University Of Southampton

Faculty Of Physical and Applied Sciences

School of Electronics and Computer Science

Hierarchical ConvNets for Traffic Sign Detection and Recognition

by

Noureldien Hussein

A dissertation submitted in partial fulfilment of the degree of

Master of Science in Artificial Intelligence

September 2015

Abstract

Traffic sign detection and recognition play important role in the advances of self-driving cars and building more informative maps. In this thesis, we focus on solving the problem of traffic sign localisation, detection and recognition using deep learning methods.

Despite recent literature showed that the problem is solved by applying state-of-the-art machine learning approaches and achieving super- human classification rates 99.4%, we argue that the problem is far from being solved. We mention the limitation of the current approaches and conclude the challenges. Then, we propose methods to overcome these challenges and push forward the current limitations. Then, we present a critical analysis for the performance of the proposed method.

Finally. we compare the performance of the proposed methods against the state-of-the art methods. We then mention the work that was not realised during the project, so it can be carried out in the future.

Acknowledgements

I would like to thank my supervisor, Dr Jonathon S. Hare, for all his support and guidance during and after the master's project; I will always feel grateful for him. He is a true example of the unconditional dedication towards his students.
Also, I would like to thank my examiner, Dr Adam Prügel-Bennett for the eye-opening advices. I learned from him that intellectual honesty and dedication are the key to a genuine scientific legacy.
Finally, I wish to thank the School of Electronics and Computer Science for providing the best facilities to the graduate students. I would like to express my gratitude to the University of Southampton for providing me the opportunity to seek knowledge and fulfil my dreams.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Object detection and object recognition make one of the fundamental problem domains in computer vision. Recent advances in machine learning paves the road for new methods and approaches to be applied to these problems. Traffic signs are typical example for an object to be detected and recognised.

1.1 Business Need

Usually, researchers exert huge efforts to research and solve the toughest scientific problems; unbiased by business needs. Nonetheless, in many cases, these business encourage focusing the research efforts on one of these scientific problems. This results in accelerating the research process, which yields in practical solutions in a reasonable time.

One of these problems is traffic sign detection and recognition. We mention 2 specific needs that accelerated the research efforts for this particular problem.

1.1.1 Autonomous Cars

Cars are a main transportation mean for humans in their day-to-day life. Over the years, this transportation mean has been transformed by the advances in industry and technology. Currently, a main modernisation wave is autonomous cars; also known as self-sufficient, self-driving or driver-less cars. The early attempts to develop autonomous cars can be dated back to 1980s and credited to Navlab, The Robotics Institute, Carnegie Mellon University [38].

Autonomous cars can be formulated as intelligent agents, for the sake of simplicity. As intelligent agents, they need to be wander in the environment, perceive signals from it and react to its events and changes. Vision, or more precisely, machine vision is one of the methods an agent use to perceive the surrounding environment. Traffic signs are one of important elements in this environment, along with traffic lights, road lines, other vehicles, pedestrians, ... etc. t Thus, detecting and recognising traffic signs is a one of the steps towards building autonomous cars.

1.1.2 Driver Assistance

While work is still in progress to achieve the goal of autonomous cars, there is an already realised business goal; the driver assistance system. A typical car driver can be distracted by traffic and not notice few traffic signs. Some of them are important for the driver to know, even after being passed, as speed limit signs. Such system recognises these speeds signs to be displayed on the dashboard. In early 2010, leading car manufactures began to offer the functionality of traffic sign recognition in the driver assistance systems [1, 2].

1.1.3 Traffic Signs on Maps

This particular business need is the main reason behind starting the research project of this thesis. Due to the collaboration between the University of Southampton and Ordnance Survey, the UK mapping agency, the later asked for a proof-of-concept of recognising traffic signs in video footages of streets. These footages are provided with the geo locations where they were taken.

These traffic signs if recognised, is coupled with the geo-locations where they were detected, then projected on the map. This results in the following benefits:

- Enrich the maps with a new type of data and information. The same way a user can search a map for places (e.g. restaurants or hospitals), now he/she can search it for the location of, for example, roundabout signs or speed limit signs.

- Feed the satellite navigation (Sat Nav) systems with the location of the traffic signs. We mentioned before that some of the recently produced cars are equipped with driver assistant systems, which come with traffic sign recognition capability. If the location of traffic signs are added to the maps on the Sat Nav systems, then older cars can enjoy this feature as well, i.e. knowing the location of traffic signs.

1.2 Problem Definition

Researcher have been studying the problem of recognising and locating traffic sign for years. Thus, it can be scientifically formulated or divided into three main parts:

- Traffic Sign Detection
- Traffic Sign Recognition
- Traffic Sign Tracking

In the following sections, we will divide each main point into smaller ones and discuss each of them in details. We only discuss the goal of each point. While later on, we discuss the methods used to achieve these goals.

1.2.1 Detection

Either we are dealing with natural images or video footage, both of them are considered as a traditional spatial domain or search space. The first stage is localisation and detection. For localisation, we are only interested in determining the coordinates of the centre of a traffic sign within the search space (i.e. image/video). While for detection, we want to determine precisely its boundaries.

1.2.2 Recognition

After we precisely determine the boundaries of a traffic sign, the next stage is to recognise or classify it. In general, we want to know which traffic sign it is. In rare cases, we are required to only recognise its super-class.
The speed and accuracy of recognition are the main characteristics of the recognition step. There has always been a trade off between them. None of them is more preferred than the other. It's up to the business need to specify the most important characteristic.

1.2.3 Tracking

In the 2 previous steps we only incorporate the spatial domain. But in this step, we make use of the spatial domain as well. Tracking takes place for detected traffic signs in video footages. Where we link the same traffic sign detected in the successive frames of the video. There are several advantages of tracking.

- Increase robustness and accuracy of recognition. Due to lighting condition of transformation of a traffic sign in one frame of the video, the detector may fail to successful detect it.

- Decrease the computational cost and increase the recognition speed. After detecting and recognising a traffic sign in a specific area of a frame in the video, we can exclude it from the computation in the next frames. We can simply depend on region matching to keep tracking this specific sign until it disappears from the video.

- Accurately calculating the geo-location of a traffic sign. One cannot calculate the depth of a traffic sign using one frame of the video. But using the sizes of a detected traffic sign in several successive frames, we can easily calculate its the depth.

1.3 Traffic Signs Categories

It is very important to present in more details the specifications of the object we are trying to detect and recognise. Traffic signs vary from one country to another. However, in 1968, the United Nations issued an agreement for unifying the specifications of traffic signs [29]. This agreement was soon adopted by the majority of European Union countries.

A special case however is the United Kingdom. Since the research project of this thesis is conducted in a United Kingdom university, we mention the categories of UK traffic signs. According to the Department of Transport, UK Government [12], traffic signs are classified into a handful of elementary types that vary significantly in shape and colour. We call these types: super-classes or categories. Figure

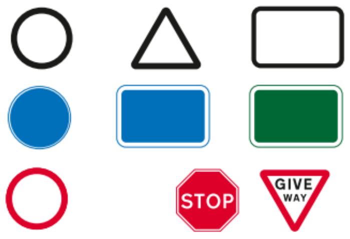


Figure 1.1: Traffic signs in the UK are classified into a handful of elementary types or categories that vary significantly in shape and colour. Reproduced from [12].

Moreover, the following are some of these super-classes and their meaning:

- Mandatory: while circle with blue background. Inside it, there is a specific sign or figure. It generally gives a positive (mandatory) instruction or indicates a route for use only by particular classes of vehicle. Figure

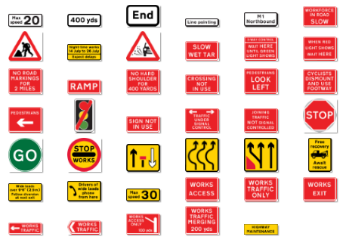
To a great extent, traffic signs fall into one of the aforementioned super-class. However, few of them have odd shapes. For example, the *Stop* sign or *Give a Way* sign in the prohibitory super-class as shown in figure

(a) Mandatory signs

(b) Prohibitory signs

(c) Warning signs

(d) Motorway signs


(e) Roadwork signs

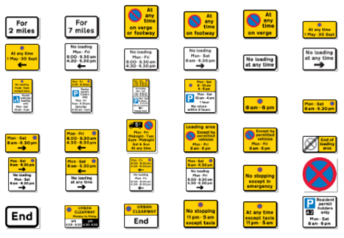

(f) On-street parking signs

Figure 1.2: Examples of traffic signs from different categories (super-classes). In general, super-classes vary significantly in shape and colour, with few exceptions breaking the rules. (a) *mandatory*, (b) *prohibitory*, (c) *warning*, (d) *motorway*, (e) *roadwork*, (f) *on-street parking*.

1.4 Research Datasets

As a typical detection and recognition problem in computer vision and machine learning, we require labelled dataset to build our models. Here, we mention the most notable traffic sign datasets available for research projects.

As we mentioned, the problem is formulated as a series of stages: detection, recognition and tracking. While tracking does not need dataset to be built/trained, both detection and recognition do. Thus, there is a different type of dataset required for the detection and recognition stages.

1.4.1 Detection Dataset

The main purpose of a detection dataset is to provide us the search space and the ground truth. The search space is videos or images that contains traffic sign. While the ground truth is the location and class name of these signs.

The first dataset for detection is the German Traffic Sign Detection Benchmark (GTSDB) [**?**]. It comprises a total of 900 images, 600 for training and 300 for test. The resolution of each is 1380x800 pixels. Each one is a natural image for a road or a street in Germany, in which there might or might not be traffic sign(s). There are in total 846 traffic signs in the 900 images (600 training images with 846 traffic signs and 300 test images with 360 signs). That means on average, an image have 1 traffic sign. But there images with no traffic signs in and other images contain up to 4-6 traffic signs. All the signs belong to only 43 classes.

Along with the images, the dataset contains the annotation or labelling. That is the location of each traffic sign within the natural image. The location is a rectangle determined by upper left corner (x_1, y_1) and lower right corner (x_2, y_2). Also, we are given the category or the super-class name of each traffic sign.

Unfortunately, this dataset is not rich enough for the following reasons. However, we consider this dataset very helpful for our research.

- Traffic signs are only labelled with the name of the super-class. They are not labelled with the name of the class.
- Traffic signs belong to only 3 super-classes: prohibitory, mandatory and warning. There are no signs from any of these super-classes: motorway, roadwork, on-street parking.

The second dataset is the Belgium Traffic Sign (BelgiumTS) [**?**]. Unlike GTSDB dataset which contains only still images, BelgiumTS consists of street view videos for random streets in Belgium. The videos capture the street view from different angles at the same time using 8 cameras in 8 different positions. This is particularly helpful in stereo image or 3D tracking research. These videos are provided with annotations of the location (as a bounding box/rectangle) and class name of each traffic sign that appears in the videos.

From these videos, another subset of the dataset exist. This subset comprises 9006 natural images with 13444 traffic signs corresponding to 4565 physically distinct signs less than 50 meters from the camera. Also the subset comprises negative examples (i.e true negative) of 16045 background images. The dataset can be obtained from [] an all the technical details can be found in [33]

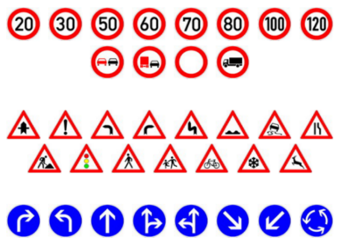
1.4.2 Recognition Dataset

The main purpose of a recognition dataset is to provide us with samples and their target/ground truth. A sample is a cropped image such that it contains only one traffic sign. A ground truth is the class name corresponding to this sign.

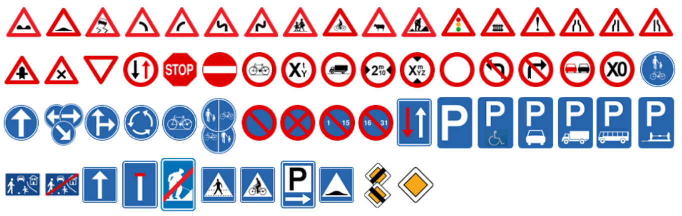
The first dataset is the German Traffic Sign Recognition Benchmark (GTSRB) [39]. It comprises 51841 images of traffic signs (39209 for training and 12632 for test). These signs belong to 43 classes and 3 super-classes as illustrated in figure

The second dataset is the Belgium Traffic Sign for Classification (BelgiumTSC) [33]. It comprises 7125 images of traffic signs (4591 for training and 2534 for test). These samples correspond to the original BelgiumTS Training and 2D Testing parts but restricted to only 62 classes. Figure

It is more rich than GTSDB in terms of the number of represented classes (62 classes in BelgiumTSC versus 43 classes in GTSRB). However, it is still limited to only 4 super-classes: prohibitory, mandatory, warning and on-street parking. Also, 62 is still very small if compared to number classes in real world.



(a) GTSRB dataset



(b) BelgiumTSC dataset

Figure 1.3: Super classes and classes included in traffic sign benchmark datasets. (a) German Traffic Sign Recognition Benchmark (GTSRB) dataset divided into 3 super-classes and 43 classes, (b) BelgiumTS for Classification (BelgiumTSC) dataset divided into 4 super-classes and 62 classes.

1.5 Roadmap

The rest of this thesis is organised as follows:

1. **Chapter 2 – Related Work**: we review the previous work and methods related to both traffic sign detection and recognition. Also, we point out the limitation of each method.

2. **Chapter 3 – Methods**: we present our approach to solve the problem of traffic sign detection and recognition. For recognition, our main methods is Hierarchical Convolutional Neural Network (H-CNN). For detection, our main method is . We point out that we discuss Deep Learning We focus on how our approach might contribute to solving the problem for large-scale datasets (100+ classes). Then, wee discuss the carried out experiments on both GTSRB and BTSDB datasets.

3. **Chapter 4 – Results**: We report our results for the carried-out experiments. Then, we present an objective analysis and critic to these results. Finally, we judge whether or not our methods (presented in chapter 3) were able to overcome the challenges and limitations (presented in chapter 2).

4. **Chapter 5 – Discussion**: In the final chapter, we present the conclusion of the project. We point the missing steps; that we did not realise with-in the time-frame of the project. Finally, We mention the natural extensions of our method and the future work.

Chapter 2

Related Work

In this chapter, we discuss the previous work done by other researchers to solve the problem of both traffic sign detection and traffic sign recognition. Also, for each method, we point out the limitations and the challenges. Later in the next chapter, we will suggest new methods for how to overcome some of these limitations.

As we previously introduced in chapter

2.1 Traffic Sign Detection

The first stage is building a detector for traffic signs. It's job is to explore the search space (i.e. images and video frames) to find the location of traffic signs. This can achieved by exploiting some or all of the visual features of a traffic sign (for example colour or shape).

The detection process itself can be divided into small components, as follows:

1. Detection Proposal: candidates or regions of interests for the detector to work on.
2. Localisation: precisely determine the location of a traffic sign in the search space, represented by (x, y).
3. Detection: Determine the boundary box of the traffic sign, represented by the top left corner (x_1, y_1) and button right corner (x_1, y_1).

2.1.1 Sliding Window

A successful detector can make use of the detection proposals. Nonetheless, it does not have to. One can build a successful detector using only the detection component. In such case, the detector has to be applied to every region in the image in a fashion called sliding window. The number of overlapped windows extracted from an image (n) depends on: the image width (w) and height (h), the dimension of the sliding window (d) and the stride s by which the sliding window is moving; and is calculated as the following:

$$n = \left(1 + \left\lfloor \frac{w-d}{s} \right\rfloor\right) * \left(1 + \left\lfloor \frac{h-d}{s} \right\rfloor\right) \tag{2.1}$$

Figure

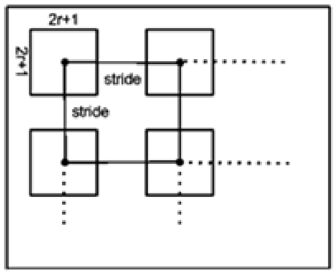


Figure 2.1: Applying sliding window to take samples from the image. Note that several parameters control how extensive we sample from the image. Both the window size and the stride length control if the sliding window will overlap or not.

Let's for example say we want to detect a warning traffic sign in an image with resolution 1380x800. Using sliding window fashion with window dimension = 90 pixels and stride 10 pixels, we extract: $(1 + (1380 - 90)/10) * (1 + (800 - 90)/10) = 130 * 72 = 9360$. Thus, in order to detect a sign, we need to run the detector over $\approx$ 1000 images. The problem gets worse when we don't know the size of the traffic sign in the images. That means a detector has to operate over multiple scales (i.e. using multiple sizes of the sliding window). This adds to the complexity of the problem. The problem becomes even more complex when we search for traffic signs in videos instead of still images. This means that a detector has to run over 10 30 frames per second (fps); according to the video specifications.

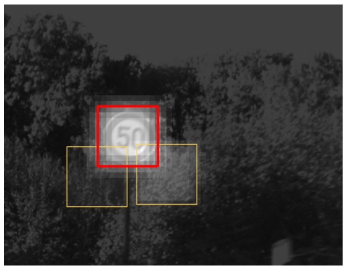


Figure 2.2: Probability (confidence map) when applying traffic sign detector using sliding window approach. Overlapping white boxes are the windows classified as traffic sign, yellow boxes with average confidence while red box with high confidence.

As we mentioned before, the trade-off is always between accuracy and speed of the detector. In case of sliding window, we sacrifice the speed in favour of the accuracy. But the question is, how can we meet both of speed and accuracy? How can we build a real time detector without compromising its accuracy? The answer to this is the detection proposals.

2.1.2 Detection Proposals

A detection proposal is a candidate (i.e. regions of interests with in the search space) for a detector to operate on. This helps in drastically reducing the research space, hence allowing the detector to run much faster than in the case of sliding window. now the question is, how does it work? how we search for proposals? First things first, we have to exploit the features of traffic signs that are easy to find. In this context, researchers use either colour or shape as a feature to search for detection proposals.

Colour-based Proposals

J. Greenhalgh *et al.* [20] used the RGB colour space to search for proposals of traffic signs. Their contribution was instead of using the RGB values, the image is transformed from RGB to normalised red-blue Ω_{RB} such that, for each pixel of the original image, we normalise the red and blue values, then the greater of these two values is used as the final value of the pixel in the normalised red-blue image.

$$\Omega_{AB} = max\left(\frac{R}{R+G+B}, \frac{B}{R+G+B}\right) \tag{2.2}$$

The step of using normalised values of RGB helped in making the method more robust. However, the RGB RGB colour space is inherently known for its variance for lighting conditions. It is not the best space to use for colour-based detection. Other researchers depend on YUV colour space.

Another attempt to exploit the colour feature for detection proposals is done by Y. Wu *et al.* [47], where RGB image is converted into grey scale image. Instead of depending on the traditional linear mapping from RGB to grey scale, they use SVM to achieve non-linear mapping; in the hope to reduce lightning variance. After that, they depend on the luminosity of the grey scale as a probability for the detection proposals.

Another notable attempt to use the colour is done by Y. Yang *et al.* [49] where they built logistic regression models to transform red and blue colours to grey scale; one model for each colour. T map an image, the models compute the probability of a pixel to belong to red or blue. They probability models were trained based on real distribution of red and blue colours taken from traffic signs.

Shape-based Proposals

Another visual feature can be used to search for detection proposals is the shape. Many researchers [49, 20] depend on method called Maximally Stable Extremal Regions (MSER) for extracting detection proposals. MSER in it's simplest definition is a blob detection methods, and it usually operates on the grey scale images. It is not specific for traffic sign detection and can be used to detect other objects. While it is not efficient, it's main advantage is speed. That's why it is used in the researches to build real time traffic sign detection [49, 20].

Other researchers exploit the natural shapes of traffic signs to extract detection proposals. As shown in figure

2.1.3 Detection

The core component in the stage of traffic sign detection is the detector it self. Whether we depend on detection proposal or sliding window, the detector is the component to judge whether an image is a traffic sign or not. There is a huge body of literature discussing detection methods for traffic signs [49, 20, 26, 27]. These methods generally tend to follow a main scheme or approach: hand-crafted feature descriptors and extractors to extract relevant features followed by a feature classifier to rule out false positives. In the following, we discuss some of these feature extractors and classifiers.

Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) has proven success as a general shape detector, for example in pedestrian detection [11] and text detection in natural images [15]. It is currently accepted that HOG is an effective way to capture shape information. This encourages many researcher to apply HOG to the problem of traffic sign detection. J. Greenhalgh *et al.* [20] used HOG to detect traffic signs based on the shape only (circle, triangle, rectangle). Figure

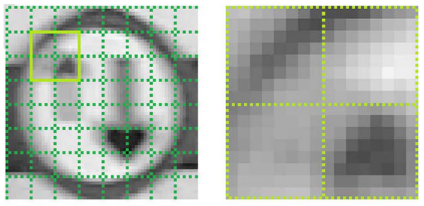


Figure 2.3: On the left, an image of traffic sign is divided into regions (windows), where HOG is applied to each of them. On the right, the result of HOG at each region. Reproduced from [20].

Integral Channel Features

Integral Channel Features (ICF), introduced by P. Dollár *et al.* [14], is another detector used for shape detection. It has proven success several applications, for example pedestrian detection [14]. This encourage researchers to apply this ICF detector to traffic signs. Researchers M. Mathias *et al.* [28] not only applied ICF to detect traffic signs and reach state-of-the-art performance, but also they argued that the problem of traffic sign detection and recognition has been finally solved. Figure

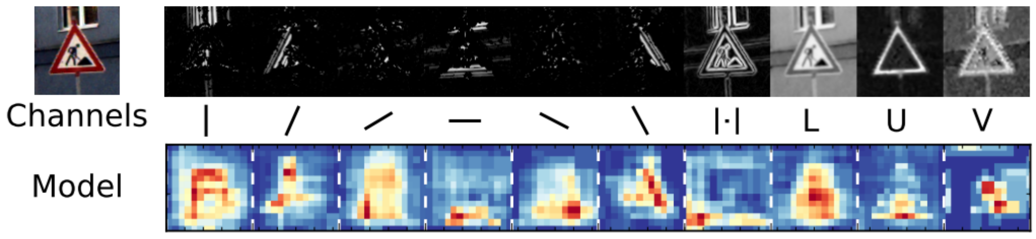


Figure 2.4: At the top, the result of applied the different channels of ICF detector. At the bottom, the heat map reflecting the probability or confidence of the detector at each channel. Reproduced from [28].

2.1.4 Haar-like Features

Haar-like features has proven huge success when used by the researchers P. Viola *et al.* (known as Viola-Jones) for face detection. They built a cascade of detectors using AdaBoost method. They added simple features at the beginning of the cascade to reject weak regions, thus improving the speed of the detector; while complex features at the end of the cascade ensure good detection results. Thus, viola-Jones method achieved the difficult balance between detection accuracy and speed. The same approach is applied to traffic sign detection by C. Lui *et al.*. [27] However, they used the extended haar-like features to enhance the performance. Figure

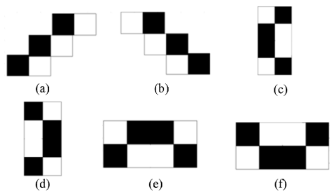


Figure 2.5: Extended haar-like features used for traffic sign detection. Reproduced from [27].

2.1.5 Locale Binary Pattern

Perhaps one of the most controversial detectors used for traffic sign detection is the Multi-block Normalised Locale Binary Pattern (MN-LBP) introduced by C. Lui *et al.* [26]. To find all the traffic signs in a certain image using this detector, we apply the kernel of the detector on this image. At each pixel, the kernel sums the values of eight rectangles around the centre are compared with an average value to obtain a binary sequence. Two kernels are illustrated in figure

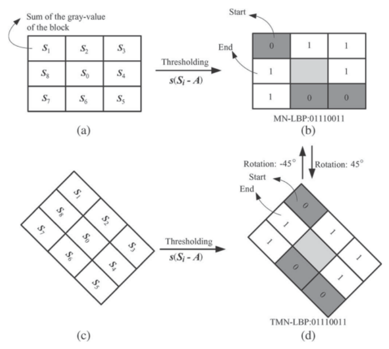


Figure 2.6: Kernels used by MN-LBP (top) and Titled MN-LBP (bottom) features to detect traffic signs. Reproduced from [26].

2.1.6 Deep Learning

Deep learning methods has proven success in several applications in machine learning. In many cases, it out-performed the veteran Support Vector Machine (SVM). For the application of computer vision, one of the deep learning networks seem to excel, it is called Convolutional Neural Network (CNN). Recently, CNN became the baseline feature extractor for many problems in computer vision. They have proven success in several applications for object detection. A. Razaven *et al.* [34] presents an extensive comparison between off-the-shelf CNN against domain specific hand-crafted feature extractors. They pointed out that the CNN outperformed the traditional feature extractors in many application.

Based on their performance and success in the recent years, deep CNNs are applied to many applications. For example large-scale visual detection and recognition [36], joint positions for human poses [42] and text detection in natural images [22, 45, 10]. However, there is no contribution in using CNNs for traffic sign detection.

2.1.7 Limitations and Challenges

After we presented the related work and methods used for traffic sign detection, in this section we point out the limitations of these methods. The sole reason for mentioning the limitations is to encourage the research and push forward the boundaries of science. We do value all the notable work done by the researchers to solve these tough problems.

As we mentioned, the main approach used for traffic sign detection is feature engineering. It depends on either colour or shape or both as the main visual features. Exploiting shape and colour as visual features is not limitation, it is hand-crafting features for them that has limitations. On the one hand, for the shape-based feature extractors, arguably the biggest limitation is the need to hand-craft these extractors. This is not automated process and is labour intensive. Further more, different extractors are needed to detect traffic signs of different super-classes [16, 7]. On the other hand, colour-based feature extractors are prone to failure. That's because colour is an unreliable and depends heavily on the lightning conditions [7].

2.2 Traffic Sign Recognition

The second stage in our project is the traffic sign recognition. The goal is to classify an image, which contains only the sign, to the correct class. This stage is heavily dependent on the existence of well prepared labelled datasets. The images must be cropped so that each image contains only the traffic sign and labelled with the class of the traffic sign.

Huge body of literature discuss several methods to classify/recognise traffic signs. It is the usual case that these methods follow two main schemes: feature engineering and deep learning.

2.2.1 Feature Engineering

In feature engineering, researchers hand-craft feature extractors to extract and describe relevant features from the images. Then they train a classifier in order to classify these features to the corresponding classes. In this approach, the extractors vary according to the characteristics of the object we want to classify. The extractors exploit visual features of the object, like colours, edges, corners, blobs or shapes. In some applications, colour is the most important feature. As an example, classifying fruits, since some of them are similar in shape and size. In other applications, shapes become more important than colours. As an example, scene classification. However, some feature extractors exploit more than one visual feature to achieve better classification.

It is usually the case that feature extractors used in detection are used in recognition as well. In section

2.2.2 Deep Learning

In deep learning, there is no feature engineering involved. The network is able to learn relevant features according to the task in hand. Nonetheless, these learned features are generally edges and corners. In other words, the network tries to learn and develop edge extractors. There are few types of deep learning networks applied to object classification problems, for example: Deep Belief Networks (DBN) and Convolutional Neural Network (CNN). It is the latter that has proven huge success in many applications in computer vision. It is save to say that it outperforms the most powerful problem specific hand-crafted feature extractors. That means an off-the-shelf CNN can replace the traditional hand-crafted feature extractor, as shown in figure

Based on their performance and success in the recent years, deep CNNs are applied to many applications. For example large-scale visual detection and recognition [36], joint positions for human poses [42] and text detection in natural images [22, 45, 10]. However, there is no contribution in using CNNs for traffic sign detection.

It wasn't until 2011 that a classification challenge for traffic signs was announced (German Traffic Sign Detection) that scientists pay more attention to this problem. Probably because the challenge indulged the researchers with a well prepared dataset for traffic signs. Another reason is the leap made by deep learning in object recognition. In the following, we discuss notable deep learning methods traffic sign recognition. But first, let's discuss NN and CNN in more details.

2.2.3 Neural Networks

Artificial Neural Network (NN) is a biologically inspired network that can be trained using labelled dataset. The output of the neuron is a weighted sum of the input signals. It can be mathematically calculated as:

$$y = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b \tag{2.3}$$

$$= \left(\sum_{i=1}^{n} w_i x_i\right) + b \tag{2.4}$$

Where $\boldsymbol{x} = [x_1,\ x_2,\ x_3,\ \ldots,\ x_n]^T$ is the input vector, $\boldsymbol{w} = [w_1,\ w_2,\ w_3,\ \ldots,\ w_n]^T$ is the weight vector and b is the bias. The output can be subjected to an activation function, some examples are: linear, step, ramp, sigmoid or Gaussian. If we take for example the signum as the activation function, the output will be:

$$y = sgn(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b) \tag{2.5}$$

Where sgn is the signum function. During the training of the network, it evolves it's belief by updating some values, specifically the weight $\boldsymbol{w}$ and the bias b. The simplest form of this network is the perceptron. Figure

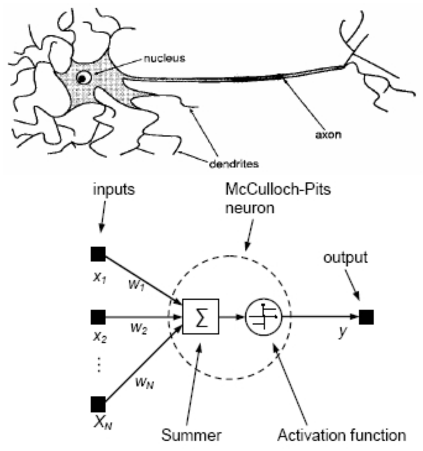


Figure 2.7: The analogy between the 2 counterparts, the biological neuron (top) and the artificial neuron (bottom). Both of them receive input signals from previous neurons, do a small processing (decision making) based on weighted sum of the input signals, and finally results in an output signal. Reproduced from [25].

Perceptron was first used to linearly-seperable data. However, one of it's critics is not being able to generalise for non-linearly separable problems. An example for that is classifying the `XOR` problem. More over, the multi-layer perceptron (MLP) was not easy to train. There was a lack of simple training method. It wasn't until the back-propagation learning algorithm was introduced by D. Rumelhart *et al.* [35] that the MLP began to gain more attention.

MLP

The convolutional and subsampling layers result in extracting relevant features of the input images. The next step is to use a classifier to predict to which category/class the input image belongs to. In our example, Le-Net5, the researchers used MLP as a classifier. It is one of the forms of the neural network, where we stack layers of perceptron networks. These layers are fully connected. The first layer is called the input layer. The middle layers are called the hidden layers. While the last layer is called the output. Figure

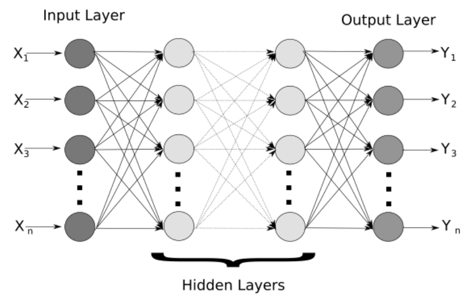


Figure 2.8: Example of MLP with input, hidden and output layers. Note that all the layers are fully connected. Reproduced from [32].

Backpropagation

In order to classify features in the Le-Net5, we have to train the network. For this, we need training algorithm. Which is the method of updating the parameters of the network according to the it's result.

One of the most successful learning algorithms is backpropagation. It was first introduced by the researchers D. Rumelhart *et al.* [35]. It is based on a very simple idea: if the classification is correct, we don't update the parameters. If wrong, we update the parameters in the direction of minimising the error. How we know this direction? By calculating the gradient of the error, starting from the output layer. But how we update the parameters of the hidden layers? The same idea, by using the error we calculated from the output layer and propagating it to the previous hidden layer, hence the naming of the algorithm: back-propagation. It was first applied to the MLP and has proven huge success.

So, how to describe it mathematically? Let l be layer number in the MLP, let $\delta^{(l)}$ be the error at the layer l. Let $J(W, b; x, y)$ be the cost function of the error, where (W, b) are the parameters (weight W and bias b) and (x, y) are the input output pair of the training set. Backpropagation means propagate the error form the front to the back layers, hence the error at layer l can be calculated using the values at the layer $l+1$ as the following:

$$\delta^{(l)} = ((W^{(l)})^{(T)} \delta^{(l+1)}) * f^{'}(z^{(l)}) \tag{2.6}$$

and the gradients of the error are:

$$\bigtriangledown_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T \tag{2.7}$$

$$\bigtriangledown_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)} \tag{2.8}$$

Researchers Y. LeCun *et al.* [23] proved that backpropagation can be used to successfully train CNN. In case of the convolutional or subsampling layers, the error can be calculated as:

$$\delta_k^{(l)} = \text{upsample}\left((W_k^{(l)})^T \delta_k^{(l+1)}\right) * f^{'}(z_k^{(l)}) \tag{2.9}$$

where k is the index of the filter in the convolutional or subsampling layer, f is the activation function.

2.2.4 Convolutional Neural Networks

As we discussed, feature classification can be achieved by MLP. But what about extracting features using deep learning. In the following, we discuss them main building block of the Convolutional Neural Network (CNN). It has become the de-facto feature extractor of many problems in computer vision.

The first introduction to Convolutional Neural Network (CNN) was made by researchers Y. LeCun *et al.* [23]. First, we know that a traditional computer vision problem comprises feature extraction followed by feature classification. The later could be achieved by the biologically inspired NN. It has proven success in many classification problems. The researchers Y. LeCun *et al.* questioned why can't we train the machine to extract relevant features the same way we train it to classify them. The intuition behind the CNN was to build a machine capable of learning relevant features. CNN was applied to handwritten digit recognition and achieve unrepresented classification rates [23]. The researchers experimented several forms of the CNN known as the LeNet. Figure

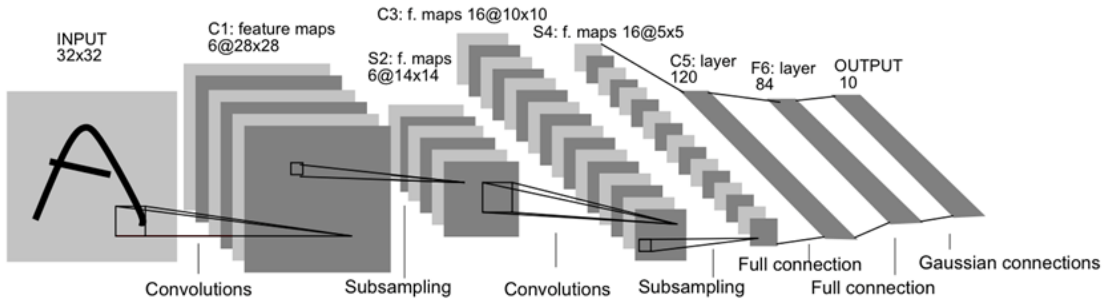


Figure 2.9: Le-Net 5, one of the earliest CNN introduced. It consists of feature extractor and feature classifier. The extractor is 2 convolutional + subsampling layers, while the classifier is 2-layer fully connected. Reproduced from [23].

Convolution

It is the first step in the CNN. Its main purpose is to mix larger information into smaller ones in an ordered way. Think of it as summing-up the information represented by a small local region of an image and representing it using only one pixel in the output image. The output image (i.e. the result of the convolution) is called a feature map. Another meaning for the convolution is that we want to get the most dominant piece of information represented by this local region.

Convolution is controlled by an important parameter, called the kernel size, which is the length of the kernel (local region) that's is going to be summed up and represented by one pixel. Of course the bigger the kernel, the smaller the convolved image. Because it means we want to sum-up the information of larger local patches (regions) into one pixel.

So, how do we convolve an image? As in the case of Le-Net5, let's start by an input image of 32x32 pixels. As we can see in figure

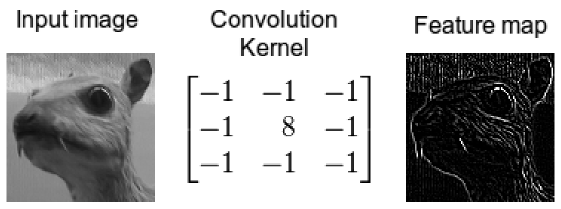

Input image
Convolution Kernel
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
Feature map

Figure 2.10: On the left, the input image (greyscale). In the middle, the kernel used to convolve the input image, also called the filter. On the right, the result of the convolution, also called the filter map. Reproduced from [13].

One layer of convolution can contain as many as 100 kernels (i.e. filters). It is up to the researcher to determine how many kernels he/she needs. Currently, there are no solid mathematical foundations for the number of kernels used in a convolutional layer. However, usually researchers add from 10 to 200 filters in the first convolutional layer.

From where did we get the values of the kernel? In the beginning of the training of the CNN, we initialise these values. There are different techniques for how to initialise the values. One of them is to initialise them as zero. Another way is to initialise them randomly within the range -1 to 1. It is found that random initialisation is better than zero-initialisation, learning coverages faster.

It is important to mention that the weights and bias values are shared among the same kernel, which means they are shared for each filter map. The reason behind this is that images are stationary. Features found in a local region of the image are likely to be found in other local parts. So, if the kernel (filter) acts as a feature extractor, so it make sense to use it all over the image. Researchers I. Sutskever *et al.* [40] discuss in details initialisation methods for the parameters of CNN, specially in the deep learning applications and how it affects the learning of the network.

Subsampling

The next step after convolution is subsampling. We want out network to be location invariant. In other words, we want to improve the network capacity to the changes in location of the features in the image. We don't want an image of, for example number 7, to be miss-classified because the location of the number changed 5 pixels to the right. Subsampling offers a solution to this. In this step, we down-sample the information represented in the convolved image. For example, we down-sample every local region of 2x2 into only one pixel. This means that the network will activate if any of these 4 pixels is the has the right value. Figure

The parameter controlling the down-sample factor is the kernel size of the subsampling. As in our example, figure

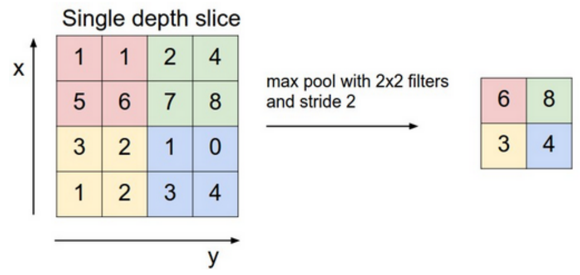


Figure 2.11: On the left, the input image (greyscale). On the right, the result of the subsampling, after using max-pooling with 2x2 kernel. Reproduced from [17].

Convolution and Subsampling

Multiple level of representation is one of the goals of deep learning. In the case of CNN, it is applied by repeating the convolutional and subsampling layers many times. Traditionally, if we have a CNN with 3 or more layers of convolution and subsampling, then it is called deep network. In the example of Le-Net5 network, as in figure

2.2.5 Deep Networks

After we have discussed the main building blocks of deep learning used in traffic sign recognition, let's discuss the notable approaches applied to this specific problem. After the announcement of German Traffic Sign Recognition Benchmark (GTSRB) competition [39] in 2011, it attracted bright minds. Many researchers excreted notable efforts in this problem. The top scoring methods and approaches are deep learning, depending mainly on deep CNN. In the following, we discuss them.

Mluti-column DNN

Researchers Ciresan *et al.* [8] suggest a committee of Deep Neural Networks (DNN), known as Multi-column DNN (MC-DNN), trained using the same training set. Previous experiments suggest that different networks trained using the same training set will develop the same belief, i.e will result in the same performance. Thus, the researchers subjected the training set to stochastic distortion before each training epoch. Which means that each network in the committee is train using practically different training set. Figure

It is to be mentioned that the researchers depend on the coloured images for training their network. But in order to avoid the colour variance problems of the RGB space, they mapped the RGB coloured images to another colour space called Lab-Space. The intuition behind this is that Lab-Space has intensity as one of its components. This technique is very similar to changing the RGB to YUV which have one of it's component as the luminance (Y). Beside that, all the input images were pre-processed before training the network. The pre-processing comprise the following steps:

- Image Adjustment (**Imadjust**)
- Histogram Equalization (**Histeq**)
- Adaptive Histogram Equalization (**Adapthisteq**)

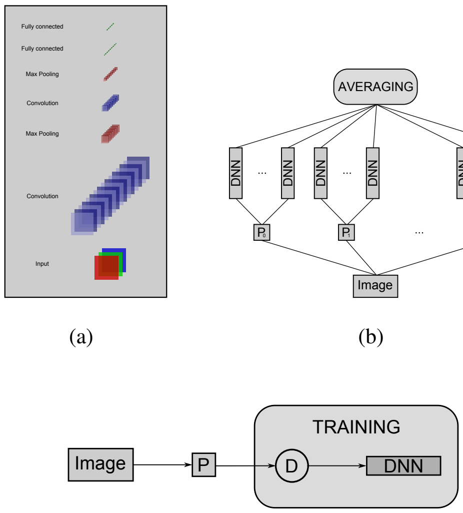


Figure 2.12: (a) Architecture of DNN. (b) Architecture of the committee of DNN, known as Multi-column DNN (MC-DNN). (c) Preprocessing the input images is done only once, whereas distortion is done stochastically before starting a new training epoch. Reproduced from [8].

– Contrast Normalization (**Conorm**)

Figure

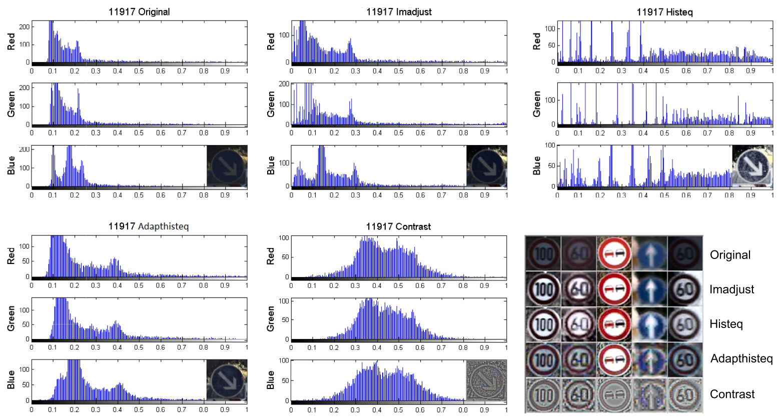


Figure 2.13: The steps of preprocessing applied to the images of the training set. Notice that the histogram is better distributed after pre-processing than before it. Reproduced from [8].

Multi-scale CNN

Researchers Sermanet *et al.* [37] used multi-scale CNN to solve the same problem. Figure 2.1 shows the architecture of the used network. They fed the fully-connected part of the network with features from 2 different scales (at stage 2). The intuition behind this step is to increase the network capacity for the input invariance and to enrich feature representation before classification. As a result, they reached accuracy of 98.3% in the GTSRB competition.

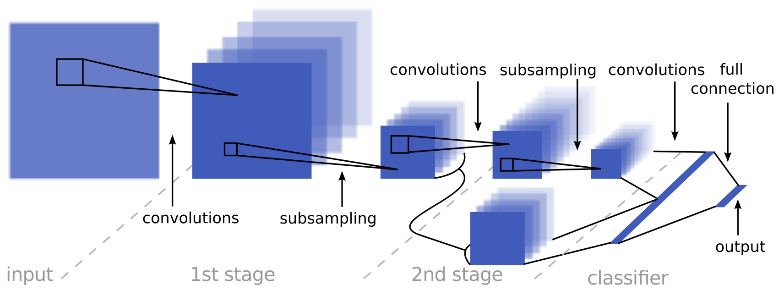


Figure 2.14: Layout of multi-scale CNN. Notice that the MLP classifier is fed by features from 2 different convolutional layers. Reproduced from [37].

In their efforts to avoid the variance problems of using the RGB space, the researchers transformed the images to YUV colour space. For the pre-processing, they only normalised the Y channel. They left a room for doing more research on normalizing the other channels and how would this effect the overall performance of the network.

Another important step is used by the researchers is augmenting the given training set. The reason is to increase the network capacity and result in more robust network. The network is likely to experience these deformations in the test set. The following deformations were applied to each image in the training set:

- Position: [-2,2] pixels.
- Scale: [.9,1.1] ratio.
- Rotation: [-15,+15] degrees.

The researcher suggested that other realistic perturbations/deformation could have been applied. They argued that these perturbations would have probably increase robustness of the network. These are some examples: affine transformations, brightness, contrast and blur.

2.2.6 Feature Classification

MLP is the de-facto classifier used in the CNN to classify the features extracted from the convolutional and subsampling layers. However, other classifiers can be used. Huge body of literature discuss the usage of other classifiers in the problem of traffic sign recognition. In the following, we discuss the most notable contributions/methods.

Support Vector Machine

Replacing the MLP by other classifiers in the CNN can be achieved in other different fashions. Y. Tang [41] replaced the MLP with a linear SVM. This resulted in improving the network performance. In their experiment using MNIST dataset [24], the error rate was reduced from 0.99% using Softmax to 0.87% using DLSVM. Let's discuss the SVM in more details.

Support Vector Machine (SVM) has proven success in classification problems. In its simplest form, the linearly separable binary case, it can learn the optimal hyperplane that separate the data into 2 classes. optimality means maximising the margin (separating distance) between the hyperplane and the data samples. Figure

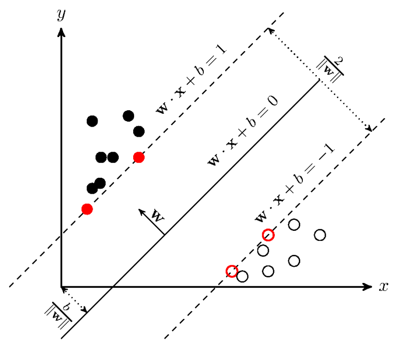

y
x
$\mathbf{w}\cdot\mathbf{x}+b=1$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=-1$
$\mathbf{w}$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$

Figure 2.15: Example of SVM, where input data is separated into 2 classes. SVM find the optimal hyperplane separating the input classes linearly.

Suppose we have a training set consists of input X and target Y. Suppose we are working in a 2D space where the input sample $X = (x1, x2)$ and the corresponding output (target) Y is either 0 or 1 according to the class it belongs to (binary classifier as we mentioned). The objective of the SVM is to maximise the margin between the separation hyperplane and the data samples. It can be formulated as a quadratic optimisation problem. In this simplest form, it ca be mathematically described as the following:

$$\underset{w',b'}{\text{minimise}} \qquad \frac{\| \boldsymbol{w}' \|^2}{2} \tag{2.10}$$

$$\text{subject to} \qquad y_k(\boldsymbol{w}'^T \boldsymbol{x}_k - b') \geq 1,\ k = 1, 2, 3, \ldots, P. \tag{2.11}$$

where $\boldsymbol{w}, b$ are the weight and bias describing the separation hyperplane, P is the number of input samples in the training set.

Of course the simplest case of the SVM (binary and linearly separable) is not of much usefulness in real life. Most data belongs to many classes and are not linearly separable. However, the SVM can tackles these cases. For multi-class problems, the SVM can be trained in a fashion called: 1-versus-all. For the separability problem, the introduction of the slack variable η helps in relaxing the optimisation problem and obtaining a good classifier. It is also known as soft margins. SVM also can solve the linearly non-separable problems by using the kernel trick. In which, the data is projected to higher dimensional space where it can be linearly separated using a hyperplane. Then, it can be formulated as an optimisation problem, mathematically described as:

$$\underset{\boldsymbol{\alpha}}{\text{minimise}} \sum_{k=1}^{P} \alpha_k - \frac{1}{2} \sum_{k,l=1}^{P} \alpha_k \alpha_l y_k y_l K(\boldsymbol{x_k}, \boldsymbol{x_l}) \tag{2.12}$$

where K is the kernel function that maps the data to higher dimensional space.

Researcher Y. Tang [41] proved that the overall performance of the CNN can be improved when replacing the MLP classifier with SVM classifier. That's the reason we mention using the SVM in this thesis.

Extreme Learning Machine

Zing *et al.* [51] used CNN. However, instead of using Multilayer Perceptron (MLP) or Support Vector Machine (SVM) as the classifier of the convolutional features, they used Extreme Learning Machine (ELM). They argued that this step results in better performance in classifying the features, hence, improves the overall performance of the CNN. Another advantage of using ELM is significantly decreasing the training time of the classifier.

2.2.7 Real Time Recognition

Another contribution is done by Yang *et al.* [49], who discussed the ability of carrying out both traffic sign detection and recognition in real time using conventional hardware.

2.2.8 Limitation and Challenges

Mathias*et al.* [28] questioned if the traffic sign problem is considered as solved, specially when the state-of-the-art deep learning methods came close to human classification rates [8, 37]. We argue that the problem is yet far from being solved, for the following reasons.

After we discussed the state of the art methods used in traffic sign recognition, it's save to mention their limitations and challenges. In the next chapter, we will present the methods presented by the thesis to overcome some of these limitations.

However, given the time-limit of this work, we were not able to present and experiment methods to overcome all of these challenges. But we will give our suggestions in the last chapter; conclusion and discussion.

Number of Classes

The performance of most of the recent work is benchmarked using the well-cited GTSRB dataset [39], which comprise only 43 classes. Not only the number 43 is far from reality, where 100+ classes exist, but also these 43 classes are represent only 3 super-classes of traffic signs: prohibitory, mandatory and danger. Figure

The situation does not change if we consider another popular benchmark dataset, BelgiumTS [33], which comprises 3 super-classes and 63 classes. For further overview on how rich (in terms of classes and super-classes) traffic signs are in real life, please refer to Vienna Convention on road signs and signals [19, 29, 30, 46].

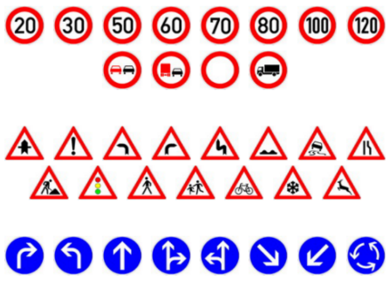


Figure 2.16: All the 43 classes of the GTSRB dataset, divided into 3 super-classes: prohibitory, danger and mandatory. Reproduced from [49].

Further more, we cast doubt on the network ability to achieve the same high accuracy rates when the number of classes per a super-class increases from tens to hundreds. Worth mentioning that the number of feature maps in the network layers grow exponentially for each new class added.

Deformed Data

In the aforementioned methods and benchmark datasets, the distortion in the images is somewhat limited. Also, it is a common practice to subject the training data to 3 different types of distortion before training [8, 9, 37], in order to increase the network capacity and improve its generalisation performance:

- Scaling with factors from 0.9% to 1.1%
- Rotation with angles from 15° to -15°
- Perturbation in position with -2 pixels (left, right, top and bottom)

However, what we meant here is a more severe type of distortion. To be specific, figure

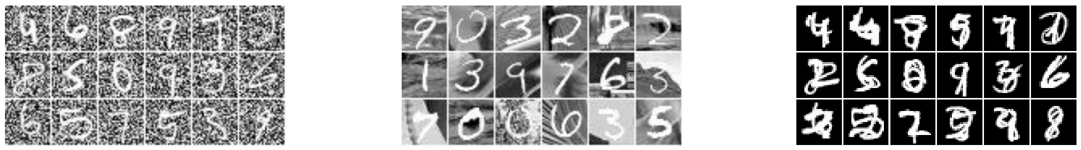


Figure 2.17: Three different types of never-addressed-before input deformation in traffic sign recognition problem. From left to right: noisy background, random background and overlaying characters. Reproduced from [44].

Chapter 3

Methods

In this chapter, we propose the methods to overcome some of the aforementioned limitations and challenges of traffic sign detection and recognition. Due to time-limitations of the research of this thesis, we couldn't overcome some of these limitations and challenges. So, we will discuss them in the last chapter, conclusion.

As we discussed in the previous chapters (

3.1 Traffic Sign Recognition

In the GTSRB competition, deep learning achieved super-human recognition levels, with success rates 99.5% [8]. However, the database comprise only 43 classes of traffic signs. Moreover, the signs belong to only 3 super-classes: mandatory, prohibitory and danger. This is far from real world situation where hundreds of classes and more than 6 super-classes exist.

Suppose we want to build a CNN to classify data into m classes. Several parameters are crucial in order to successfully build a CNN. These parameters are:

– Number of convolutional layers (c) and subsampling (s) layers
– Number of filters (feature maps) in each convolutional layer ($f_{c1}, f_{c2}, \ldots, f_{cc}$)
– Number of filters (feature maps) in each subsampling layer ($f_{s1}, f_{s2}, \ldots, f_{ss}$)
– Number of hidden layers in the MLP classifier (h)
– Number of neurons in the input layer of the MLP (n_i)

– Number of neurons in each of the hidden layers of the MLP classifier $(n_{h1}, n_{h2}, \ldots, n_{hh})$

Currently, there are no strong mathematical foundations that link or rule these parameters together. Researchers depend on exploring the parameter space in order to find the optimal values and build the best CNN. Searching the parameter space can be carried out manually or automatically. Nonetheless, we read huge body of literature for building CNN for several object recognition problem [8, 9, 10, 22, 36, 37, 41, 42, 45, 51] We conclude from that the following:

– The more the categories in the dataset, the bigger the network we need to build.
– It is safe to say at the worst case, the network capacity increases ***linearly*** as the number of categories increases.

Based on the previous conclusion, we thought of how can we increase the network capacity without increasing it's size? Thus, we propose Hierarchical CNN (H-CNN). Hierarchical approach has been used in different machine learning methods to reduce the complexity of the problems with many classes to recognise, traffic sign detection and recognition is not exception.

Ensemble methods share some characteristics with the architectural approach. For example, boosting methods build a pipeline of classifiers. The problem complexity is reduced by ruling out the weak candidates using week classifiers in beginning of the classifier pipleline. While more complex classifiers are placed at the end to ensure more accurate classification results. Thus, one can safely say that ensemble methods share some characteristics with the architectural approach; depending on the basic and dominant visual features of the object for early assessment/judgement. The only difference is that learning in the ensemble methods is completely automatic. While in H-CNN there are decisions are made manually to lay out the hierarchy.

Random forest is an example of ensemble methods. It is applied to traffic sign recognition. Researchers J. Greenhalgh *et al.* [21] used random forest to achieve state-of-the-art results for both detection and recognition. Figure

Also, in deep learning, hierarchical approach is utilised by the researchers Y. Zhicheng [48] used hierarchical Deep CNN (HD-CNN) to solve the problem of large scale visual recognition. They embedded the out-of-the-box CNN into a hierarchy based on the category of the objects. An HD-CNN separates easy classes using a coarse category classifier. Later

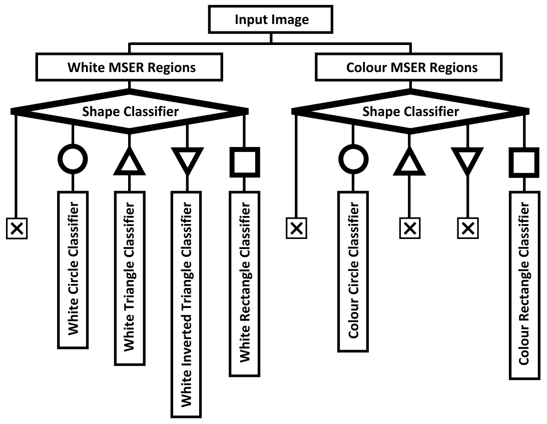


Figure 3.1: The cascade of the classifiers used to build the random forest, which is used to classify traffic sign belongs to 100 classes. Reproduced from [21].

on, the network distinguish difficult classes using fine category classifiers. It looks like this approach is a hybrid of ensemble methods and deep learning methods. Figure

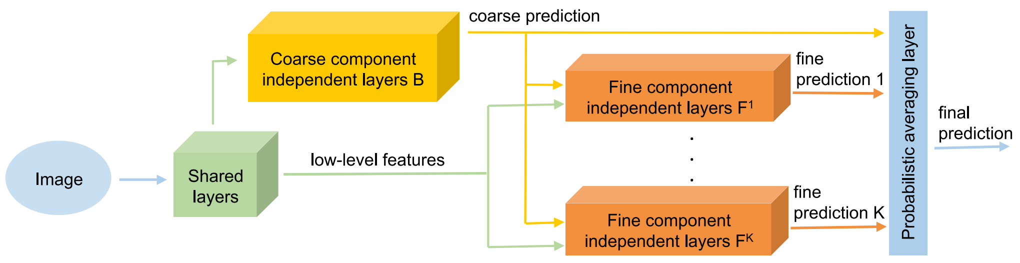

coarse prediction
Coarse component independent layers B
Fine component independent layers F¹
fine prediction 1
Image
Shared layers
low-level features
final prediction
Probabilistic averaging layer
Fine component independent layers Fᴷ
fine prediction K

Figure 3.2: The layout of the HD-CNN. We notice the coarse-to-fine layers work as a cascade of classifiers. It resembles boosting approach. Reproduced from [48].

Before presenting our method, there is one last contribution about hierarchical approaches need to be discussed. Which is the work done by researchers R. Girshick *el al.* [18]. They a

3.1.1 Hierarchical CNN

Based on the previous introduction about the importance of architectural approach, how it helps in reducing the complexity and how it has been applied before to different machine learning methods, we present the C-NN for traffic sign recognition. Out approach depending of directly exploiting the main visual feature of traffic signs: the shape. The colour may be considered as another dominant visual features for human. But it has proven less importance for the machine. Researchers [37] have done experiments using both grayscale and coloured images (YUV). They reported very slight to no difference in performance. That's why we totally neglect the colour as a feature.

So, how do we exploit the shape of the traffic sign? We notice that the traffic signs are shape-based, based on the super-class. Take for example the prohibitory: it is a circle with white background and red circumference. Another example is the warning signs: they are triangles with white background and red edges. So why don't carry out the classifcation on 2 separate steps: First, classify all the traffic signs to their super-class. Second, classify each super-class to their classes. Each step is considered as a classification problem by its own. It can be solved by a out-of-the-box CNN. Figure

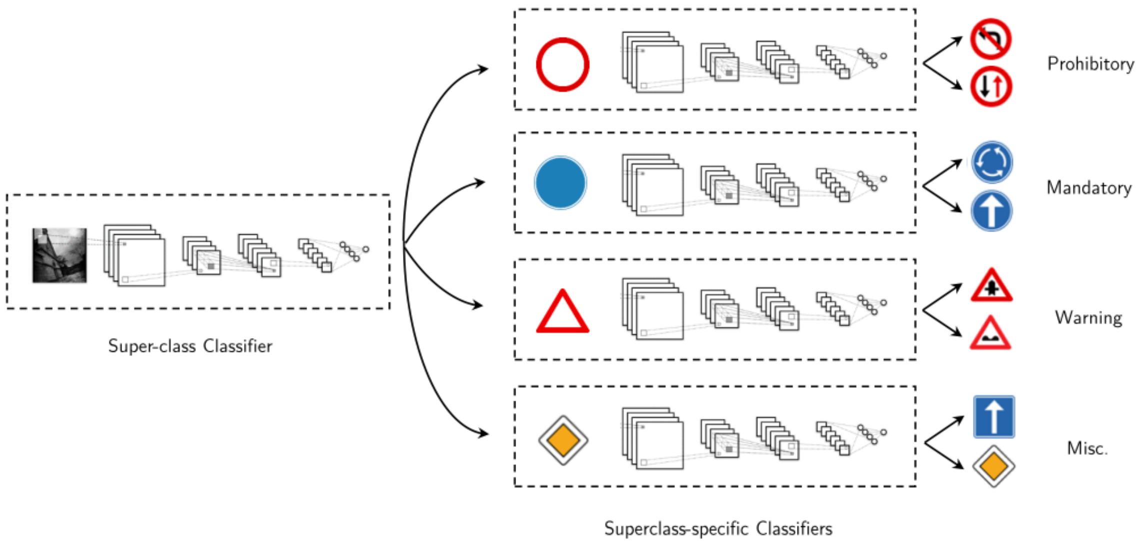


Figure 3.3: The layout of the proposed C-NN. In the first layer of the hierarchy (left), a traffic sign is classified to its super-class. In the second layer (right), it is classified into its class.

The main building block of each classifier is out-of-the-box CNN. We have previously mentioned (in the beginning of this section) several parameters to build a successful CNN. Researchers [] discussed and applying an automated to search for the optimal value of these parameters in the parameter space. However, in our experiment, we used a typical CNN in all the H-CNN. Figure

Before we start talking about the details of the network, let's first discuss an important step that has to be done before training the images. The step is image pre-processing.

3.1.2 Pre-Processing

We depend on images of traffic signs obtained from the GTSRB dataset [39]. The images given in this dataset are coloured, in RGB. We do the following to the images:

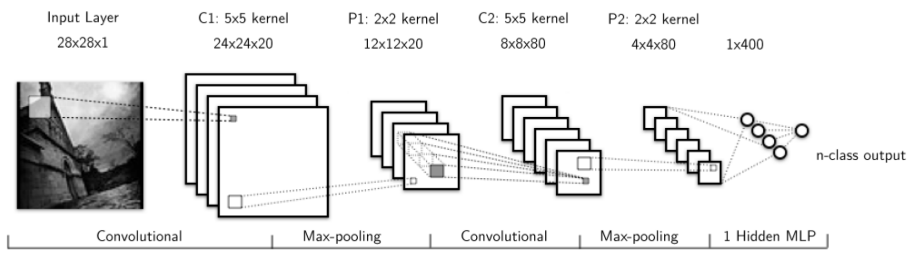


Figure 3.4: The details of the CNN used typically to build the H-CNN. The parameters of the network were chosen manually. Search for the optimal parameter values is done manually. Reproduced from [17]

- Convert the image to grayscale. As we mentioned before, we took the decision neglecting the colour as a feature.
- crop the images so the boundary of the image aligns with the boundary of the traffic sign with in it. Luckily, we were given the ground-truth of these boundaries.
- Rescale the image to be squared (if needed).
- Resize all the images to be of the same length (28 pixels). We take the decision of training the CNN with 28x28 input images.

Now, the images are ready for few more image processing steps. We call these steps: pre-processing. They are as the following:

- Image Adjustment (**Imadjust**)
- Histogram Equalization (**Histeq**)
- Adaptive Histogram Equalization (**Adapthisteq**)
- Contrast Normalization (**Conorm**)

The reason behind pre-processing is to make the image invariance to brightness as possible. Also, this helps in obtaining a well-distributed histogram. Eventually, extracting more robust informative features from the training images. This step is equivalent to normalisation of training data before using it in SVM or linear regression. Figure

Another step applied by other researchers in the same problem is to augment/jitter the dataset [37, 8]. They argued that this step is helpful in obtaining more training samples out of the original/given training samples, the GTSRB dataset. They said enriching the training set helps in increasing the network capacity and results in better performance over the test set. However, we didn't augment the data for these reasons:

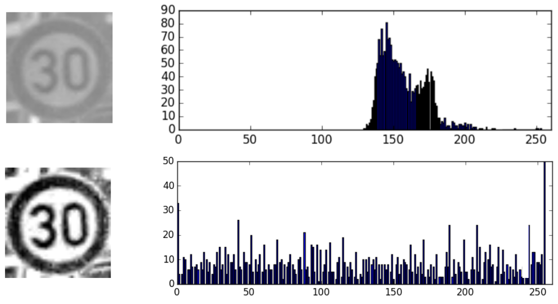


Figure 3.5: Effect of pre-processing on the traffic sign image. This results in brightness invariance.

- Enough training samples was given. On average, every super-class contains 300-500 training samples.
- Our goal is a proof-of-concept for the proposed CNN. We don't aim to reach the state-of-the art performance.

The last step before feeding the image to the CNN is normalisation. In order to achieve smooth learning and prevent weight overshooting, we normalise the pixel values of the image from $[0 : 255]$ to $[-1 : 1]$.

3.2 Traffic Sign Detection

We presented in the previous chapter

We reviewed a huge body of literature discussing the state-of-the-art object detection methods. We experimented the viability of applying some these methods to detecting traffic sign. We concluded that one approach in particular can be further investigated. It seems viable. This approach is called OverFeat [36]. In the following, we illustrate how this approach works. Then we present how we apply it to the problem in hand; traffic sign detection.

3.2.1 OverFeat

CNN can be used for detection. It was applied in different problems, for example English text detection in natural images [15, 45], Thai text detection [42]. The main approach followed in these applications is sliding window. While it results is outstanding performance and achieve state-of-the-art detection rates, it suffer from the problem of complexity. Having to split the natural image into overlapping regions and doing this on different scales, results in huge number of images to be processed by the CNN. Sliding window means we are using brute force to explore the search space; which is the natural image.

To put this into context, let's for example say that we want to detect an object (a traffic sign for example) in a natural image of size 1380x800. Regardless of the size of the traffic size in this image, we need to search the image at different levels (i.e. at different resolutions), let's say at 6-8 levels, with down-scaling factor of 0.5. Then if we consider a sliding window of size 40x40 and stride 10, then we need to run the CNN over $\sim 12,000$ regions. While there are some steps that can decrease the computation, still this is very high cost we have to pay to do detection with CNN using sliding window.

That's why OverFeat avoid exploring the search space with sliding window. So, how does it work? A location of an object is determined by its boundary box, more precisely using 4 variables: $x1, y1, x2, y2$, where $(x1, y1)$ is the top left corner of the boundary box and $(x2, y2)$ is its bottom right one. OverFeat deals with predicting the location as a regression problem. We train a MLP classifier, in a regression fashion, to predict the location of a traffic sign with-in a fixed-size region. Here are the steps of applying OverFeat to achieve prediction:

– A deep CNN is trained using the ILSVRC dataset, which contains millions of training images classified into 1000 category. After training, this network would have learned a strong, general-purpose feature extractor. Figure **??** shows some of the objects you expect to find in the ILSVRC dataset.

– Another CNN is built. Its feature extraction part is not trained. However, its parameter (weight w and bias b of the filters in convolutional and subsampling layers) is initialised by the weights obtained from the CNN in the previous step. The classification part is trained in a regression fashion to predict the location of the object in the images. The target (the object location) is expressed as: $x1, y1, x2, y2$.

Overfeat has since proven success. It achieved state-of-the-art performance in ImageNet Large Scale Visual Recognition Challenge 2013 (ILSVRC2013). Also, it was applied to solve several detection problems. For example, Researchers *et al.* body joints detection in action images [22].

3.2.2 Deep Learning for Traffic Sign Detection

After we have discussed OverFeat as, possibly, one of the best deep learning methods for object detection, we decided to pursue it and apply it to our problem. However, there is an important difference has to be taken into consideration:

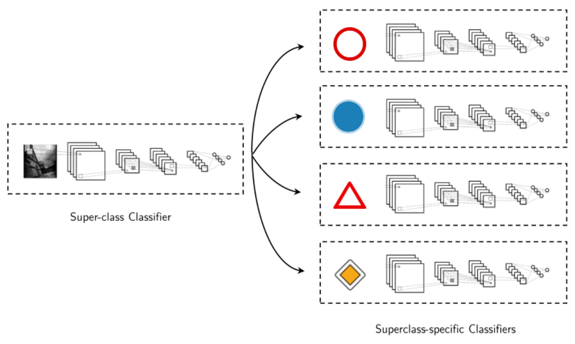
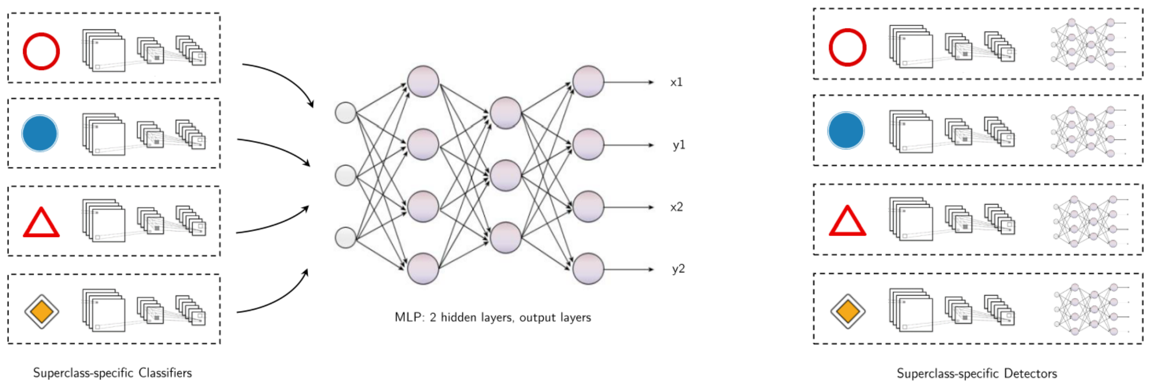


Figure 3.6: The H-CNN we previously used in traffic sign recognition. For detection, we make use of the feature extraction part of each superclass-specific CNN.


x1
y1
x2
y2
MLP: 2 hidden layers, output layers
Superclass-specific Classifiers
Superclass-specific Detectors

(a) Prohibitory signs (b) Warning signs

Figure 3.7: The steps to achieve detection, inspired by OverFeat [36]. (a) Initialise the feature extractor part of the network with parameters from other previously trained CNN classifiers, and we train the feature detector part (MLP) in regression fashion to predict the traffic sign location. (b) We need to build a detection CNN for each super-class, where the CNN acts mainly as a shape detector.

- Size of object: OverFeat was originally applied to ILSVRC competition where the goal is to detect general objects in images. The fact is the size of these objects are usually 30-60% of the image itself. However in traffic sign detection, it is a usual case that the size of the sign is about 5-15% of the image. Figure **??** shows example of typical image from ILSVRC and another one for traffic sign.

Training Samples

We need to sample images for training. We are given the German Traffic Sign Detection Benchmark (GTSDB) dataset. For a certain traffic sign in a certain natural image, we take several samples where the traffic sign lies with-in the image sample. We do this for different positions of the sign with-in the regions and for different scales. The resulted samples have to be resized to 80x80 pixels to be convenient with our detection CNN. Figure

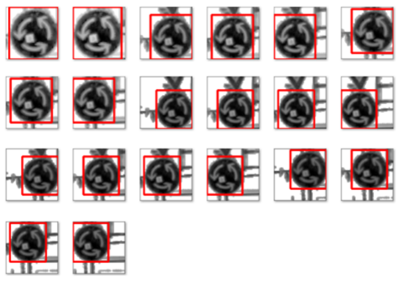


Figure 3.8: Sampling images for detection. Samples are extracted from GTSDB dataset. For a traffic sign, we sample regions with different sizes and different locations.

So, how do we trained OverFeat in our case? The following are the steps:

- We build the detection CNN. Each super-class has a its won detection CNN. The feature extraction part (convolutional and subsampling layers) will not be trained. It will be initialised with parameters (weight w and bias b) previously obtained from recognition CNN, as shown in figure

  It is to be mentioned that all the sampled training images were subjected to pre-processing. The same pre-processing we illustrated before in subsection

3.2.3 Experiments

The following are the experiments we carried out in order to apply OverFeat in traffic sign detection and to refine this approach. This is the first time deep learning is applied to traffic sign detection.

Sampling with True Positives and True Negatives

IN the first detection CNN we built, we trained it using samples of only true positive. In other words, the network built a hypothesis to predict the location a traffic sign in a specific region, but it was considering some background regions as traffic signs as well. The result of such detector is shown in figure

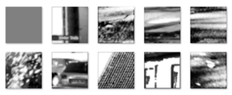
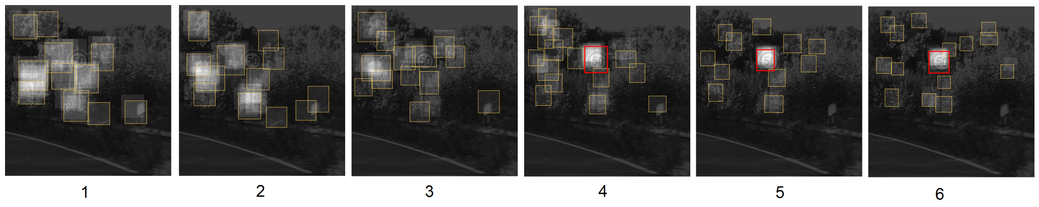


Figure 3.9: Examples of true negatives sampled from the background of GTSDB dataset.

We concluded that the network didn't built a hypothesis about the background as it was not trained with true negatives in the first place. So, we sampled true negatives (as shown in figure


(a) CNN trained with true positives only.

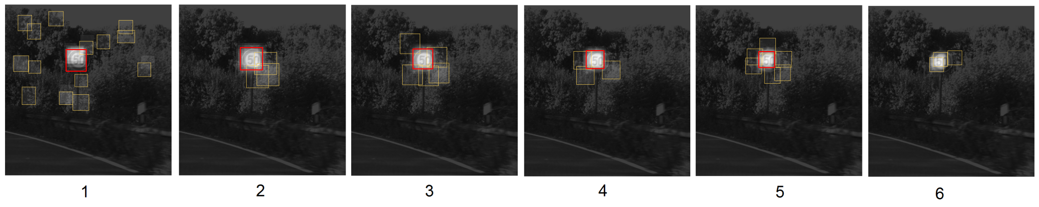

(b) CNN trained with both true positives and true negatives.

Figure 3.10: Detector in action from higher to lower scale (left to right). On the top, result of the CNN trained with only true positives. Despite the traffic sign is eventually detected, the detector gives false positives for background regions. Red and yellow boxes mean high and low detection confidence respectively. At the bottom, result of the CNN trained with both true positives and true negatives. We can see clearly that the network rejected most of the background.

Detection Proposals

The main purpose of detection proposals is to overcome the complexity of searching. Generally, traffic signs are confined in natural images, known for their big sizes. For example, the image provided by GTSDB dataset are 1380x800 pixels.

So, how detection proposals work. They work as an initial guidance for the detector. They try to find the regions of interests. In other words, the regions with high probability of finding traffic signs. Mainly, colour is considered as the dominant feature for detection proposals. While it is known for it's high invariance, researchers come up with different ways to overcome this high variance.

In our case, we tried shape-based detection proposals. We used a robust circle detector using Hough Transform. This helped us in reducing the search space, hence reducing the complexity. Since it is circle detection, it was helpful for only mandatory and prohibitory signs. However, we didn't have the chance to build a robust triangle detector for warning signs.

Chapter 4

Results

In the previous chapter

4.1 Traffic Sign Recognition

We presented the H-CNN in the hope to achieve large-scale traffic sign recognition, where the real life is filled with thousands of different categories/classes. We argued if we classify first to super-class then to class we will reduce the size of the network, reduce training time, reduce the complexity of training and hence, be able to address more classes. Table

Table 4.1: Comparison of the size of networks, between our method and the top-scoring methods in GTSRB competition.

Method	**Network Architecture**	**Size**	**Input**
Multi-Column CNN [8]	3 Conv [20, 40, 800] + 2 Hidden Layers * 8 CNN	40e6	48x48
Multi-Scale CNN [37]	2 Conv [104, 8640] + 2 Hidden Layers	05e6	32x32
H-CNN (our method)	2 Conv [20, 80] + 1 Hidden Layer [400]	0.75e6	28x28

Again, If there is any point we want to stress on, it would be reducing complexity. That was our goal from the beginning. Arguably, we reached a good recognition performance at a much lower cost. The following are the factors of comparison:

- ∗ The size of our network is much smaller.
- ∗ The number of parameters used in the network is much smaller.
- ∗ The number of convolutonal and subsampling layers.
- ∗ The number of hidden layers in the MLP classifier.

* The size of the input image.

To compare the performance of our approach against the top-scoring methods in GT-SRB, we submitted the recognition result of our method to the GTSRB competition website: Luckily, the submission is still open for benchmarking and research. We report the result. Table

Table 4.2: Comparison of the overall performance (accuracy rates) between our method and the top-scoring methods in GTSRB competition.

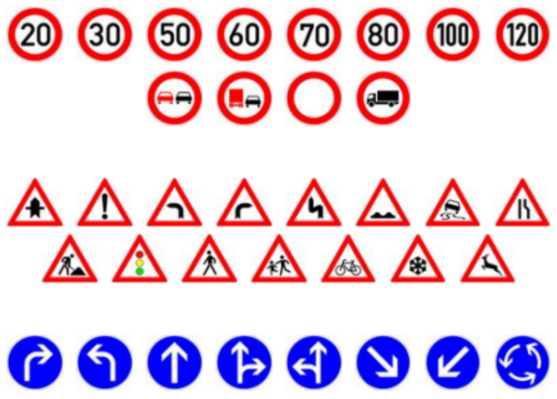
No.	Method	Score
1	2-stage HOG+SVM [49]	99.52%
2	Multi-Column CNN [8]	99.46%
3	**Benchmark - Human Performance**	98.84%
4	Multi-Scale CNN [37]	98.31%
5	Random Forests [50]	96.14%
6	**H-CNN (our method)**	94.89%

4.2 Traffic Sign Detection

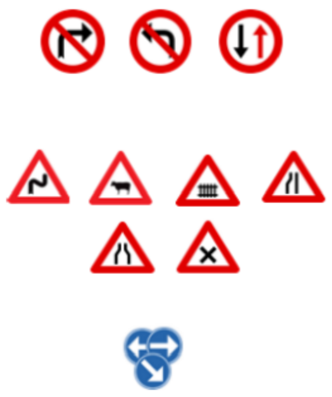
In chapter

To further test this hypothesis, we did the following:

* Split the GTSRB to the 3 main super-classes: prohibitory, mandatory and warning, as shown in figure



(a) Training images from GTSRB.



(b) Test images from BelgiumTS.

Figure 4.1: Training a detection CNN with GTSRB dataset (left), then testing our hypothesis: will it work as a shape detector? To test this hypothesis, we use test images from never-seen-before classes. These images are taken from BelgiumTS dataset (right).

The reason we take images from the BelgiumTS, is that we wanted to test the detection CNN using images from never-seen-before classes, i.e. classes that were not statistically-represented in the training set. We wanted to test our hypothesis: will the detection CNN work as a shape detector? After we did this experiment, we got the following results, in table

Table 4.3: The detection rates of the 3 super-class detectors. The test is done using images from never-seen-before classes.

CNN Detector	**Detection Rates**
Prohibitory	67.4%
Mandatory	62.8%
Warning	72.6%

The conclusion is: we don't have sufficient evidence that the network is primarily and successfully working as a shape detector. The detection rates we achieved over the test images are not sufficient to make a correct conclusion.

Chapter 5

Conclusion

In chapter

5.1 Future Work

The following point are the future work that can be conducted based on the work done in this thesis. They can be considered as an extension to our work. While initially planned to be covered, some of these points were not realized during the time-frame of the project due to lack of experience and over-optimistic attitude.

5.1.1 Tracker

As we initially wanted to build a system to detect, recognise and locate traffic sign in run-time, an essential part of the system is the tracker. It's up to the tracker to check the correctness of the results after detection and recognition. Also, tracker is the component able to accurately calculate the position of the traffic sign as the car moves and hence, precisely locate it on the map. Researcher has conducted further investigation in this area [**?**].

5.1.2 Detection Using Deep Learning

This research area is novel. It has not been addressed yet by researchers. During our project, we did a limited attempt to apply deep learning to detect traffic signs.

My personal judgement is that there is a huge potential for deep leaning and CNNs if applied to this problem.

Appendix A

Implementation

Here, we mention some notes regarding the implementation of the project of this thesis. The project is implemented using Python. It was developed used PyCharm development environment. For deep learning, we used open-source libraries, listed in table

Table A.1: Important Python libraries used in the project.

Library	Function
Theano [3]	Deep Learning and ConvNets
Lasagne	Deep Learning and ConvNets
NoLearn	Deep Learning and ConvNets
OpenCV [4, 5, 6]	Computer Vision
Scikit-Image [43]	Computer Vision
PIL	Computer Vision

A.1 Source Code

The source code of the project was hosted on GitHub. You can find it here: `https://github.com/noureldien/TrafficSignRecognition`. It's completely open for experimenting and research. Please feel free to use it without any permession.

A.2 Dataset

The project must be used along with the dataset. It is hosted locally and is not available yet online. But a description will be added in the ReadMe file on the

GitHub project. It will contain all the details of how to get the data and how to use it in the project, out of the box.

Bibliography

[1] Traffic sign assist. `http://techcenter.mercedes-benz.com/en/traffic_sign_assist/detail.html`. Accessed: 26- Aug- 2015.

[2] Traffic sign detection - driver assistance system. `http://www.ford.co.uk/Cars/NewFocus/Driving-experience`. Accessed: 26- Aug- 2015.

[3] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012.

[4] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[5] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[6] G. Bradski. *Dr. Dobb's Journal of Software Tools*.

[7] Karla Brkic. An overview of traffic sign detection methods. *Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing Unska*, 3:10000, 2010.

[8] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

[9] Dan Cireşan and Jürgen Schmidhuber. Multi-column deep neural networks for offline handwritten chinese character classification. *arXiv preprint arXiv:1309.0261*, 2013.

[10] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew Y Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 440–445. IEEE, 2011.

[11] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[12] UK Government Department of Transport. *Know Your Traffic signs*. 6th edition, 2007.

[13] Tim Dettmers. Understanding convolution in deep learning. `http://timdettmers.com/2015/03/26/convolution-deep-learning/`, 2015. [Online; accessed 19-July-2015].

[14] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *BMVC*, volume 2, page 5, 2009.

[15] Jonathan Fabrizio, Beatriz Marcotegui, and Matthieu Cord. Text detection in street level images. *Pattern Analysis and Applications*, 16(4):519–533, 2013.

[16] Ivan Filkovic. Traffic sign localization and classification methods: An overview. *VISTA, Computer Vision Innovation for Safe Traffic*, 2014.

[17] DL4J Deep Learning for Java. Convolutional networks. `http://deeplearning4j.org/convolutionalnets.html`, 2015. [Online; accessed 19-July-2015].

[18] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[19] Gov.uk. Traffic-sign images - gov.uk. `https://www.gov.uk/traffic-sign-images`, 2015. [Online; accessed 19-July-2015].

[20] Jack Greenhalgh and Majid Mirmehdi. Real-time detection and recognition of road traffic signs. *Intelligent Transportation Systems, IEEE Transactions on*, 13(4):1498–1506, 2012.

[21] Jack Greenhalgh and Majid Mirmehdi. Traffic sign recognition using mser and random forests. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1935–1939. IEEE, 2012.

[22] Thananop Kobchaisawat and Thanarat H Chalidabhongse. Thai text localization in natural scene images using convolutional neural network. In *Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA)*, pages 1–7. IEEE, 2014.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[25] Béla Lipták. Artificial intelligence in process automation. `http://www.controlglobal.com/articles/2006/221/`, 2008. [Online; accessed 19-July-2015].

[26] Chunsheng Liu, Faliang Chang, and Zhenxue Chen. Rapid multiclass traffic sign detection in high-resolution images. *Intelligent Transportation Systems, IEEE Transactions on*, 15(6):2394–2403, 2014.

[27] Chunsheng Liu, Faliang Chang, Zhenxue Chen, and Shuang Li. Rapid traffic sign detection and classification using categories-first-assigned treeâŃĘ. *J. Comput. Info. Syst*, 9(18):7461–7468, 2013.

[28] Mayeul Mathias, Radu Timofte, Rodrigo Benenson, and Luc Van Gool. Traffic sign recognitionâĂŤhow far are we from the solution? In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013.

[29] United Nations. Vienna convention on road traffic (1968). `https://treaties.un.org/Pages/ViewDetailsIII.aspx?src=TREATY&mtdsgno=XI-B-19&chapter=11&Temp=mtdsg3&lang=en`, 1968. Accessed: 15- Jul- 2015.

[30] OpenStreetMap.org. Road signs in belgium - openstreetmap wiki. `http://wiki.openstreetmap.org/wiki/Road_signs_in_Belgium`, 2015. [Online; accessed 19-July-2015].

[31] Gary Overett and Lars Petersson. Large scale sign detection using hog feature variants. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 326–331. IEEE, 2011.

[32] Peretti. Perceptron layer. `https://perceptronlayer.wordpress.com/`, 2015. [Online; accessed 19-July-2015].

[33] K. Zimmermann R. Timofte and L.V. Gool. Multi-view traffic sign detection, recognition, and 3d localisation. *Machine Vision and Applications*, 25(3):633–647, 2014.

[34] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

[35] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

[36] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[37] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2809–2813. IEEE, 2011.

[38] S. Shafer and W. Whittaker. Development of an integrated mobile robot system at cmu. 1989.

[39] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[40] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.

[41] Yichuan Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013.

[42] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient object localization using convolutional networks. *arXiv preprint arXiv:1411.4280*, 2014.

[43] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[44] Qian Wang, Jiaxing Zhang, Sen Song, and Zheng Zhang. Attentional neural network: Feature selection using cognitive feedback. In *Advances in Neural Information Processing Systems*, pages 2033–2041, 2014.

[45] Tao Wang, David J Wu, Andrew Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[46] Wikipedia. Comparison of european road signs. `https://en.wikipedia.org/wiki/Comparison_of_European_road_signs`, 2015. [Online; accessed 19-July-2015].

[47] Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, and Xiaolin Hu. Traffic sign detection based on convolutional neural networks. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7. IEEE, 2013.

[48] Zhicheng Yan, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Robinson Piramuthu. Hd-cnn: Hierarchical deep convolutional neural network for image classification. *arXiv preprint arXiv:1410.0736*, 2014.

[49] Yi Yang, Hengliang Luo, Huarong Xu, and Fuchao Wu. Towards real-time traffic sign detection and classification. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 87–92. IEEE, 2014.

[50] Fatin Zaklouta, Bogdan Stanciulescu, and Omar Hamdoun. Traffic sign classification using kd trees and random forests. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2151–2155. IEEE, 2011.

[51] Yujun Zeng, Xin Xu, Yuqiang Fang, and Kun Zhao. Traffic sign recognition using extreme learning classifier with deep convolutional features. In *The 2015 international conference on intelligence science and big data engineering (IScIDE 2015), Suzhou, China*, 2015.