VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Matúš Zakarovský

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DETEKCE DOPRAVNÍCH ZNAČEK POMOCÍ METOD STROJOVÉHO UČENÍ

TRAFFIC SIGNS RECOGNITION BY MEANS OF MACHINE LEARNING APPROACH

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE AUTHOR	Matúš Zakarovský
VEDOUCÍ PRÁCE SUPERVISOR	Ing. Karel Horák, Ph.D.

BRNO 2018

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Matúš Zakarovský ***ID:*** 186391
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Detekce dopravních značek pomocí metod strojového učení

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti detekce a rozpoznání svislých dopravních značek v obraze monokulární kamery z běžného silničního provozu za použití metod strojového učení. Body zadání:
1. Prostudujte metody detekce základních příkazových a zákazových dopravních značek v obrazu.
2. Vyberte vhodnou metodu založenou na konceptu strojového učení a implementujte ji pro zadanou množinu vstupníc h dat.
3. Ověřte funkčnost a stanovte úspěšnost klasifikace na reálných datech ze silničního provozu.
Téma zpracujte ve spolupráci s firmou Artin s.r.o. v rámci projektu Roboauto. Při implementaci použijte podle potřeb nástroje C++, OpenCV, Tensorflow, Keras, Caffe, ROS apod.

DOPORUČENÁ LITERATURA:

1. GONZALES, R.C., WOODS, R.R.: Digital image processing (3rd edition). Prentice-Hall, Inc., 2008. 954 pages. ISBN 978-0131687288.

2. SZELINSKI, R.: Computer Vision: Algorithms and Applications. Springer,2010. ISBN 978-1848829343.

3. Zapletal, O. Rozpoznávání obrazů konvolučními neuronovými sítěmi - základní koncepty. Brno 2017. Diplomová práce. VUT FEKT v Brně. Ústav automatizace a měřicí techniky. Vedoucí práce Karel HORÁK.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Karel Horák, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRACT

This thesis researches methods of traffic sign recognition using various approaches. Technique based on machine learning utilizing convolutional neural networks was selected for further implementation. Influence of number of convolutional layers on neural network's performance is studied. The resulting network is tested on German Traffic Sign Recognition Benchmark and author's dataset.

KEYWORDS

Traffic Sign, Traffic Sign Detection and Recognition, Machine Learning, Convolutional Neural Networks, CNN

ABSTRAKT

Táto práca skúma metódy rozpoznávania dopravných značiek. Implementovaný prístup využíval strojové učenie založené na konvolučných neurónových sieťach. V rámci tejto práce bola zistená závislosť úspešnosti neurónovej siete od počtu konvolučných vrstiev. Výsledná neurónová sieť bola testovaná na datasete GTSRB a na datasete vytvoreným autorom.

KĽÚČOVÉ SLOVÁ

Dopravná značka, Detekcia a rozpoznávanie dopravných značiek, Strojové učenie, Konvolučné neurónové siete, CNN

ZAKAROVSKÝ, Matúš. *Traffic Signs Recognition by Means of Machine Learning Approach*. Brno, 2018, 53 p. Bachelor's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Advised by Ing. Karel Horák, Ph.D.

ROZŠÍRENÝ ABSTRAKT

Systém rozpoznávania dopravných značiek je dnes možné nájsť vo väčšine moderných áut. Tento systém deteguje rýchlostné obmedzenia, zobrazuje ich vodičovi na displeji a robí náš život bezpečnejším a komfortnejším. Existuje veľa potenciálnych odvetví, kde môže byť táto technológia využívaná, ako napríklad, robotika, umelá inteligencia (AI) alebo vývoj autonómnych vozidiel.

Cieľom tejto práce bol návrh neurónovej siete, ktorá dokáže detegovať a klasifikovať príkazové a zákazové dopravné značky v obraze. V prvej časti bola vytvorená rešerš metód rozpoznávania. V nej sú prezentované rozdielne prístupy k riešeniu tejto úlohy pomocou geometrických vlastností značiek alebo pomocou strojového učenia. V tomto konkrétnom prípade bol zvolený koncept na základe strojového učenia, keďže je to jedna z najvýkonnejších a najvšestrannejších metód počítačového videnia.

Druhá kapitola je zameraná na základnú teóriu konvolučných neurónových sietí (CNN), ich vrstvy a ďalej popisuje fungovanie detekčnej siete Faster R-CNN.

Tretia kapitola popisuje implementáciu niekoľkých detekčných sietí v záujme určenia vplyvu počtu konvolučných vrstiev na úspešnosť neurónovej siete. Zvolená konfigurácia siete bola trénovaná a verifikovaná na datasete GTSRB. Táto sieť bola neskôr testovaná aj na datasete vytvoreným autorom.

V rámci rešerše boli spracované štyri rozdielne prístupy k rozpoznávaniu značiek. Prvý na základe geometrických vlastností hľadaných dopravných značiek a zvyšné metódy využívali strojové učenie, konkrétne dve použili podporné vektory (SVM) a posledná konvolučné neurónové siete (CNN). Na základe tejto rešerše bolo zvolené riešenie na báze konvolučných neurónových sietí, keďže táto metóda bola najrobustnejšia a dosahovala najlepšie výsledky.

Tento návrh bol vytvorený v prostredí MATLAB s využitím balíčkov pre neurónové siete a paralelné výpočty. Na trénovanie bol využitý dataset GTSRB, ktorý je tvorený 39 000 obrázkami, kde značka zaberá približne 60% plochy obrazu. Na jednej vzorke je vždy iba jedna značka a jej veľkosť je v rozmedzí 15x15 až 256x256 pixelov. Na testovanie bol opäť použitý dataset GTSRB a taktiež dataset, ktorý nafotil autor. Tento dataset bol vytvorený v Brne, značky boli orezané v štýle GTSRB, ale bolo na nich nechané viac pozadia. Značky väčšie ako 180x180 pixelov boli zmenšené na pätinu ich veľkosti. Obrázky v tomto datasete môžu byť rozmazané, zrnité a sú fotené v rozdielnych svetelných podmienkach. Všetky značky majú anotovanú iba ich triedu. Detektor bol založený na sieti typu Faster R-CNN, ktorý automaticky trénuje sieť potenciálnych návrhov výskytov hľadaného objektu (RPN), v tomto prípade značiek. Prvá navrhnutá CNN využívala architektúru AlexNet s upravenými hodnotami parametrov a zredukovaným počtom max pooling vrstiev. Výsledky rozpoznávania boli veľmi zlé, preto bol tento návrh zavrhnutý. Namiesto

toho bola trénovaná sieť iba s dvoma konvolučnými vrstvami a tromi plne prepojenými vrstvami, ktorá dosahovala lepšie výsledky ako upravený AlexNet, ktoré ale stále neboli postačujúce. Preto boli navrhnuté ďalšie neurónové siete s inkrementujúcim sa počtom konvolučných vrstiev. Maximálne bolo použitých päť vrstiev. Tieto detektory boli porovnané na základe F_1 kritéria na testovacej množine z GTSRB. V testoch mala najvyššie skóre sieť so štyrmi konvolučnými vrstvami s hodnotou 0,823. Táto konfigurácia siete bola preto zvolená na tréning s upravenou množinou dát z GTSRB. Počet klasifikačných tried v tomto datasete bol znížený na 29, vzorky zo zanedbaných tried boli presunuté do pozadia. Kvôli zbytočne veľkému počtu obrázkov v pozadí bolo vyradených približne 8000 vzoriek. Celkový počet dát v trénovacej množine bol cca 32 000 zákazových, príkazových značiek a značiek upravujúcich prednosť mimo značky P 1 (Križovatka s vedľajšou pozemnou komunikáciou). Trénovanie tejto siete zabralo 17 hodín na jednej grafickej karte EVGA GeForce GTX 1080 SC. Následne bola táto sieť verifikovaná na testovacej množine GTSRB. Čas detekcie značky v obraze je 50 ms a detekčná sieť je schopná rozpoznať čiastočne prekryté dopravné značky a je invariantná k zmenám osvetlenia.

Tento Faster R-CNN detektor mal presnosť (precision) 86,76%, odvolanie (recall) 98,47% a hodnotu F_1 kritéria 0,922. Následne bol otestovaný aj na datasete, ktorý vytvoril autor, kde dosiahol presnosť 30,87%, odvolanie 95,12% a hodnota F_1 kritéria 0,466. Tento rozdiel v presnosti je spôsobený tým, že podiel plochy značky k pozadiu je v tomto datasete menší ako pri GTSRB a detekčná sieť je na to citlivá. Pri vzorkách väčších ako 200x200 pixelov je generované veľa chýb typu I (false positives), pretože RPN sieť nedokáže vytvoriť dostatočne veľký bounding box a preto hľadá menšie oblasti, v ktorých sa môže vyskytovať značka. Do budúcnosti by som tento návrh vylepšil tak, že by som sieť trénoval na dátach z reálnej premávky, čím by tento detektor stal robustnejším.

ACKNOWLEDGEMENT

I would like to thank my supervisor, mister Ing. Karel Horák, Ph.D., for professional guidance, consultations, patience and contributing ideas to my thesis.

Brno . . . .

author's signature

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

ACKNOWLEDGEMENT

Research described in this Bachelor's Thesis has been implemented in the laboratories supported by the SIX project; reg. no. CZ.1.05/2.1.00/03.0072, operational program Výzkum a vývoj pro inovace.

Brno . . . .
author's signature

MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

Contents

List of Figures

List of Tables

Listings

Introduction

Traffic sign recognition can be found in most of modern cars. It detects speed limit signs and displays them for the driver to see, thus making our lives safer and more comfortable. There are many potential fields where this technology can be further utilized, such as robotics, AI and development of autonomous vehicles.

The goal of this thesis was to design a network which can detect and classify mandatory and prohibitory traffic signs in image. In the first part, a research on recognition methods is concluded. It presents various approaches to the problem using geometrical properties or machine learning. In this particular approach, ML concept for traffic sign recognition is applied, as it is one of the most powerful and most versatile method of computer vision.

The second chapter focuses on basic theory of convolutional neural networks, their layers and provides description of Faster R-CNN detection network operation.

The third chapter follows the implementation of multiple detection networks to determine the influence of number of convolutional layers in CNN on their performance. Selected network configuration is trained and verified on German Traffic Sign Recognition Benchmark. The same network was later tested on dataset created by the author.

1 Research

This chapter describes a research I did for semestral thesis where I studied methods of traffic sign recognition. I selected four methods which I found interesting, because of their differences in approach to the problem. The first method does not utilize machine learning, but it prepares weights for neural network from color segmentation. The other three methods use different types of machine learning methods to achieve traffic sign recognition. These methods are described below.

1.1 Method Hassan Shojania

This method, based on [1], heavily relies on color information and geometrical properties of sign's shape to distinguish traffic signs. The whole algorithm used consists of four stages shown on 1.1.

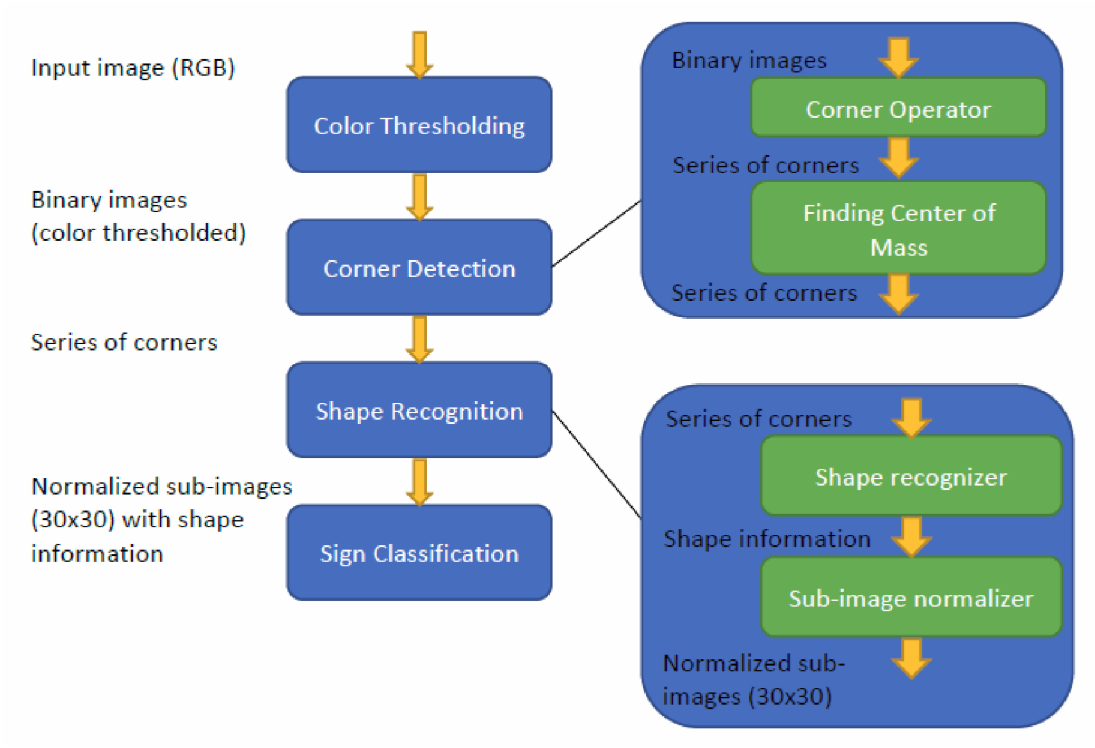


Fig. 1.1: Block diagram of the algorithm [3]

1.1.1 Color segmentation

As the signs of interest have certain characteristic color and its position, (e. g. red circle), it is possible to color threshold the RGB color spaced input image resulting in a binary image. This process saves computational time in the following steps of the algorithm. The RGB color space is very sensitive to lightning changes. Due to different weather conditions, the color range defined in equation would be unusable. There are two possible solutions to this problem. One of them is image conversion to HSI color space, which is computationally expensive, or use ratio of pixel's red component to its overall intensity to define range of red. It is possible to use one range for red component and other ranges for ratio of blue and green components.

$$g(x,y) = \begin{cases} k1, \quad \text{if} \begin{cases} R_{min} \leq f_r(x,y) \leq R_{max} \\ G_{min} \leq f_g(x,y) \leq G_{max} \\ B_{min} \leq f_b(x,y) \leq B_{max} \end{cases} \\ k2, \quad otherwise \end{cases} \tag{1.1}$$

where $g(x,y)$ is the thresholding output, $k1$ and $k2$ are the binary output values and $f(x,y)$ are the color components of input image.

1.1.2 Corner detection

Canny edge detector [2] is used on the binary image, which resulted from color segmentation. Its first step consists of convoluting the image with masks representing first derivative of Gaussian. Similar masks can be derived for other shapes. Common corner detectors use edge information or eigenvectors of gradient in pixel neighborhood. Optimal corner detector is suitable for real-time detection, because it works faster as aforementioned detectors. It convolves the image against defined masks and can classify type of detected corner by angle and direction, reducing complexity of shape recognition. Optimal corner detector models local grey level around a corner and attempts to find the optimal function, a mask, which when convolved with the image, yields a maximum at corner point. Noise is represented by additive Gaussian noise and the corner has fixed angle and orientation.

A good corner detector should satisfy following qualitative objectives:

- Good detection
- Good localization
- Single response to an edge
- It should not delocalize corner
- Detected corner should be an edge point

- The corner point should have at least two neighbors with different gradients than the corner itself

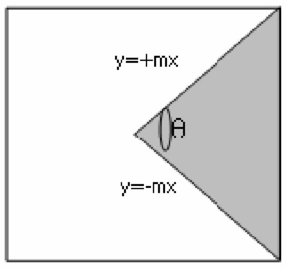


Fig. 1.2: Corner model [3]

A large number of masks is needed to detect all corners in the image. It is possible to reduce the number of masks and make the process computationally less expensive using a class of detectors approximating most of the possible corners. Corners are divided to groups based on which quadrants they occupy, and each group is assigned one mask. The total number of masks is 12, which is a lot less than having a mask for every corner possible. These masks work well, even though they do not have the same high response as masks specially tailored to certain corner.

Yield sign masks

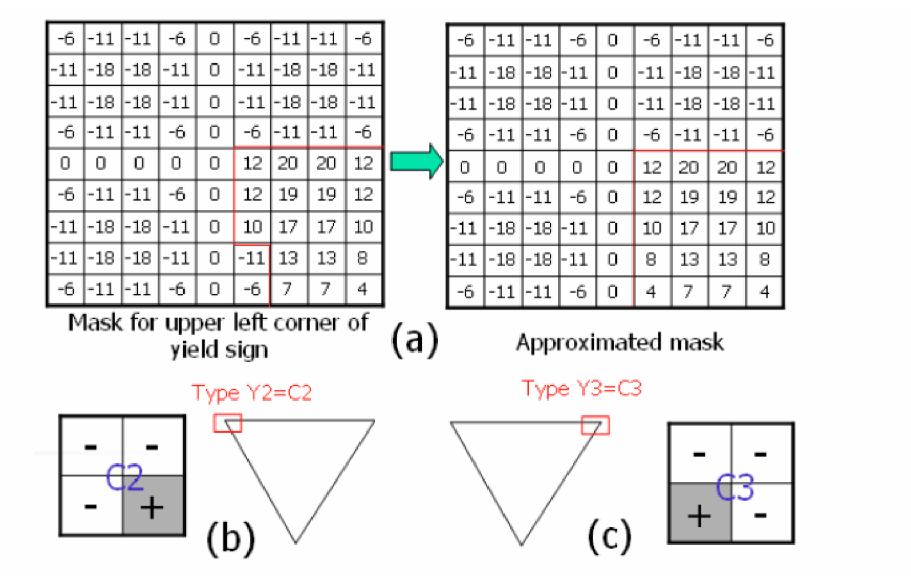
Yield sign forms an equilateral triangle. For the bottom corner, a Y1 mask (60° angle mask) was applied. Upper corners were approximated by C2 and C3 masks (90° angle mask), which are very similar to Y2 and Y3 masks and are needed for detection of other shapes as well. This process is shown on figures 1.3 and 1.4.

Stop and circular sign masks

Stop sign is an octagonal traffic sign. Its corner points p1 and p8 are detected via 60° angle mask and lines L1 through L4 are detected by 90° angle masks. It is not necessary to use 135° masks, because 90° masks approximate the corner well. Circular signs are detected using four 90° corner masks. This process is shown on figure 1.5.

Center of Mass (CoM) calculation

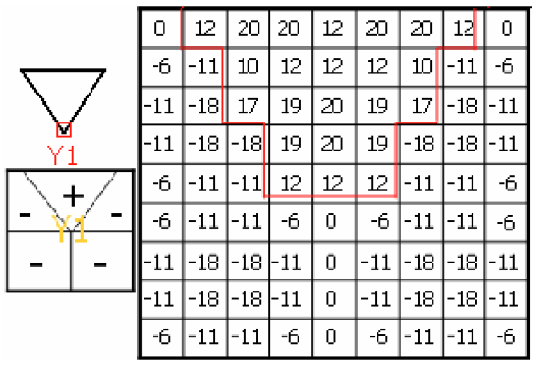
Single corner response is very rare because of errors encountered from multiple issues such as imperfect corner in input image, corner detector not being perfect, noise, approximating with a corner mask close to corner angle, etc. Therefore multiple corners within corner area are detected. Center of mass is calculated for these corners



-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6
0	0	0	0	0	12	20	20	12
-6	-11	-11	-6	0	12	19	19	12
-11	-18	-18	-11	0	10	17	17	10
-11	-18	-18	-11	0	-11	13	13	8
-6	-11	-11	-6	0	-6	7	7	4

-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6
0	0	0	0	0	12	20	20	12
-6	-11	-11	-6	0	12	19	19	12
-11	-18	-18	-11	0	10	17	17	10
-11	-18	-18	-11	0	8	13	13	8
-6	-11	-11	-6	0	4	7	7	4

Fig. 1.3: Yield sign upper corners: (a) approximating 60° corner with 90°; (b) 90° mask for upper-left corner; (c) 90° mask for upper-right corner [3]



0	12	20	20	12	20	20	12	0
-6	-11	10	12	12	12	10	-11	-6
-11	-18	17	19	20	19	17	-18	-11
-11	-18	-18	19	20	19	-18	-18	-11
-6	-11	-11	12	12	12	-11	-11	-6
-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6

Fig. 1.4: Corner detector for bottom of the yield sign [3]

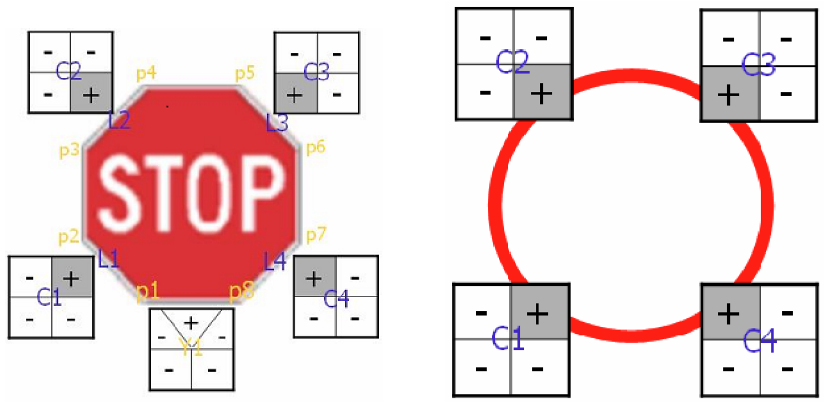
resulting in fewer corners and their location closer to the real corners. For each corner point, all points within its neighborhood are considered.Using convolution output as their weight, average point is calculated which represents center of mass. Equation for center of mass is shown below.

$$x_{cm} = \frac{\sum x_i}{n} \qquad y_{cm} = \frac{\sum y_i}{n} \tag{1.2}$$

where x and y are x, y corner coordinates and n represents number of corners.

1.1.3 Shape recognition

Recognition method used is based on geometry between detected corners and is similar to Interpretation Tree, where many subtrees are eliminated because of geometrical constraints. Number of corners in each class is also reduced due to classification



(a) (b)

Fig. 1.5: (a) Corner detectors for stop sign; (b) Circular sign corner detectors [3]

of corners (Y1, C2, C3) and because of picking order of the corners, the number of combinations is greatly reduced.

Yield sign recognition

Recognition process is very similar to approach in [1], but with added verification. The algorithm consists of picking a point *p1* from Y1 corner set. Finding points *p2* and *p3* from C2 and C3 corner set which satisfy following conditions:

- $(-60 - \theta_t) \leq \angle p3p1 \leq (-60 + \theta_t)$
- $(-120 - \theta_t) \leq \angle p2p1 \leq (-120 + \theta_t)$
- $l_{min} \leq \|p3p1\| \leq l_{max}$
- $l_{min} \leq \|p2p1\| \leq l_{max}$

where θ_t, l_{min} and l_{max} are configurable parameters.

Stop sign recognition

A pair of corners from Y1 corner set is picked. These corners are verified, if they roughly form a horizontal line, (configurable parameter), and a distance d is calculated between them. Then a corner like *p4*, which is set above *p1*, is found. This point has to be within certain angle range and its distance to *p1* must be $(1 + \sqrt{2}d)$. A mirrored process is done for corner *p5*. Next, a corner *p3* from C2 is picked, that satisfies constraints from p4. The process is repeated for *p6* comparing it with *p5*. Corners *p2* and *p7* that satisfy constraints from p1 and p8 respectively. Lines *p3p2* and *p6p7* should be almost vertical.

Circular sign recognition

Two corners *p1* and *p4* are picked and it is verified, if they roughly form a horizontal line, (configurable parameter), and a distance is calculated between them. Using computed distance *p1p4*, a search range in vertical direction,(configurable parameter), is set up to find *p2* above *p1*. The last step is repeated for corner *p3* above *p4*. To verify it is a circle, Y1 corner set is used to detect points in the bottom of the circle. There should be a few points. This verification is simple and inefficient as there are frequent false positives. Very similar method can be used to detect rectangles.

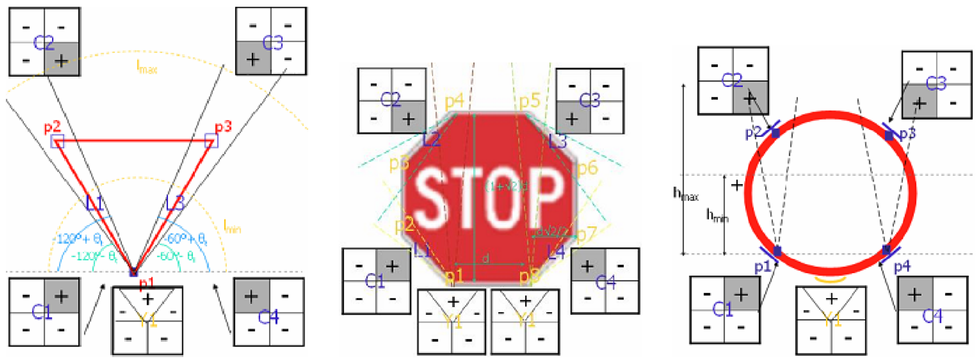


Fig. 1.6: (a) Yield sign recognition; (b) Stop sign recognition; (c) Circular sign recognition [3]

1.1.4 Results of the method

Figure 1.7 shows how this method fails to detect signs and generates a lot of false positives. This method is not very robust method and invariant to lightning changes.

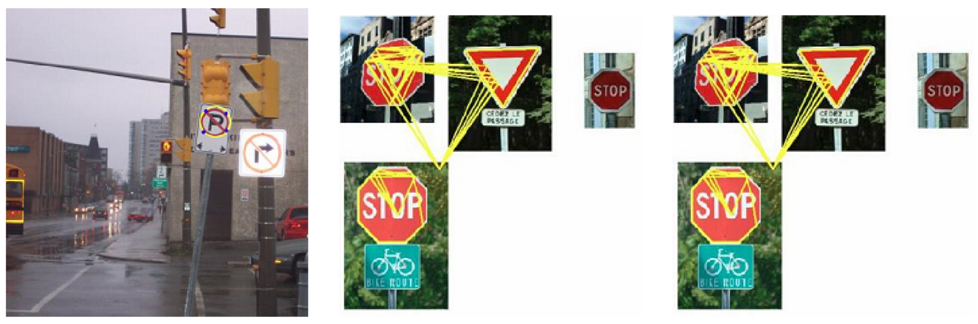


Fig. 1.7: (a) Classification; (b) Detection; (c) Result of wrong color threshold ratio [3]

1.2 SVM for Traffic Signs Recognition

The study [4] analyzes use of Support Vector Machines (SVM) with different feature representations, different kernels and SVM types, to identify and classify traffic signs. System is based on Swedish traffic signs and does not recognize all traffic signs, for instance rectangular informative signs. This study focused on recognition of seven traffic sign categories and five speed limit signs, shown on figure 1.8. The dataset consists of traffic signs in various conditions, e. g., damaged, faded or occluded by snow, etc. Due to sign placement and camera location, traffic sign dimensions can appear distorted. To solve this problem Zernike moments representation was implemented alongside direct binary representation.

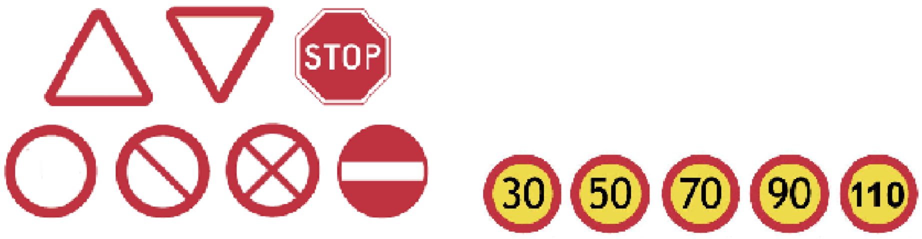


Fig. 1.8: Traffic and speed limits signs used [4]

Recognition system used is represented by figure 1.10. It consists of four main blocks. In the first one, possible traffic signs are separated from the background using color information. In the extraction process, signs are extracted from the image, their dimension normalized and saved as binary files. The following block represents features of every binary image, serving as an input to recognition step.

Traffic signs were extracted using a shadow and highlight invariant color algorithm, which has shown high robustness in various lightning conditions Result of the extraction can be seen on figure 1.9.

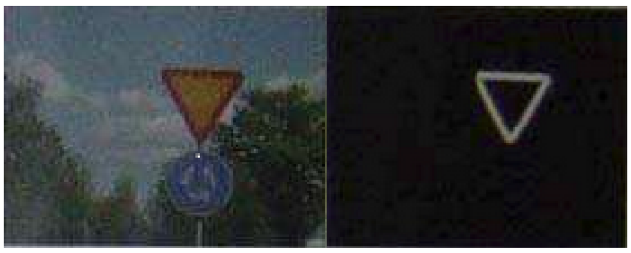


Fig. 1.9: Sign extraction example [4]

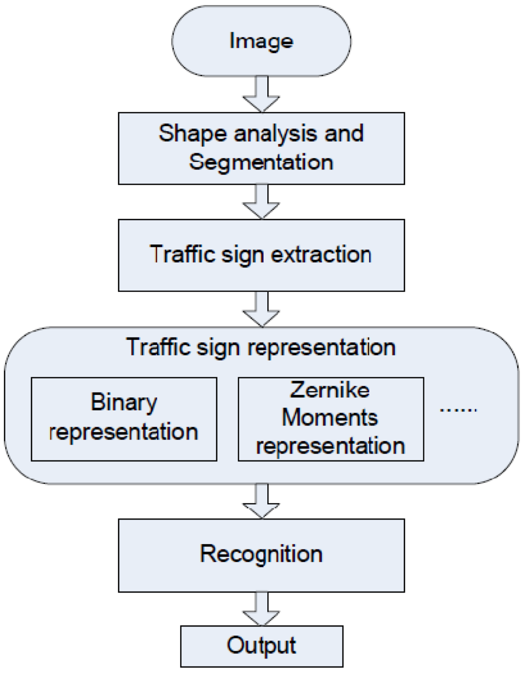


Image
Shape analysis and Segmentation
Traffic sign extraction
Traffic sign representation
Binary representation
Zernike Moments representation
......
Recognition
Output

Fig. 1.10: Block diagram of the system [4]

1.2.1 Feature representation

Each traffic sign is represented by an N dimensional feature vector. Two different feature selection methods were tried in this study, direct binary representation and Zernike moments.

Direct Binary Representation

Use of color segmentation described in [1] and [3], outputs a binary image where black is represented by 0 and white by 1. Resulting image size is 36x36 pixels resulting in 1296 attributes for one vector.

Zernike Moments

Image moment is a certain weighted average of pixel's intensity or its function, which has some attractive property. Moments extract a feature set. A collection of these moments can be computed to capture global features of an image and used as a feature vector for pattern recognition. Depending on type of pattern various moments can be used. The authors of this study decided to use Zernike moments to represent traffic signs. Zernike moments are made up by a sequence of polynomials forming complete orthogonal set over the interior of the unit circle. These specific moments were chosen because they are invariant to rotation and have high noise robustness. To determine Zernike moments of an image, a minimal circle containing traffic sign had to be defined. Then pixel coordinates of the sign were mapped inside the circle, which was regarded as a unit circle. Zernike moments are calculated from generated coordinates.

Two-dimensional Zernike moments of order p with repetition q of an discrete image, where $f(x, y)$ represents current pixel, are defined as,

$$Z_{pg} = \frac{p+1}{\pi} \sum_x \sum_y f(x,y) V^*_{pq}(x,y) \tag{1.3}$$

where $x^2 + y^2 \leq 1$, $V^*_{pq}(x, y)$ are the Zernike polynomials in a unit circle defined as,

$$V_{pq}(x,y) = R_{pq}(r_{xy}) e^{jq\Theta_{xy}} \tag{1.4}$$

where $r_{xy} = \sqrt{x^2 + y^2}$, $\Theta_{xy} = \arctan \frac{y}{x}$ and the real valued radial polynomial $R_{pq}(r)$ is,

$$R_{pq}(r) = \sum_{k=0}^{(p-|q|/2)} (-1)^k \frac{(p-k)!}{k!(\frac{p+|q|}{2})!(\frac{p-|q|}{2} - k)!} \tag{1.5}$$

where $0 \leq |q| \leq p$ and $p - |q|$ is even.

1.2.2 SVM classification

Signs in this study are recognized by SVM based on their shape only. Color properties do not play any role in the recognition process. Support Vector Machines are machine learning methods based on statistical learning proposed by Vapnik [5]. They use hyperplane of linear functions in high or infinite dimensional feature space. Pattern is found by building decision boundaries that optimally separate data in the hypothesis space. The basic SVM training principle analyzes and searches for optimal hyperplane that has a good generalization. The function describing hyperplane is as follows.

$$f(x) = \sum_{i=1}^{l} a_j y_j K(x_i, x) + b \tag{1.6}$$

where x is the input vector, l is the number of training samples and K represents kernel. Kernel is a function which implicitly maps input space to feature space. In this study four different kernels were used:

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- Radial basis function (RBF): $K(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

where γ,r and d are kernel parameters.

Two different types of SVM classification were used because the optimal hyperplane was not able to separate input vectors completely. The first type was C-support vector classification (C-SVC), which for given input vector $x_i \in R^n$ solves the primal problem $y_i \in -1, 1$ following the next steps.

$$\begin{aligned} &Minimize_{\xi,w,b} && \frac{1}{2}\langle w^T w\rangle + C\sum_{i=1}^{l}\xi_i \\ &Subject\ to && y_i(w^T\Phi(x_i)+b)+\xi_i \geq 1 \\ &\quad \xi_i \geq 0,\ i=1,\ldots,l && \end{aligned} \tag{1.7}$$

where w is the optimal hyperplane, slack variable ξ_i allows some misclassification and C is an a priori constant giving trade-off between maximum margin and classification error.

The other type of classification used in this study is ν-SVC. Parameter ν controls the number of support vectors and errors. It solves the primal problem according to the following algorithm.

$$\begin{aligned} &Minimize_{\xi,w,b} && \frac{1}{2}\langle w^T w\rangle - \nu\rho + \frac{1}{l}\sum_{i=1}^{l}\xi_i \\ &Subject\ to && y_i(w^T\Phi(x_i)+b)+\xi_i \geq \rho \\ &\quad \xi_i \geq 0,\ i=1,\ldots,l,\ \rho \geq 0 && \end{aligned} \tag{1.8}$$

where the parameter $\nu \in (0, 1]$ is an upper bound on the fraction of training errors and a lower bound of the fraction support vectors.

SVM is a binary classification algorithm. Various strategies as one-against-one, one-against-all, all-together and DAGSVM have been proposed to utilize it in a multiclass problem. One-against-one approach was used for traffic sign classification in this study. It constructs $k(k-1)/2$ binary classifiers, where k represents number of classes. Each classifier trains data from two different classes and then takes votes on the data. Data is assigned to the class with the most votes and if two classes get the same votes, data is assigned to class with smaller index. SVM overcome the curse of dimension in computation and generalization, therefore, they are able to perform pattern recognition without preprocessing.

1.2.3 Results of the method

The dataset consists of 600 samples, divided into 12 classes (7 sign shapes categories and 5 speed limit sign types), each category had 50 samples randomly distributed between 30 training and 20 testing samples. Experiments were divided into two parts. The first part trained and tested SVM with different feature representation.

The second part focused on use of various kernel and SVM types. Zernike moments could detect six types of sign shapes, since they are invariant to rotation and could not distinguish between upward and downward facing triangle. SVM models were trained on the same training dataset with for aforementioned kernels and two SVM types. Parameters of the SVMs were: $C = 1, \nu = 0.5, \gamma = 1/n, r = 0, d = 3$, where n is the number of input vector attributes.

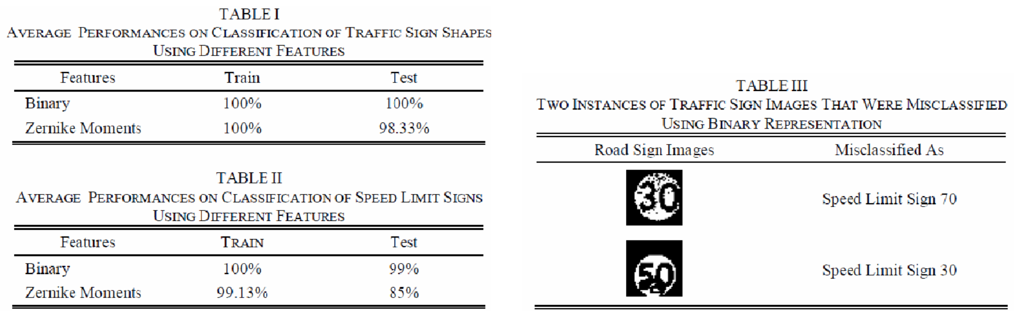


TABLE I
AVERAGE PERFORMANCES ON CLASSIFICATION OF TRAFFIC SIGN SHAPES USING DIFFERENT FEATURES

Features	Train	Test
Binary	100%	100%
Zernike Moments	100%	98.33%

TABLE II
AVERAGE PERFORMANCES ON CLASSIFICATION OF SPEED LIMIT SIGNS USING DIFFERENT FEATURES

Features	TRAIN	Test
Binary	100%	99%
Zernike Moments	99.13%	85%

TABLE III
TWO INSTANCES OF TRAFFIC SIGN IMAGES THAT WERE MISCLASSIFIED USING BINARY REPRESENTATION

Road Sign Images	Misclassified As
30	Speed Limit Sign 70
50	Speed Limit Sign 30

Fig. 1.11: Comparison between binary representation and Zernike moments [4]

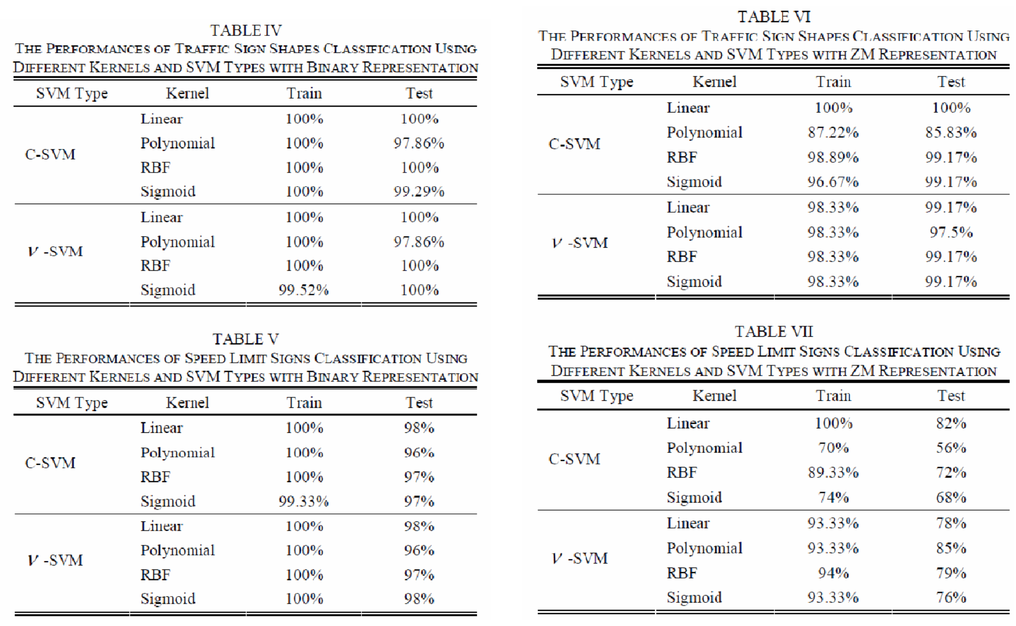


TABLE IV
THE PERFORMANCES OF TRAFFIC SIGN SHAPES CLASSIFICATION USING DIFFERENT KERNELS AND SVM TYPES WITH BINARY REPRESENTATION

SVM Type	Kernel	Train	Test
C-SVM	Linear	100%	100%
	Polynomial	100%	97.86%
	RBF	100%	100%
	Sigmoid	100%	99.29%
ν -SVM	Linear	100%	100%
	Polynomial	100%	97.86%
	RBF	100%	100%
	Sigmoid	99.52%	100%

TABLE V
THE PERFORMANCES OF SPEED LIMIT SIGNS CLASSIFICATION USING DIFFERENT KERNELS AND SVM TYPES WITH BINARY REPRESENTATION

SVM Type	Kernel	Train	Test
C-SVM	Linear	100%	98%
	Polynomial	100%	96%
	RBF	100%	97%
	Sigmoid	99.33%	97%
ν -SVM	Linear	100%	98%
	Polynomial	100%	96%
	RBF	100%	97%
	Sigmoid	100%	98%

(a)

TABLE VI
THE PERFORMANCES OF TRAFFIC SIGN SHAPES CLASSIFICATION USING DIFFERENT KERNELS AND SVM TYPES WITH ZM REPRESENTATION

SVM Type	Kernel	Train	Test
C-SVM	Linear	100%	100%
	Polynomial	87.22%	85.83%
	RBF	98.89%	99.17%
	Sigmoid	96.67%	99.17%
ν -SVM	Linear	98.33%	99.17%
	Polynomial	98.33%	97.5%
	RBF	98.33%	99.17%
	Sigmoid	98.33%	99.17%

TABLE VII
THE PERFORMANCES OF SPEED LIMIT SIGNS CLASSIFICATION USING DIFFERENT KERNELS AND SVM TYPES WITH ZM REPRESENTATION

SVM Type	Kernel	Train	Test
C-SVM	Linear	100%	82%
	Polynomial	70%	56%
	RBF	89.33%	72%
	Sigmoid	74%	68%
ν -SVM	Linear	93.33%	78%
	Polynomial	93.33%	85%
	RBF	94%	79%
	Sigmoid	93.33%	76%

(b)

Fig. 1.12: Results of the method for: (a) Binary representation; (b) Zernike moments [4]

This recognition method is robust and despite beneficial properties in pattern recognition, binary representation outperformed Zernike moments. Four types of kernel were tested, showing that linear kernel worked the best with both tested SVM types. This method is able to recognize wide variety of traffic signs, which include noise, damage, translation or rotation.

1.3 Evolutionary Adaboost Detection and F-ECOC Classification

As I already mentioned, there are two main approaches to recognize a traffic sign, color-based and gray-scale based. Color-based sign recognition reduces false-positive results during recognition phase using color. Grayscale based recognition concentrates on geometrical shape of the object. At the time of study, more recent studies have shown that combination of both methods improves detection rates. Many objects can share color with traffic signs, thus creating false-alarm regions. It is possible to lower the number of these regions by filtering out areas which aspect ratio is either smaller or larger than a specified ratio or ratio range. Region is normalized to predefined size and a linear SVM is used to determine region in a possible shape. The gathered color and shape data is used as a coarse classification, and an SVM with Gaussian kernels performs the fine classification.

1.3.1 Detection

Detector in this method uses attentional cascade concept. Feature selection is done by boosting and the image is represented by integral image. This method solves restriction of the boosting-process computation on large feature sets. This evolutionary boosting drastically reduces training time and allows use of huge feature sets. Dissociated dipoles are used as a features. These can be calculated using an integral image. In object detection, a large number of negative regions is discarded, whereas only some regions represent looked up object. Attentional cascade architecture allows discarding non-object regions at low computational cost and interesting regions are deeply analyzed. One attentional cascade is made of a set of classifiers, where input to each classifier corresponds to regions classified as object-regions by previous stage. These regions classified by the last stage form output of the detector.

Dissociated dipoles

Based on a study by Lienhart and Maydt [6], accuracy of the detector increases with the number of available features. Dissociated dipoles or sticks, shown on figure

1.13, are more general type of features than Haar-like features, and are able to deal with larger feature set. They are made of a pair of rectangular regions name excitatory and inhibitory dipole. The mean value of all pixels in inhibitory dipole is subtracted from the mean value of all pixels in excitatory dipole. Integral image is used to calculate sum of pixels located inside the regions. This increases feature set enormously and makes it computationally unfeasible for classical approach. To solve this problem, this study defines an evolutionary Adaboost approach.

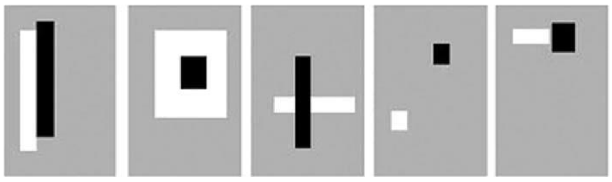


Fig. 1.13: Dissociated dipoles. Black represents inhibitory dipole and white the excitatory dipole [7]

Evolutionary Adaboost

Boosting is a learning technique that combines performance of many simple classification functions or Weak classifiers and produces a Strong classifier. The examples are re-weighted at each learning round to emphasize the examples incorrectly classified by previous Weak Classifier. The final Strong Classifier is a decision stump made of weighted combination of Weak Classifiers followed by a threshold. Classical boosting uses exhaustive search to determine Weak Classifier, making it computationally unfeasible for larger feature sets. Therefore, authors of this study defined evolutionary weak learner, which minimizes the weighted error function ε of the Adaboost scheme as follows, $\varepsilon = \sum_{i:h(x_i)\neq y_i} w_i$, where $X = \{(x_i, y_i)|i = 1 : m\}$ are the pairs of sample-label that compound the training set, $W = \{w_1, w_2, \ldots, w_m\}$ is the Adaboost weights distribution over the training set and $h(x_i)$ corresponds to the label that the hypothesis h predicted for training object x_i.

Weak learner can be considered as an optimization problem, where parameters of the Weak Classifier minimizing the error function are needed to be found. This function is full of discontinuities making classical approaches based on gradient descent unusable. The solution to this problem is use of evolutionary strategy using a genetic algorithm, which searches over the spaces of solutions using three basic concepts of Darwin's theory. Mutation, crossover, natural selection.

While working with evolutionary algorithm, two elements should be defined. The individual, representing a point in solution space and the evaluation function, which measures the function value at the point the individual represents. Weak

Classifier based on dissociated dipoles is defined as $h(I, Thr, x_i) \rightarrow \{-1, +1\}$, where $I = (Re_x, Re_y, Re_w, Re_h, Ri_x, Ri_y, Ri_w, Ri_h)$, with Re being the excitatory dipole, and the type parameter T is changed by the parameters of the inhibitory dipole Ri. To reduce evaluation time, regions are compared qualitatively, which eliminates need for illuminance normalization and threshold learning process.

Learning algorithm

The algorithm is used to learn all the stages of the detection cascade using a set $\langle (x1, y1) \dots (xm, ym) \rangle$ of samples positively classified by previous stage. It iteratively uses genetic algorithm to minimize weighted error and to instantiate the parameters of a new Weak classifier added to the final ensemble.

Algorithm 1: Evolutionary Discrete Adaboost.

```
Given: $(x_1, y_1), \dots, (x_m, y_m)$
    where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $W_1(i) = 1/m$
for t = 1,..., T do
    Use a genetic algorithm to minimize

        $$\epsilon_t = Pr_{i \sim W_t}[h_t(x_i) \neq y_i]$$

    The given solution is taken as the hypothesis $ht$.
    Get the weak hypothesis $h_t : X \mapsto \{-1, +1\}$ with error $h_t$.
    Choose $\alpha_t = (1/2) \ln (1 - \epsilon_t / \epsilon_t)$
    Update

        $$W_{t+1}(i) = \frac{Wt(i)}{Z_t} \times \begin{cases} e^{-\alpha_t}, & \text{if} \quad h_t(x_i) = y_i \\ e^{\alpha_t}, & \text{if} \quad h_t(x_i) \neq y_i \end{cases} = \frac{W_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

    where $Z_t$ is a normalization factor (chosen so that $W_{t+1}$ will be a distribution).
end for
Output the final hypothesis:

    $$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$
```

1.3.2 Classification

Once the object was located, it needs to be classified. The authors of this study decided to use ECOC classification technique, which combines base classifiers to address the multiclass problem.

ECOC

This design is based on coding and decoding. Coding assigns a codeword to each of the N_c classes and decoding aims to assign class label to new test codeword. Codewords are arranged in a coding matrix $M \in \{-1, 1\}^{N_c \times n}$, where n is the length of code, which from point of coding represents n binary learning problems, (dichotomies), each corresponding to a column of ECOC matrix M. Each dichotomy defines sub-partition of classes coded by {+1, -1} according to their membership. Code obtained from decoding of output of binary classifiers is compared with the base codewords in matrix M. Data points is assigned to the closest codeword. Distances used for decoding are Hamming and Euclidean distances.

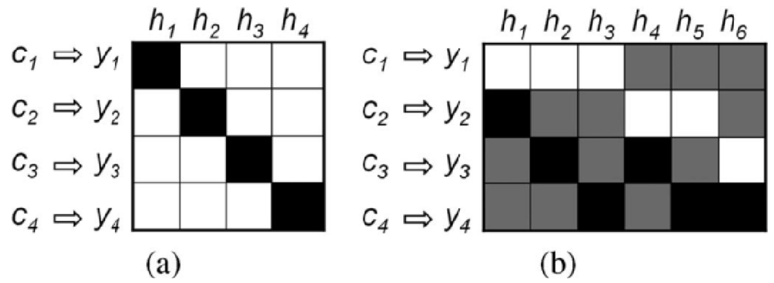


Fig. 1.14: Four-class ECOC designs: (a) One-vs-all; (b) One-vs-one [7]

F-ECOC

Most of discrete coding strategies are predesigned problem-independent codewords. Pujol et al. proposed a method for embedding a tree in the ECOC framework. Root contains all classes and nodes associated with the best mutual information partition is found. This process is repeated until sets with single class are obtained. Based on the mentioned approach, the authors embedded multiple trees forming an F-ECOC (Forest – Error Correcting Output Code). Optimal tree, with highest classification score at each node, and several sub-optimal trees, which are closer to the optimal tree under certain conditions.

Algorithm 2: Training algorithm for F-ECOC.

```
Given: $N_c$ classes: $c_1, \ldots, c_{N_c}$ and $T$ trees to be embedded
$\Omega_0 \leftarrow 0$
$i \leftarrow 1$
for t = 1,. . . , T do
    Initialize the tree root with the set $N_i = \{c_1, \ldots, c_{N_c}\}$
    Generate the best tree at iteration $t$:
    for each node $N_i$ do
        Train the best partition of its set of classes $\{P_1, P_2\} | N_i = P1 \cup P_2$,
        $N_i \notin \Omega_{t-1}$ using a classifier $h_i$ so that the training error is minimal
        According to the partition obtained at each node,
        codify each column of the matrix $M$ as:

        $$M(r,i) = \begin{cases} 0, & \text{if} \quad c_r \notin N_i \\ +1, & \text{if} \quad c_r \in P_1 \\ -1, & \text{if} \quad c_r \in P_2 \end{cases}$$

        where $r$ is the index of the corresponding class $c_r$
        $\Omega_t \leftarrow \Omega_{t-1} \cup N_i$
        $i \leftarrow i + 1$
    end for
end for
```

This technique provides sub-optimal solution due to robust classifier combination. The main advantage of this method is that trees share information between classes. It is done by joint decoding step and each tree of N_c classes introduces $N_c - 1$ classifiers. At the time of study, it has been shown that zero symbols introduce errors in decoding distances. Attenuated Euclidean decoding distance has been proposed as solution to this problem. It is defined as follows. $d_j = \sqrt{\sum_{i=1}^{n} |y_i^j|(x_i - y_i^j)^2}$, where d_j is distance to the row j, n is the number of dichotomies, x_i is the response of the classifier h_i over test sample and $|y_i^j|$ is the value of coding matrix M at the i-th row and j-th column. Factor $|y_i^j|$ was introduced to avoid error of the zero symbol.

1.3.3 Sign Recognition Process

All data acquired is given to the detector. Detectors form an attentional cascade, where at each stage almost all objects of interest are detected, while a certain fraction of non-sign pattern is rejected. Signs are grouped based on similarity and for each

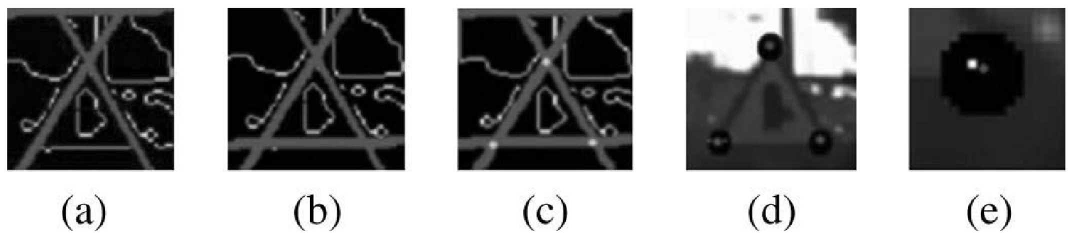


Fig. 1.15: Model Fitting. (a) Detected lines. (b) Corrected line. (c) Intersections. (d) Corner region. (e) Corner found [7]

group a different cascade was trained. Detector output consists of all detected objects by cascades.

Region of interest is determined via Evolutionary Adaboost, and based on the type of detected sign, a different model fitting is applied looking for affine transformations performing spatial normalization. Fast radial symmetry is applied for circular signs, which provides high robustness to noise and approximates the center and the radius of the sign. Method based on Hough transform is used for triangular signs. If a false line is detected, Hough procedure is iterated to detect next representative line. When three lines are detected their intersections are calculated and verified by corner detector.

Spatial normalization consists of four steps. Image transformation, to make recognition invariant to small affine deformations. Resize the object to the signs database size. Filter the image using Weickert anisotropic filter. Mask the image to exclude background pixels at the classification step. To make classification invariant to illumination conditions, histogram equalization is used to improve contrast and yield a uniform histogram.

Signs are classified via F-ECOC, where optimal trees consider the best subpartitions of the classes to obtain robust classifiers based on grey-level pixel values.

1.3.4 Results of the method

Genetic Algorithm with a population of 100 individuals, Gaussian based mutation probability and scattered crossover strategy with 0.8 fraction were used to perform tests with Evolutionary Adaboost. As shown on the figure 1.16, both methods converge with the same number of iterations, showing that even though Evolutionary Adaboost has a random component, it still performs well. It has been able to determine triangular sign with 90.87% probability and circular sign with 90.16% probability.

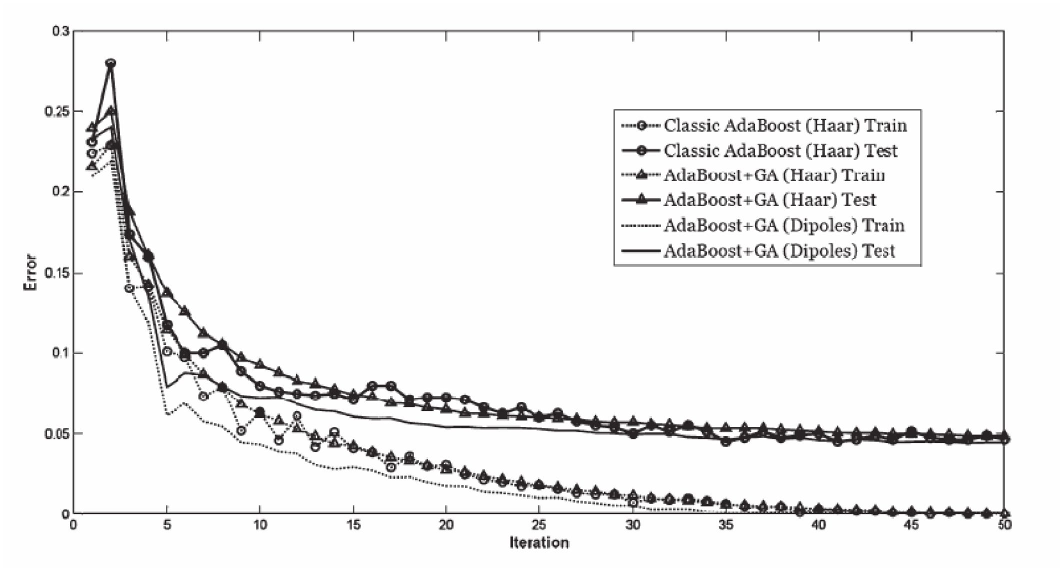


Fig. 1.16: Error comparison between different Adaboost approaches [7]

1.4 Tsinghua-Tencent 100K

This method was concluded in China and uses traffic signs in realistic world conditions with large variations in illumination, weather conditions and occlusion. The dataset consists of 100 000 high resolution images cut from Tencent Street Views (Chinese equivalent of Google Street View). Traffic signs take around 0.2% of an image. Traffic signs were classified into three categories: warning, prohibition and mandatory. Other traffic signs and signs resembling traffic signs were classified as "other".

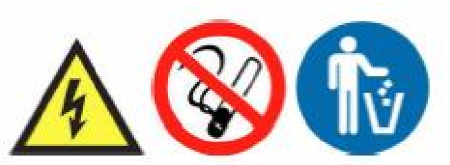


Fig. 1.17: Traffic-like signs [8]

1.4.1 Traffic sign classification

Before common adoption of CNN, detection methods based on SVMs or sparse representations were used. Sermanet et al. [9] observed that CNN are more efficient when used in a sliding window fashion because many computations can be reused in overlapping regions. Later on they demonstrated a CNN that can determine object's bounding box together with its label. There is also a different commonly used, but less efficient and slower method R-CNN, which as a first step calculates some generic object proposals and then performs classification only on these candidates. Usage of spatial pyramid pooling network (SPP-Net) improves efficiency of R-CNN and speeds it up about 100 times. This approach calculates convolutional feature map for the entire image and extracts feature vectors from the shared feature map for each proposal. Another improvement of R-CNN called Fast R-CNN, uses softmax layer above network and not a SVM classifier as R-CNN does. Region proposal networks (RPN) were unveiled later, which share full-image convolutional features with the detection network.

Based on knowledge that deep learning methods show superior performance in image classification and speech recognition, study [8] used CNN to identify traffic signs as they deem it the most suitable deep learning method for image classification, localization and detection.

1.4.2 Data collection and annotation

Because of lack of real-world traffic sign images and will to mimic real world scenario, authors decided to use Tencent Street Views as a source of images. In the time of study Tencent Street Views covered about 300 Chinese cities and roads connecting them. The dataset consisted of images with and without traffic signs to evaluate if detector is working properly. Images collected were annotated by hand. During annotation bounding box, boundary vertices and sign's class label were recorded. To determine sign's pixel mask two modes were used: polygon which marks polygon vertices, and ellipse which marks vertices alongside ellipse boundary. Vertices are used to automatically fit the shape of the traffic sign. Circular signs appear as ellipses unless occluded. Occluded signs are the most difficult thing to detect. In this case bounding box boundary, polygon boundary, and if appropriate, ellipse boundary is marked, and intersected.

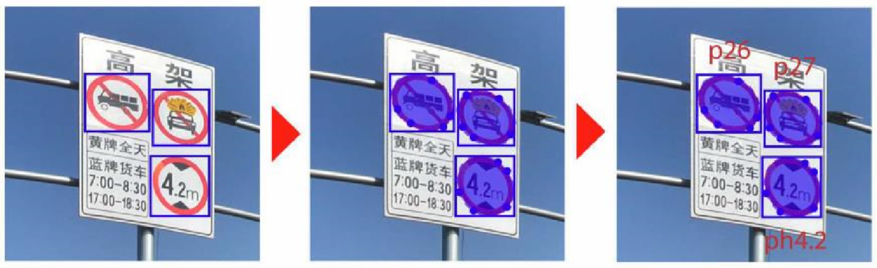


Fig. 1.18: Annotation pipeline [8]

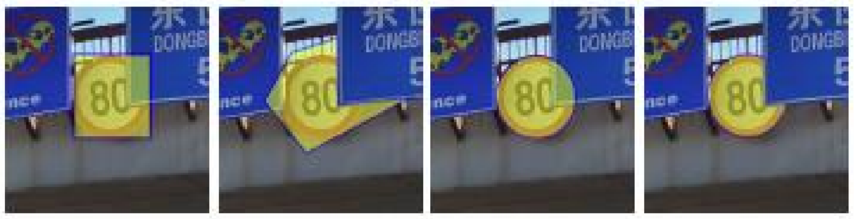


Fig. 1.19: Occluded sign annotation [8]

1.4.3 Neural networks

Two neural networks were trained, one for detection only and the other for simultaneous detection and classification. The networks have 8 layers and branch after layer 6. If the network was branched earlier, it would have a potential to perform better but it would need more training time and would consume more memory. The output layer is branched into three streams, bounding box layer, pixel layer and label layer; which allows traffic sign classification as well as traffic sign detection. As there were uneven numbers of examples of different classes, data augmentation technique was used during training. Classes with fewer than 100 instances were ignored. Classes with 100 – 1000 instances were augmented to have 1000 instances. Classes that had more than 1000 were left unchanged. To extend the dataset, the augmented classes were randomly rotated in [-20°, 20°] range, scaled in [20, 200] range, and random amount of noise was added. Afterwards some images with no traffic signs were manually added and distorted with random noise.

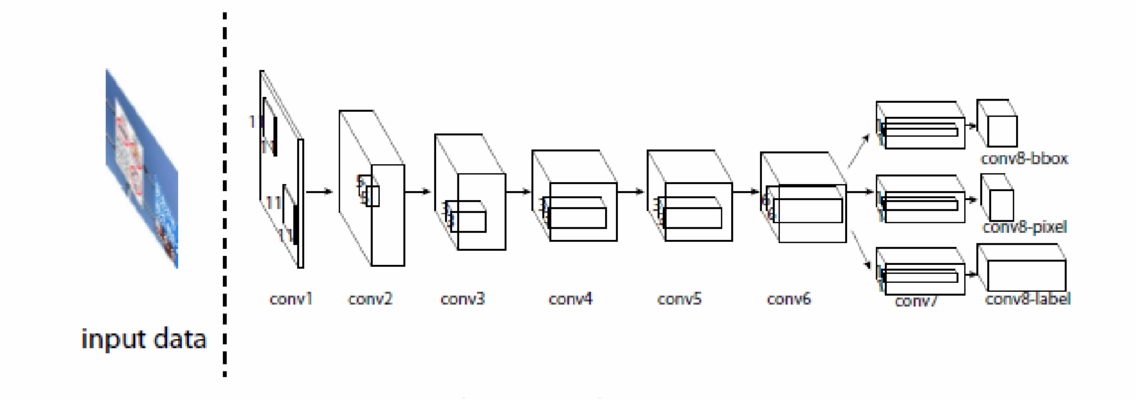


Fig. 1.20: Architecture of used CNN [8]

1.4.4 Results

Training and testing of the neural network was done on Linux based PC with Intel Xeon and two NVIDIA Tesla GPU. 10 000 panoramas were divided in to the training and testing set in 2:1 ratio. The other 90 000 panoramas were included in testing. Neural network used in traffic sign detection was compared with typical object proposal methods as Selective Search, Edge Boxes and BING. Selective Search and Edge Boxes do not need training data. BING was trained using the same data as methods neural network. Tsinghua-Tencent 100K detection network achieved 84% accuracy and 91% recall. This network outperforms previously mentioned object detection methods and was able to identify that, all of 90 000 panoramas without traffic sign are considered background. Simultaneous traffic sign detection and classification neural network was compared with Fast R-CNN. Overall Fast R-CNN has shown better performance for larger objects. It has a 0.56 recall and 0.50 accuracy compared to Tsinghua-Tencent 100K's 0.91 recall and 0.88 accuracy. This method is slow, tested on a powerful hardware and not suitable for real-time recognition.

Object size	[0,32]	[32,96]	[96,400]
Fast R-CNN recall	0.24	0.74	0.86
Fast R-CNN accuracy	0.45	0.51	0.55
T-T 100K recall	0.87	0.94	0.88
T-T 100K accuracy	0.82	0.91	0.91

Tab. 1.1: Results of the comparison between Fast R-CNN and T-T 100K [8]

2 Theory

Based on the research done in the previous chapter, the approach of Tsingua-Tencent 100K benchmark seemed to be the most robust recogniotion network, therefore, a solution solution using convolutional neural networks was implemented. In the following sections, basic theory of CNNs and Faster R-CNN is described.

2.1 Convolutional Neural Networks

Convolutional neural network is a type of neural network which use matrices as input instead of vectors. They are a very powerful tool commonly used in deep learning methods, in applications such as image and video recognition, natural speech processing and recommender systems. Architecture of CNN varies and is dependent on given problem. It consists of layers which can, but do not have to branch.

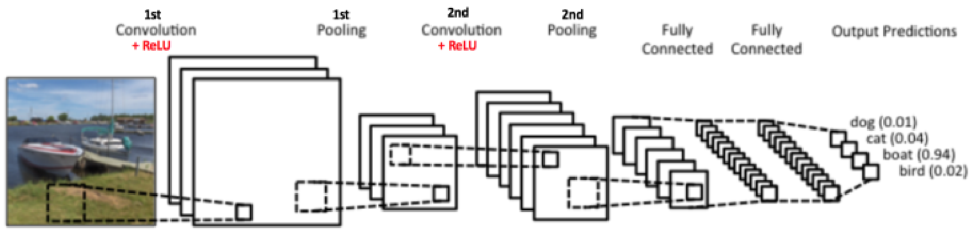


Fig. 2.1: CNN example [11]

2.1.1 Layers used in CNN

There are many types of layers which can be used when designing a neural network architecture. This subsection explains the basic layers used in design.

Convolutional layer

This layer performs convolution on its input with defined kernel. Input to this layer can be some data, such as image; or a feature map computed in previous convolutional layer. Kernel, also called filter, is a square matrix and there may be multiple filters in one layer. Because of the nature of convolution, output size is different than size of the input. To further change the output size or preserve it, parameters such as stride and zero padding are implemented. Stride defines a step the kernel matrix makes on the input matrix. The bigger the stride, the smaller the feature map gets. Default value for stride is 1. Zero padding puts zeros around the borders of the input matrix, which can be useful in some scenarios.

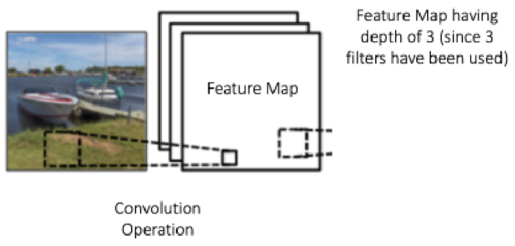

Feature Map having depth of 3 (since 3 filters have been used)
Feature Map
Convolution Operation

Fig. 2.2: Example of convolution [11]

Activation layer

There are many activation functions which present non-linearity to feature map, for example sigmoid or hyperbolic tangent. In CNN typically a Rectified Linear Unit (ReLU) is used or its versions, such as leaky ReLU.

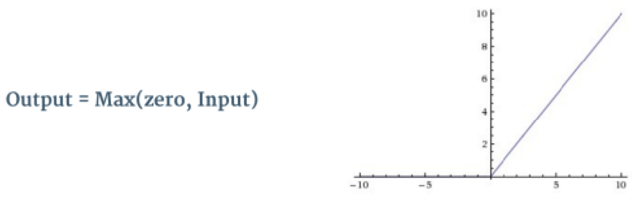


Fig. 2.3: ReLU example [11]

Pooling layer

The most used type of pooling layer is max pooling layer, which downsamples the feature map, while the most important information is preserved. Figure 2.4 illustrates operation of max pooling.

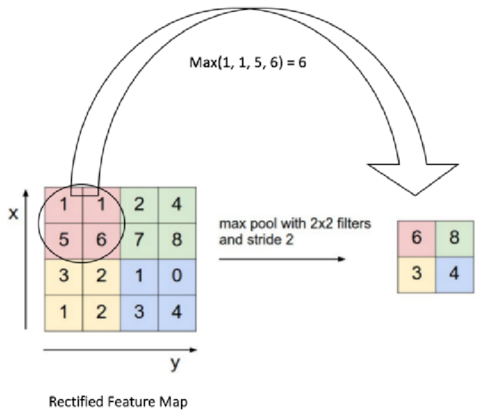


Fig. 2.4: Max pooling example [12]

Dropout layer

Dropout layer simply sets some features to 0 with given probability. This greatly prevents over-fitting of the network and makes it more redundant to error.

Fully connected layer

This layer represents Multi Layer Perceptron with softmax activation layer. As the name suggests, every neuron in one layer is connected to every neuron in the next layer.

2.2 Faster R-CNN

Faster R-CNN consists of two different networks. Region proposal network (RPN) and network for object detection. RPN creates anchors, boxes, which are ranked and only those with highest possibility to contain an object of interest are proposed. The output of RPN is passed to classifier and regressor which verify if object is present or not. This method is faster than its predecessors R-CNN and Fast R-CNN, because it shares most of the features with object detection network.

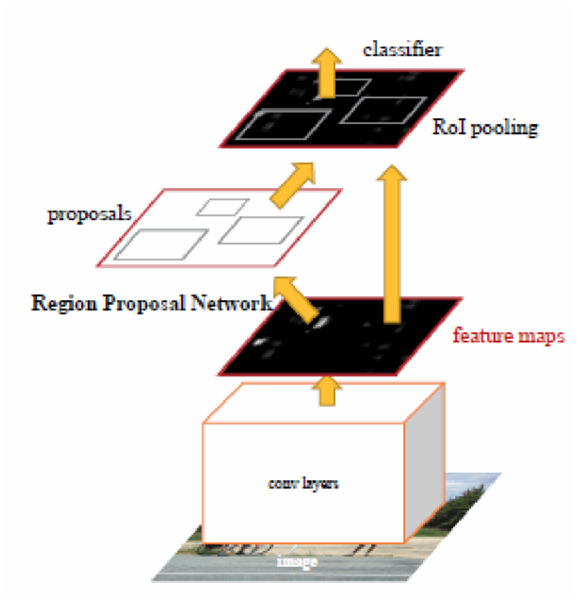


Fig. 2.5: Faster R-CNN architecture [14]

3 Experimental part

This chapter is divided into four sections. In the first one, datasets used and their properties are discussed, the second part describes the reasons for the chosen hardware and software configuration, section three follows the implementation of the network and determining the number of layers and parameters. The last section presents the obtained results from the network designed with the best performance.

3.1 Datasets

In the implementation two datasets have been used. One from German public source called GTSRB, and the other one was created by the author.

3.1.1 German Traffic Sign Recognition Benchmark (GTSRB)

This dataset is publicly available and free to use. It is made up from cca 50 000 images in RGB color space, which vary in size between 15x15 pixels and 256x256 pixels. The images are taken in various lightning and weather conditions. There are 43 sign classes and every class comes with file containing annotations in the following format:

```
filename;width;height;ROI.x1;ROI.y1;ROI.x2;ROI.y2;class
```

There is always only one traffic sign in the image. It takes around 60% of the image, making it the major item. The dataset is divided into training and testing set containing around 39 000 and 12 500 traffic signs respectively. As the objective of this study is the recognition of mainly mandatory and prohibitory traffic signs, the number of classes was reduced to 29. Priority signs as yield, stop and main road sign were amongst the detected classes. There are no real background images in the dataset, therefore, the undetected sign classes were used as background. Approximately 8 000 background images were excluded from the dataset as there was no need for so many samples and it reduced the training time by couple of hours.

3.1.2 Author's dataset

Images in this dataset were taken in Brno, Czech Republic by iPhone 6 camera with 8 megapixel resolution and f/2.2 aperture. They were taken at a steady walking pace approximately every 2 seconds. There are 262 RGB images in the dataset with dimension varying between 21x21 and 787x940 pixels. Every image is annotated in the following manner:

```
filename;class
```

Annotation of bounding boxes and any sort of data augmentation technique to extend the dataset was not used due to lack of time. Images were cropped to make traffic sign the major scene object, but with more background than GTSRB to achieve viable results in detection and classification. Samples larger than 180x180 pixels were scaled down to one fifth of their size. Traffic signs in this dataset may be occluded, blurred, grainy and with mediocre illuminance changes in the scene. These images were not used during training, they are for testing purposes only.

3.2 Hardware and software

Training of neural networks is computationally very expensive. In terms of mathematical operations used, it represents matrix multiplication. This process can show some similarities to rendering of an image, for which graphic cards are used. GPUs have more computational power than a CPU and are optimized for this kind of operations. Based on this knowledge, a decision to use graphic card as main computational device in this study was made. Nvidia is the current market leader in the best performing GPUs, therefore, a card from this company was chosen. Following hardware configuration was used. Intel Core i7-6700K CPU clocked at 4.5 GHz frequency, 16 GB of DDR4 3200 MHz RAM and one EVGA GeForce GTX 1080 SC GPU.

The software chosen was MATLAB due to multiple reasons. It has many toolboxes available, including toolboxes for deep learning and parallel computing, which is necessary to train the network on GPU. Another reason why MATLAB was chosen over other platforms is that the author was already familiar with it. The version used was R2018a because of its full CUDA support for Nvidia Pascal architecture and deep learning functions availability, which previous versions available to the author lack.

3.3 Implementation

As mentioned in the previous section, MATLAB was the chosen software platform for NN training and Faster R-CNN as the default type of network utilized. MATLAB's Faster R-CNN detector does not support multi-gpu pool and can train on one GPU only, which is no problem as there is only one graphic card available. To be able to train the network, following data are needed:

- Truth table
- Layers
- Training options

3.3.1 Truth table

Every image in the dataset along with its annotations has to be recorded in the truth table, which has to correspond to format shown in table 3.1. Bounding box is a 1x4 vector providing initial x and y coordinates of the rectangle in which the traffic sign is located and its width and height. If traffic sign of certain class is not located in the image, it is signified by an empty matrix.

FILEPATH	CLASS 1	...	CLASS N
full filepath	bounding box/empty matrix	...	bounding box/empty matrix

Tab. 3.1: Truth table example

3.3.2 Layers

Network layers in MATLAB are described either by a column vector of independent layers or by a Direct Acyclic Graph (DAG). In this thesis, the vector representation was sufficient as there was no need for more complicated branching of the network. Layers are defined as shown in the following sample code.

Listing 3.1: Definition of layer types in MATLAB

```
layers=[
    imageInputLayer(imageSize)
    convolution2dLayer(kernelSize,numKernels,...
        'Stride',strideStep,'Padding',padSize)
    reluLayer()
    crossChannelNormalizationLayer(numChannels)
    dropoutLayer(probability)
    maxPooling2dLayer(poolSize,'Stride',strideStep,...
        'Padding',padSize)
    fullyConnectedLayer(numNeurons)
    softmaxLayer()
    classificationLayer()
];
```

where `imageSize` are the dimensions of input image, `kernelSize` is the size of the kernel, `numKernels` represents the number of kernels used, `strideStep` is the number of pixels to step over, `padSize` is the number of zeros added around the matrix, `numChannels` represents the number of channels used to perform normalization of element, `probability` is the probability of disabling the neuron, `poolSize` is the size of pooling matrix and `numNeurons` is the number of neurons in fully connected layer.

3.3.3 Training options

These options specify which minimization algorithm is used, what batch size and learning rate to use, and how long should be the network trained.

Listing 3.2: Example of training options in MATLAB

```
options = trainingOptions('sgdm', ...
    'MaxEpochs', 5, ...
    'MiniBatchSize', 1024, ...
    'InitialLearnRate', 1e-3, ...
    'ExecutionEnvironment','gpu',...
    'Shuffle','every-epoch');
```

3.3.4 Training

Listing 3.3: Example of network training in MATLAB

```
net = trainFasterRCNNObjectDetector(truthTable,...
    layers,trainingOptions);
```

After using the command shown in listing 3.3, the training process has started. The network is fitted four times with RPN trained after first and third repetition. The region proposal network architecture is created by MATLAB by default.

3.3.5 Designs

In the first design AlexNet has been used as an architecture template, having modified the input layer size to 15x15x3 instead of the default 227x227x3, changing filter size in convolutional layers, reducing the number of max pooling layers and setting the number of neurons in fully connected layers to 1024. All these changes had to be made in order to use this neural network on selected dataset. The network was trained on 5 000 sample images to save training time. This network resulted in disappointing performance with precision of 54%, 77% recall and 0.63 F_1 score. It has generated so many false positives that number of detected objects almost doubled the number of images in testing set.

Because of this failure, new network with only two convolutional layers was used. It had kernel size of 3 with stride of 1 and zero padding of 1, which did not reduce the feature map size. Parameters as number of channels in LRN layer was reduced to 3, stride of max pooling layer was changed as well. The number of filters remained unchanged from AlexNet. The architecture used can be seen in table 3.2.

Layer	Name	Kernels/Neurons	Kernel size	Stride	Padding	Add. layers
1	Input	—	—	—	—	—
2	Conv1	96	3x3	1	1	ReLU, LRN
3	Conv2	256	3x3	1	1	ReLU
4	MaxPool2	—	3x3	3	0	—
5	FCNN1	1024	—	—	—	ReLU, Dropout
6	FCNN2	1024	—	—	—	ReLU, Dropout
7	FCNN3	29	—	—	—	Softmax

Tab. 3.2: Architecture of the second network

Training time of this network was greatly reduced compared to previous attempt. It generated fewer false positives, as can be seen in its confusion matrix represented by table 3.4 in columns 2 and 3, and generally performed better with 58% precision and 86% recall.

The results were still unsatisfying, therefore multiple networks were trained with increasing number of convolutional layers with ReLU activation functions. Every new layer added 384 3x3 kernels with stride 1 and 1 zero padding. Another LRN layer with 3 channel normalization was added to the second convolutional layer. This increase in layers certainly improved performance as shown in tables 3.3 and 3.4.

Number of conv. layers	Precision	Recall	F_1 Score
2	58.19%	85.69%	0.693
3	67.91%	92.61%	0.784
4	75.00%	91.25%	0.823
5	72.66%	87.33%	0.793

Tab. 3.3: CNN result comparison

Each network was trained on 5 000 samples randomly chosen from the whole dataset and then tested on 12 630 images. The training time was in average around

	2 layers		3 layers		4 layers		5 layers	
	+	-	+	-	+	-	+	-
+	7808	5611	8440	3988	8386	2796	7820	2943
-	1304	1394	673	1956	804	2406	1135	2468

Tab. 3.4: CNN result confusion matrices

50 minutes with increase of approximately 2 minutes for every additional convolutional layer. Based on results in table 3.3, neural network with four convolutional layers performed better than other networks and was selected for final training described in the next section.

3.4 Results

The final Faster R-CNN network was trained on the whole dataset with the architecture shown in table 3.5, using stochastic gradient descent with momentum (SGDM). Following parameters were used, batch size of 1024, 0.9 momentum value, 0.001 learning rate and trained 4 times for 5 epochs. Training set was shuffled every epoch.

Layer	Name	Kernels/Neurons	Kernel size	Stride	Padding	Add. layers	Output
1	Input	—	—	—	—	—	15x15x3
2	Conv1	96	3x3	1	1	ReLU, LRN	15x15x96
3	Conv2	256	3x3	1	1	ReLU, LRN	15x15x256
4	Conv3	384	3x3	1	1	ReLU	15x15x384
5	Conv4	384	3x3	1	1	ReLU	15x15x384
6	MaxPool4	—	3x3	3	0	—	5x5x384
7	FCNN1	1024	—	—	—	ReLU, Dropout	1x1024
8	FCNN2	1024	—	—	—	ReLU, Dropout	1x1024
9	FCNN3	29	—	—	—	Softmax	1x1

Tab. 3.5: Architecture of the final Faster R-CNN

Training of the network took approximately 17 hours on single GPU. These additional samples helped to improve precision by 11% and recall by 7%. The outcome has shown very good performance on GTSRB testing set but performed poorly on the author's dataset, which is presented in tables 3.6 and 3.7.

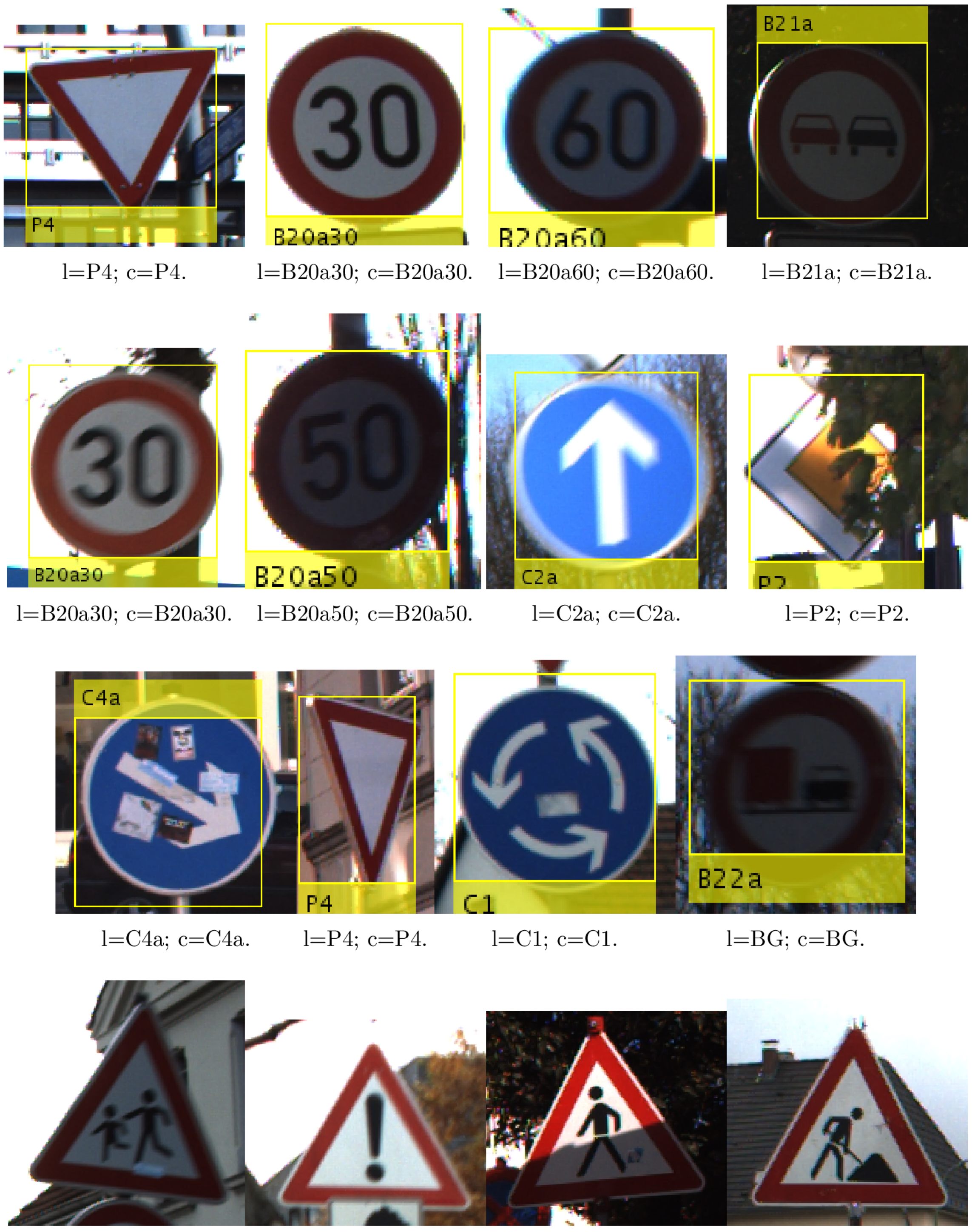
	GTSRB	My dataset
Precision	86.76%	30.87%
Recall	98.47%	95.12%
F_1	0.922	0.466

Tab. 3.6: Result comparison of testing sets

	GTSRB		My Dataset	
	+	-	+	-
+	9562	1459	117	262
-	149	2136	6	26

Tab. 3.7: Confusion matrices of testing sets

Bounding box of the sign is sometimes off due to mistakes in GTSRB annotations on which the network was trained. Many false positives are generated on images larger than 200 pixels in either dimension, due to RPN anchor scaling, which does not provide large enough bounding box. The detection time for one sign is 50 milliseconds for the first testing set and 86 milliseconds for the second. This demonstrates that the detection network requires very small amount of background in the image and that it is very sensitive to its change. This could remedied by training the detector on images taken directly in traffic. Labels used as classification output are corresponding to the official labels defined by the traffic law. Figures 3.1 and 3.2 illustrate a small part of the recognition results. The network is able to correctly classify occluded and blurred traffic signs. It is also almost invariant to lightning changes and change of view angle. False positives are mainly generated by traffic signs that are dependent on rotation or include similar part as other signs.



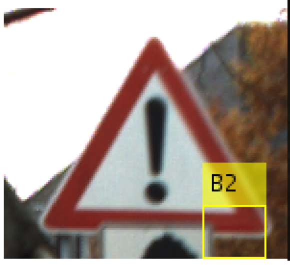
l=P4; c=P4. l=B20a30; c=B20a30. l=B20a60; c=B20a60. l=B21a; c=B21a.

l=B20a30; c=B20a30. l=B20a50; c=B20a50. l=C2a; c=C2a. l=P2; c=P2.

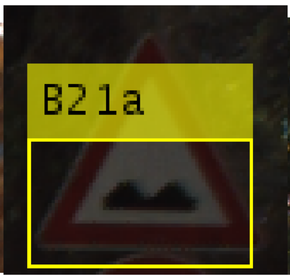
l=C4a; c=C4a. l=P4; c=P4. l=C1; c=C1. l=BG; c=BG.

l=BG; c=BG. l=BG; c=BG. l=BG; c=BG. l=BG; c=BG.

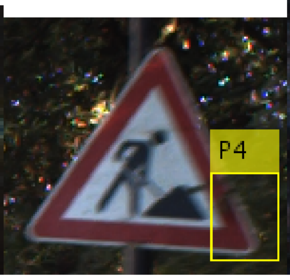
Fig. 3.1: Correctly recognized signs; l – label given by detector, c – class of the sign, BG – background



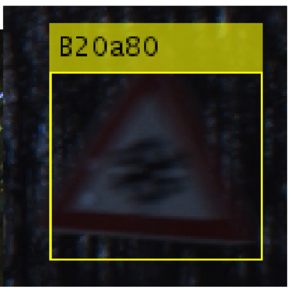
l=B2; c=BG.



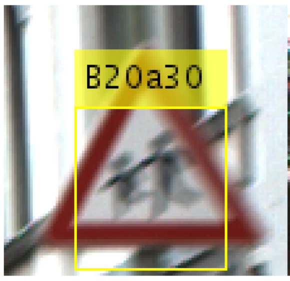
l=B21a; c=BG.



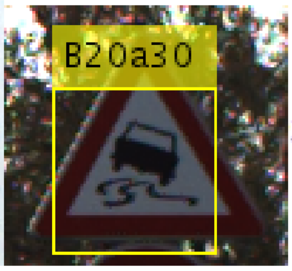
l=P4; c=BG.



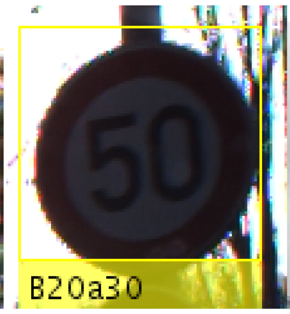
l=B20a80; c=BG.



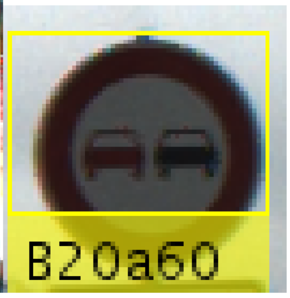
l=B20a30; c=BG.



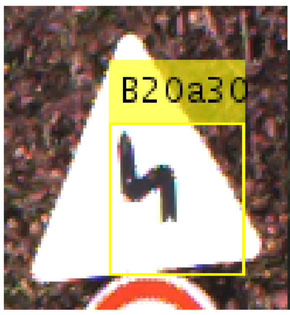
l=B20a30; c=BG.



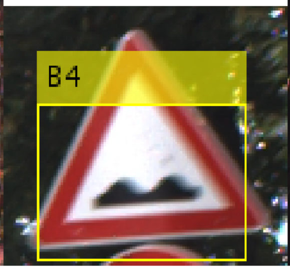
l=B20a30; c=B20a50.



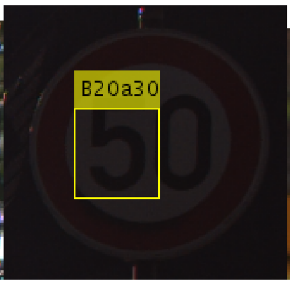
l=B20a60; c=B21a.



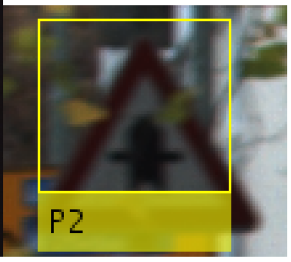
l=B20a30; c=BG.



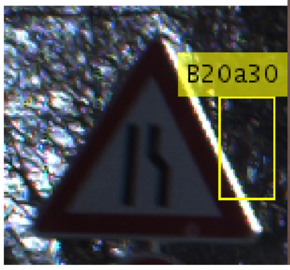
l=B4; c=BG.



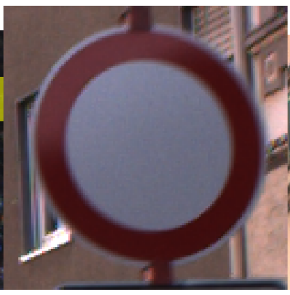
l=B20a30; c=B20a50.

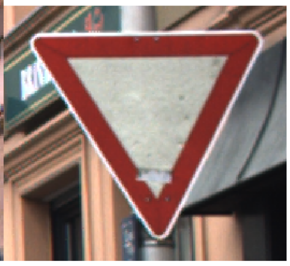


l=P2; c=BG.



l=B20a30; c=BG.


l=BG; c=B1.


l=BG; c=P4.

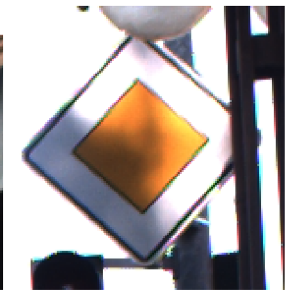

l=BG; c=P2.

Fig. 3.2: Incorrectly recognized signs; l – label given by detector, c – class of the sign, BG – background

4 Conclusion

Research of traffic sign recognition methods was concluded in this thesis, which has shown that Convolutional Neural Networks provide superior performance in detection.

Based on the results, multiple Faster R-CNN networks were trained to determine the best layer configuration. The whole solution was implemented in MATLAB R2018a and computed on a CUDA-accelerated Nvidia GeForce GTX 1080 SC GPU. Network with four convolutional layers has outperformed all other networks.

It was trained and tested on GTSRB dataset with 86.76% precision, 98.47% recall and 0.922 F_1 score. The network was further verified on the dataset created by the author, which produced F_1 score of 0.466. This drop in performance can be explained by way higher content of background in the images to which the network is very sensitive.

Future work would consist of creating a new dataset with images from actual traffic situations and further improvement of the network's architecture to make it more robust.

Bibliography

[1] DE LA ESCALERA, A.; MORENO, L. E.; SALICHS M. A.; ARMINGOL, J. M. *Road traffic sign detection and classification* in IEEE Transactions on Industrial Electronics, vol. 44, no. 6, pp. 848-859, Dec 1997. [online] Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=649946&isnumber=14174>

[2] CANNY, J. *A Computational Approach to Edge Detection* in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986. [online] Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4767851&isnumber=4767846>

[3] SHOJANIA, H. *Real-time Traffic Sign Detection.* [online] Available at: <http://hassan.shojania.com/pdf/TrafficSignDetection-Paper.pdf>

[4] SHI, M.; WU, H.; Fleyeh, H. *Support vector machines for traffic signs recognition.* 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 3820-3827. [online] Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4634347&isnumber=4633757>

[5] VAPNIK, V. *Statistical Learning Theory.* 1st Edition. Wiley, New York, NY, 1998.

[6] LIENHART, R.; MAYDT, J. *An extended set of Haar-like features for rapid object detection* Proceedings. International Conference on Image Processing, 2002, pp. I-900-I-903 vol.1. [online] Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1038171&isnumber=22252>

[7] BARO, X.; ESCALERA, S.; VITRIA, J.; PUJOL, O.; RADEVA, P. *Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification* in IEEE Transactions on Intelligent Transportation Systems, vol. 10, no. 1, pp. 113-126, March 2009. [online] Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4770199&isnumber=4796392>

[8] ZHU, Z,; LIANG, D.; ZHANG, S.; HUANG, X.; LI, B.; HU, S. *Traffic-Sign Detection and Classification in the Wild.* 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2110-2118. [online] Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7780601&isnumber=7780329>

[9] SERMANET, P.; EIGEN, D.; ZHANG, X.; MATHIEU, M.; FERGUS, R.; LECUN, Y. *Overfeat: Integrated recognition, localization and detection using convolutional networks.* Cornell University Library [online] Available at: <https://arxiv.org/abs/1312.6229>

[10] ZAPLETAL, Ondřej. *Image Recognition by Convolutional Neural Networks - Basic Concepts.* Brno, 2017, 68 p. Master's Thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Advised by Ing. Karel Horák, Ph.D. [online] Available at: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=146227>

[11] An Intuitive Explanation of Convolutional Neural Networks, Data science blog, 2016 [online] Available at: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

[12] CS231n Convolutional Neural Networks for Visual Recognition, Stanford. [online] Available at: <http://cs231n.github.io/convolutional-networks/>

[13] Faster R-CNN explained, Medium, 2017 [online] Available at: <https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8>

[14] REN, S.; HE, K.; GIRSHICK, R.; SUN, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* Cornell University Library [online] Available at: <https://arxiv.org/abs/1506.01497>

[15] STALLKAMP, J.; SCHLIPSING, M.; SALMEN, J.; IGEL, C. *The German Traffic Sign Recognition Benchmark: A multi-class classification competition.* In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011. [online] Available at: <https://ieeexplore.ieee.org/document/6033395/>

[16] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. *Imagenet classification with deep convolutional neural networks.* [online] Pages 1106–1114, 2012. Available at: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

List of symbols, physical constants and abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DAG	Direct Acyclic Graph
ECOC	Error Correction Output Code
F-ECOC	Forest Error Correction Output Code
GPU	Graphical Processing Unit
GTSRB	German Traffic Sign Recognition Benchmark
ML	Machine Learning
RAM	Random Access Memory
R-CNN	Regional Convolutional Neural Network
ReLU	Rectified Linear Unit
ROI	Region of Interest
RPN	Region Proposal Network
SC	superclocked – providing higher level of overclocking by factory
SGDM	Stochastic Gradient Descent with Momentum
SVC	Support Vector Classification
SVM	Support Vector Machine

List of appendices

A Contents of the CD

The code on the CD was run in MATLAB R2018a. No other versions were tested.

```
/ ...................................................... root folder of the CD
├── CNN ................................. Folder with MATLAB code and datasets
│   ├── Data ........................................ Folder containing datasets
│   │   ├── DatasetBrno ................................ Dataset created by author
│   │   ├── GTSRB ...................... German Traffic Sign Recognition Benchmark
│   │   ├── annotationsDB.mat
│   │   ├── annotationsGTSRBtest.mat
│   │   ├── trainedDetector.mat
│   │   └── truthTable.mat
│   ├── detectandVerify.m
│   ├── generateSmallTrainSet.m
│   └── script.m ..................................................... Run this file
└── Zakarovsky,Traffic_Sign_Recognition.pdf ..................... Bachelor's thesis
```