DETECTION AND LOCALIZATION OF MOTORWAY OVERHEAD DIRECTIONAL SIGNS BY CONVOLUTIONAL NEURAL NETWORKS TRAINED WITH SYNTHETIC IMAGES

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Hakan HEKİMGİL

July 2019
İZMİR

We approve the thesis of **Hakan HEKİMGİL**

Examining Committee Members:

Assoc. Prof. Dr. Yalın BAŞTANLAR
Department of Computer Engineering, İzmir Institute of Technology

Asst. Prof. Dr. Kaya OĞUZ
Department of Computer Engineering, İzmir University of Economics

Asst. Prof. Dr. Mustafa ÖZUYSAL
Department of Computer Engineering, İzmir Institute of Technology

18 July 2019

Assoc. Prof. Dr. Yalın BAŞTANLAR
Supervisor, Department of Computer Engineering,
İzmir Institute of Technology

Assoc. Prof. Dr. Tolga AYAV
Head of the Department of
Computer Engineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

I would like to thank my supervisor, Assoc. Prof. Dr. Yalın BAŞTANLAR, for his patience, support, assistance, and understanding through so many years. Every time I came up with an unusual question or request about my coursework, he was always patient and able to come up with an answer or solution. This thesis would probably not be finalized if it was not for his support and understanding.

I would also like to thank all people in the Department of Computer Engineering in İzmir Institute of Technology. Professors, instructors, assistants, students, staff, everyone turned my late academic adventure into a beautiful experience that has literally changed my life.

Finally, I would like to thank the examining committee members Asst. Prof. Dr. Kaya OĞUZ and Asst. Prof. Dr. Mustafa ÖZUYSAL for their valuable time and effort for their evaluation of this work.

ABSTRACT

DETECTION AND LOCALIZATION OF MOTORWAY OVERHEAD DIRECTIONAL SIGNS BY CONVOLUTIONAL NEURAL NETWORKS TRAINED WITH SYNTHETIC IMAGES

Image classification, object detection and recognition have gone a long way in the last decade. The competitions, starting with ImageNet, have shown that various improving implementations of Artificial Neural Networks are the best Machine Learning techniques at the time for such tasks. However, machine learning methods require much training data and the such data for image related tasks come at a cost in terms of time and effort, if it can be obtained at all. When training data is scarce or not representative of the whole target set, synthetic data and data augmentation methods are used to increase the training data using what is already available.

This thesis work shows that when the target classification images have a structure, even a loose one, it is still possible to use machine learning methods, deep learning in this case, without any real data to begin with and still produce a good detection model.

In this work, a Convolutional Neural Network model is trained to detect and localize informative motorway lane direction signs. Starting with no real samples of the target images, a large computer-generated training set is created to train the model. The resulting detector can detect the required sign types with high accuracy, localizing their position by bounding boxes and categorizing them.

ÖZET

OTOYOL ÜST YÖNLENDİRİCİ TABELALARININ YAPAY GÖRÜNTÜLERLE EĞİTİLEN EVRİŞİMLİ SİNİR AĞLARI İLE TESPİTİ VE KONUMLANDIRILMASI

Görüntü sınıflandırması, nesne tespit ve tanımasında son on yıl içinde epey bir mesafe kat edildi. ImageNet ile başlayan yarışmalar, Yapan Sinir Ağlarının muhtelif geliştirmelerle uygulanmasının, içinde bulunduğumuz zaman için bu tarz görevler için en uygun Yapay Öğrenme yöntemi olduğunu gösterdi. Ancak yapay öğrenme yöntemleri oldukça fazla veriye ihtiyaç duyar ve görüntü ile ilgili işlemlerde bu tarz veriler, temin edilebilseler bile zaman ve çaba olarak oldukça maliyetlidir. Eğitim verisi kısıtlı veya tüm hedef seti temsil edemediği durumlarda yapay veri oluşturulması ve veri çoğaltma yöntemleriyle elde olan eğitim verisinin arttırılması yoluna gidilir.

Bu tez çalışması, hedef sınıflandırma görsellerinin gevşek de olsa bir yapıya sahip olduğu durumlarda, herhangi bir gerçek veri olmasa bile yine de yapay öğrenme yöntemlerinin (ki buradaki örnekte derin öğrenim yöntemleri kullanılmıştır) kullanılabileceğini ve iyi bir tespit modeli oluşturulabileceğini göstermektedir.

Bu çalışmada bir Evrişimli Sinir Ağı modeli, bilgilendirici otoyol şerit yönlendirme işaretlerinin tespit ve konumlandırılması için eğitilmiştir. Hedef görsel işaretlerin hiçbir gerçek örneği olmadan yola çıkılarak, modeli eğitmek için bilgisayar kodu ile geniş bir eğitim seti oluşturulmuştur. Bu set kullanılarak oluşturulan tespit sistemi, istenen işaretleri yüksek doğruluk düzeyi ile tespit ederek, görüntü içindeki konumlarını sınırlayıcı kutular ile işaretleyerek sınıflandırabilmektedir.

DEDICATION

I would like to dedicate this work to my great family: mother, father, wife, daughter, sisters, nephews. May we have many more years to share together.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

LIST OF ABBREVIATIONS

ANNs:	Artificial Neural Networks
CNNs:	Convolutional Neural Networks
CV:	Computer Vision
HIS:	Hue-Saturation-Intensity color space
HSL:	Hue-Saturation-Lightness color space
HSV:	Hue-Saturation-Value color space
ILSVRC:	ImageNet Large Scale Visual Recognition Challenge
IoU:	Intersection over Union
NNs:	Neural Networks
R-CNNs:	Region-based Convolutional Neural Networks
RGB:	Red-Green-Blue color space
RPN:	Region Proposal Network
SSD:	Single Shot Multibox Detector
YOLO:	You Only Look Once

CHAPTER 1

INTRODUCTION

1.1 A general history

Image processing and computer vision has long been a popular field. While general image processing could easily be handled thanks to hard-coded algorithms, image understanding required more advanced methods and computing power.

Initial works involved finding colors, geometric constructs such as lines and circles, and finding invariants in images through specialized algorithms.

Machine learning methods were not used much not only because image libraries were not as common but also because images and videos also carried the burden of manual annotation. In addition, popular machine learning methods were not very suitable, and the computing requirements were high. The world has witnessed to a great leap in the field in less than a decade thanks to the popularization of digital imaging, sharing of images through Internet, advancement in computing power, and most importantly the advances in various methods and techniques in Artificial Neural Networks (ANNs).

The theory behind ANNs was a very old idea but it was unjustly discredited for quite a long while. Various improvements and work [1] on their techniques created a spark for some but were still not enough to popularize them until a model, now dubbed as AlexNet [2], using deep convolutional networks, which is a type of ANN, outperformed others by an unexpectedly wide margin (with almost half the error rate of the nearest competitor) in the ImageNet competition of 2012. In the years following that, still other CNN models kept obtaining the best performances and CNN methods and architectures started attracting everyone's attention especially in CV. In fact, nowadays, almost all entries in such competitions use Convolutional Neural Networks (CNNs).

The size of the datasets and the complexity of tasks grew hand in hand with improvements of the CNN techniques. However, the amount of data still posed a problem as the manual annotation of images was a tedious and costly work. Using data augmentation methods, the data at hand could be reshaped and reused but this also had a limited use. In fact, in some fields, such as learning disparity and optical flow estimation

required pixel-accurate data which could simply not be obtained by human annotation and the ideas of using synthetic data [3] have started to be utilized.

Even when there was available data, combining real data with the synthetic seemed to offer some advantages in modeling.

1.2 Aim and objective

The aim of this work is to show that it is possible to train a CNN with totally synthetic data prepared with a minimal effort for target sets that can vary much but still have a general structure. Motorway overhead directional signs are a good example for such target data and are used as the practical application of this idea.

The resulting model performs well and does not require much data gathering, preparation, or annotation. The testing data, however, naturally requires such work for performance measurement.

1.3 Related work

Computer Vision, Object detection and localization are very general and popular fields. Due to the recent commercial race in autonomous driving, traffic scenes are also very popular. Older work (produced more than a decade ago) on traffic scenes usually concentrated on lane detection and road-side regulatory traffic signs. On the other hand, most of the recent work concentrates on general scene understanding and segmentation towards autonomous driving goals. “Vision-based Road Sign Detection” [4] concentrates on the same real-world application as this work, but uses a low-level color-based segmentation in HSL color space.

There are work that use synthetic traffic scenes from very advanced simulations [5], work that use computer games as simulators [6], work that use synthetic training data for objects in indoor scenes [7] or human 3D pose estimation [8], or work that use GAN-based virtual-to-real scene adaptations but they are all on different tracks compared to this work.

The ResNet50 [9] architecture and the Faster R-CNN [10] techniques are used for modeling a detector DNN in this work.

1.4 Organization of this document

The organization of the rest of this document is as follows: Part 2 gives basic background on some technical concepts. Chapter 3 gives brief information on the generation of the synthetic data for the application. Chapter 4 describes the methodology used for training a model. Chapter 5 discusses experiments and their results. Chapter 6 briefly mentions some possible future extensions. And finally, Chapter 7 presents the conclusions.

CHAPTER 2

BACKGROUND

2.1 Earlier work on traffic sign detection and recognition

Initial works on processing vision in vehicles was more on guidance systems and concentrated on road [11] and obstacle detection as they were the first problems to be solved for starting with vehicle guidance in non-populous areas. As the research moved to autonomous driving in urban environments, other works on lane detection, vehicle detection, pedestrian detection, sign detection began to emerge. New sensor technologies brought detection using a variety of different sensors, such as mono camera, stereo cameras [12], omnidirectional cameras [13], and more recently lidar. Naturally, sensor fusion [14] (combining the information from different sensor types) and use of hybrid cameras [15] have also been tested.

Sign detection usually referred to detecting danger warning signs, prohibitory or restrictive signs, mandatory signs, or special regulation signs as these regulate the traffic. Directional signs were not a priority as the vehicles were not expected to depend on the signs for guidance. Directional signs had more variation in content, but the other signage usually belonged to a finite set of fixed signs. For example, a stop sign did not have any variations. Even the speed limit signs had a limited set of variations due to a limited set of speed limits. In addition, regulation signs had a strict format, unlike any other object to attract the attention of drivers: Round signs with red circles, triangular signs with a red triangle, blue signs with fixed pictograms, etc.

Due to the nature of regulatory signs, color thresholding and segmentation was the usual first step to detect possible positive regions. Shape analysis would come as a likely second step to confirm the existence of signs [16], [17]. Hue-based color spaces were usually preferred for robustness to various outdoors lighting conditions [18], [19]. For shape analysis, outer edges of the signs were usually analyzed with methods such as edge detectors, genetic algorithms, Hough transforms [20]. The step after detection would be classification and neural networks were suggested even before they gained their recent popularity [16], [21], [22] but variations of other methods such as Clustering classifiers,

Nearest Neighbor Classifiers, Laplace Kernel classifiers, fuzzy classifiers [20], PCA, Discriminant Analysis, and SVM [17] were also suggested.

As with other research examples, the research on traffic signs also caused a various number of data sets to be formed such as Belgium Traffic Sign Dataset [23], German Traffic Sign Recognition Benchmark (GTSC) [24], German Traffic Sign Detection Benchmark [25].

2.2 A brief history of CNNs

The ideas of ANNs date back to 1943 when Warren McCulloch, a neurophysiologist, and Walter Pitts, a mathematician wrote an article [26] on modeling how neurons might be working. Donald Hebb's contributions [27] on how neural pathways might be strengthening as they are used paved the way to the basic idea of the mechanism of neural networks. With the advances in computing in 1950s, these ideas could also start to get tested as algorithms. While works on perceptrons, such as that of Rosenblatt [28], were drawing interest along with research funding, the book on the subject by Minsky and Papert [29] had blown a big negative impact on the field. As a result of this impact, the term "Neural Networks" (NNs) had bad connotations for a long time and shadowed the works of Werbos [30] and Hopfield [31].

Rumelhart, Hinton, and Williams [1] created a new spark for the research and applications of in neural networks. Although the computing power and the available data was not ready for a wide-spread use at the time, there were promising works, such as that of LeCun [32]–[35] showing the capabilities of NNs in laboratory environments [36].

Availability of data is also an important factor. ANNs shine at tasks too complex to be manually modeled but collection and labeling in fields like Computer Vision (CV) is very costly, at least in terms of time and manual labor. Competitions like ImageNet [37] provided a great platform both as a massive library of labeled data and as a publicly open arena where the results of various algorithmic techniques can be compared. When a model, now dubbed as AlexNet [2], using deep convolutional networks, which is a type of ANN, outperformed others by an unexpectedly wide margin (with almost half the error rate of the nearest competitor) in the ImageNet competition of 2012, CNN methods attracted everyone's attention especially in CV. So much that, nowadays, almost all entries in such competitions use Convolutional Neural Networks (CNNs).

The size of the datasets and the complexity of tasks grew hand in hand with improvements of the CNN techniques. As more could be done, more would start to be expected. The Pascal Visual Object Classes (VOC) Challenge of 2005 [38] had only 4 classes, 1578 images containing 2209 objects and required classification and detection. While the 2012 challenge had 20 classes, 11530 images containing 27450 Region-of-Interest (ROI) annotated objects and 6929 segmentations. ImageNet had 1000 object categories and started with a classification task in 2010; in 2017, the tasks were object localization, object detection, and object detection from video. Microsoft COCO dataset [39] offered photos of 91 object types with 2.5 million labeled instances in 328 thousand images where objects are labeled in using per-instance segmentations for object localization.

As these competitions provided a common ground for comparison of CNN architectures, the ones with good performance have also been popularized among the community. After the great success of AlexNet [2] in 2012, improvements with new ideas came almost annually with ZF Net [40] in 2013, VGG Net [41] in 2014, GoogLeNet [42] in 2015, and MS ResNet [9] again in 2015.

As more has been started to be expected, new techniques were devised to include new functionalities. While early works started with classification of the entire input image as a single class, the trend moved to detection and classification of a multiple number of objects, then adding localization as well, and then changing localization from bounding boxes to pixel segmentations. Of interest, architectures such as R-CNN [43], Fast R-CNN [44], and Faster R-CNN [10] incorporated object localization; Mask R-CNN [45] added an instance mask for instance segmentation; YOLO [46], YOLOv2 or YOLO9000 [47], YOLOv3 [48], and SSD [49] incorporated a single-pass system for speed improvements; SqueezeNet [50] and MobileNets [51] improved on the computational costs.

As such architectures gained interest and popularity, their pre-trained versions have started to be available. This not only saved time by eliminating the need to reconfigure network layers from scratch for new projects but it also helped with the training process as well due the concepts of domain adaptation and transfer learning [52], [53]. where a pretrained network is used at the start and the training adapts the results to a different target data set with beneficial results [54]. With the introduction of platform exchange formats, such as ONNX [55], architectures and pre-trained data might be used even more freely in the near future.

2.3 Some datasets for traffic scenes

The advances in CV also made its impact on other fields that benefit from it, such as medical diagnosis and autonomous driving. While this may have driven the use of different types of sensors and sensor fusion in areas like autonomous driving, possibly causing them to slightly linger away from CV, it has also supported the production of various other data sets and competitions. Some traffic scene datasets are Leuven [56], CamVid [57], Daimler Urban Segmentation [58], KITTI [59], [60], Cityscapes [61], [62], Oxford RobotCar Dataset [63], Mapillary Vistas [64], ApolloScape [65], Berkeley Deep Drive [66], and nuScenes [67], as well as the synthetic data set Synthia [68].

2.4 Popular network architectures

Although they look very simple now and not used at all, LeCun's various "LeNet" networks must still be mentioned as an early architecture that used CNNs. LeNet-5 had only 7 layers, 3 of which were convolutional. And the paper on gradient-based learning [34] is considered to be a pioneering the field.

The following networks have made their popularity within very short time periods among each other and have helped the fast evolution of the CNN architectures.

AlexNet [2] had 8 trainable layers, of which 5 were convolutional and 3 fully connected. But it also had ReLu activation, data augmentation, dropout, overlapping pooling, and local response normalization.

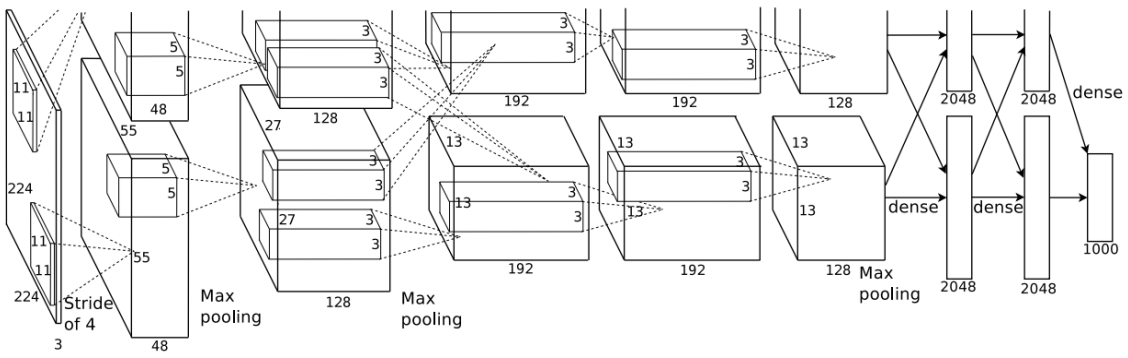

Figure 1. Layer diagram of AlexNet (Source: [2])

ZF Net [40] in 2013 improved on AlexNet architecture by using 7x7 filters instead of 11x11 ones and using a stride of 2 for the convolution instead of 4 to retain much more information for the further layers of the architecture.

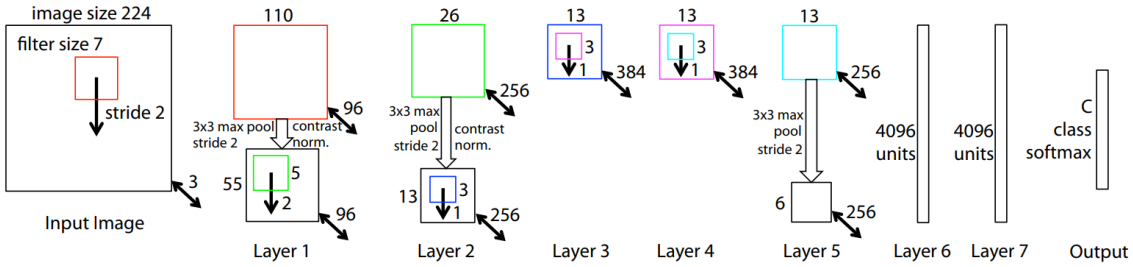

Figure 2. Layer diagram of ZF Net (Source: [40])

VGG Net [41] in 2014 was a simple in that, it decreased the CNN filter sizes further down to 3x3, but it was also deeper than its precedents, with 19 CNN layers.

GoogLeNet [42] in 2015 went even deeper with a 22 layers when counting only layers with parameters but the overall number of layers were around 100. It introduced the “inception module”.

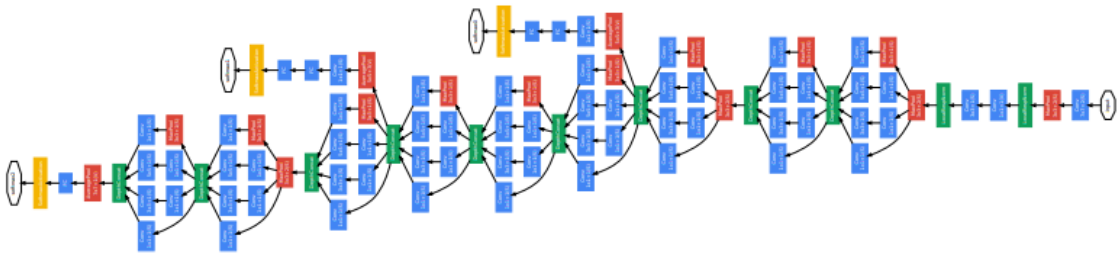

Figure 3. Layer diagram (displayed sideways) of GoogLeNet (Source: [42])

MS ResNet [9] in 2015 introduced “identity shortcut connections” and offered different versions with varying number of levels with 50, 101, and 152 being popular sizes used in following work by others.

As the network architectures got deeper, the issue of model size, memory size and other computational costs have also started to gain an increasing focus [69]; SqueezeNet [50] and MobileNets [51] have provided some good solutions in such matters.

ResNet50 architecture is very popular for various CNN-related work so it is used for the feature extraction part (until the final stages of the DNN) in this work.

2.4.1 ResNet-50

As mentioned previously, ResNet [9] was one of the successful improvements on the CNN structures. This is mainly because it could use a deep layered network through a new structure it introduced: identity shortcut connections or a residual network.

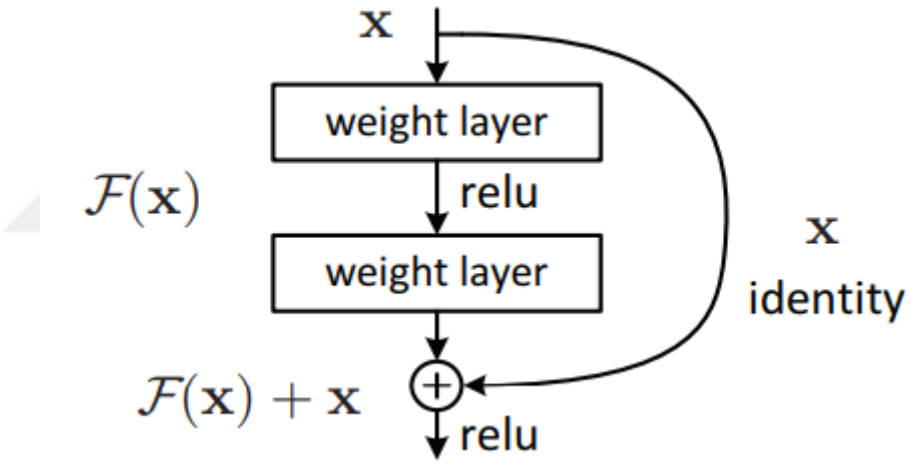

Figure 4. Residual learning: a building block from ResNet (Source: [9])

The CNNs seemed to improve as they get deeper, but they also experience other problems with the depth so an "infinitely deep" structure is not possible. Some of these problems are vanishing gradients and degradation problem. CNNs calculate gradients and use backpropagation to update the weights in the network; however, as the number of layers increase, the updates for the earlier layers can get smaller and smaller. This is called vanishing gradient. In addition to that, as the network gets deeper, every layer introduces

extra training error and the number of parameters to be optimized get larger; this degradation in training accuracy is called the degradation problem.

The idea of residual learning adds identity shortcut connections every few layers forming “residual modules” and they help improve on both of these problems. These shortcuts allow better backpropagation of gradients and the weight calculation is eased as it is no longer necessary to compute for a long line of layers but only compute the residual weights alongside the identity shortcut connection.

Networks of various depth are tested with this architecture and the ones with 50, 101, and 152 showed a good level of accuracy. The 50-layer version, referred to as ResNet-50, has become popular for as a base for the initial feature extraction layers.

2.5 Network architectures involving detection and localization

R-CNN [43], Fast R-CNN [44], and finally Faster R-CNN [10] have become popular CNN architectures for detection and localization. YOLO [46], YOLOv2 or YOLO9000 [47], YOLOv3 [48], and SSD [49] enabled real-time performance through their “single-pass” systems.

Although a real-time implementation would make more sense for the practical application of sign detection, this works focuses on the theory of use of completely synthetic data so Faster R-CNN is used for ease of implementation to obtain the results. A YOLO implementation is being considered for future real-time testing.

2.5.1 Faster R-CNN

Although all three versions of R-CNN algorithm group are mentioned together, they were introduced within short intervals. They are, in order, R-CNN, Fast R-CNN, and Faster R-CNN. The last version, Faster R-CNN, is the most improved one.

Faster R-CNN has become a very popular structure for performing classification and localization together when time is not a special concern. As mentioned above, although Faster R-CNN is not suitable for 30 frame-per-second real-world videos, it can still serve as a proof-of-concept, showing that the main ideas of this paper do work. Again, as mentioned above, a future YOLO implementation can be prepared for real-time video inputs.

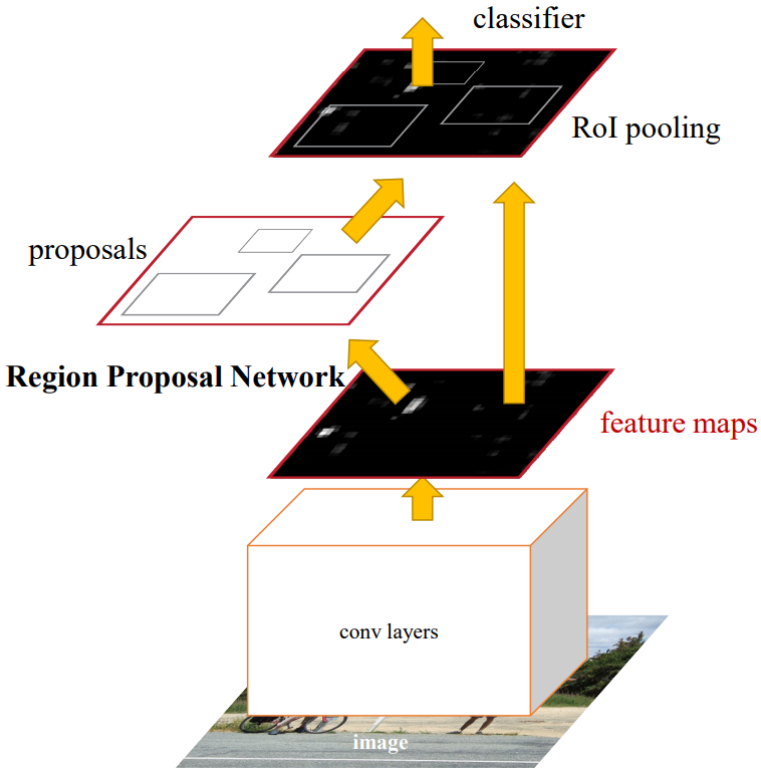

Figure 5. Structure of a Faster R-CNN network (Source: [70])

The main idea of R-CNN methods is the use of "region proposals". Previous methods used techniques like sliding windows of various sizes which took much time. Both of the initial R-CNN versions used "selective search" algorithm. R-CNN proposed a certain number of regions which it cropped, resized, and passed to the CNN for classification. Instead of feeding the regions separately, Fast R-CNN moved the region proposals to a further stage; it would first pass the whole image through the CNN and then process the region proposals, avoiding multiple processing of overlapping parts of proposed regions.

Faster R-CNN [10] further improved on this by abandoning the selective search algorithm which slowed down the process and obtaining the region proposals through a separate but parallel Region Proposal Network after the feature maps and before the classification.

Although, there have been huge speed improvements to classification with localization techniques, most importantly with YOLO [48] and SSD [49], it is a matter of discussion how these improvement may have affected classification performance. There

have been suggestions on both the original papers and third-party review papers [71] but it is hard to make a definitive comparison due to differences in the original platforms, comparisons through different data sets, and possible implementation differences.

2.6 Use of synthetic data

CNN models improve with the amount of training data; however, the production of ground truth data for image classification problems is a very costly (at least in time and effort) manual process. This led the consideration of the use of synthetic data and augmentation methods of the available data.

Several different methods of use of synthetic data are proposed. Mayer et al. [3] stress the importance of synthetic data for learning disparity and optical flow estimation, since they require pixel-accurate data which cannot be obtained by human annotation, and suggest multiple ways of generating such data. The Synthia dataset [68] proposes using totally synthetic urban scenes images in combination with other publicly available annotated urban scenes for semantic segmentation. While Tremblay et al. [72] propose training a model with synthetic data with non-realistic random domain parameters and then fine-tuning it on real data, Hinterstoisser et al. [73] go the other way by proposing using a network pre-trained with real images and freezing its initial feature extraction layers and then training the following layers through synthetic images obtained by OpenGL rendering. As mentioned previously, there are also work that use synthetic traffic scenes from very advanced simulations [5], that use computer games as simulators [6], that use synthetic training data for objects in indoor scenes [7] or human 3D pose estimation [8].

2.7 Data augmentation

As stated in the previous section, CNN models improve with the amount of training data but the production of ground truth data for image classification problems is costly. This led the consideration of the use of synthetic data and augmentation methods of the available data.

Data augmentation involves using the data at hand to produce more data that fits the real-world situation for obtaining a larger training set. It may have a longer history

than synthetic data generation. Various methods are used for oversampling of unbalanced numerical data in data analytics.

In image classification problems, data augmentation is even more important because the amount of training data is very important, the collection and manual annotation of data is costly, and data augmentation also help with generalization (avoiding overfitting) of the data. As an additional bonus, data augmentation of images are relatively simple operations that do not change the integrity or correctness of the data, such as horizontal or vertical flipping, scaling, noise, rotation, cropping, shear, warp, etc. GAN-based data augmentation methods are also being proposed increasingly. In fact, previously mentioned superior-performing networks such as AlexNet [2], VGG [41], GoogLeNet [42], and ResNet [9] all used data augmentation methods, turning this into standard practice.

There are also two popular alternatives in standard practice for incorporating data augmentation: in preparing the training data set (offline) or during training (online). These two alternatives provide a trade-off between dedicating more memory for the training set versus dedicating more time and computing resources during the training phase. Image data augmentation methods are becoming part of the popular deep learning libraries and platforms.

2.8 Color representations in digital images

There are have been various color representations (or color spaces) used for digital images. Different representation types can be selected not only for efficient storage purposes but also for enabling different types of analyses and manipulations.

2.8.1 RGB

RGB is probably the most popular method by far. It is based on the human perception of color. Colors are represented by a combination of separate levels for Red, Green, and Blue components. A mixture of various levels of these values would produce the desired ouput.

2.8.2 HSV

HSV color space is based on the Hue, Saturation, and Value parameters where the hue represents a certain point in a circular color scala, saturation represents the shade of a bright color (hue) and the value roughly represents a mixture of white or black.

2.8.3 Other

Some other popular color representations are HSL (Hue, Saturation, Lightness), HIS (Hue, Saturation, Intensity), HCL (Hue, Chroma, Luminance), CIE L*a*b* (Commission Internationale de l'Éclairage, Lightness*, a*, b*).

Color spaces using Hue, such as HSV and HSL, are sometimes used in color-based algorithms since the hue value is considered to be less affected by lighting conditions (illumination, shadows, etc.) and easier to work with as a single hue value can be tested for selecting areas with a certain color. They can be used for selecting a certain group of image pixels corresponding to the desired hue values, such as the green background of motorway signs, without being concerned on how various lighting conditions can change.

2.8.4 Alpha (transparency) channels

Sometimes, an extra layer of information (as a fourth layer or a separate layer from the 3-layer color channels) is used if an image is to be used over another image or background.

2.9 Morphological operations

Morphological operations in CV are used for changing, selecting, or segmenting the pixels in a digital image according to its neighbors and through a pre-defined shape, called a structuring element.

Some of the basic morphological operations are erosion where larger CCs can be “eroded” into smaller ones, dilation where smaller CCs can be “dilated” into larger ones, opening, and closing. Morphological “opening” operation is an erosion operation

followed by a dilation operation. Morphological "closing" operation is a dilation operation followed by an erosion operation.

These operations are usually done in pre-planned groups in manual handling of selection of certain groups of pixels and are usually followed by a grouping of "connected components" (CCs) in the end. They can be used, for example, to fill in the impurities or other elements inside a desired area, such as filling in the spaces of letters inside a selected rectangular sign area.

2.10 Traffic sign conventions

Although various international conventions have been used, such as the United Nations ECE's "Convention of Road Signs and Signals" of Vienna, 1968, to provide a degree of uniformity of traffic signs, there are still a noticeable degree of variation among different regions and countries. Even countries that seem to have agreed on a common standard, such as those in the European Union, can still have varying degrees of differences in their signs. Even the categorization of traffic signs varies from country to country.

2.10.1 A general look

Vienna Convention on Road Signs and Signals had 52 signing countries by 2004. This convention categorizes traffic signs into eight categories, of which some are: danger warning signs, prohibitory or restrictive signs, mandatory signs, special regulation signs. The category that relates to this work is the "direction, position, or indication signs".

2.10.2 Motorway and non-motorway directional signs

As the Vienna Convention did not set any standards for direction signs, they have the most variations among different countries and regions.

Similarly, motorway signs also have a large variation. As an example, considering the region we are in, directional signs in motorways use:

- White on green in Turkey, Greece, Bulgaria, Italy, Switzerland, Denmark
- White on blue in Germany, France, Austria, Spain, Portugal, Norway

In non-motorway signs, there is even a greater variation:

- White on blue in Turkey, Greece, Bulgaria, Italy, Switzerland
- White on green in France, Portugal, United Kingdom
- Black on yellow in Germany, Norway
- Red on white in Denmark
- Black on white in Spain

2.10.3 Overhead motorway directional signs in Turkey

This work concentrates on a very small part of the traffic signs scheme to test the CNN training by synthetic data. That part is the overhead motorway directional signs in Turkey. In motorways we usually see a road-side sign as the one on the right to indicate the forward directions and the exit directions for the motorway before we come to the exit. We can then see an exit sign, again on the side of the road, indicating where the exit leads. We are then likely to see an overhead sign indicating the continuing lane directions and the exit lane directions.

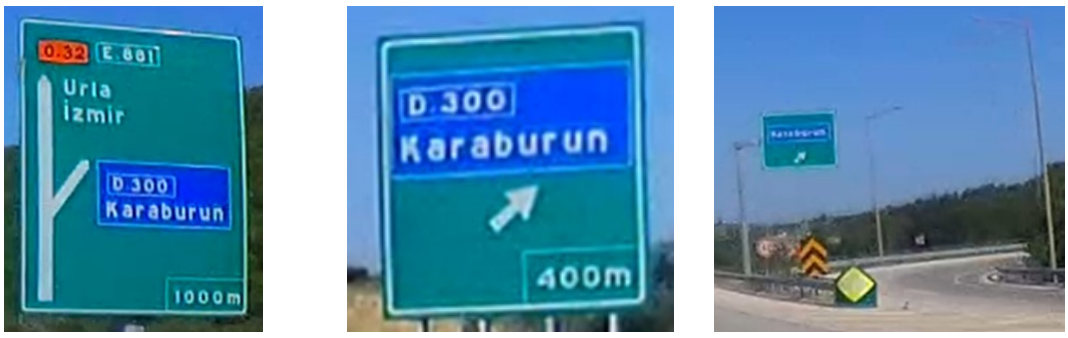

Figure 6. Motorway directional information signs (two are road-side, one overhead)

We can yet have another sign at the exit point indicating the direction the exit will lead. As can be seen in this example, although the main motorway signs are standardized as white on green background, various other elements of different colors can also be present. In this example, the continuing motorway is indicated with an orange colored motorway code of “O. 32”. The exit leads to a non-motorway road which is indicated by

white text on a blue background. Similarly, the highway code “D. 300” is also indicated by a design of white text on a blue background. There can even be other directional information such as black text on white background for city roads or white text on brown background for touristic places.

This signage can be interesting because it is not fixed; in a way, it has a certain amount of variation in each sign, including names, backgrounds, road codes, and even the number and type of arrows used. On the other hand, it does have a certain structure as described above: green background, arrow signs, text, and more text with possible backgrounds of green, blue, white or brown.

Figure 7. Motorway overhead directional information signs

2.11 A brief overview of color-based method used for comparison

The main idea is “masking” the pixels that contain the possible color ranges the target signs may contain, doing some morphological operations to turn this “mask” into possible areas, and going over those areas separately to determine the possibility of them containing the target signs.

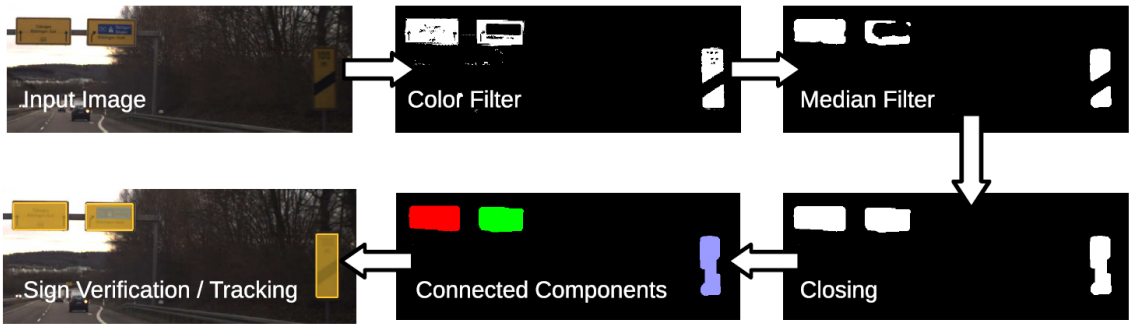

Figure 8. A sample detection pipeline for road signs (Source: [4])

2.11.1 Color ranges to be scanned for

First of all, the RGB color space representation is converted into HSV color space. The hue values allow us to scan for a certain color without worrying about its lighting conditions or other variations such as shadow falling on them.

Unfortunately, this is a more tedious process than it seems. First of all, unfortunately there is no single color to scan for. Some countries do follow a "white text on a certain color background" strictly, Turkey is unfortunately not one of them. Although the motorway signs are defined as "white text on green background", some of the text (name of places as directions) are written on different colored backgrounds on that sign. So the green background can have other rectangles with different colored backgrounds of blue, brown, and white.

The addition of white background complicates things even further because: 1) white is not a color and it cannot be scanned by a hue range, and 2) the white background also introduces black text, again not a hue-ranged color.

Furthermore, road codes can be placed on the signs and they add another new color: orange.

So, to sum up, a color-based scan for Turkish motorway signs would include: 1) green, 2) blue, 3) brown, 4) white, 5) black, 6) orange.

"Real" colors (green, blue, brown, orange) are mainly scanned by a hue (H) range but the saturation (S) and value (V) ranges also need to be checked. There are different areas of the HSV color space that are described as the chromatic area, unstable chromatic

areas, and achromatic area [74]. The H value does not guarantee by itself that a pixel is of a certain target color.

Similarly, white and black are not chromatic colors. They can be better scanned by V and S values while they can contain a random H value.

So, a number of masks are produced, at least one for each color, even more for other colors. For example, since green is the most common and important color, 5 different masks are prepared and combined together for detection.

2.11.2 Problems with color scanning

Perhaps the most important problem is that certain colors ranges can be found in a number of places in a motorway scene image. A clear blue sky is very likely to match our blue scans for the signs. A distant view of sea can also cause problems. Similarly, the greenery on the side of the motorway can trigger our green scans. Different shades of white and black can be found in even a larger number of regions but luckily, most of them would be smaller regions.

2.11.3 Image processing and morphological operations

After the color masks are determined an initial filtering would be done based on the location of the pixel masks. Then to get a clearer group of points median filtering would be applied. To turn the separate pixels groups into a more robust blocks, morphological operations would be applied. A “morphological closing” operation would result in block of masked pixels.

2.11.4 Connected components and final processing

The resulting blocks of pixels from the previous step would be turned into separate connected components. A great number of connected components are usually found at this stage. Various reasonings can be used to filter them. The most obvious ones are by size, location, and aspect ratio. Then more intricate reasoning methods can be applied such as the ratio of specific colors in each connected component, the rate of color

variations at the edges of the determined regions, distributions of the locations and sizes of remaining connected components, etc.

At this point, we would have a number of connected component block which can easily be represented by rectangular regions or bounding boxes. A separate classification algorithm would still be needed at this stage both to make sure that the bounding boxes contain a sign and to classify what that sign is. To achieve a more robust system, a tracking algorithm can also be applied to smooth out any residual differences that may result from the previous steps.

CHAPTER 3

GENERATING THE DATA SET

3.1 Real images used

Although it is stated that the training set is created using no real data, it must be noted that the background images used for creating the training set, as can be seen below, are actually real images from a dashboard vehicle video camera but they do not contain any road signs at all.

Figure 9. Some examples of the images used as background for the synthetic data

There are 113 such background images with 1920x1080 resolution, all selected from frames that did not contain any road signs and all from a single video recording file recorded in July 2017. That particular recording is selected because it was taken on a part of the motorway where there were not many signs for a long time and because it contained a clear sky, road side greenery (although it was not really green due to the extremely warm July weather in İzmir), and some partial view of sea scenery from far. This choice was due to the fact that clear skies, greenery, and sea can be problematic for color-based selections. Below are some examples of the background images used.

The background images could also have been synthetically created too or otherwise, copied from elsewhere. However, such images are easy to obtain, and they do not clash with the claim of "no real data" since they do not contain the targeted positive class sign images. The signs, which are the actual focus of training for detection, are completely synthetic.

3.2 Elements used for creating synthetic signs

Synthetic signs are created by mixing together different visual elements such as coored backgrounds, text, arrow signs, road code signs, etc.

3.2.1 Motorway sign background

MATLAB code is used to produce a green sign background with a white border. The sign is generated at a specific size (200x200) which can later be changed to any desired size.

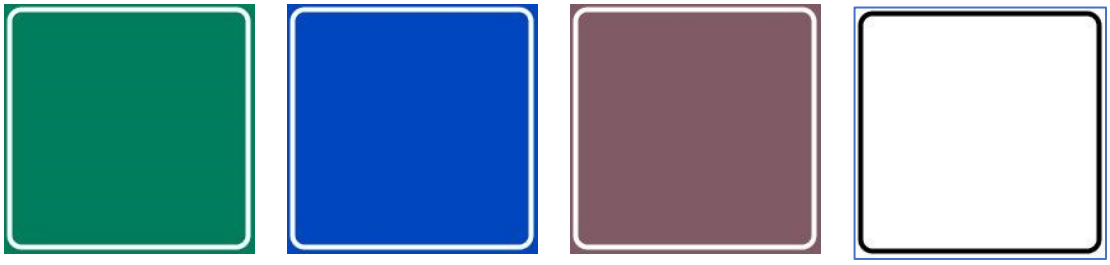

Figure 10. Computer generated sign backgrounds

3.2.2 Other colored sign backgrounds

Similar to the green background, other sign backgrounds are prepared in blue, brown, and white. These backgrounds can be used according to the status of the road leading to a location. In Turkey blue is used for non-motorway roads out-of-city, white is used for roads within the city limits, and brown is used for touristic destinations.

Note that an extra black border is added to the white sign box on the right to show its borders in this document; the original sign does not have an extra border.

The sizes are, again, set to 200x200 to be resized again according to need.

3.2.3 Text for place names

In order to make the synthetic signs look close to the original ones, various fonts are searched on the Internet. A font named “Overpass Bold” seemed to be close enough although the spacings between letters were somewhat tighter than needed. Deep learning architectures are often considered as “black box” functions where you cannot see what happens inside and how the data is used. Thus, it is not clear how the similarity of the place name fonts affect the results of the detection.

3.2.4 Arrow signs

The bottom-pointing arrow signs for continuing lanes are used from another font set specifically created for British road signs: the “Transport” font.

However, the arrow sign for exit lanes had to be recreated using PowerPoint.

3.2.5 Road code signs

Occasionally, there are road codes put on the signs such as O.32, E.90, D.550. Although these could as easily be generated through code, their synthetic versions already available on the Internet were used for simplicity.

abcçdefgğhıijklmnoöprsştuüvyz 1234567890

ABCÇDEFGĞHIİJKLMNOÖPRSŞTUÜVYZ

Figure 11. Letters and numbers printed with the "Overpass Bold" font

Figure 12. Arrow sign from the "Transport" font

Figure 13. Arrow sign produced with Microsoft PowerPoint

Figure 14. Synthetic road code sign samples from Internet

3.3 Sign generation

For the continuing lane signs, the standard green box gets loaded, then its width is changed to a random number within an acceptable range. The height is kept constant at 200 pixels for ease of application since the size of the whole sign will later be scaled anyway. Then with random weights, its number of lane markers (1, 2, 3), number of columns (single or double), the number of text lines (place names) for each column, number of rows for each column (1, 2, 3), and background box colors, if there will be any, are decided.

Figure 15. Computer generated synthetic signs for continuing lanes (class 1)

The place names are randomly selected from a list of place names in the İzmir region with a chance of generating a random text as a place name to avoid overfitting text content. Random text starts with a capital letter and is 4 to 10 characters long.

Signs for exit lanes are generated in a similar manner except there are four initial exit arrow configuration possibilities along with their appropriate text locations.

Besides the width, the locations and sizes of text, text areas, arrows, and road code signs are also varied randomly within certain ranges.

Figure 16. Computer generated synthetic signs for exit lane (class 2)

The two types of signs are generated separately but combined together by placing them next to each other as they usually are placed in motorways. Then the transformations explained in the next section are applied equally to both signs for data augmentation. A random distance is placed between the signs and this distance is defined as a transparent area.

3.4 Image transformations used for data augmentation

As mentioned earlier, there are many image transformations that can be used for data augmentation. These transformations are applied only to the generated sign images and not to the background image. The most popular transformations are usually horizontal and vertical flipping but since they are not used as they are not relevant for this case. The transformations used are as follows:

3.4.1 Scaling

Scaling is done as a whole and equally on both axes and according to the placement on the view. If the image would be placed closer to the vertical center of the view, then the sign would be scaled smaller as if it were farther away and closer to the horizon. Conversely, if it would be placed higher up on the view then it would be scaled

larger as if it were close to the camera and viewed by a vehicle that was almost about to passed underneath it.

3.4.2 Color space variations

In order to generalize the detectors and make them more robust, some variations are applied to the channels in the HSV color space components.

3.4.3 Gaussian noise

This is made to cause a degree of variance in the image to avoid overfitting. However, the amount of noise used is very low.

3.4.4 Rotation

There is a limited amount of random rotation within the range of 10 degrees is applied to cover for possible misalignment of the camera holder and lens distortion to a degree. The starting degree of the range of rotation is determined according to the positioning of the sign in the view, which is also random within a range function; this is, again, for simulating the lens distortion to a degree.

3.4.5 Shear (Horizontal)

A random amount of horizontal shear is applied within a limited range to simulate the possible effects of perspective from lens distortion. Although lens distortion is quite non-linear and bends the lines and borders out of shape, applying a true lens distortion effect would require much effort with a limited return. The range of shear is, again, determined according to the positioning of the sign in the view.

Vertical shear is not used; however, it could be used in future versions since it might be helpful in simulating the lens distortion with a minimum amount of computing power.

3.4.6 Darkening

To account for various differences of lighting, varying degrees of darkening is used. This is done by converting the image into HSV color space, toning down the V value randomly, and converting the image back to RGB color space again. As the model is planned to be run in normal lighting conditions, making images lighter is not considered. Darkening can be especially useful if the light source (sun) is at an angle in the front, facing the camera, making the signs look darker.

3.4.7 Cropping

Cropping is not really applied as a transformation, although it might have been useful for training the detector for the conditions of obfuscation. Parts of the image signs are cropped only when those parts fall out of the visible area of the camera due to the random positioning.

3.4.8 Transparency

There are two situations where transparency conditions are taken into consideration. One is due to combining two signs, a continuing lane sign and an exit sign; the other is due to rotations. Rotations usually involve a larger image to fit the rotated original, filling in the extended parts with black pixels or they require keeping the same image size and cropping off the rotated parts falling out of view. The former option is used for rotations, with a transparency mask is used for the black fill-ins when the sign images are patched on to the background. Although a similar situation exists in the case of shears, transparency mask is not used for shear transformations for simplicity.

3.5 Training set

The background images and the video camera recording resolutions are 1920x1080. For each training sample, two random sign generators, one for the continuing lanes and one for the exit lane, are generated, placed together with a random range of separation, assigned a random location within an area, passed through various data

augmentation transformations as explained above, and finally patched on to the selected random background image. The final image, however, is reduced to a 960x540 size for faster processing and to avoid GPU memory problems during training.

As will be explained in the next section, Faster R-CNN architecture is used to train for both classification and localization. This requires that both the classifications and the localizations within the training image must be provided. Since the images are patched by a constrained random placement function, which means that images are placed according to where they could be expected, the localization of each class of sign is also known. One important thing here is, of course, keeping track of the size and localization changes as various transformations are applied before the patching. For every image generated and patched, the data for the classification and localization (in terms of the upper left corner coordinates, width and height) are also recorded in a training data table.

Not shown in the figure is the addition of the orange road code signs as the background. There are some signs that are very similar to the two defined classes but do not belong to either. They can naturally be detected as a target class due to the similarity to them and the dissimilarity to the background patterns. In order for the detector to learn a distinction, such negative sample should be introduced in the background of the synthetic training data. In this case, separate orange road code signs on top of the directional information signs could have been identified as a target class since they can be observed on the target class signs. Putting them in the background parts of synthetic images have helped us prevent that.

Figure 17. Examples of synthetically generated training data

CHAPTER 4

METHODOLOGY

Although the main contribution of this work is in showing that a computer-generated data set without any real data can be used for training a successful CNN model for detection and localization for a somewhat constrained target set, knowing the methodology that follows for training a detector can also be useful not only for understanding and reproducing the work but also for examining how the data generation parameters affect the results of the final detector.

Below are some information on the methodology and how things are set up, along with some of the decisions made during the set up and the reasoning behind them.

4.1 Picking a CNN architecture for the final functionality

The CNN architecture should be chosen according to various aspects of the project such as the final output required, performance, speed, available computing power, available memory, processing speed, etc.

As classification and localization for images are required, it would be wiser to choose an architecture with localization output. As we require simple localization and not object segmentation, architectures with simple bounding box localizations would be adequate.

In a real-world application for classification and localization from video, processing speed of the architecture would also be important; however, since the main goal of this work is presenting the possibility and practicality of a synthetically generated training data, the speed will not be stressed.

As mentioned before, some of the choices with bounding box localization are R-CNN [43], Fast R-CNN [44], Faster R-CNN [10], YOLO [46], YOLOv2 or YOLO9000 [47], YOLOv3 [48], and SSD [49]. While the YOLO and SSD algorithms also provide the added advantage of real-time processing, Faster R-CNN [10] is chosen due to familiarity for testing of the idea and the popularity of the technique.

4.2 Planning the feature-extraction layers

There have been various CNN implementations that achieved good performances on well-known competitions and data sets. Most popular ones have been offered as pre-trained, ready-to-use models. Not only using such pre-trained models can help save time and efforts by going around the need to design an CNN architecture from scratch, but they can also help with improving the training phase if the features to be detected for the new application is somewhat similar to those of the pre-trained model. Therefore, it has started to turn into standard practice to pick a well-performing pre-trained model, use a wide (deep) range of its initial layers as a feature-extraction network, and make the necessary changes after the so called feature-extraction layer to convert the model into the desired architecture.

As mentioned previously, for this work, the popular ResNet50 architecture is selected and a pre-trained ResNet50 network was used as the starting point.

4.3 Adapting ResNet50 to Faster R-CNN

The figure shows the original blocks of ResNet50 network in lighter blue and the modifications in dark blue, red, and green.

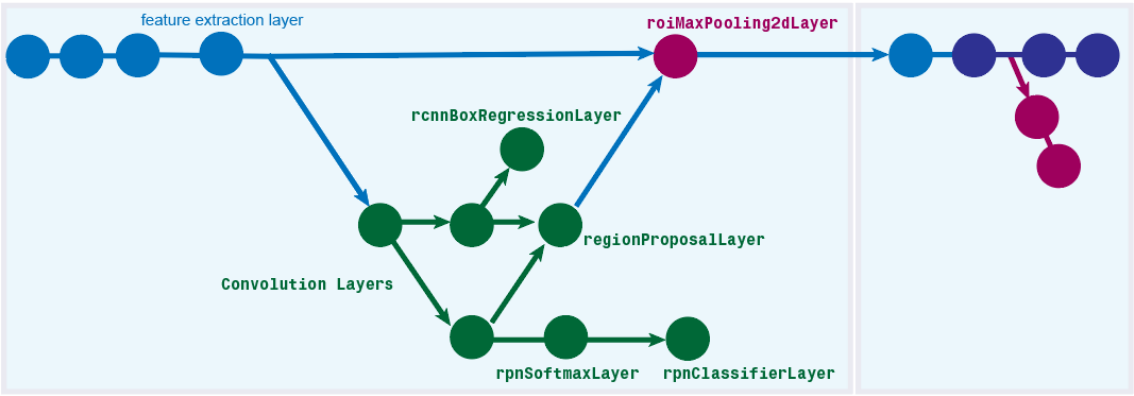

Figure 18. Adaptation of ResNet50 into Faster R-CNN (Source: [75])

As is the usual process in using a pre-trained network for transfer learning, the three last layers of the network (fully connected + softmax + classification) are replaced with new ones of the same type. These are shown as the dark blue layers in the figure.

Next, box regression layers are added for the localization output. These are shown in red (the two at the right) in the figure. In addition, an roiMaxPooling2dLayer is added after a feature extraction layer. This added layer is shown as the single red layer on the left in the figure.

Finally, the Region Proposal Network (RPN) is added. RPN is a specialty of Faster R-CNN and is used to have the network suggest the region proposals instead of getting them from an external algorithm, such as the selective search, which would slow down the process. These are shown as the green layers in the network. They initially take an input from the selected feature extraction layer, they have a feed connection to the roiMaxPooling layer, and they also have two outputs, a classification layer classifiying each anchor as a specific object or background and a box regression layer predicting 4 box offsets for each layer.

4.3.1 Problem with anchor boxes

An important parameter while adding these layers to adapt the ResNet architecture into a Faster R-CNN one is the anchor boxes. These are used to increase the speed and efficiency of the detection process and define the expected scales and aspect ratios of the objects. The RPN uses the given anchor box sizes in creating the region proposals.

One problem in this work is that there is no real data to start with. General practice is scanning through the training data to see what sizes of bounding boxes there are for the objects and then use an algorithm, such as k-means clustering, to decide on the number of anchor boxes and their sizes that would cover well the data in training set, assuming that the training set is also a good representation of the real data.

When using synthetic data, if the model that generates the synthetic data is a good representation of the real world then this would not be a problem. When the synthetic data is generated with the hope that it would generalize well to the real-world situations, which is the case in this work, then the anchor boxes become another parameter to be guessed manually.

Another problem is the scales to be used for anchor boxes. When detecting road signs from a moving vehicle, we have a wide variety of scales for the signs. When the signs are far away, they are very small. As one gets closer, the size increases. As one passes by them, they may be as wide (for the overhead signs) as the width of the view or almost as high (for the road-side signs) as the height of the view. This calls for a very wide range of anchor box scales. When different aspect ratios are added, the number gets multiplied.

One possible point of elimination is using a criterion for the desired size for the signs to be detected. When the signs are too far away, detection and classification would not be possible for man or machine; furthermore, even if the sign can be detected and classified, it would not be very useful if the information (text) on the sign is not legible. That is to say, even if we can see that there is a directional information sign far away, it is no use for us if we cannot read the text that indicates where the direction leads. In a similar manner, if we have already read the information on a sign from a distance, there is really no point in still reading it again over and over as it gets closer. That is to say, if we have read where a direction leads, we do not really need to read it again as we pass it. Thus, using a size criterion, such as the minimum or maximum pixel size for the height or the width of a sign, for ignoring very small (very far) or very large (very very close) signs makes sense and we can use this criterion to limit the scales of the anchor boxes to use.

One other problem may arise when the size of the input images for the training set and the real-world data do not match. As will be explained in the next section, this might happen due to hardware limitations, such as the GPU memory, during the training phase of a CNN. If we use smaller images for training, we should consider how this would affect the resulting detector. We should also decide whether we would use the original image sizes of the inputs or resize it to the size of the training set.

We should also consider the object sizes of the synthetically generated data. If the original input is rescaled to a smaller size, then the detector will need to detect smaller targets in the new scale. If the input data is not resized, then we run a risk of training with a different size and detecting with another.

Furthermore, since the training of a CNN takes a rather long time, we might not have the liberty of trying a wide range of options to empirically find ones that produce a good result.

4.4 Training set

The training set had to be prepared according to the final goal and the planned training procedure.

By “final goal”, it is meant what signs are desired to be detected, what signs are intended to be treated as background, and finally, what signs are not relevant, meaning that it does not matter if they are detected or not.

To give some examples, the overhead motorway signs are intended as the target classification groups with only two types. Sometimes, there are small orange signs on top of these marking the road codes; these would be intended to be treated as background and not signs. Similarly, any orange road code signs within the intended overhead directional signs are not to be detected separately on their own. Finally, there are other motorway signs on both sides of the road; these are signs that are not relevant. So, whether they are detected or not, they would not count as a positive or negative detection.

How to treat the first group is obvious: examples of this group are placed in the training set and marked by their classification and location for the training. The last group is also somewhat easier to handle. No care is taken to see that the model would detect them and similarly, no care would be taken to see that they would be treated as background either. They can be left out of the results in the test set, without having them affect the results positively or negatively. The second group is a bit more complicated. If there are signs that we certainly do not want the model to detect as an object, we need to provide examples of them in the training set where they are not marked as the object so that the model would be trained to ignore them and treat them as background. This method has been put to test in a later version of the synthetic data. The model trained with the earlier version with no negative examples would rarely pick up the orange signs on top of the intended directional signs, as well as, again rarely, the orange code sign within the intended directional signs. A later version of the synthetic data generation code started placing some orange signs in the background images randomly and this helped the model to distinguish them as background when they are on top of other signs and as not separate objects when they are on the intended target road signs.

Another design issue with the training set generation is the size of the training images and the size of the classification objects on those signs. As will be mentioned in the next section, hardware limitations might require us to use smaller sized images for

being able to train our model with the GPU memory we have. Having the training set and real-world input data images in different sizes would lead us to consider various other factors such as the anchor box sizes and the minimum dimensions of the signs that need to be detected. This would in turn lead us to consider the sizes of the target objects that would be used in the synthetically generated training data. For example, if we do not intend to detect sign smaller than 50 pixels in width, we would arrange both the anchor boxes and the synthetic signs that way. But if we are required to use half the image size for training then we would need to either include half the minimum size of the anchor boxes for the training or we would not use twice the minimum size criterion so that they would not be smaller than the minimum size when the image is reduced in size for the training.

Other important factors in the test set can be the color variations in the background and in the objects, making sure to include in the background other image patches that might be falsely detected as objects

4.5 Training

The hardware capabilities are almost always a limiting feature in Deep Learning. Computer memory and, even more often, GPU memory limits the amount of data that can be processed. As such, the GPU array were not enough to train the network with the desired parameters. The first resort to remedy the situation is usually decreasing the input size and the second is usually decreasing the mini-batch size.

Various parameters were tested to see if the process can go through with the training by using such parameters. The training would be run with a small number of epochs such as 2 or 4 and if the training would complete without any memory errors then it would be run with longer epoch parameters such as 20 or 30. Two combinations that worked with the available hardware were:

- ¼ input image size (480x270 pixels) and a mini-batch size of 8
- ½ input image size ((960x540 pixels) and a mini-batch size of 4

There is no set limit on the mini-batch size. It seems that various opinions indicate that they should be in the range of 1 to 32. Both mini-batch sizes seem to work fine in different training trials.

The image size, however, is a different matter. When ¼ reduction is used, we have to plan the positive sample sizes in the synthetically generated sets and the anchor box sizes carefully.

As mentioned at the end of the subsection on anchor boxes, the training takes a considerable amount of time. With the set up available configuration it took around 14 hours to train with 2000 ½ size (960x540 pixels) images. So, both the training set and the training parameters require careful consideration as repeating the training due to bad parameter set up is costly in time.

4.6 Test set

The regular test set is created using a video camera installed inside a vehicle. The scenes were recorded in May 2019 on the motorways in İzmir between Bornova and Çeşme. No special consideration is taken for the setup; that is, no special cleaning is done on the windshield, no caution is taken against reflections from the windshield. However, the recordings were made between noon and 17:00 to make sure the lighting conditions are not adverse or extreme.

The images are used as they are, read straight from the video file frame by frame during the testing and used in their original 1920x1080 size.

Although initially a separate labeling code was used written through modifications on an available code from another research, in the final stage of the work, Matlab's "Video Labeler" app from its Image Processing and Computer Vision Toolbox was used for convenience. This toolbox had a KLT Tracker functionality, so it was possible to spend less time on marking and have the app mark a great number of frames on its own using the tracker as long as the initial marked sample was not too much bent out of shape and the final signs in the far distance were not very small for the tracker to falsely track. The initial manual markings were done when the signs were fairly close but not so close that they are bent out of shape due to lens distortion. The tracking feature was used in reverse so that the KLT Tracker would track the large sign backwards as it gets smaller and smaller.

The model did not seem to have a serious problem with false positives so initially only frames that contained the target signs were used.

It was observed that the targeted overhead signs appeared when there are exits in the motorway so the number of different instances of such signs was strictly limited to the number of exits in the motorway route used. However, since the frames were directly taken from a video recording, each instance of a sign would produce around a hundred frames of testing data in different sizes and locations according the time it can be seen in the view, considering the recordings are taken at a rate of 30 frames per second.

Other motorway signs on the side of the road were marked as “other”, a third, non-trained category, only to be ignored in the testing so that a positive or negative detection on this “other” class would not affect the experiment results positively or negatively.

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 Software and hardware

Generation of the synthetic data, modeling of the Convolutional Neural Network, training, annotation of real-world video frames, testing, and experimentation were all performed using MATLAB (R2019a) on Windows 10.

Although the coding part was done on two laptop computers, the code was run on a computer with Intel Core i7-8750H CPU and NVIDIA GeForce GTX 1060 GPU.

The real-world videos were recorded by a dashboard camera in 1920x1080 resolution and 300 fps.

5.2 Settings

The videos were recorded in the motorways of İzmir between Bornova and Çeşme. The videos used for testing were recorded in May 2019 mostly between the hours of 12:00 and 17:00. The frames were fed and processed in their original resolution.

The output of the detector was passed through a very simple filter to eliminate the obvious false results using size and position. Furthermore, the testing was done for the ground truth elements that are neither very far, nor very near. This was due to the fact that for practical applications, very far signs that are nor really legible are not valuable for detection, as well as the very near signs that would have already been detected. However, it must be stated that the elimination of the very near signs may have dropped the false negative rates for two reasons: 1) the signs go through a more pronounced lens distortion when they are near, and thus, towards the outer parts of the camera view, and 2) it gets harder to detect and classify the target classes that are partially visible and partially out of the range of view.

Initial testing was done only on frames that contained the target classes. Although this may seem to underrepresent the false positives rate, it is worth mentioning that in the 696 frames that contained the target classes, there were only 2 false positive detections

that were not related to signs so inclusion of frames with no signs is not expected to change the false positive rates much. Through manual experimentation, it was observed that false positives were only a minor problem; therefore, a lower threshold probability of 0.3 was used for the detector instead of the generally accepted 0.5. For ruling a match (true positive) or a mismatch (misclassification) an Intersection over Union (IoU) value of 0.5 is used.

5.3 Initial experiments

Initial experiments are done with 4 3-minute videos. 696 frames containing 16 unique instances of the target classes from varying distances were used for testing. No detailed filtering or tracking were applied although it was obvious that they would be easy to implement and would improve the performance.

Table 1. Performance of initial experiments

	Class1	**Class2**
Ground truth data	345	280
True Positives	307	151
Misclassifications	0	57
False Negatives	38	72
False Positives	20	0

The initial results indicate that the detector for class 1 is more eager. It detects 89% of its targets, misses only 11%. The false positives it generates are usually related to other sign-related picks and only 2 of those are unexplained detections from the background.

Class 2, when compared, is more recessive. It detects 54% of its targets, mainly because a 20% is misclassified by a greater probability by the class 1 detector. The remaining 26% is not detected by any detector. This is probably due to the fact that nearly half of the class 2 signs are on the exit lane on the side of the road so that they are greatly

affected by lens distortion and either not detected or although detected, misclassified due to the higher probability from the other detector. Class 2 detector does not interfere with the other class by triggering misclassifications.

We can turn this table into a detection table, rather than a classification table, by simply treating both classes as one and treating misclassifications as true positive detections. Then we would have the following table showing an 82% detection rate.

Table 2. Detection performance of initain experiments

	Object
Ground truth data	625
True Positives	515
False Negatives	110
False Positives	20

5.4 Further refinements

Although the results are better than what can be achieved with color-based methods, there is still much room for improvement.

First of all, the synthetic training data can be modified to be closer to the real world. There turns out to be some class 1 objects with a 1:1 aspect ratio in the testing data which we did not have in our synthetic set. The class 2 signs in the real-world are almost never located on the left half of the view while our synthetic data does generate quite a lot of class 2 objects on that side. Almost all of the class 2 signs that are misclassified as class 1 have the sky as their background and the class 1 detector selects a portion of that blue sky background from the right side of those signs; this maybe probably due to the fact that the background images used for synthetic data generation were all from a certain portion of the motorway where the right side of the road had a hill with greenery in the background. The detector may have associated the class 1 signs with blue background of sky, as is the right side of the background images used for generating synthetic data.

Other color-based checks can also be done to filter most of the false positives. For example, in some cases, the detector picks up some road construction notices with some text on a yellow-orange background. It might be picking up “text on a solid background” without a differentiation of the color of the background. This can be remedied either by using some similar negative examples in the background of the synthetic data generation, or, rather, by using a simple ratio-of-colors check on the detected regions to ensure that they are indeed in line with what is expected.

In Figure 19, you can see two detection annotations of the same frame. The orange road code sign was falsely detected as a sign with an earlier version of the training set because the target signs contained such portions and there were no such negative examples in the background. Using a refined version of the synthetic data generation code, such orange road code images were included in some of the background images for training. As a result, they are not falsely detected any more.

As can be seen in Figure 20, simple false positives can easily be filtered out by using a check on the location and size of the bounding boxes.

Figure 19. Orange signs before and after adding negative samples in training

Figure 20. Most false positives can be corrected with location and size filters

Other types of signs can easily be falsely detected. These can be corrected either by providing negative samples in the background of the training data or by simple post-processing filters. The sign below can be filtered out by a color content ratio filter.

Figure 21. Other signs require extra measures

5.5 Further testing

Further testing was done with a larger set test data which consisted of two separate frame sets: one with positive samples and one with no positive samples. The positive sample set had 2075 image frames that contained multiple samples of 54 unique instances of overhead signs (20 continuing and 34 exit signs) from various distances and angles and in various settings (inside and outside the city limits, with and without traffic, various background scenes, etc.). These frames contained 3490 positive samples of the signs, of which a small portion (137) were later excluded in some tests to have meaningful sized samples. The continual / exit signs ratio was around 7 / 10.

It must be noted that some extra filters were applied for the final testing stage of the detector. The outputs of the detector were filtered by location, size, aspect ratio, and color content to weed out false positives.

5.5.1 . The effect of target object area

The tables below show that the area (in pixels) of the target objects affects their detectability to a large degree as expected.

Table 3. TP and FN rates for continuing lane signs by sign area in pixels

Cont signs	**≥1.000**	**≥2.000**	**≥4.000**	**≥8.000**	**≥16.000**	**≥32.000**	**≥64.000**
TP	0	81	364	350	254	166	24
FN	9	109	34	0	0	0	0
TP%	**0%**	**43%**	**91%**	**100%**	**100%**	**100%**	**100%**

Table 4. TP and FN rates for exit lane signs by sign area in pixels

Exit signs	**≥1.000**	**≥2.000**	**≥4.000**	**≥8.000**	**≥16.000**	**≥32.000**	**≥64.000**
TP	0	91	419	443	238	55	2
FN	153	409	137	13	2	0	0
TP%	**0%**	**18%**	**75%**	**97%**	**99%**	**100%**	**100%**

Naturally, it gets harder to detect the signs as they get smaller. As they get larger, a larger percentage is detected. With the continual signs, this detection rate reaches a perfect peak very quickly, but with exit signs, it takes a bit longer. This may be due to several reasons. One can be that as the exit signs being usually rectangular, and thus, taking the smallest dimensions. Another can be due to the fact that half the exit signs are located on the far right side of the view and get affected by radial lens distortion more. As the exit lanes are a separate category, it may be that the detector was not well trained for this category.

Since training is mentioned, this should be a good point to mention some of the manually selected parameters for creating the training set and the hyperparameters for training.

First, as the training set is created synthetically, the parameters used in this creation may affect the results of the detectors. If the parameters do not match the real-world values, we can be training our detectors to look for wrong clues.

Furthermore, there may be some hyperparameters in the training stage that may affect the behavior of the detectors. For the Faster R-CNN, there is a hyperparameter called “anchor box” sizes. This refers to the possible shapes and sizes of the objects and is usually determined through a statistical manner, such as k-means clustering, from the

real-world data in the training set. As our training set is synthetic, we do not have any real-world values to consider for this parameter, so it is simply manually estimated.

Finally, we may not have included very small samples in our synthetic training set so that may as well be another reason for the problem in detecting small objects.

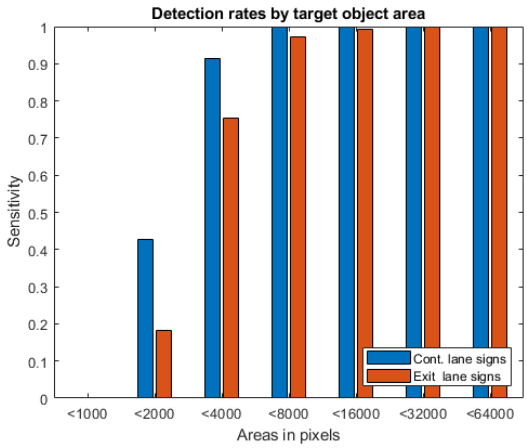

Figure 22. Sensitivity rates by target object area in pixels

5.5.2 . The effect of detector threshold

The threshold for the detector naturally affects its ability to detect correctly; however, its effects are not very dramatic. There does not seem to be a “sweet spot” for the threshold that would produce an “elbow point” in detection or misdetection rates. A detector threshold of 0.3 is used in testing and comparison purposes although even a lower rate seems to be viable.

For threshold analyses, a minimum target object height condition was applied as explained before and the total positive sample numbers became 3353, of which 1391 were continuing lane samples and 1962 were exit lane samples.

The “sensitivity” is also called “recall” or “true positive rate” (TPR) and is calculated as the ratio of positive samples detected.

Precision or positive predictive value (PPV) is calculated as the ratio of true positives to all positive predictions.

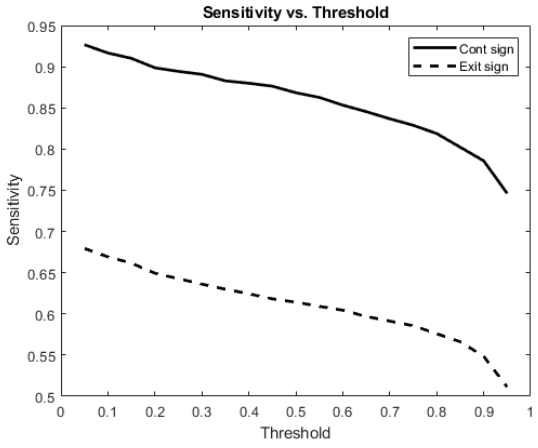

Figure 23. Sensitivity rates with varying detection thresholds

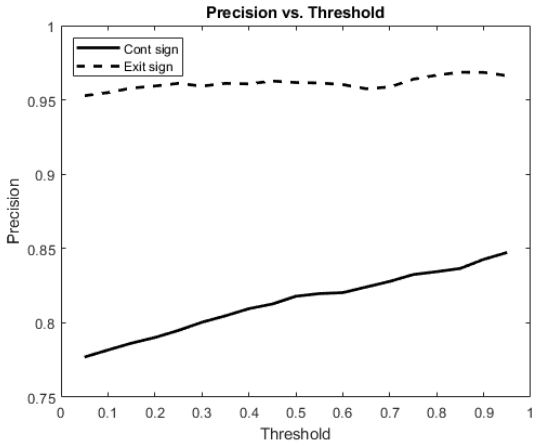

Figure 24. Precision rates with varying detection thresholds

As seen in the previous graph, precision rate follows a rather flat pattern without any elbow formation and has a very rate of 80% and above in general. What is more interesting is that the false positive and false negative rates in this analysis are in fact much better than it looks in these statistics.

Upon examining the detection boxes, it was clear that the post-detection filters were working wonderfully and there were no false positive detections in unrelated areas of the frames. All false positives were due to localization errors where the detection box does contain the target object but as the box is so large, the IoU (Intersection over Union) ratio is below the threshold, causing an increase in both false positives and false negatives.

The false positives are made up of correct detections with bad boundaries, the number of such false positives are still very low, making the precision statistics good.

The exit lane sign detector seems to be somewhat problematic with quite high false negative rates accompanied by very low false positive rates. It seems that its detector makes less detections, causing very low false positive statistics and very high false negative statistics.

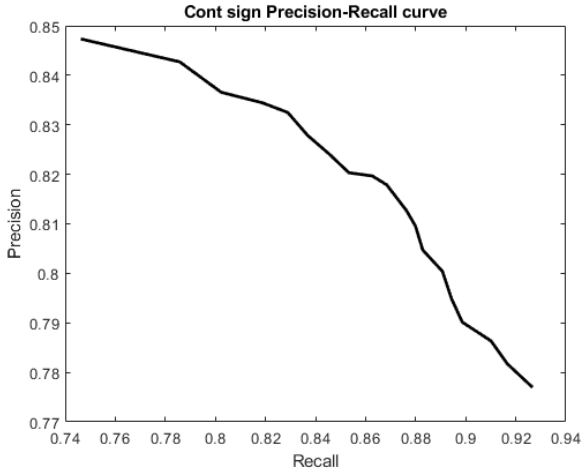

Figure 25. Precision-Recall curve for continuing lane sign detections

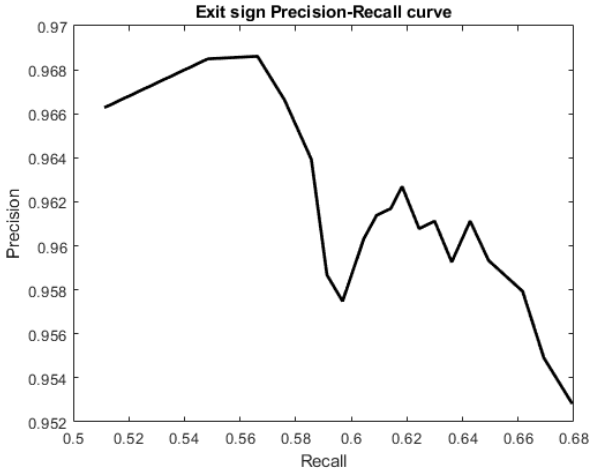

Figure 26. Precision-Recall curve for exit lane sign detections

Threshold	0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95
TP	1289	1275	1266	1250	1244	1239	1228	1224	1219	1208	1200	1187	1176	1164	1153	1139	1116	1093	1038
FP	370	356	344	332	321	309	298	288	281	269	264	260	251	242	232	226	218	204	187
FN	102	116	125	141	147	152	163	167	172	183	191	204	215	227	238	252	275	298	353
Sensitivity	93%	92%	91%	90%	89%	89%	88%	88%	88%	87%	86%	85%	85%	84%	83%	82%	80%	79%	75%
Precision	78%	78%	79%	79%	79%	80%	80%	81%	81%	82%	82%	82%	82%	83%	83%	83%	84%	84%	85%

Figure 27. TP, FP, FN statistics for continuing lane sign detection

Threshold	0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95
TP	1333	1313	1298	1274	1261	1248	1236	1225	1213	1205	1195	1186	1171	1160	1149	1130	1111	1076	1003
FP	66	62	57	54	51	53	50	50	47	48	48	49	52	50	43	39	36	35	35
FN	629	649	664	688	701	714	726	737	749	757	767	776	791	802	813	832	851	886	959
Sensitivity	68%	67%	66%	65%	64%	64%	63%	62%	62%	61%	61%	60%	60%	59%	59%	58%	57%	55%	51%
Precision	95%	95%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	97%	97%	97%	97%

Figure 28. TP, FP, FN statistics for exit lane sign detection

5.5.3 . The effect of post-detector filters

The main purpose of the filters is to eliminate or “filter out” the bad detections. This carries a chance of removing true positives as well as false positives and false negatives. The precision curves indicate that the filters work good in eliminating false positives effectively without disturbing the true positive numbers. In addition, when applied on their own, the location filter seems to be more effective than the color filter.

The sensitivity curves show that neither filter has much of an effect on class 1 as all curves coincide on top of each other. For class 2, however, location filter does not have much of an effect, its curve coinciding on top of the curve with no filters, while the color filter eliminates a small number of true positives, decreasing the sensitivity.

The numerical data is presented in Figure 32 and Figure 33 for further reference. The f0, f1, and f2 suffixes indicate no filters, filter 1 (location) only, and filter 2 (color), respectively. The number before those suffixes indicate the class of the object.

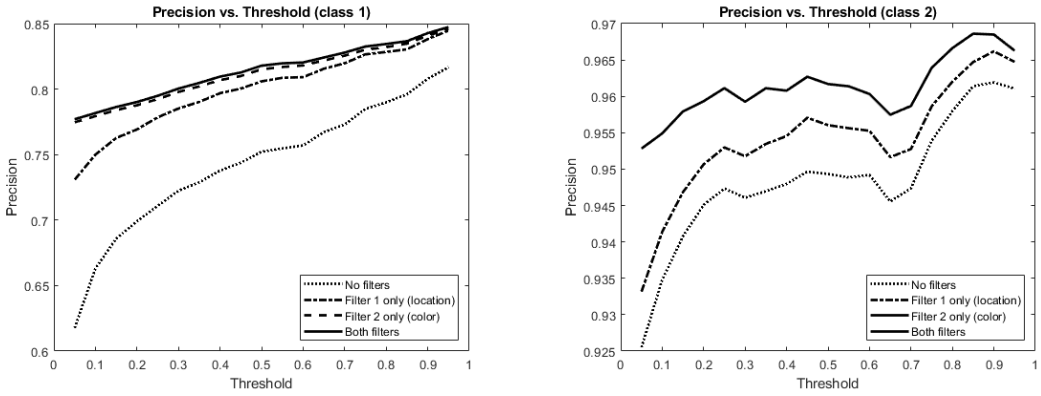

Figure 29. The effect of filters on precision statistics

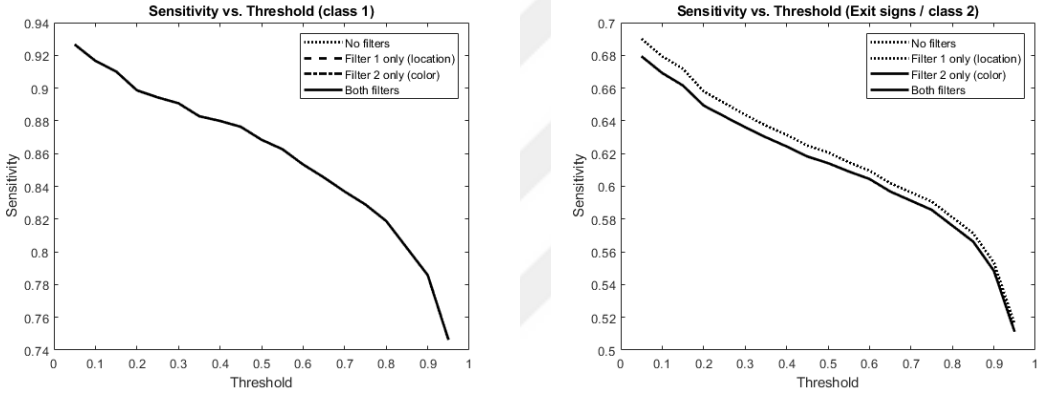

Figure 30. The effect of filters on sensitivity statistics

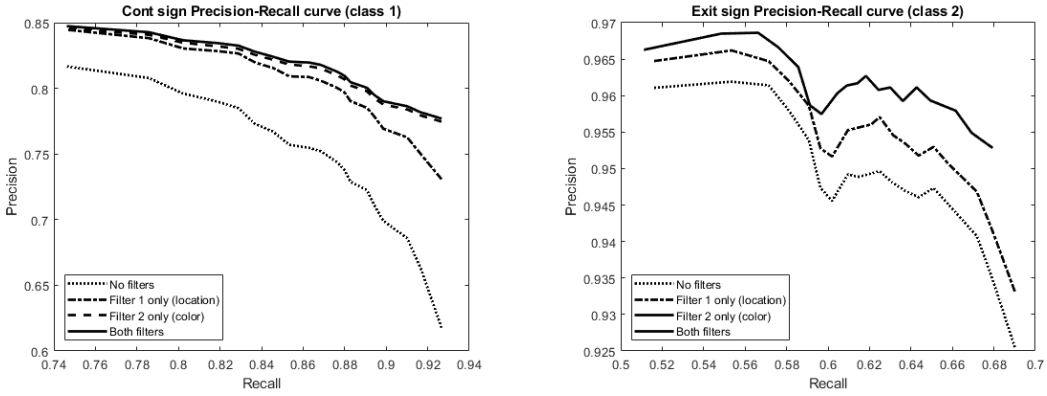

Figure 31. The effect of filters on precision-recall curves

Threshold	0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95
TP1	1289	1275	1266	1250	1244	1239	1228	1224	1219	1208	1200	1187	1176	1164	1153	1139	1116	1093	1038
FP1	370	356	344	332	321	309	298	288	281	269	264	260	251	242	232	226	218	204	187
FN1	102	116	125	141	147	152	163	167	172	183	191	204	215	227	238	252	275	298	353
Sensitivity	93%	92%	91%	90%	89%	89%	88%	88%	88%	87%	86%	85%	85%	84%	83%	82%	80%	79%	75%
Precision	78%	78%	79%	79%	79%	80%	80%	81%	81%	82%	82%	82%	82%	83%	83%	83%	84%	84%	85%
TP1f0	1289	1275	1266	1250	1244	1239	1228	1224	1219	1208	1200	1187	1176	1164	1153	1139	1116	1093	1038
FP1f0	798	648	580	538	506	476	457	435	420	398	390	381	357	342	316	303	286	260	233
FN1f0	102	116	125	141	147	152	163	167	172	183	191	204	215	227	238	252	275	298	353
Sensitivity	93%	92%	91%	90%	89%	89%	88%	88%	88%	87%	86%	85%	85%	84%	83%	82%	80%	79%	75%
Precision	62%	66%	69%	70%	71%	72%	73%	74%	74%	75%	75%	76%	77%	77%	78%	79%	80%	81%	82%
TP1f1	1289	1275	1266	1250	1244	1239	1228	1224	1219	1208	1200	1187	1176	1164	1153	1139	1116	1093	1038
FP1f1	475	425	394	375	354	339	326	312	304	291	284	280	266	256	242	236	228	211	191
FN1f1	102	116	125	141	147	152	163	167	172	183	191	204	215	227	238	252	275	298	353
Sensitivity	93%	92%	91%	90%	89%	89%	88%	88%	88%	87%	86%	85%	85%	84%	83%	82%	80%	79%	75%
Precision	73%	75%	76%	77%	78%	79%	79%	80%	80%	81%	81%	81%	82%	82%	83%	83%	83%	84%	84%
TP1f2	1289	1275	1266	1250	1244	1239	1228	1224	1219	1208	1200	1187	1176	1164	1153	1139	1116	1093	1038
FP1f2	375	361	349	337	326	314	303	293	286	274	269	264	255	246	236	230	221	207	189
FN1f2	102	116	125	141	147	152	163	167	172	183	191	204	215	227	238	252	275	298	353
Sensitivity	93%	92%	91%	90%	89%	89%	88%	88%	88%	87%	86%	85%	85%	84%	83%	82%	80%	79%	75%
Precision	77%	78%	78%	79%	79%	80%	80%	81%	81%	82%	82%	82%	82%	83%	83%	83%	83%	84%	85%

Figure 32. TP, FP, FN statistics by filters for continuing lane sign detection

Threshold	0,05	0,10	0,15	0,20	0,25	0,30	0,35	0,40	0,45	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95
TP2	1333	1313	1298	1274	1261	1248	1236	1225	1213	1205	1195	1186	1171	1160	1149	1130	1111	1076	1003
FP2	66	62	57	54	51	53	50	50	47	48	48	49	52	50	43	39	36	35	35
FN2	629	649	664	688	701	714	726	737	749	757	767	776	791	802	813	832	851	886	959
Sensitivity	68%	67%	66%	65%	64%	64%	63%	62%	62%	61%	61%	60%	60%	59%	59%	58%	57%	55%	51%
Precision	95%	95%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	97%	97%	97%	97%
TP2f0	1354	1333	1318	1291	1277	1263	1250	1239	1226	1218	1206	1196	1181	1170	1159	1140	1121	1086	1012
FP2f0	109	93	83	75	71	72	70	68	65	65	65	64	68	65	56	50	45	43	41
FN2f0	608	629	644	671	685	699	712	723	736	744	756	766	781	792	803	822	841	876	950
Sensitivity	69%	68%	67%	66%	65%	64%	64%	63%	62%	62%	61%	61%	60%	60%	59%	58%	57%	55%	52%
Precision	93%	93%	94%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	95%	96%	96%	96%	96%
TP2f1	1354	1333	1318	1291	1277	1263	1250	1239	1226	1218	1206	1196	1181	1170	1159	1140	1121	1086	1012
FP2f1	97	83	74	67	63	64	61	59	55	56	56	56	60	58	50	45	41	38	37
FN2f1	608	629	644	671	685	699	712	723	736	744	756	766	781	792	803	822	841	876	950
Sensitivity	69%	68%	67%	66%	65%	64%	64%	63%	62%	62%	61%	61%	60%	60%	59%	58%	57%	55%	52%
Precision	93%	94%	95%	95%	95%	95%	95%	95%	96%	96%	96%	96%	95%	95%	96%	96%	96%	97%	96%
TP2f2	1333	1313	1298	1274	1261	1248	1236	1225	1213	1205	1195	1186	1171	1160	1149	1130	1111	1076	1003
FP2f2	66	62	57	54	51	53	50	50	47	48	48	49	52	50	43	39	36	35	35
FN2f2	629	649	664	688	701	714	726	737	749	757	767	776	791	802	813	832	851	886	959
Sensitivity	68%	67%	66%	65%	64%	64%	63%	62%	62%	61%	61%	60%	60%	59%	59%	58%	57%	55%	51%
Precision	95%	95%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	96%	97%	97%	97%	97%

Figure 33. TP, FP, FN statistics by filters for exit lane sign detection

5.5.4 . A note on “other” categories

Although this work concentrated on motorway overhead directional signs, there are still motorway signs that are not overhead or directional, as well as overhead signs that are not directional or not specifically for motorways. Such signs are considered as “other” categories because an extension of this work might later include such categories.

In addition, color-based methods do not distinguish the signs by their content and the motorway signs that are not directional of overhead can also be detected without classification. So, in this work, other motorway signs (not overhead and/or not directional) are ignored and assigned to an “other” category to avoid confusion in the resulting statistics. In fact, in the final stage, only scene frames with motorway overhead directional signs were selected to avoid any confusion.

5.5.5 . The case with negative samples

The case with negative samples in object detection, localization, and multiple object detection is slightly different from general data classifications. As previously explained, the location and size of a detection bounding box is important as a misplaced and/or mis sized bounding box can turn a single true positive detection into a false positive and a false negative detection together.

Another issue is with the true negative statistics. A “true negative” case in the setting of this work is a traffic scene frame that does not contain any positive samples. However, in truth, localization of multiple objects in a single frame means that the detector is already going through multiple “true negative” regions of a single frame to detect and localize the “true positive” region in a frame with a positive sample.

Furthermore, when going through a dashboard video, a very large ratio of the frames would not contain the determined positive sample signs. Thus, a video would consist mainly of negative sample frames. This would result in negative samples dominating the analysis of the detectors which would not be very informative.

As a result, most of the analysis is done with frames containing positive samples. Since this is also leaves a part of the analysis out, a separate study is made using only negative samples, i.e. scenes containing no positive objects.

For this a separate set of frames with no positive samples are formed. 622 frames are selected from the motorway from different locations, containing different scenery e.g. trees, greenery, buildings, mountains, cars / no cars, trucks, side rails / no side rails, overpasses, etc.

Although the detector could produce some false positives in those negative sample set, all these false positives could easily be filtered with a post-detection filter using size, location, and color content information. However, this was done as part of a comparison with the color-based method so the detailed results will be presented in the following section.

5.5.6 . The effect of the selected IoU threshold

Intersection over Union (IoU) ratio has turned to be a standard in determining the correctness of localization of objects. However, as will be explained in the coming section, localization problems lead to misleading true positive, false positive and false negative statistics. The effect of this can be observed on the change in these statistics when using the same detection methods but with different IoU thresholds for marking localizations correct or otherwise.

As can be seen in the next figure, the detectors work fine and do not report many false positives, partly thanks to filters. This means that they do not misdetect non-existent objects. However, their localization can be problematic, causing the detections to be reported as false positives and false negatives as the IoU measure can fall below the selected criteria.

This case is confirmed in Figure 34. When the IoU ratio criterion is low (0.3 and below), the precision statistics approach to 100%. They also approach to zero when the criterion is very high, i.e. 90%, but this is a common issue.

It can be seen from Figure 34 that sensitivity statistics also tend to remain rather stable for IoU ratios of 40% and below.

The generally used IoU ratio of 0.5 is slightly above the two ratios of 0.3 and 0.4 mentioned above.

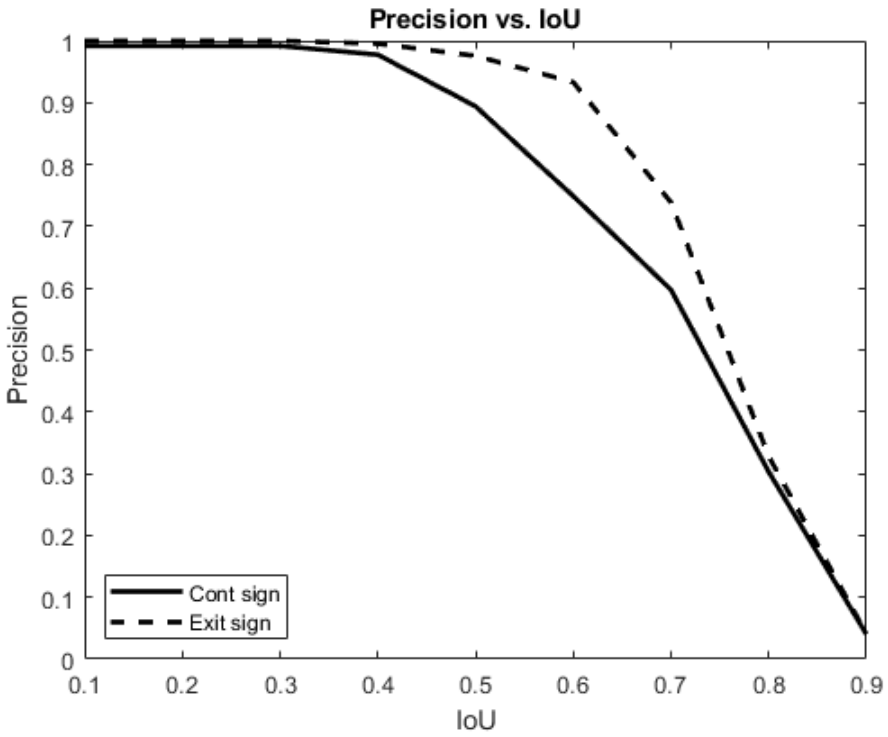

Figure 34. The effect of IoU criterion on precision statistics

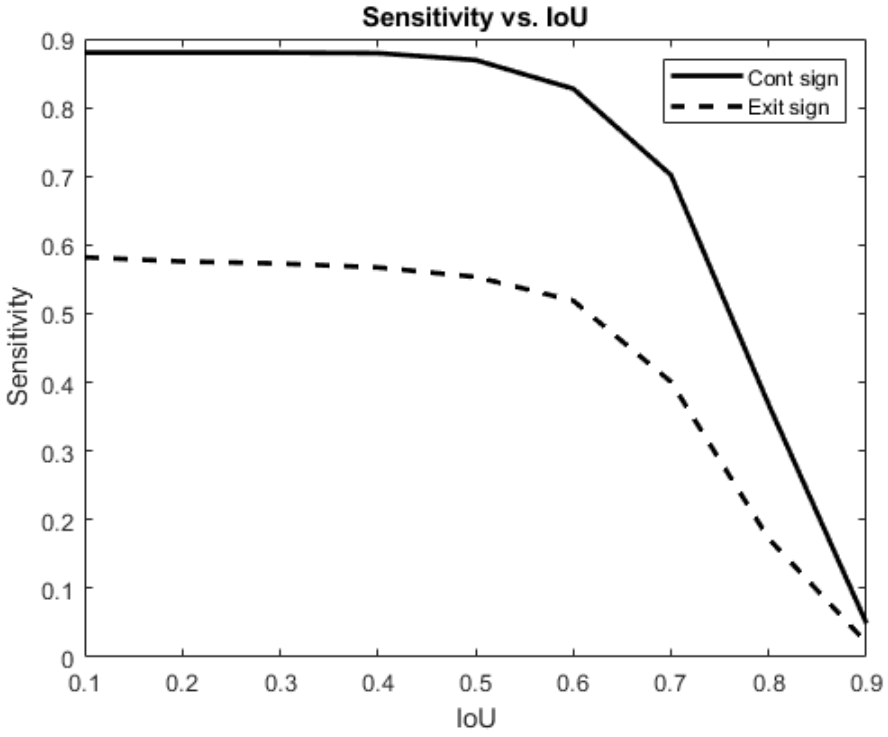

Figure 35. The effect of IoU criterion on sensitivity statistics

5.6 A comparison with hard-coded color-based methods

"Vision-based road sign detection" paper [4] was the starting point of the hard-coded color-based methods used for comparison. An overview of how these methods were applied is briefly described in Chapter 2 of this work. In this section, a comparison is made with such a method.

5.6.1 . A qualitative comparison

This use of CNNs start on the same level of input with color-based hard-coded methods, having no initial data set for training, but they produce better results in general. This is mainly because the color-based methods do not seem to be robust to minor changes in the environment. Although the color-based method uses the Hue values to be more robust, lighting conditions still affect them.

The unbeatable advantage of the Faster R-CNN architecture over color-based methods is that it provides detection, classification, and localization, all in one go. Color-based methods locate possible regions that might contain the object classes, but they cannot produce the level of classification CNNs can offer.

Another interesting point is the false positives rates. Color-based methods initially find many possible regions for the signs to be present. Many parts of a rural motorway scene can include regions that can pass the color-based filtering, such as a clear dark blue sky, greenery on the side of the road, a patch of blue sea in the distance, etc. That means many false positives in the initial sweep. While these possible regions can be further processed to eliminate the false positives, this process requires extra work and does not always produce satisfactory results. In addition, there is always the chance of an unrelated object having the same color mix as the overhead signs to make it past the filters as a false positive.

There is also the complication of different colored boxes within the motorway directional signs. Not only do blue and brown backgrounds cause a degree of separation, white backgrounds cause even more problems as white is not a color that can be selected on the 360-degree hue range. Sometimes such different colored boxes, the white text, and the white arrow signs might cause a color-based system to treat a single sign as two signs separated by those colored parts.

In comparison, CNN techniques tend to pick their proposed detection boxes by relying more on the texture content. This helps the detector to distinguish better among the texture contents of the regions rather than color similarities and possible color variations. However, CNNs can misdetect regions with different texture such as the back windows of cars, flat and dull colored vehicle bodies, rails on the side, etc. Luckily, these regions can easily be filtered by their location in the image and by their color content.

In addition, there is the situation of rating the likelihood of a match. CNN techniques provide naturally a likelihood rating of a match. Color-based methods concentrate on producing possible regions for an object, but they do not inherently have a rating system for likelihood of a match.

The problematic part with the Faster R-CNN methods is the hyperparameters to be selected at the time of training. Anchor boxes is one of those hyperparameters that is very hard to judge. Normally, when real-world data is used for training, anchor box hyperparameters can be evaluated from what is available in the training data, usually through a k-means clustering algorithm. But without real-world data, as in our case, one has to make estimations for the anchor box values.

This is an advantage for color-based methods. When the color of the signs are within the expected range and the foreground is distinctively different, color-based algorithms can clearly determine the boundaries of the regions of interest (ROI)s. The Faster R-CNN method, however, may have to depend on the anchor box hyperparameters defined and may not determine the boundaries as closely as expected.

But overall, color-based systems can be useful in producing possible regions or filters for detection but for the more important part of classification and assigning probabilities, they need to switch to other methods and algorithms, of which CNNs are likely to be preferred. So, it is also intuitive to start with CNNs in the first place and complete the whole process in one sweep of a single method.

The best method might be to use the best of both worlds by combining the two methods. The color-based methods can support the CNN methods for performance improvements. While earlier works in literature would start with a color-based filtering and then move on to other classification methods, including CNNs, the reverse might be productive as well. The CNNs work very good in detecting, classifying and localization but the errors they make seem to be simple ones that can be detected by color-based methods. So, while CNNs perform better on their own than methods that start with color-

based operations, CNNs can still benefit from a simple color-based filtering action after the CNN-based detection is performed.

5.6.2 Comparison through external examples

In the figure on the next page, you can find some examples scenes from the Internet processed by a color-based code (on the left) and the Faster R-CNN trained by synthetic data (on the right). The images are selected from the Internet to provide a variation in size, contrast, and color in order to how the two methods generalize to samples from different sources.

In the first two examples, it can be seen that the color-based method can have problems in separating two signs side by side. This is a common case. The color-based method seeks regions containing the desired color-content and selects the connected components in such regions. If two signs are very close or have a background containing the desired color, the regions of the two signs get merged.

In the third example, it can be seen that when a sign has too much of different colored backgrounds, it may be treated as a different object due to filtering.

The fourth example is also interesting because color-based method misses obvious signs; the initial color-scanning might have included the sky on the background and the large selection might have been rejected by color-rate, size, or location filters.

The Faster R-CNN does a good job in detecting the two separate signs, as well as another sign further away. However, it does not classify them successfully and the bounding box areas are not very accurate as we would expect. Such accuracy problems with the bounding boxes cause other problems with the performance statistics as we will see in the following sections.

The worst part of the results of the color-based methods is that they tend to vary drastically with lighting conditions and color distribution in the images. Since the coding is done manually (assuming that we do not have any training set to start with), any variation in the test set (direction of light, cloudiness, shadows, etc.) causes problems.

Figure 36. Color-based and R-CNN applied to images from Internet

5.6.3 Comparison through negative samples

As explained in the previous section, a separate data set of negative samples was formed by selecting 622 frames on different locations of the motorway containing different scenery and other objects.

Both the color-based detector and various variations of the Faster R-CNN detectors were very successful with the negative samples, provided that the outputs are passed through various filters.

Table 5. Comparison of false positives on a negative sample set

Method	**Color-Based**	**Faster R-CNN**
Initial FPs	NA	73
After location filter	Included	16
After color filter	Included	0
Final FPs	**6**	**0**

The color-based method seems to perform good. However, it must be stated that this is not the case in general. There are usually objects or background regions that can easily interfere with the color-based selection scheme. For example, if we were to detect regular highway overhead signs, their blue background would usually mix up with the blue sky in the background depending on the weather and lighting conditions. Similarly, the greenery on the side of the road gets selected very often by the color-based algorithms. Therefore, such algorithms should contain location, size, aspect ratio filters inside them; otherwise, the number of false positives would really be very high.

The two false positive frames in the following figures show that there are situations that the color-based method cannot escape from.

A high vehicle, such as a truck, with a color content similar to the target objects can easily be detected by the color-based methods. In this case, the location of the region in the image, size and aspect ratio of the selection, as well as the color content were all appropriate to make this selection get past the filters.

Figure 37. A false positive example from the color-based method

Figure 38. A false positive example from the color-based method

Furthermore, such problems with the colored sides of high vehicles would be augmented since the vehicles causing the problem are moving in the same direction with the camera recording. While only two frames were selected from this scene to make up the negative sample data set, the original video recording would have numerous frames

until the vehicle is passed. Since the vehicle is moving in the same direction, there would be even more frames than a stationery item in the scene would have.

Faster R-CNN method trained by synthetic samples does not perform very good on its own. There are many initial false positive outputs and a manual check reveals that they are selections that really do not resemble the target objects. This turns out to be a strength because then it is very easy to filter them based on their location, size, aspect ratio, and color content.

The table in the previous page might seem to suggest that false positives are better filtered through location/size filters but that is only because those filters are used as the first filtering stage as it is a much faster process compared to color content filtering.

5.6.4 Comparison through positive samples

Another comparison is made using the positive samples data set used in the previous section. Here, there are 2075 image frames, containing 3051 positive samples obtained from 54 unique instances of signs.

As the color-based method does not distinguish between sign types, the two-class output of the CNN detector was also combined into a single class. The comparison table is presented below with regular IoU of 0.5 and a test IoU of 0.3.

Table 6. Comparison of performance on positive sample set

Method	C-B	C-B IoU3	CNN	CNN IoU3
TP	1987	2245	2451	2550
FP	329	43	290	9
FN	1064	806	600	501
Sensitivity	65%	74%	80%	84%
Precision	86%	98%	89%	100%

The numbers in this table show that the CNN detector performs better than the color-based method. But as explained before, these numbers do not reflect the exact detection performance due to the way bounding boxes are formed. Although both

methods seem to exhibit many false positives, the actual outputs do not contain any output that is not associated with any signs. This is more evident from the drop in false positivies when a lower IoU of 0.3 is used.

Color-based methods have the problem of the content of signs dispersing the important colors and separating the detected regions. The images in the figure all show that signs are detected by a color-based method but as the signs are far, the images are small and the regions get separated by the content on the signs. This makes the statistics look much worse than the performance of the algorithm. For example, for the sign on the right, the white background and the arrow sign divides the color regions. The color-based method outputs two regions, which are actually part of a positive sample, but this output only results in two false positives (small portions of a sign) and one false negative (the majority of the sign region not detected).

Figure 39. Problem of separated regions with color-based methods

Figure 40. Problem of missized bounding boxes with CNN methods

R-CNN methods experience a similar problem due to the anchor box hyperparameters and errors in sizing and localization. The two detection boxes above correctly detect the signs, but the bounding box sizes were not determined correctly. In both cases, although the detections were correct (true positive), the IoU (Intersection over Union) criteria results in one false negative (IoU below threshold) and one false positive.

Other similar problems occur for both methods when the bounding boxes are not set up correctly and contain the combination or parts of two signs as can be seen in the figure below. Such problems not only increase the false positive figures, but also the false negative ones.

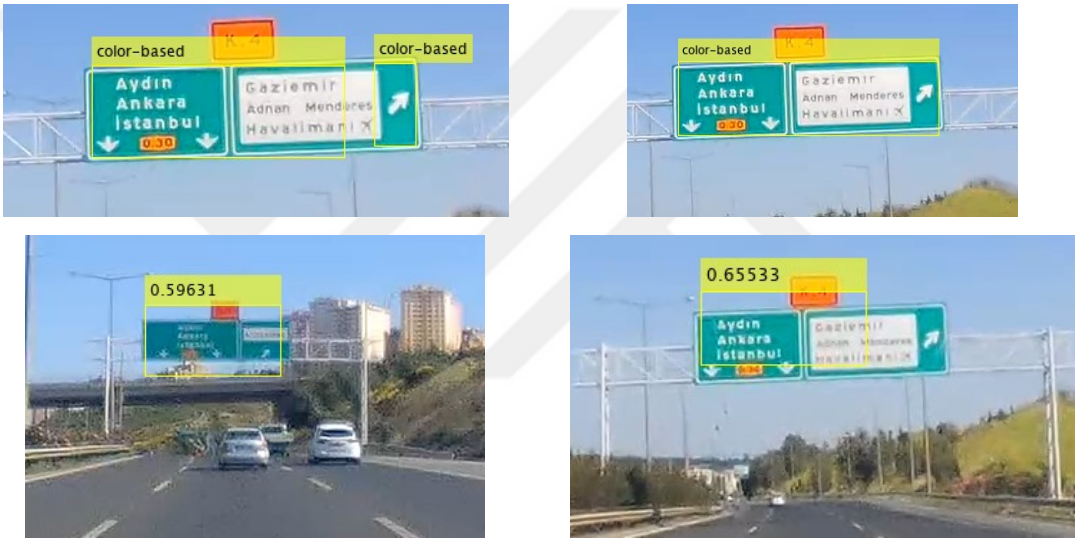

Figure 41. Boundıng box problems with both methods

5.6.5 Speed comparison

A speed comparison of the Faster R-CNN method and the color-based method is done on the positive and negative sets. These two sets are used separately to see if the existence of target objects may differ the processing time.

In addition to the two methods, optional pre-processing and post-detection filters are also timed to see if their positive effects are worth their toll on the time.

The tests are done in the MATLAB interpreter environment without any special configurations such as compiled codes or functions; therefore, the times do not indicate the real speed that could be reached with a compiled language. No special measure is taken to utilize the GPU; however, MATLAB does seem to utilize it on its own. Thus, the results here are only for comparison and should not be taken as the expected speed with specialized software and hardware setups. Timings are measured on a single call of the methods for each frame (rather than repeating several calls and averaging) since this would be the regular way they would be run.

The results of the timing test on the positive set is presented in the following table. As can be seen, The Faster R-CNN method takes 63% more time on its own. The optional pre-processing filter takes a noticeable time so its usefulness can be reviewed. The optional location filter takes an insignificant time as its work is done only on the location parameters of the bounding boxes which is why this filter is applied before the color-content filter. The optional color filter takes a very small amount of time.

Table 7. Speed measurements on the positive set

Method	**Cumulative time**	**Counts of pass**	**Average time**
Color-based	1029,71 sec.	2075	**0,49624 sec.**
Faster R-CNN	1683,12 sec.	2075	**0,81114 sec.**
Opt. pre-processing	124,01 sec.	2075	0,05976 sec.
Opt. location filter	0,17 sec.	1857	0,00009 sec.
Opt. color filter	22,88 sec.	1847	0,01239 sec.

The results of the timing test on the negative set is presented in the following table. The results seem to be similar for both sets. Post-filters seem to differ in terms of ratio, but since their times are already low, it may be due to measurement errors and they do not contribute a significant change on the overall figures anyway. In addition, there seems to be a drop in the Faster R-CNN timing, but without going into the statistical significance of this drop, we can conclude that it is still significantly slower than the color-based method.

Table 8. Speed measurements on the positive set

Method	Cumulative time	Counts of pass	Average time
Color-based	312,90 sec.	622	**0,50306 sec.**
Faster R-CNN	474,79 sec.	622	**0,76333 sec.**
Opt. pre-processing	37,42 sec.	622	0,06016 sec.
Opt. location filter	0,01 sec.	70	0,00012 sec.
Opt. color filter	0,12 sec.	15	0,00821 sec.

As mentioned above, these are timings from MATLAB's interpreter and using a compiled version would naturally run faster. But for real-time purposes, we would still need a faster CNN architecture such as YOLO or SSD to work with the common 30 frames per second standard.

5.7 A hybrid approach

When conditions are favorable, color-based approach can determine a more accurate bounding box. CNN approach, on the other hand, can determine bounding boxes that are just acceptable for an acceptable level of Intersection over Union. However, as explained on the section on comparison through positive examples, problems with the bounding boxes may adversely affect true positive, false positive, and false negative statistics. A hybrid approach can improve this situation.

Color-based methods already need to use a hybrid approach for classification. Those methods have to make a trade-off between filtering negative regions and not missing positive regions, but after they determine the regions that are likely to contain the target objects, a second run with another method is still needed to confirm or classify the objects.

CNN methods, in theory, do all this by themselves: find the object, determine its region, and classify. Some of the problems caused by bad localization can be partially fixed by adding a color-based localization improvement after the output of the CNN. This part can also be a more relaxed selector as at that stage, we would already have an object and its bounding boxes and all we need is small improvements in the boundaries. At that stage, neither mistakenly detecting a false positive, nor the possibility of eliminating a

true positive are problems. However, there is still the possibility of messing a correct bounding box due to parameters used in the color-based approach.

The best place to add the color-based localization would be after the location filter and before the color-content filter. The location filter runs very fast and eliminates most of the unwanted output from the CNN. It would eliminate most of the unnecessary detections before applying the color-based localization corrections. On the other hand, color-content filter uses the whole bounding box to test its color content. Having a larger than necessary bounding box would include background parts of the box as noise and adversely affect the results of this filter. As a result, the workflow of the hybrid approach would be as shown in Figure 42.

The examples in Figure 43 show some instances where the localizations of the CNNs (on the left) would normally increase both the false positive and false negative statistics by one, due to bad localization and an IoU below a 0.5 threshold. A color-based bounding box correction step, as a hybrid approach, would improve the localization (on the right) and change those statistics into a true positive.

If we go over the procedure for the first example, the detection on the left correctly identifies a sign but draws a bounding box that is too wide. Here, the IoU ratio goes below 0.5 and therefore, this is not marked as a true positive. Even worse, since there is a detection that is not classified as a true positive, it is considered as a false positive. Furthermore, since the detection was not classified as a true positive, the sign is considered undetected and, as a result, a false negative.

However, since color-based approach does not work perfectly in all conditions, this final improvement stage can also worsen the results in certain situations or with some bad parameter matchings as in the example in Figure 44.

The results of running the hybrid approach over the positive sample set is presented in the Table 9. The two columns indicate the results with the IoU ratios of 0.5 and 0.3 respectively.

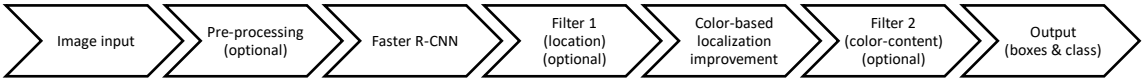

Figure 42. Workflow for the hybrid approach

Figure 43. Localizations (left) can be improved (right) by a hybrid approach

Figure 44. A hybrid approach can also worsen the situation rarely

Table 9. Results of the hybrid approach performance on positive sample set

Method	CNN	CNN IoU3	CNN-h	CNN-h IoU3
TP	2451	2550	2510	2550
FP	290	9	95	2
FN	600	501	541	501
Sensitivity	80%	84%	82%	84%
Precision	89%	100%	96%	100%

According to the test results, hybrid approach seems to improve the overall results. As can be seen in Figure 45, filters are effective in decreasing the false positives, but a hybrid approach takes this improvement even further. Filtering removes unrelated matchings and decreases false positives; hybrid approach improves detection boundaries to correct good detections (true positives) being registered as false positives due to localization errors.

With false negatives, filters may work in the wrong direction. Filters can actually increase false negatives if they remove good matchings. Hybrid approach, however, works in the same with false negatives, by improving detection boundaries so that true matchings do not get registered as false negatives due to localization errors. These improvements show their effects on the sensitivity and precision statistics. And finally, the improvements can naturally be observed in the precision-recall curves as well.

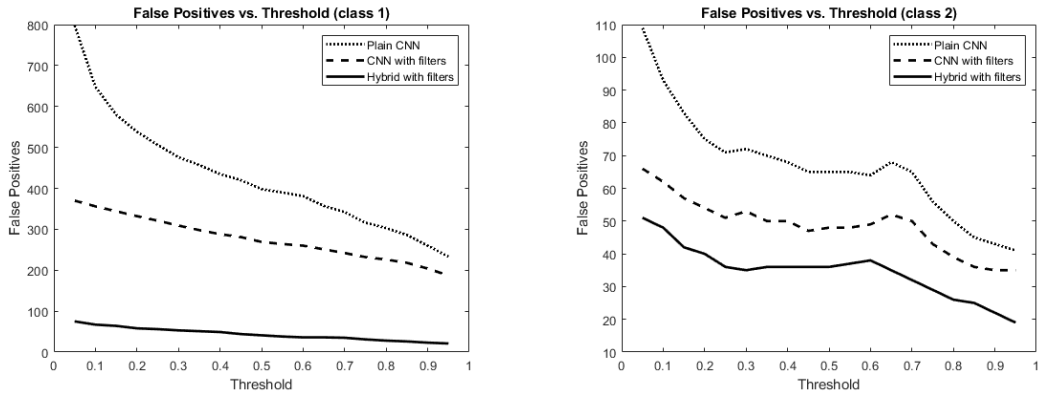

Figure 45. Effect of hybrid approach on false positives

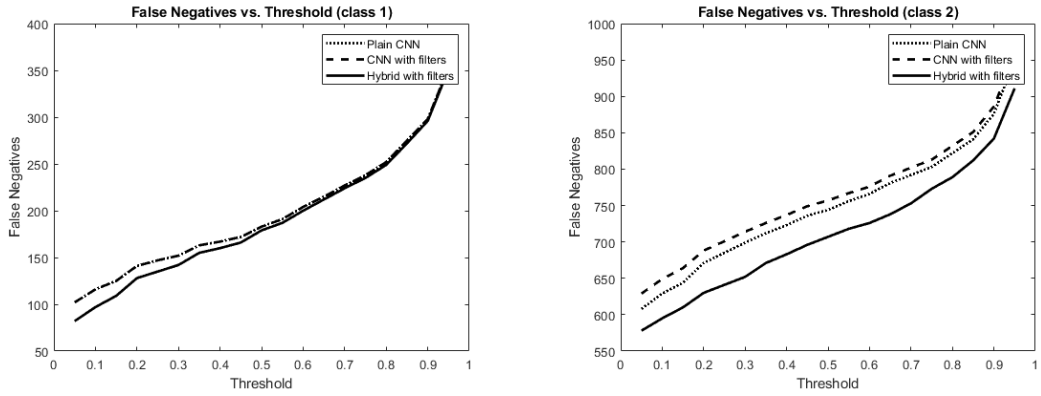

Figure 46. Effect of hybrid approach on false negatives

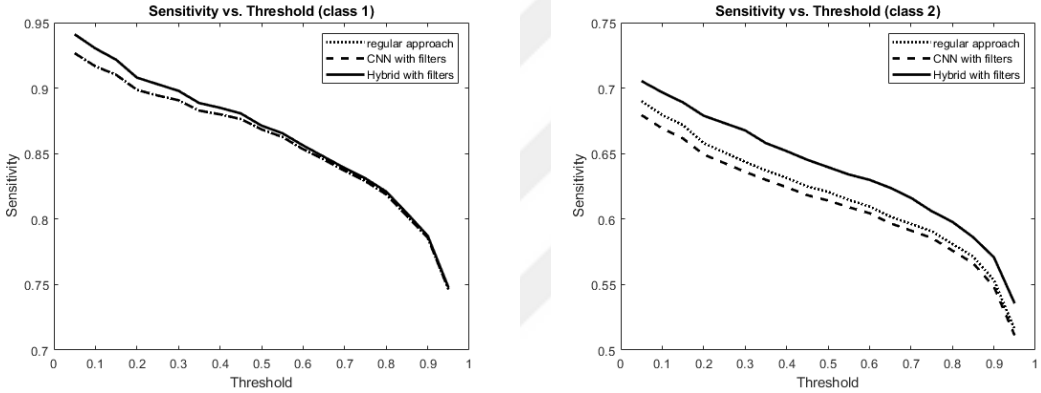

Figure 47. Effect of hybrid approach on sensitivity

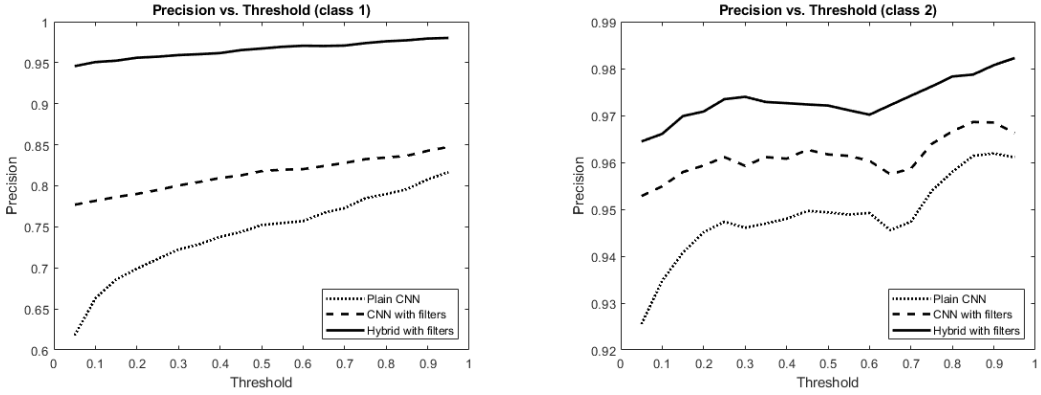

Figure 48. Effect of hybrid approach on precision

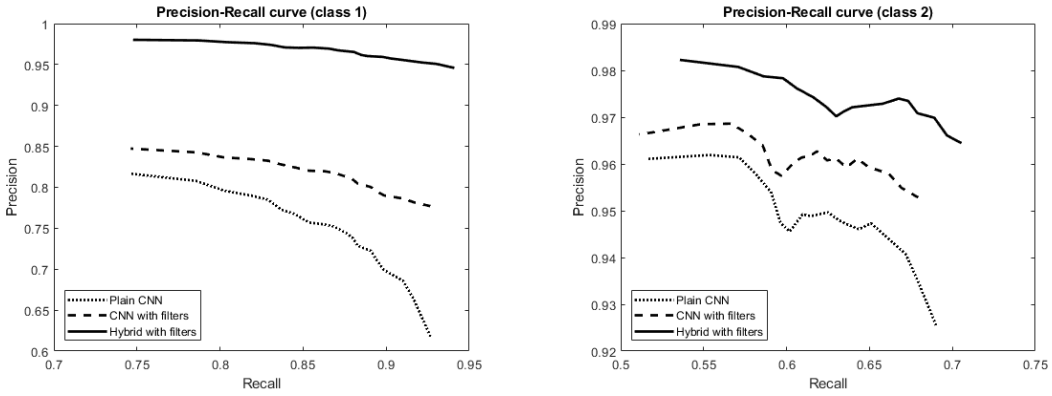

Figure 49. Effect of hybrid approach on precision-recall curves

5.8 Adverse conditions

The main idea of this work revolved around the viability of using a synthetic training set to detect and localize real objects in real-world scenes. The results were naturally tested on regular and common real-world scenes and situations. However, there can also be some adverse conditions, such as sun shining from ahead, cloudy and rainy weather, and nighttime scenes. Color-based methods mostly fail in adverse conditions as the color ranges are not robust to such changes even when the hue data is used. The CNN methods are also not very robust in such situations, but they can still produce some acceptable outputs. However, the post-detection filters were coded with regular conditions in mind and usually filter the acceptable outputs in adverse conditions. Some qualitative examples are presented below with post-detection filters disabled.

Dark weather is one of the worst adverse conditions because video camera recordings from a moving vehicle get very blurry. When there is less blur and the signs are illuminated, there is higher chance of detection.

However, as can be seen in Figure 51, when moving fast, night shots are usually blurry and not very usable.

The study was done using synthesized training data on a regular day so failures with adverse conditions are expected. Further work can be done to see if simulating adverse conditions in the synthesized training data can improve the detection rates in such conditions.

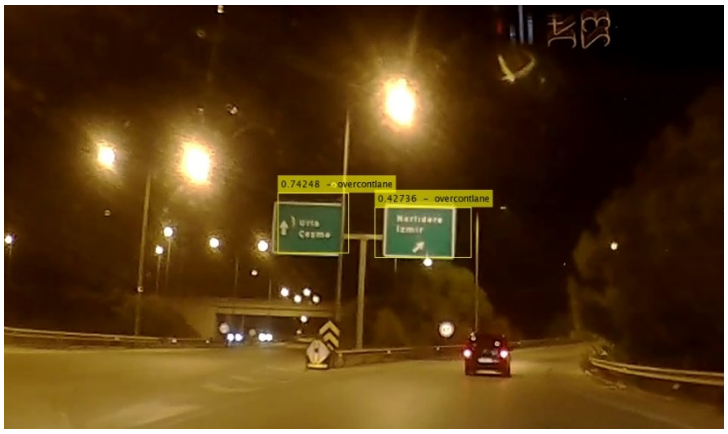

Figure 50. A fine nighttime example

Figure 51. A blurry night frame example

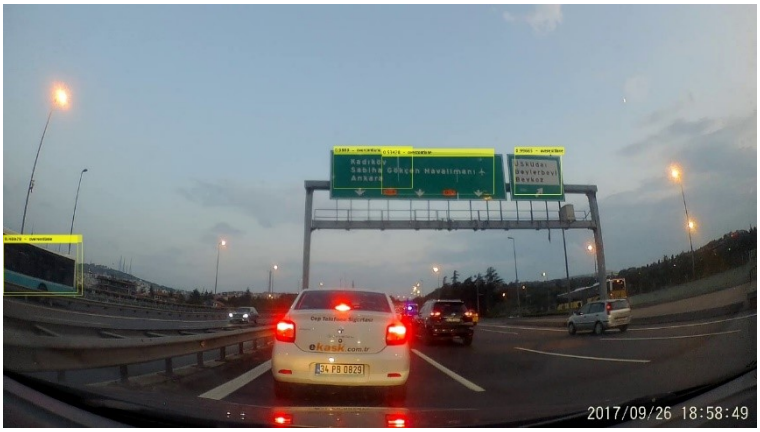

Figure 52. A mildly dark example where the traffic has stopped

Figure 53. A cloudy weather example

Figure 54. An example where the sun shines at the camera

Figure 55. An acceptable example of a rainy scene

Figure 56. A rainy weather example not working well

CHAPTER 6

POSSIBLE FUTURE EXTENSIONS

6.1 Faster CNN architectures for real-time detection

Current work is prototyped using the Faster R-CNN [10] architecture to show that training a good performing DL network is possible by starting with no real data at all. However, for the specific sample application of detecting motorway directional signs, an architecture that can perform in real-time would be more useful. Current popular examples of such systems are YOLO [47] and SSD [49] systems. Converting the current architecture to such systems would be more impressive for the current application field.

6.2 Adding more sign classes

Current work is prototyped to show that training a good performing DL network is possible by starting with no real data at all. This has been shown by using a very small subset of road signs, specifically two categories of informative motorway directional signs placed over the road, indicating where the lanes lead to. These signs were divided into two groups for continuing lanes and the exit lane at the right. For the specific needs of different applications, more subsets of specific signs can be included for the desired sign types.

6.3 Incorporating more types of data augmentation transforms

Current work is prototyped to show that training a good performing DL network is possible by starting with no real data at all. This has been done with a small set of data augmentation transforms: scaling, gaussian noise, darkening, rotation, and shear. These transforms were enough for the demonstration task at hand, detecting large signs over the road. However, there can be at least two more data augmentation methods for improving the possible performance for the road-side signs: obfuscation and lens distortion. In

addition, other transformations, such as reflections, night-time view, weather conditions, etc. can be considered to create a perfect system.

6.3.1 Obfuscation

The signs over the road usually have a clear direct sight of vision that can only be blocked by higher-than-usual vehicles such as trucks, and usually only when the signs are far. This blocking is usually close to full obfuscation and it is not worth including this type of obfuscation in data augmentation transformations. However, road-side signs in Turkey can usually be partially blocked by trees and bushes and still can be identified by human drivers. It would make sense to include obfuscation as part of the data augmentation transforms if the signs on the side of the road are to be added in the detection and classification list.

6.3.2 Lens distortion

Similarly, lens distortion takes its full effect on view pixels further from the center of the camera lens. For the common “landscape” view where image width is more than the height, the signs on the far left and far right side of the view would suffer from the non-linear camera lens distortion the most. Simulating such lens distortions can be considered to be added in the data augmentation transforms, if higher performance numbers are required. However, for practical purposes, this does not seem to be very useful for two reasons. First, it is more important to detect and process a road sign earlier in the process when it is still far away and, likely, not yet distorted as it is closer to the center of the camera lens. Second, when it is close enough to be distorted radically, it is already detected and processed anyway. In addition, at such viewpoints from a moving vehicle, the signs are visible for only a small amount of time in their distorted form due to the speed of the vehicle. In our example, since the objects move to the top of the viewpoint as they get close and since the topside of a “landscape” image is still close enough to the center of view for the camera lens, the distortion is not as much. The signs are already detected from a distance and as they move closer to the top of the view, they quickly move out of the field of view and disappear anyway.

6.3.3 Reflections of light

Reflections from sunshine or vehicle headlights can rarely affect partially the vision of road signs. These can be simulated in a similar way to obfuscation transformations.

6.3.4 Night-time conditions

The training and testing were done for close-to-ideal lighting conditions. For detecting the signs in night-time conditions, both the background images and the data augmentation sets will have to be modified for night-time conditions, if this is a necessity for the required application.

6.3.5 Adverse weather conditions

Similarly, other augmentation transforms can be considered for data augmentation for adverse weather conditions. While it may be easier to simulate a “snowy weather” for the augmentation transforms, other conditions such as “heavy rain” may, of course, drop the performance of the detectors even if they could be trained with real data from such conditions. On the other hand, it is worth noting that it is much easier to simulate such conditions through data augmentation transformations than waiting for a day with heavy snow or rainfall.

6.4 Fine-tuning with a limited set of real data

Although this work concentrated on showing that CNN models can be trained with synthetic data, other work in the literature have also tried training a model with synthetic data and fine-tuning it with real data later. It might be interesting to test how adding a very limited set of real training data might affect the results of the model and whether it would be worth including real data.

CHAPTER 7

CONCLUSIONS

The frames were initially manually tested to observe any possible problems. While these manual tests are not turned into numerical results, they provided valuable insight both on how the model works and on locating possible problematic issues to be fixed before a full experiment would be run.

7.1 The possibility of using entirely synthetic data

As can be seen from the literature and observed from the results, a CNN can be trained using entirely synthetic data. In this work, “entirely” may not be fully correct as real motorway images are used as the background. But, considering that no real objects to be detected and classified were present in those images, we can say that the CNN was trained for classification using entirely synthetic examples of the classes to be detected.

This method was especially practical for the application presented, detecting highway signs, because although the signs had a degree of variance, they also had a degree of structure; in addition, they were also easy to generate through code. More detailed objects, such as 3D household objects, or more detailed environments, such as traffic scenes where an OpenGL rendered 3D simulation was used for the entire traffic scenes, would require much in part of producing the synthetic images.

7.2 The importance of synthetic data generation step

Although it might seem fairly straightforward to produce the synthetic data, it does have intriguing details to be taken into consideration.

First of all, the real-world image background data should not contain any positive instances of the objects to be detected; these objects would be synthetically generated and added. This is rather obvious. What may not be so obvious at first is that we should also include negative classification objects in the background as well. That is, if there are any objects that we do not want to be detected and classified as targets, we need to include

those in the background in the training set in order for our model to learn that they are irrelevant objects in the background. The objects that look similar to our target objects are especially important.

The range of variations in your training data may affect the range of parameters your model can detect. An initial trial with a training set of limited “lightness” of colors would produce a model that may fail with scenes with very good lighting. Although this was a trivial problem that can easily be solved by preprocessing the image frames by automatically making them darker if needed, it would still be a good idea to plan ahead and keep a wider range of variations for the synthetic data generation.

7.3 Thresholds, false positives, and misclassifications

Thresholds, false negatives, and misclassification rates would naturally vary from domain to domain. In this selected application of detection of motorway signs, false positives are very rare with the CNN classifiers, thanks to post-filters. The regions a CNN classifier might trigger a false positive in a motorway scene can be easily filtered by location, size, aspect ratio, and color content. As a result, setting lower thresholds for detection is possible. However, this would also depend on the background images used. When there is an object that the network did not previously see in the training data (a billboard, a specially shaped tree, a special vehicle, a truck with writing on the side, etc.), the model can decide that it looks more similar to a sign than the limited backgrounds it has seen so far.

Misclassifications, however, seem more likely since the two classes of signs look very similar. A single sign can be classified for both classes. Overlapping classifications can be compared to pick the right one. This is, of course, a post-processing action and a feature of the application domain rather than a feature of the model. But it will improve the performance of a model in this domain.

7.4 Comparison with color-based hard-coded algorithms

CNN methods are superior to color-based methods. For one thing, CNNs can provide detection, classification, and localization, all in one pass. Color-based methods can only present possible regions that may contain the object classes, but they cannot

provide a satisfactory classification. The examples in literature that use color-based detection methods, usually follow them up with other classification methods, of which CNNs are one of the recommended alternatives.

Furthermore, color-based methods can produce too many false positives if the target colors are likely to exist in many other areas in image frames. Although these false positives can be filtered in the post-processing stage, the filtering is not perfect and it still takes a toll on computing time. Similarly, since many different colors are used within those signs in Turkey (blue, white, black, brown, orange), color-based filtering gets more complicated and might often produce multiple separate areas for a single sign with many colors and they will register as multiple false positives. However, both methods can also benefit from each other. CNNs can use color-based methods to filter possible ROIs in preprocessing or as a filter in postprocessing to weed out false positives.

CNNs also have the extra advantage of training for variations of color parameters without having access to real data. Color-based methods depend on hard-coded rules; they can only handle variations either through tedious work of hard-code or through some parameter tuning with access to real-world data.

7.5 Final verdict

CNN methods have currently been widely accepted as the best way of detection, classification, and localization in problems with images.

Their weak points in the past were the time it takes them to process, the extra time required for localization, the large amount of data they may require, and the costly process of manually annotating the data. Of these problems, localization and the processing time are solved with recent techniques such as Faster R-CNN, YOLO, and SSD. The required data and the cost of annotation can be solved with synthetic data.

This work presents a specific real-world application where the CNNs can be used by synthetically producing both the training data and its annotations with a very reasonable amount of work. The results are very promising show that the CNN methods can generalize well from synthetic data to real-world tests.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations By Error Propagation (original),” in *Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, 1986, pp. 318–362.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” pp. 1097–1105, 2012.

[3] N. Mayer *et al.*, “What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?,” *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 942–960, Jan. 2018.

[4] M. Kehl, M. Enzweiler, B. Froehlich, U. Franke, and W. Heiden, “Vision-Based Road Sign Detection,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 505–510.

[5] F. S. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, and J. M. Alvarez, “Effective Use of Synthetic Data for Urban Scene Semantic Segmentation,” in *European Conference on Computer Vision*, Springer, Cham, 2018, pp. 86–103.

[6] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for Data: Ground Truth from Computer Games,” in *European Conference on Computer Vision*, Springer, Cham, 2016, pp. 102–118.

[7] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka, “Synthesizing Training Data for Object Detection in Indoor Scenes,” Feb. 2017.

[8] W. Chen *et al.*, “Synthesizing Training Images for Boosting Human 3D Pose Estimation,” in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 479–488.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[11] J. D. Crisman and C. E. Thorpe, “UNSCARF-a color vision system for the detection of unstructured roads,” in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 2496–2501.

[12] U. Franke, C. Rabe, H. Badino, and S. Gehrig, “6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception,” Springer, Berlin, Heidelberg, 2005, pp. 216–223.

[13] H. C. Karaimer, I. Baris, and Y. Bastanlar, “Detection and classification of vehicles from omnidirectional videos using multiple silhouettes,” *Pattern Anal. Appl.*, vol. 20, no. 3, pp. 893–905, Aug. 2017.

[14] F. Zhang, D. Clarke, and A. Knoll, “Vehicle detection based on LiDAR and camera fusion,” in *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, 2014, pp. 1620–1625.

[15] I. Baris and Y. Bastanlar, “Classification and tracking of traffic scene objects with hybrid camera systems,” in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018, vol. 2018-March, pp. 1–6.

[16] A. De La Escalera, L. E. Moreno, M. A. Salichs, and J. M. Armingol, “Road traffic sign detection and classification,” *IEEE Trans. Ind. Electron.*, vol. 44, no. 6, pp. 848–859, 1997.

[17] S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Gil-Jiménez, H. Gómez-Moreno, and F. López-Ferreras, “Road-sign detection and recognition based on support vector machines,” *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 2, pp. 264–278, Jun. 2007.

[18] S. Vitabile, G. Pollaccia, G. Pilato, and E. Sorbello, “Road signs recognition using a dynamic pixel aggregation technique in the HSV color space,” in *Proceedings - 11th International Conference on Image Analysis and Processing, ICIAP 2001*, 2001, pp. 572–577.

[19] H. Fleyeh, “Color detection and segmentation for road and traffic signs,” in *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, 2004, vol. 2, pp. 809–814.

[20] H. Fleyeh, “Road and traffic sign detection and recognition,” *Proc. 16th Mini-EURO Conf.*, pp. 644–653, 2005.

[21] M. de Saint Blancard, “Road Sign Recognition: A Study of Vision-based Decision Making for Road Environment Recognition,” Springer, New York, NY, 1992, pp. 162–172.

[22] P. Sermanet and Y. Lecun, “Traffic sign recognition with multi-scale convolutional networks,” in *Proceedings of the International Joint Conference on Neural Networks*, 2011, pp. 2809–2813.

[23] R. Timofte, K. Zimmermann, and L. Van Gool, “Multi-view traffic sign detection, recognition, and 3D localisation,” in *2009 Workshop on Applications of Computer Vision (WACV)*, 2009, pp. 1–8.

[24] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition,” *Neural Networks*, vol. 32, pp. 323–332, Aug. 2012.

[25] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, “Detection of traffic signs in real-world images: The German traffic sign detection benchmark,” in *Proceedings of the International Joint Conference on Neural Networks*, 2013,

pp. 1–8.

[26] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[27] D. O. Hebb, *The organization of behavior; a neuropsycholocigal theory*. 1949.

[28] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[29] M. Minsky and S. Papert, *Perceptrons: Anlntroduction to computational geometry*. MITPress, Cambridge, Massachusetts, 1969.

[30] P. Werbos, “‘Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences,’ unpublished Ph.D. Dissertation, Harvard University, Department of Applied Mathematics,” 1974.

[31] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 79, no. 8, pp. 2554–8, Apr. 1982.

[32] Y. LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[33] Y. Lecun *et al.*, “Comparison of Learning Algorithms for Handwritten Digit Recognition,” in *INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS*, 1995, pp. 53–60.

[34] L. Yann, B. Leon, B. Yoshua, and H. Patrick, “Gradient-Based Learning Applied to Document Recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] Y. Le Cun *et al.*, “Handwritten Digit Recognition with a Back-Propagation Network,” *Adv. Neural Inf. Process. Syst.*, no. April, pp. 396–404, 1990.

[36] “Prof. Yann LeCun demonstrating the use of Convolutional Neural Networks for recognizing handwritten digits with his ‘LeNet 1’ in 1993.” [Online]. Available: https://www.youtube.com/watch?v=FwFduRA_L6Q.

[37] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[38] M. Everingham *et al.*, “The 2005 PASCAL Visual Object Classes Challenge,” Springer, Berlin, Heidelberg, 2006, pp. 117–176.

[39] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8693 LNCS, no. PART 5, pp. 740–755.

[40] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Lecture Notes in Computer Science (including subseries Lecture*

Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2014, vol. 8689 LNCS, no. PART 1, pp. 818–833.

[41] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014.

[42] C. Szegedy *et al.*, "Going Deeper with Convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[43] R. Girshick, D. Jeff, D. Trevor, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)," in *Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[44] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, vol. 2015 Inter, pp. 1440–1448.

[45] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *The IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[46] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 779–788.

[47] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv Prepr. arXiv1612.08242*, pp. 7263–7271, 2016.

[48] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arxiv*, Apr. 2018.

[49] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9905 LNCS, pp. 21–37.

[50] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," Feb. 2016.

[51] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017.

[52] Y. Bengio, "Deep Learning of Representations for Unsupervised and Transfer Learning," in *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop*, 2011, vol. 27, pp. 17–37.

[53] Y. Bengio *et al.*, "Deep Learners Benefit More from Out-of-Distribution Examples," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 164–172.

[54] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[55] “ONNX - Open Neural Network Exchange Format.” [Online]. Available: https://onnx.ai/.

[56] B. Leibe, N. Cornelis, and K. Cornelis, “Dynamic 3D Scene Analysis from a Moving Vehicle - Leibe, Cornelis, Cornelis - 2007.pdf,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[57] G. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, “Segmentation and Recognition using SfM Point Clouds,” in *ECCV*, 2008, pp. 44–57.

[58] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth, “Efficient Multi-Cue Scene Segmentation,” in *German Conference on Pattern Recognition*, 2013, pp. 435–445.

[59] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[60] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[61] M. Cordts *et al.*, “The Cityscapes Dataset,” in *CVPR*, 2015.

[62] M. Cordts *et al.*, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 3213–3223.

[63] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 year, 1000 km: The Oxford RobotCar dataset,” *Int. J. Rob. Res.*, vol. 36, no. 1, pp. 3–15, Jan. 2017.

[64] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, “The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, vol. 2017-Octob, pp. 5000–5009.

[65] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, “The ApolloScape Open Dataset for Autonomous Driving and its Application,” Mar. 2018.

[66] F. Yu *et al.*, “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling,” May 2018.

[67] H. Caesar *et al.*, “nuScenes: A multimodal dataset for autonomous driving,” Mar. 2019.

[68] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 3234–3243.

[69] P. Goodman, A. Namdeo, F. Galatioto, M. C. Bell, E. Foster, and C. Shield, "Rethinking the Inception Architecture for Computer Vision Christian," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[70] C. C. Nguyen *et al.*, "Towards real-time smile detection based on faster region convolutional neural network," in *2018 1st International Conference on Multimedia Analysis and Pattern Recognition, MAPR 2018 - Proceedings*, 2018, vol. 2018-Janua, pp. 1–6.

[71] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, 15-Jul-2019.

[72] J. Tremblay *et al.*, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2018, vol. 2018-June, pp. 1082–1090.

[73] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[74] S. Vitabile, A. Gentile, and F. Sorbello, "A neural network based automatic road signs recognizer," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, 2003, pp. 2315–2320.

[75] "R-CNN, Fast R-CNN, and Faster R-CNN Basics," *Mathworks, Documentation*. [Online]. Available: https://www.mathworks.com/help/vision/ug/faster-r-cnn-basics.html.