Recognition Module for Traffic Sign Classification

Jannes Bikker

Technische Universität Clausthal, Clausthal-Zellerfeld 38678, Deutschland

1 Introduction

As part of our project, I developed the recognition part for the traffic sign classification. For this, I used the German Traffic Sign Recognition Benchmark (GTSRB) dataset. Unlike the German Traffic Sign Detection Benchmark (GTSDB), which includes non-localized pictures, the GTSRB dataset already provides localized traffic signs. Other team members handled the localization step, and I trained the model using the already localized data.

Note: Initially, Marvin and I attempted/played around with a classification model using the initially selected fruit dataset. However, due to incorrect labeling in the dataset, this approach was ultimately unsuccessful. Since this implementation did not contribute to the final solution, I will focus solely on my CNN-based traffic sign recognition model in this report.

2 Data Preparation

I created scripts to download and prepare the datasets for training, validation, and testing. Since the images were in .ppm format, I converted them to .jpg to match my training pipeline. The test data, initially stored in a single folder, was reorganized into separate class-specific folders to analyze classification "difficulty" across different traffic sign classes.

3 CNN Training and Evaluation

To train the recognition model, I implemented a CNN using TensorFlow and Keras (a lot of trial-and-error as explained in the group report). The final architecture consists of multiple convolutional layers with batch normalization, dropout layers for regularization, and dense layers for classification.

Training was significantly accelerated by using computational resources provided by Professor Ohl (Ostfalia). This reduced the average training time per epoch from around 6 minutes on my local machine to just 26 seconds. It technically would have been possible to train the model on my local machine nevertheless, I saved a lot of time with the provided VM.

Each training epoch was saved to monitor progress. Once the model architecture was finalized, I evaluated its performance using the test dataset. The global results were:

- Mean accuracy: 99.36%
- Correct classifications: 12,507/12,630 (99.03%)
- Misclassifications: 123/12,630 (0.97%)

Given the overall high accuracy and low misclassification rate, I was very satisfied with the model performance.

4 Prediction and Deployment

The goal was to develop a model that could take an image as input and return a classification result. However, I encountered issues deploying the trained model on my local machine, likely because I do not have an NVIDIA graphics card to support CUDA acceleration.

After testing various TensorFlow and Python versions without success, I discovered that I could disable model compilation during loading. To demonstrate the prediction process, I created a `predict.ipynb` notebook that allows users/team members to input an image and obtain a classification result. This model can now be used to classify localized traffic sign images effectively.

It also proved to work as expected in the later implemented pipeline that sends a non-localized picture into the localization model and then passes the results into my recognition model.

5 Interesting Problems

One of the more challenging traffic sign categories for the model were prohibition signs with a large red diagonal line in the middle. These signs had the lowest classification accuracy. The likely reason is that the red diagonal line is a strong distinguishing feature, but it is common across multiple prohibition signs while not being visually similar to other traffic sign categories. This made it harder for the model to correctly differentiate between specific prohibition signs, leading to a higher misclassification rate.

6 Source Code

Available at github.com/jarboyd/ai-lecture-project-ws2425. The recognition source code is located in `./recognition`.