Traffic Sign Recognition

Lukcs-Roland Elekes
Computer Science
Technical University of Cluj-Napoca
Cluj-Napoca, Romania

I. Introduction

Driver assistance systems became a very popular topic in the last years and many systems were developed that are capable of detecting danger or avoiding obstacles. The reason these systems exist is to make driving more comfortable and to improve safety.

Traffic sign recognition is an important part of these systems, since the information available on traffic sings are significant for the driver. Knowing speed limits or the rules imposed by the traffic signs are useful information for the driver making the act of driving more pleasent and improving safety on the road. Traffic sign recognition systems can recognize these and having a system the outputs the information obtained from the sings is a helpful tool for the driver.

The proposed objectives are to develop and implement a system that is capable of traffic sign recognition following the three essential steps summarized in [1]. Firstly the objective is recognizing traffic sings on static images and then developing the system to analyze videos and be a real-time traffic sign recognition system.

II. Related Work

The three steps described in [1] are the following:

1) Preprocessing: color segmentation or adjustment which includes video/image decoding and color space transformations.
2) Detection: selection of regions of interests (ROI) where traffic signs are present in the image
3) Recognition: identification of traffic signs.

For each step several methods are presented and summarized in [2].

A. Preprocessing

The image obtained is of RGB nature. These images or sensitive to lighting variations, for reducing the effect of light other color spaces are used. The most important color spaces are HIS, YC_bC_r, HSV, YUV and RGB.

The YUV color space give good results for red/yellow/orange signs, but it fails for white/black or low resolution colors. The HSV (Hue-Saturation-Value) color space has been used in many works. This color space is preferred as it is ineffective to illumination effects. The YC_bC_r color model was used on grayscale images.

B. Detection

The detection phase selects the ROIs from the image and send them to recognition. According to [2] there are different methods for detection through color, shape or machine learning based techniques.

1) Color-based Techniques:

- Color thresholding: consists of selecting areas where a specific amount of valid traffic sign colors appear. According to [1] [2] these techniques tend to be unreliable in urban areas and they are used for highways only.
- Color learning methods: Genetic Algorithms for detecting speed limit signs. The authors concluded that methods based on learning algorithms give better results that color segmentation.

2) Shape-based Techniques:

- Using edges: detection of signs by using rectangular pattern matching algorithm
- Using template matching: the captured image is matched with the images of known good traffic signs in the database (Hausdroff distance)
- Based on machine learning: machine learning algorithms like support vector machines (SVMs) and neural networks (NNs) can be used to extract shapes
- Based on Hough transform: Hough transform is used to detect circles and lines based on curve fitting algorithm

C. Recognition

The recognition stage receives the ROIs sent by the detection stage and classify the signs. There are mainly two approaches: template matching or machine learning methods.

- Based on template matching: sample images are stored in a database and the algorithms calculate the distance between candidate regions and template images, the template having the minimum distance is the matches traffic sign
- Based on machine learning: there are techniques like support vector machines (SVMs) and neural networks (NNs) which can be used to recognize signs.

III. Proposed Solution

The **preprocessing** phase consists of finding an appropriate color space, because using the picture as they are, in RGB nature, results are nearly impossible to reach. A good choice is to use the HSV color space, since it is simple to work with.

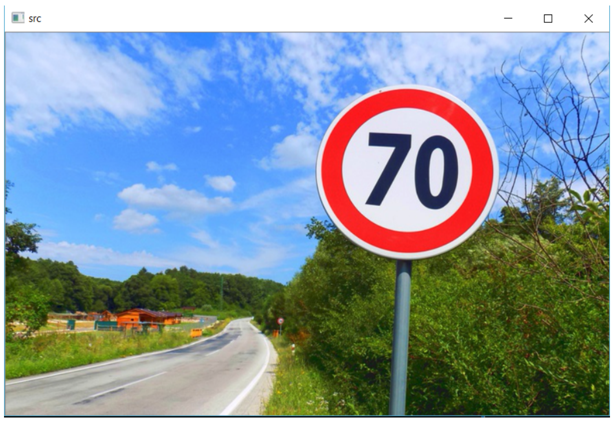
However, there are some disadvantages, they are sensible for distance, weather and age of traffic sign conditions.

HSV is closer to how humans perceive color. It has three components: hue, saturation, and value.

1) **Hue**: This channel encodes color color information. Hue can be thought of an angle where 0 degree corresponds to the red color, 120 degrees corresponds to the green color, and 240 degrees corresponds to the blue color.
2) **Saturation**: This channel encodes the intensity/purity of color. For example, pink is less saturated than red.
3) **Value**: This channel encodes the brightness of color. Shading and gloss components of an image appear in this channel.

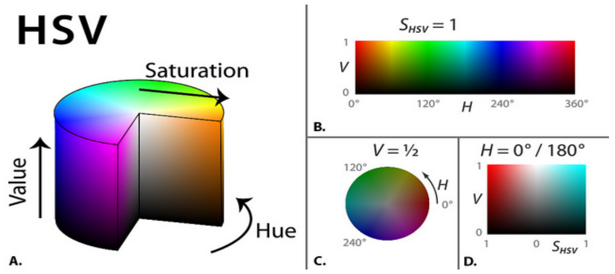
The second phase, the **detection** consists of finding the regions of interests (ROI). Color thresholding will be used, I will try to extract areas, where a specific amount of valid colors appear, that are relevant for traffic signs, an example is the color red. We know, that these methods are not efficient in areas where objects of many colors appear, because in case of buildings bricks can be very confusing, or people dressed in red or red cars. This method will mainly work on highways, were the surrounding environment consists mainly of trees, hills etc.

The test image I am working with is the following:



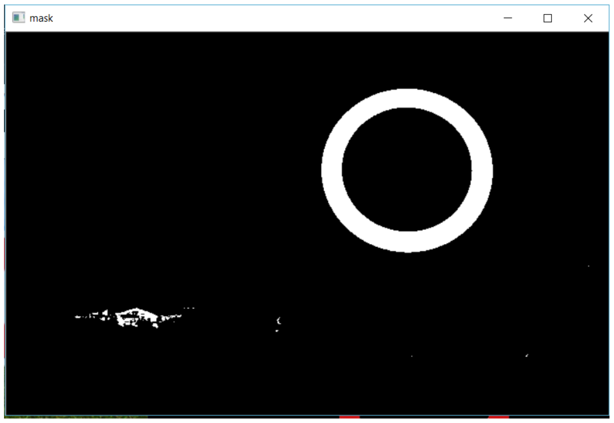
By creating masks based on the HSV values, we can find the colors that we want to extract from the picture.

The Hue values are actually distributed over a circle (range between 0-360 degrees) but in OpenCV to fit into 8bit value the range is from 0-180. The red color is represented by 0-30 as well as 150-180 values, therefore we can use two masks and finally we combine them.



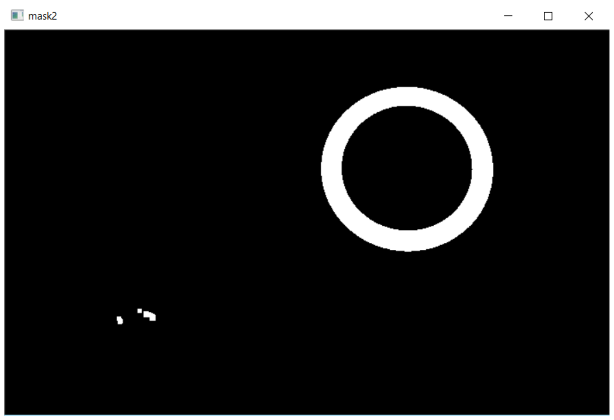
HSV color space representation

We can vary these values to not catch areas with low saturation or that are not really red, for example to not recognize the color of skin as red. But the saturation can be set quite high, since we assume that the color of a traffic sign is highly saturated. A first result after a certain variation of the range to catch red color looks like the following:



We can see that there are small regions that are captured also (the color of the house in the distance). By applying some morphological operations, some parts of these small areas can be filtered, if a good structuring element is chosen. Opening is a morphological operation that eliminates all pixels in regions that are too small to contain the structuring element.

After applying an opening with a certain structuring element it is visible, that some regions are filtered out:



The obtained image is binary, and it contains almost only the traffic sign. As it is binary, the objects are already labeled, so by applying a border tracing algorithm, we can find the borders of the ROI. Having the borders, we can find the region that we need to crop and to transmit to the next phase.

The best approach for **recognition** is using Convolutional Neural Networks (CNNs) [3]. The convolution operation is able to extract features. In the first layers low-level features are extracted and moving from layer to layer we reach high-level features. However, the number of layers should not be too high, because the system could be become slow and we want a compromise between speed and accuracy. For recognizing the traffic sign, a model is trained using a large databse of traffic signs of different classes.

IV. Implementation and Results

For the implementation Python programming language is used with the OpenCV library. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

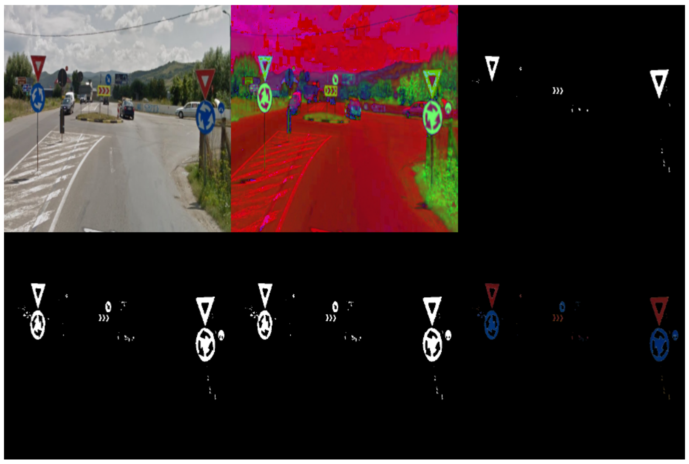
For the CNN Keras framework and Tensorflow software library is used

A. *Preprocessing and Detection*

The preprocessing phase consists of resizing the pictures, if we are not taking them from a camera, which has a predefined resolution; and converting to HSV image space. This is the first part of the detection pipeline.

The second part of the pipeline is filtering different regions which where specifig amounts of valid traffic sign colors appear. A filter is created for red colors, which consists of two different masks. The results are combined by using a bitwise or. The filter for other colors are similar, except that there is no need for two mask, since only red appears two times if we are looking to the linear representation of the hue values. The difficulty consists of calibrating the lower and upper thresholds for HSV values to filter colors in as many conditions as possible (shadow, wrong illumination etc.), but to not make the filters too sensitive.

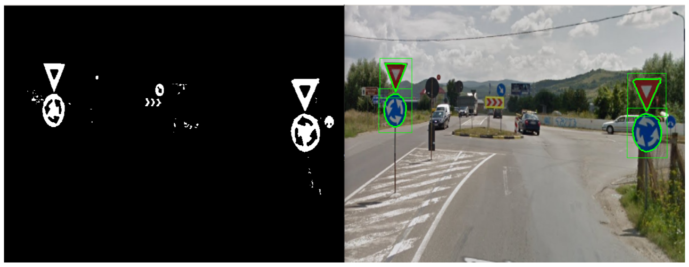
In the next step all the filters are combined to achieve the mask wich represents a binary image, where are the white pixels represent areas that contain the wanted colors. Finally, a morphological close is applied, to fill smart parts or connect adjacent areas.



Source image, HSV, red mask, all masks, after close, bitwise and with original image.

The third part of the detection pipeline is finding the signs on the filtered images and cropping them. The contours are found using the opencv function named *findContours*, which finds the contours in binary images.

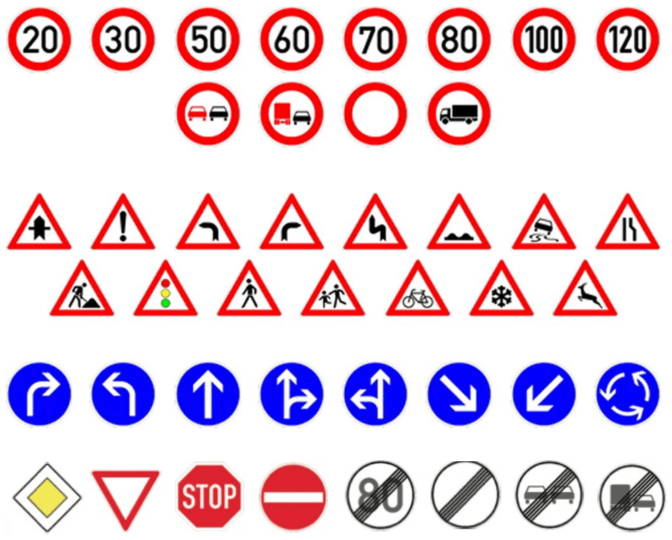
In many cases a lot of noise is present on the filtered image, therefore the identification of these areas should be avoided. Small parts, areas under the thresold area are filtered (in my impementation 500), only bigger regions are taken in consideration. These areas are approximated with curves/polygons with the *approxPolyDP* function which uses the Douglas-Peucker algorithm [4]. Using the result from the approximation, we can easily find a bounding rectangle for that object, which gives us coordinated that indicate the points where we should crop the image. Before cropping, there is another filtering introduce. In order to avoid long lines or rectangles with too long width or height, we check the ration of height and width to avoid these types of shapes.



Mask and bounding boxes on image.

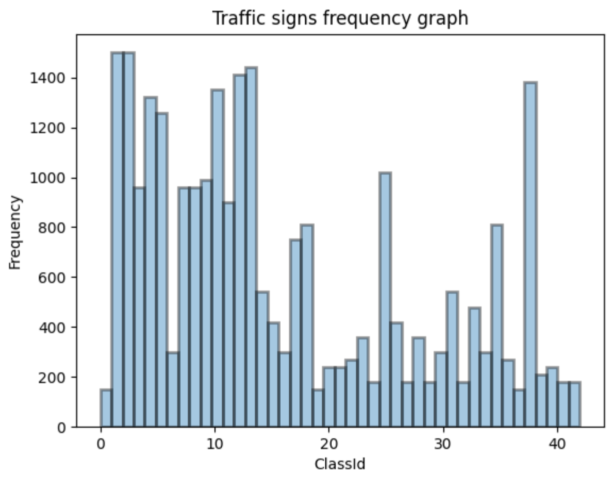
B. *Recognition*

For training the CNN model, the German Traffic Sign Recognition Benchmark (GTSRB) data set is used available on [5].



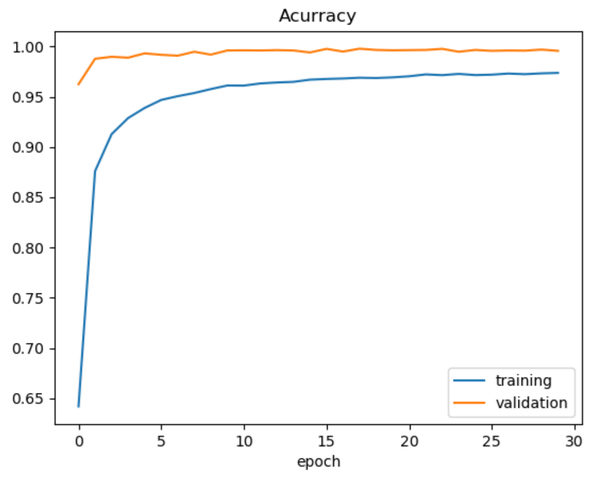
Traffic sign classes in GTSRB.

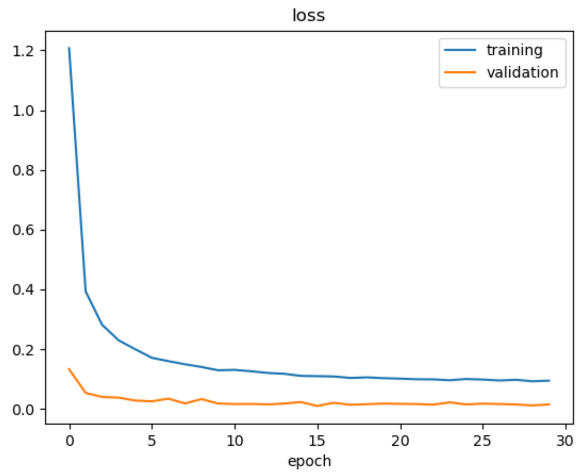
It contains 43 classes of traffic signs, but they are not equally distributed. Some classes contain over 1500 images, while other classes contain only 200 samples.



Traffic sign classes frequency in GTSRB.

I trained different models to see the results and their speed. The design of the first CNN is inspired by [6]. The results of [6]'s model are the following:





The images are converted to be grayscale, because the CNN uses one channel only. The contrast of the images are improved by applying histogram equalization and the values are normalized.

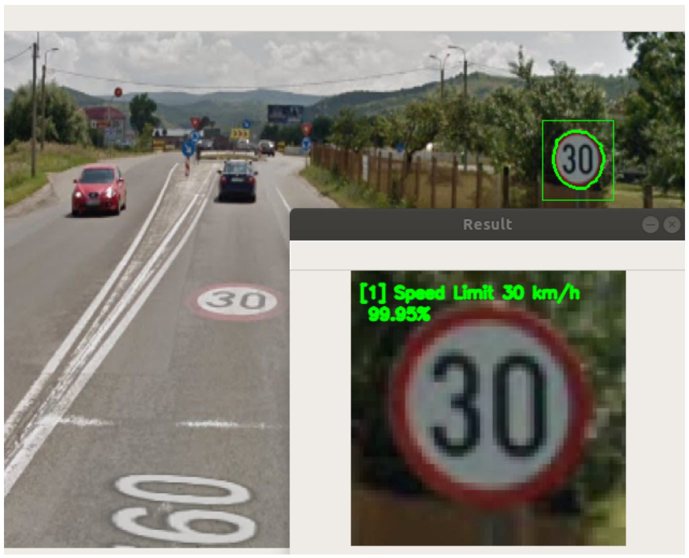
I used another CNN inspired by [7]. This approach keeps all the 3 channels of the images, but the model is trained on the original data set without data augmentation, although the results are quite good.

With the experience based on [6], [7], [8], I also tried to build my own CNN using all the three channels of the images, with the input shape (32, 32, 3) and using data augmentation.

C. Results

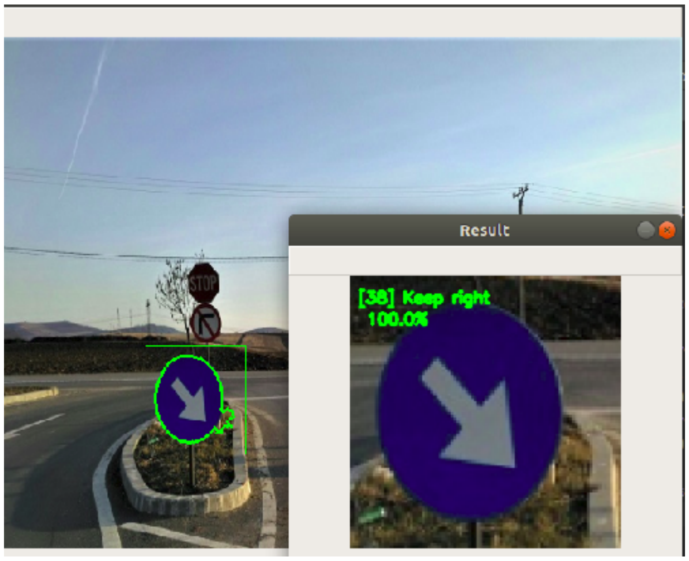
This section presents some examples, successful and unsuccessful detections and recognitions.

There are two modes to use the application, load pictures by selecting them using an open file dialog or using the web camera. If manually load pictures is active, then the system detects all the traffic signs on the images and predicts them one-by-one. If we are using the camera, it detects only the largest traffic sign.

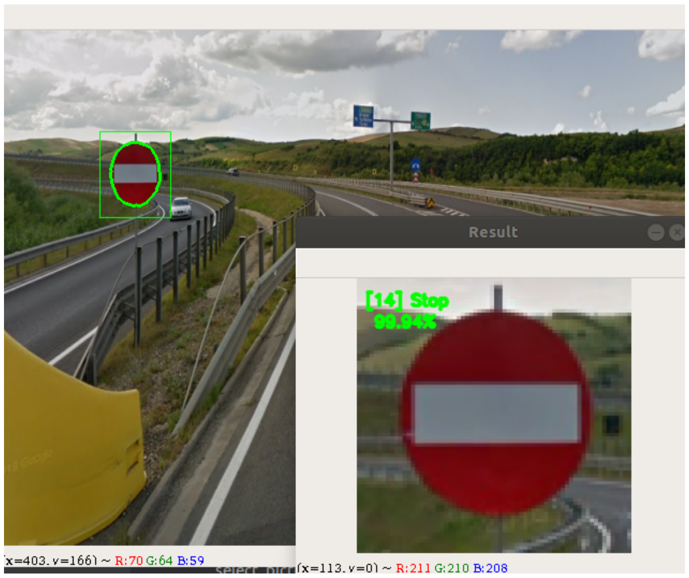


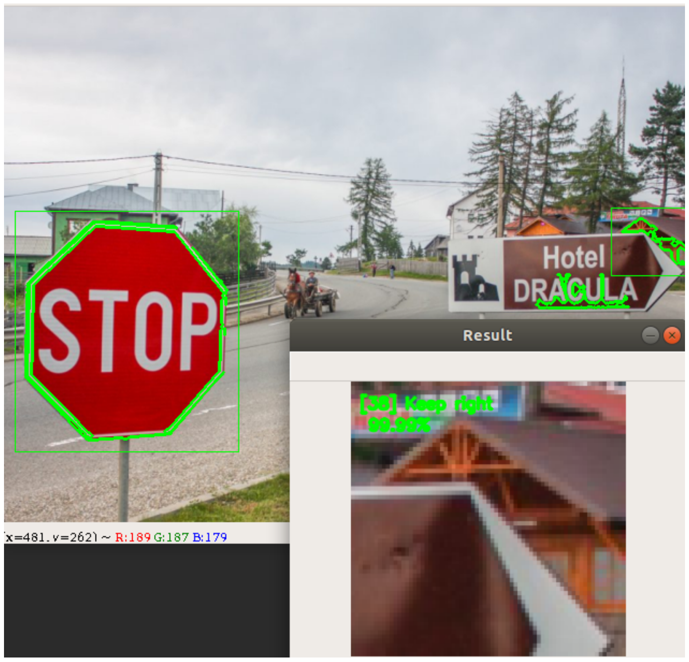
Correct detection and recognition

There still are regions detected that are not even traffic sings, these are false positive detections. There are also cases, when a traffic sign is not detected, even though it is on the frame (false negative detections).



Correct detection and recognition


Result
[14] Stop
99.94%
(x=403, y=166) ~ R:70 G:64 B:59
(x=113, y=0) ~ R:211 G:210 B:208
Traffic sign detected, but falsely recognized



False region detected

V. Conclusions

This project presents a simple approach to the traffic sign detection and recognition problem. The focusing is on the detection part, which uses classical image processing methods to find ROIs in an image. The second part, the classification uses machine learning techniques, which can be treated as a completely separate module than image processing.

The results obtained for detection are good in environments in which color thresholding is used, like highways according to [1] and [2]. In urban areas, there are a lot of factors that can cause too much noise or false detections.

The classification system also give satisfying results for the traffic sings that are available in our data set.

Both parts of the project need developements in the future. Some ideas would be:

- More precise color filters
- Reducing the number of false detections
- Some very important road signs are missing from the data set - adding new classes
- Balanced data set

In the future other methods could be used, like shape detection, or object detection CNNs. It is interesting to study the state of the art techniques that are used in object detection, like Faster RCNN, Single Shot Detector (SSD) or YOLO.

References

[1] Zsolt T. Kardkovcs, Zsombor Parczi, Endre Varga, dm Siegler and Pter Lucz. ”Real-time Traffic Sign Recognition System.” https://www.researchgate.net/publication/3724756_A_real-time_traffic_sign_recognition_system
[2] Dewan, Prachi, et al. ”An overview of traffic signs recognition methods.” International Journal of Computer Applications 168.11 (2017): 7-11.
[3] Shustanov, Alexander & Yakimov, Pavel. (2017). CNN Design for Real-Time Traffic Sign Recognition. Procedia Engineering.
[4] https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html#approxpolydp
[5] http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

[6] Murtaza Hassan. ”Classify Traffic Sins using CNN”. https://www.murtazahassan.com/classify-traffic-signs-using-cnn/ https://github.com/murtazahassan/OpenCV-Python-Tutorials-for-Beginners/tree/master/Advance/TrafficSignsCNN

[7] Sanket Doshi. ”Traffic Sign Detection using Convolutional Neural Network” https://towardsdatascience.com/traffic-sign-detection-using-convolutional-neural-network-660fb32fe90e

[8] Adrien Rosebrock. ”Traffic Sign Classification with Keras and Deep Learning” https://www.pyimagesearch.com/2019/11/04/traffic-sign-classification-with-keras-and-deep-learning/