Machine Learning project

Group 10:
Pieter De Cremer
Rian Goossens
Harald De Bondt

Project Report
Ghent University
Department of Computer Science

Ghent, November 5, 2015

Contents

Introduction **1**

1 Evaluation methods **1**
1.1 Filtering out images . . . 1
1.2 Evaluation Metrics . . . 1
1.2.1 Percentage of correctness . . . 1
1.2.2 Logloss . . . 1
1.3 K-fold . . . 1

2 Image pre-processing **2**
2.1 Resizing and cropping . . . 2
2.2 Normalization of colors . . . 2

3 Feature extraction **3**
3.1 Mean channels . . . 3
3.2 Weighted angle segments . . . 3
3.3 Perceived brightness . . . 4
3.4 2D Discrete Fourier Transform: High Pass Filter . . . 5
3.5 Histogram of Oriented Gradients . . . 6

4 Linear Model **6**
4.1 Model choice . . . 6
4.2 Limitations . . . 6

5 Performance **6**
5.1 Separate Features . . . 6
5.2 Combined Features . . . 6

Conclusion **7**

Introduction

For the course "Machine Learning" we are competing in a Traffic Sign recognition competition. The goal is to classify traffic signs as accurately as possible from 81 classes. For the first part we are assigned to use a linear model. This report will discuss our current evaluation methods, the pre-processing steps we perform and features we extract. It will also briefly discuss the linear model used and set some suggestions on whet the second part of the competition will bring.

1 Evaluation methods

In this section we will discuss the various steps we took to try to accurately predict the performance of our models.

1.1 Filtering out images

We noticed that a lot of the images in the training set were pictures of the same physical traffic sign pole but from different angles. None of the images in the training set share the same pole with images from the test set, so using images from the same pole would be an inaccurate way of evaluating our methods. We tried using one image from each pole to evaluate, but when doing this we underestimate our model quite significantly. To combat this, in the future we will produce custom kfold folds so that no poles are shared between folds.

1.2 Evaluation Metrics

We use two kinds of metrics to evaluate our models.

1.2.1 Percentage of correctness

This metric simply tells us the percentage of images which were correctly classified by our model. While this is not indicative of having a good model for the competition, this did give us a more intuitive score to consider at first. We noticed however that a better percentage did not always result in better competition score and no longer use it.

1.2.2 Logloss

We then implemented the logloss metric that is used in the competition, this should give us a more accurate view on how our current model would perform on the test set if we upload it.

1.3 K-fold

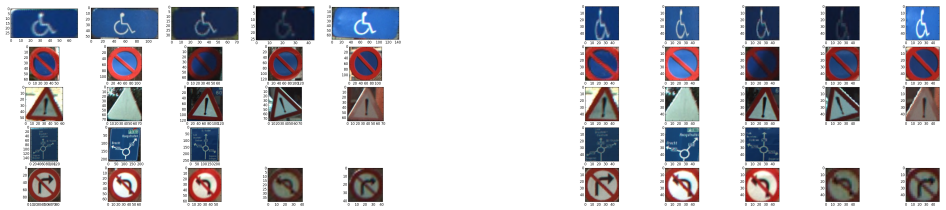
For every evaluation we use k-fold cross validation. We compare the different results to detect when we are over-fitting. For each fold we print out the current metric, and we also calculate the mean and standard deviation of our metric. We want the mean to be as good as possible while the standard deviation is as low as possible to have a strong and stable model.

2 Image pre-processing

We tried out various image pre-processing techniques to improve the quality of the images prior to extracting their various features. In this section we outline each of them and why we think they can be helpful. In the next section we always mention which pre-processing techniques we used for each particular extraction method. Note that not every pre-processing technique might be used in feature extraction, we still included them for completeness.

2.1 Resizing and cropping

We resize our images to 50x50 thumbnails. This way every image has the same amount of detail and feature extraction will always produce values with the same scales. We also cut of part of the sides of the images, this is to prevent any background details from interfering with our feature extraction.



(a) Regular images

(b) Resized and cropped images

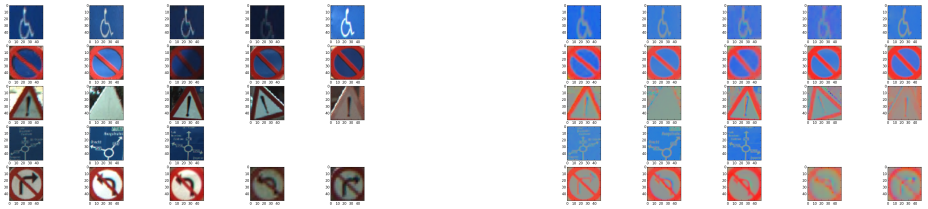
Figure 1: Effect of resizing and cropping

You can clearly see how this will benefit feature extraction. For example, in the fourth column of the third row of figure 1 part of the blue sign in the background gets cut off, which would otherwise only confuse the algorithms. The operation is not too destructive, as every traffic sign is still clearly distinguishable.

2.2 Normalization of colors

We noticed that the images have been taken in a variety of lighting conditions. This means that the red in one image does not correspond to the red in another image. A simplified way to solve this is by taking each RGB triple as a vector and then normalizing this vector so it's length is 1. A downside of this technique is that colors representing black and white become the same color after normalization.

In figure 2 you can clearly see both the advantages and disadvantages of this operation. In the first two rows especially you can see that the blues and reds from images with different lighting conditions become the same color. Even more amazing is the effect it has on dirt or partial shadows as seen in the middle image of the second row, the transformation takes these out completely. You can also see that colors that should be black become all kinds of colors like red and white, which makes them useless for any kind of extraction, a clear disadvantage of this transformation.



(a) Cropped images

(b) Normalized color images

Figure 2: Effect of normalizing colors

3 Feature extraction

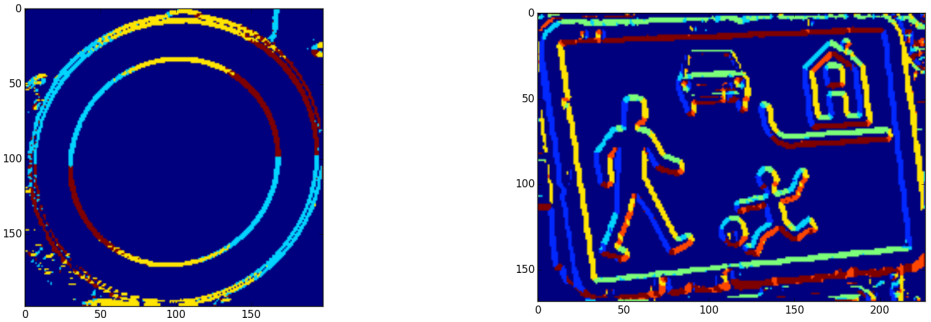
In this section we outline all of our feature extraction methods. For each method we explain the method, when possible show the efficiency in a graph for a certain example, clarify which pre-processing techniques we tried out and we provide the logloss scores when using the features in a linear model. The scores of these features might be low for some, but the final result will be a combination of several features to obtain an optimal feature set with better scores than the individual features.

3.1 Mean channels

This is very basic feature extraction where we calculate the means of every color channel. We divide the resized images into a 3x3 grid and for each section we calculate the means. By using a 3x3 grid we improve performance because we extract more data from the image, we don't use a larger grid because this causes overfitting. We also experimented with using more moments and altough they gave slightly better logloss scores, the standard deviation tripled, which we also attribute to overfitting. We tried this technique with normalized and regular images and were surprised to see the regular images gave better results. We suspect that this is because using the means will produce good, relevant means for the individual colors, which is similar to what we try to do with normalization, but normalization also suffers from losing dark and white color information.

3.2 Weighted angle segments

We calculated the angles and magnitudes at each pixel by using 2d convolve matrices to calculate the horizontal and vertical differences at a pixel. By using these as X and Y differences we calculate the angle with atan2 and the magnitude by calculating the distance. Then we divide the angles into n segments (we found 7 to be quite optimal) and for each segment we calculate the weighted sum by using magnitudes as weights. This makes it so that harder edges contribute more than flat surfaces. The idea is that certain shapes will have more prominent segments than others. In figure 3 we plotted which pixels are assigned to what angle segment, for pixels with large magnitude. You can clearly see that the 2 signs will have quite different values. We later found out that this technique is quite similar to using histogram of oriented gradients and we will investigate this technique for the final report.



(a) Angle segments of circular sign

(b) Angle segments of other sign

Figure 3: Angle segments

3.3 Perceived brightness

Since images of the same traffic sign can be taken under different lighting conditions the fluctuations in perceived brightness of images could be used to determine its shapes easier.

Before trying to extract these features we resize the images to thumbnails of size 50x50 and we trim the borders of the image to try and minimize the influence of details in the background.

To extract these values we divide the images in NxN blocks and apply the HSP color model. The result is a gray-scale image of which we normalize the brightness values with the image's brightness histogram. This should give a good indication of which areas are "darkly colored" or "brightly colored", independently of the image's lighting conditions. An example of the transformation can be seen in Figure 4.

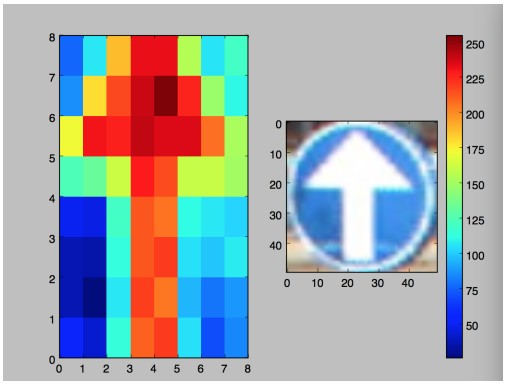


Figure 4: Effects of perceived brightness

A good interval for the NxN blocking of the image is between 5-8, dividing in too many blocks will yield more risk for overfitting. The low resolution is however not capable of distinctions signs with similar figures and form (e.g. A7B and A7A).

3.4 2D Discrete Fourier Transform: High Pass Filter

Transform As learned in previous classes we know that High Pass Filtering is an edge detection operation. It helps filter out background noise and is a good candidate for feature extraction.

Numpy has a Fast Fourier Transform package to help with these operations, `np.fft.fft2()` provides the frequency transform. Once we have the result the zero frequency component (DC component) will be at the top left corner. We perform a fourier shift of $N/2$ in each direction, with `np.fft.fftshift()`, to bring it to the center. We then mask a rectangular ring of the specified frequency and set all the other frequencies to zero. We then apply the inverse shift and inverse fourier transformation on the result. In Figure 5 we applied this transformation on several images. For each image we devided the frequency domain in 25 classes (0-24) and the images are the results for the filtering of 5 highest of these classes (20-24). It is clear that this is an edge detection technique which could result to useful features.

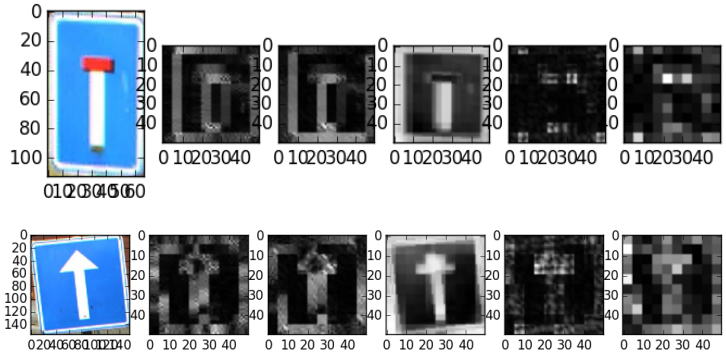


Figure 5: Effects of high frequency filtering

Features Before applying this transformation on our images we perform some pre-processing steps. The image is resized to a 50x50 pixel thumbnail. Then color normalization is performed on the thumbnails. Finally they are converted to a grayscale representation using the `color.rgb2gray()` function.

To extract features using the high pass filtering we took the approach of dividing the image into equal parts and performing the frequency pass on each part and summing the total of the frequency values over this part. From exhaustive testing we determined the best performing feature can be acquired from dividing the image into 16 parts (4x4) and the frequency domain into 25 classes. We then filter for each part of the image the frequency classes 22 and 23 and sum them, resulting into 32 values to be used as features. It comes to no surprise these classes would be the most useful, we could already see earlier in the images that they provided most info about edge detection and shape.

We sample from these values every 8th, so values 1, 8, 16 and 32 and only use these 4 as a feature. When using more features we tend to overfit, dividing the image of frequency domain more coarsely was less efficient than simply sampling from these values.

It performs not as good as some other features we have tried but it might be useful in combination with other features to improve prediction. It could also improve even further if we apply different pre-processing. This is something we still have to look into.

3.5 Histogram of Oriented Gradients

4 Linear Model

4.1 Model choice

We tried out several linear models and found linear discriminant analysis to be the most effective at reducing logloss score.

4.2 Limitations

Some traffic signs that are labeled in the same class can still differ significantly. Some examples are upwards and downwards arrows or arrows curving to the right and the left that are classified as the same image. In a multi dimensional plot of features these could form two (or more) clusters, each labeled the same. However linear models are not capable of doing this and for these classes we can have much better performance when using different models.

5 Performance

We evaluated all of our features and some combinations for 5-fold cross validation on the test data using the LDA model.

5.1 Separate Features

Below you can see the logloss scores of this experiment.

Feature	**Mean Logloss**	**Std Logloss**
Mean channels (MC)	1.81411028249	0.0409970895337
Moment channels (MoC)	1.75118315246	0.12558773033
Weighted angles (WA)	2.56789278104	0.0446447443474
Perceived brightness (PB)	1.86452911161	0.138022595034
High pass frequencies (HPF)	3.60176469027	0.0140854574591

5.2 Combined Features

We then tested the performance of several combined features. Below are the most effective combinations:

Feature	**Mean Logloss**	**Std Logloss**
MC & WA	1.6259736261	0.037701722482
MC & PB	1.43089790075	0.199693040802
MC & PB & HPF	1.3803822428	0.121196992587
MC & WA & PB	1.35202822524	0.105785298642
MC & WA & PB & HPF	1.22136703082	0.129258855416

We notice that adding the right features improves the mean logloss. On unique data (no two pictures of the same pole) we noticed that the Std also decreased for these features.

Validating the performance of these combinations can be done by executing the file `report_1_features_extraction_data.py`.

Conclusion

Our current best performing features are a combination of Mean Channels, Weighted Angles, Perceived Brightness and High Pass Frequency filtering. For these features the LDA model reaches a mean logloss score of 1.2213 and a standard deviation of 0.12925 on a 5-fold cross validation.

During the first part of the competition we have not done much research and tried to find a suitable classification ourselves. As a result we have re-invented the wheel a few times, we will try to avoid this in the future.

For future endeavors in the competition we will use a more advanced model, which will improve the performance significantly for some signs. We also have planned some more features such as Histogram of Oriented Gradients. We plan on fixing our K-fold so that the same physical poles all end in one fold. We will also do literature research for more inspiration. Finally we want to experiment to create a base image for each of the 81 classes.