Roadster

Traffic Sign Recognition System

Victor Pădurean

March 2020

Contents

1 Introduction

The theme of autonomous driving vehicles is becoming a concern for a larger and larger part of the industry. It goes without saying that the problem of autonomous automobiles is the most addressed, but some other types of vehicles benefit of this improvements, like drones.

There are a lot of ideas and adaptations that are still needed for a completely autonomous, problem free vehicle, but in the meantime, some of our attention may be shifted from creating a fully autonomous automobile to the assistance of the driver. There is some pressure that can be lifted off the shoulder of the driver with the appropriate support, this contributing greatly to the driver's overall health and good disposition.

One of this driver helping functionalities is represented by the traffic sign recognition and detection. This may help the driver in keeping track of them, this being a crucial part involved in the driving process. The idea can be extended nicely in many different directions, this representing only its basis. The Roadster project will present a method for road signs detection and classification. Its main objectives are, firstly, to recognize the meaning of the sign present in different images; secondly, to develop a method for traffic sign detection in a static image, or, preferably, in a video; lastly, to integrate both parts and provide real time traffic sign detection and identification.

2 Related Work

As mentioned before, the problem can be divided into two sub-problems: traffic sign detection and traffic sign classification. [13] presents an overview of both detection and recognition systems.

2.1 Detection

The first method involves detection based on color, inside the RGB space [7], by thresholding. The corners are detected using a convolution of two functions: one which detects the gray levels around the corner and one which is called a "corner detector". Then the algorithm selects the points above the given threshold, finally, calculating the center of mass. If the sign is round, an approximation for rectangular sign detection algorithms is used.

In [16], the HSV space is used, so that light doesn't contribute to affecting color detection. An interesting approach is using the YUV color space, based on separation of luminance and and chrominance, in [12]. The actual threshold are determined by analysing the distribution of data in the YUV color space. In [11], High-Contrast Region Extraction is used, in order to extract regions of

interest with high local contrast.

Several articles present methods based on shape recognition, as color based recognition may have many different flaws, due to change in light and distance. Some use Hough transform for detecting the closed contours of the traffic signs, yet this approach is sensitive to noise and occlusions. This method is used in [9].

An interesting approach, presented in [10], makes use of Fourier Descriptors (FDs). The main steps described are: extraction of FDs, matching of FDs, and matching of previously acquired prototypes with spatial models. The FD of a contour is obtained by applying the Fourier transform repeatedly, resulting in a shape descriptor in the frequency domain.

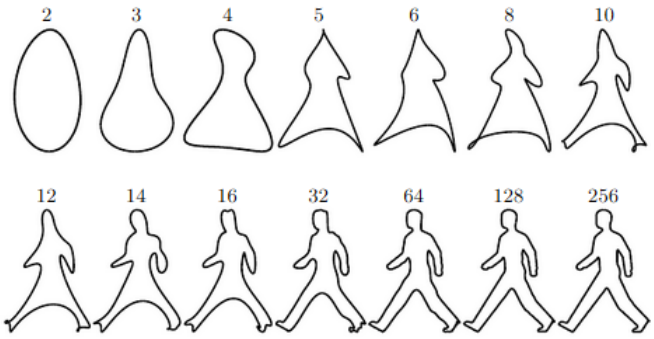

Figure 1: Representation of a pedestrian crossing sign using increasing number of coefficients, taken from [10]

The approach is useful due to its behaviour under different geometric transformations. They may be easily detected and eliminated. The mathematical apparatus needed for matching is then described, as it will be needed for matching the detected contours onto previously computed, prototype contours.

More and more articles develop learning based solutions, due to the recent popularity in machine learning, yet this require large amounts of data for training the models. The Viola-Jones algorithm is used in [6] for triangular, warning sign detection. The Viola-Jones algorithm uses increasing complexity detectors, each one being a set of classifiers based on Haar wavelet, AdaBoost being used as learning algorithm. This particular detector was trained on over 800 images, and the resulting detector consists of 17 cascade stages and uses a total of 299 features. The detector works by sliding a detection window across the image. When the window reaches the end of the image, it is enlarged and the processes is repeated.

Finally, method used in [8] for traffic sign detection is worth mentioning.

Non relevant areas are cropped if no interesting colors of the focus zone do not appear in other regions. Then, a concept called chromatic density value is defined for obtaining bounding boxes faster, with color information.

2.2 Classification Methods

There are several methods for classifying traffic signs, but the majority are based on learning algorithms. A more difficult approach, more appropriate for experts, yet with limited representation power is based on hand-crafted features. For example, [17] used histogram of oriented gradients descriptors and distance transforms to evaluate the performance of K-d trees and random forests.

In [8] Haar-like features are used for classifying images. It defines a function for mapping the features in images to sets of possible traffic signs. An alternative is given, so that the function is replaced by AdaBoost, working with Haar-wavelets.

Deep learning algorithms enable a more precise classification, less dependent on the domain knowledge. In [14], an architecture for a CNN is presented, starting from a traditional ConvNet. Multi-Scale features are presented: as opposed to a sequentially layered architecture, where output is fed only to succeeding layers, the given method is branched and fed to multiple stages. Several Non-Linearities are proposed and discussed. The data used for training the models was the popular German Traffic Sign Recognition Benchmarks. Several architectures were tested and the best one resulted in a performance of 98.97% accuracy, better than that of human performance (98.81%). The best architecture didn't use color information.

3 Proposed Solution

Regarding implementation, several methods will be combined for detecting the traffic signs in a video, whereas classification will be done using a CNN.

3.1 Solution Description

Detection is inspired by [4], with a few improvements. The image is first preprocessed, so that the signs can be detected more easily, in a geometric manner. Preprocessing consists of enhancing the contrast, so that colors can be recognized more easily; filtering the image in such a manner that only colors that may represent a traffic sign remain, the rest being faded to black; applying the canny edge detection algorithm; and finally binarizing the image. After the image was preprocessed, the small components are removed, all the external contours are found and the sign having the greatest distance from its center to the contour is extracted from the image. Finally, a square image is cropped, based on the sign's contour.

Classification is done using CNNs, which implement a voting method. Each CNN has a different architecture, so that they behave differently, otherwise voting will have no importance. For the models' training is done on the "The German Traffic Sign Recognition Benchmark" dataset ([2]).

Reconciliation must finally be obtained between the two parts which were largely developed separately. The output of the detection algorithm, which will be the input of the CNN, must resemble the images the CNN was trained on. That way, the accuracy should be as high as possible. When classification works satisfyingly enough for each model separately, voting should be implemented.

3.2 Implementation Flow

The implementation process has been following and will follow the diagram listed in Figure 2. It presents the main steps. It can clearly be seen that the process is divided into two parts: one regarding classification and one regarding detection.

3.3 Algorithm Description

3.3.1 Detection

The majority of the algorithms are implemented by default in the cv2 library in Python.

Contrast The image is first converted to YCrCb colorspace, consisting of Y - luminance or Luma, Cr - how far Luma is from the red component and Cb - how far Luma is from the blue component. Then the histogram of the Luma is equalized, spreading out the values from any cluster, and finally, the image is converted back to RGB.

Color Filtering The image is converted to HSV color space at first. Then 5 masks, meaning images where only black and a color ranging between some given values, are obtained. The masks correspond to red, white, blue and white. Two masks are obtained for the color red as, the HSV color space being a circular one, red's range is present on the upper, as well as on the lower boundary. Finally, the masks are combined.

Canny Edge Detection This step is done with the classical Canny algorithm, by first reducing the noise with a Gaussian filter. Then the intensity gradient is found for each pixel, all the pixels that are not local maximums being removed afterwards. Thresholding is then considered, based on two dynamic values, computed using the median of the image. Finally, only hard edges are obtained.

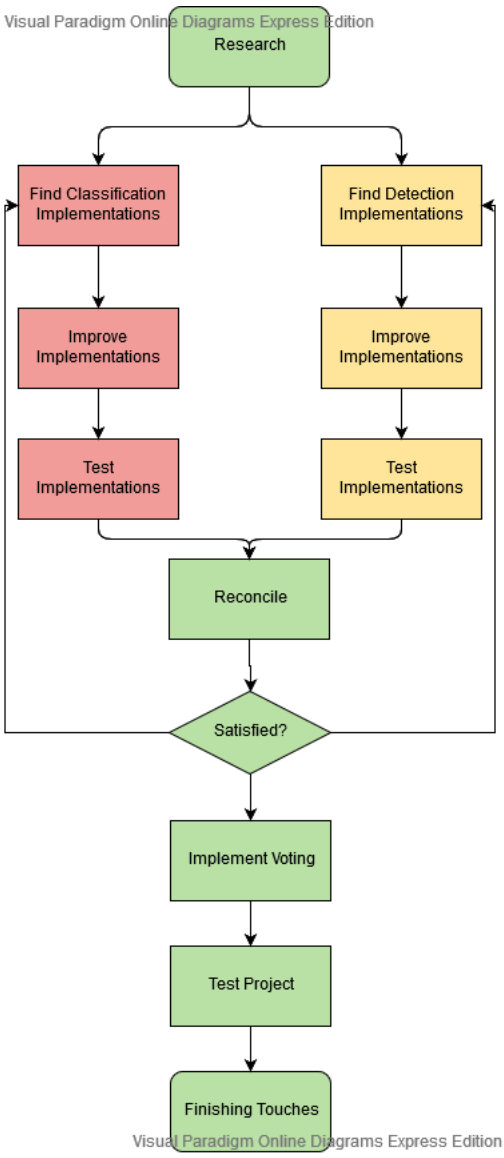

Figure 2: Roadster Implementation Flow

The final part of preprocessing is converting the image to binary, leaving only black and white. After this, all the small components are removed by finding if their sizes, found through analyzing connected components, are below a given threshold.

Contour Handling Contours can be found easily on the obtained images by using the cv2 function called *findContours()*, based on the algorithm described in [15]. To find out if the contour really is a sign, its signature is compared to that of a circle. The signature is found by dividing the distance from the center to each point of the contour to the maximum found distance, and avenging them. If the threshold to which the value is compared is high enough, it will work for triangles and diamonds too, as all the signs are symmetric. Only the largest signs is cropped out of the image and passed on to classification.

3.3.2 Classification

Various CNNs will be used for classification, but the dataset, training, validation and evaluation will be done in the same way, only the architecture of the neural network being different.

The training data was split into three parts: training, with the most numerous images; validation, used for computing the loss, a metric measuring how far the desired output is with respect to the actual output; and evaluation, which helps evaluate the trained model. When loading the images in memory, all of them are resized to 32x32 pixels, then local details are enhanced using an adaptive histogram and finally scaled between 0 and 1. The classes are weighted, as the numbers of images differ dramatically from one class to another. Weighting the classes prevents the model from certain classes taking over the probabilities. Augmenting the images by applying transformations like zooming, rotation, shifting and shearing prevents overfitting.

Germansignsnet1 was the first try of obtaining such a traffic signs classifying model and may be used as a standard for the ones that will come. First, it used a pooling layer stacked on top of a convolutional layer. Then it uses two sets of layers, each made out of a pooling layer stacked on two convolutional layers. The head of the model is composesd of two fully connected layers and a softmax (a function that takes as input a vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers) classifier. Through the final layers, there are some dropout one that work as a form of regularization which aims to prevent overfitting, the result often being a more generalizable model. The architecture is inspired by [3]. Categorical cross-entropy is used for this model as a loss function. The learning rate is low at the beginning, but the Adam optimizer adapts it on the run.

Besides the softmax function, the sigmoid function could also be used; and regarding the loss function, binary cross-entropy is also a choice.

3.3.3 Voting

When trying to classify an input image, more CNN models will be used, each of them voting three labels, based on prediction accuracy. In order for the label to be voted, the accuracy should be greater than a given threshold, by default 90%. The label with the greatest sum is the winner, if there exists one. The algorithm is described below, in Python.

```
1
2  def vote_on_image(models, image):
3      voting_dict = {}
4      for voter in models:
5          preds = voter.predict(image)
6          top = argsort(-preds, axis=1)
7          for i, vote in enumerate(top[:3]):
8              if preds[vote] > 0.9:
9                  if vote not in voting_dict:
10                     voting_dict[vote] = preds[vote]
11                 else:
12                     voting_dict[vote] += preds[vote]
13
14     winner = None
15
16     if len(voting_dict) > 0:
17         winner = max(voting_dict, key=voting_dict.get)
18
19     return winner
```

Listing 1: Models Voting

4 Results

4.1 Detection

Detection was done as explained in 3 Proposed Solution. It works well most of the time, but it can become faulty, as it depends on illumination and the whole scene. There are cases when the algorithm doesn't detect any sign due to it being oddly illuminated. In other cases, the algorithm prefers to yield a different object to the detriment of a sign, due to it begin closer and having the right colors.

Figure 3 presents visually the way an image is processed, including contrasting, masking, canny-edge detection and small parts removal.

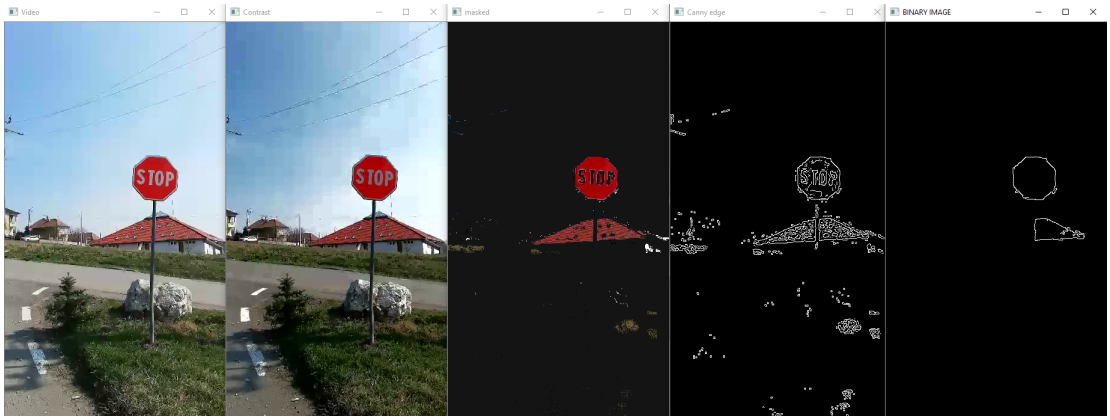

Figure 3: Preprocessing Pipeline and small part removal

Next, Figure 4 shows the detected contour and the cropped sign image, which will be fed to the models.

Figure 4: Contour detection and cropping

Figure 5, on the other hand, displays an erroneous course the pipeline may take, due to illumination.

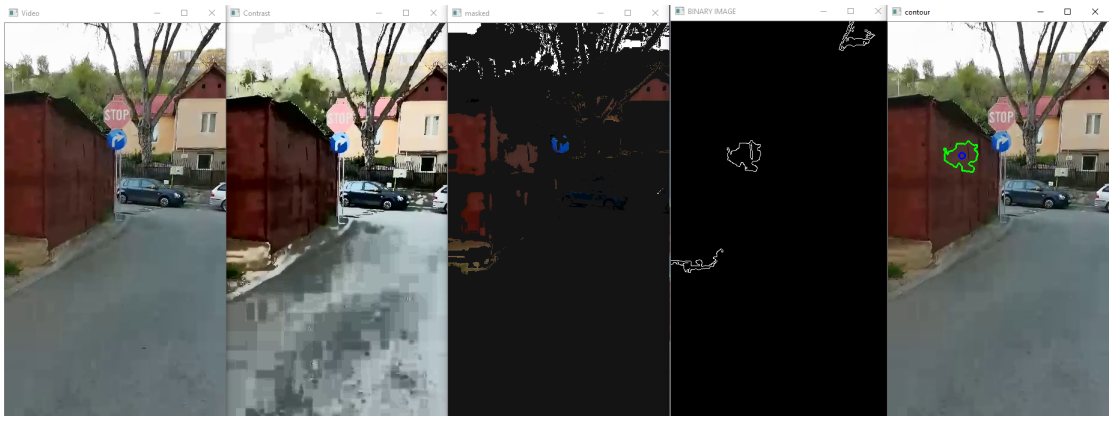

Figure 5: Erroneous detection

This erroneous findings, however, will be filtered in most cases by the classifiers, obtaining low likeliness and not being voted.

4.2 Classification

Three main architectures were used: germansignsnet1.x inspired by [3]; germansignsnet3.x inspired by [5]; and germansignsnet4.x inspired by [1]. They have gone through various improvements, modifications and training runs, but in this section only the latest models' results will be presented.

Germansignsnet1.8 has mostly remained the same as Germansignsnet1, but the last softmax layer was replaced by a sigmoid layer. Loss was calculated using binary cross-entropy. Graphs presenting loss and accuracy evolution are rendered below.

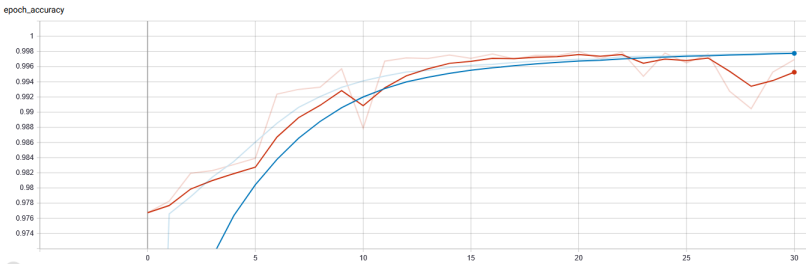

Figure 6: 1.8 Accuracy on epochs

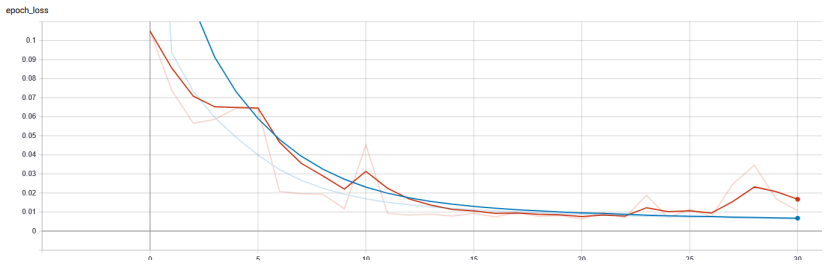

Figure 7: 1.8 Loss on epochs

Next, barcharts regarding evaluation data are presented.

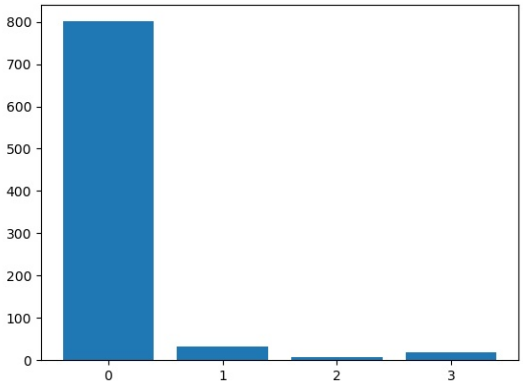

Figure 8: 1.8 Evaluation total statistics (0 - recognized first choice; 1 - second choice; 2 - third choice; 3 - others [unrecognized])

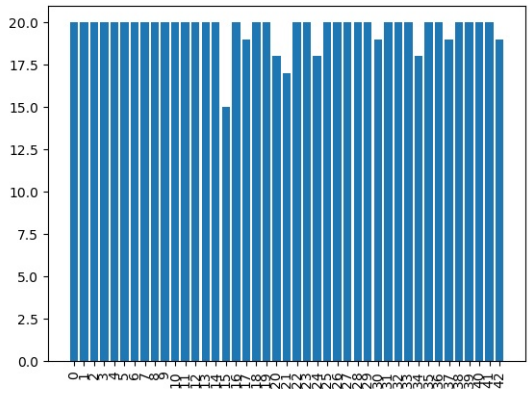

Figure 9: 1.8 Evaluation statistics per class - number of successfully recognized signs

Germansignsnet3.7 hasn's changed much as well, the most important and useful modification being the change of softmax to sigmoid and categorical cross-entropy to binary cross-entropy.

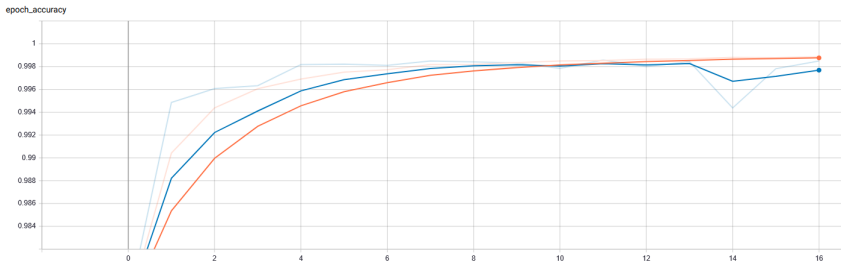

Figure 10: 3.7 Accuracy on epochs

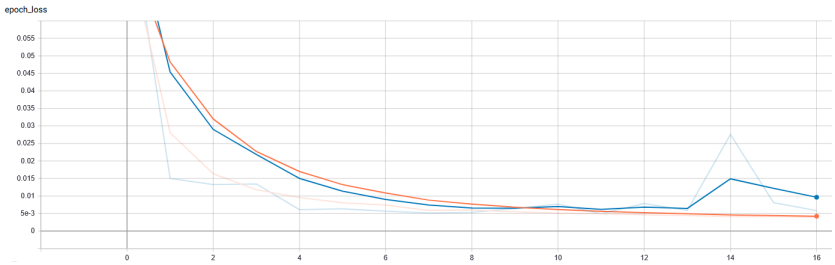

Figure 11: 3.7 Loss on epochs

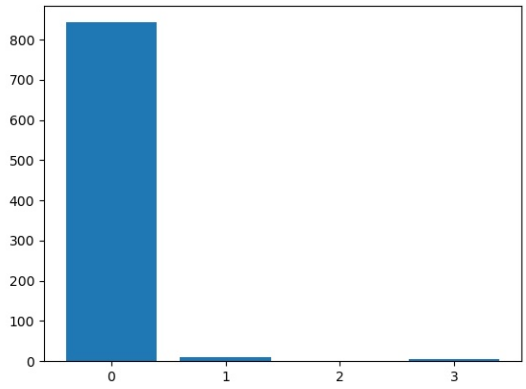

Figure 12: 3.7 Evaluation total statistics (0 - recognized first choice; 1 - second choice; 2 - third choice; 3 - others [unrecognized])

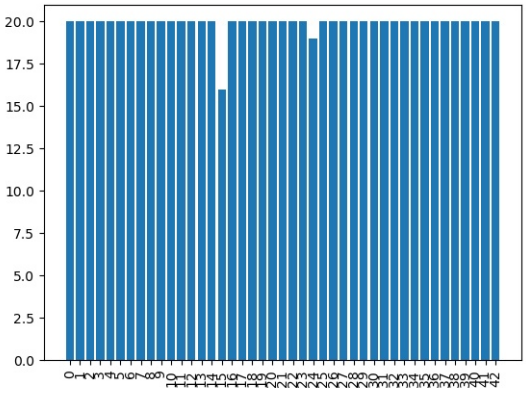

Figure 13: 3.7 Evaluation statistics per class - number of successfully recognized signs

Germansignsnet4.4 , on the other hand, has suffered several changes from the original. It uses, like the other two, sigmoid and binary cross-entropy, but it has some additional layers added to the original architecture. A batch normalization layer was added right after the first relu activation and dropout layers were added before each dense layer. Additionally, instead of the 16-sized filter used in the original as the second convolutional layer, this model used 66 filters, and seems to work better.

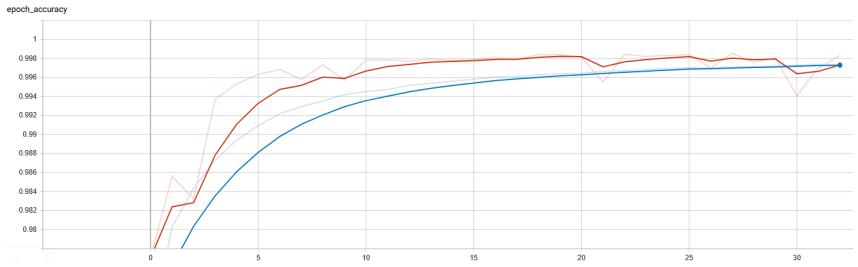

Figure 14: 4.4 Accuracy on epochs

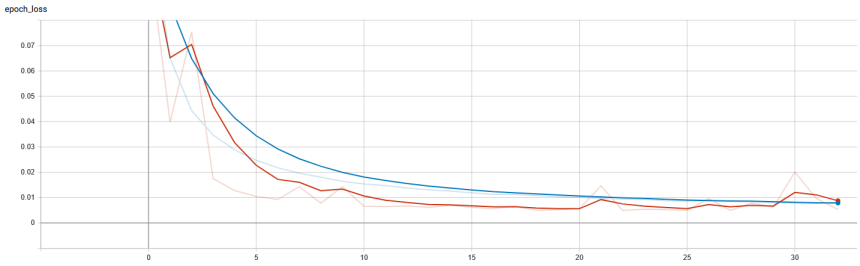

Figure 15: 4.4 Loss on epochs

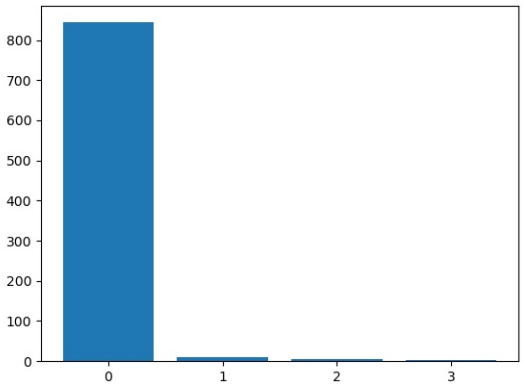

Figure 16: 4.4 Evaluation total statistics (0 - recognized first choice; 1 - second choice; 2 - third choice; 3 - others [unrecognized])

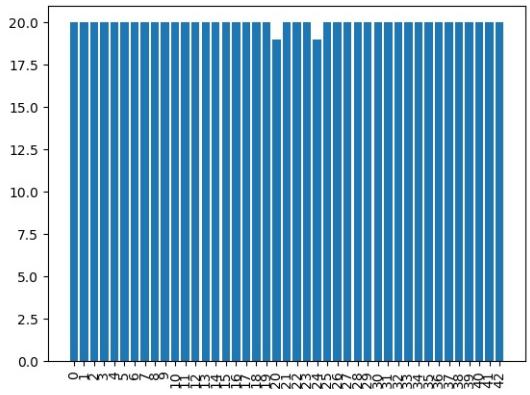

Figure 17: 4.4 Evaluation statistics per class - number of successfully recognized signs

4.3 Reconciliation

When put together, the two parts work well, as the majority of the erroneous data sent by the detector is rejected by the classifier. Voting certainly helps, as more models that behave differently can classify an image and come up with the best statistical result.

Night conditions with flash or headlight illumination seem to be the best. The combination of low ambient noise due to lack of ambient light and the high visibility of the sign due to the reflected light makes the detector's job, and, as a result, the classifier's job too, very easy.

Shade conditions make the project yield good results, as color is not denatured and the masks do not cover the signs. The classifier has to reject a moderate amount of noise, most commonly rocks, a very clear blue sky or blue cars, and some patches of grass.

Intense light conditions makes the detector skip some signs, their colors being denatured and, as a result, masked by the color mask. In that case, the classifier has no power, being able only to reject the noise it is fed with.

5 Conclusion

In this project, a traffic sign recognition system, divided into two parts, was presented. The first part is based on classical image processing techniques, for traffic signs extraction out of a video, whereas the second part is based

on machine learning, more explicitly, convolutional neural networks, for image labeling.

An improvement that can be brought to classification is a new class, consisting of different kinds of objects that are not traffic signs, so that the models may know how to handle noise better. This would be an alternative to using sigmoid.

Detection can be done using machine learning as well. Architectures like YOLO can detect and crop images from videos, as well as classify objects in those images, with outstanding speed.

References

[1] German traffic sign recognition. `https://github.com/hparik11/German-Traffic-Sign-Recognition`. Accessed: 2020-05-10.

[2] The german traffic sign recognition benchmark. `http://benchmark.ini.rub.de/?section=gtsrb&subsection=news`. Accessed: 2020-29-03.

[3] Traffic sign classification with keras and deep learning. `https://www.pyimagesearch.com/2019/11/04/traffic-sign-classification-with-keras-and-deep-learning`. Accessed: 2020-20-03.

[4] Traffic-sign-detection. `https://github.com/ghostbbbmt/Traffic-Sign-Detection`. Accessed: 2020-04-05.

[5] Traffic sign detection using convolutional neural network. `https://towardsdatascience.com/traffic-sign-detection-using-convolutional-neural-network-660fb32fe90e`. Accessed: 2020-05-10.

[6] Karla Brkić and Axel Pinz. Traffic sign detection as a component of an automated traffic infrastructure inventory system. 03 2020.

[7] Arturo de la Escalera, Luis Moreno, Miguel Salichs, and J.M. Armingol. Road traffic sign detection and classification. *Industrial Electronics, IEEE Transactions on*, 6:848 – 859, 12 1997.

[8] stéphane Estable, J. Schick, Fridtjof Stein, R. Janssen, R. Ott, W. Ritter, and Y.-J Zheng. A real-time traffic sign recognition system. pages 213 – 218, 11 1994.

[9] Miguel Garrido, Miguel-Angel Sotelo, and E. Martm-Gorostiza. Fast traffic sign detection and recognition under changing lighting conditions. pages 811 – 816, 10 2006.

[10] Fredrik Larsson and Michael Felsberg. Using fourier descriptors and spatial models for traffic sign recognition. pages 238–249, 05 2011.

[11] C. Liu, F. Chang, Z. Chen, and D. Liu. Fast traffic sign recognition via high-contrast region extraction and extended sparse representation. *IEEE Transactions on Intelligent Transportation Systems*, 17(1):79–92, Jan 2016.

[12] Jun Miura, Tsuyoshi Kanda, and Yoshiaki Shirai. An active vision system for on-line traffic sign recognition. volume E85d, pages 52 – 57, 02 2000.

[13] Yassmina Saadna and Ali Behloul. An overview of traffic sign detection and classification methods. *International Journal of Multimedia Information Retrieval*, 6:1–18, 06 2017.

[14] Pierre Sermanet and Yann Lecun. Traffic sign recognition with multi-scale convolutional networks. pages 2809 – 2813, 09 2011.

[15] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, apr 1985.

[16] Pavel Yakimov. Traffic signs detection using tracking with prediction. pages 454–467, 03 2016.

[17] Fatin Zaklouta, Bogdan Stanciulescu, and Omar Hamdoun. Traffic sign classification using k-d trees and random forests. pages 2151 – 2155, 09 2011.