2020

Real-Time Traffic Sign Detection and Classification Based on a Video Feed

Victor Ciuntu
vicciu@yahoo.com

Follow this and additional works at: https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations

Part of the Artificial Intelligence and Robotics Commons, and the Engineering Commons

Recommended Citation
https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations/6927

This Dissertation/Thesis is brought to you for free and open access by the Graduate Research & Artistry at Huskie Commons. It has been accepted for inclusion in Graduate Research Theses & Dissertations by an authorized administrator of Huskie Commons. For more information, please contact jschumacher@niu.edu.
ABSTRACT

REAL-TIME TRAFFIC SIGN DETECTION AND CLASSIFICATION
BASED ON A VIDEO FEED

Victor Ciuntu, MS
Department of Electrical Engineering
Northern Illinois University, 2020
Hasan Ferdowsi, Co-Director
Mansour Tahernezhadi, Co-Director

Autonomous vehicle development is currently progressing at a very fast pace and traffic sign detection and classification has an important role in it. This thesis looks at the history of autonomous vehicles as well as different implementations for traffic sign detection and classification. Multiple possible approaches are analyzed with the final goal of doing this task in real-time using a portable system.

To accomplish this task, the final solution uses a convolutional neural network for detection and classification combined with a custom optical character recognition algorithm for speed limit signs. The optical character recognition algorithm is built from the ground up using a combination of custom code and basic OpenCV color manipulation. The training and testing dataset is based on a combination of the Belgian Dataset, German Dataset, as well as images taken while driving in Illinois, United States. The final results are compared against other research papers in this field.
REAL-TIME TRAFFIC SIGN DETECTION AND CLASSIFICATION
BASED ON AVIDEO FEED

BY
VICTOR CIUN T U
©2020 Victor Ciuntu

A THESIS SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE
MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

Thesis Co-Directors:
Hasan Ferdowsi
Mansour Tahernezhad
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
</tr>
<tr>
<td>Historical Background</td>
</tr>
<tr>
<td>Autonomous Vehicles in the Everyday Life</td>
</tr>
<tr>
<td>Traffic Sign Detection and Classification</td>
</tr>
<tr>
<td>Objectives and Limitations of the Study</td>
</tr>
<tr>
<td>CHAPTER 2 LITERATURE REVIEW</td>
</tr>
<tr>
<td>Traffic Sign Detection and Classification</td>
</tr>
<tr>
<td>Machine Learning in Traffic Sign Detection and Classification</td>
</tr>
<tr>
<td>Optical Character Recognition in Traffic Sign Classification</td>
</tr>
<tr>
<td>CHAPTER 3 METHODOLOGY</td>
</tr>
<tr>
<td>Detection and Classification Algorithm</td>
</tr>
<tr>
<td>Optical Character Recognition Algorithm</td>
</tr>
<tr>
<td>Accuracy and Performance Testing</td>
</tr>
<tr>
<td>Training Data Acquisition</td>
</tr>
<tr>
<td>Testing Data Acquisition</td>
</tr>
<tr>
<td>CHAPTER 4 RESULTS</td>
</tr>
</tbody>
</table>
Detection and Classification Results ......................................................................................... 34
Optical Character Recognition Results ..................................................................................... 36
Results for the Complete System .............................................................................................. 37
CHAPTER 5 DISCUSSION, FUTURE WORK AND CONCLUSIONS .........................................39
Discussion .................................................................................................................................. 39
Future Work ............................................................................................................................... 41
Conclusion ................................................................................................................................. 42
REFERENCES ..........................................................................................................................43
LIST OF TABLES

Table 1 OCR Decision Making Table ................................................................. 27
Table 2 Detection and Classification Results .................................................... 35
Table 3 Optical Character Recognition Results .................................................. 36
Table 4 Results for the Complete System .......................................................... 38
Table 5 Final Result Comparison ....................................................................... 41
LIST OF FIGURES

Figure 1. Two participants at a 4-way stop sign at the DARPA Urban Challenge......................... 3
Figure 2. Waymo Self-Driving Chrysler Pacifica........................................................................... 4
Figure 3. Illustration of how layer propagation works in a convolutional neural network. ............ 12
Figure 4. AlexNet neural network layout. .................................................................................. 13
Figure 5. Example of US traffic sign containing text. .................................................................. 15
Figure 6. Sample output from an image processed by the YOLO algorithm. ..................... 19
Figure 7. Sample output from an image processed by the Mask R-CNN algorithm.................. 20
Figure 9. Sample output from an image processed by the Tensorpack Mask R-CNN............. 23
Figure 10. Example of US Speed Limit Signs............................................................................... 24
Figure 11. Speed limit sign before and after Otsu thresholding. Left – before. Right – after. ...... 25
Figure 12. Sample image from the Belgian Dataset................................................................. 30
Figure 13. Image of a speed limit sign taken using a Google Pixel 3 phone................................. 31
Figure 14. Image from the Belgian Dataset before the insertion of a traffic sign...................... 33
Figure 15. Image from the Belgian Dataset after a traffic sign is inserted.............................. 33
CHAPTER 1
INTRODUCTION

Historical Background

Every year more and more efforts are put into developing autonomous vehicles. This trend has been going on for dozens of years already and it doesn’t seem that the research rate is going to slow down anytime soon. The first concept of an automated roadway was shown during the 1939 World Fair by General Motors. In 1950, GM presented the Firebird II concept car which was capable of autonomous driving by following underground cables that emitted a radiofrequency signal. Even though concepts were shown from time to time, no real progress was made until the 1980’s and the 1990’s. That’s when microcomputers started to become available and they were finally small enough to be placed inside a vehicle. This progress lead to the start of true autonomous vehicle research based on image processing and other external sensors [1].

One of the first competitions involving autonomous vehicles was the DARPA Grand Challenge. Its first edition was held in 2004 where the contestants had to build vehicles capable of driving across a predefined off-road route. Before the actual contest, a small one-mile-long test course was built at California Speedway. 21 teams attempted to complete the test course and only 7 completed it. An additional 8 teams were able to navigate a sufficient amount of the course to be allowed to participate in the main event. The main event was 150 miles long and
was held in the Mojave Desert, California. It was initially planned to run the event for 10 hours, however only a few hours in, all vehicles suffered mechanical failures, got stuck on rocks, have been disqualified or simply withdrew from the contest. Even though this contest finished with no winners, it managed to motivate people to keep working on developing autonomous vehicles.

Next year, at the 2005 DARPA Grand Challenge things evolved a lot more different when compared to the 2004 edition. The qualification event was held once again at the California Speedway where 43 teams competed for a chance to participate in the final event. After the qualification, 23 teams were selected for the final challenge. 5 teams were able to complete the full 132-mile course. Also, 22 of the 23 teams were able to beat the previous year’s distance record of 7 miles, showing that that in just one year, significant progress was made in the area of autonomous vehicles.

Considering that the vehicles were capable of driving in off-road situations without other traffic, the organizers of the DARPA competition decided to step up the game by requiring the vehicles to be able to drive autonomously in an urban environment. The competition was called DARPA Urban Challenge and it required vehicles to be able to drive autonomously while following all California driving laws. From the initial 53 teams, only 11 were selected for the final. The main reason for such a strict elimination process was safety in order to avoid injuries to the people driving the non-autonomous vehicles. The human driven vehicles were used to create a higher traffic flow around the urban environment. The competition ended with 6 teams successfully completing the course. Figure 1 shows 2 of the participating vehicles stopped at a stop sign during the competition.
Even though no more DARPA challenges involving autonomous vehicle followed, the research for autonomous vehicles continued. One of the most know programs was the Google Self-Driving Car Project. The project was originally led by Sebastian Thrun, former director of the Stanford Artificial Intelligence Laboratory. He was also known for leading the team which won the 2005 DARPA Grand Challenge. Google’s project started gaining traction and in May 2012 they were able to obtain the first ever license for an autonomous vehicle in the United States. By August 2012, Google announced that their vehicles drove 300,000 autonomous miles without being involved in any accidents. For comparison, the average US driver has a crash once every 165,000 miles. Even though in 2016 the Google Self-Driving Car Project name was
dropped in favor of Waymo, their research keeps on going and by January 2020 their fleet of vehicles has driven 20 million autonomous miles. An example of a Waymo self-driving vehicle is shown in Figure 2.

Figure 2. Waymo Self-Driving Chrysler Pacifica. [2]

Considering the amount of people who are interested in autonomous vehicles and the rate of progress in the area, more and more companies start working on their own solutions. Some of the most known ones are Tesla, Apple, Uber and Amazon. Some less known companies like Comma.AI are also working on developing autonomous driving solutions which are focused at certain target demographics. In case of Comma.AI, the target is people who want to convert the vehicles they drive every day into autonomous vehicles for a low amount of money.
Autonomous Vehicles in the Everyday Life

So far it is obvious that a lot of effort is put into developing autonomous vehicles, but a question that might come up is how will they fit into our existing lives. The most obvious use for them is simply acting as a personal driver. Considering that in the United States the average person commutes 26 minutes to work, people can save 52 minutes daily by simply not having to drive. They can spend this time watching the news, reading a book, working on a laptop, or doing pretty much anything that doesn’t require too much movement.

Another possible benefit is the ability to request your vehicle to pick you up. Let’s say your vehicle is parked in a parking lot at a supermarket and it’s raining outside. Instead of walking to the vehicle in the rain, the owner can just request the vehicle to come and pick him up. This feature is currently implemented as a beta on some Tesla vehicles and there are countless videos of people benefiting from this feature.

A third benefit of autonomous vehicles is improved car sharing in a household. If two members of the family have commutes at different times of day, the car can drive the first person to work, then come back and pick up the second one. This way the number of vehicles per household will reduce and also the number of parking spaces needed in cities will be reduced. One disadvantage of this type of car sharing is that the cars will be driving more miles on the road because of the additional trips needed with no passenger on board.

The last, but not least important advantage is the increased safety. As real-world data shows, on average, autonomous vehicles are safer than human drivers. Considering that in the future the safety will continue to go up, there won’t be many reasons for people to drive their own vehicles. Also another safety benefit that autonomous vehicles have is that they are instantly
trained to the level of an experience driver. Human drivers need to drive tens of thousands of miles until they become experienced in driving in a variety of circumstances.

Now that the advantages of autonomous vehicles are clear, the next question is what does it take to convert a normal car into a self-driving one. The main aspects that need to be tackled are the following: traffic light monitoring, traffic sign detection and classification, lane detection, detection of other vehicles on the road, object and obstacle detection, and vehicle handling control. While there are a lot of systems that need to be developed, each one uses just a few sensors, and they can often be shared between the systems.

Traffic Sign Detection and Classification

One of the simpler systems in terms of the number of sensors required is the traffic sign detection and classification system. In most cases just one camera mounted at the front of the vehicle can be sufficient for the task. While this system is simple in terms of hardware, the software needed to correctly detect traffic signs is quite complex and vehicle manufacturers spend a lot of time and money on improving them. BMW, Mercedes, Tesla and others all have active traffic sign monitoring systems which show the current restrictions on the vehicle display. For example, if the car is currently driving in a no passing zone with a speed limit of 50 miles per hour, the appropriate restrictions will be shown on the car display. While at the moment most such systems are purely informative, they still help increase driver attention on the road by allowing the driver to focus more on the road and less on traffic signs.
Aside from the obvious application of helping drivers, traffic sign detection and classification has a set of other useful applications. The first and most researched one is autonomous vehicles. For a vehicle to be able to drive itself, it needs to be able to detect signs and know how to behave when encountering them.

Another possible application is the creation of an inventory of all traffic signs on the road. By driving through a certain area, the system would detect all traffic signs in that area and map them. In addition to mapping it could also analyze the current state of the traffic sign and notify the appropriate authorities if a traffic sign should be replaced soon.

Although it can be seen that there are a variety of areas where traffic sign detection and recognition solutions can be used, there are some difficulties implementing such systems. The main encountered difficulties are in regards to Autonomous Vehicle applications as such vehicles need to be able to drive in a variety of conditions (day, night, rain, fog, etc.). Aside from these meteorological conditions, there are also other issues to be taken into account.

For example, if half the traffic sign is covered by a shadow while the other half is bright, both the camera and the detection algorithm will struggle with this situation. The camera will either under-expose or over-expose half of the traffic sign, while the detection algorithm will not be able to detect the proper edges of the traffic sign.

Another challenging situation is when the sun is straight ahead of the camera, making the whole image almost unusable. Human drivers use the sun visor as well as sunglasses to combat this situation. While a similar solution could be implemented on a real vehicle, this would significantly increase the complexity of the traffic sign detection and classification system.
Objectives and Limitations of the Study

The main goal of this research was to find better ways to detect and classify traffic signs. In addition to this general objective, some additional requirements were set. Firstly, the final solution needs to be able to work in real-time based on a video feed from a camera mounted inside the vehicle. Also the complete system should be small enough to fit inside a vehicle without taking a significant amount of space. Lastly, the final solution needs to be able to scale linearly or almost linearly so that performance improvements can be easily done as more powerful hardware keeps getting developed.

As with any study, there are certain limitations which are necessary to make in order to define the conditions under which the final solution will work. The main limitation is the fact that this solution will only work on a limited set of traffic signs. That means that if it is desired to detect a new traffic sign, it will be necessary to collect sufficient training samples containing the traffic sign in question and retraining the system.

Another limitation is that depending on the quality of the camera, the visibility under low light conditions will be reduced. Use of an infrared camera at night might help, but in that case the solution will have to be trained for use with both cameras. Lastly, it is assumed that the visibility conditions are good and no phenomena like dense fog or snow are obstructing the view of traffic signs.
CHAPTER 2
LITERATURE REVIEW

Traffic Sign Detection and Classification

When looking at prior research in the Traffic Sign Detection and Classification area, a wide variety of papers were analyzed. An article released in 1997 that was titled “Road Traffic Sign Detection and Classification” [3] showed multiple creative solutions to the discussed problem, especially considering the limited processing power the computers had at that time. Traffic sign detection was done in multiple steps. First, Color Thresholding was applied to the original image in a modified HSI color space. The second step was corner detection using a convolutional mask. Different masks were designed for different shapes of traffic signs. Lastly, after all corners are detected, the center of mass for each traffic sign is calculated.

Traffic sign classification was done using a combination of neural networks and traditional image processing methods. The first step after a traffic sign was detected was resizing it to an image 30 x 30 pixels using the nearest neighbor method. Afterwards, the image was fed to one of 2 neural networks based on the shape of the sign. One neural network was for triangular signs and another one for circular signs. The results showed an average processing time of 220 ms for a 256x256 image for the sign detection algorithm and 1.2s for the neural network using a PC486 processor. Unfortunately testing was done on a very limited dataset and no detection and classification accuracy results were provided.
With the evolution of processors and graphics cards, more computation heavy solutions could be implemented. In a paper published in 2010, a new approach for traffic sign detection was shown [4]. It used a modified Histogram of Oriented Gradients (HOG) algorithm. The first step of the algorithm was to convert the initial image into HOG features. In order to allow detection for both small and large traffic signs, 2 images are used, one full size image, and one downscaled image. Next, a sliding window algorithm was used to detect traffic signs based on the HOG features in the image.

The modified HOG algorithm consists of 3 steps. First, uninteresting parts of the image are discarded by analyzing color information. Next SIFT features are extracted at interest points and they are matched to a dictionary of traffic sign features. Lastly, the detected traffic sign is validated by checking color consistency and doing template matching. The final results for this solution were 80% recall and 80% precision for the detection of blue circular traffic signs and for the triangular traffic signs.

While most researchers focused on just the software side of traffic sign detection and classification, some came up with creative hardware solutions. One such paper is called “An active vision system for on-line traffic sign recognition” [5]. The key difference from other papers was the use of 2 cameras: one wide-angle camera and one telephoto camera mounted on a servo motor. The wide-angle camera handled the detection of candidate signs and when a detection was made, a signal was sent to the servo to redirect the telephoto camera towards the traffic sign. This 2-step approach worked reasonably well in practice, especially considering the limited technology available in the year 2000.
Another paper which brought new ideas to this research field is “Traffic sign recognition and analysis for intelligent vehicles” [6]. In addition to the task of doing traffic sign detection and classification, this paper also focuses on the analysis of the traffic signs. Monitored characteristics are visibility of the traffic sign, condition of the traffic sign and how good the traffic sign placement is. This project was part of the AUTOCAT project where a van was built with this system on board. After each driving session, a report can be generated to show the detected traffic signs as well as their current state.

Machine Learning in Traffic Sign Detection and Classification

In the research previously mentioned in this chapter, most solutions used some kind of custom algorithm for detecting traffic signs and then a neural network to do the classification. There were 2 main reasons for that. First of all, the computer hardware up to that point was too slow to run a neural network on the whole image. The second reason was that until 2012 almost no implementations of convolutional neural networks provided both good results and low processing time.

Convolutional neural networks work on a sliding window approach. They have a kernel value which defines the width and the height of the window. That window moves over each position on the input of the layer and produces a certain number of outputs depending on the depth of the output. In case of the convolutional layer in Figure 3, the input has a depth of 3, while the output depth is 5. In addition, as a result of the convolution operation, the output width and height of the layer are reduced by the kernel size minus one.
One of the first research papers which proved the effectiveness of convolutional neural networks on image processing was “ImageNet Classification with Deep Convolutional Neural Networks”, also known as AlexNet [7]. The network was 8 layers deep, 5 of which were convolutional layers, with the last 3 being fully connected. The activation function used was ReLU which showed better performance when compared to the more traditional tanh and sigmoid functions. At the 2012 ImageNet challenge, this solution significantly outperformed the competition. It achieved a 15.3% top-5 classification error, while the nearest competitor achieved a 26.2% error. The layout of this neural network can be seen in Figure 4.
From that point, convolutional neural networks started becoming to go-to method for doing almost any kind of computer vision task, including detection and classification of traffic signs. This shift became even more predominant once libraries like Tensorflow and CNTK were released in 2015 and 2016 respectively. Also the release of cheaper and more powerful graphics cards further helped advance research in the area, especially in cases where the research was budget limited.

One paper which makes extensive use of convolutional neural networks was published in 2017 and it used a modified YOLO (You Only Look Once) neural network for the detection of traffic signs [8]. The main advantage of YOLO is it’s processing speed. Because the image is passed through the neural network only once, framerates of up to 60 FPS are pretty typical on modern hardware. The previously mentioned research built 3 different variations of the YOLO neural network and tested their accuracy. The final results for the best network on the German Traffic Sign Dataset [9] were 95.31% precision and 90.37% recall. Computation performance was also good with an average of 20 ms processing time per frame.
Another paper which uses convolutional neural networks for both detection and classification of traffic signs is “Simultaneous Traffic Sign Detection and Boundary Estimation using Convolutional Neural Network” [10]. In addition to the previous solutions which do detection, classification and estimate the bounding box of the traffic sign, this paper also aims to provide a boundary estimation for the traffic sign. The detection of traffic signs was tested using 2 separate neural networks: VGG16 and Inception V2. The boundary calculation is done in a multi-step process. First of all, during the training process templates are generated for all shapes of the traffic signs. Next, when predicting a traffic sign, after the detection is done, 4 points are chosen to represent a rough estimation of the traffic sign. Next, based on the previously mentioned template, and using the information from the previous step, the boundary box is generated. The final results of the paper showed an 89.4% precision and 73.2% recall for the detection of traffic signs in the German Traffic Sign Dataset.

Considering the previous work mentioned, this thesis will focus on alternative machine learning solutions for traffic sign detection and classification. In addition, a custom optical character recognition algorithm for speed limit detection will be analyzed as a way of increasing the performance for the complete system.

Optical Character Recognition in Traffic Sign Classification

While most of the research in the traffic sign detection and classification field focuses on machine learning or custom image processing methods, significant research is also done in the Optical Character Recognition (OCR) area. This is especially true for countries like the United
States where many traffic signs have text written on them. An example of such a sign is shown in Figure 5.

![Road Work Ahead sign](image)

**Figure 5.** Example of US traffic sign containing text.

One paper published in 2005 looking into this topic is “Recognizing Text-Based Traffic Sings” [11]. It focuses on recognizing text on traffic signs on the roads of the United Kingdom. A 4-stage approach was used to do the OCR. The first step was distortion correction. Because the camera might not be perfectly level with the traffic sign, it might not appear as a perfect rectangle. To fix this, the detected sign is rotated and reshaped to look like a rectangle. Next, each line of text is separated from each other based on the Maximally Stable External Region. Lastly, each line of text is sent to the Tesseract engine for text recognition. In addition, results from multiple frames are fused together to provide a more accurate result. The final results showed a comparison between using just Tesseract on the detected sign and using the method described in the thesis. Tesseract was able to achieve 48% precision and 34% recall. The solution in the thesis achieved 87% precision and 91% recall.
Another research paper that was analyzed used 3 different criteria to classify traffic signs [12]. First, certain features like edges, corners and angles are detected on the candidate traffic sign and are compared against a database of known traffic signs. In addition, the shape of the traffic sign is also compared in order to increase the final accuracy. Lastly, optical character recognition is applied to detect text inside the traffic sign. By combining all 3 methods they were able to achieve competitive results for traffic sign classification at the cost of processing time. The average accuracy achieved was 92.1%, however the processing time for one traffic sign was 6.42 seconds.
CHAPTER 3

METHODOLOGY

When designing the system for detecting and classifying traffic signs, it was necessary to divide the task into multiple smaller tasks which could be tackled independently.

The initial plan was to use 2 neural networks. The first neural network would detect the presence of traffic signs in images captured by the onboard camera. If a traffic sign is detected, the algorithm would also provide the bounding box for the sign. This information is then sent to the second neural network which classifies the detected traffic sign.

While the above solution seemed promising at first, 2 main issues became noticeable after further testing. Firstly, the classification algorithm had trouble identifying traffic signs which looked similar. An example of this are speed limit signs where there are a lot of signs with tiny differences between them. The second issue was that in order to get good accuracy, the classification algorithm was over 20 layers deep and that added a significant amount of computation time on top of the already demanding traffic sign detection neural network.

Because of the above issues, it was decided to make 2 changes to the system. The first modification was to replace the 2 neural networks with a single one which can do both detection and classification of traffic signs. The second change was to bundle all speed limit signs in one class and then run an Optical Character Recognition algorithm on them to detect the posted speed limit.
Detection and Classification Algorithm

Considering the amount of machine learning object detection and classification solutions which already exist, it didn’t make much sense to develop a new algorithm. There were 3 main criteria when looking for machine learning solutions.

The first criterion was the amount of processing power required. The algorithm needs to run in real time at a decent framerate on a portable system. In this case a portable system would be something that can be installed in a vehicle without taking too much space or weight. Since typical consumer GPUs fall in this category, testing was done on an NVIDIA GeForce 1070. The minimum acceptable framerate to allow detection and classification at highway speeds is around 2 frames per second. At a speed of 70mph, that would give the neural network 2-3 opportunities to detect a traffic sign on the side of the road. Of course, the higher the framerate, the better the final results will be.

The second criterion when choosing a machine learning algorithm was the amount of data required for training. Acquiring training data can be an extremely time consuming process. Because of this, one goal was to find a solution capable of achieving good accuracy after training on 1000-2000 training images.

The last criterion taken into account was the suitability of the machine learning algorithm for detection of small objects. While there are solutions which offer fast processing, one of the main drawbacks is poor detection for small objects. Since traffic signs occupy a small amount of space in a typical driving scenario, it was important to find a solution which doesn’t have this drawback.
One of the first deep learning algorithms analyzed was YOLO (You Only Look Once) [13]. The main benefit of this algorithm is its speed. Depending on the implementation, on a GTX 1070, typical framerates are between 30 and 60 FPS. The main issue however is the inability to detect small objects, and even if detected the bounding box usually has a high error. For this reason, this algorithm is not really suitable for our application. An output example from the YOLO algorithm is shown in Figure 6.

![Sample output from an image processed by the YOLO algorithm.](image)

Figure 6. Sample output from an image processed by the YOLO algorithm.
Another deep learning solution analyzed was Mask R-CNN [14]. It has no issues detecting small objects and in addition it also provides a mask for the detected object. This feature was particularly useful for traffic signs which were later sent to the OCR algorithm. Unfortunately, the performance is significantly worse than the YOLO approach. On the previously mentioned GPU, the average framerate was hovering around 1 FPS. An output example from the Mask R-CNN algorithm is shown in Figure 7.

Figure 7. Sample output from an image processed by the Mask R-CNN algorithm. [15]
The final algorithm analyzed is a combination of multiple research papers into one solution. It’s part of a deep learning toolkit called Tensorpack [16] and it combines the Mask R-CNN algorithm with 3 others to build a solution with both good performance and accuracy. The framerate is around 5 FPS on a GeForce GTX 1070. There are also no issues detecting small objects and after detection a mask of the detected object is provided. Lastly, this implementation allows the use of pre-trained weights for the backbone of the neural network, thus accelerating the training process. Because of all these advantages combined, it was decided to use this solution for detection and classification of traffic signs.

Because the Tensorpack solution mentioned before allows a user defined backbone for the neural network, it was important to choose one suitable for this project. After testing it was decided to use the ResNet-50 [17] backbone because of its low processing time while maintaining a good accuracy. The structure of the ResNet-50 network consists of one input layer, followed by a multitude of convolutional layers. The last 2 layers are a fully connected layer and the output layer.

One important feature of the ResNet-50 network is its ability to bypass layers during training. One big issue when training deep neural networks is vanishing or explosion of weights during backpropagation. By allowing the training algorithm to skip some of the layers, badly trained weights can be ignored temporarily or even retrained. This leads to much better training results for deep neural networks like ResNet-50. The ability to skip layers can be observed in Figure 8.
During testing it was observed that more than half of the “false alerts” were reported as large objects, usually being more than half the width or the height of the image (a false alert is when the algorithm reports a traffic sign present where there isn’t one). Since traffic signs in normal conditions can’t be that big, those false alerts were filtered out by ignoring all detections which had a width larger than 1/3 of the total image width or a height larger than 1/3 of the total image height. In addition, all detections with a confidence score lower than 50% were also ignored. An example output from the neural network after training and filtering is shown in Figure 9.
After the results are filtered, one last check is to verify if we need to apply the OCR algorithm on the sign or not. In this case the check was to see if the detected sign is a speed limit sign. If it is, then the OCR algorithm is applied before the final output. Otherwise the result is sent directly to the output.
Optical Character Recognition Algorithm

As previously mentioned, some traffic signs are very similar to each other and neural networks have a hard time distinguishing them. For example, if we look at 2 United States Speed Limit Signs, one with a 50 mph limit and another one with a 60 mph limit, they will be almost identical. The difference between them is just one digit which is slightly different from the other one. Because of this it was decided to use a OCR algorithm to differentiate between different speed limit signs. An example of 3 different speed limit signs is shown in Figure 10.

![Figure 10. Example of US Speed Limit Signs.](image)

The first approach was to analyze the off the shelf solutions. At the moment, Tesseract is the most popular OCR solution and it seemed that it might work well for this use case. Unfortunately, testing showed that it only works on longer chunks of text, 2 – 3 words being the minimum for proper recognition. In our case the speed limit signs have just 2 large digits and
they were simply ignored by Tesseract. The same issue was observed when testing GOCR which is another off the shelf solution for optical character recognition.

The above test results showed that a custom solution is needed to detect speed limit signs. The task of detecting the speed limit on a speed limit sign was divided in 2 tasks: separating the candidate digits from the rest of the image and classifying the candidate digit.

The first task was further divided in the following sub steps:

1. Convert the original image to grayscale format.
2. Run Otsu thresholding using OpenCV to get a binary image (See Figure 11).
3. Detect all continuous blocks of black pixels.
4. Remove all blocks which have a width of less than 20% of the traffic sign width and a height of less than 20% of traffic sign height.
5. Send the remaining blocks to the second step of the algorithm.

Figure 11. Speed limit sign before and after Otsu thresholding. Left – before. Right – after.

The second part of the OCR algorithm looks for the presence of 6 different features on the target image block. Afterward, based on those results a decision is made regarding the digit represented in the image. The 6 features are:
1. The number of continuous white blocks in the image, excluding those which touch the border of the image. For example, this feature returns 2 for the digit 8, 1 for the digit 6 and 0 for the digit 1.

2. Does the image have a horizontal black line at the bottom of the image? This is done by checking the widest continuous black block in the bottom 10% of the image. If the width of the block is above 50% of the image width and above 40% of the image height, the feature returns true. Otherwise it returns false.

3. Does the image have a horizontal black line at the middle of the image? This is done by checking the widest continuous black block in the middle 20% of the image. If the width of the block is above 50% of the image width and above 40% of the image height, the feature returns true. Otherwise it returns false.

4. Does the image have a horizontal black line at the top of the image? This is done by checking the widest continuous black block in the top 20% of the image. If the width of the block is above 50% of the image width and above 40% of the image height, the feature returns true. Otherwise it returns false.

5. Does the image have a vertical black line at the right side of the image? This is done by checking the tallest continuous black block in the rightmost 33% of the image. If the height of the block is above 75% of the image height, the feature returns true. Otherwise it returns false.

6. Does the image have a diagonal line from the bottom-left to the top-right of the image? This is done by checking the pixels which deviate at most by one seventh of the image width from the diagonal. If the largest continuous block of black pixels has a height of
over 60% of the image height and a width of over 60% of the image width, the feature returns true. Otherwise it returns false.

For all 6 features, the continuous blocks were detecting by using the recursive “Flood Fill” algorithm in 4 directions. For example, if the feature was looking for a black block, the algorithm would search for the first black pixel in the image and then run the Flood Fill algorithm starting from that point. This was repeated until all black pixels have been processed.

After feature detection is done, the results are compared against a table (see Table 1) which represents the expected result for each digit. For feature 1 the number in the table represents the number of continuous white blocks detected. For other features 1 means “True” and 0 means “False”. For all features -1 means that that feature is ignored.

<table>
<thead>
<tr>
<th>Digit/Feature</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>
After the prediction is made for all candidate digits, the digits are combined from left to right to build the final number in the speed limit.

Accuracy and Performance Testing

To properly identify the effectiveness of each algorithm, the testing results were split in 3 categories: Detection and Classification Performance, OCR Performance, Performance for the complete system.

For the Detection and Classification Performance, 6 metrics were used:

- Time required to process one frame.
- Percentage of traffic signs correctly detected.
- Percentage of traffic signs missed.
- Percentage of “false” traffic signs detections.
- Percentage of traffic signs correctly classified from the list of correctly detected traffic signs.
- Percentage of traffic signs incorrectly classified from the list of correctly detected traffic signs.

To measure the OCR algorithm performance, 3 metrics were used:

- Time required to process one speed limit sign with a resolution of 100 x 130 pixels.
- Percentage of correctly identified speed limits.
• Percentage of incorrectly identified speed limits.

To test the performance of the complete solution, 6 metrics were used:

• Time required to process one frame.
• Percentage of traffic signs correctly detected.
• Percentage of traffic signs missed.
• Percentage of “false” traffic signs detections.
• Percentage of traffic signs correctly classified from the list of correctly detected traffic signs.
• Percentage of traffic signs incorrectly classified from the list of correctly detected traffic signs.

Training Data Acquisition

When building a training dataset for a deep learning neural network there is a number of important characteristics that have to be taken into account.

Firstly, it is important to have good quality photos. Training won’t succeed if the traffic signs are barely distinguishable. This includes both traffic signs which are too far away from the camera as well as photos which are blurred or grainy.

Another important characteristic to take into account is that the dataset has to be balanced. If there are 1000 training samples for one class and only 10 samples for another class, the neural network won’t be able to properly predict both classes.
Lastly, each class needs a sufficient amount of training samples. If there are too few training samples, the neural network will simply overfit the existing training samples without actually identifying what makes each class unique.

Having considered the above points, the decision was made to build the training dataset using European traffic signs. There are 2 main datasets which contain European traffic signs: The German Dataset (GTSDB) [9] and the BelgianTS Dataset [18] (See Figure 12). Both datasets were annotated, however the annotations contained only the bounding box of the traffic signs, not the actual outline. Because of this it was necessary to reannotate the traffic signs and build the dataset as a COCO dataset. The reason for this is the fact that the Tensorpack Mask R-CNN solution takes COCO style datasets as inputs. The annotation and conversion to COCO dataset was done using the “labelme” tool for Python [19].

![Figure 12. Sample image from the Belgian Dataset. [18]](image-url)
Because one of the desired abilities for the final solution was detection and classification of US speed limit signs, some additional data had to be collected. This was done by recording car rides around Illinois, US, using a Google Pixel 3 phone. After the recordings were done, for each traffic sign encountered in the video, 5 photos were extracted representing distances between 5 and 30 meters from the speed limit sign. In the final training dataset there were 50 images containing speed limit signs. They were then annotated and added to the complete dataset. For a sample image from the speed limit dataset see Figure 13.

![Figure 13. Image of a speed limit sign taken using a Google Pixel 3 phone.](image)

**Testing Data Acquisition**

The testing dataset is used to test the accuracy of the trained neural network. When building the testing dataset, it is important to take into account the following considerations:
Firstly, the testing dataset can’t share images with the training dataset. One important test for any trained neural network is to check if it overfits the training dataset or not. If the testing dataset is too similar to the training one, this isn’t possible. The second point to take into account when building a testing dataset is that the dataset has to contain images showing edge cases. By having said edge cases, testing results will be able to highlight cases where the neural network makes both “false alarms” as well as missed detections. Lastly, the testing dataset has to be large and balanced in order to provide accurate results. If the dataset is too small or unbalanced, the results won’t reflect the performance of the system in a real-world environment.

Considering the above points, it was decided to take unused images from the Belgian Traffic Sign Dataset. Those images don’t contain any of the 20 traffic signs that the neural network is capable of detecting and thus are the perfect candidates to test the “false alarm” rate of the neural network.

The detection accuracy was tested by placing crops of traffic signs on top of the previously mentioned images. The crops were placed in one of 2 areas of the image to simulate traffic signs placed on the left or the right side of the road. The first area consisted of the leftmost 20% of the image while the second area consisted of the rightmost 30% of the image. In addition, for both areas, the traffic signs could only be placed in the middle 50% of the image on the Y axis.

In Figure 14 and Figure 15, 2 images are shown. One without a traffic sign inserted and one with a traffic sign placed in a random location. Figure 15 shows the image after the detection algorithm finished processing the information. In this case, it shows the bounding box of the traffic sign and also the fact that it’s a pedestrian crossing.
Figure 14. Image from the Belgian Dataset before the insertion of a traffic sign. [18]

Figure 15. Image from the Belgian Dataset after a traffic sign is inserted.
CHAPTER 4
RESULTS
Detection and Classification Results

The results for the detection and classification solution are shown in Table 2. As it can be seen, the average processing time is 197.5 ms which leads to a framerate of about 5 FPS on a GeForce GTX 1070. For traffic sign detection this kind of framerate is more than acceptable and actually offers multiple chances to detect each traffic sign. If some additional filtering is added by requiring the same traffic sign to appear in 2 or 3 consecutive frames, the number of falsely detected traffic signs can be further reduced. Also, note that a percentage of 5.29% for missed traffic signs is not a cause for concern. The average time a traffic sign is in view of the camera at a speed of 30 mph is about 8 seconds. At a framerate of 5 FPS, the neural network has 40 opportunities to detect the traffic sign. If we look at faster speeds, similar to those encountered on the highway, there are still enough opportunities to detect the traffic sign. At a speed of around 70 - 80 mph, the traffic sign would be in view for about 15 frames. Because of this, in real world situations, the number of missed traffic signs would be significantly smaller.
Table 2
Detection and Classification Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Average processing time (ms)</td>
<td>197.5 ± 36.25</td>
</tr>
<tr>
<td>Correctly Detected Traffic Signs (%)</td>
<td>94.71 ± 1.68</td>
</tr>
<tr>
<td>Traffic Signs Missed (%)</td>
<td>5.29 ± 1.68</td>
</tr>
<tr>
<td>Falsely Detected Traffic Signs (%)</td>
<td>9.75 ± 3.5</td>
</tr>
<tr>
<td>Classification Accuracy (%)</td>
<td>91.72 ± 2.46</td>
</tr>
<tr>
<td>Classification Error (%)</td>
<td>8.28 ± 2.46</td>
</tr>
</tbody>
</table>

When comparing results found in other papers to the results from this thesis, they have similar performance in most cases. In a paper published in 2013, the results achieved for the detection of mandatory traffic signs were 90% detection precision and 90% recall [20]. In another paper published in 2017 which uses a modified YOLOv2 neural network the results were 90.37% precision and 95.31% recall [8]. For both papers, only detection of 3 types of traffic signs were taken into account: danger signs, prohibitory signs and mandatory signs. All traffic signs in each of the 3 categories share significant features. An example can be the large red triangle that is present on the outside of all signs in the danger signs category.

In the previously mentioned papers, precision signifies the same thing as “Correctly Detected Traffic Signs” in this thesis. Recall can be calculating by subtracting the percentage of “Falsely Detected Traffic Signs” from 100%. For this thesis, a precision of 94.71% and a recall of 90.25% was achieved. When comparing these results to those in [8], this thesis achieves a higher precision rate, while having a slightly lower recall rate. One thing to keep in mind however is the fact that the results in this thesis were achieved while targeting a much higher
number of traffic sign categories than [8] and [20]. The solution in this thesis was capable of
detecting 20 different traffic signs that can be grouped in 9 distinct categories.

Optical Character Recognition Results

The optical character recognition results are shown in Table 3. The average processing
time was measured by running the recognition of one speed limit sign with a resolution of 102 x
129 on an Intel Core i7 7700. The task was single threaded so if desired, multiple sign detection
can run in parallel with minimal impact on the processing time. The performance of this
algorithm was also tested on a portable Jetson TX2 module. The average processing time in this
case was around 200ms for one speed limit sign. Once again, this performance can be increased
by running multiple traffic sign detections in parallel. Also considering that the processing time
for one frame by the neural network is also around 200ms, such a processing time for the speed
limit signs shouldn’t create any slowdowns.

Table 3

Optical Character Recognition Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Average processing time (ms)</td>
<td>54.027 ± 12.49</td>
</tr>
<tr>
<td>Classification Accuracy (%)</td>
<td>93.1 ± 2.83</td>
</tr>
<tr>
<td>Classification Error (%)</td>
<td>6.9 ± 2.83</td>
</tr>
</tbody>
</table>
The classification accuracy was 93.1% which is pretty competitive with the existing solutions. While this accuracy might not seem too high, it is important to remember that no other text can be used as a reference by the OCR algorithm. Tesseract, one of the leading OCR solutions, detects chunks of text and does text re-alignment before doing the character recognition. This is not possible to do when we only have 2 digits as a reference. According to a paper published by Ray Smith [21] average accuracy for Tesseract is between 95% and 97%.

Results for the Complete System

The processing time for a frame which doesn’t contain speed limit signs is 197.5ms. The total processing time for a frame which contains speed limit signs is around 250ms. Even though the processing time for the frames with speed limits is higher, it doesn’t affect the framerate of the complete system. This is because the neural network uses mostly GPU resources while the traffic sign detection uses just CPU resources. As a result, while the speed limit signs from the current frame are being processed, the deep learning algorithm can already work on processing the next frame. As mentioned in the first part of this chapter, a framerate of 5 FPS is perfectly adequate for normal driving and even offers multiple chances to detect each traffic sign. Also additional filtering can be implemented to improve the detection performance. The final test results for the complete system are shown in Table 4.
Table 4

Results for the Complete System

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Average processing time (ms)</td>
<td>197.5 ± 54.027</td>
</tr>
<tr>
<td>Correctly Detected Traffic Signs (%)</td>
<td>94.71 ± 1.68</td>
</tr>
<tr>
<td>Traffic Signs Missed (%)</td>
<td>5.29 ± 1.68</td>
</tr>
<tr>
<td>Falsely Detected Traffic Signs (%)</td>
<td>9.75 ± 3.5</td>
</tr>
<tr>
<td>Classification Accuracy (%)</td>
<td>91.39 ± 2.49</td>
</tr>
<tr>
<td>Classification Error (%)</td>
<td>8.61 ± 2.49</td>
</tr>
</tbody>
</table>

Since the OCR algorithm doesn’t affect the detection of traffic signs, the detection accuracy, the number of traffic signs missed and the number of falsely detected traffic signs stay the same as in the first part of this chapter.

The classification accuracy did drop a bit because the speed limit signs need to be classified correctly by both the neural network and by the OCR algorithm to provide a correct result. The drop was insignificant though, with the average classification accuracy dropping from 91.72% to 91.39%.
CHAPTER 5
DISCUSSION, FUTURE WORK AND CONCLUSIONS

Discussion

When starting this research, 3 main goals were set:

1. The processing time for the algorithm should be adequate for real time processing from a video feed.

2. The detection and classification accuracy should be high enough to work in real driving conditions.

3. The complete system should be installable in a vehicle without taking too much space or weight.

The final achieved framerate for the complete system was 5 FPS. Considering a driving speed of 70mph on the highway, the traffic sign is big enough for detection and classification for an average of 3 seconds. This offers 15 chances for the neural network to correctly detect and classify the sign. If we go down to city driving speeds of 25-30 mph, the neural network will have 30-40 chances to do a detection. Considering this, it can be said that the first goal was met. Additionally, with a more powerful GPU this framerate can be easily increased as the neural network is able to scale linearly.
The final system had a detection accuracy of 94.71% and a classification accuracy of 91.39%. While these numbers could be higher, there are multiple ways to achieve better accuracy in real driving scenarios. One example would be requiring the same traffic sign to be detected and classified for a few frames in a row. This way a lot of the false detections would be filtered out. Because of this, the second goal can also be consider satisfied in real world driving conditions.

In terms of space and weight, the final system can be smaller than a box of shoes and weigh under 5 lbs. All it needs to run is a board with a PCI-Express slot like the NVIDIA Jetson TX2, an external graphics card and a power supply. Considering the above, it can be said that the third condition was also satisfied.

When comparing the results achieved in this thesis to those in other papers, this research is competitive and brings new approaches to solving the problem of traffic sign detection and classification. In particular, one approach that isn’t commonly used is the use of Optical Character Recognition to improve detection and classification accuracy for visually similar traffic signs. In [5], OCR was used, however their goal was to read text like city names from traffic signs, not improve detection and classification accuracy.

Table 5 shows a comparison between the results achieved in this thesis and those in other papers. As it can be seen, the approach shown in this thesis achieves better results when compared to other papers in one or both accuracy metrics.
Table 5

Final Result Comparison

<table>
<thead>
<tr>
<th>Solution</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Traffic Sign Detection pipeline based on interest region extraction (2013) [20]</td>
<td>90%</td>
<td>90%</td>
</tr>
<tr>
<td>A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2 (2017) [8]</td>
<td>90.37%</td>
<td>95.31%</td>
</tr>
<tr>
<td>Improved Faster R-CNN Traffic Sign Detection Based on a Second Region of Interest and Highly Possible Regions Proposal Network (2019) [22]</td>
<td>90%</td>
<td>85.2%</td>
</tr>
<tr>
<td>Multi-Feature Fusion and Enhancement Single Shot Detector for Traffic Sign Recognition (2020) [23]</td>
<td>82.6%</td>
<td>88.1%</td>
</tr>
<tr>
<td>Ours</td>
<td>94.71%</td>
<td>90.25%</td>
</tr>
</tbody>
</table>

Future Work

As with any research, there are always areas where further improvements can be made. Below is a list of possible improvements that have been identified and which could be implemented in the future. Firstly, the training dataset could be improved. Currently the dataset has 1221 images and when it comes to neural networks, the larger the dataset, the better the final results will be. Increasing the dataset size will increase detection and classification accuracy for the machine learning algorithm.

Another possible improvement is adding a filtering algorithm after the machine learning detection. It would check for multiple detections of the same traffic sign before returning a
positive detection. This would increase the detection and classification accuracy. Another area where improvements can be made is the OCR algorithm. If lens distortion was taken into account, traffic signs close to the edge of the image would get better detection results. Lastly, detection and classification results could be further improved by adding a gyroscope stabilizer to the camera.

Conclusion

The proposed traffic sign detection and classification system shown in this thesis demonstrates good performance and accuracy when using a neural network in conjunction with a custom OCR algorithm for traffic sign detection and classification. This neural network implementation of Mask R-CNN in Tensorpack shows an average of 5FPS when running it on a GTX 1070. This kind of framerate is sufficient for real time processing and can be improved with a more powerful graphics card. Both the detection and the classification accuracy were over 90% and can be further improved with better filtering. The custom OCR algorithm for digit detection also showed good accuracy and performance. It achieved a classification accuracy of 93.1% and a framerate of 18.5FPS on an Intel Core i7 7700. The task is single threaded so if there is more than one speed limit sign on one image, the framerate will stay the same as long as there are available processing cores. Future directions for this research could be experimentation with different neural networks, writing an algorithm for filtering output data, as well as general improvements to the OCR algorithm.
REFERENCES


[9] Institut Fur Neuroinformatik, German Traffic Sign Benchmark. Available at: http://benchmark.ini.rub.de/


[15] Matterport’s implementation of Mask-RCNN. Available at: https://github.com/matterport/Mask_RCNN


