Detection of Traffic Signs using a Single Shot Detector

Marvin A. Ruder

Heidelberg University
Summer term 2020
Lecture "Deep Vision"

August 12, 2020

1 Introduction

The detection and recognition of traffic signs is a task neccessary to be solved in the upcoming field of autonomous driving. An autonomous car needs to detect traffic signs accurately in real time, without misdetections or overlooked signs, to adjust to traffic regulations while driving.

We propose a Deep Learning network that is able to detect traffic signs in the *German Traffic Sign Detection Benchmark* (GTSDB) dataset [4], which contains 600 training and 300 test images of street scenes with traffic signs in Germany.

Our proposed method uses a Single Shot Detector (SSD). An SSD is said to be much faster than other methods while reaching a comparable or even better accuracy. [6] This characteristics make it ideal for our purpose of detecting traffic signs from the perspective of an autonomous car.

This report is organized as follows: Section 2 provides information on the GTSDB dataset, followed by section 3 explaining the SSD model and changes made by us to optimize it for the dataset used. In section 4 the evaluation methods are explained, while the results obtained by it are discussed in section 5. Section 6 concludes the report and gives an outlook to possible future research.

2 Dataset

The *German Traffic Sign Detection Benchmark* dataset [4] was introduced at the IEEE International Joint Conference on Neural Networks 2013. It consists of 600 training images showing 846 traffic signs, and 300 test images showing 360 traffic signs, in raw PPM image format. The images contain zero to six traffic signs each. Ground truth information is provided in a separate file, containing the corresponding image identifier, pixel-based bounding box coordinates and a class identifier.

Traffic signs are sorted into one of 43 classes. The frequency of traffic signs in a class can be seen in figure 1.

All images show street scenes from the viewpoint of a car in regular traffic. They have been captured during daylight or dusk near Bochum in Germany in spring and autumn 2010, featuring various weather conditions. Different scenarios like urban, rural, or highways are represented. [4] The team "took special care on a diverse and representative compilation of single image frames" [3]. Also, images were sorted randomly into the training or test set [4]. A selection of example images is shown in figure 2.

The images vary in lighting conditions and motion blur, and traffic signs were captured from different angles and distances. All images have dimensions of 1360×800 pixels, while the traffic sign bounding boxes vary from sizes of 16×16 to 128×128 pixels. Information on the distribution of sizes and aspect ratios of the traffic sign bounding boxes can be obtained from figure 3.

The dataset was published together with a competition announcement, where participants were invited to propose algorithms that detect traffic signs of three different categories (prohibitory, mandatory, and danger traffic signs). The goal was to reach the highest possible area under the precision-recall curve, where values for precision[1] and recall[2] were obtained by running the algorithm with different threshold

[1] i.e. the percentage of detection results that are actually traffic signs [3]. We can calculate the precision using equation 5.
[2] i.e. the percentage of given traffic signs that were actually found [3]. We can calculate the recall using equation 6.

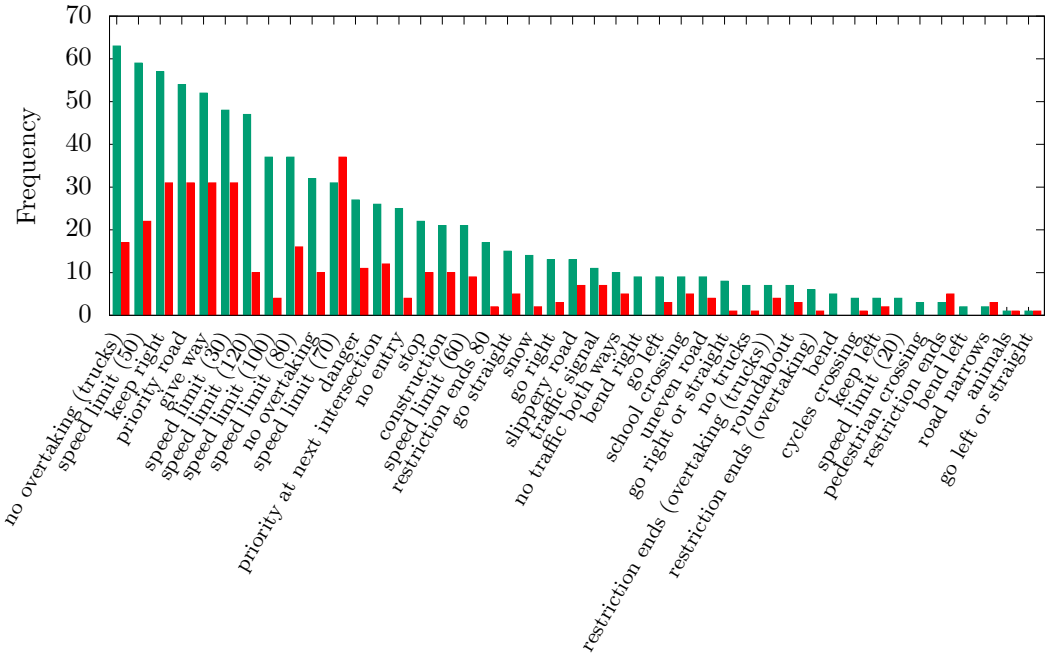
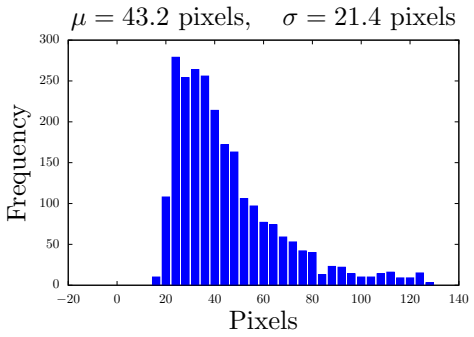
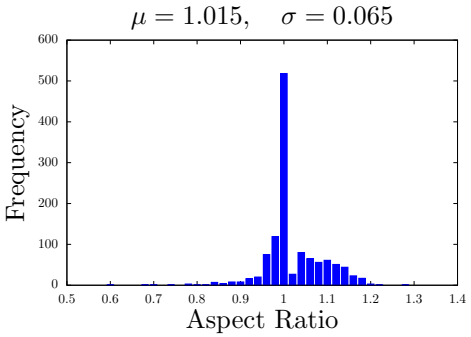


Figure 1: Frequencies of traffic signs in the training (green) and test (red) dataset.

Figure 2: Some example images from the dataset. They represent the variance in weather, lighting, and driving scenarios. [4]



(a) Size distribution of the traffic sign bounding boxes from GTSDB.



(b) Aspect Ratio distribution of the traffic sign bounding boxes from GTSDB.

Figure 3: Sizes and Aspect Ratios of the traffic sign bounding boxes from GTSDB.

values. Detection results from all submissions were published [5], which will later allow for a comparison to our method.

Apart from the competition submissions, the team from [4] proposed three different baseline approaches where results for the traffic signs of the categories mentioned above are published, also allowing for a comparison.

3 SSD detector

This section describes the proposed SSD detector. In section 3.1, the general structure of an SSD detector is explained, while changes made by us to it to optimize its results on the GTSDB dataset are presented in section 3.2.

3.1 Structure

The Single Shot Detector was introduced in 2016 by Liu et al. [6]. The team presented "a method for detecting objects in images using a single deep neural network". In this report, the SSD implementation from [10] was used as a starting point.

3.1.1 Base Network

The VGG-16 network architecture is used as the base network of the SSD model. It consists of 16 convolutional + ReLU layers with a total of 5 pooling layers, followed by 3 fully connected + ReLU layers. The number of filters per convolutional layer varies between 64 to 512, and their kernel size is 3×3. 1 pixel padding is used, so the image size does not decrease between convolutional layers. All pooling layers use Max pooling and a kernel dimension of 2×2 together with a stride of 2, halving the spatial dimension of their input for the following layers. [9]

However, slight changes were applied to the original VGG architecture by the teams from [6, 10] to work better within the SSD network. The input image size is set to 300×300 pixels, as opposed to the 224×224 image dimensions from [9]. Since this change in dimensions would lead to uneven dimensions in the layers after the third pooling layer, which was considered inconvenient, the ceiling funcion was chosen instead of the floor function to determine the third pooling layer's output size, resulting in an output of size 38×38 instead of 37×37. Also, the fifth pooling layer was modified to no longer halving the feature map dimensions by setting both stride and padding to 1 and using a 3×3 kernel. [10]

Another problem with VGG-16 is that it contains fully connected layers, which are not needed in the SSD approach. Instead, the first two fully connected layers are reshaped to convolutional layers with 1024 filters per layer, while the last fully connected layer is simply disposed of. This process leads to the network being fully convolutional. [10]

3.1.2 Auxiliary Convolutional Layers

To support detection of larger objects, 8 additional convolutional layers were appended to the modified VGG-16 architecture. Instead of using pooling layers, size reduction is reached using a stride of 2 in every

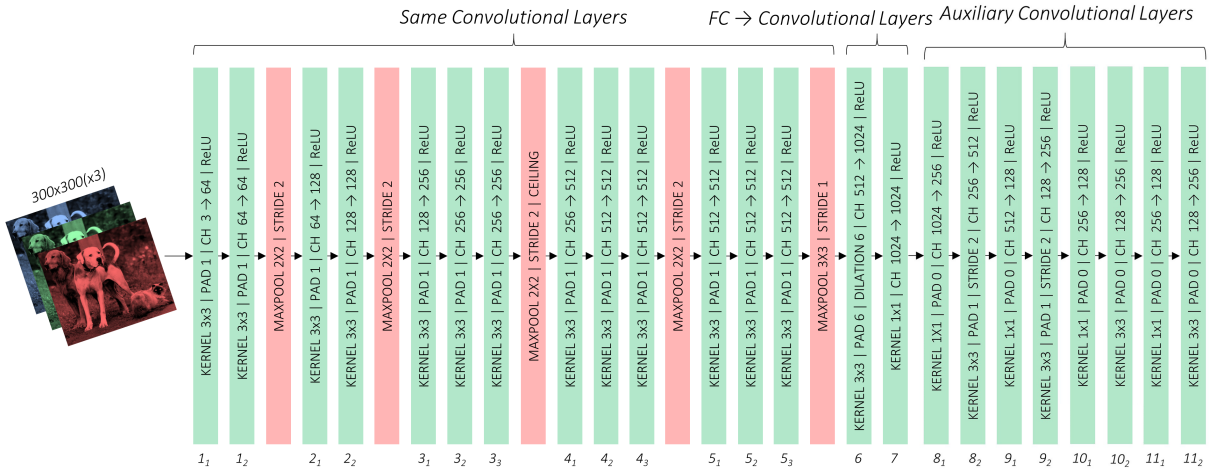


Figure 4: Base network and auxiliary layers. [10, edited]

second layer, resulting in feature maps of spatial dimensions between 19×19 and 1×1. The number of filters per layer is either 128 or 256. [10]

Both the base network and the auxiliary layers are visualized in figure 4.

3.1.3 Priors and Multi-Scale Feature Maps

To support detection of both larger and smaller objects, layers of different scales from the architecture described above are used as feature maps for detection. To set positions in an image where an object may be located, so-called *priors* are used.

[10] defines priors as "precalculated, fixed boxes which collectively represent this universe of probable and approximate box predictions". For every feature map, a prior scale s relative to the feature map dimension is chosen. Also, different aspect ratios a are selected. Using the equations

$$w \cdot h = s^2 \qquad \text{and} \qquad \frac{w}{h} = a, \tag{1}$$

the width and height in terms of "cells" in the corresponding feature map is calculated. Using this parameters, priors of all aspect ratios are placed above every cell in every feature map. Additionally, another prior having a $1:1$ aspect ratio and a scale between the current and the next feature map prior scale is placed above every cell in every but the last feature map. Table 1 shows the set of 8732 priors used in [10]. An example of the placement of priors on a 5×5 feature map is shown in figure 5.

3.1.4 Convolutional Layers for Prediction

For every feature map used for object detection, two convolutional layers are created that predict both the location and class of objects in every prior.

Table 1: Prior parameters used in [10].

Feature Map Dimensions	**Prior Scale**	**Aspect Ratios**	**# Priors per Position**	**Total Number of Priors on this Feature Map**
38×38	0.1	1:1, 2:1, 1:2 + extra prior	4	5776
19×19	0.2	1:1, 2:1, 1:2, 3:1, 1:3 + extra prior	6	2166
10×10	0.375	1:1, 2:1, 1:2, 3:1, 1:3 + extra prior	6	600
5×5	0.55	1:1, 2:1, 1:2, 3:1, 1:3 + extra prior	6	150
3×3	0.725	1:1, 2:1, 1:2 + extra prior	4	36
1×1	0.9	1:1, 2:1, 1:2 + extra prior	4	4

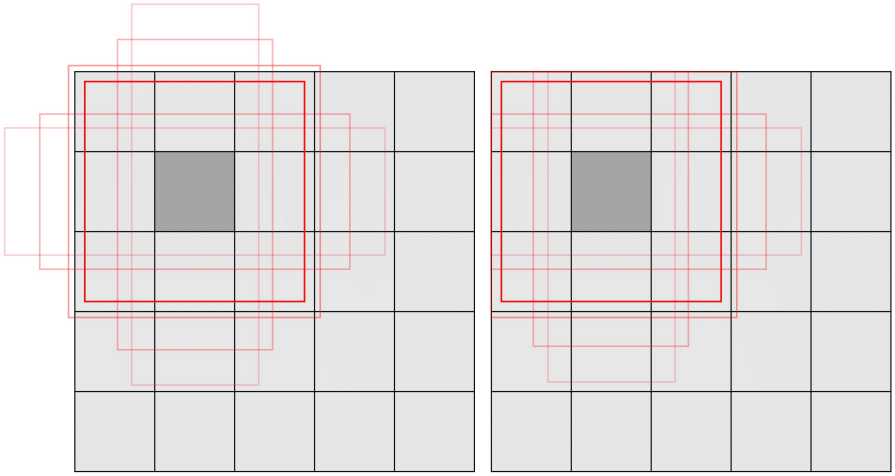


Figure 5: An example set of priors placed above one cell in a 5×5 feature map using the corresponding parameters listed in table 1. When priors reach beyond the feature map, they are clipped. [10, edited]

The exact location of an object with respect to the location of the prior is predicted by a localization prediction layer. Four filters per prior are trained to learn the translational and size deviations in both vertical and horizontal direction between the prior and the object.

The class of an object is predicted by a class prediction layer with n filters, if n represents the number of classes. The output of every filter is a score related to a certain object class and indicates how similar the object in the prior is to objects of the class.

Both types of prediction layers work with a 3×3 kernel and a stride of 1.

3.1.5 Multibox Loss

The loss function used in the SSD approach is called *Multibox Loss* and takes two types of losses into account; the bounding box localization loss and the class score confidence loss.

We start by matching priors to ground truth information. For that, the IoU[3], also called *Jaccard overlap*, of all priors and all ground truth objects is calculated. For every prior, we now determine the ground truth object with which the prior has the greatest Jaccard overlap, but only considering Jaccard overlaps greater than 0.5. We call priors now matching with a ground truth object *positive matches*, while priors with no such match are *negative matches*.

Localization Loss The localization loss is irrelevant for the negative matches that contain only image background, where no ground truth bounding boxes are available. For that reason, the localization loss is computed using only the positive matches. Here, the averaged *Smooth L1 loss* is used to compute the loss:

$$L_{\text{localization}} = \frac{1}{n_{\text{positive matches}}} \cdot \left(\sum_{\text{positive matches}} \text{SmoothL1}\left(\text{prediction box}, \text{ground truth box}\right) \right) \tag{2}$$

Confidence Loss For the confidence loss, all priors can be considered, since a class label can be assigned to every prior by introducing a background class for all negative matches. The *Cross Entropy Loss* is used for loss calculation. However, since thousands of priors are placed above an image and typically only few objects are shown in an image, most priors will be negative matches. Considering all priors for determining the confidence loss would result in the model mostly learning to detect the background class, which is not our intention. Instead, we choose only some negative matches to include in the confidence loss calculation. This procedure is called *Hard Negative Mining*. The ratio between negative and positive matches is set to $3:1$ and only the negative matches which lead to the highest confidence loss are included in the loss calculation. In other words, only the negative matches where "the model was most wrong

[3]Intersection over Union; the ratio between the area where two rectangles overlap (intersection) and the area covered by at least one of the rectangles (union).

about" [10] and "found it hardest to recognize that there are no objects" [10] are chosen. The team from [6] found that this led to "faster optimization and a more stable training".

In conclusion, the confidence loss can be described using the equation

$$L_{\text{confidence}} = \frac{1}{n_{\text{positive matches}}} \cdot \left(\sum_{\substack{\text{positive + hard} \\ \text{negative matches}}} \text{CELoss}\left(\text{prediction scores}, \text{ground truth scores}\right) \right). \quad (3)$$

Total Loss The total loss can be computed by simply adding both loss components together using a weight α:

$$L = L_{\text{localization}} + \alpha \cdot L_{\text{confidence}}, \quad (4)$$

where α was set to 1.

3.1.6 Predictions

After the model has been (pre-[4])trained using the methods described above, it can predict the class and location of objects in images. For that, the image is propagated through the network. The prediction layers now propose class scores and object locations for each prior. By considering only the detections where a score above a certain threshold value is computed for an object class, a first set of candidates for detected objects is obtained.

Non-Maximum Suppression However, since many priors overlap each other, most objects are detected multiple times by more than one prior. To determine the actual number of objects in an image, redundant detections must be removed. This is done using the concept of *Non-Maximum Suppression*.

For every class, the Jaccard overlap of all possible pairs of detections of this class is computed. Wherever the Jaccard overlap is above a certain threshold—0.5 was used in [10]—it is assumed that the same object was detected. Consequently, the detection with the lower score is removed so that only the more likely candidate is kept. If neccessary, this procedure is repeated with the remaining predictions until only non-overlapping predictions of the same class remain as final predictions.

3.1.7 Data Augmentation

Objects can be captured in very different ways; from different distances, angles, or perspectives, during different lighting conditions or using different capture devices. To assist the model in detecting the same objects under different conditions, a number of image transformations is randomly applied to some of the training images.

Photometric distortions With a probability of 50 percent each, the contrast, brightness, hue, and saturation are adjusted. Contrast, brightness, and saturation are, if adjusted, either increased or decreased by 50 percent. If the hue is adjusted, it is shifted by an amount of $\frac{18}{255}$ in a random direction. This adjustments help compensating different lighting conditions.

Zoom out With a probability of 50 percent, the image is shrinked by a random factor between 1 and 4. The surrounding area is filled with the mean pixel value from the ImageNet dataset. This adjustment helps detecting small objects.

Crop image With a probability of 87.5 percent, the image is cropped using a random crop factor between 1 and 3 and a random aspect ratio between 0.5 and 2. During cropping, it is ensured that at least one ground truth bounding box remains at least partly within the cropped area, where the Jaccard overlap of the bounding box and the cropped area must exceed a randomly chosen threshold. This adjustment helps detecting large objects.

Flip image With a probability of 50 percent, the image is flipped horizontally. This adjustment helps detecting objects captured from different perspectives or in different poses.

[4] Pretrained VGG-16 layers are used that have been trained on the ImageNet dataset [8].

Transformations to all images All images, including test images, are cropped to 300×300 pixels, since this is the input image dimension for the detector. After that, all images are normalized using the mean and standard deviation of the ImageNet dataset.

3.2 Changes to the original detector

The SSD detectors in [6, 10] are designed to detect objects in datasets like VOC2007 [1] or VOC2012 [2], which contain images of everyday objects, animals or people. Detecting traffic signs is in many ways different from the task the aforementioned SSD detectors are supposed to solve. Henceforth, the SSD implementation from [10] was changed in several ways to be best suited to detect traffic signs.

3.2.1 Changes to Data Augmentation

Typically, street signs are rather far away from a camera built into a car. While the dimensions of the images in the dataset are 1360×800 pixels, the mean height and width of a street sign bounding box is 43.2 pixels, as shown in figure 3a, or approximately 4 percent of the image height or width. A further zooming out of images while data augmentation, as described in section 3.1.7, would result in many street signs being too small for learning or detection. Hence, no zooming out is performed in our image preprocessing.

Since an object from the VOC dataset, e.g. a car, shall still be detected as a car regardless of the side from which it is photographed, data augmentation randomly flips an image horizontally to enhance the recognition of cars photographed from the left and right side. However, traffic signs have only one relevant side: the front side showing the symbols relevant to traffic rules. In addition, traffic signs have different meanings when its symbol is horizontally flipped. A very simple example would be the "turn left sign"; if its symbol was flipped horizontally, it would show the "turn right" sign having a very different traffic rule associated with it. For that reason, no horizontal flipping is performed in our image preprocessing.

3.2.2 Changes to the detector architecture

As explained in section 3.2.1, traffic signs within the GTSDB dataset typically cover only small areas. When images are scaled down to a size of 300×300 pixels in the detector, many street signs would become too small for learning or detection. For that reason, all layers' spatial dimensions in the model were scaled up by a factor of 2, resulting in an input image size of 600×600 pixels.

For the same reason, a new set of smaller priors was placed above another feature map. While in the original model the first feature map used for detection was `conv4_3` with dimensions of 38×38 pixels together with a prior scale of 0.1, we also use the layer `conv3_3` with new[5] dimensions 150×150 pixels and a prior scale of 0.04. The scale of 0.04 was chosen here because the mean height and width of a street sign bounding box is approximately 4 percent of the image height or width.

Because of the typically small size of traffic signs in the dataset images, there is no need for large priors. Therefore, all priors with scales of 0.55 and higher were removed completely from the network, together with the associated auxiliary convolutional and prediction layers.

From figure 3b, we can see that most traffic sign bounding boxes have an aspect ratio near $1:1$. It is for that reason that we chose the set of prior aspect ratios of all feature maps to be $\{0.5, 0.6, 0.7\}$, since the aspect ratio of the bounding boxes after scaling the images down from 1360×800 pixels (aspect ratio: 1.7) to 600×600 pixels is $\frac{1}{1.7} \approx 0.6$. The extra prior with a scale between the current and the next feature map prior scale is still part of our network, but with an aspect ratio of 0.6 instead of $1:1$.

The architecture of our final model is displayed in figure 6, while our set of 119720 priors is listed in table 2.

3.2.3 Extension of Non-Maximum Suppression

As described in section 3.1.6, non-maximum suppression prevents the same traffic sign being detected by multiple priors. However, this only applies to situations where different priors propose the same class for the traffic sign. Priors proposing different classes for the same sign in the image result in multiple signs being detected at the same location in the image. This situation occured frequently, since some traffic signs are very similar to each other and differ only in small details. The "speed limit (120)" and "speed limit (100)" signs serve as a good example here.

[5] The term "new" here refers to the upscaling process explained in the preceding paragraph.

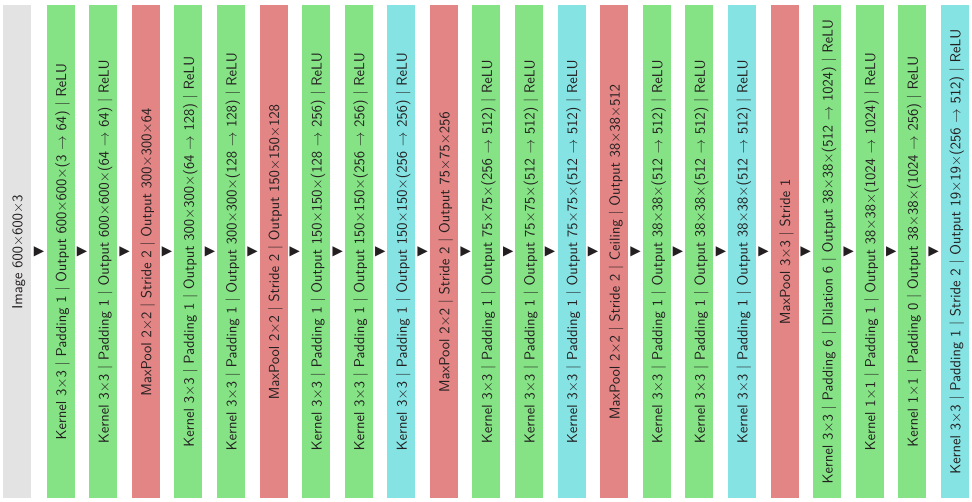


Figure 6: The layer architecture of our model. Layers marked in blue were used for detection.

To solve this problem, non-maximum suppression was extended in a way that no two bounding boxes must exist in the image that overlap each other with a Jaccard overlap above a certain threshold, regardless of their corresponding class.

4 Evaluation

For evaluation, the model was trained on the GTSDB training set with a batch size of 4 for 150 epochs with a learning rate of $3 \cdot 10^{-4}$, then for 50 epochs with a learning rate of $1 \cdot 10^{-4}$ and for another 50 epochs with a learning rate of $3 \cdot 10^{-5}$. Training took approximately one minute per epoch on a NVIDIA GeForce GTX 1080 Ti.

For evaluation, precision and recall were computed on the GTSDB test set using the equations

$$\text{Precision} = \frac{\#\text{ true positives}}{\#\text{ true positives} + \#\text{ false positives}} \tag{5}$$

and

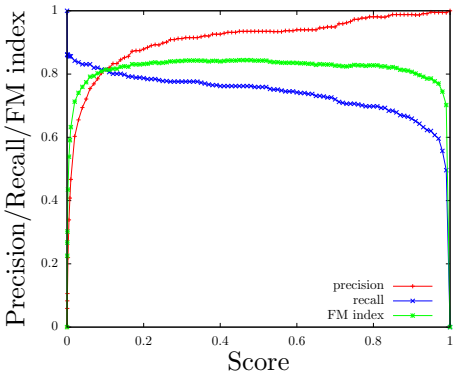
$$\text{Recall} = \frac{\#\text{ true positives}}{\#\text{ true positives} + \#\text{ false negatives}}. \tag{6}$$

Hereby, a detection result was considered only if its class score was above a certain threshold value. By changing this threshold, different value combinations of precision and recall were obtained to create a precision-recall plot.

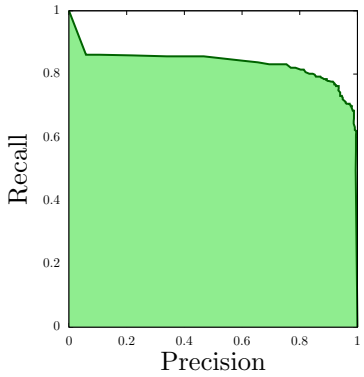
A detection result was counted as true positive if it overlaps with a ground truth bounding box of the same class with a Jaccard overlap of at least 0.3. If there is no such ground truth bounding box, the result was counted as false positive. After all results are processed, the ground truth bounding boxes that were not matched with detection results are counted as false negatives.

Table 2: Prior parameters used in our model.

Feature Map Dimensions	**Prior Scale**	**Aspect Ratios**	**# Priors per Position**	**Total Number of Priors on this Feature Map**
150×150	0.04	0.5, 0.6, 0.7 + extra prior	4	90000
75×75	0.1	0.5, 0.6, 0.7 + extra prior	4	22500
38×38	0.2	0.5, 0.6, 0.7 + extra prior	4	5776
19×19	0.375	0.5, 0.6, 0.7 + extra prior	4	1444



(a) Plot of precision, recall, and Fowlkes-Mallows index against the score threshold.



(b) Precision-recall plot. The area under the curve amounts to 83.46 percent.

Figure 7: Plots related to precision and recall.

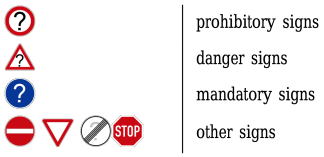


Figure 8: Traffic sign categories relevant for the competition. [4]

5 Results

During evaluation, around 10 images were processed per second. The precision and recall values obtained by using different thresholds during evaluation are visualized in figure 7.

To find a well-balanced combination of precision and recall, the Fowlkes-Mallows index

$$FM = \sqrt{\text{Precision} \cdot \text{Recall}} \tag{7}$$

is used, which reaches its highest value at a threshold of 0.45, where the precision is 93.5 percent and the recall is 76.2 percent[6]. The area under the precision-recall curve amounts to 83.46 percent, i.e., our detector produces results better than 66 percent of all competition submissions published in [5], and better than all baseline algorithms proposed in [4], apart from the Viola-Jones algorithm detecting prohibitive traffic signs. However, the task during the competition was to only detect traffic signs from one of the categories “prohibitory”, “mandatory” or “danger”. Within this categories, the signs look quite similar, as can be seen in figure 8. Therefore, comparability between our and the published results is somewhat limited.

The results for every class of traffic signs can be found in table 3. One may note that for some classes, an evaluation is hardly possible because there are no signs of this class in the test set.

All images from the test set were annotated by the detector, using the optimal score threshold of 0.45. Some examples can be found in figure 9, whereas the full annotated test set is available at [7].

5.1 Weaknesses

Overall, it can be said that with only 600 training images containing 846 traffic signs the dataset is quite small. In total, 19 classes contain less than 10 training traffic signs and 9 classes contain less than 5 training traffic signs, making it hard for the detector to learn the characteristics of these traffic signs and leading to false positive detections, as shown in figure 10a.

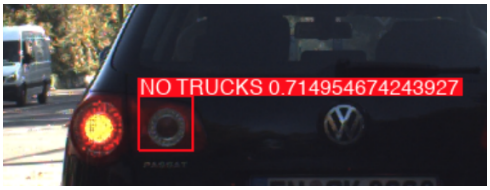
Another problem were traffic signs that appeared in the test set but not in the training set. A good example is the bus stop sign, shown in figure 10b. The detector recognized the general shape of a traffic sign and thus matched the sign with the class fitting best, but failed to name the correct class, since it was unknown to the detector.

[6]During the project presentation, a recall of 86 percent was mentioned. This value was erroneous due to a bug in the evaluation script.

Table 3: Precision and recall for all traffic sign classes at a threshold of 0.45.
* = No detections
† = No signs in test set

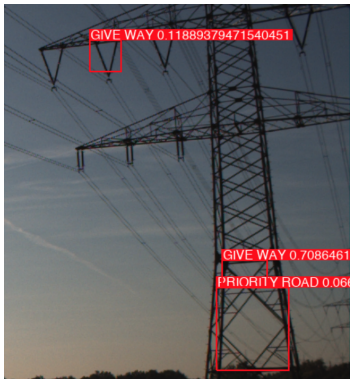
Class	**Frequency in Training Set**	**Frequency in Test Set**	**Precision**	**Recall**
no overtaking (trucks)	63	17	0.923	0.706
speed limit (50)	59	22	0.857	0.818
keep right	57	31	1	0.839
priority road	54	31	1	0.871
give way	52	31	0.931	0.871
speed limit (30)	48	31	1	0.71
speed limit (120)	47	10	0.875	0.7
speed limit (100)	37	4	1	1
speed limit (80)	37	16	0.857	0.75
no overtaking	32	10	1	0.889
speed limit (70)	31	37	0.958	0.622
danger	27	11	0.8	0.727
priority at next intersection	26	12	1	1
no entry	25	4	1	1
stop	22	10	1	0.8
construction	21	10	0.636	0.7
speed limit (60)	21	9	1	0.778
restriction ends 80	17	2	1	1
go straight	15	5	1	0.8
snow	14	2	1	1
go right	13	3	1	1
slippery road	13	7	1	1
traffic signal	11	7	1	0.714
no traffic both ways	10	5	1	0.8
bend right	9	0	*	†
go left	9	3	1	0.667
school crossing	9	5	0.75	0.6
uneven road	9	4	1	0.5
go right or straight	8	1	0	0
no trucks	7	1	1	1
restriction ends (overtaking (trucks))	7	4	1	0.75
roundabout	7	3	0	0
restriction ends (overtaking)	6	1	1	1
bend	5	0	*	†
cycles crossing	4	1	*	0
keep left	4	2	*	0
speed limit (20)	4	0	*	†
pedestrian crossing	3	0	*	†
restriction ends	3	5	1	0.4
bend left	2	0	*	†
road narrows	2	3	1	0.333
animals	1	1	*	0
go left or straight	1	1	1	1

Figure 9: Some images from the dataset, annotated by the detector. All annotated test images can be found at [7].



(a) A car backlight misdetected as a “no trucks” sign.

(b) A bus stop sign misdetected as a “go right” sign.



(c) An electricity pole misdetected as several “give way” and “priority road” signs.

Figure 10: Images from the test set of [4] showing common weaknesses of the detector.

When some traffic signs have a unique shape, such as the "priority road" or the "give way" sign, the detector "paid more attention" to the shape of the signs and "less attention" to other aspects like hue or saturation. The results can be seen in figure 10c, where objects that are not traffic signs but having a similar shape were falsely detected.

6 Conclusion

The SSD model presented in this report detected traffic signs from the GTSDB dataset fastly and accurately. A detection throughput of 10 images per second was achieved, which should be enough for real-time applications in a car.

The area under the precision-recall curve reached 83.46 percent, which is better than two thirds of the results proposed by other research groups working on the same dataset, however with a slightly different task. At the best precision-recall ratio, determined using the Fowlkes-Mallows index, the detector reached a precision of 93.5 percent and a recall of 76.2 percent.

Many weaknesses of the detector resulted from the small number of training or test images and some traffic sign classes not being represented at all in one of the image sets. Training and evaluating the detector on more images would potentially improve the detection results by a considerable amount. The amount of data could possibly also be increased using techniques of data augmentation.

References

[1] Mark Everingham et al. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.* http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[2] Mark Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.* http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[3] Sebastian Houben et al. *About. German Traffic Sign Benchmarks.* May 2019. URL: `http://benchmark.ini.rub.de/?section=gtsdb&subsection=about` (visited on 08/05/2020).

[4] Sebastian Houben et al. "Detection of traffic signs in real-world images: The German traffic sign detection benchmark". In: *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4–9, 2013.* IEEE, 2013, pp. 1–8. DOI: `10.1109/IJCNN.2013.6706807`.

[5] Sebastian Houben et al. *Precision-Recall plots for the 10 highest ranked results. German Traffic Sign Benchmarks.* May 2019. URL: `http://benchmark.ini.rub.de/?section=gtsdb&subsection=results&subsubsection=all` (visited on 08/06/2020).

[6] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016 – 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I.* Ed. by Bastian Leibe et al. Vol. 9905. Lecture Notes in Computer Science. Springer, 2016, pp. 21–37. DOI: `10.1007/978-3-319-46448-0_2`.

[7] Marvin A. Ruder. *Detection of Traffic Signs using a Single Shot Detector.* Aug. 2020. URL: `https://github.com/marvinruder/ssd` (visited on 08/12/2020).

[8] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252. DOI: `10.1007/s11263-015-0816-y`.

[9] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: `http://arxiv.org/abs/1409.1556`.

[10] Sagar Vinodababu and Micah Seneshen. *SSD: Single Shot MultiBox Detector.* A PyTorch Tutorial to Object Detection. 2019. URL: `https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection` (visited on 07/20/2020).

A Code

Code is are available at [7].

B Acknowledgements

Portions of the software documented in this report utilize the following copyrighted material, the use of which is hereby acknowledged.

B.1 SSD: Single Shot MultiBox Detector — a PyTorch Tutorial to Object Detection