Traffic Signs Detection and Recognition System Using Deep Learning

[1]Marco Magdy William, [2]Pavly Salah Zaki, [3]Bolis Karam Soliman, [4]Kerolos Gamal Alexsan, [5]Maher Mansour, [6]Keroles K. Khalil, [7]Magdy A. El-Moursy
[1-5]*Computer and Communication Department, Faculty of Engineering, Helwan University, Cairo, Egypt*
[6,7]*Mentor, A Siemens Business, Cairo, Egypt*
E-mails: [1]marcomagdy241@gmail.com, [2]pavlysz472@gmail.com, [3]boliskaram8@gmail.com, [4]kerolsgamal113@gmail.com, [5]manmaher24@yahoo.com, [6]keroles_khalil@mentor.com, [7]magdy_el-moursy@mentor.com

***Abstract*—With the rapid development of technology, automobiles have become an essential asset in our day-to-day lives. Which is despite being useful, it also has its dangers. Those dangers are, of course, road accidents. Road accidents may occur due to a multitude of reasons; mainly due to the drivers' inability to focus on the road, on driving and on traffic signs all while avoiding all possible distractions, which can be a fairly difficult task. This is why researchers began thinking about automating most of the drivers' responsibilities in order to increase road safety. One of the more important researches is Traffic Signs Recognition (TSR) systems. This paper describes an approach for efficiently detecting and recognizing traffic signs in real-time, taking into account the various weather, illumination and visibility challenges through the means of transfer learning. [1, 2] We tackle the traffic sign detection problem using the state-of-the-art of multi-object detection systems such as Faster-RCNN and SSD combined with various feature extractors such as ResNet 50, MobileNet v1 and Inception v2, and also Tiny-YOLOv2. The aforementioned models that were previously pre-trained on Microsoft's COCO dataset, [3] were fine-tuned on the German Traffic Signs Detection Benchmark (GTSDB) dataset. [4]**

We have also tested our models on the Raspberry Pi 3 Model B+ and the TASS PreScan simulation, [5] and we will discuss the results of all aforementioned models in the conclusion section.

***Keywords*— Advanced Driver Assistance System (ADAS); Traffic signs detection; Traffic signs recognition; Tensorflow**

I. Introduction

With the rapid technological advancement, automobiles have become a crucial part of our day-to-day lives. This makes the road traffic more and more complicated, which led to more traffic accidents every year. According to the Association for Safe International Road Travel (ASIRT) organization, about 1.3 million people die (including 1,600 children under 15 years of age!), and about 20-50 million are injured or disabled annually due to traffic accidents. [6]

There are numerous reasons that lead to those horrifying numbers of road accidents: according to San Diego Personal Injury Law Offices, the leading causes for such traumatic accidents are distracted driving and speeding. [7] Hence, a serious and immediate action needed to be taken, and that action was the Advanced Driver Assistant System (ADAS). ADAS refer to high-tech in-vehicle systems that are designed to increase road safety by alerting the driver of hazardous road conditions. Examples of the crucial ADAS sub-systems are Lane Departure, Collision Avoidance, and Traffic Signs Recognition (TSR). Recently, Traffic Signs Recognition has become a hot and active research topic due to its importance; there are various difficulties presented to the drivers that hinder their ability to properly see the traffic signs. Some of those difficulties are shown in Fig. 1. Hence it was necessary to automate the traffic signs detection and recognition process efficiently.

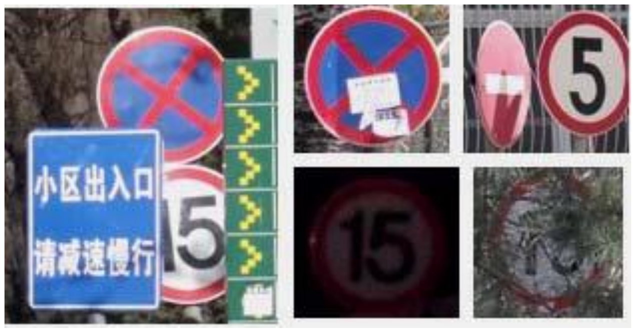


Figure 1. Difficulties that may face TSR systems in real-life

According the GTSDB, Road traffic signs are divided into three main categories: ***Prohibitory***, ***Mandatory*** and ***Danger***. ***Prohibitory*** signs are used to ban certain behaviors, ***Mandatory*** signs indicate pedestrians, vehicles and intersections, and finally ***Danger*** signs alert drivers to be aware of dangerous targets.

TABLE 1. Traffic signs categories according to GTSDB

Category	Shape	Color	Example
Prohibitory	Circular	Red, Blue, White & Black	
Mandatory	Rectangular & Circular	Blue, White & Black	
Danger	Triangular	Red, White & Black	

In this paper, we took the deep learning approach because a model can learn the features from images autonomously from the training samples specially when manually designing a feature extractor is proven to be arduous while producing better and more efficient results.

This paper is organized as follows: In *section II*, some previous related work is presented. The network structure and its implementation is described in details in *section III*. In *section IV*, experimental results for the proposed algorithm are provided. Finally, the paper is concluded with our personal notes regarding the system in *section V*.

II. RELATED WORK

In this section, the related work in TSR is reviewed. The study of traffic signs recognition started as the Program for European Traffic with High Efficiency and Unreasonable Safety (PROMETHEUS) funded by automobile companies such as Mercedes Benz in order to study traffic sign recognition system. [8] The traffic sign recognition consists of three stages—*detection*, *tracking* and *recognition*.

A. *Detection*

The goal of the detection phase is to locate the regions of interest (RoI) in which the object is most likely to be found and indicate the object's presence. During this phase the image is segmented, [9] a potential object is then proposed according to previously provided attributes such as color and shape.

B. *Tracking*

In order to assure the correctness of the proposed region, a tracking phase is needed. Instead of detecting the image using only one frame, the algorithm would track the proposed object for a certain number of frames (usually four). This has proven to increase the accuracy significantly. The most common object tracker is the Kalman Filter. [10]

C. Recognition

The recognition phase is the main phase in which the sign is classified to its respective class. Older object recognition techniques may include statistical-based methods, Support Vector Machine, Adaboost and Principal Component Analysis. [11]

However, in the more recent years, and thanks to the exponential technological advancement, deep learning approaches have become more and more popular and efficient. Convolutional Neural Networks(CNNs) [12] have achieved great success in the field of image classification and object recognition. Unlike the traditional methods, CNNs can be trained to automatically extract features and detect the desired objects significantly faster and more reliable. [13]

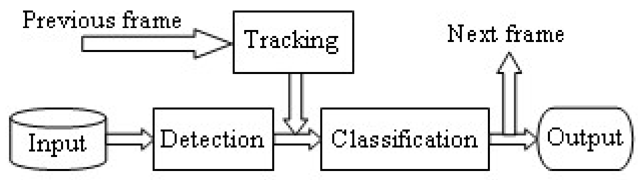


Figure 2. Procedure of traffic sign recognition [14]

III. Traffic Signs Detection and Recognition

A. Dataset

Training and testing a Deep Convolutional Neural Network requires a large amount of data as a basis. The German Traffic Sign Detection Benchmark (GTSDB) has become the de facto of training DCNNs when it comes to traffic sign detection. It includes many types of traffic signs in extreme conditions—weather, lightening, angles, etc… which help the model train on and be able to recognize the signs found in those conditions. The GTSDB contains a total of 900 images (800 for training and 100 for testing). However, this number is clearly not enough for large-scale DCNN models such as F-RCNN Inception.

B. Pre-processing

Instead of converting each image to grey scale, the image is normalized so that the pixel values are between 0 and 1. It has been proven that neural networks perform better on such normalized values.

C. Network Structure

We have trained and tested various models, but in this section we will focus on the F-RCNN Inception v2 and Tiny-YOLO v2 models since they produced the best overall results.

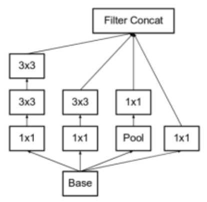
a. Faster RCNN Inception v2

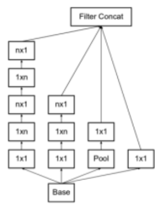
The first model is the Inception v2 [15] model as a front-end network structure (Fig. 6) of the Faster Recurrent Convolutional Neural Network (F-RCNN) [16] algorithm to detect and classify traffic signs. F-RCNN consists of Fast R-CNN detector and a Region Proposal Network (RPN) and then NMS is applied to choose the best region.

Equation 1 is for calculating the Intersection over Union in RPN to determine whether the proposed region contains an object or not.

$$\text{IoU} = \frac{\text{A n Gt}}{\text{A u Gt}} \begin{cases} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{cases}$$

Equation 1





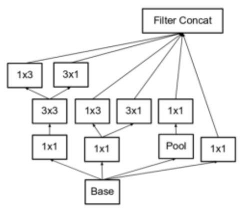


Figure 3 **Figure 4** **Figure 5**

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 3	35×35×288
5×Inception	As in figure 4	17×17×768
2×Inception	As in figure 5	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 6. Inception v2 structure

b. *YOLO v2*

The second used model is the YOLO v2, [17] YOLO v2 is Better, Faster and Stronger than YOLO v1[16].

Some YOLOv2 improvements over YOLOv1 (Fig. 7):

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Figure 7. Incremental improvements of YOLO v2[16]

The new model structure (Fig. 8), specially the usage of 1x1 convolution layers which reduces the number of parameters significantly, which in turn makes the model much faster.

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Figure 8. YOLOv2 structure

D. *Training the Models*

In this paper, the model is trained on 900 images from the GTSDB dataset and ***4 classes***— ‘***Prohibitory***’, ‘***Mandatory***’, ‘***Danger***’ and ‘***Stop***’. Although the GTSDB contains 43 sub-classes (Fig. 9), we opted to go for the 4 ‘main’ classes (Fig. 10) because 900 images are simply not enough to train a deep CNN model

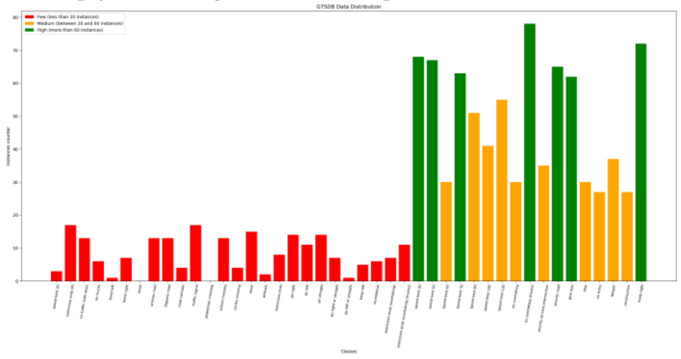


Figure 9. Data visualization of the GTSDB on 43 classes

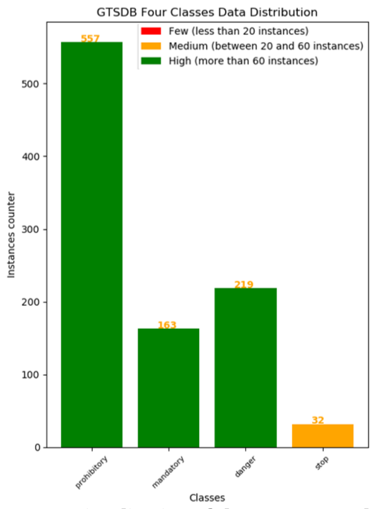
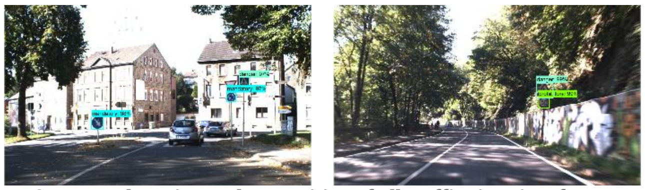


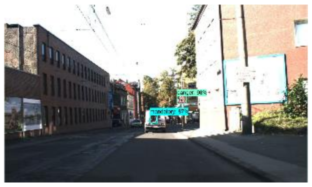
Figure 10. Data visualization of the GTSDB on the 4 main classes

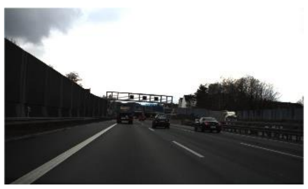
IV. EXPERIMENTAL RESULTS

Fig. 11 shows some of the obtained results including False Positives and False Negatives.



Accurate detection and recognition of all traffic signs in a frame





False Positive False Negative

Figure 11. Obtained results

TABLE 2. Performance comparison on GTX 1070

Model	mAP	Avg speed (FPS)
SSD MobileNet v1	83%	42
F-RCNN ResNet50	90%	~20
F-RCNN Inception v2	96%	~25
Tiny-YOLO v2	92%	~75

*Video resolution: 1280*720

TABLE 3. F-RCNN Inception v2 and Tiny-YOLO v2 models average accuracies on the four classes

Model	Prohibitory	Mandatory	Danger	Stop
F-RCNN Inception v2	98%	98%	96%	95%
Tiny-YOLO v2	74%	72%	73%	73%

TABLE 4. Speed comparison between different hosts operating the F-RCNN Inception v2 and Tiny-YOLOv2 models

Host Specs	F-RCNN Inceptionv2	Tiny-YOLOv2
CPU: I7 6500U @2.5GHz	~1	~18
GPU: GTX 1050 4GB	~13	~36
GPU: GTX 1070 6GB	25-30	60-75
GPU: Quadro P400 6GB	~20	45-50

Testing the SSD MobileNet v2 that was trained on 4 classes* on the Raspberry Pi 3 Model B produced a gracious speed of an average of 1 FPS, which can be considered enough for real-time applications.

*We tested other models as well, but some (e.g. FRCNN Inception v2) didn't even load properly and the process was 'killed' because the model was too heavy.

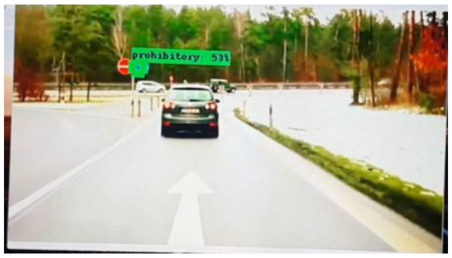


Figure 12. Results on Raspberry Pi 3 Model B+

Testing the FRCNN Inception v2 on the PreScan simulation gave us a speed of (speed of PreScan – 4 frames) FPS which is adequate for real-time applications.

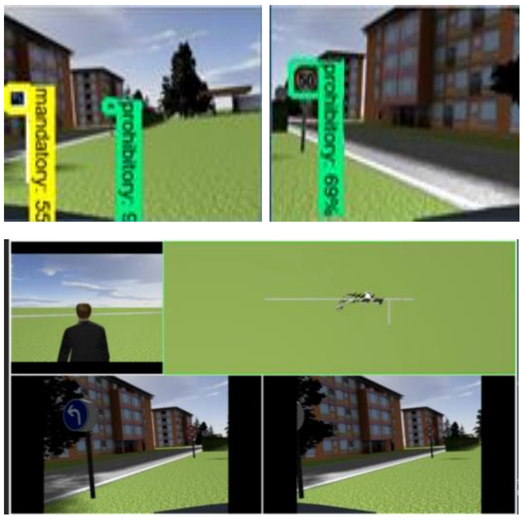


Figure 13. Results on TASS PreScan simulation

V. CONCLUSIONS

In this paper, we propose a fast and effective method to detect and classify traffic signs. The main contributions of this paper are as follows:

- Using a fully convolutional network and transfer learning, the F-RCNN Inception v2 model has managed to achieve accurate, reliable and fast results even in complex real-life road situations (average of 96% accuracy).
- Tiny-YOLO v2 is a super-fast model with a decent accuracy, but if you want higher accuracy you should opt for YOLO v2 or YOLO v3, just make sure to have a high-end GPU. (e.g. GTX 1080-Ti or any of the RTX series).
- After training the Inception v2 model on the GTSRB, [18] on 39,200 images, 43 classes and using similar configuration as shown in section III-D, we have managed to achieve an accuracy of 99.6% which is a record according to GTSRB competition. [18]
- Accuracy improvements can be achieved by adding significantly more training data (at least 40k images, for an average of 1,000 images for each class).

VI. REFERENCES

[1] Maxime Oquab, Leon Botton, Ivan Laptev, Josef Sivic, "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks," in INRIA, Paris, France, 2014.
[2] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu, "A Survey on Deep Transfer Learning," in Cornell University, 2018
[3] "Coco dataset," [Online] Available: http://cocodataset.org/#home
[4] "GTSDB dataset," [Online] Available: http://benchmark.ini.rub.de/?section=gtsdb&subsection=news
[5] "TASS PreScan simulation," [Online] Available: https://tass.plm.automation.siemens.com/prescan

[6] "ASIRT Organization," [Online].
Available: http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics.

[7] “San Diego Personal Injury Law Offices,” [Online]
Available: https://seriousaccidents.com/legal-advice/top-causes-of-car-accidents/

[8] Wang Canyong, “Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning,” in International Conference of Robots & Intelligent System, 2018.

[9] Lu Ming, “Image Segmentation Algorithm Research and Improvement,” in 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), 2010.

[10] C.-Y. Fang, S.-W. Chen, and C.-S. Fuh, “Road-Sign Detection and Tracking”, in Vehicular Technology, Sept. 2003.

[11] Er. Navjot Kaur, Er. Yadwinder Kaur, “Object Classification Techniques Using Machine Learning Model,” in International Journal of Computer Trends and Technology (IJCTT), 2014.

[12] Y. Wu, Y. Liu, J. Li, H. Liu, and X. Hu, “Traffic sign detection based on convolutional neural networks,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Aug. 2013.

[13] Bedi, Rajni, et al. “Neural Network Based Smart Vision System for Driver Assitsance in Extracting Traffic Signposts,” in Cube International Information Technology Conference, 2012.

[14] Meng-Yin Fu, Yuan-Shui Huang, “A Survey of Traffic Sign Recognition,” in the International Conference on Wavelet Analysis and Patter Recognition, July 2010.

[15] Chrstian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, “Rethinking the Inception Architecture for Computer Vision,” arXiv:1512.00567, 2016.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” arXiv:1506.01497, 2015.

[17] Joseph Redmon, Ali Farahdi, “YOLO9000: Better, Faster, Stronger,” in University of Washington, Allen Institute for AI, Dec. 2016.

[18] “GTSRB,” [Online]
Available:
http://benchmark.ini.rub.de/?section=gtsrb&subsection=news

[19] “GTSRB Competition,” [Online]
Available:
http://benchmark.ini.rub.de/?section=gtsrb&subsection=news