A tool for Automatic generation and annotation of a French traffic signs dataset for deep learning

Ibrahim Yunuslu and Kanan Jafarli

ibrahim.yunuslu@ufaz.az kanan.jafarli@ufaz.az

French-Azerbaijani University

Abstract

Vision-based traffic sign detection plays a vital role in intelligent transportation systems. Recently, many deep learning-based traffic sign detection methods have been proposed, and compared to traditional methods, they show better performance. However, due to the difficult conditions of the driving environment and the size of the traffic signs in the traffic scene images, the performance of the deep learning-based method in detecting small traffic signs remains limited. Furthermore, the inference speed of state-of-the-art traffic sign detection methods is still very slow. Also models have to be trained on good labeled datasets in order to get good results. But collecting and annotating dataset takes too much time and done manually by people. This paper proposes two approaches to make this process faster by making it automated. First aproach is to detect traffic signs using pretrained models(transfer learning) trained on The German Traffic Sign Recognition Benchmark(GTSRB) dataset which anotated manually. And the second one is to use Haar Cascades Classifier. After detecting traffic signs, they annotated automatically with region of interest, height, width and other labels.

Contents

1 Introduction

Vision-based traffic sign recognition plays a vital role in intelligent transportation systems, such as autonomous driving systems and advanced driver assistance systems. There are two stages: detection and recognition of traffic signs. Traffic sign detection uses the image captured by the camera to accurately locate the traffic sign area, while traffic sign recognition classifies each traffic sign into a corresponding category. The traffic signs detected in the detection phase are used as input in the recognition phase. Therefore, the accuracy of traffic sign detection has a great influence on the accuracy of the entire system. Many methods have been proposed to detect traffic signs [1]. Traditional methods [2, 3, 4, 5] generally detect road signs in images based on hand-made features (such as color, texture, edges, and other low-level features). In the driving environment, due to the diversity of appearance of traffic signs, the influence of other objects on the traffic signs, and the influence of lighting conditions, the performance of traditional traffic sign detection methods is deficient.

With the rapid development of deep learning, many road sign detection methods based on deep learning have recently been proposed [6, 7, 8, 9, 10, 11, 12]. Compared with traditional methods, they have excellent performance. The road sign detection method based on deep learning first creates road sign candidates, and then uses a classifier to identify road signs and background categories. The quality of a machine learning models depend on how you deal with three important factors: data collection, data pre-processing, and data labeling. However, logging data is often time consuming and complex. For example, image recognition systems generally need to draw boundaries around specific objects, while product recommendation and sentiment analysis systems may require complex cultural knowledge. And don't forget that a dataset can contain thousands of samples (if not more) that need

to be labeled. With this in mind, choosing the right method for a machine learning project means considering the complexity of the task, the size of the project, and the project schedule. Considering these factors, this document lists two methods for detecting traffic signs then automatically labeling data:

- using pretrained models(transfer learning).
- Haar Cascades Classifier.

After detecting traffic signs, they annotated automatically with Width, Height, X1, Y1, X2, Y2, ClassID, Image_Path.

2 Methodology or Approaches

2.1 Transfer Learning

In transfer learning, the knowledge from a well-trained machine learning model will be applied to other but related problems. Considering a simple classifier to predict if an image has a specific object in it, then using this knowledge that the model gained during training to identify other objects. Basically trying to use the knowledge model learned in one task to improve the generalization of another task, transfering the weights in task x to the new task y.

Common ideas are new tasks that have a lot of data, and using many training data labeled to use the knowledge learned from the task. We do not start the learning process from scratch, but from the patterns learned in solving related tasks.

2.2 Haar Cascade

Under current technological trends, computer vision has become an important entity in the technical field leading to limitless computer innovation. Consider com-

puter vision from this perspective: As humans, our eyes are an important part of the body, so we incorporate vision into computers / machines to enable them to see. To do this, we will use the Haar Cascade method. First, let's first understand what the Haar Cascade is. Haar Cascade is an efficient machine learning object detection approach that it is used to recognize objects in videos or images. The cascade function is trained using so many negative(background) and positive(object we need to detect) images. After training, it can be used for detecting objects in different images and videos. The Haar function considers adjacent rectangular areas at specific positions in the detection window, sums the pixel intensities in each area, and then calculates the difference between these sums.

3 Discussion

3.1 Code Explanation

We implemented three python files (*functions.py, main.py and write_csv.py*) for detect traffic signs (*'stop' sign*) from road videos, extract them and write their properties to csv file.

3.1.1 functions.py

This file consist of libraries :

- **os** – Provides functions for interacting with the operating system. Used to create folders and to work with paths.
- **cv2** – Provides functions for computer vision, machine learning, and image processing. Used to process images and videos to identify traffic sign.
- **copy** – Used to create copies that do not depend on the original.

- **numpy** – Provides functions for computations.
- **csv** – Provides functions for to work with csv files. Used to write properties of extracted traffic signs to csv file.
- **shutil** – Used to delete non-empty folders.
- **natsort** – Used to sort extracted images.

and functions :

- **create_dir(...)** – Takes a name and the path of parent directory as arguments and creates directory using *mkdir()* function of os module. If the same directory already exists, this function deletes this directory using *rmtree()* function of shutil module and creates new one. At the end, it returns the path of directory.
- **create_file(...)** – Takes a name as argument and creates csv.file in write mode that delete previous data and start from beginning using *open()* function. If the file is empty, the names of images properties (*Width, Height, X1, Y1, X2, Y2, ClassID, Image_Path*) is written using *writer.writerow()* function of csv module. The function returns writer object.
- **img_properties(...)** – Takes detected object, frame, path as arguments and deduces coordinates of sign (x_1, y_1, x_2, y_2). There may be more than one sign in the one frame. Each detected object has starting coordinates, width and height. Final coordinates (x_2, y_2) are calculated like this: $x_2 = x_1 + width$ and $y_2 = y_1 + height$. Then draw a green rectangle around every recognized sign using these coordinates with help of *rectangle()* function of cv2 module. To extract sign images we did some padding in coordinates. Extracted images are named in format of properties (count, width, height, starting and final

coordinates). At the end, function saves these extracted sign images as ppm file.

3.1.2 main.py

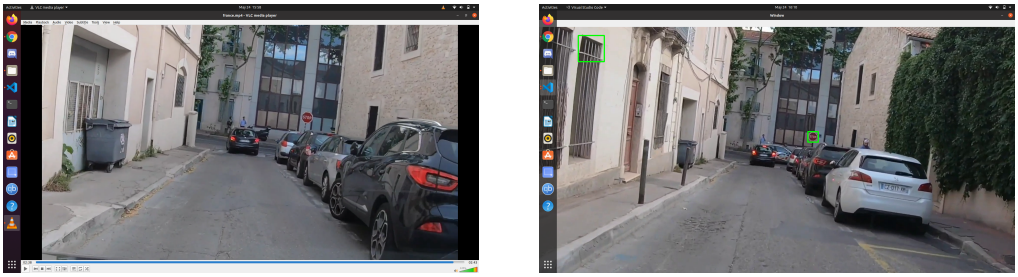
This file detects traffic signs with importing functions (functions.py). Firstly, creates parent directory and child directory which will be called as name of detected sign using *create_dir()* function. A necessary xml file (which are created using Haar Cascade method) is loaded using *CascadeClassifier()* function of cv2 module. Then opens video streams or images using *VideoCapture()* function of cv2 and creates detected object using *detectMultiScale()* function. The video we used is taken from Youtube (link). In the each frame, perform detection using *img_properties()* function. At the end, destroys all windows and exits video stream or image.

3.1.3 write_csv.py

This file writes properties (Width, Height, X1, Y1, X2, Y2, ClassID, Image_Path) of extracted sign images to csv file. It creates csv file using *create_file()* function. Then sorts images according to their names. In the each image, it splits their properties from their names and renames images. Finally, it writes these properties to csv file and also add ids and paths of images.

3.2 Results

In the following images we can see road images in the first column and detection of stop signs in the second column.



In this data we can see the properties of the extracted stop sign images which are widths and heights of images, starting and final coordinates of the stop signs, id of the stop sign and paths of the images.

	Width	Height	X1	Y1	X2	Y2	ClassID	Image_Path
0	61	61	5	5	56	56	1	image/stop_img/img_0.ppm
1	63	63	5	5	58	58	1	image/stop_img/img_1.ppm
2	63	63	5	5	58	58	1	image/stop_img/img_2.ppm
3	68	68	5	5	62	62	1	image/stop_img/img_3.ppm
4	67	67	5	5	61	61	1	image/stop_img/img_4.ppm
5	76	76	6	6	70	70	1	image/stop_img/img_5.ppm
6	66	66	5	5	60	60	1	image/stop_img/img_6.ppm
7	66	66	5	5	60	60	1	image/stop_img/img_7.ppm
8	70	70	5	5	64	64	1	image/stop_img/img_8.ppm
9	72	72	6	6	66	66	1	image/stop_img/img_9.ppm
10	70	70	5	5	64	64	1	image/stop_img/img_10.ppm
11	79	79	6	6	72	72	1	image/stop_img/img_11.ppm
12	80	80	6	6	73	73	1	image/stop_img/img_12.ppm
13	86	86	7	7	79	79	1	image/stop_img/img_13.ppm
14	93	93	7	7	85	85	1	image/stop_img/img_14.ppm
15	68	68	5	5	62	62	1	image/stop_img/img_15.ppm
16	68	68	5	5	62	62	1	image/stop_img/img_16.ppm
17	79	79	6	6	72	72	1	image/stop_img/img_17.ppm
18	78	78	6	6	71	71	1	image/stop_img/img_18.ppm
19	74	74	6	6	68	68	1	image/stop_img/img_19.ppm

Figure 1: Data

4 Drawbacks and perspectives

The main point of this paper is to automatically annotate French traffic signs dataset. However the biggest problem we faced during development is lack of data. To collect the data from France roads is hard to find, not much reliable sources(we can't be sure completely if the image taken in France). So we decided to use Youtube videos(download one by one manually, takes to much time) which captured in French roads, but still there is not so much of video.

Another problem is to detect traffic signs we have to train our classifiers for each sign, because there were no such amount of negative and positive images, we could not able to train our own classifier to detect all traffic signs.

5 Conclusion

Our task was to build automatically dataset including French traffic signs images Collected images must be in different sizes and forms of external noise like fading and blurring effect, affected visibility, multiple appearances of the sign, chaotic background, viewing angle problem, damaged and partially obscured sign, etc. From our detected images we can say pros side of datecting signs from videos is they include almost all of this noises in every frame. Traffic signs images have to be automatically Labelled (annotating) the signs in the images and extracting the traffic sign Region of Interest (ROI). Generated automatically csv file containing data having the following format: Width, Height, X1, Y1, X2, Y2, ClassID, Image_Path. Where:

- Width, Height of the image
- X1, Y1, X2, Y2: are the two points from which the sign (ROI) is properly framed

Considering the drawbacks, we managed to detect Stop sign and annotate it automatically.

References

[1] Chunsheng Liu, Shuang Li, Faliang Chang, and Yinhai Wang. Machine vision based traffic sign detection methods: Review, analyses and perspectives. *IEEE Access*, 7:86578–86596, 2019.

[2] C. Bahlmann, Y. Zhu, Visvanathan Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 255–260, 2005.

[3] Samuele Salti, Alioscia Petrelli, Federico Tombari, Nicola Fioraio, and Luigi Di Stefano. Traffic sign detection via interest region extraction. *Pattern Recognition*, 48(4):1039–1049, 2015.

[4] Selcan Kaplan Berkaya, Huseyin Gunduz, Ozgur Ozsen, Cuneyt Akinlar, and Serkan Gunal. On circular traffic sign detection and recognition. *Expert Systems with Applications*, 48:67–75, 2016.

[5] Zhe Zhu, Jiaming Lu, Ralph R. Martin, and Shimin Hu. An optimization approach for localization refinement of candidate traffic signs. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3006–3016, 2017.

[6] Zhigang Liu, Juan Du, Feng Tian, and Jiazheng Wen. Mr-cnn: A multi-scale region-based convolutional neural network for small traffic sign recognition. *IEEE Access*, 7:57120–57128, 2019.

[7] Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, and Xiaolin Hu. Traffic sign detection based on convolutional neural networks. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2013.

[8] Yingying Zhu, Chengquan Zhang, Duoyou Zhou, Xinggang Wang, Xiang Bai, and Wenyu Liu. Traffic sign detection and recognition using fully convolutional network guided proposals. *Neurocomputing*, 214:758–766, 2016.

[9] Jia Li and Zengfu Wang. Real-time traffic sign recognition based on efficient cnns in the wild. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):975–984, 2019.

[10] Yi Yang, Hengliang Luo, Huarong Xu, and Fuchao Wu. Towards real-time traffic sign detection and classification. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):2022–2031, 2016.

[11] Cuiping Li, Zhenxue Chen, Q. M. Jonathan Wu, and Chengyun Liu. Deep saliency with channel-wise hierarchical feature responses for traffic sign detection. *IEEE Transactions on Intelligent Transportation Systems*, 20(7):2497–2509, 2019.

[12] Tingting Yang, Xiang Long, Arun Kumar Sangaiah, Zhigao Zheng, and Chao Tong. Deep detection network for real-life traffic sign in vehicular networks. *Computer Networks*, 136:95–104, 2018.