# Enhanced Road Sign Detection


## Abstract

Automated recognition of traffic signs is an important topic in the field of Computer Vision. It has practical application primarily in the realm of self-driving/autonomous vehicles. Through the use of traffic sign detection, vehicles are capable of detecting the position, classifying the purpose of, and forming insights based on road signs such that human drivers are no longer necessary. In this paper, a detection and classification algorithm utilizing image pre-processing techniques, AdaBoost **[8]**, one-vs-all binary classification **[6][7]**, color-based classification, and bagging ensembles (for enhanced generalization) and marginal accuracy improvements **[9]** is proposed acquiring respectable accuracy on the Kaggle Road Sign Detection dataset consisting of 877 images belonging to 4 distinct classes. These images are segmented into positive and negative images based on the included annotations for each image. Negative samples are generated from acquiring image segments outside the annotation parameters, totaling 3,296 images. Each image in the dataset potentially contains more than a single positive sample; a total of 1,243 positive images were acquired. The specifics of the paper's implementation are discussed, and method for improving the algorithm's performance are proposed.


## Introduction

Traffic sign detection state-of-the-art is mostly reliant on transfer learning via convolutional neural networks. Transfer learning is the act of pre-training a neural network model on a larger dataset of images such that it is able to detect basic features such as colored blobs and lines; then the model is re-trained on a target dataset where more domain specific features are detected (traffic signs, stop signs, etc.) **[1]**. Various convolutional neural networks such as ResNet V1 101, Inception ResNet V2, and Darknet-19 have been used to acquire state-of-the-art performance on the German Traffic Sign Detection benchmark dataset **[2]**. These models are evaluated using mean average precision (mAP) and other runtime benchmarks. The average precision of these models is 99.89% **[2]**.

Due to convolutional neural networks not being within the confines of this paper's specifications, state-of-the-art implementations of other object-recognition systems are discussed. In **[3]**, superior accuracy is achieved using a series of image detections steps containing learned decision trees, AdaBoost detection, and finally a support vector classification layer. This paper published in 2016 achieved a precision and recall of 94.52% and 80.85%, respectively **[3]**. This was a markable performance increase from the previous state-of-the-art of precision and recall of 91.35% and 77.00%, respectively.

In a paper published in 2013, an AdaBoost binary classifier followed by a circular Hough transform was used to achieve a performance of 95% on images found in differing light conditions: sunny, cloudy, foggy, etc. The traffic signs were broken up into two different groups: warning and prohibitory signs with recognitions of 98.4% and 97%, respectively **[4]**. The methods employed in this paper include color segmentation, edge detection, shape voting, and support vector machine classifiers. The color segmentation was employed using AdaBoost **[4]**.

Dating back to 2010, Viola-Jones was proposed as an efficient method to reliably perform traffic sign object-detection. The paper outlines the 3 qualities a successful tracking system should have:

1. detection of a sign in an image,

2. proper classification of the sign, and
3. sign tracking through time.

Viola-Jones is described as a cascade of boosted Haar features where boosted Haar features, also known as AdaBoost, is a series of weak classifiers (single Haar features) being combined to create a strong classifier **[5]**. The advantage of using Haar features in is that they can be calculated in constant time in tandem with integral images; and because of the cascading method employed in AdaBoost, many of the Haar features are not used. The Viola-Jones detector works by sliding a detection window across and image in the horizontal and vertical direction. The scale factor of the window and the granularity of the sliding window movement all have a linear relationship on the runtime of the algorithm. The size of the detection window and the movement granularity all effect the classification accuracy of the algorithm with a smaller detection window and more granularity improving the classification accuracy at the cost of efficiency **[5]**.

## Data Collection

The Kaggle Road Sign Detection was utilized in the experimental section of this paper. This dataset was chosen due to its relatively small size and ease of use. 1,243 positive images were extracted from the dataset, and the pseudo-code for extraction is outlined below:

---

***Function CreatePositiveImages():***
*for img, xml in **zip**(list of images, list of xml files):*
*for object in xml:*
    *object -> xmin*
    *object -> ymin*
    *object -> xmax*
    *object -> ymax*
    *img = img[ymin : ymax, xmin : xmax]*
    *save img*

---

The positive images are then labeled and inserted into another folder by on their 4 distinct classes: crosswalk, speed limit, stop sign, and traffic light. Sample positive images are shown below:


**Figure 1. Sample Positive Images of 4 Distinct Classes**

A total of 3,427 negative images are generated from the image dataset by taking corners outside the bounding box of xmin, xmax, ymin, and ymax. Then extraneous images containing signs are manually identified and selected from the output set and deleted; the resulting negative dataset contains 3,296 images. The algorithm used to generate these images is described below:

---

***Function CreateNegativeImages():***
*for img, xml in **zip**(list of images, list of xml files):*
*position_dict = {xmin = [], ymin = [], xmax = [], ymax = []}*
*for object in xml:*
*object -> xmin*
*position_dict['xmin'].append -> xmin*
*object -> ymin*
*position_dict['ymin'].append -> ymin*
*object -> xmax*
*position_dict['xmax'].append -> xmax*
*object -> ymax*
*position_dict['ymax'].append -> ymax*

*xmin = min(position_dict['xmin'])*
*ymin = min(position_dict['ymin'])*
*xmax = max(position_dict['xmax'])*
*ymax = max(position_dict['ymax'])*

*img_1 = img[0 : ymin, 0 : xmin]*
*save img_1*
*img_2 = img[0 : ymin, xmax : ]*
*save img_2*
*img_3 = img[ymax : , 0 : xmin]*
*save img_3*
*img_4 = img[ymax : , xmax : ]*
*save img_4*

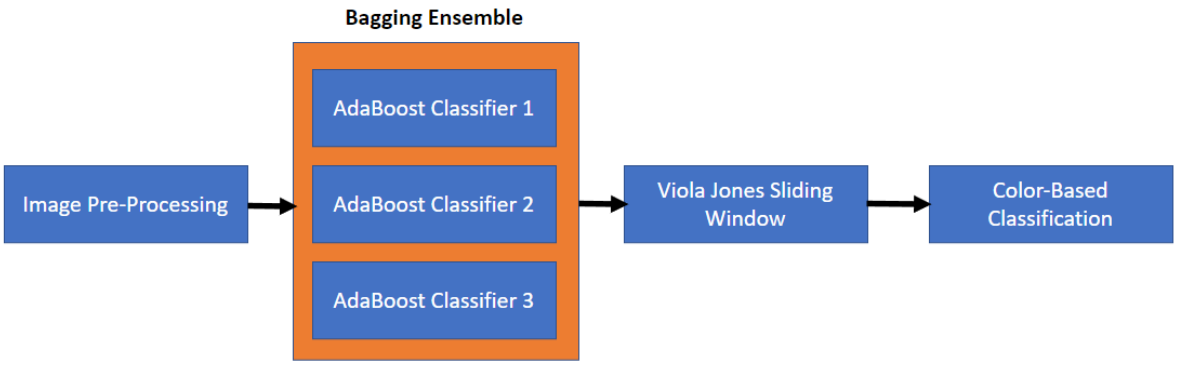
---

**Figure 2. Sample Negative Images**

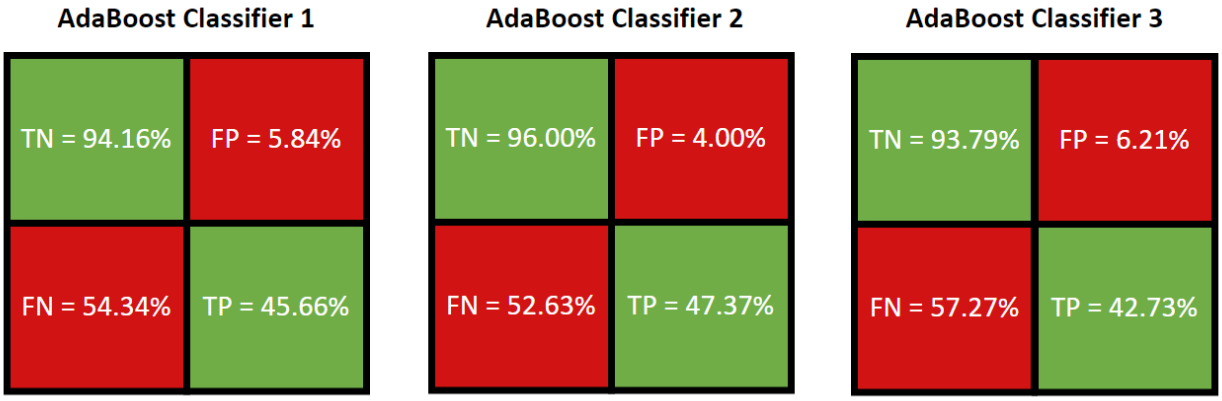
## Experimental Procedure

The experimental procedure was broken up into 4 disparate parts. Image pre-processing was performed as described in the Section "Data Collection". The provided image annotation was used to get positive images containing only the signs, and then negative images were created from parts of the original images outside the positive image area. The representation of negative images is increased such that false positive rate is minimized (this is more important than reducing the false negative rate due to the sliding window being used). During the start of the training phase, these images are converted to greyscale and then resized to (24, 24) images. Finally, they are flattened such that they can be more efficiently used for training in the AdaBoost classifiers. These images are split into 90% training and 10% test set, respectively.

Next, using this newly created binary dataset, three AdaBoost classifier were trained for 100 iterations. The AdaBoost classifier consists of cascaded "tests" which perform image thresholding (slicing the image with a line). If an image passes each test, then it is identified as a positive image.




Upon training the AdaBoost classifiers, the below confusion matrices were achieved with overall accuracies of 80.61%, 83.04%, and 79.46%, respectively. Special attention was placed to reduce the ratio of the true positive rate to the false negative rate. Because the sliding window algorithm will classify a section of any given image 50 * 50 = 2500 times, there are plenty of opportunities for the image to correctly classify an image, but also plenty of opportunity for it to misclassify negative images as positive. By maximizing the true positive to false negative rate, k-means clustering can then be utilized reliably in the image. To maximize this ratio, the representation of the negative images was selectively duplicated (by a factor of 2). This weighted negative images more highly in the AdaBoost classifiers, thus resulting in the confusion matrices below:



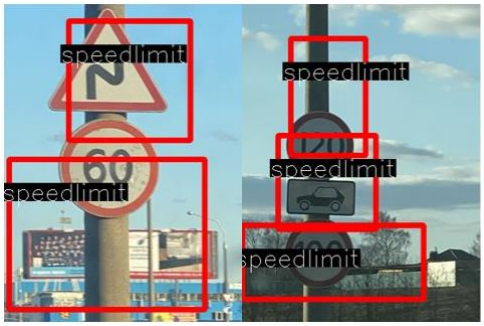

Once each classifier was trained, they were then combined into a majority-vote classifier, otherwise known as a bagging ensemble. This served to increase the generalization of the model such that overfitting was minimized. It should be noted that the boosting ensemble functions as a light intensity and edge detection algorithm. This means that color has minimal impact on the classification output. By taking into account more advanced features (such as those a convolutional neural network would seek), the classification accuracy of this model could be increased to nearly 99% accuracy as opposed to the current 80% accuracy found on the base dataset. Implementing a PCA or SVM classifier in additional to then AdaBoost classifier could provide significant performance improvements for a bagging ensemble.

Based on the results of hundreds of runs of the sliding window algorithm, it normally takes the algorithm 2-3 seconds to classify 2500 sub-images. By increasing the classification accuracy, the number of classifications per image could be greatly reduced by a factor of 100x execution time. This would make it possible for the algorithm to run at 30-50 fps, making it applicable to real-time video analysis.

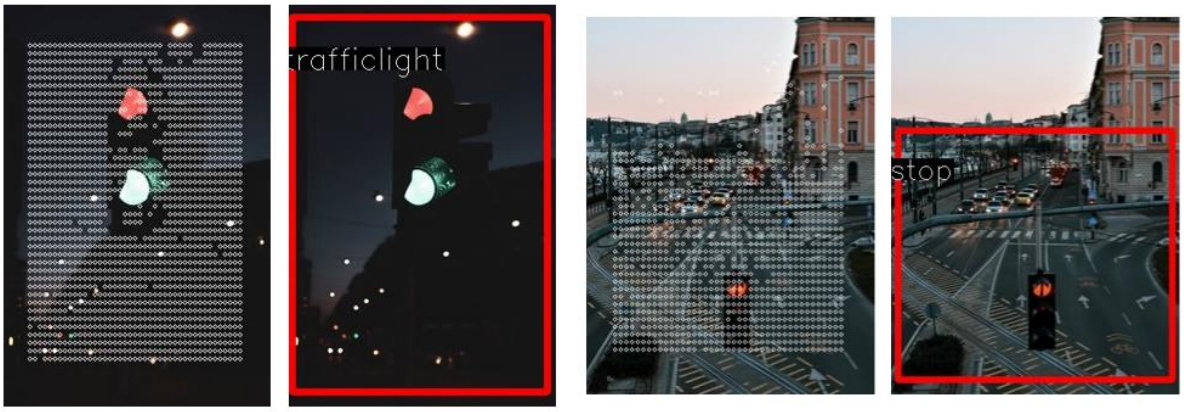
Additionally, parallel distributed architecture could be utilized to further reduce execution time by a factor of the number of additional threads.


After performing the AdaBoost classification via the sliding window, the positive prediction coordinates are added to a dataframe in which k-means clustering is performed on them. A significant weakness of this approach is two-fold: poor classification accuracy makes clusters difficult to distinguish in images where signs can be found within close proximity to one another; additionally, k-means clustering requires an input for the number of clusters, so a clustering algorithm such as BIRCH which doesn't require this input parameter would be more appropriate for this problem.

Finally, a bounding box is generated by performing a row-wise and column-wise z-score analysis on each resulting cluster. The minimums and maximums are then found from the z-score analysis, and these are used as bounds for the bounding box. From there, the sub-images within each bounding box are sent through a final color classifier which uses simple color-based classification logic on the image. This color-based logic is not very accurate and only works about 40% of the time.




A point of contention in the segmentation and classification algorithm is that the AdaBoost ensemble functions as a light intensity and edge detection algorithm. If the sign has a similar light intensity to the surrounding background or the surrounding background has polygonal edges, the classification algorithm will fail. This effect can be seen in the images below:




Also, it was difficult to tweak the color logic such that images are correctly classified. One solution that was attempted to remediate this issue was to implement several cascaded one-vs-all binary AdaBoost classifiers. These cascaded one-vs-all binary classifiers would form a multi-class classifier; however, the AdaBoost classifiers used in each stage of the cascade were not effective in accurately classifying these images.

Another such option was to use a decision tree for color-based classification, but there was not enough time to implement this algorithm. Hough transforms could be used within the resulting bounding boxes of each image as a final segmentation layer. This would be the only way to efficiently use Hough Transforms as they are computationally expensive, but the number of bounding boxes in the resulting output image is rarely greater than three.

In future work under the same restrictions as the assignment instructions, a one-for-all cascaded multi-class classifier would be implemented using cascaded ensembles of AdaBoost, SVM, and PCA. The sliding window procedure would have reduced granularity to achieve a potential 100x performance improvement, and a hierarchal clustering algorithm un-reliant on an n-cluster argument would be utilized. Finally, a Hough transform would be used on the final bounding box output to reduce the bounding box region and provide a better image for the one-vs-all multi-classifier.

# Citations


[1] Zhuang, Fuzhen, et al. “A Comprehensive Survey on Transfer Learning.” ArXiv:1911.02685 [Cs, Stat], June 2020. arXiv.org, http://arxiv.org/abs/1911.02685.

[2] Arcos-García, Álvaro, et al. “Evaluation of Deep Neural Networks for Traffic Sign Detection Systems.” Neurocomputing, vol. 316, Nov. 2018, pp. 332–44. ScienceDirect, https://doi.org/10.1016/j.neucom.2018.08.009.

[3] T. Chen and S. Lu, "Accurate and Efficient Traffic Sign Detection Using Discriminative AdaBoost and Support Vector Regression," in IEEE Transactions on Vehicular Technology, vol. 65, no. 6, pp. 4006-4015, June 2016, doi: 10.1109/TVT.2015.2500275.

[4] Fleyeh, Hasan & Biswas, R. & Davami, Erfan. (2013). Traffic sign detection based on AdaBoost color segmentation and SVM classification. 2005-2010. 10.1109/EUROCON.2013.6625255.

[5] A. Martinović, G. Glavaš, M. Juribašić, D. Sutić and Z. Kalafatić, "Real-time detection and recognition of traffic signs," The 33rd International Convention MIPRO, 2010, pp. 760-765.

[6] Rifkin, Ryan & Klautau, Aldebaro. (2004). In Defense of One-Vs-All Classification. Journal of Machine Learning Research. 5. 101-141.

[7] Ng, Selina, et al. “A One-Versus-All Class Binarization Strategy for Bearing Diagnostics of Concurrent Defects.” Sensors, vol. 14, no. 1, Jan. 2014, pp. 1295–321. DOI.org (Crossref), https://doi.org/10.3390/s140101295.

[8] Viola, P., and M. Jones. “Rapid Object Detection Using a Boosted Cascade of Simple Features.” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, IEEE Comput. Soc, 2001, p. I-511-I–518. DOI.org (Crossref), https://doi.org/10.1109/CVPR.2001.990517.

[9] Bühlmann, Peter. (2012). Bagging, Boosting and Ensemble Methods. Handbook of Computational Statistics. 10.1007/978-3-642-21551-3_33.