Machine Learning Engineer Nanodegree

Capstone Proposal

Spandan Roy
July 07th, 2019

Proposal

Domain Background

Traffic sign recognition is one of the most important parts in self-driving car field. It enables vehicles to recognize modern traffic signs on the road during driving such as speed limit and stop sign. On November 8th, 1968. The Vienna Convention on Road Signs and Signals was held to increase road safety and aid international road traffic by standardising the signing system for road traffic, such as road signs, traffic lights and road markings
(Wikipedia, 2018). The first TSR systems which recognized speed limits were developed appeared at the end of 2008, on the redesigned BMW 7 Series. Deep learning neural networks method is one of the major algorithms for pattern recognition problems currently. It is also widely used on Traffic signal recognition on self-driving cars.

Reference:

Academic paper - Radu Timofte*, Markus Mathias*, Rodrigo Benenson, and Luc Van Gool, Traffic Sign Recognition - How far are we from the solution?, International Joint Conference on Neural Networks (IJCNN 2013), August 2013, Dallas, USA.

Problem Statement

The problem to be solved is how to teach computers recognize the traffic sign when it receives a traffic sign image no matter what the scale is. For instance, we want the machine to know that the sign in the picture is a 40km/h speed limit sign even though the sign is relatively small and not perfectly clear in the picture, and output “40km/h Speed Limit”.

Datasets and Inputs

The dataset I used is a subset of the Belgium Traffic Signs database, which is a multi-class, single-image classification challenge. There are 62 classes in total, and is a large, lifelike database. The images are taken in different angles and lighting conditions. There is one directory for each class. Each directory contains the corresponding images of the sign in .ppm format (RGB color), as well as a csv file that has annotations, but the number of images in each class is not same ,i.e., the classes are not balanced. There are training dataset and testing

dataset already separated in advance. Each of the two directories contain 62 subdirectories, named sequentially from **00000** to **00061**. The directory names represent the labels, and the images inside each directory are samples of each label.

Note: the delimiter of the csv file is semicolon (;).

The csv file contains the following details,

Filename – Image file the following information applies to

Width, **Height** – Dimensions of the image

Roi.x1, Roi.y1, Roi.x2, Roi.y2 – Location of the sign within the image

ClassId – The class of the traffic sign

The input images are the keys to train our machine to learn how to recognize traffic signs. The csv file also helps us focus on the right spot on the image and classify different traffic signs. It should be clear how the dataset(s) or input(s) will be used in the project and whether their use is appropriate given the context of the problem.

Solution Statement

The solution to this traffic sign recognition system project is Convolutional Neural Network method in Deep Learning. Through CNN and the input images, our machine is expected to: classify traffic signs in a video stream or in a picture.
CNN is an operation of filtering input data by setting up layers (filters). In image recognition, we use 2D filters to process data to generate the features. During training, the coefficients of these features are computed and optimized through backpropagation algorithm with gradient descent.

Benchmark Model

The paper "Traffic Sign Recognition – How far are we from the solution?" by Markus Mathias, Radu Timofte, Rodrigo Benenson, and Luc Van Gool can serve as a benchmark for this problem. It was published at the International Joint Conference on Neural Networks (IJCNN 2013),Dallas, USA, and can be found here:
https://btsd.ethz.ch/shareddata/publications/Mathias-IJCNN2013.pdf.

Authors report that their models reached an accuracy between 95% and 99% without including traffic sign specific knowledge in the classifiers.

Evaluation Metrics

Due to the fact that the dataset I chose is not evenly distributed, I chose weighted F1 score as the metric to this problem. F1 score is given by the following formula:

$$\text{F1\_score} = 2 * \frac{precision * recall}{precision + recall}$$

To understand how to calculate precision and recall, we need this chart to understand what are True Positive, False Positive, False Negative and True Negative:

		Predicted class	
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Figure 3: metric chart. Retrieved from http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

The calculation of precision and recall are given by the following formula:

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

The final F1 score is given by $F\ 1 = \frac{F\ 1score_1 * weight_1 + F\ 1score_2 * weight_2 + \ldots + F\ 1score_n * weight_n}{\sum_{i=1}^{n} weight_i}$

Project Design

In order to solve the traffic sign classification problem, I will follow these steps:

- **Data Exploration First:**

  I will load the data and explore the given images. I will write some basic code to see how the images look like, how the data is organized and decide which modifications have to be done. I will consider using data augmentation techniques, image normalization or cropping the images.

- **Implementation of a Model Architecture:**

  Build a traffic sign detector using OpenCV CascadeClassifier, which helps detect objects in a video stream.

  The final expected behavior is that, our model is able to detect the traffic sign in a video stream or in a picture. Building such a classifier is important and will boost our confidence even though it cannot tell what the sign means yet.

  Create a Convolutional Neural Network with transfer learning to classify traffic signs by applying proper number of layers with certain parameters, we will have the first CNN to train our model.

- **Training:**

  When the model architecture is defined, the model has to be trained. I will use Google Colab in order to accelerate training with GPU support. During training, I will monitor train and test losses to avoid overfitting. I will also evaluate accuracy improvement during training on a validation set.

- **Testing and Evaluation:**

  Finally, I will test the performance of the trained classifier on a test set. I will evaluate the precision, recall and f1 score, along with accuracy on new traffic sign images. I might include visualizations such as accuracy over time plots. I will compare the accuracy with the given benchmark accuracies. I will provide a text discussion of the final model architecture and its performance and conclude with a summary