on public roads. Fully autonomous driving is a scenario that lays in the future but building components for DAS may save plenty of lives on the way.
My personal motivation behind this is to start understanding what components are necessary to build autonomous vehicles and how we can continue to improve them. As the majority of accidents on the road stem from human error I believe it is a crucial task for humanity to invest in the development of self-driving cars. We could easily save hundreds of lives every day.

2 Problem Statement

As mentioned in chapter 1 the problem of recognizing Traffic Lights using Computer Vision is trying to be solved in this Capstone project. The paper [2] describes the problems and challenges that this task offers. So far basically all solutions have been based off local and small datasets which does not offer a general solution to our problem. Finding a general solution is hard, because traffic lights are different in every part of the world. In [2] Figure 4 the author shows that even in different states in the USA the traffic lights have different positions and shapes.
We also face the issue of environmental influences which makes it hard for an algorithm to always capture the Traffic Light's shape, size and color respectively. One problem might also be that our algorithm confuses the tail lights of the driver in front of our car with a red traffic light and abruptly stops causing an accident with vehicles behind us.

3 Dataset

Finding a dataset that is challenging and provides variety is not an easy task. For this problem I worked with the LISA Traffic Light Dataset [1] which I obtained from Kaggle.com [4].
It contains more than 44 minutes of annotated traffic light data. The database consists of continuous test and training video sequences, totaling 43,007 frames and 113,888 annotated traffic lights. The sequences are captured by a stereo camera mounted on the roof of a vehicle driving under both night- and daytime with varying light and weather conditions. Only the left camera view is used in this database, so the stereo feature is in the current state not used.

It was collected in San Diego, California, USA. The database provides four day-time and two night-time sequences primarily used for testing, providing 23 minutes and 25 seconds of driving in Pacific Beach and La Jolla, San Diego. The stereo image pairs are acquired using the Point Grey's Bumblebee XB3 (BBX3-13S2C-60) which contains three lenses which capture images with a resolution of 1280 x 960, each with a Field of View(FoV) of 66 degrees.
The left camera view is used for all the test sequences and training clips. The training clips consists of 13 daytime clips and 5 nighttime clips. The annotation classes are "go", "go forward", "go left", "warning", "warning left", "stop", "stop left".

4 Solution statement

For the Solution of this problem I choose to use the YOLO Algorithm, which provides fast and accurate state-of-the-art performance in Object Detection. It runs an input image through a Convolutional Neural Network and outputs an encoding where each cell contains information about 5 boxes (a box tries to capture an object).
We then filter these boxes to only keep those with the highest probability. As this algorithm is fairly non-trivial and a full explanation would go far beyond the dimensions of this proposal I attached the two main papers [5][6] for anyone who wants to delve deeper into the Math behind it.
In general I will take the pre-trained weights from the official YOLO Website [7] and fine-tune them for our Traffic Light dataset. After that our model should be able to recognize Traffic Lights on picture frames and annotate them. In contrary to some methods discussed in [2] I am choosing an end-to-end Deep Learning approach here. Other algorithms might include a pipeline such as Detection-Classification-Tracking but my approach will try to solve all of that in one go.

5 Benchmark model

In [2] Table 3 we can observe the performance of many Traffic Light Detection systems with different approaches. The algorithms specifically used on our Dataset achieved a Precision of 30.1 during mixed day-time illumination conditions and 65.2 at night time and a Recall of 50.0 respectively.
Both are using aggregated channel features for detection. The only algorithm that used CNNs for classification was able to score 97.83% on Accuracy. In that case prior knowledge of the TL was used for detection.Therefore we only have the knowledge that YOLO performs well on Object-Detection and has been used for Traffic Sign Detection, which makes it eligible to try on our problem.

6 Evaluation metrics

The performance measures most commonly used on this problem are Accuracy, Precision and Recall. The definitions are shows in the following equations. TP, FP, FN, TN are abbreviations for true positives, false positives, false negatives and true negatives.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

A precision close to one indicates that all the recognized TL states are in fact correctly recognized.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

A recall close to one indicates that all the TL states, in a given video sequence, were correctly recognized by the proposed system.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{3}$$

An accuracy close to one indicates that the system detects all TLs with no false positives.

7 Project Design

My intended workflow is as followed:

1. Data Preprocessing
   - get familiar with the data
   - get the data in the right format
2. Implement Transfer Learning
   - get the pretrained yolo weights
   - re-train the last layer to our classes
   - measure performance
3. Use the new weights to build a working detection system
   - make last steps to have the algorithm draw boxes on frames
   - test performance on the test sequences using evaluation metric

The biggest challenge in this project will be to implement the Transfer Learning as this is highly non-trivial and requires a good understanding of the algorithm and frameworks like Keras or Tensorflow.
To retrain the weights we have to get the data into the right format so we can use it for our model. This will require a decent amount of Data exploration and preprocessing as well as writing programs to process it.
As the architecture of the model's body is already given through the Yolo weights we don't have to engineer the Neural Network architecture. In a nutshell the main task is to remove the last layer of the existing model and replace it with a Softmax layer which is fitted to our seven classes.
It will be useful though to read the documentation and the papers regarding the YOLO algorithm to get a better understanding of the CNN architecture being used. If this step is successfully implemented the task is to write the necessary functions to have our models draw boxes on the frames so we are able to visually observe how it performs. After visual observation the last step will consist of implementing functions to test the performance of our system using the evaluation metrics discussed in Chapter 6 on page 4. I expect to spend the most time on the Data Preprocessing and Transfer Learning steps.
As the Yolo model is far from being simple, the training time is to be planned in as I expect the model to retrain for about 1-2 days, using an AWS GPU instance.

References

[1] https://www.kaggle.com/mbornoe/lisa-traffic-light-dataset.

[2] Morten Bornø Jensen, Mark Philip Philipsen, Andreas Møgelmose, Thomas Baltzer Moeslund, and Mohan Manubhai Trivedi. Vision for looking at traffic lights: Issues, survey, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1800–1815, 2016.

[3] Mark Philip Philipsen, Morten Bornø Jensen, Andreas Møgelmose, Thomas B Moeslund, and Mohan M Trivedi. Traffic light detection: A learning algorithm and evaluations on challenging dataset. In *intelligent transportation systems (ITSC), 2015 IEEE 18th international conference on*, pages 2341–2345. IEEE, 2015.

[4] https://www.kaggle.com.

[5] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[6] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[7] https://pjreddie.com/darknet/yolov2/.