Ruprecht-Karls-University Heidelberg

Faculty of Mathematics and Computer Science

Project: Object Recognition and Image Understanding

Traffic Sign Detection in Colour Images

two approaches in comparison

Author:
Daniela Schacherer, 3165890,
M.Sc. Angewandte Informatik
Marvin Klaus, 3486809, M.Sc.
Angewandte Informatik
Lecturer:
Prof. Björn Ommer
Date
24.07.2018

0ptContents

0ptIntroduction

Autonomous driving is nowadays one of the most growing research fields. In this context visual detection and recognition of road signs becomes particularly important. The project presented in this report aims at the exploration of two state-of-the-art approaches towards the detection of traffic signs in color images. In contrast to traffic sign recognition, detection aims merely at locating the traffic signs in the images, however, not at classifying them.

0ptDataset

The dataset employed for this task was made publicly available by the University of Bochum as part of a traffic sign detection competition in 2013 [**?**]. The data set comprises 900 images (1360 $\times$ 800 pixels) in PPM format containing 1206 traffic signs. Additionally, the image sections containing only the traffic signs and a CSV file containing ground truth information (location of the traffic signs within the images) are provided. Each image contains zero to six of those traffic signs included in the competition like, for example, speed limit or stop signs. For a complete list of the competition relevant categories see Appendix A. The sizes of the traffic signs in the images vary from 16 to 128 pixels w.r.t the longer edge [**?**].
The images were collected such that different street scenes as urban, rural, or highway as well as different lightning and weather conditions are equally represented in the dataset. Figure 1 shows some examples contained in the data set. [**?**].

Figure 1: Example images from the dataset.

0ptFirst approach: Support Vector Machine (Daniela)

In the first approach we use a linear support vector machine (SVM) based on Histogram of oriented gradient (HOG) features to detect traffic signs in colour images. This combination is often used for object detection since HOG descriptors were introduced by Dalal and Triggs in 2005 [**?**]. First, the SVM is trained to distinguish image patches showing traffic signs from image patches showing something else. For detection of road signs in a given image, a window is slided across the image at different scales employing the SVM for every image section.

0ptData preprocessing

The training set was randomly divided into 600 images for training and 300 images for testing.

Positive training samples

All traffic signs were extracted from the training set images using the ground truth locations provided. As the ground truth bounding boxes are not necessarily quadratic, they were converted to a quadratic shape by taking the larger side length and enlarging the smaller one equally in both directions. The extracted patches were then resized to a size of 30×30 pixels.

Negative training samples

The easiest way to generate a "starter" set of negative samples was to take some images from the training set which do not show any sign. 30×30 pixel sized patches were extracted from these images using a stepping window. A stepping window was preferred over a sliding one in order to avoid the negative samples being statistically dependent on each other. For a start, we sampled 5000 negative image patches.

Some positive patches showing a traffic sign, as well as some of the negative patches, showing something else, are depicted in Figure 2.

Figure 2: The first and second row show positive patches containing traffic signs, the third and fourth row show negative patches without a traffic sign contained. The size of all patches is 30×30 pixels.

0ptFeature extraction using HOG

0ptTheory HOG

The **H**istogram of **O**riented **G**radients (HOG) is a feature descriptor widely used in the context of object detection. This method basically summarizes gradient orientations in localized sections of an image by performing the following five steps [**?**]:

1. Global image normalization using gamma compression: thereby one can compensate for variations in illumination or local shadowing effects.
2. Computation of first order image gradients.
3. Computation of gradient histograms: in so-called cells of predefined size (e.g. 5×5 pixels) gradient orientations over all pixels in the cell are accumulated in a histogram.
4. Normalization across blocks: as gradient strength must be locally normalized cells are grouped together into so-called blocks, on which normalization is performed. Possible methods for block normalization are, for instance, the L1-norm or the L2-norm.
5. Collection of HOG descriptors from all cells into a combined feature vector.

0ptApplication

We used the HOG implementation from scikit-image [**?**]. As proposed by Houben *et al.* who performed a similar approach, we considered 8 gradient orientations, used cells of size 5×5 pixels and a block size of 2×2 cells [**?**]. As block normalization method we chose the L2

normalization. The final training set used to train the SVM is then constructed from the HOG features of the positive and negative patches.

0ptClassification using SVM

0ptTheory SVM

A support vector machine is a well established supervised classification and regression method. Given a set of labeled training examples from two classes - positive and negative - a SVM training algorithm creates a model that can assign new examples to one category or the other. Formally, a SVM constructs a hyperplane by maximizing the margin between positive and negative data points (see Figure 3).

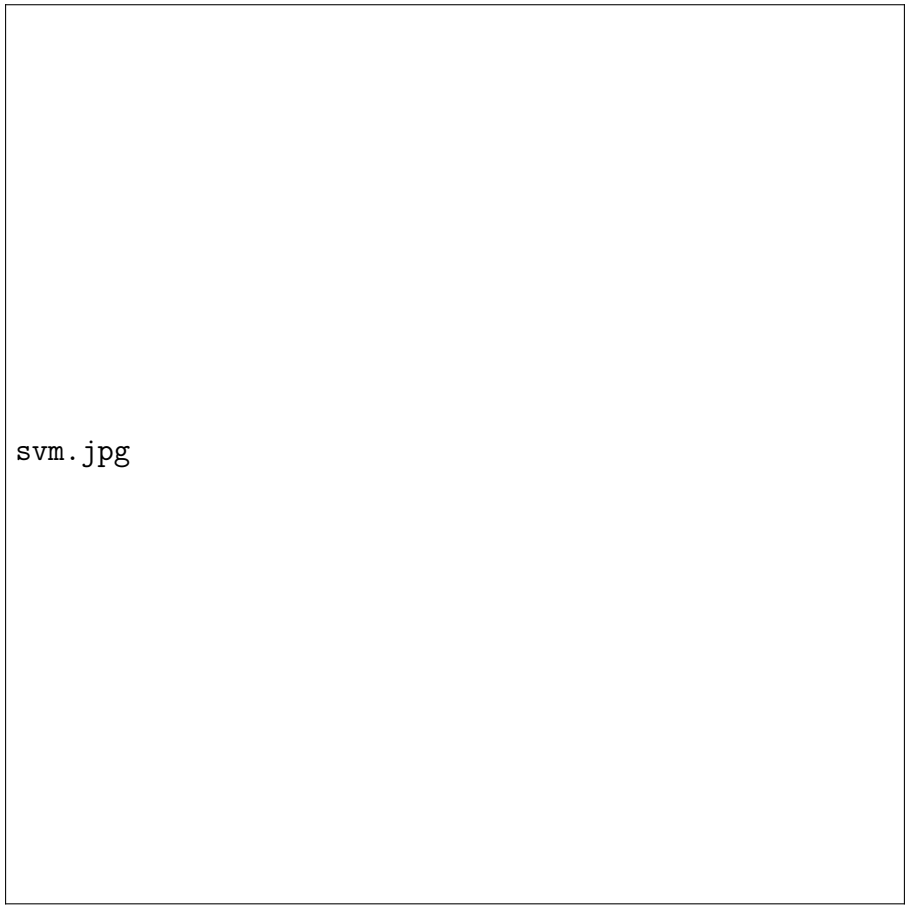

Figure 3: A Support Vector Machine tries to maximize the margin between positive and negative data points. Source: `https://www.slideshare.net/kambliruta/event-classification-prediction-using-support-vector-machine`

0ptTraining

We used the SVM implementation provided by sklearn [**?**] with the default values except for the kernel which we chose to be linear. With the intention to improve performance of the SVM as well as to obtain a sufficiently large training set, we additionally included a step of Hard Negative Mining (HNM). Therefore, the SVM is first trained with the previously constructed

training set. Then, for every image and possible scale of the images dedicated for training, that contain no traffic sign, we apply the sliding window technique. At each window we compute the HOG descriptors and employ the pre-trained SVM. Each falsely detected patch is taken and explicitly added as a negative example to the training set. Finally, the SVM is retrained using the enlarged training set. In the results section we compare the outcome with and without Hard Negative Mining based on the achieved training / test accuracies.
We expect a significant improvement when training with Hard Negative Mining as we obtain more variety in the negative training samples especially by using different scales of the images. Here we used $\sqrt{(2)}^i$ with $i = 0, ..., -5$ as scale factor to obtain an image pyramid with 6 layers including the original image. By the use of an image pyramid for Hard Negative Mining we can capture larger structures which might get confused easily with a traffic sign and explicitly learn the classifier to recognize them as negative examples.

0ptDetection
For detection of traffic signs in a given image we slide a window of size 30×30 pixels across the image at different scales - here we also used an image pyramid with scale factor $\sqrt{(2)}^i$ where $i = 0, ..., -5$ - and obtain a prediction for every image section from the SVM. If the thereby received regions possibly showing a traffic sign are obtained from another than the original scale of the image these are back computed to the original image's size. As a result we finally obtain a list of predicted bounding boxes relating to the original input image.

0ptPerformance Measure
For a quantitative measure of the detection performance each predicted bounding box P is compared against every ground truth bounding box G by means of the Jaccard similarity, which is defined as the intersection over union of the two bounding boxes:

$$S(P, G) = \frac{|P \cap G|}{|P \cup G|}$$

If $S(P, G) \geq 0.5$ for any predicted bounding box P, the ground truth sign G is considered as detected. In the end, we record the fraction of detected ground truth signs in all images (recall).
In addition, a measure for the amount signs detected erroneously - i.e. the false positive rate - is needed. Therefore, we assess the fraction of predicted bounding boxes that have zero overlap with any of the ground truth bounding boxes. By computing $1-$ the false positive rate we yield the detection precision.

0ptSecond Approach: R-CNN (Marvin)

In the second approach we created a R-CNN to detect traffic signs in traffic scene images. For this we first find all interesting regions in an image with the selective search algorithm and then use a CNN network to classify if there is an traffic sign in this region or not.

0ptData preprocessing

Before training the CNN we need to do some data preprocessing for this approach. The CNN will train and test with the help of the traffic sign images, classified to different types of signs by the creators of the dataset. Since we just want to distinguish between "no sign" and "sign", we collect them all together and use them as one single class of objects.

Since the used dataset just offer traffic signs and no *negative* training samples we need to create them. Similar like in the first approach, we filtered out all the traffic scene images without any traffic sign present. After that we took 32x32 parts from different images in random locations of the image. We collected 1000 negative samples through this approach.

0ptImprovement proposal

With more time left or a bigger group size there are several possible steps to improve the data preprocessing. For example as you can see in figure 4 some traffic signs are less represented than others. To do something about this you could apply transformations on the images to rotate or tilt the traffic signs.

Also the collection of negative samples can be improved. For example you can collect these images more targeted, like collect more shapes which look like signs but aren't or collect just a fraction of a sign. Also you can collect bigger pictures than 32x32 images and scale it down after that so you have more objects in one image. As we will see later the selective search algorithm also returns regions with bigger size.

0ptDataset creation

With the preprocessed data we then create a dataset of these images. To make sure we have every traffic sign type in both the train and the test data, we split the images of every type in two parts where 0.3% are test data and the rest train data. The results you can see in figure 4 where the blue columns show the train data and the orange columns the test data. With this data we created two datasets for training and testing. For both all the images get resized to 32x32x3. One row in the finish dataset then looks like this: {*'image': image, 'id': sign_id, 'folder': image_folder*}. The keys of the dictionary are self explanatory.

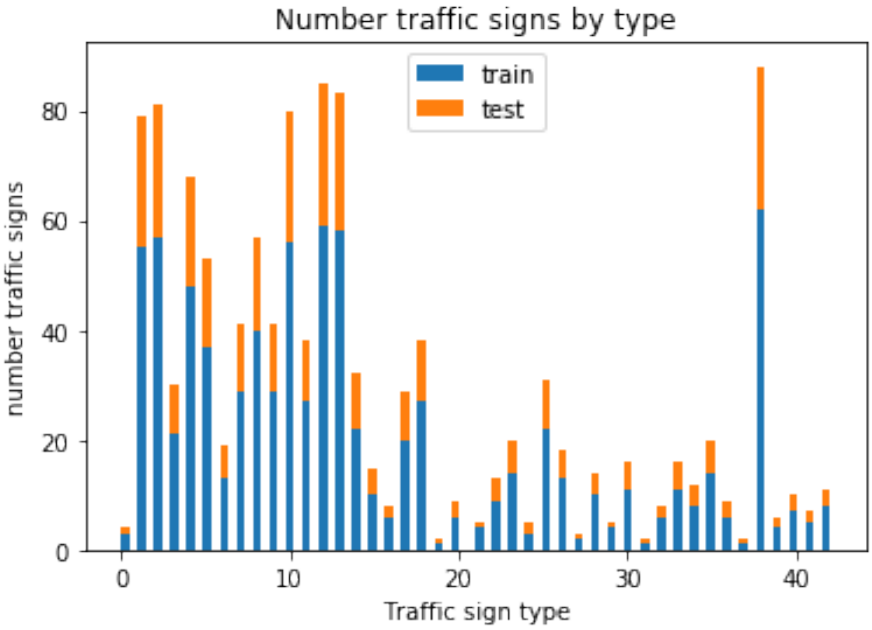

Figure 4: number of traffic signs by id divided by train and test data

0ptCNN

0ptBasics

A convolutional neural network (CNN) is artificial neural network which is used to analyzing and classifying images. A CNN consists essentially of filters (convolutional Layer) and aggregation layers (pooling Layer), which repeat alternately. At the end there are one or more fully connected layers to output the probability of different objects the image could represent.

0ptArchitecture

To classify if there is a traffic sign in one image region we use a CNN. The network consists of two convolutional layers, both have a size of 5x5 where the first outputs 16 filters and the second layer 32 filters. Both layers are follows by an normalization, ReLU and a pooling layer. After the convolutional layers we use fully connected layers to break the output down to two singles values.

0ptTraining

To train the network we used the CrossEntropyLoss function and the Adam Optimizer. The output of the network consists of two values, the first should imply that there is no traffic sign in the image and the second one shows the probability that there is a traffic sign in the given image. With the Adam optimizer the accuracy of the network already looks really good after just 15 epochs. As you can see in 11 we got an accuracy close to 94%.

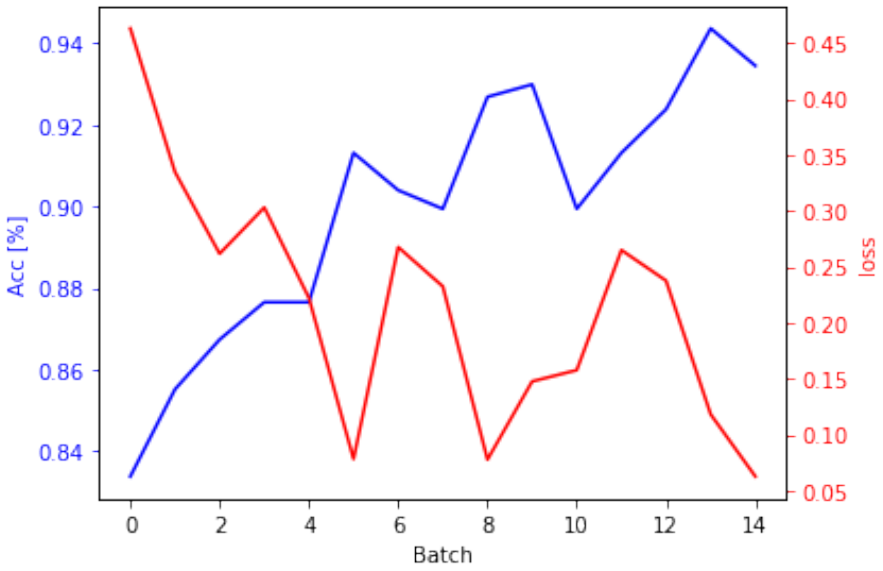

Figure 5: loss and accuracy of net CNN

0ptSelective search

0ptBasics

Selective Search is a region proposal algorithm often used in object detection. It is based on computing hierarchical grouping of similar regions based on color, texture, size and shape compatibility.

0ptRegion proposals

Since we want to detect traffic signs in an image we first need to get some regions which we can classify with our CNN. For this reason we use the selective search algorithm which return bounding boxes of interesting regions in an image. While working with the selective search algorithm we noticed some issues with this algorithm. With the two parameters you can adjust, we found no perfect solution to find all the traffic signs in every image. At the end we took parameters where a good amount of traffic signs are represented in the interesting regions (scale=250, sigma=0.9). The network returns a huge amount of interesting regions for which reason we need to filter these regions. For example we just need regions which are in an certain size interval and are nearly quadratic. After this step an output of the selective search algorithm looks for example like in figure 6 where each red rectangle shows an interesting region in the image.

0ptTesting

To get an accuracy of our traffic sign detector, we need to run all the found interesting regions in an image with our trained CNN. Like in the creation of datasets all of the regions need to be resized to a size of 32x32x3. After all regions of the wanted images are classified we need to examine how good our detection with the R-CNN works. For this we took the ground truth

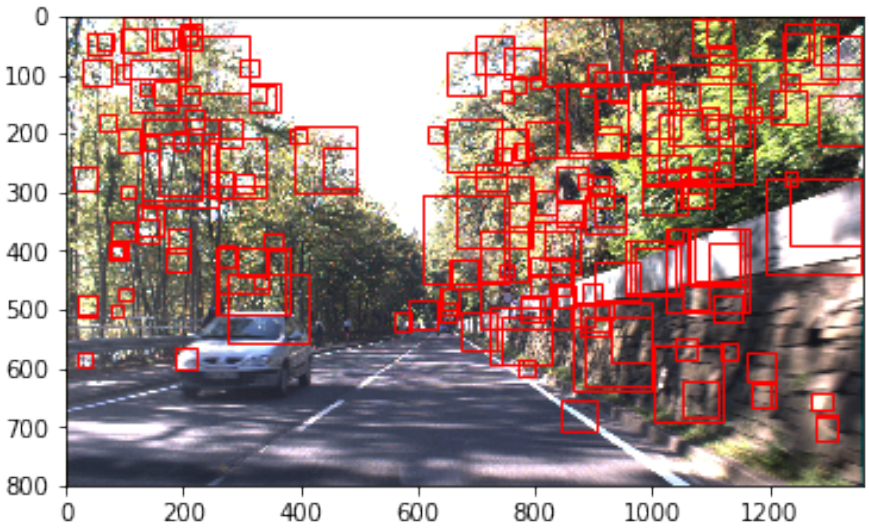

Figure 6: Example of the selective search algorithm after region filtering. All red rectangles are interesting regions.

data and computed the Jaccard similarity for all the bounding boxes in the ground truth data and the regions which were classified as a traffic sign. A successful detection is then a region with a similarity ≥ 0.5. With this measurement the R-CNN gets an accuracy of just 32%. This result is not really good, why is described in the chapter 5.2.

0ptResults

0ptFirst Approach (Daniela)

The average detection performance measured by recall, precision and false positive rate on 30 randomly chosen images from the training respectively test set is reported in Table 1. Contrary to our expectations the prediction accuracy (recall) decreases when training the SVM with HNM. However, from precision and false positive values or when having a look at the bounding box predictions of the SVM trained with respectively without HNM, which are presented in Figure 7, one can see that Hard Negative Mining significantly lowers the false positive rate. On the left image both ground truth bounding boxes are detected, however there is a large amount of false positives. In the right image the second ground truth bounding box is not found, but the false positive rate is much lower. Due to the a better precision respectively lower false positive rate, we nevertheless assess Hard Negative Mining as an improvement and keep applying it when training our classifier.

Table 1: Recall, precision and false positive rate on the train respectively test set obtained with and without HNM. For training and detection a sliding window of 30×30 pixels was used. The window's step size was 10 pixels for training and 5 pixels for detection.

	train acc			**test acc**		
	recall	precision	false positives	recall	precision	false positives
w/o HNM	95.00%	15,09%	84,91%	99,17%	6,29%	93,71%
HNM	84,73%	56,54%	43,46%	91,67%		11.6%

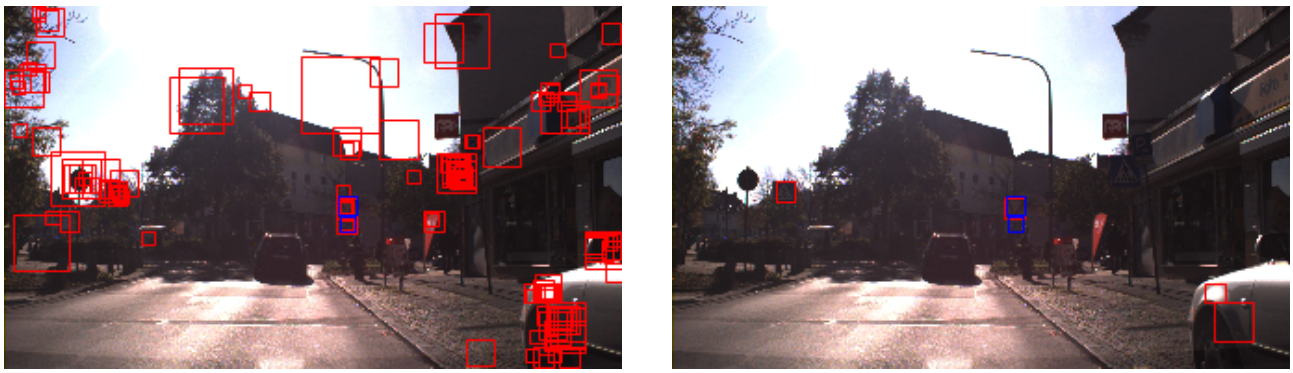

Figure 7: Bounding box prediction for the same image once using a SVM trained without HNM (left) and once trained with HNM (right). Ground truth bounding boxes are indicated in blue and predicted bounding boxes in red.

In Figure 8 some training images are shown in which the ground truth bounding boxes are indicated in blue while the predicted bounding boxes are drawn in red. The respective detection performances are noted below. Almost all traffic signs - if present - are detected and there are only few false positive predictions. Especially interesting is the image in the upper middle:

while the construction site sign is not detected possibly due to the poor lighting conditions, the hexagonal light spot in the upper left corner is recognized as a traffic sign instead.

Figure 8: Images from the training set with ground truth bounding boxes in blue and predicted bounding boxes in red. Prediction accuracy in percentage according to the prediction measure explained in section 3.3.4 is given below each image.

Figure 9 shows images from the test set along with ground truth and predicted bounding boxes. One can see that almost all traffic signs are detected as it was the case for the training images except for the rightmost sign in the upper middle image and the lower right image. If there is no traffic sign at all like in case of the bottom middle image, the algorithm does also recognize that correctly. However, the upper right and lower left image show that there are still sometimes false positives, which is not punished by our performance measure. Possibly the number of false positives can be reduced if another step of Hard Negative Mining is included.

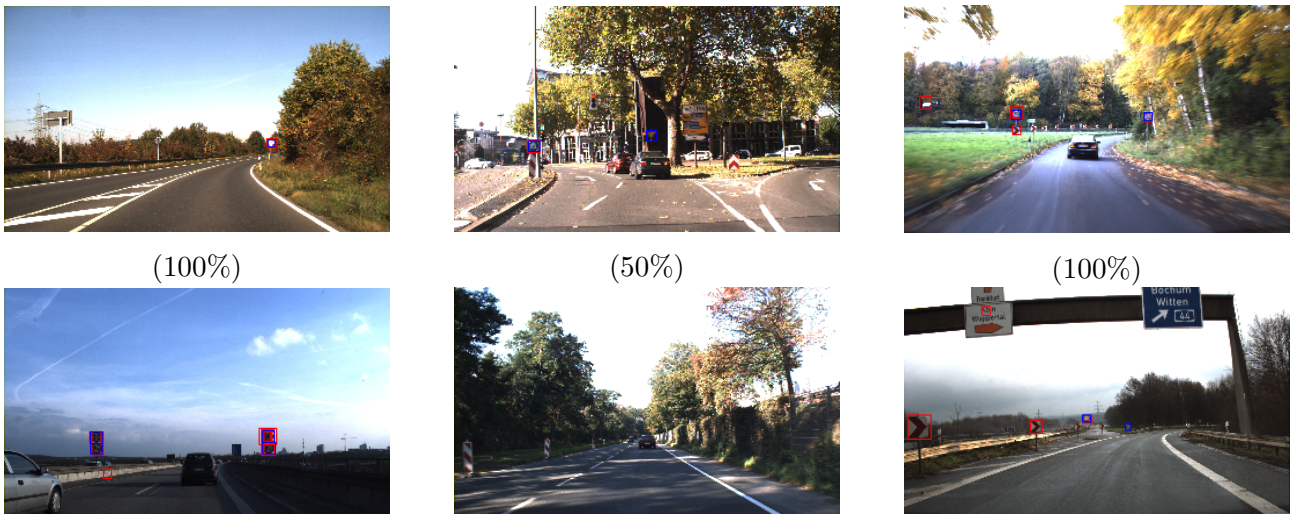

Figure 9: Images from the test set with ground truth bounding boxes in blue and predicted bounding boxes in red. Prediction accuracy in percentage according to the prediction measure explained in section 3.3.4 is given below each image.

Finally, we wanted to observe how the sliding window's step size during detection affects the performance. Figure 5.1 plots the resulting performance against the detection step size. Already with very few data points it becomes clear that the detection step size is very critical to the success of traffic sign detection.

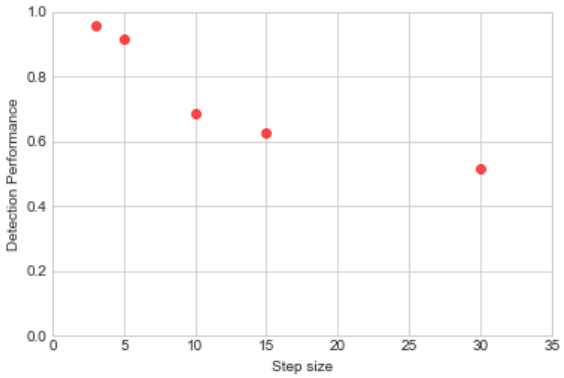

Figure 10: Detection performance measured as recall on 30 randomly chosen images from the training set versus the step size of the sliding window used for detection.

0ptSecond Approach (Marvin)

Now we want to look at the results of our network. For this we need to look at the three main steps we did in this approach and look at the results individually.

0ptNetwork

The first thing we did was to train and test the CNN. The accuracy and loss function are displayed in figure 11. We can see that the network can identify a traffic sign in approximately 94% of cases. But remember that we trained this network with the provided traffic signs of the dataset and not with the regions we get out of our selective search algorithm. If we would train the network with manually classified regions of the selective search algorithm, maybe the results in the detection, which we will see later, would be better.

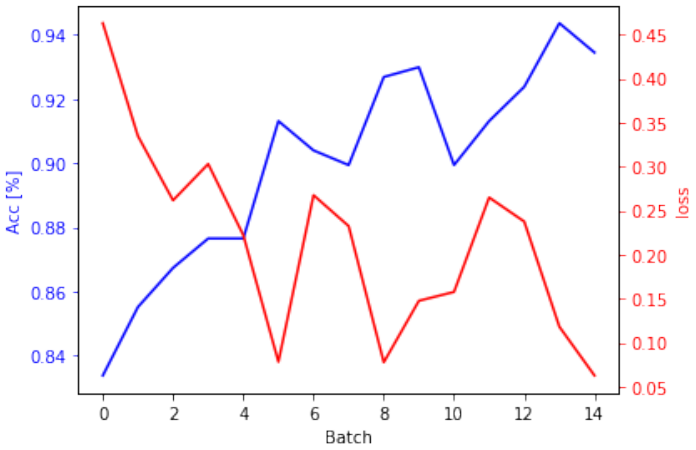

Figure 11: Loss and accuracy in each epoch

0ptSelective Search

Next we tried to get all interesting regions in a traffic scene image to classify each of these regions with our network. Figure 6 shows an example output of this algorithm. We can see, that in this image some traffic signs are recognized and some are not (the traffic signs are on the right side of the road and just the one at the top is marked). We tried several parameters for the selective search algorithm but couldn't find a perfect setting to recognize all traffic signs in all images. If all traffic signs in the example image were recognized, some traffic signs in other images will not. Keep that in mind when looking at the results of the traffic sign detection.

0ptDetection

After getting all interesting regions in 40 random images, we classify each region with our CNN. We compared each region classified as traffic sign with the ground truth bounding box. The classification was successful if the two bounding boxes had a Jaccard similarity ≥ 0.5. With this criterion the proportion of successful found traffic signs in 50 images is just 32.9%. This

number depends a lot on the random creation of the datasets. We also saw accuracies up to 43%.

The results are a little bit surprising considering a CNN which recognized around 94% of traffic sign images. Next we want to try to explain this phenomenon.

1. As we saw in the results of the selective search algorithm, some traffic signs weren't even recognized and therefore not classified which counts as a not found traffic sign.
2. Also we looked at some images that were classified as traffic signs but were not counted as a successful detection. We saw some images which displayed a traffic sign but even with a good proportion of the sign in the image, the jaccard similarity was less than 0.5.
3. As we saw in figure 4 some traffic sign images are just underrepresented so that the network couldn't learn them good enough. This can affect the success of classification.

With all these points we can explain the surprising bad results for the traffic sign detection with R-CNN's. But of course since simple R-CNN are a older approach there are better solutions to use for this problem nowadays. For example Fast R-CNN's and Faster R-CNN's are introduced which improve the region proposal computation. The Faster R-CNN for example introduce the Region Proposal Network which enables nearly cost-free region proposals. This would help us in two of the three enumerated points.

0ptDiscussion

To sum up the first approach, one can say that HOG features learned by a SVM are very well suited to detect traffic signs in color images. By the use of Hard Negative Mining the false positive rate can be lowered significantly, however, one looses some accuracy at the same time. An interesting idea could be to combine the predictions of a simply trained SVM with the ones from a HNM-trained SVM. Generally, the approach is very sensitive to the setting of the hyper-parameters especially towards the step size of the sliding window for detection. The smaller the step size the better the detection performance - at the price to a much higher computational effort. It could thus be an improvement to use region of interest extraction instead of a sliding window, like for instance selective search. Another idea could be to include some image patches slightly overlapping with a traffic sign in the negative samples such that the classifier would get less easily confused by objects partly being similar to traffic signs. This could possibly further reduce the amount of false positive predictions.

In the second approach we saw that the big problem with R-CNN's is that the selective search and the combination of region proposals and CNN's don't work very well together even if the CNN is well trained. To solve this problem it is probably the best to forget simple R-CNN's and work with Fast R-CNN's and Faster R-CNN's. If you really want to use a simple R-CNN, we count out some solutions for the problems that occur in these types of approaches in chapter 4.1.1.

In comparison, the SVM based on HOG features provides more convincing results than the R-CNN. Regarding computational effort the R-CNN is faster with the main bottleneck being the selective search algorithm. Faster R-CNNs provide one solution to this problem. They combine a region proposal network (RPN) with a R-CNN. The RPN is a fully convolutional network which is trained end-to-end to generate valuable region proposals, which are then used by the CNN for detection [**?**].

0ptAppendix

0ptAppendix A

0 = speed limit 20 (prohibitory)
1 = speed limit 30 (prohibitory)
2 = speed limit 50 (prohibitory)
3 = speed limit 60 (prohibitory)
4 = speed limit 70 (prohibitory)
5 = speed limit 80 (prohibitory)
6 = restriction ends 80 (other)
7 = speed limit 100 (prohibitory)
8 = speed limit 120 (prohibitory)
9 = no overtaking (prohibitory)
10 = no overtaking (trucks) (prohibitory)
11 = priority at next intersection (danger)
12 = priority road (other)
13 = give way (other)
14 = stop (other)
15 = no traffic both ways (prohibitory)
16 = no trucks (prohibitory)
17 = no entry (other)
18 = danger (danger)
19 = bend left (danger)
20 = bend right (danger)
21 = bend (danger)
22 = uneven road (danger)
23 = slippery road (danger)
24 = road narrows (danger)
25 = construction (danger)
26 = traffic signal (danger)
27 = pedestrian crossing (danger)
28 = school crossing (danger)
29 = cycles crossing (danger)
30 = snow (danger)
31 = animals (danger)
32 = restriction ends (other)
33 = go right (mandatory)
34 = go left (mandatory)
35 = go straight (mandatory)
36 = go right or straight (mandatory)

37 = go left or straight (mandatory)
38 = keep right (mandatory)
39 = keep left (mandatory)
40 = roundabout (mandatory)
41 = restriction ends (overtaking) (other)
42 = restriction ends (overtaking (trucks)) (other)

0ptAppendix B: Requirements

As programming language python 3.6 was used. The following modules are needed to execute the code:

- os
- numpy
- math
- matplotlib
- imageio
- random
- seaborn
- sklearn
- skimage
- scipy
- pandas
- selectivesearch
- torch
- pathlib
- functools