Traffic-sign recognition system based on YOLO object detector and multi-class classifier

Lorenzo Busin
lorenzo.busin@studenti.unipd.it

Nicolò Tartaggia
nicolo.tartaggia@studenti.unipd.it

Abstract

The aim of the project is to build a traffic signs detector able to recognize different types of traffic signs. The task is similar to the built-in computer of modern cars, in fact those systems are capable to read traffic signs and to show, for example, which is speed limit in that road. The system will be based on a convolutional neural network, a specific neural network particularly suitable for the area of image recognition. The main focus of this project is, given a road image with a traffic sign, to analyze the image, to detect the sign and to classify it via a multiclass classifier and then telling the user which kind of sign it is (or telling the driver what actions to perform in that scenario). In addition to this, we have expanded the functionality of this project to make it works even with video clips, so that the recognition of traffic signs takes place in real time.

1. Introduction 10%

Image processing and its related algorithms are used to process digital images. Although it may seem recent technology, many of the techniques were developed in the 1960s. However, computers were expensive at this time and the digital imaging process required too much resources for many to even consider. The situation changed in the 1970s, when digital image processing began to be used massively as cheaper computers and dedicated hardware became available. With the accessibility to faster computers in the 2000s, digital image processing has become the most common form of image processing, able to compute different tasks such as image classification, video and real time detection. With the birth and growth of intelligent systems, computer vision is increasingly used in the field of intelligent transport, which includes the subject of this paper: traffic sign recognition. It can be seen as a third eye which concerns a wide area of usages such as advanced driver assisting systems, road surveying, autonomous driving, building and maintaining maps of signs, mobile mapping systems, vehicle navigations system, surveillance and self-govern robot navigation systems. These systems are typically based on detecting a region of interest (ROI), in which the traffic sign is located, recognising typical characteristics such as colour and geometric form. These information provide crucial visual details in order to understand the proper driving conditions. For example, they inform about speed limits, drivable lanes, obstacles, temporary situations, roadway access, restrictive areas, etc. Reasons why they are designed to be easily detectable, recognizable and interpretable by humans.

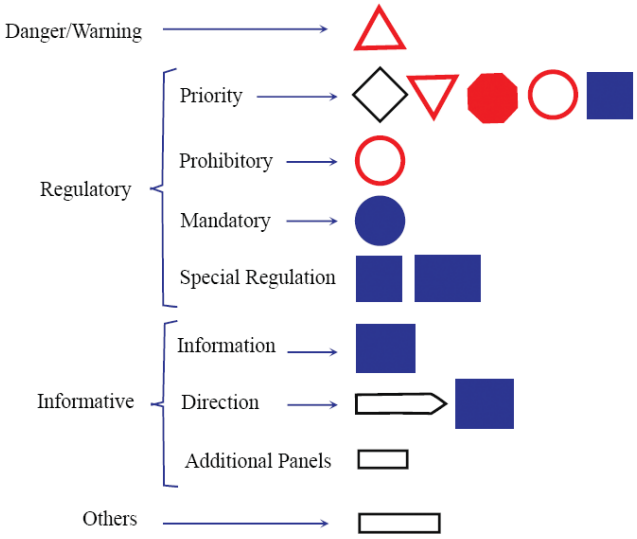

Figure 1. European Traffic Sign Categories

The application developed includes not only the standard functionality of image processing but also image and video detection and image classification. The idea is to concatenate different processes in order to compute the final labels for each traffic-sign without requiring a big amount of resources. In fact, traffic-sign detectors for large numbers of categories remains a challenging problem.
The process starts with the detection stage for the input image or video. This phase exploits the YOLO [1] library for object detection which detects the area where a possible traffic sign is located and draws the so-called bounding boxes around it. We chose YOLO because of its main fea-

ture of being significantly faster than other framework, trading a bit of accuracy. Once the first step is completed, each object is passed to the classifier which is in charge to find out the correct label for the image. For this purpose we lean on Keras, a deep-learning framework.

2. Related Work 10%

The idea behind this project is the development of an object recognition system. The term recognition refers to a suite of challenging computer vision tasks which can be divided into three categories:

- **image classification**: predict the type or class of an object in an image;
- **object localization**: locate the presence of objects in an image and indicate their location with a bounding box;
- **object detection**: locate the presence of objects with a bounding box and types or classes of the located objects in an image.

In the last few years many application have been developed to explore different scenarios of object recognition through different techniques and deep learning models specifically dedicated [2, 3, 4]. Moreover with the continuous improvement of computing power available for the user, great progress has been made in the field of general object detection. The large amount of online projects and papers demonstrate how this field is growing year after year.
Our application aims to contribute to the traffic sign recognition problem, exploring the latest approaches of object detection and image classification. Here we present some published works related to our project. Also, readers are referred to the following surveys for more details [5].

2.1. Traditional methods

Traditional traffic sign detection methods can be summarized in colour-based methods and shape-based methods. This phase includes image preprocessing, enhancement and segmentation according to the base attributes of colour and shape. Given an image, the goal is to detect potential pixel areas which is the region of interest where an object may be located. Then the potential traffic sign is normalized in order to enter the classification phase.
Creusen et al. [6] propose a system based on Histogram of Oriented Gradients (HOG) algorithm. Using the colour and shape informations they show that system performances improves because the gradient is less dependent on the background contents. Wang et al. [7] try to reduce the impact of lightning conditions with a RGB normalization process. These methods, among others similar ones, suffer the environment features of the image. Anyway, they reach significant results.
For the classification purpose, researchers leaned on different machine learning algorithms. Support vector machine (SVM) is the most popular one [8, 9, 10].

2.2. CNN-based methods

With the development of deep learning algorithms, the convolutional neural network (CNN) brings very good results. CNNs improve validation accuracy and robustness and reduce the requirements of quality of images and related computation. Qiao et al. [11] proposed a system based on faster R-CNN to achieve region proposal, feature extraction and classification, which greatly reduces the repetitive computation and speeds up the operation. Mehta et al. [12] compared different approach to regularize overfitting and showed that Adam optimizer adapts faster than other ones like Stochastic Gradient Descent (SGD). Kong et al. [13] contributed with a lightweight cascaded CNN approach based on YOLO v2-tiny which reduced hardware complexity by decreasing the number of parameters. Wang et al. [14] explored CNN area including an accurate preprocessing stage to improve quality of the images.

3. Dataset 15%

In this section we present the datasets we have used: one for the detection task and one for the classifier models.

3.1. Dataset for the detector

For training the detector model using the YOLO algorithm, we built our own dataset. Firstly, we extracted a hundred JPG images from the DFG Traffic Sign Dataset which consists of about 7000 images with 1920x1080 resolution captured in Slovenian roads. The RGB images were acquired with a camera mounted on a vehicle that was driven through six different Slovenian municipalities. The image data was acquired in rural as well as urban areas. Each image contains at least one traffic sign. After that, using a utility tool we manually selected for each image the coordinates of the position where the traffic signs are located. In such way, for each image in the training set there is a .txt file that indicates in each row the position of the traffic signs. To locate target objects we used the YOLO bounding box format (0, x-top left, y-top left, width, height) where 0 represents the class of the object (the model has only to detect the traffic signs location), x and y represent the coordinates of the top left corner, width and height represent the object dimensions. For testing this model we've taken other images from the same dataset. The same was done to build the validation set.

The Belgium Traffic Sign Dataset refers to a set of 62 classes of traffic signs: training data files are divided into 62 folders that contain the images in PPM format. The same for validation data. We decided to cut some classes and to add some new ones in order to build an accurate classifier which focuses on most meaningful traffic signs. The dataset we have used to train and to validate the classifier

Figure 2. Differences between signs of the same class

model is made of 56 traffic sign classes with 6229 images for training and 2493 for the validation. Every image is a cropped portion taken from a road picture which represents one traffic sign. Typically the images are characterized by small square dimensions and sizes. The aim of the classifier is to pick the right class for each input image, but in Europe there could be small appearance differences between the same traffic sign classes from a state and another.

3.2. Dataset for the classifier

The Belgium Traffic Sign Dataset refers to a set of 62 classes of traffic signs: training data files are divided into one folder per class, each one containing the images in PPM format. The same for validation data. We decided to cut some classes and to add some new ones in order to build an accurate classifier which focuses on most meaningful traffic signs. For some classes the number of images was too low to obtain a good result, so we merged this dataset with some images taken from the German Traffic Sign Recognition Benchmark, a dataset made for a multi-class image classification benchmark in the domain of advanced driver assistance systems and autonomous driving. The dataset we have used to train and to validate the classifier model is made of 56 traffic sign classes with 6229 training images and 2493 validation images. Each image is a cropped portion taken from a road picture which represents one traffic sign. Typically the images are characterized by square dimensions and small sizes. The aim of the classifier is to pick the right class for each input image, but in Europe there could be small appearance differences between the same traffic sign classes from a state and another. To fix this issue, we manually added some images to training and validation data for classes which suffer this problem. Also, we adopted the augmentation technique to build a more generalized dataset and to prevent the overfitting problem. We added small variations to the input images introducing rotation by 10%, width and height resizing, shearing and zooming randomly up to a tenth. Is important to not exaggerate with the changes because it could cause problem with the classification, due to the fact that some signs have specific orientation.

Model	Accuracy	Train time (h)
YOLO	0.97	24
tiny-YOLO	0.84	4

Table 1. Metrics for detection model

Model	Accuracy	Train time (m)
CNN1	0.86	7
CNN2	0.95	41
CNN3	0.96	23
LeNet	0.93	12

Table 2. Metrics for classification models

3.3. Custom made test dataset

The application is supposed to work in real-time situations and for this reason we recorded some videos using a front car camera. For video recording we have used an iPhone XR setting the quality on 1080p and 30 fps. We've taken short videos, from 30 seconds to a minute, for testing how good the detector and the classifier works.

4. Method 30%

Our method is composed of two consecutive operations: detection and classification. For the first one we adopted YOLO, an one-stage detector based on CNNs able to predict bounding boxes from input images directly without a region proposal step, achieving time efficiency and high inference speed. Therefore, it can be used not only for image detection but also for real-time devices. For the latter we built a multi-class classifier which receives as input the object detected. The reason why we don't adopt YOLO for the whole process is linked to the fact that there is a large number of classes to which a traffic sign could belong. Build a network capable of execute both detection and recognition requires a very long training time and one or more accurate and deep datasets.
One significant aspect that contributed to the choice of YOLO is the fact that it is a young framework constantly updated and supported by its team. In fact, the version used for this project is YOLOv3 [16], released in 2018. The latest version, YOLOv4 [17], was published on April 2020 but the lack of papers and projects to compare our results with lead us to choose the v3.

4.1. Network setup

To perform detection we setup the YOLO network in order to make it works with traffic signs.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3. Yolov3 structure called Darknet-53

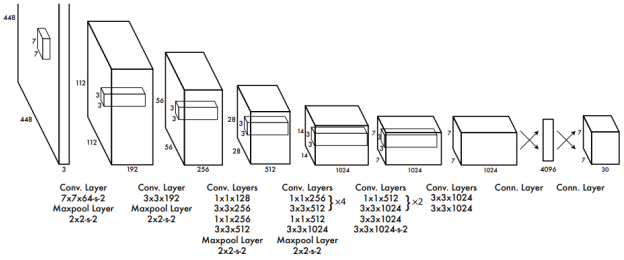

Figure 4. YOLO architecture

4.2. YOLO architecture

5. Experiments 30%

Initially we trained the models and tested them independently using the validation sets for the classifiers and for the detectors. After that, we proceeded to manually testing the 2 step detection and classification technique, analyzing both images and videos with different scenarios.

5.1. Detection models

We trained the detector models on Google Colab with 2 CPUs @ 2.2 GHz, 13 GB of RAM and Nvidia Tesla K80 GPU. The YOLO model was trained for 4000 epochs with 64 batch size and it took about 24 hours and the tiny-YOLO model was trained for 4000 epoch with 64 batch size and it took about 4 hours. For training the models we used convolutional weights that are pre-trained on ImageNet model[1]. Pre-trained weights was used as initializers for the new training procedures. This technique is also called transfer learning and avoids learning everything from scratch. The dataset used to compute this metric was the validation set manually built, as explained above. The computation of the accuracy value for a detection task is not trivial. We developed a specific method for this purpose. The main idea is that the detector select with bounding boxes the position of the traffic signs, selecting the x and y coordinates of the top left corner, the width and the height of each sign, hence using the YOLO format as for the annotations which represented the ground-truth. So, we compared for each traffic signs one by one these values with 10% of confidence: this value has been chosen considering the human error made during the dataset construction and a small prediction error. The results (see Table 1) show up a substantial difference between the accuracy values but also a substantial difference on the train times. This is due to the fact that tiny-YOLO is a lighter version of the YOLO model. In any case, considering a detection as a bad one doesn't mean that the detection is wrong, but only not well defined for that sign and consequently implies a higher probability of error for the classifier.

5.2. Classification models

We trained the classifier models with 4 CPUs Intel Core i7-6700HQ @ 2.6 GHz, 8 GB of RAM and Nvidia GeForce GTX 950M. 16 batch size and 100 epochs for each model. All models were trained for 100 epochs with 16 batch size and they took different train time. The accuracy value of each models was computed after their training and it used the validation set, as explained above. This metric computes the percentage of correct predictions. The results (see Table 2) show up that CNN1 has obtained the worst accuracy value because of its shallowness. The other models reached good and similar accuracy values.

5.3. Detection with classification testing

Analyzing the obtained results we selected the best combination of detector and classifier. As detector we had chosen the YOLO model because the detection task is fundamental for a good classification: in fact, for getting accurate predictions the bounding boxes have to be precise as possible. As classifier we had chosen the CNN3 model because it reached the best accuracy value with a reasonable train time. Testing automatically the fully working application is a very difficult and for this reason we spent a lot of time for testing it manually. Initially we tested the 2 step detection and classification for road images analysis and then

[1] https://pjreddie.com/darknet/imagenet

we tested it for videos analysis. During the video analysis, in fact, each frame is processed independently like a single image. The dataset we used for the experiments is our custom made dataset, as above explained. We also compared the same road in different light conditions, like day and night. During the day, the detector is able to spot traffic signs from a great distance and hence to classify them. During the night, instead, signs can be spotted only at close range: the detection distance can be increased if the road is provided by street lamps which light up the signs enough for their identification. The difficulty of this task is the variety of different scenarios that we have to face on road. According with Serna et al. [15] results, we found that most of the misclassified traffic signs show the following characteristics:

- Bad lightning;
- Strong motion blur;
- Human added artifacts;
- Poor image quality;
- Strong shadows or highlights;
- Occlusions;
- Strong similarity with other signs.

All of the elements listed above are commonly found on the roads and all of them contribute to the increase of the probability of error.

6. Conclusion 5%

Recap esplorato nuovo framework in crescita, topic importante riconoscimento cartelli per macchine guida autonoma ecc Yolo funziona bene punti di forza, ma anche debolezze non sempre accurato per la natura stessa del modello al contrario di altri modelli basati su CNN che si focalizzano sui cartelli però sono molto più lenti, delle diverse condizioni ambientali 2 step permette l'estensibilità e modularità

what we learned yolo, opencv, keras leggere paper per ispirazione

espandere modello yolov4, espansione classi e constestualmente espandere dataset personalizzare loss function per ottenere bounding boxes ancora più preciso migliorare data augmentation

6.1. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example [1]. Where appropriate, include the name(s) of editors of referenced books.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection". In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, June 2016.

[2] C. Wang, "Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning". In *2018 International Conference on Robots & Intelligent System (ICRIS)*, Changsha, pp. 150-152. DOI: 10.1109/ICRIS.2018.00047.

[3] Y. Sun, P. Ge and D. Liu, "Traffic Sign Detection and Recognition Based on Convolutional Neural Network". In *2019 Chinese Automation Congress (CAC)*, Hangzhou, China, pp. 2851-2854. DOI: 10.1109/CAC48633.2019.8997240.

[4] D. Tabernik and D. Skočaj, "Deep Learning for Large-Scale Traffic-Sign Detection and Recognition". In *April 2020 IEEE Transactions on Intelligent Transportation Systems* vol. 21, no. 4, pp. 1427-1440. DOI: 10.1109/TITS.2019.2913588.

[5] B. Sanyal, R. K. Mohapatra and R. Dash, "Traffic Sign Recognition: A Survey". In *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, Amaravati, India, pp. 1-6. DOI: 10.1109/AISP48273.2020.9072976.

[6] I.M. Creusen, R.OJ. Wijnhoven, E. Herbschleb, and P.H.N. de With, "Color exploitation in hog-based traffic sign detection". In *September 2010 Proc. of the IEEE International Conference on Image Processing (Ic/P)*, pp. 2669-2672. DOI: 10.1109/ICIP.2010.5651637.

[7] Qiong Wang and X. Liu, "Traffic sign segmentation in natural scenes based on color and shape features". In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, Ottawa, ON, pp. 374-377. DOI: 10.1109/WARTIA.2014.6976273.

[8] D. Soendoro and I. Supriana, "Traffic sign recognition with Color-based Method, shape-arc estimation and SVM." In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics* Bandung, pp. 1-6. DOI: 10.1109/ICEEI.2011.6021584.

[9] J. Greenhalgh and M. Mirmehdi, "Real-Time Detection and Recognition of Road Traffic Signs". In *December 2012 IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 4*, pp. 1498-1506. DOI: 10.1109/TITS.2012.2208909.

[10] S. I. Rashid, M. A. Islam and M. A. M. Hasan, "Traffic Sign Recognition by Integrating Convolutional Neural Network and Support Vector Machine". In *2019*