Traffic Sign Detection based on Convolutional Neural Networks

Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, Xiaolin Hu

Abstract— **We propose an approach for traffic sign detection based on Convolutional Neural Networks (CNN). We first transform the original image into the gray scale image by using support vector machines, then use convolutional neural networks with fixed and learnable layers for detection and recognition. The fixed layer can reduce the amount of interest areas to detect, and crop the boundaries very close to the borders of traffic signs. The learnable layers can increase the accuracy of detection significantly. Besides, we use bootstrap methods to improve the accuracy and avoid overfitting problem. In the German Traffic Sign Detection Benchmark, we obtained competitive results, with an area under the precision-recall curve(AUC) of** 99.73% **in the category "Danger", and an AUC of** 97.62% **in the category "Mandatory".**

I. Introduction

Driver assistance systems (DAS) have received more and more attentions from both academy and industry areas. Among various functions of DAS, the traffic sign detection has become one of the most important modules since it provides alerts for the drivers to relieve the pressure of driving. Detection of traffic signs has been a popular problem in intelligent vehicles since the middle of 1990s, and various methods have been proposed by researchers.

Because traffic signs usually have specific colors, the color-based methods are commonly used. These methods often use a threshold to separate traffic signs from background [1]. Some researchers use HSI color space instead of RGB and has achieved good performance [2]. A novel color space Eigen color proposed based on Karhunen-Loeve (KL), is used for traffic sign detection [3]. The main disadvantage of these color-based methods is that it is difficult to set the value of threshold because the color information is not invariant in real-world environment with different lightening conditions.

Methods based on shape of the traffic signs, have also been widely used. In [4] a method is proposed using smoothness and Laplacian filter to detect round signs. In [5] a method designed to detect triangle signs based on gradient and orientation information is proposed. In [6] a detection algorithm by using Hough transform is introduced. In order to speed up the detect algorithm, [7][8] use a fast detection method based on the symmetry on Radial direction of triangle, square, diamond, octagon and round signs. Most of the methods above rely on gradient features, which are really sensitive to noise. Because color information and shape information are both useful to traffic sign detection, it is natural to combine these two kinds of features. In [9] images are segmented in HSI color space, and template matching techniques are then used to find traffic signs.

Although the detection of traffic signs has been studied for years, there still exist many challenges. For example, the background clutter may introduce strong disturbances. In addition, the color of traffic sign is very sensitive to lighting conditions (sun, shadow), weather (sunny, rain, snow) and time (morning, noon, night), etc. Last but not least, the partial occlusion dramatically affects the detection performance.

Recently, Convolutional Neural Network has been adopted in object recognition for its high accuracy [10] [11] [12] [13]. In [10], a multi-layer convolutional networks is proposed to boost traffic sign recognition, using a combination of supervised and unsupervised learning. This model can learn multi stages of invariant features of image, with each layer containing a filter bank layer, a non-linear transform layer, and a spatial feature pooling layer. Feeding the responses of both two convolutional layers to the classifier can achieve an accuracy of recognition as high as 99.17%.

Inspired by the excellence of traffic sign recognition using Convolutional Neural Network (CNN), we proposed a method based on CNN, using fixed and learnable filters to detect traffic signs on scene images. To accelerate the detection speed, color information is used to choose the areas we are interested in. Besides, the responses of images convolving with fixed filters we defined before training are fed to learnable filters. The results of the two learnable filter layers are branched to a 2-layer nonlinear classifier separately. The learnable filter layers and the classifier are trained in a supervised way. The fixed filter layer can decrease the number of windows we need to analyze. We obtained good results in the competition of German Traffic Sign Detection Benchmark [14], with an AUC of 99.73% in the category "danger", and an AUC of 97.62% in the category "mandatory".

The specific contributions of this paper are as follows: we used a convolutional neural network combined with fixed and learnable filters to detect traffic signs(see section 3), and obtain competitive results in the category of "danger" and "mandatory". Besides, bootstrap method is adopted to learn from the misclassified training samples to decrease the rate of false positives and false negatives(see section 3 part C). In addition, the data augmentation of enlarging positive samples by rotation, translate, scale transformation

The authors are with the State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. (e-mail: wuyhwinifred@gmail.com, lyl890808@gmail.com, lijianmin@mail.tsinghua.edu.cn, hpliu@mail.tsinghua.edu.cn, xlhu@mail.tsinghua.edu.cn)

This work was supported by the National Basic Research Program (973 Program) of China (Grant Nos. 2013CB329403 and 2012CB316301), National Natural Science Foundation of China (Grant Nos. 61273023 and 91120011) and Beijing Natural Science Foundation (Grant No. 4132046).

978-1-4673-6129-3/13/$31.00 ©2013 IEEE

can prevent overfitting problem(see section 4 part A).

II. The Dataset

The task of GTSDB [14] is to detect 3 types of traffic signs: “Prohibitory”, “Mandatory” and “Danger”. Our work mainly focused on the “Mandatory” and “Danger” category. “Mandatory” signs are circle, with white arrows in the middle of blue background. “Danger” signs are triangle with white background and red borders. Examples of the two categories are shown in Figure 1.

(a) Mandatory Samples

(b) Danger Samples

Fig. 1. Samples

The training dataset contains 600 scene images (1360 x 800 pixels) and the traffic signs cropped in these images. The size of the traffic signs ranges from 16 x 16 to 128 x 128. The testing dataset contains 300 scene images (1360 x 800 pixels) with zero to six traffic signs occurred in each image.

III. The Architecture

In this paper, we use color information and CNN to detect traffic signs. A simple flow chart shows the whole process of our algorithm in Figure 2 and Figure 3.

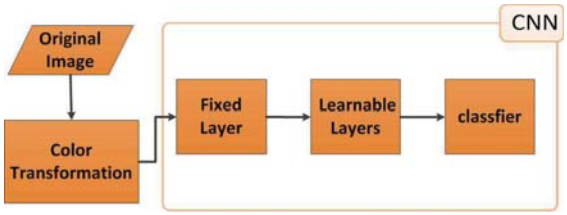

Fig. 2. An illustration of the processing steps of our algorithm. To detect each kind of traffic signs from background or other traffic signs, first transform RGB images to gray scale images using SVM, then feed the results to CNN. The fixed layer detect ROIs, and learnable layers extract distinguishing features for the classifier to find out traffic signs of the target group.

To accelerate the detection process, we use color information to do some data preprocessing for testing dataset. It converts the original scene image into a gray scale image. Traditional color transformation methods use color spaces like HSV or Lab to deal with the difficulty introduced by color deviation due to various lighting conditions, different

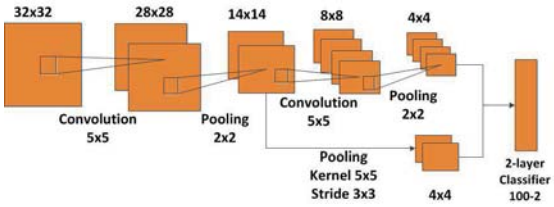

Fig. 3. Architecture of learnable layers. The outputs of two learnable layers are fed to the classifier separately. The parameters of learnable layers and the classifier are trained simultaneously in supervised way. The 2-layer classifier is fully connected with 100 neurons in the first layer and 2 neurons in the second layer.

weather conditions or natural fade, and static or dynamic thresholds are used to segment the whole image. But we learn the threshold instead of using manually fixed or dynamic threshold to establish the mapping between RGB value and gray scale value (intensity value). The color transformation based on Support Vector Machine in the preprocess step can avoid the sensitivity to color differences in different lightening conditions [15]. At first we extract pixels from training data, and classify them into positive pixels and negative pixels with Support Vector Machine (SVM). For example, in the category “mandatory”, positive pixels are those blue ones inside mandatory signs, and negative pixels are non-blue pixels. Then we train a classifier [16], and use the classifier’s offset as the map between RGB and gray scale value.

Convolutional Neural Networks (CNN) are hierarchical neural networks with multiple layers (see Figure 2). The first layer convolves the gray scale image obtained in the color transform step with fixed filters and compares the correlation coefficient value with threshold to detect areas possibly containing traffic signs. The learnable layers extract multi-scale features for classifier to judge whether it is a traffic sign in the required category or not.

A. Fixed Layer

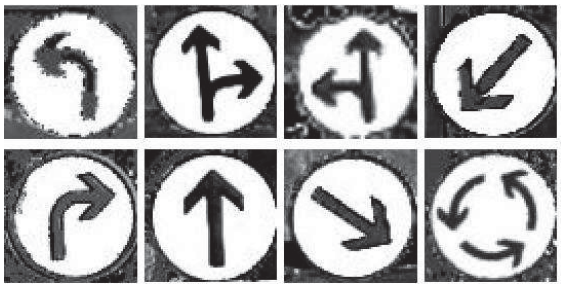

Fig. 4. Mandatory Filter

During the processing of fixed layer, correlation coefficient values are used to describe the degree of matching between a filter and a test patch. A higher value indicates that the region is more likely to contain a traffic sign.

Filters we use in mandatory signs are shown in Figure 4. The shape of this kind of signs is not sensitive to rotation

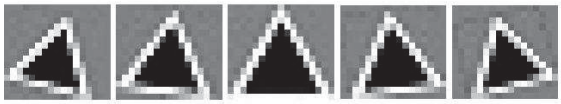

Fig. 5. Danger Filter

because they are circular, so we only use one filter for each class of mandatory signs. Danger signs' filters are shown in Figure 5. The reason why we use five filters to match one shape is that rotation has a significant effect on the shape of this category of signs.

Because the size of traffic signs in the images ranges from 16 x 16 to 128 x 128, multi-scale matching is required. In our experiment, we choose 1.05 as our filter scale-rate because this value can achieve satisfactory result in the acceptable computing time. For every specific filter in each scale, we extract every patch which has a correlation coefficient value larger than the threshold. By changing the threshold, we can generate a group of regions of interest (ROI).

Since this algorithm does not check the overlapping patches, there may exist a lot of ROIs around one traffic sign. In order to solve this problem, a simple algorithm is introduced to merge ROIs. For each image, the ROIs are sorted by correlation coefficient value in descending order, then the one with highest value is chosen as a positive region and all regions near this region are deleted. Repeat this step until no regions left. In this paper, nearby areas are regions whose distances of top-left points are less than 16 pixels in both x-axis and y-axis(16 is the minimum size of traffic signs in the data set).

The whole process of ROI extraction is shown in Figure 6.

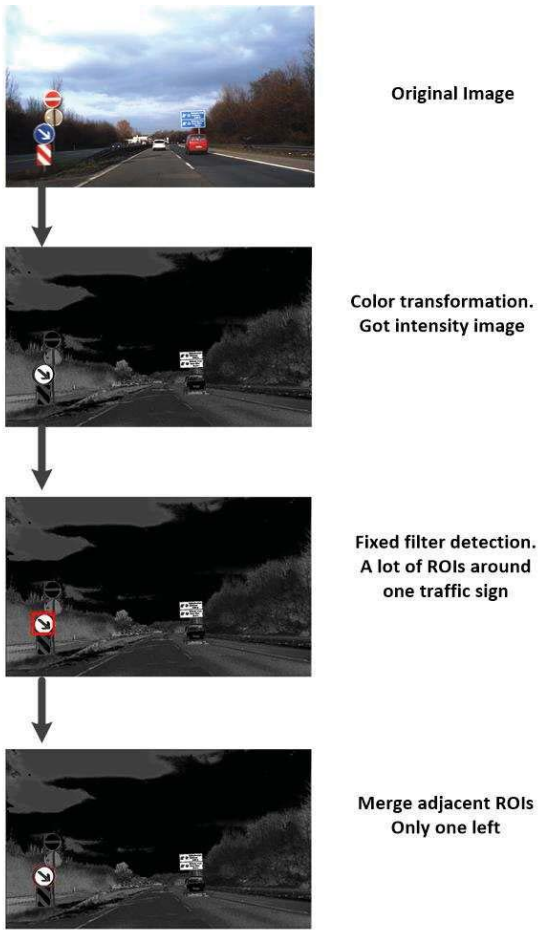

Fig. 6. Process of ROI extraction

B. Learnable Layers

The learnable layers constitute the traditional convolutional network. The choice of architecture affects the efficiency of CNN significantly. Each learnable filter layer contains a filter bank layer which convolves different filters with images, a non-linear transform layer using $|\tanh|$, a pooling layer, and a local norm layer. We discussed several architectures to choose an appropriate one.

1) Filter Bank Layer: In [10] different choices of CNN architecture are compared. A multi-stage CNN feeding the features extracted in both of the two stages into the classifier outperforms the CNN which also has two stages but only uses the second stage's responses to classify. Since the features extracted in the first stage are more local and detailed while the ones from second stage are relatively more global, feeding responses of both of the two stages can increase the accuracy. In our experiment, we adopted the multi-stage CNN feeding features of both two stages into the classifier and compared different multi-stage CNN architectures, like 6-16, 16-512, 108-200, where the left number is the number of filters extracted in the first layer while right is the number of the second layer (see Figure 3).

2) L_P pooling: The spatial pooling layer is often used after the feature extraction to summarize the joint distribution of the nearby pixels. In [17], pooling in a local region boosts invariance with little shift and small noise of the region. The pooling method processes the input image as:

$$O = (\sum\sum I(i,j)^P \times G(i,j))^{1/P} \quad (1)$$

Where I stands for the input image, G stands for the gaussian kernel. The choice of P number varies from 1 to $p \to \infty$. When $P = 1$, it is average pooling; when $p \to \infty$, it's maximum pooling. During the experiment, we set $P = 4$.

3) Normalization: In [18], a local normalization method is proposed, which can be divided as subtractive normalization and divisive normalization. It can decrease the relevance of nearby pixels thus boosts the contract of images with noise. The subtractive normalization computes the output of pixel x_{ijk} as following: $v_{ijk} = x_{ijk} - \sum_{ipq} w_{pq} \cdot x_{i,j+p,k+q}$ where j,k is the x-coordinate and y-coordinate of pixels; i is the filter number; p,q is the width and height of kernel, w is the gaussian weight of the kernel subject to $\sum_{pq} w_{pq} = 1$.

The divisive normalization computes the output of x_{ijk} as following: $y_{ijk} = v_{ijk} / \max(c, \sigma_{jk})$ where $\sigma_{jk} = (\sum_{ipq} w_{pq} \cdot v^2_{i,j+p,k+q})^{1/2}$, $c = mean(\delta_{jk}), \forall j, k$

C. Bootstrap

Bootstrap is an efficient method to improve the performances of classifiers. Adding samples wrongly classified in the validation set to training set by bootstrap not only automatically includes traffic signs difficult to detect due to occlusion, exposure, spotted or other circumstances, but also enables the classifier to distinguish some patches which are similar to a certain category of traffic signs but are not. In the detection of category "mandatory", we add a bootstrap step during training. We detect ROI in all of 600 images of training data, use ROI as a test on the training data, and extract the patches which are wrongly classified compared to ground truth. We jitter the patches and add them to the training data like below:

1) 1-class wrongly classified as 0-class: perturbed like other 1-class training data(see section 4 part A), with 30 jitters of each patch;
2) 0-class wrongly classified as 1-class: perturbed like 1-class, with 7 jitters of each patch.

After training for a second time, both the false positive and false negative rate decreased more than 1% in the validation set. In the test phase, the classifier training with bootstrap improves the AUC from 93.81% to 97.62%, which shows that bootstrap is helpful to reduce the misjudgement.

IV. Experiment

In the experiment, we used a system as following:

1) CPU: Intel(R)Xeon(R) CPU E5620 @2.40GHz x 2
2) memory: 32G DDR3

The learnable layers of CNN was implemented using the EBLearn C++ open-source package [19].

A. Data Preparation

The training data provided on the benchmark contains four categories: "Prohibitive", "Danger", "Mandatory" and Other traffic signs, and we only need to distinguish each category from others. For each category, the training data is divided into two classes, the class that contains traffic sign labeled 1, and the class that contains no traffic sign labeled 0. During training phase, the 1-class mainly contains the patches provided in the TrainIJCNN2013 grouped under the category and perturbed in position ($[-4, 4]$ pixels, step 2), in scale ($[.9, 1.1]$, step .1), in rotation($[-15, 15]$, step 6). Each patch has 30 randomly chosen jitters. Jittered patches can increase the robustness of classification in case of different views of point, and different alignments of bounding boxes. The 0-class data contains the patches provided grouped under other 3 categories as well as small patches randomly chose from the 600 scene images. From each image, we randomly chose 15 patches, with random locations and random sizes ranging from 16 x 16 to 128 x 128 (not include or overlap traffic signs).

In order to make better use of training data, we assigned the proportion of training data and validation to be $2:1$. To test the efficiency of different architectures, we extracted ROIs in all 600 scene images for training and compared the results with ground truth to get labels of ROIs. These ROIs were used for test on the training data. The data size of training, validation, and test on the data of TrainIJCNN2013 are listed, each column contains the size of 0-class and 1-class (see Table I).

TABLE I
DATA SIZE ON THE TRAINING DATA

Category	Training(0:1)	Validation(0:1)	Training Test(0:1)
Danger	6400:2840	3200:1420	890:1849
Mandatory	6093:2080	3046:1040	8656:1640

Since we need to convolve the filters with patches in the same size (need not be in the same size with filters), we resized the training data (ROIs) to 32 x 32 and converted the images from RGB to YUV space. Then we extracted the data of Y channel to train our model, and discarded the U and V channels. In other words, we only used gray-scale information, because we have already used color information during the ROI extract phase(see section 3).

B. Experiments on the Training Set

We generated 25 groups of ROI using the method of color transform and fixed layer. Each group of ROI was generated in a specific threshold on the correlation coefficient values. In our experiment, the value ranges from 0.41 to 0.65 step by 0.1. The sizes of regions are between 16 and 128.

To compare the performances of different architecture, we used three architectures: 6-16, 16-512, 108-200 under the category of "Mandatory" (see Table II), "Danger" (see Table III). FP stands for false positive, FN stands for false negative. The test set is the ROIs extracted in the TrainIJCNN2013 (see Data Preparation).

TABLE II
COMPARE OF DIFFERENT ARCHITECTURES, CATEGORY MANDATORY

architecture	No. of parameters	FP	FN
6-16	40406	9.12662%	10.2439%
16-512	934382	6.86229%	8.84146%
108-200	1290326	5.27957%	9.26829%

TABLE III
COMPARE OF DIFFERENT ARCHITECTURES, CATEGORY DANGER

architecture	No. of parameters	FP	FN
6-16	40406	11.236%	6.11141%
16-512	934382	10.3371%	5.40833%
108-200	1290326	9.55056%	6.27366%

We can see 108-200 outperformed than other architectures. We suggest that more features extracted in the first stage can learn more local features and provide more randomly chosen features for the second stage to choose. However, with more complicated architecture to test, more time would be spent in training the model and detecting one image.

The training time of learnable layers is increased with the number of parameters to learn. We use the misjudgement of validation to determine the training iterations. Figure 7 shows

the misjudgement on validation datasets of different architectures. Generally, simple architectures need more iterations to achieve the same validation error. 108-200 can achieve the lowest validation error in the given time. However, the training time of each epoch has to be considered. 6-16 can finish one epoch in 2 minutes, while 16-512 in 16 minutes, and 108-200 in 1h 15m.

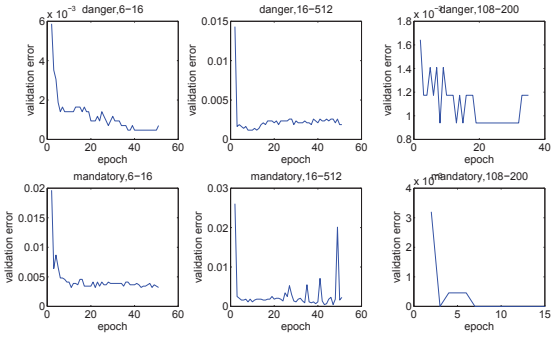

Fig. 7. Influence of different architectures of learnable layers on the validation errors through training. Top row: the validation error of 6-16,16-512,108-200 in the category "danger". Top row: the validation error of 6-16,16-512,108-200 in the category "mandatory". Other parameters are the same, η = 1e-5, $P = 4$.

The detection time of CNN on the training set is shown in Table IV and Table V.

TABLE IV
DETECTION TIME ON DANGER TRAING SET, SIZE 2739

architecture	No. of parameters	time per ROI(s)	val error(%)
6-16	40406	0.0025	0.32
16-512	934382	0.022	0.05
108-200	1290326	0.108	0

TABLE V
DETECTION TIME ON MANDATORY TRAING SET, SIZE 10296, NOT FINISHED DURING THE COMPETITION PHASE

architecture	No. of parameters	time per ROI(s)	val error(%)
6-16	40406	0.0027	0.05
16-512	934382	0.023	0.12
108-200	1290326	0.128	0.09

From Table IV and Table V we can see that as the number of parameters increase, the time cost rise significantly, from 0.002s to 0.100s (no parallel computing was implemented). To achieve real-time detection speed, we can use simpler architecture. In the category "Mandatory", 108-200 can achieve 0 val error after training with bootstrap, which is the best performance in the three architectures, so we detect the traffic signs with 108-200 architecture in the category "Mandatory" and "Danger". After competition, we compared performance of the three architectures in the category "Danger", and found detection of 6-16 is 0.17% more accurate than 108-200 (99.9% vs 99.73%). Besides, using simpler architecture can reduce the time spent on training and detection.

From the color transform and fixed filter layer, we can usually get 5-20 interest areas per image to be classified more carefully, and this process costs 10-30s per image depending on the assignment of threshold of color transform and fixed layer. For learnable filter layers with the architecture of 108-200, a scene image would need about 2s (0.1 s per ROI). However, considering that if we use sliding window, we need to search in multi-scale and every location in the image, which is far more than 5-20 regions currently used. If we search traffic signs size ranges from 16-16 to 128-128 in a image as large as 1600x800, we need to detect some 10000 regions, which may cost far more time with complicated networks. Considering if we use Graphics Processing Units (GPUs) or some parallel computing method, the time of detection will be decreased further. What's more, to improve the detection speed in reality, we need to use efficient tracking method to decrease the frames we actually deal with.

The error of FP and FN in the validation set decreased significantly after bootstrap (see Table VI). We would compare the result of bootstrap and no boost in the test data (provided in the test phase) to see the efficiency of bootstrap.

TABLE VI
COMPARE BETWEEN BOOTSTRAP AND NO BOOTSTRAP IN THE VALIDATION SET, MANDATORY, 108-200

Method	FP	FN
no bootstrap	5.27957%	9.26829%
bootstrap	3.74307%	7.9878%

C. Experiment on the Test Set

In the test phase, we compare the result using only ROIs and using both ROIs and CNN recognition module to see the contribution of each module (see Figure 8).

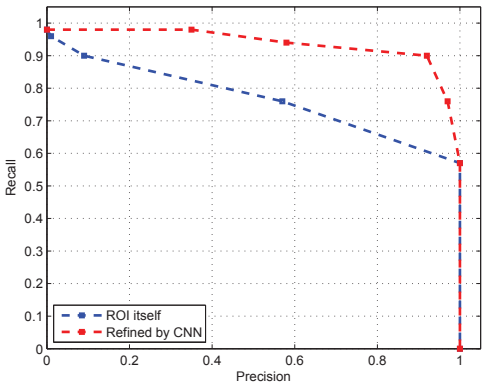

Fig. 8. fixed layer only, fixed and learnable layers , Mandatory, test-phase

Using learnable layers, with the same recall, we can get pretty much higher precision rate.

Based on the experiment on the training set, we compared the efficiency with bootstrap or not using the same architecture of 108-200 in the test set in Category "Mandatory"(see

Figure 9 and Figure 10). The bootstrap method improves the recall rate, with less false negatives.

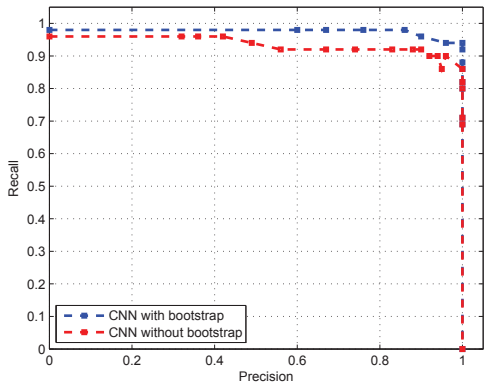

Fig. 9. bootstrap and no bootstrap, 108-200, Mandatory, test-phase

In the Category "Danger", the result without bootstrap using architecture of 108-200 is good enough, so we didn't do bootstrap(see Figure 10).

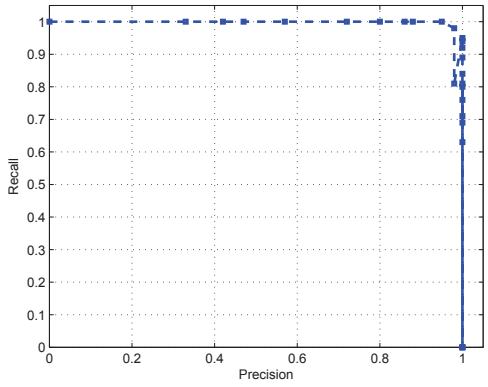

Fig. 10. 108-200, Danger, test-phase

The results of category mandatory and danger in the competition are listed in Table VII and Table VIII. Our team is "wff" with the result of 99.72% in the category "Danger", and 97.62% in the category "Mandatory". We use the same architecture to detect both categories in the competition phase, namely a fixed layer, and 108-200 learnable layers with L_p pooling ($P = 4$).

TABLE VII
THE RESULT OF AUC OF CATEGORY DANGER

TEAM	METHOD	AUC
visics	$boosted\_intChn\_ratio\_scales$	100%
visics	$boosted\_intChn\_ratio\_norm$	99.95%
wgy@HIT501	$HOG\_LDA\_SVM\_ADJ2$	99.91%
wff	$color + cnn$	99.73%

TABLE VIII
THE RESULT OF AUC OF CATEGORY MANDATORY

TEAM	METHOD	AUC
wgy@HIT501	HOG_LDA_SVM2	100%
wff	$color + cnn + boost$	97.62%
visics	$boosted\_intChn\_multiScale$	96.98%

V. CONCLUSIONS AND FUTURE WORK

In this paper, an approach based on the combination of color transformation and Convolutional Neural Networks (CNN) is proposed. Working on the image preprocessed by color transformation, the CNN with fixed and learnable layers has achieved good results. The merits of the CNN we used are as follows: First, fixed layer can reduce the amount of areas the classifier need to deal with, which could speed up the detection significantly. Second, the ROIs generated by fixed filter are very close to the borders of traffic signs, therefore the problem of alignment is avoided, otherwise performance of supervised convolution network would degrade. Third, CNN with appropriate architecture learned in the supervised way has been proved to be suitable to extract features for traffic sign classification. Our experiment results strongly supported our conclusion.

A drawback of the proposed model is that it can not do real-time detection. Our future work is to improve the efficiency of this algorithm. Parallel algorithm can be introduced to speed up the process time of fixed and learnable layers. The process time of multiple learnable layers could be decreased by introducing sparsity in extracting features.

REFERENCES

[1] W. C. U. Ritter and D. B. AG, "Traffic sign recognition in color image sequences," in *Intelligent Vehicles '92 Symposium., Proceedings of the*. IEEE, 1992, pp. 12–17.

[2] G. W. N. Ubong Lydia Jau, Chee Siong Teh, "A comparison of rgb and hsi color segmentation in real - time video images: A preliminary study on road sign detection," in *Information Technology, 2008. ITSim 2008. International Symposium on*, vol. 4, 2008, pp. 1–6.

[3] L.-W. Tsai, J.-W. Hsieh, C.-H. Chuang, Y.-J. Tseng, K.-C. Fan, and C.-C. Lee, "Road sign detection using eigen colour," *Computer Vision, IET*, vol. 2, no. 3, pp. 164–177, 2008.

[4] Y. Aoyagi and T. Asakura, "A study on traffic sign recognition in scene image using genetic algorithms and neural networks," in *Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on*, vol. 3, 1996, pp. 1838–1843 vol.3.

[5] R. Belaroussi and J. P. Tarel, "Angle vertex and bisector geometric model for triangular road sign detection," in *Applications of Computer Vision (WACV), 2009 Workshop on*, 2009, pp. 1–7.

[6] B. Hoferlin and K. Zimmermann, "Towards reliable traffic sign recognition," in *Intelligent Vehicles Symposium, 2009 IEEE*, 2009, pp. 324–329.

[7] G. Loy and N. Barnes, "Fast shape-based road sign detection for a driver assistance system," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, 2004, pp. 70–75 vol.1.

[8] Y.-S. Huang, M.-Y. Fu, and H.-B. Ma, "A combined method for traffic sign detection and classification," in *Pattern Recognition (CCPR), 2010 Chinese Conference on*, 2010, pp. 1–5.

[9] P. Paclłk, J. Novovicov, P. Pudil, and P. Somol, "Road sign classification using laplace kernel classifier," *Pattern Recognition Letters*, pp. 1165–1173, 2000.

[10] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 31 2011-aug. 5 2011, pp. 2809 – 2813.
[11] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Computer Vision, 2009 IEEE 12th International Conference on*, 29 2009-oct. 2 2009, pp. 2146 –2153.
[12] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *IJCNN'11*, 2011, pp. 1918–1921.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS'12*, 2012, pp. 1106–1114.
[14] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks (submitted)*, 2013.
[15] X. H. J. L. Ming Liang, Mingyi Yuan and H. Liu, "Traffic sign detection by roi extraction and histogram features-based recognition." IJCNN 2013, 2013.
[16] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
[17] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *ICML'10*, 2010, pp. 111–118.
[18] S. Lyu and E. P. Simoncelli, "Nonlinear image representation using divisive normalization," in *CVPR'08*, 2008, pp. –1–1.
[19] P. Sermanet, K. Kavukcuoglu, and Y. Lecun, "Eblearn: Open-source energy-based learning in c++."