Traffic Sign Recognition using CNN

Benjamin Mende
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
bmende@cs.umass.edu

Rahul Handa
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
rhanda@cs.umass.edu

Abstract

We are trying to solve the problem of real time traffic sign detection by a computer. This report examines the various techniques that have already been used and then we experiment using our own neural networks to see how various factors affect the accuracy of detecting the different signs. The sample consists of more than 50,000 images in total divided into 43 various classes. Experiments include increasing the training data by applying distortions and using more complex network architectures.

1 Introduction

Traffic sign recognition is a multi-category classification problem with unbalanced data distribution. It is a challenging real-world computer vision problem of high practical relevance. It is quite intriguing how various feature recognition techniques can be used according to how you decide to approach this problem. Vision-based systems on-board vehicles hold quite some promise in assisting future drivers at their driving task; they might even allow autonomous navigation under certain simplified conditions.

Generally speaking, road signs are designed to be easily detectable by anyone on the road. They follow clear design principles using color, shape, icons and text. Usually similar classes have a similar sign appearance for example speed limit signs are all circular along with the stop sign. Sometimes conditions such as illumination changes or weather conditions cause some signs to look more like each other. Hence even though they are designed to be easily identifiable, conditions cause even us humans to confuse them with each other. It is interesting to see how any classifier can confuse some of this signs and we use confusion matrices to study these tendencies.

While driving, viewing a sign from different angles(as the car moves closer or further away from the sign) also has various implications. Humans are capable of recognizing these changes with near perfect accuracy. We see how it affects a classifier by having the image of the same sign taken from 5 different viewing angles .

We report our findings by comparing the accuracies obtained to the already exisitng IJCNN 2011 Competition result table. Our aim is to produce results as close to the human performance accuracy(98.84%) and evaluate our experiments as we progress.

The paper is organized as follows: Section 2 presents prior work in the field. Section 3 provides details about the benchmark dataset and our approach. Section 4 explains what results we obtained and how the experiments affected the accuracy. The last sections presents experimentation details and what further work we could have done given enough time.

Computer Vision CS 670 (2016), Umass, Amherst.

2 Related Work

A vision-based traffic sign recognition system can detect signs by their colour and shape. For that reason, we see a lot of early approaches using these two methods to solve the problem. One of the methods involves detection using color or shape features to generate candidate regions where traffic sign might reside in a particular image. Fang, Chen, et al, used the relations between red-green-blue components to do this[1]. Since the RGB space is dependent on light it makes the dataset very particular. Some researchers, like Escalara and Maldonado-Bascon[2,3]used hue-saturation intensity to overcome this. This colour-space is more immune to illumination changes.

As more and more work was done in edge detection, from Gravila[4]using template based correlation method to identify the signs to using a variation of Hough Transform by Barnes and Zelinsky[5] and finally much better results using Histogram of oriented gradients[6] for pedestrian and road sign detection there has been great progress in the feature detection methods. The stage after getting the features is classification which, is of higher interest to us. The result of the previous stage is analysed by an algorithm which determines whether the detected regions are actual traffic signs or not. For example, Bahlmann, Zhu, Ramesh, Pellkofer, and Koehler (2005) classify using Bayesian generative modeling. Their classifier gives a 94% accuracy on 1700 images from 23 classes, initially trained on 4000 traffic sign images.

Moutarde, Bargeton, Herbin, and Chanussot (2007) took the classification to the next level by using a neural network on greyscale images .They use the netowrk for digit recognition and this system gave a performance of 89% for US and 90% for European speed limit signs.

Bascon, Arroyo, et al (2010) used support vector machines to achieve a 95.5% accuracy for classification. However the study was conducted only on Spanish traffic signs and the database was not made available to public. Also, the division of training and test set was done randomly but with a supervision to keep the same probabilities of each class, so we cannot assume both to be independent.

On the GTSRB dataset[7] itself there are a number of benchmark methods, for example during the first stage of the competition, there were around 20 teams which employed state of the art machine learning techniques for classification including several kinds of neural networks, support vector machines, linear discriminant analysis, subspace analysis, ensemble classifiers, slow feature analysis, kd-trees, and random forests.

Some of the best results (mostly using CNN) were obtained by team Sermanet and Lecun(2011): In their implementation, raw images are used as input and multiple feature extraction stages are trained using supervised learning. Each feature extraction stage of the network consists of a convolutional layer, a non-linear transformation layer and a spatial pooling layer. In contrast to traditional CNNs, not only the output of the last stage but of all feature extraction stages are fed into the classifier.

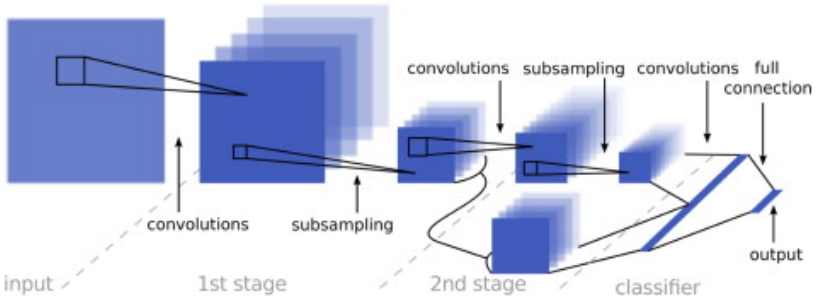

Figure 1: The Sermanet architecture, with intermediate convolutinal levels feeding into the classification layer.

The team CAOR[Zaklouta, Stanciulescu, and Hamdoun (2011)] used a random forest of 500 trees where each tree is trained on a random subset of the dataset. In each node of a tree, a small, randomly chosen subset of features is selected and the best split of the data is determined based on this selection. For classification, a sample is passed through all decision trees. The outcome of the Random Forest is a majority vote over all trees. Team CAOR used the official HOG 2 dataset.

The winning team was Team IDSIA[Ciresan, Meier, Masci, and Schmidhuber (2011)] used a committee of CNNs in form of multi-column deep neural network (MCDNN). In contrast to team Sermanet, team IDSIA only uses the central ROI containing the traffic sign and ignores the margin.They also preprocessed the data by using three image adjustments methods namely Hsitogram stretching ,histogram equalization and Contrast normalization.

3 Dataset

The dataset was created from approx. 10 h of video that were recorded while driving on different road types in Germany during daytime. The sequences were recorded in March, October and November 2010.From 144,769 labelled traffic sign images of 2416 traffic sign instances in 70 classes, the GTSRB dataset was compiled according to the following criteria:

1. Discard tracks with less than 30 images.
2. Discard classes with less than 9 tracks.
3. For the remaining tracks: If the track contains more than 30 images, equidistantly sample 30 images

The selection procedure outlined above reduced the number to 51,840 images of the 43 classes. Now,the set contains images of more than 1700 traffic sign instances. The size of the traffic signs varies between 15×15 and 222×193 pixels. The images contain 10% margin (at least 5 pixels) around the traffic sign to allow for the usage of edge detectors. The dataset is split into approx 39000 training examples and 12500 test examples randomly.

In addition to the images, the competition organizers provided pre-computed HOG and HAAR-like features to be used by those without a computer vision background. While we did not use these features in our project, some of the highest performing methods in the competition were classic machine learning techniques such as Random Forests and LDAs using them.

Figure 2: A sample of each class of sign in the dataset.

4 Our Approach

For this project, we implemented a convolutional neural network to learn features automatically and classify on them. Our network architecture was inspired by the LeNet that was used to in the MNIST handwritten digits task as well as the network architecture used in the IDSIA committee of CNNs.

4.1 Network Architecture

Convolutional Neural Networks require all inputs to be the same size, so we begin by resizing all images to 40×40 pixels, with 3 color channels. Our network consists of essentially three layers of

convolution followed by a fully connected layer and then a decision layer. In each of our convolutional layers, we use a 5×5 kernel and the ReLU activation function. This is followed by a max pooling layer with a 2×2 kernel that essentially halves the size of image for the next layer. After the final convolutional layer and max pooling, we go to a fully connected layer, or essentially a multi-layer perceptron. This is the layer that does the actual classification step, using the features we learned above to classify each image. The first fully connected layer has 1024 nodes. This feeds into the decision layer of 43 nodes. Each node represents a class of sign, and our decision as to which class we think the image belongs is by picking the node in this layer with the highest activation value.

Table 1: Network Architecture

Layer	Type	# Maps	Kernel
0	Input	3	
1	Convolutional	50	5x5
2	Max Pooling	50	2x2
3	Convolutional	100	5x5
4	Max Pooling	100	2x2
5	Convolutional	200	5x5
6	Max Pooling	200	2x2
7	Fully Connected		1024
8	Fully Connected		43

4.2 Image Pre-processing

We performed a little bit of pre-processing on our images before feeding them into our CNN. First, we re-sized all images to 40×40 pixels so that our network could handle all images. We picked this size as it was close to the median image size of our data set.

One of our first experiments with preprocessing was to grayscale the image and have only one input channel instead of three for color. Interestingly, we only saw a modest decrease in accuracy of about $2 - 5\%$. This is likely because there is not that much variation in color among signs, with almost all of them being red or blue. This suggests that most of the sign information comes from shape and gradients.

In order to provide some invariance from illumination changes, we added a contrast normalization step before feeding the image to the CNN. This basically helps put all images in a similar pixel value range. Unfortunately for us, we did not see much improvement from this step because it was expensive computationally, and we did not get as far in the same amount of time. We expect this to help if we can run the experiment for longer.

In order to effectively increase the size of our training set, we added a few randomly generated distortions to the images. These included random crops and translations, randomly flipping the image horizontally, as well as random brightness and contrast changes. Unfortunately, as with the normalization above, the added computational cost of this step prevented us from seeing much benefit.

4.3 Results

4.3.1 Hardware and Framework

Our CNN was implemented in python using the Tensorflow deep-learning network developed by Google. Tensorflow includes many utilities for backpropogation and convolutional nets, as well as some image processing tools. We were lucky to have a computer that had an NVIDIA GeForce 720 gpu and 8 Intel core i7 cpus. The gpu is the lowest model that supports CUDA, so hopefully we would do better in the future with more powerful hardware and a gpu with more memory available. With our system, we currently process about 1,500 images per second when looking to classify.

4.3.2 Results on Data

The best results we obtained with our CNN was 86.98% on the test data after about 8 hours of training and 60,000 batches of 50 images each. This is compared with about 81% after about an hour

training. With our best test results, we found an accuracy of about 95% on the training data, which suggests that we have learned almost all of the training data. However, there is still more to go before completely overfitting, so we hope that running for longer will increase our test accuracy.

While the accuracy on the overall data was 87%, there is considerable variation when looking at subsets of the data, for example our accuracy on just speed limit signs.

Table 2: Accuracy on subsets of data

Type	Accuracy
All	86.98
All Red Round	88.07
Speed Limits	84.15
Other Red Round	95.47
Restriction End	83.61
Blue Round	87.51
Triangle Danger	79.93
Other Signs	96.32

It is interesting to see what kinds of mistakes our network makes. For example, while we do ok on speed limit signs, we do very well on the other red round signs. This suggests that what we have trouble with is likely identifying which particular speed rather than the fact that it is a speed limit sign at all. The triangle signs likely do the worst because while there are many different kind of triangle signs, but not all that many examples of each one. Also, there are many shapes that could be confused with shapes appearing in other sign types.

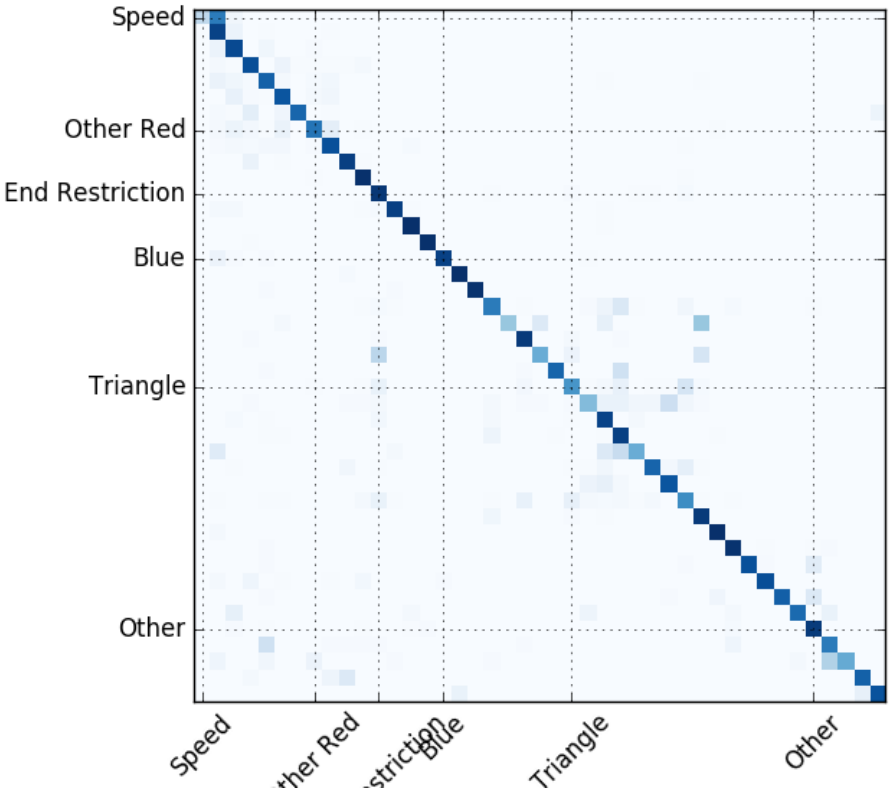

Figure 3: Confusion matrix of our CNN.

Above, we see a confusion matrix of our data with the true value as the y-axis and our guess as the x-axis. We see how most of the mistakes for speed limit signs occur with other speed limit signs, and red round signs in general are confused more for each other than other types of signs. The strangest result we see is that we seem to be confusing quite a few triangle signs for blue signs as well as with each other.

5 Conclusions and Future Work

We have built a system that can classify images of German traffic signs. While it is not as accurate as humans, it does make the same kinds of mistakes, such as misclassifying speed limit signs as other speeds. We were inspired by similar systems which did beat humans at this task, and we believe that with more training time and better hardware, we could get much closer to that level.

One extension of our network we would like to have tried would have been implementing the idea of multi-scale CNN as with Sermanet. Unfortunately, our attempts to do so met up with the memory limits of our gpu, so we would have to get better hardware or drastically reduce the size of the network in order to try this technique.

This project worked on German traffic signs. We would like to see how the trained model would work with American traffic signs, and how transfer learning could help here.

Finally, we would like in the future to attempt traffic sign detection using our classifier. Luckily, a detection data set is available from the GTSRB group with the same signs as in our classification dataset, just this time not cropped out by a human before hand. Some of the challenge here would include speeding up our classifier if our system would ever be used in real time. Currently, we can process about 1,500 images per second, and the best object proposal methods can reduce the number of regions in an image to test to about 2,000 per image, which means it would take over a second to detect and classify traffic signs, which is much too slow for this application. This could be achieved by getting better hardware. We also might want a pre classifier that determines if an image patch is a traffic sign at all before passing it on to the slower classifier that we have built here.

References

[1] Fang, C.Y.; Chen, S.W.; Fuh, C.S. Road-sign detection and tracking. IEEE Trans. Veh. Technol. 2003, 52, 1329–1341.

[2] de la Escalera, A.; Armingol, J.; Pastor, J.; Rodriguez, F. Visual sign information extraction and

[3] Maldonado-Bascon, S.; Lafuente-Arroyo, S.; Gil-Jimenez, P.; Gomez-Moreno, H.; Lopez-Ferreras, F. Road-sign detection and recognition based on support vector machines. IEEE Trans. Intell. Transp. Syst. 2007, 8, 264–278.

[4] Gavrila, D. Traffic sign recognition revisited. In Proceedings of the DAGM-Symposium, Bonn, Germany, 15–17 September 1999; pp. 86–93

[5] Barnes, N.; Zelinsky, A. Real-time radial symmetry for speed sign detection. In Proceedings of the Intelligent Vehicles Symposium, Parma, Italy, 14–17 June 2004; pp. 566–571.

[6] Overett, G.; Petersson, L.; Andersson, L.; Pettersson, N. Boosting a heterogeneous pool of fast HOG features for pedestrian and sign detection. In Proceedings of the 2009 IEEE Intelligent Vehicles Symposium, Xi'an, China, 3–5 June 2009; pp. 584–590.

[7] Human Performance, INI-RTCV , Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, August 2012, Neural Networks (32), pp. 323-332