Capstone_Proposal

February 5, 2019

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Antje Muntzinger
30th January 2019

1.2 Traffic Sign Recognition with Convolutional Neural Networks

1.2.1 Domain Background

Autonomous driving is one of the main research areas of artificial intelligence and machine learning. Traffic sign recognition has been available in advanced driver assistance systems since 2008 (https://en.wikipedia.org/wiki/Traffic-sign_recognition). Although research has been done for many years in this domain, there are still unsolved problems, such as computer vision in bad weather conditions, at nighttime, or additional traffic signs that are difficult to classify. I am particularly interested in this subject because I work at Mercedes-Benz and have written my PhD in the field of environment perception.

1.2.2 Problem Statement

In my capstone project I want to implement a traffic sign detector. The detector should get images of traffic signs of different classes as input and return the most likely class as output. The classifier should also correctly classify signs in different angles and lighting situations.

1.2.3 Datasets and Inputs

In order to achieve this goal, I want to use the Belgian Traffic Sign Recognition Benchmark (https://btsd.ethz.ch/shareddata/). The training and testing sets can be found here: https://btsd.ethz.ch/shareddata/BelgiumTSC/BelgiumTSC_Training.zip and https://btsd.ethz.ch/shareddata/BelgiumTSC/BelgiumTSC_Testing.zip, respectively.

The Belgian Traffic Sign Recognition Benchmark is a multi-class, single-image classification challenge. It contains images from 62 classes in total and is a large, lifelike database. The images are taken in different angles and lighting conditions. Therefore, the use of this dataset is appropriate given the context of the problem.

1.2.4 Solution Statement

To solve this classification problem, I want to use a Convolutional Neural Network (CNN). The CNN should get an image as input and give the probabilities of the classes as output. I will design a CNN from scratch in PyTorch rather than use transfer learning because I want to understand the neural net's behavior in detail.

1.2.5 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-IJCNN-2013.pdf. Authors report that their models reached an accuracy between 95% and 99% without including traffic sign specific knowledge in the classifiers.

1.2.6 Evaluation Metrics

The performance of the model can be evaluated using accuracy score. This is appropriate since all benchmark models are evaluated using accuracy score as well. Accuracy is defined as

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

1.2.7 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 Then I will implement a model architecture. I want to use a CNN because it conserves spacial structures in the image, unlike a simple Markov Decision Process. I will probably use PyTorch for implementation, other options would be TensorFlow or Keras, for example. I want to implement a CNN from scratch. Alternatively, I could use transfer learning on pre-trained models if I am not satisfied with the CNN's performance. I will try different architectures, layers and activation functions and also tune hyper parameters such as learning rate or number of epochs.

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 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.