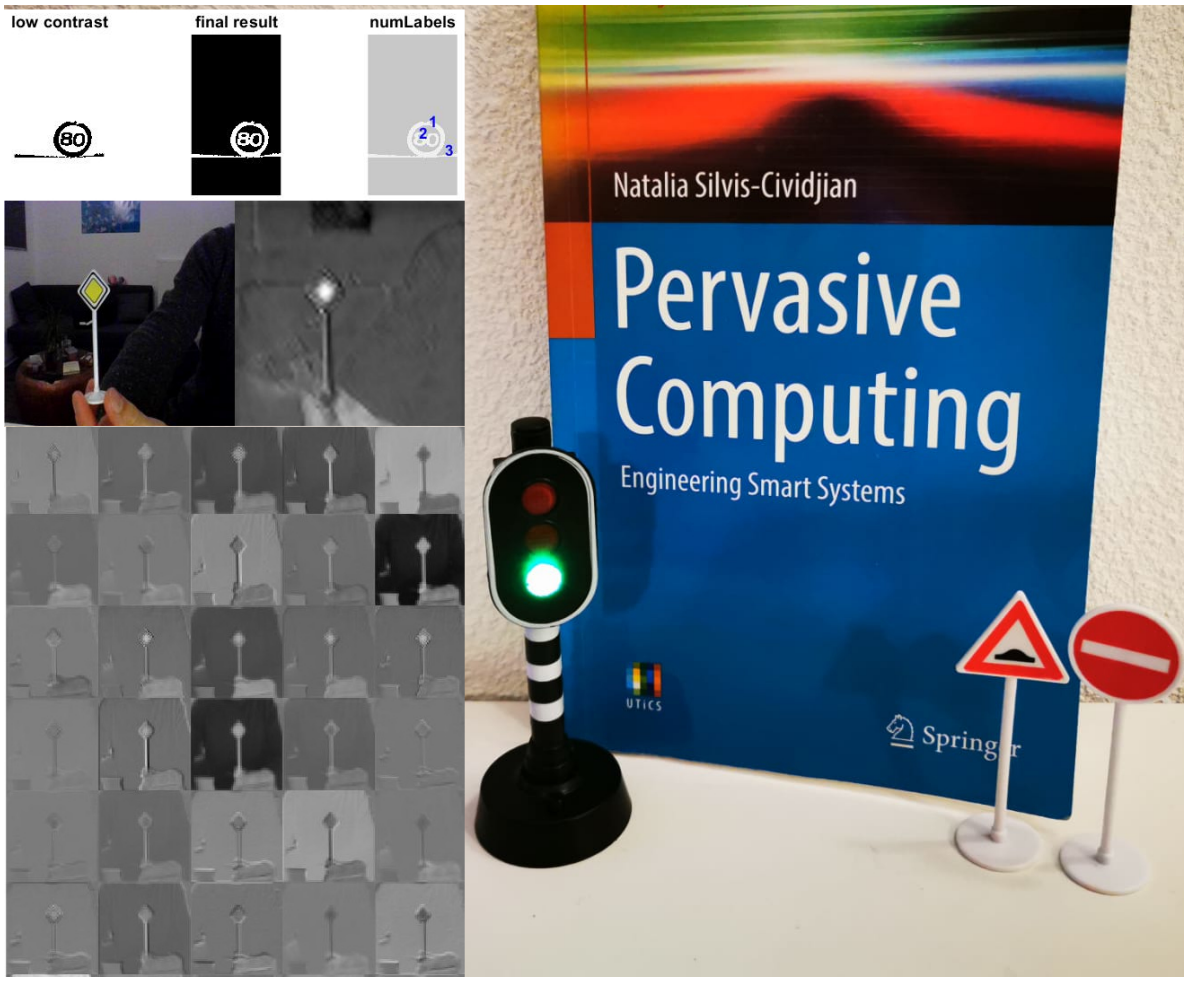
Classification & distance estimation of traffic objects using a convolutional neural network and morphological operations



Pervasive Computing project by:

Stephen Kwan

Saman Shahbazi

Table of content

Introduction

Imagine you're you're driving at a speed of 80 km/h in a *one-way* road and you're about to join a new road. Even though you're usually an attentive driver, you're tired and are in an unfamiliar area and fail to see a new sign that states *"Danger: two way road ahead!*". You're human, and humans make mistakes. This could potentially lead to a fatal accident, all because you failed to read a sign and assumed you're still driving in a *one-way* road. This is one example where failing to see a traffic sign could have massive permanent negative consequences. This scenario is amplified and is even more likely if you're an unexperienced or senior driver, or worse a driver under influence. [1].

To really illustrate the importance of this subject, take some time to process the fact that in 2018, *1.35* million deaths worldwide were caused by traffic accidents. Making it the number one cause of death among children aged 5-14 and among young adults aged 15-29 [2].

Modern vehicles are in general safer and are equipped with ever more advanced safety systems that have decreased road fatalities by a sizable amount. [3] However, as the figures from 2018 show, there is still a lot to be improved.

With the auto industry inevitably moving towards automatization, detection, identification, classification and distance management of traffic objects play a crucial role in the development of such safety systems.

Maybe the scenario described earlier could simply be prevented by a vocal warning system that would raise your attention to the changing road situation up ahead, even take preventive measures when you fail to take action. We believe such a future is not too far ahead.

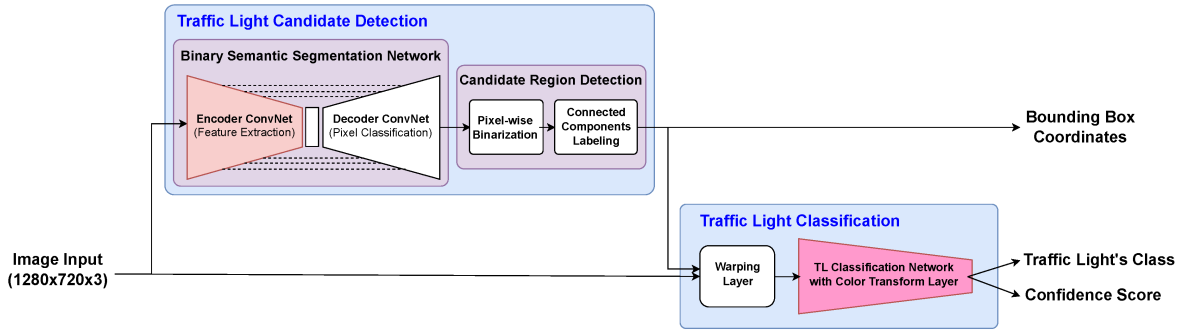
Our project focuses on a small part of this equation, we attempt to design a system which does the following

1. Measures the distance to an (assumed) traffic object
2. When in proximity: classify the traffic object

Literature Study

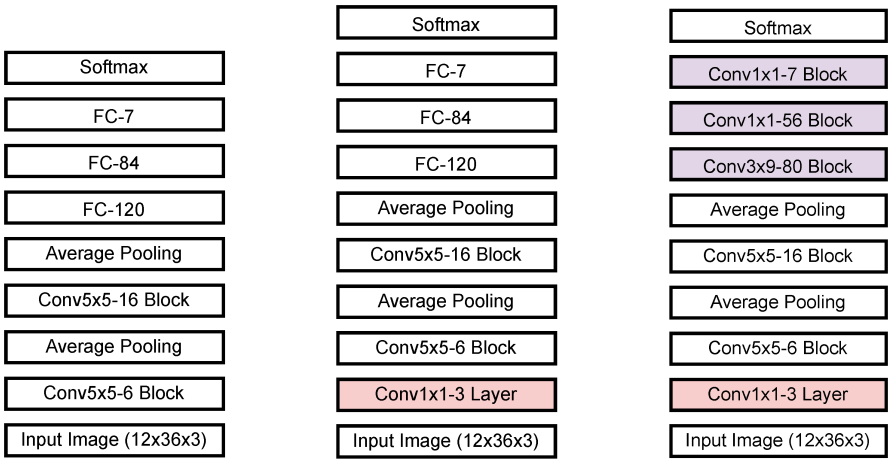
In preparation for this project, we examined several sources that discuss different methods to implement traffic object recognition and classification.

In the first study by Hyung-Koo Kim et al. [4], the team proposed a 2 staged deep-learning-based traffic light recognition system, as can be seen in the image below.



In the first stage they use a pixel-wise semantic segmentation neural network to detect traffic lights. Hyung-Koo Kim et al. state that the idea behind the proposed method is to employ pixel-wise semantic segmentation that is applicable to very small objects as "conventional object detection (OD) methods are not suitable to detect small objects, because they use very deep depth networks with pooling operations for feature extraction".

In the second part, namely the classification stage, the team designed and implemented 3 convolutional neural networks (CNN) as pictured below. (Each convolutional block is composed of a convolution layer, a batch-normalization layer, and an activation function).



(**a**) TLC1Net (**b**) TLC2Net (**c**) TLC3Net

The results showed that the traffic light recognition method outperformed conventional Faster R-CNN in terms of recognition performance.

In the study by Sadna et al. [5], an overview is given of several detection & classification methods for detecting traffic signs. This was a great article as it referenced many different methods and studies. For the detection stage they discuss the following detection studies and methods:

- Colour based methods
- Shape based methods
- Learning based methods

For classification, they discuss the following studies and methods:

- Handcrafted (manual) learning methods
- Deep learning methods

Furthermore, the books *Guide to Convolutional Neural Networks* [1] and the Deep Learning book (chapter 9) [6] were used as to gain insight into inner workings of convolutional neural networks.

From these studies we concluded that the best system for a traffic light/sign detection & classification would be a 3 stage system, as follows:

1. Detect the region of a traffic object from a given image using an Region-CNN or a similar region bounding CNNs.
2. Use a CNN to classify the super category (I.E. the type of traffic object)
3. Use a CNN to classify the traffic object

Because of limited time, resources and inexperience with such a project we decided to keep our project simple an focus on stage 3 mostly.

Concept of operations

The possible stakeholders of this system are, vehicle manufacturers, vehicle consumers, insurance companies and researchers in convolutional neural networks. Our research and results could help improve Advanced Driving Assistance Systems (ADAS).

Project Specifications

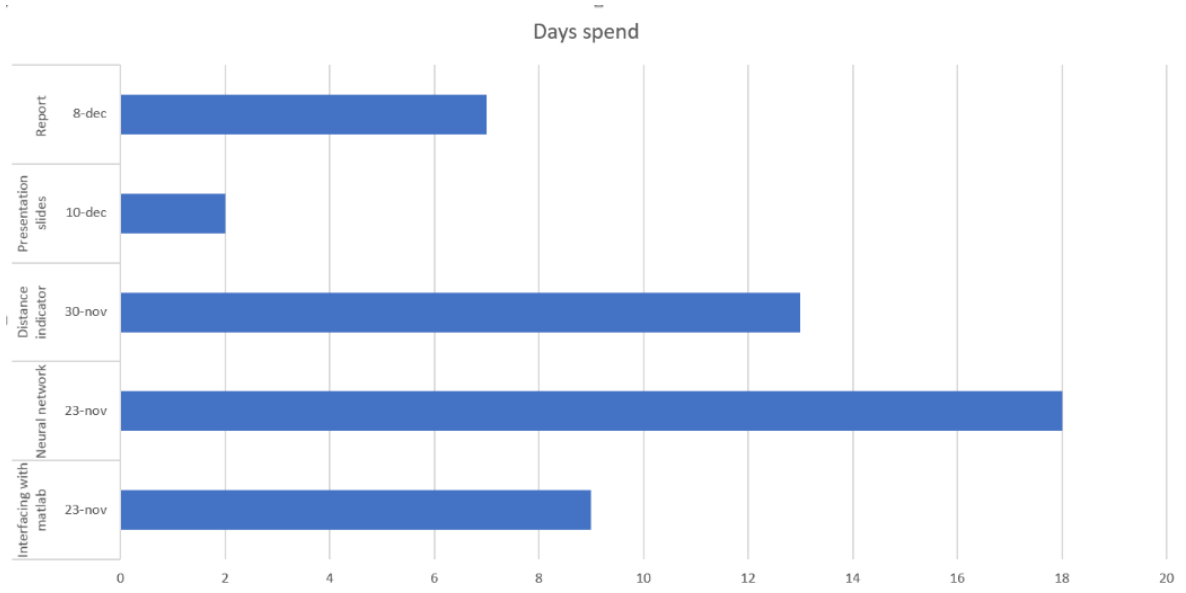
It is worth mentioning that initially we had planned to make a semi-autonomous mini driving robot which would follow lines drawn on the ground and react to traffic objects on it's path. However, due to trouble interfacing our robot with our framework and the limited time we had, we did not continue with this system.

Furthermore, we choose to use our own dataset of traffic objects, this consisted of 5 different classes: namely, *traffic light* (green + red), a cautionary sign *bump ahead* sign, a *yield* sign and a *do-not-enter*. All of these objects were in relatively empty albeit not completely surroundings (not too many other conflicting objects in the background).

Project Plan

We decided to indicate our time distribution of the project that has lasted approximately 3 weeks with a Gantt chart.

Note that all parts of the project were successfully completed *except* for the interfacing with MATLAB.



Date	Tasks
Project Week 1 11-29-2019 to 12-02-2019	Literature Study Interfacing Neville with Matlab System requirements Neural network research Gathering dataset
Project Week 2 12-02-2019 to 12-09-2019	Neural network experimentation (Interfacing Neville with Matlab) Distance detection experimentation Prepare presentation Prepare report
Project Week 3 12-09-2019 to 12-15-2019	Presentation Finish Classification systems Work on report

Task Division

Name	Task(s)
Stephen Kwan	Distance Detector Interfacing Matlab & Neville Presentation Report Research
Saman Shahbazi	Traffic object detection Dataset gathering Presentation Report Research

Tools

We used Matlab for all of our software implementations. The following toolboxes were used within Matlab:

- Neural Network Toolbox
- Image acquisition toolbox

Further, we used a Microsoft Webcam to capture training data and to test our system.

To train the neural network one must have a relatively capable CPU and GPU. Our network was trained using a GeForce GTX 1050TI GPU and Intel® Core™ i5-8300H Processor with 8 GB of RAM.

System Requirements

As mentioned previously, our project was initially planned to be much larger (with a driving robot car) and therefore our MoSCoW approach was also based on this initial design.

Requirements	Must have	Should have	Could have	Won't have
Functional	Follow the driving.	Measure distance to traffic object. Classify the correct colour on a traffic light and react accordingly.	Classify signs.	Too many objects in the driving ground.
Non-functional	Keep correct speed and the robot must not veer off the lines.	Accuracy of distance indicator should be within 5 cm. The CNN should classify the traffic light colour with at least 90% accuracy. The robot should react on time.	CNN should classify the traffic signs with at least 90% accuracy.	

System Design

Interfacing with MATLAB

The initial plan was to control the robot with MATLAB. However, we weren't able to accomplish the goal due to the difficulty of it. We genuinely tried our hardest but lost too much time thus we fully focused on the actual classification.

Distance Indicator

We are utilizing morphological operation techniques in combination with rule-based classifying in order to obtain the distance between the object and the capturing device (camera).

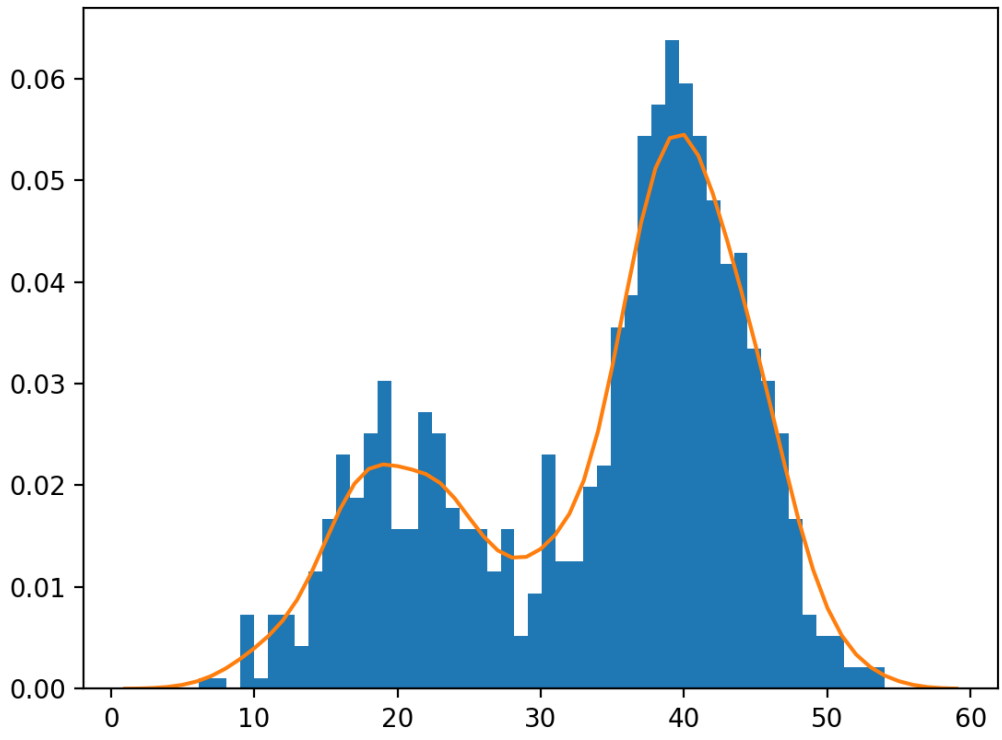
Methods

Note that the methods are described in the algorithmic order of the source code, the algorithm would simply be invalid if the order is changed.

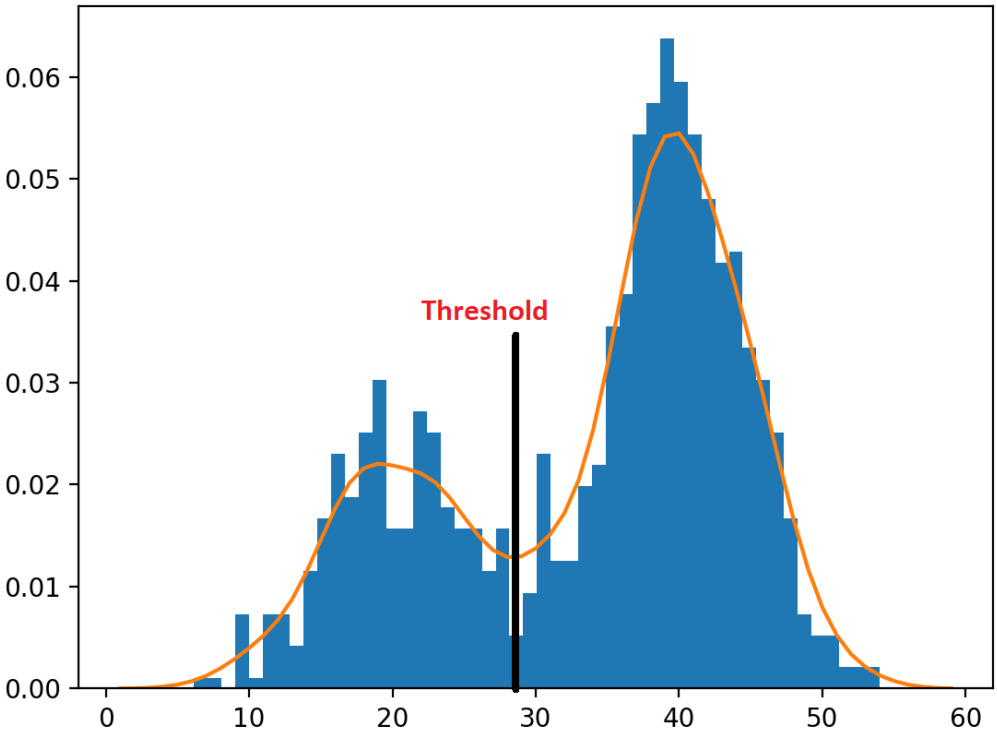
Point processing

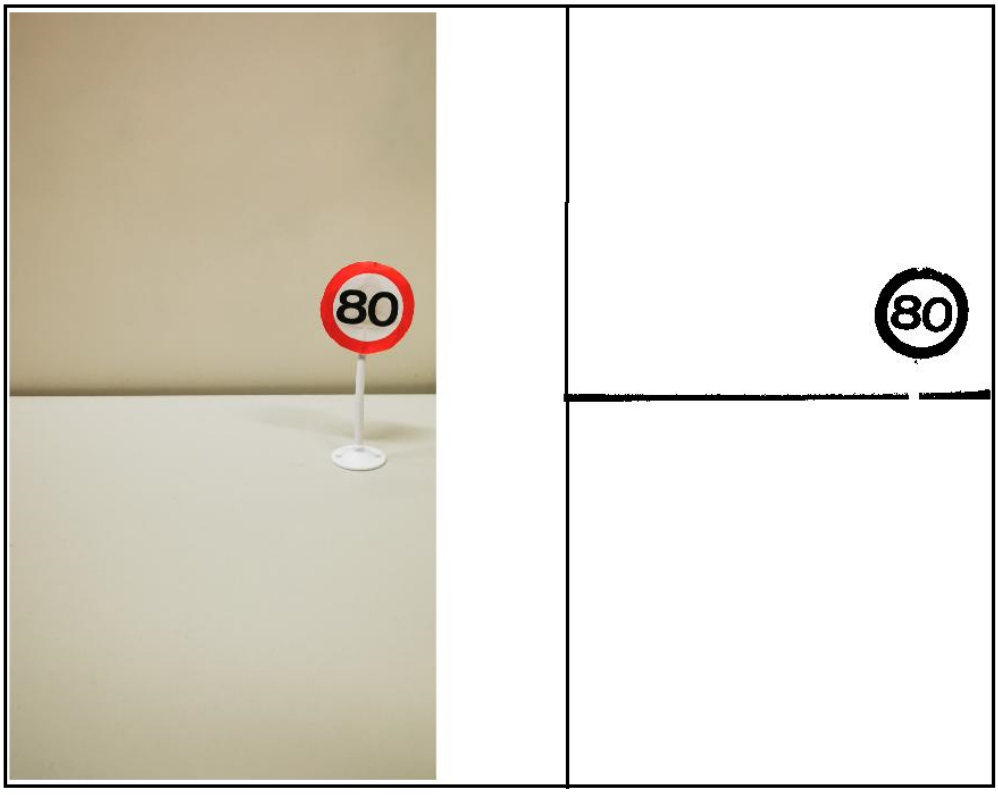
In order to achieve a full black and white picture without grey. It is essential to convert the non fully black or white(grey) into either pitch black or plane white, specifically; by determining a certain fixed threshold, every pixel that exceeds the threshold will be black and vice versa, the value of black is zero and white is 255.

Firstly, generate a histogram with the data that would indicate the frequency on the y-axis and the colour intensity on the x-axis.



Secondly, choose the x value that is right in between the two peaks



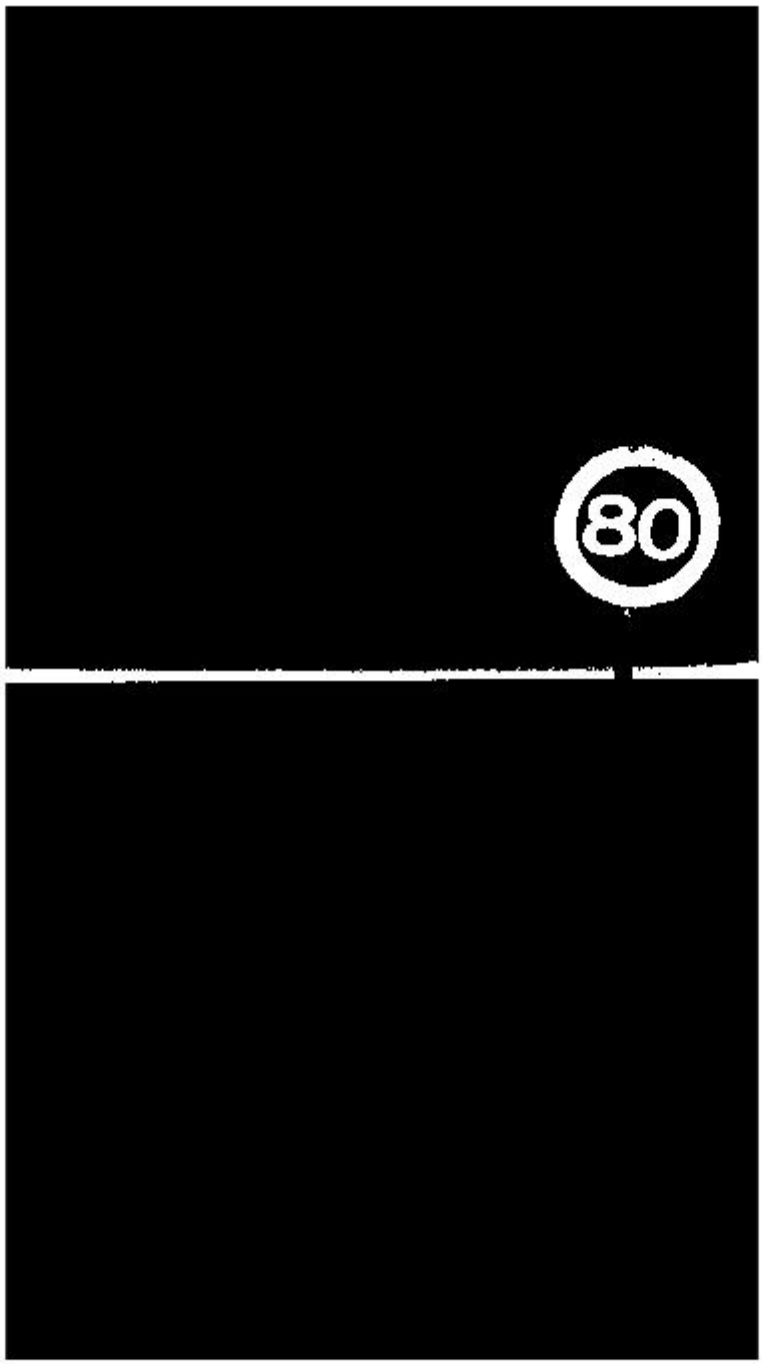


BEFORE **AFTER**

Morphological operation

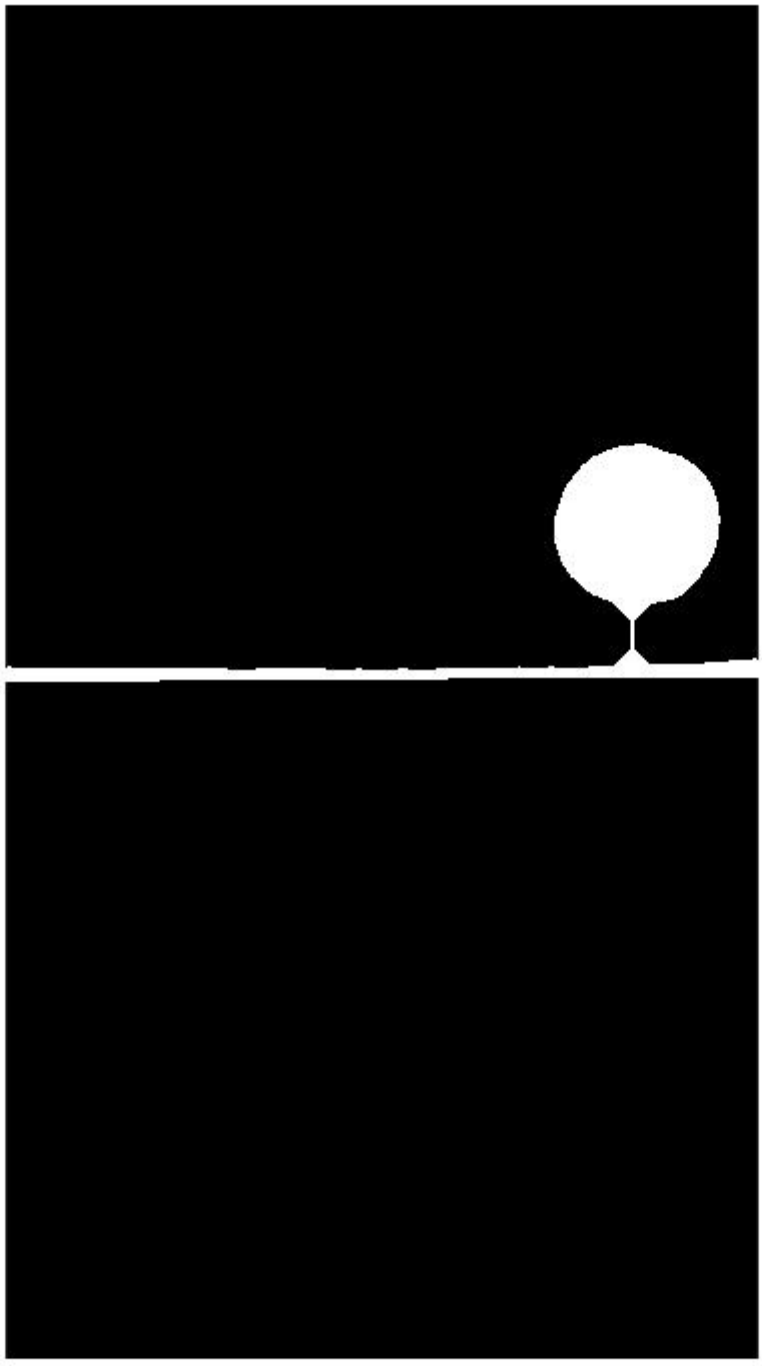
This operation is used to eliminate more noise, it is done with a certain value that is equivalent to the amount of pixels, this part is especially important for the inner part of the object, in our case: it is crucial to erase the inner part of the sign board and try to preserve the outer part(circle) in order to get a viable area value eventually.

A key point before applying this operation is to complement the point processed picture first because otherwise there is no way to see how much noise is removed on a micro scale level.



COMPLEMENTED (BEFORE)

It is *absolutely* critical to find the right value/balance, if the value is too high the object would *deform* and whereas if the value is too low the elimination would be too minimum which will lose it's meaning to apply this operation.



AFTER

Rule based classification

After 'filtering' , getting the actual data(in this case the area) and inserting into the distance formula(see below for elaboration) some conditions has to be defined in order to prevent invalid answers. in this case: the distance cannot be negative, otherwise return false to the user and indicate it in the console.

```
if sign_distance > 0
    fprintf("The distance of the object is: %f cm", sign_distance); %Distance
indication
else
    fprintf("Unknown distance");
end
```

Form factor

The form factor is based on a mathematical formula (see below) that indicates the round-ness of the object, it is between 0 and 1.

The 1 indicates that the object is fully 2 dimensional singular sphered, the 0 indicates a straight line.

However, In order not to make this project rather too simple, we used a picture that contains a straight line in the picture thus the form factor is unreliable in this project.

A = Area
P = parameter

$$f = 4\pi A/P^2$$

Distance formula

In order to obtain the distance it is crucial to establish a solid mathematical formulation of the distance order to calculate the distance. However it is significantly situational based to the resolution since the algorithm basically counts the pixels as the area, the heavy dependency of the morphological value that is determined which manifests that the formula can't be standardized. Thus, the intensity level's variable X is different for the traffic light and the sign board in this case. Also, brute force is required in order to obtain the valid value of X.

A = Area

X = pixels ratio

M = Maximum distance

$$f = M - (A/X)$$

Classification step process

(1)Capturing the picture → (2)cropping → (3)greying → (4)point processing → (5)complementing → (6)Morphological operation → (7)numerating → (8)colouring → (9)Acquire specification → (10)Establish distance formula →(11)Distance calculation

1. The pictures were captured with the Microsoft webcam from the lab, the pictures different from in each other in the sense of distance, the angles didn't change tremendously.
2. In order to get rid of the spontaneous shades of the corners it is crucial to crop the image. In addition, it is essential that every picture has to be cropped at fixed coordinates in order to keep track of the distance that will be eventually calculated.
3. Greying the picture.
4. Converting the picture to only black and white(point processing) so the colour grey is also eliminated, we used the im2bw function with an automatic threshold chooser by not passing an second argument in the function(see the source code for clarification).
5. Complement the picture (the black turns white and vice versa)

6. Eliminate the noise(Morphological operation) by using the Imopen library
7. Retrieve the specifications of the object, emphasize on the area in this case.
8. Using the vislabels library in order to get the number of object by illustrating and numbering it.
9. Filling each blob with a colour.
10. Formulate it mathematically in such a way by utilizing the area.
11. Obtain the distance by using the established formula from step (10).

Difficulties throughout the classification process

Regardless of all the obstacles we were able to succeed however we encountered quite some problems as well.

It was a success with the pictures that have a clean/white background that contain a minimum amount of noise. However it was rather tragic to classify it in a continues real time spectrum(continues picture capture/video).

It came to my attention that the form factor was smaller when the resolution was lower and bigger when the resolution was higher due to the reason that the form would be rather more 'zig-zaggy' which means that two identical pictures that seems the same but having different resolution properties would not work with this algorithm. Also, the form factor was not reliable, the reason is because that the form factor of the traffic light and the rounded traffic sign are indistinguishable because the deformed shape during the morphological closing process.

Even if the written algorithm would have two objects while there was only just one because of the noise, the size measurement of the actual object was still viable due to the smart system of the ability to specify which of the two object could be chosen from. To elaborate on the size: The reason why the size is heavily emphasized is because of the actual distance insight, the greater the size the shorter the distance and vice versa.

Performing the morphological closing is very situational based since the disk radius has to be done manually and the perfect threshold differs significantly from two objects, which manifests that the flexibility is significantly low. Also, the risk of an overly high threshold would reshape the object into something that would make the object unrecognizable which will result in a false area indication. Also, the morphological closing procedure affects the distance due to the reason that there will be additional space added into the pixel of the object in order to specify it more specifically which translate into a less reliable/precise distance indication.

The mathematical formulation to indicate the distance can not be used as a general case due to the different pixel quantities that may hold, because the area will always differ from each other.

Convolutional Neural Network

Motivation

Traditional neural networks use matrix multiplication by a matrix of parameters, with a separate parameter describing the interactions between each input and output unit. CNNs however, use sparse interactions accomplished by making the kernel/filter smaller than the input. As an example, an input image may have thousands of pixels but the CNN is able to detect features such as edges that are occupy only ten pixels of the image. This in turn means that we can use less parameters, resulting in better computational efficiency. [6]

Experimentation

We experimented with multiple CNN designs. The designs and results are listed below:

All networks were trained with the stochastic gradient descent with momentum as the optimizer.

The first training set (TS1) we had was 1000+ images of the objects and comprised of 10 sets of burst shots (100 per set) with 5 different background locations. We used burst shot because this was the fastest way we could make our dataset. However, we believe that burst shooting the images turned out in our disadvantage as the images were too much alike. Training set two (TS2) was made using 350 images taken individually with mostly very simple backgrounds as we tried to simplify the problem.

In the paper published for AlexNet [7] it was stated that they doubled their dataset by reflecting their images in the Y-Axis. We did the same and TS2 was now 700+ images large. Furthermore, before the switch to TS2 we used an input layer which took images of 32x32 however after we stared using TS2 we also switched to images of size 256x256.

We started out with 4 object. In the last week added another object (the yield sign).

The training diary is displayed in the appendix part Training Diary

Live performance below means turning the webcam on and moving the objects in and out of the screen and counting how many times it's correct and where in the screen it displayed wrong results.

The most promising CNN that we designed (#19 in the training diary) was based off of AlexNet [7], which we aptly named Little-Alex (LA). AlexNet has 5 convolutional layers whereas our network has 3 convolutional specs with similar hyperparameters as AlexNet's first 3 convolutional layers. Even though our network was much smaller than AlexNet's (18 vs 27 layers), we achieved acceptable performance for our relatively simple, intended purposes. However, it's worth mentioning that AlexNet still performed better than our network - but AlexNet is also pretrained with thousands of images.

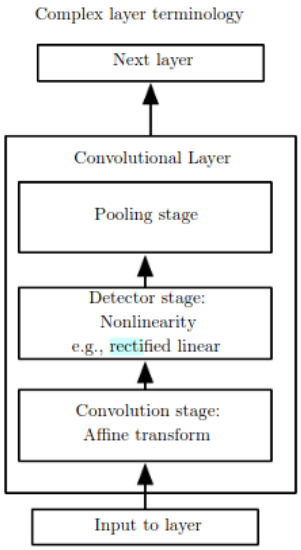
Final design

Our CNN accepts images of 256x256 and consists of effectively 3 convolutional blocks and 2 fully connected layers.

Each convolutional block consists of

- convolutional layer
- Rectified Linear Unit (ReLu) layer
- (cross channel) normalization layer
- max-pooling layer

Similar to the convolutional block illustrated in the figure below [6]:

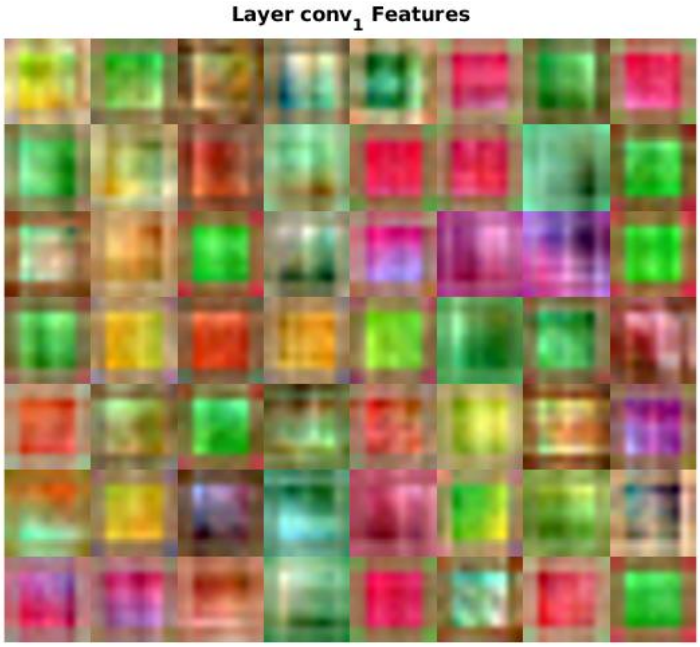


A detailed view of the layers can be seen in the Appendix section Layers.

Layer analysis

Feature Maps

The following image illustrates what kind of images strongly active each filter in convolutional layer 1. The feature maps of the first 56 filters are shown below.



Our objects consisted of 3 colours: Red and green (traffic light) and yellow (yield sign). It is quite clear that the first convolutional layer is trained o pick up on the colours of the network. This makes a lot of sense as we also intentionally did not include many other colourful objects in the background of our training set.

Activations of images

By feeding an object's image to the network, we made 'activations maps' for the object - showing what parts of an image result in high activation numbers in each filter/channel. The activation images are in gray scale, the following is what the colours imply [8]:

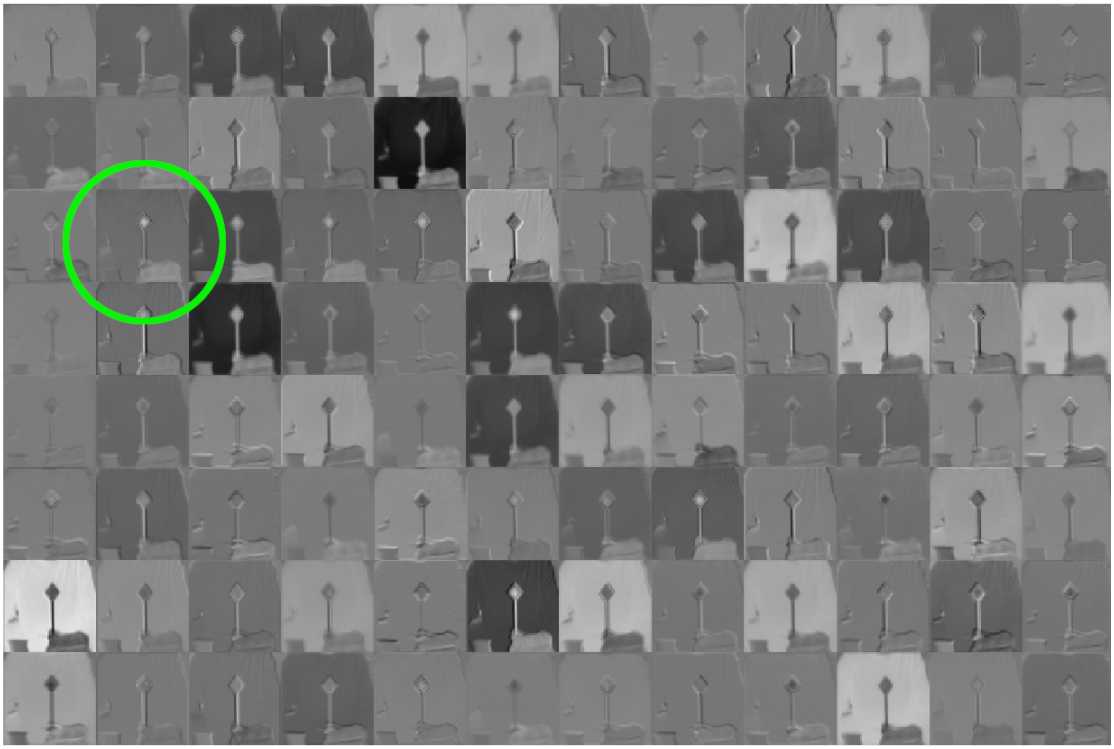
- White Pixels: Strong positive activations
- Black Pixels: Strong negative activations
- Grey Pixels: Neutral activations

Therefore a white pixel at some location in a channel indicates that the channel/filter is strongly activated at that position.

Yield Sign:

Feeding the yield shown below:

Results in the following activation map (all 96 filters/channels) in the first convolutional layer.



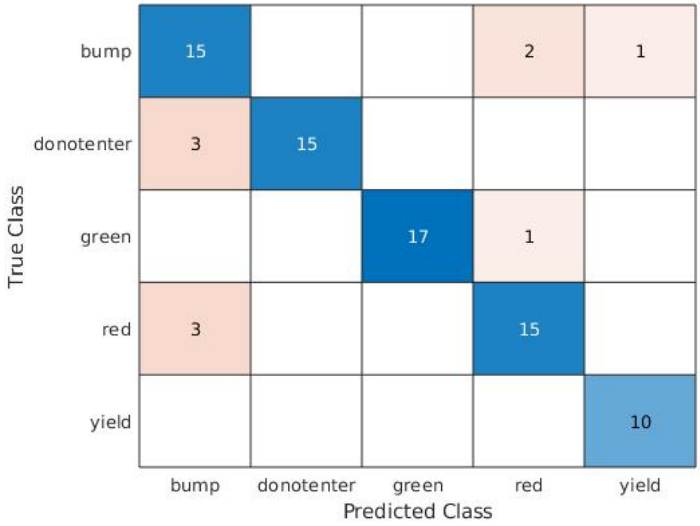
Let us focus on the filter circled in the above map. This is filter number 26, as shown below.

Below is also filter number 26 from the feature map.

Comparing these images together, we can indeed see that filter 26 specializes in finding yellow colors as in the activations map the part of the yield map that is yellow is showing high activations.

Results

We took 84 images that were similar to our training dataset (similar backgrounds) our test dataset and the results when fed to the network were as follows as shown in the confusion matrix:



Bump ahead sign: $\frac{15}{18}$

Do not enter sign: $\frac{15}{18}$

Traffic light on green: $\frac{17}{18}$

Traffic light on red: $\frac{15}{18}$

Yield sign: $\frac{10}{10}$

For an averaged accuracy of 88%.

Conclusions

Even though our testing is not completely realistic (our training dataset was also not completely realistic as we did not have a large enough dataset containing realistic environments), we achieved a relatively high accuracy as shown in the confusion matrix.

We believe that the unique coloured objects such as the traffic light on green and the yield sign (yellow) allowed for a much higher accuracy because there were no other like coloured objects for the network to confuse it with. Whereas the traffic light on red, the *do not enter* sign and *bump ahead* sign are all red of colour, and the network clearly confused these objects on several occasions. Furthermore the *bump ahead* sign and the *yield* sign are shaped the same.

These findings are similar to the findings reported by Kaixuan Xie et al. [9]. By making a two stage classification process where the signs are first sorted by their category and then specified even further in a secondary stage - we believe that these errors can be prevented or at the very least reduced.

Improvements

We believe all of the following have the potential of greatly improving the usability and accuracy of the network:

- Larger Neural Network
- Larger Training Set with varying backgrounds
- Larger Test Set
- Extensive fine tuning of hyper parameters

Appendix

CNN

Layers

Training Set example

4 images from the from the training set of the *bump ahead* sign merged into a single image illustration purposes.

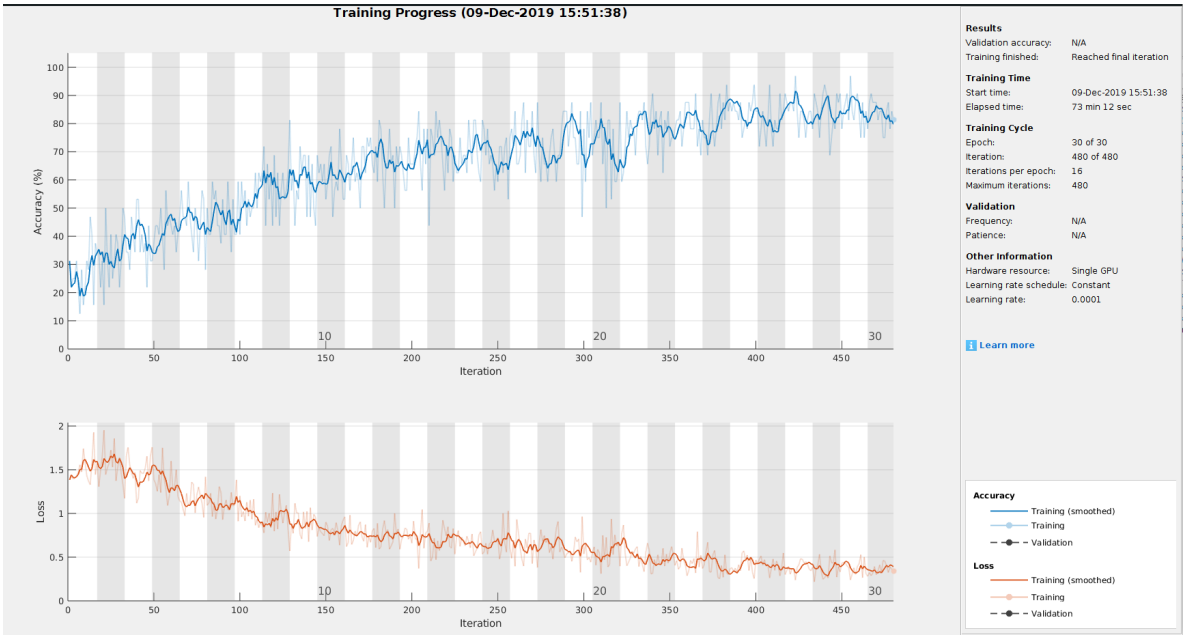
Training Diary

Attempt	CNN Architecture	Tranining Set	Training Options	Training Performance	Live Performance
1.	2 convolutional layers: Conv1: 32 filters of 8x8x3 Conv2: 64 filters of 3x3x32 Fully connected layer: 64 neurons FC layer 4	Training Set 1	Mini-batch size: 4 , learning rate 10^{-4}	10 epochs: 100% accuracy	Very poor
2.	Conv1: 96 filters of 32x32 Conv2: 32 filters of 5x5 Fully connected layer: 64 neurons Fully connected layer: 4 neurons	Training Set 1	Mini-batch size: 4 , learning rate 10^{-4}	Shut down epoch 4 accuracy 20%.	-
3.	Conv1: 16 filters of 8x8 Conv2: 16 filters of 5x5 Fully connected layer: 64 neurons FC layer 4 neurons	Training Set 1	Mini-batch size: 4, learning rate 10^{-4}	Shut down epoch 4 accuracy 40%.	-
4.	3 convolutional layers: Conv1: 96 filters of 11x11 Conv2: 32 filters of 5x5 Conv3: 64 filters of 5x5 Fully connected layer: 64 neurons FC layer: 4 neurons	Training Set 1	Mini-batch size: 4 , learning rate 10^{-4}	Epoch 5: 90% accuracy.	About 35% accuracy

Attempt	CNN Architecture	Tranining Set	Training Options	Training Performance	Live Performance
5.	Same as previous with difference in conv2 being a grouped of group size 16, 2 groups, filter size 15x15	Training Set 1 (TS1)	Same as previous	Epoch 10: 90% accuracy	Poor
6.	Same as previous with difference in conv1 being 16 filters of 8x8	TS1	Same as previous	Epoch 10: 95% accuracy.	Poor.
7.	Added a dropout layer of 50% before the first FC.	TS1	Same as previous	Epoch 10: 90% accuracy	Seems better than 6
8.	Same as before	TS2	Batch size changed to: 16	20 epoch: 90% accuracy	Much better performance. About 50% correct
9.	Conv1 changed to: 32 filters of 12x12 Grouped conv2: 2 groups consisting of 32 filters of 8x8	TS2	Batch size: 32	20 epoch: 90% accuracy	About 60% correct
10.	Conv1: changed to 32 filters of 15x15	TS2	Same as previous	Shutdown epoch 15 at 50% accuracy - not converging	-
11.	Grouped grouped conv2 from (9) changed to 64 filters.	TS2	Same as previous	Not converging - at 20 epoch 50%	-

Attempt	CNN Architecture	Tranining Set	Training Options	Training Performance	Live Performance
12.	Grouped conv2 switched out for single group conv2 of 64 filters of 5x5	TS2	Same as previous	Even slower convergence. Seemed to plateau. Shut down	-
13.	Everything from (9) and conv1 stride set to 4.performs better.	TS2	Same as previous	70% at epoch 25performs better.	Best performance so far. Especially when objects are centered.
14.	Changed conv1 stride to 2.	TS2	Same as previous	Similar to (13)	Worse than (13).
15.	Add cross channel normalization layer after the ReLu layer of the first two conv layers.	TS2	Same as previous	Slower training.	Slightly better performance than (13). Better spread of object detection.
16.	Conv1: 96 filters of 11x11, stride:4, padding:2 Conv2 (grouped): 2x128 filters of 5x5 Conv3 512 filters of 3x3	TS2	Same as previous	30 epochs to converge to 80%.	Much better performance. About 65-70%.
17.	FC1 changed to 128 and also 32.	TS2	Same as previous	FC_1 128: No convergence.	FC_1 32: Worse performance than (16)
18.	Same as (16).	TS2	Learning rate: $10^{-3} $	No convergence.	-
19.	Same as (16) with the exception of FC1 being 64.	TS2	Learning rate back to 10^{-4}	Converges to 80% after 30 epochs	Best performance 80%+ during live session

Little Alex Training Plot



Matlab Code

```
% Returns the 18 layer CNN, Little Alex
% input size is other than 256x256 the readFunctionTrain function has to be
% adjusted as well
% categories example: {'cat', 'hat'}
function [cnn]=littleAlex(trainingFolder, categories, inputSize, maxEpochs)

imds = imageDatastore(fullfile(trainingFolder, categories), 'LabelSource',
'foldernames');
% create image data store with own dataset
imds.ReadFcn = @readFunctionTrain;
% resize images to 256x256

conv1 = convolution2dLayer(11, 96,'Padding',2,'BiasLearnRateFactor',2, 'Stride',
4);
% 96 filters of 11x11 with padding [2 2 2 2] and stride [4 4]
conv1.Weights = single(randn([11 11 3 96])*0.0001);
% initialize random weights for conv1
fc1 = fullyConnectedLayer(64,'BiasLearnRateFactor',2, 'Name', 'fc1');
fc2 = fullyConnectedLayer(length(categories),'BiasLearnRateFactor',2, 'Name',
'fc2');

% create layers
layers = [
    imageInputLayer([inputSize inputSize 3]);
    conv1;
    reluLayer();
    crossChannelNormalizationLayer(5);
    maxPooling2dLayer(3, 'Stride', 2);
    groupedConvolution2dLayer(5, 128, 2, 'Padding', 2); % 2 groups of 128
filters, size: 5x5, padding [2 2 2 2]
    reluLayer();
    crossChannelNormalizationLayer(5); % test
    maxPooling2dLayer(3, 'Stride', 2);
    convolution2dLayer(3, 384, 'Padding', 1); % 384 filters of 3x3 padding [1 1
1 1]
```

```
    reluLayer();
    maxPooling2dLayer(3, 'Stride', 2);
    dropoutLayer(0.5);
    fc1;
    reluLayer();
    fc2;
    softmaxLayer();
    classificationLayer();
    ];
opts = trainingOptions('sgdm', 'InitialLearnRate', 0.0001, 'VerboseFrequency',
10 ,'MaxEpochs', maxEpochs, 'Shuffle', 'every-epoch','MiniBatchSize', 32,
'Plots', 'training-progress');
cnn = trainNetwork(imds, layers, opts);
```

Matlab function included for completeness:

```
% This function simply resizes the images to fit in AlexNet
% Copyright 2017 The MathWorks, Inc.
function I = readFunctionTrain(filename)
% Resize the images to the size required by the network.
I = imread(filename);

I = imresize(I, [256 256]);
```

Demo

Vimeo link to the demonstration of the neural network.

Distance indicator

Samples

The program is tested with approximately 50 pictures per object.

Used libraries/functions

Function	Description
imread	Read the picture
imrotate	Rotates the pictures
imcrop	Crop the picture
rgb2gray	Apply a grey filter on the image
im2bw	Convert to black & white by point processing it with automatic threshold chooser
imcomplement	Turn the black into white and vice versa
imclose	Perform the morphological closing
bwlabel	Obtain array from the picture
gen_subplots	Subplot the graphs
vislabels	Numerate the object(s)
label2rgb	Fill RGB colours in every numerated object to see the area

Matlab code

NOTE: The corresponding pictures are submitted with the document

80 KMH sign source code

```
%NOTE that one of the j has to be commented

j= imread('IMG_20191126_163725_BURST001.jpg'); %retrieve picture(far distance
80KMH sign)
%j= imread('IMG_20191126_163725_BURST040.jpg'); %retrieve picture(close distance
80KMH sign)

rotated = imrotate(j,270,'bilinear'); %rotate picture

cropped = imcrop(rotated,[420,591,1500,2700]);% x : 405 - 2300    |    y : 591 -
3400
                                              %([x min, y min, x max(whidth), y
max(height)])

grey = rgb2gray(cropped); % convert to grayscale


bw = im2bw(grey); % Convert to black & white with autmatic threshold chooser

preComp = imcomplement(bw); %imcomplement the picture

finalFilters = imclose(preComp, strel('disk', 10)); %Performing the
morphological closing

[labels,numlabels]=bwlabel(finalFilters);
```

```
coloured = label2rgb(labels); %Fill in RGB colour in every blob
imshow(coloured);

gen_subplots(rotated, cropped, grey, bw, finalFilters, labels, coloured); %
Subplot

eighty_sign_stats = regionprops (labels,'all'); %get the specifications of the
object
eighty_sign_stats(1) %acquisition specified for merely object one

formFactor = 4*pi*eighty_sign_stats(1).Area/((eighty_sign_stats(1).Perimeter)^2)
%form factor calculation for precise classification

eightsign_intensity = eighty_sign_stats(1).Area / 4000; %Distance calculation
max_distance = 100;
sign_distance = max_distance - eightsign_intensity;

if sign_distance > 0
    fprintf("The distance of the object is: %f cm", sign_distance); %Distance
indication
else
    fprintf("Unknown distance");
end
```

Traffic light source code

```
%NOTE that one of the j has to be commented

j= imread('IMG_20191126_163912_BURST003.jpg'); %retrieve picture(far distance
traffic light)

%j= imread('IMG_20191126_163912_BURST042.jpg');%retrieve picture(close distance
traffic light)

rotated = imrotate(j,270,'bilinear'); %rotate picture

cropped = imcrop(rotated,[420,591,1500,2700]);%     x : 405 - 2300    |    y :
591 - 3400
                                              %     ([x min, y min, x
max(whidth), y max(height)])

grey = rgb2gray(cropped); % convert to grayscale

bw = im2bw(grey); % convert to black & white with autmatic threshold chooser

preComp = imcomplement(bw); %imcomplement the picture

finalFilters = imclose(preComp, strel('disk', 20)); %Performing the
morphological closing

[labels,numlabels]=bwlabel(finalFilters);

coloured = label2rgb(labels); %Fill in RGB colour in every blob
imshow(coloured);

gen_subplots(rotated, cropped, grey, bw, finalFilters, labels, coloured); %
Subplot
```

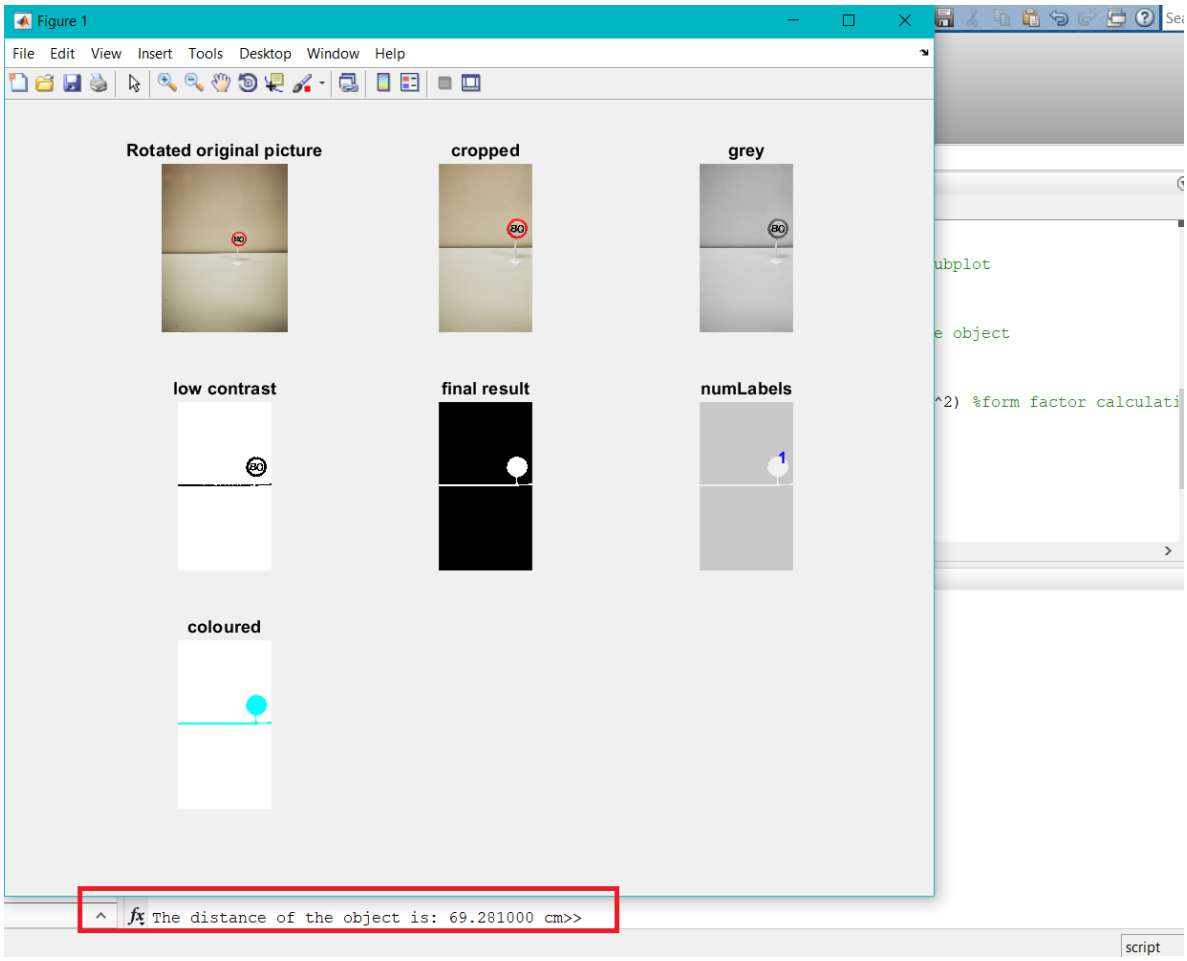
```
traficLight_stats = regionprops (labels,'all'); %get the specifications of the
object
traficLight_stats(1) %acquisition specified for merely object one

f = 4*pi*traficLight_stats(1).Area/((traficLight_stats(1).Perimeter)^2) %form
factor calculation for precise classification

light_intensity = traficLight_stats(1).Area / 12500; %Distance calculation
maxdistance = 100;
light_distance = max_distance - light_intensity;

if sign_distance > 0
    fprintf("The distance of the object is: %f cm", light_distance); %Distance
indication
else
    fprintf("Unknown distance");
end
```

Self composed functions corresponding to the source code

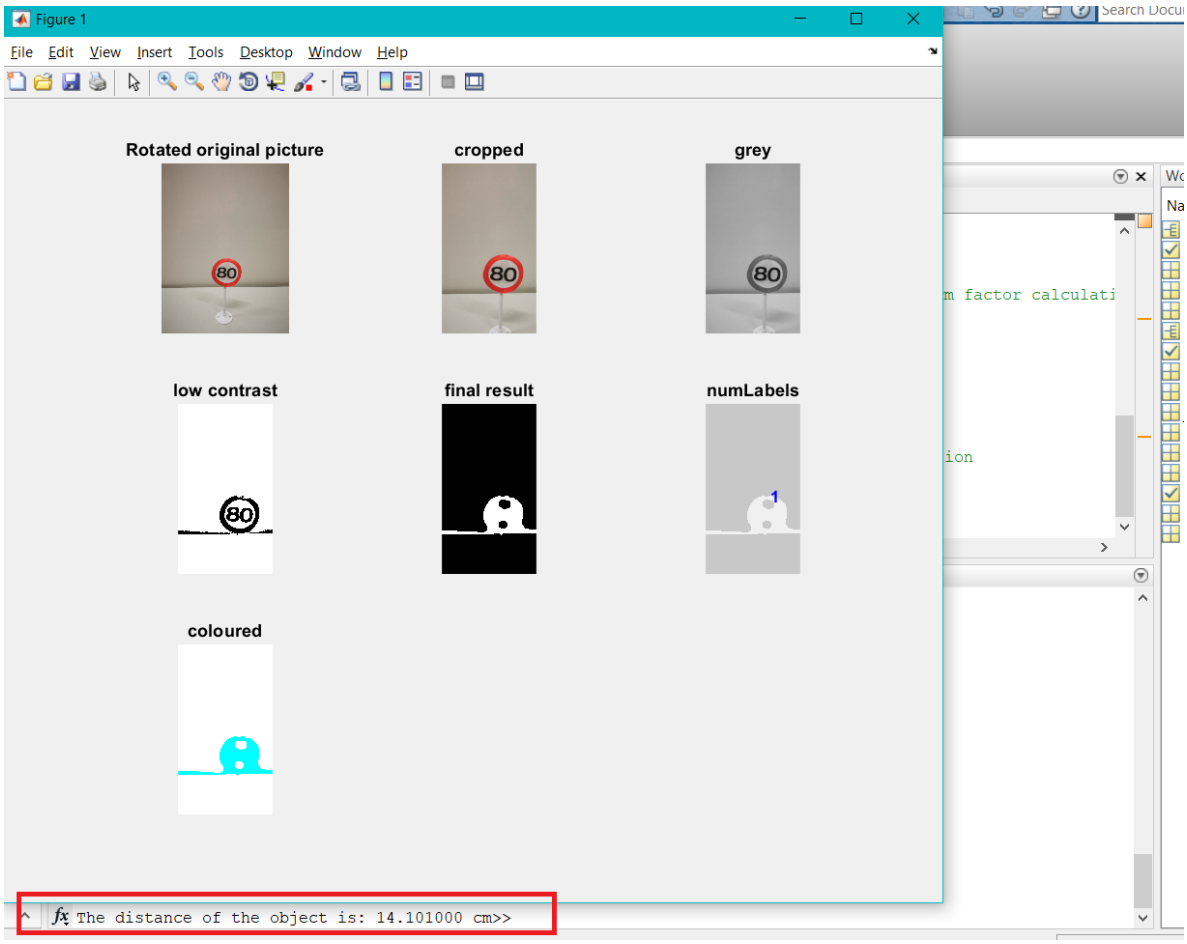
```
function gen_subplots(i1, i2, i3, i4, i5, i6, i7)
subplot(3, 3, 1);
imshow(i1);
title("Rotated original picture");

subplot(3, 3, 2);
imshow(i2);
title("cropped");

subplot(3, 3, 3);
imshow(i3);
title("grey");

subplot(3, 3, 4);
imshow(i4);
title("low contrast");

subplot(3, 3, 5);
imshow(i5);
title("final result");

subplot(3, 3, 6);
vislabels(i6);
title("numLabels");

subplot(3, 3, 7);
imshow(i7);
title("coloured");
```

Demo

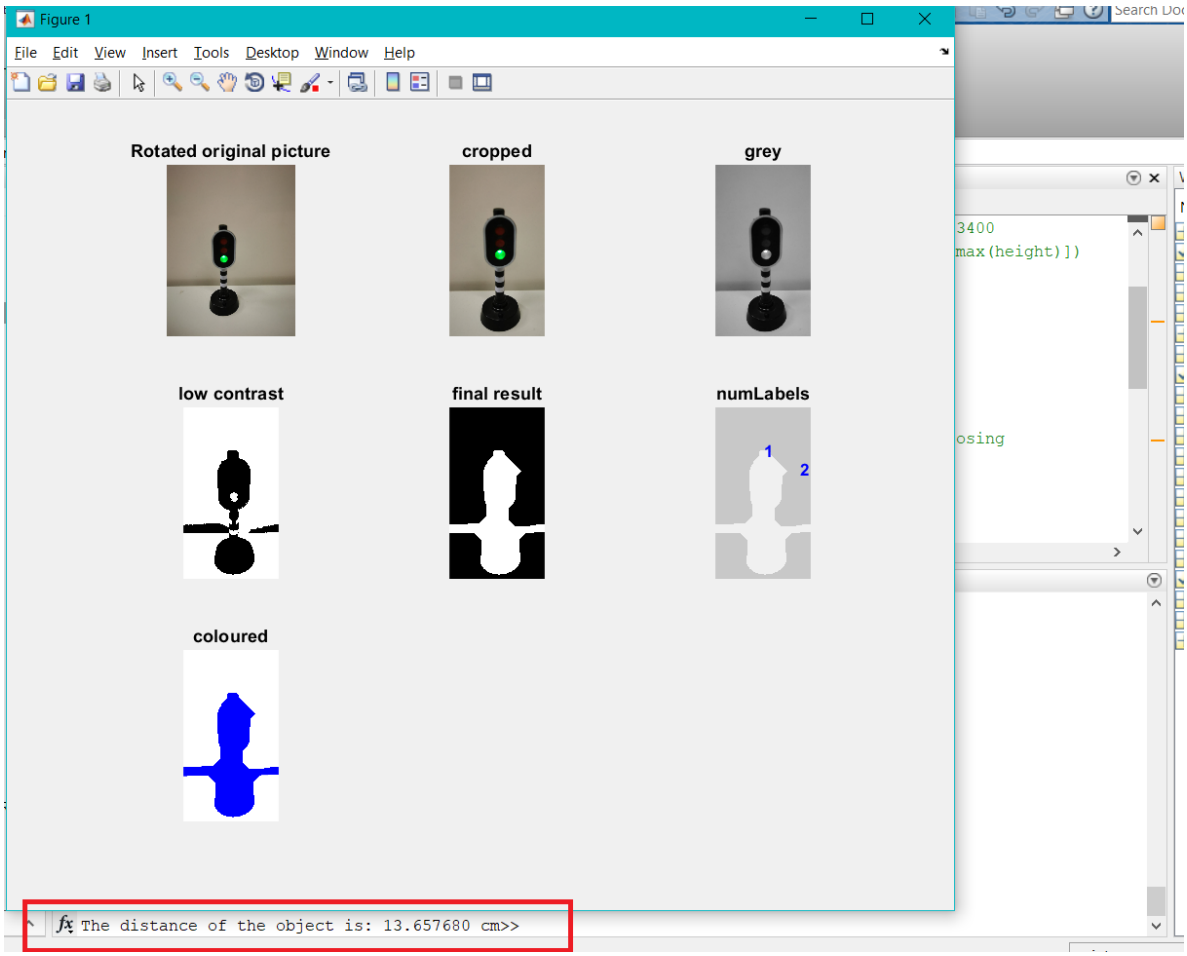
80 KMH sign that is far



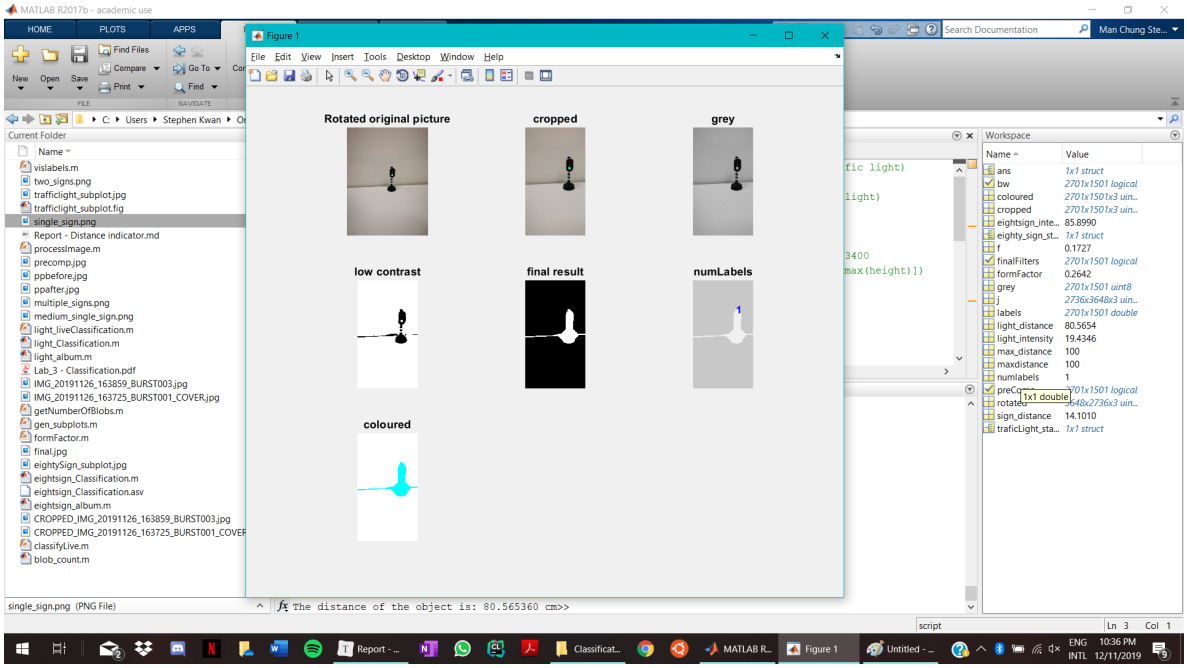
80 KMH sign that is close



traffic light that is far



traffic light that is close



References

1. Guide to Convolutional Neural Networks. H. Habibi et al.↩↩

2. WHO Global status report on road safety 2018↩

3. EU commission: 2018 road safety statistics: what is behind the figures? Url↩

4. Traffic Light Recognition Based on Binary Semantic Segmentation Network by Hyun-Koo- Kim et al. Url ↩

5. An overview of traffic sign detection and classification methods. Saadna, Y. & Behloul, A. Int J Multimed Info Retr (2017) 6: 193. Url↩

6. Deep Learning book by Ian Goodfellow and Yoshua Bengio and Aaron Courville. Url↩↩↩

7. ImageNet Classification with Deep ConvolutionalNeural Networks. Alex Krizhevsky et al. Url↩↩

8. Matlab Neural Network Toolbox documentation on visualizing activations. Url↩

9. Traffic Sign Recognition Based on Attribute-Refinement Cascaded Convolutional Neural Networks, Kaixuan Xie. Url↩