Real-Time Road Sign Detection and Classification

1st Dumitru Mițca
dept. CTI
Technical University ”Gheorge Asachi” of Iași
Iași, Romania
dumitru.mitca@student.tuiasi.ro

2nd Mihai Mihalache
dept. CTI
Technical University ”Gheorge Asachi” of Iași
Iași, Romania
mihai.mihalache2@student.tuiasi.ro

3rd Otilia Zvorișteanu
dept. CTI
Technical University ”Gheorge Asachi” of Iași
Iași, Romania
otilia.zvoristeanu@academic.tuiasi.ro

4th Simona Caraiman
dept. CTI
Technical University ”Gheorge Asachi” of Iași
Iași, Romania
simona.caraiman@academic.tuiasi.ro

***Abstract*—Road signs are a vital part of traffic infrastructure and their correct detection and classification is a vital task in autonomous driving applications. In this paper, we present a cursory study over state-of-the-art software solutions and related papers in the field, and then we present our attempts to implement various solutions based on what we researched, in an exploratory manner.**

***Index Terms*—computer vision, road sign detection, autonomous driving**

I. Introduction

Road sign detection and classification is a significant task in the computer vision space, that has been studied for at least a decade and a half. Applications in this domain have strict requirements, as mistakes or software bugs can lead to significant financial loss or even human death in the worst case, requirements such as:

- low, tending towards 0%, misclassification rate — classifying a stop sign as a right-of-way sign can lead to accidents, this example is unlikely to happen in a real application, but it shows the worst case scenario; more likely missclassication would be ”must go left” as ”must go right”, or misclassifying the various ”road thins ahead” signs between each other
- high speed — a vehicle should not stop at every road sign it encounters in order to recognize it then proceed, abiding its instructions, this process should be real-time
- high tolerance to bad weather conditions — fog, rain and snow are very common conditions in various parts of the world, an autonomous vehicle should be able to function correctly in any of these conditions. This point ties into the low missclassification rate point
- high tolerance to bad lightning — similar to the above
- low hardware cost — cars are already expensive, by including expensive high resolution cameras or expensive processors to run the detection and classification steps, the costs would be very unreasonable for the vast majority of people

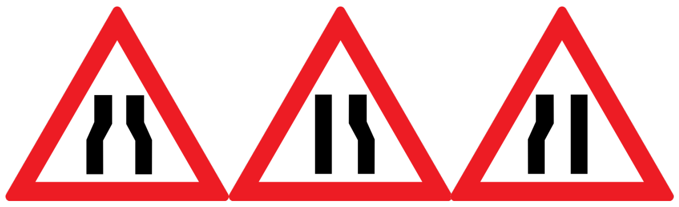


Fig. 1. Three confusable signs, in order from left to right: ”Îngustarea șoselei din ambele sensuri”, ”Îngustarea șoselei din dreapta”, ”Îngustarea șoselei din stânga”[2]

Our initial goal was to implement a real-time detection model, however, our work morphed into more of an exploratory study of the field.

A. Related work

We've surveyed a number of papers in this domain in order to discover what approach we should follow in our own application.

Reference [1] was interesting to read but ultimately we found the approach in this paper to be limiting as it could only detect road signs with red borders that were either circles or triangles.

Reference [2] proposed an approach that had multiple layers of preprocessing before the detection and classification itself, which was done using the YOLO (You only look once) architecture.

Reference [3], a metastudy on the state of computer vision in traffic sign detection and recognition in 2019, was very useful as it showed us approaches that would be likely be useful with good accuracy and clear paths to take in development. It confirmed to us that an approach where the detection and recognition steps were split was a good idea as it is

[2]By Gigillo83, own work CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=40362644, https://commons.wikimedia.org/w/index.php?curid=40362647, https://commons.wikimedia.org/w/index.php?curid=40362645, stitched by Mițca Dumitru into a single image under the same terms.

what most of the literature surveyed did. We initially believed that we could use grayscale datasets to simplify our machine learning (ML) models, but [3] infirmed this belief showing us that colour-based approaches would have higher accuracy. Reference [4] claims that a grayscale approach would be based on shape detection and be computationally expensive.

We approached [5] due to its generic title, but it is an application in traffic sign inventory management, and as such has less strict requirements than our application, and a much higher error rate is acceptable in its domain.

Reference [6] presents an approach where the detection of road signs is done using classical segmentation methods, while the recognition itself is done using a multilayer perceptron neural network.

Reference [7] presents an approach based on HOG[3] features and linear support vector machines[4] (SVMs). The paper proposes an approach where the HOG and SVM -based step is used for traffic sign detection and a convolutional neural network is used for classification. Our final approach is similar to the one proposed by this paper.

Conclusions:

- this kind of application should be split in two separate modules: a detection module and a recognition module, with any number of pre-processing before them
- the detection module can be implemented using classical methods, while the recognition module must be implemented using neural networks
- grayscale approaches are not computationally viable

B. State of the art

For existing software solutions we surveyed the pretrained YOLOv8 [8][5] model YOLOv8n and Roboflow.

1) YOLOv8: Through experimentation using the pretrained YOLOv8 model, we came to the conclusion that pretrained models are unlikely to be useful to particular applications, due to the pretrained models' generality and datasets where weather and or lightning conditions aren't varied enough. We believe using similar, or the same, architectures used by pretrained models or fine-tuning them is a good path forward however.

2) Roboflow: Roboflow[6] is a freemium service that provided us with two important services:

- a dataset service that allows one to annotate datasets and also allows conversions between different dataset formats used by various preexisting architectures
- a model preview service, in which Roboflow will use your dataset to train a model that you can try in your browser — this service can be used a maximum of 3 times per project for free

However the freemium nature of the service is a disadvantage as one might invest a lot of time in this service due to its apparent free nature and latter be forced to buy-in when that might not have been the original intention.

We also noticed that there was considerable delay in using the models in one's browser, we are however not sure if this is a limitation of the models trained by Roboflow, the hardware on which they run, or the nature of the internet connection to it.

[3]Histogram of Oriented Gradients

[4]See Appendix A for a summary on SVMs

[5]You Only Look Once

[6]Roboflow can be accessed at https://roboflow.com/

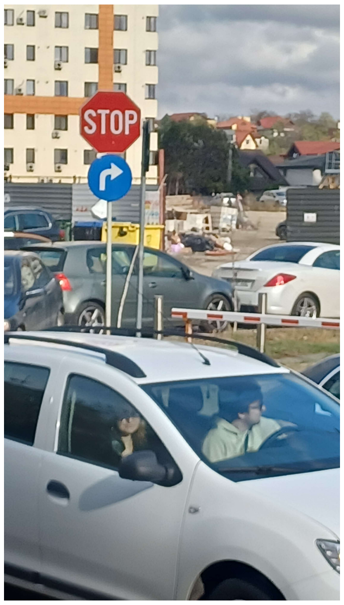


Fig. 2. YOLOv8n with no additional training detects 2 stop signs in this image

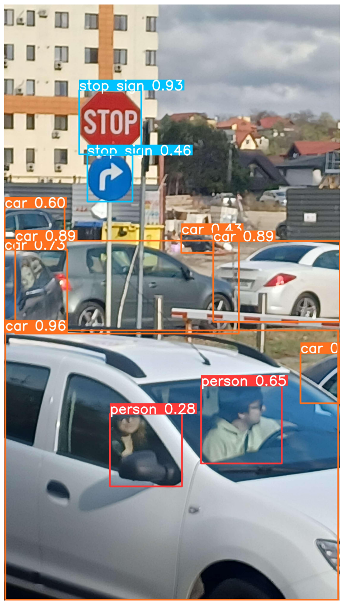


Fig. 3. Figure 2 with labels, as can be seen, the "must go right" sign is misclassified as a "stop sign"

II. Method Description

We propose an approach in which the process is split in two steps: a detection step and a classification step, following

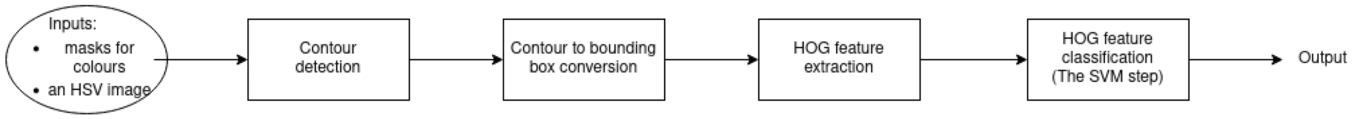


Fig. 4. Block diagram for the SVM-based detection step

existing practices in literature.

The detection step takes in an image, detects possible objects in the image that might be road-signs (based on color, shape, etc.). The classification step takes in an image containing only a possible road-sign and classifies it into the kind of road sign it is, or classifies it as a non-road-sign if the detection step misdetected an object as a road-sign. These steps are chained together using a middle layer that does any needed preprocessing on the segments from the original image given by the detection step, and then feeds them to the classification step.

A. The detection step

We've explored three approaches to detection in our work on this paper: a YOLOv8-based approach, a fully classical approach and the SVM-based approach described in [7].

The YOLOv8-based approach: We've chosen a neural-network-based approach for our detection step in order to see the what differences would arrise from existing approaches taken in literature, but also in part due to perceived ease of development[7].

We've decided to use a model based on the YOLOv8 architecture that we've trained on various datasets, but have yet to find any success. Next we will explore a more classical approach to, hopefully, obtain more success in the detection step.

The purely classical approach: Our work on the classical approach was not a great success and did not yield great results. This was likely due to misuse of the tools we used and incomplete understanding.

The SVM-based approach: We generated HOG features for the images offered by the German Traffic Sign Recognition Benchmark (GTSRB) [9] and trained an SVM, specifically one using a Linear Support Vector Classification algorithm, using facilities provided by scikit-learn [10].

[7]Reality often differs from one's expectations.

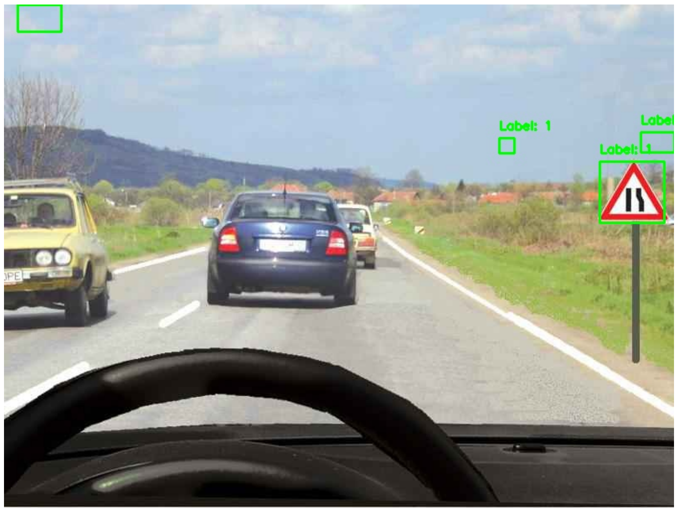


Fig. 6. An image with areas of interest labeled by our SVM-based traffic sign detection module

B. The classification step

We explored two approaches to CNN-based[8] classification: one was based on PyTorch [11] and one was based on Keras [12]. We used the GTSRB dataset for both approaches, as we believe it to be high quality.

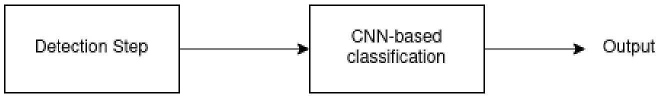


Fig. 7. Block diagram that applies to both our CNN-based approaches

1) PyTorch: The GTSRB dataset resembles the CIFAR10[9] dataset so all we had to do is take an existing network that worked well on CIFAR10 and tweak it to work for our dataset. For training we used PyTorch [11] and got over 94% accuracy on the test set. Unfortunately we highly suspect that our implementation was heavily overfitted on our training set, and as such became unable to recognize traffic signs in new images.

2) Keras: Our Keras-based CNN is a self-made implementation of the one mentioned in [7], and as such it uses the same layers:

[8]Convolutional Neural Network, see APPENDIX B for more information
[9]More information at https://www.cs.toronto.edu/~kriz/cifar.html

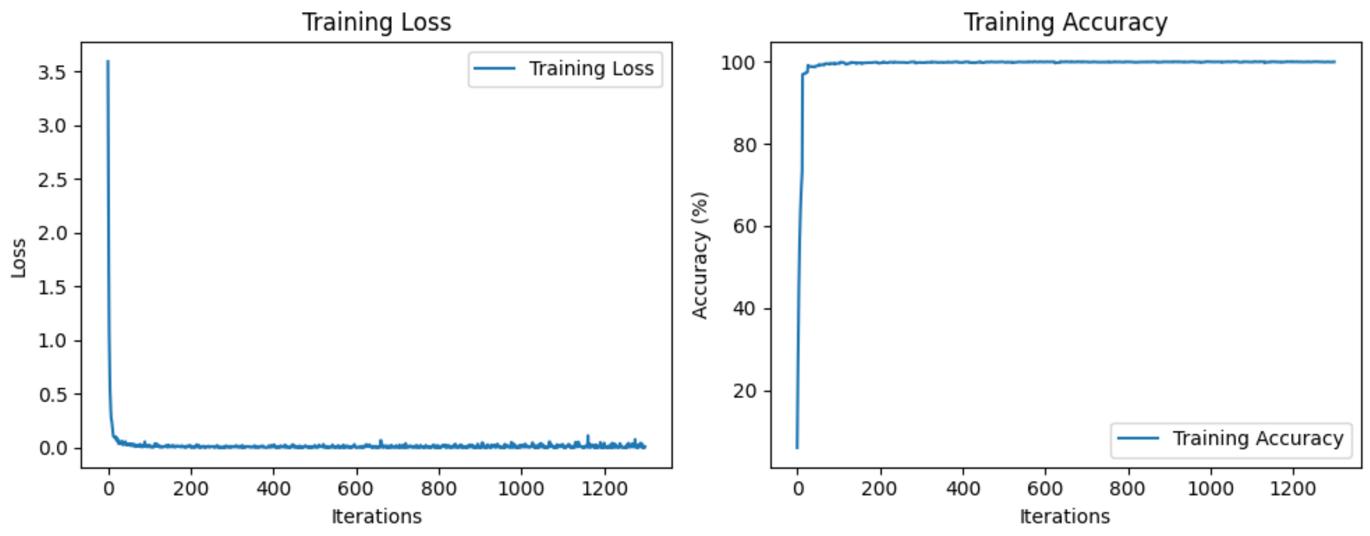


Fig. 5. Plot of the PyTorch-based classification training progress showing a surprisingly rapid increase in accuracy

	Layer	Output Size	Parameters
1	Input	56 × 56	—
2	Convolution	96 × 52 × 52	kernel = 5 × 5; strides = 1 × 1
3	Max Pooling	96 × 26 × 26	kernel = 2 × 2; stride = 2 × 2
4	Convolution	140 × 24 × 24	kernel = 3 × 3; strides = 1 × 1
5	Max Pooling	140 × 12 × 12	kernel = 2 × 2; stride = 2 × 2
6	Convolution	240 × 10 × 10	kernel = 3 × 3; strides = 1 × 1
7	Max Pooling	240 × 5 × 5	kernel = 2 × 2; stride = 2 × 2
8	Flatten	6000	—
9	Densely-connected[a]	150	—
10	Dropout	150	—
11	Densely-connected	43	—

[a]Called ”Fully connected” in the paper

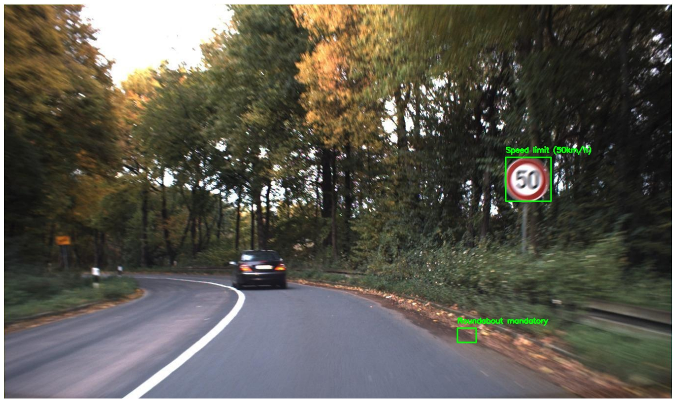


Fig. 8. Results of the final system, using the SVM-based detection module and the Keras-based classification module, sadly there are still false positives in the results, as can be inferred from the accuracy reported by the training process

The ReLU[10] layers from the original papers are missing, as they're merged with the Convolution layers in our Keras implementation. The LRN[11] layers are missing as they are not built-in to Keras and we decided to not reimplement LRN ourselves. The softmax layer from the original paper is part of the densely-connected layer.

III. Results

When we tested the approach that made use of a PyTorch-based CNN, we got over 94% classification accuracy, and aproximately 80% on segmentation but it perfomed badly on our personal test, that being showing a video of a car driving in town, or no detections on 2. The training results can be seen in figure III and the confusion matrix can be seen in III.

For our Keras-based CNN, after training, our model showed an accuracy of 98.96%, which is similar to the results shown in Table 5 §4.3 [7], where the various CNNs shown show a mean accuracy of 99.11% and a standard deviation to accuracy of 0.64. These results are similar (but slightly better) to the ones of the MPPs-CNN mentioned by [13], however are approximatively 0.2% worse than the ones of the CNN proposed by [7], despite our best efforts to follow the paper, this could be due to mistakes in our development process, or different libraries having slight differences in how the algorithms are implemented.

[10]Rectified Linear Unit activation function

[11]Local Response Normalization

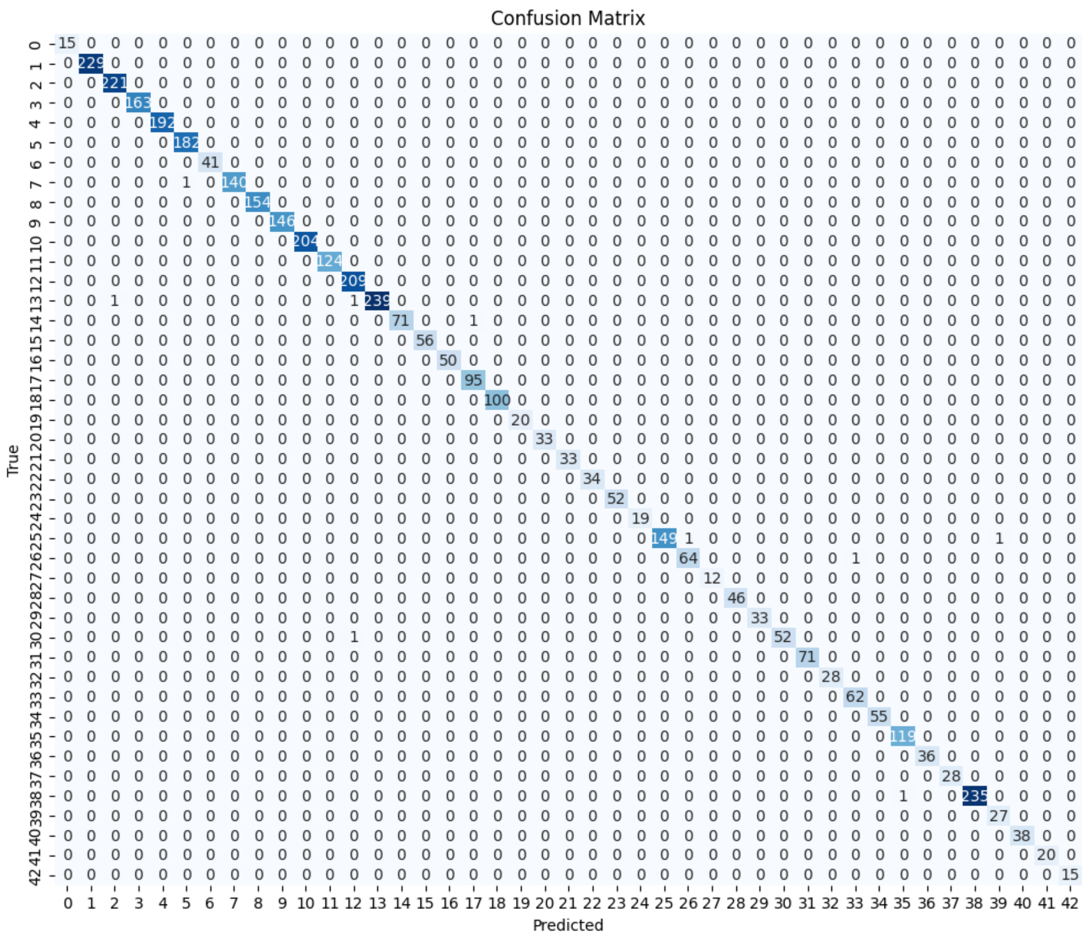
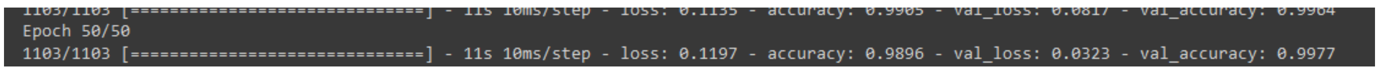


Fig. 9. The confusion Matrix for one of our Keras-based CNN



```
1103/1103 [==============================] - 11s 10ms/step - loss: 0.1135 - accuracy: 0.9905 - val_loss: 0.0817 - val_accuracy: 0.9984
Epoch 50/50
1103/1103 [==============================] - 11s 10ms/step - loss: 0.1197 - accuracy: 0.9896 - val_loss: 0.0323 - val_accuracy: 0.9977
```

Fig. 10. Results of training the Keras-based CNN

IV. Conclusions

Neural-network-based approaches are very accessible — *in theory* — but getting good results out of them is difficult due to not being aware of mistakes that can lead to overfitting or due to the powerful hardware needed for model training.

However, we do believe that real-time recognition of traffic signs becomes a more achievable goal as more and more time is passing due to this high accessibility, to people who are more experienced in the domain than us, and powerful hardware becoming more cheap with time.

We also believe that SVM-based approaches for traffic sign detection show great promise due to their nature which allows easy composition and great focus on their own task. We believe these approaches should make use of colour and shape to detect possible classes of signs — i.e. an SVM for blue, round signs, an SVM for triangle signs, etc., these would correspond to various classes of traffic signs as their shape and colour is already standardized to make recognition easy for human drivers — and compose the SVMs into one module that can then detect multiple classes.

We do not have any new insight into the classification step that would make us believe other approaches other than neural-network and machine learning approaches should be used for it.

Appendix A

A support vector machine (SVM) is a supervised learning algorithm used for many classification and regression problems, including signal processing

> medical applications, natural language processing, and speech and image recognition.
>
> The objective of the SVM algorithm is to find a hyperplane that, to the best degree possible, separates data points of one class from those of another class. “Best” is defined as the hyperplane with the largest margin between the two classes, represented by plus versus minus in the figure below. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points. Only for linearly separable problems can the algorithm find such a hyperplane, for most practical problems the algorithm maximizes the soft margin allowing a small number of misclassifications. [14, Support Vector Machine (SVM) Explained - MATLAB & Simulink]

As the quoted article says, SVMs are binary classifiers. It is possible to create more complex classifiers by chaining SVMs, i.e. if SVM 1 can classify circles, and SVM 2 can classify triangles, one can build a classifier that classifies both circles and triangles, by feeding the inputs classified as ”not circles” by SVM 1 to SVM 2.

APPENDIX B

Convolution neural networks are a type of neural network, a good explanation of them is the following quote from an IBM Article:

> Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:
>
> - Convolutional layer
> - Pooling layer
> - Fully-connected (FC) layer[12]
>
> The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. [15, What are convolutional neural networks?]

REFERENCES

[1] R. Malik, J. Khurshid, and S. N. Ahmad, “Road sign detection and recognition using colour segmentation, shape analysis and template matching,” in *2007 International Conference on Machine Learning and Cybernetics*, vol. 6, 2007, pp. 3556–3560.

[2] R. Karthika and L. Parameswaran, “A novel convolutional neural network based architecture for object detection and recognition with an application to traffic sign recognition from road scenes,” *Pattern Recognition and Image Analysis*, vol. 32, no. 2, pp. 351–362, Jun 2022. [Online]. Available: https://doi.org/10.1134/S1054661822020110

[3] S. B. Wali, M. A. Abdullah, M. A. Hannan, A. Hussain, S. A. Samad, P. J. Ker, and M. B. Mansor, “Vision-based traffic sign detection and recognition systems: Current trends and challenges,” *Sensors*, vol. 19, no. 9, p. 2093, May 2019. [Online]. Available: http://dx.doi.org/10.3390/s19092093

[4] A. Broggi, P. Cerri, P. Medici, P. P. Porta, and G. Ghisio, “Real time road signs recognition,” in *2007 IEEE Intelligent Vehicles Symposium*, 2007, pp. 981–986.

[5] D. Tabernik and D. Skočaj, “Deep learning for large-scale traffic-sign detection and recognition,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1427–1440, 2020.

[6] A. Hechri and A. Mtibaa, “Automatic detection and recognition of road sign for driver assistance system,” in *2012 16th IEEE Mediterranean Electrotechnical Conference*, 2012, pp. 888–891.

[7] ——, “Two-stage traffic sign detection and recognition based on svm and convolutional neural networks,” *IET Image Processing*, vol. 14, no. 5, pp. 939–946, 2020. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-ipr.2019.0634

[8] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics,” Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[9] 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*, no. 1288, 2013.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[12] F. Chollet *et al.*, “Keras,” https://keras.io, 2015.

[13] R. Qian, Y. Yue, F. Coenen, and B. Zhang, “Traffic sign recognition with convolutional neural network based on max pooling positions,” in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2016, pp. 578–582.

[14] MatLab, “Support vector machine (svm) explained,” https://www.mathworks.com/discovery/support-vector-machine.html.

[15] IBM, “What are convolutional neural networks?” https://www.ibm.com/topics/convolutional-neural-networks.

[12] In our paper we've referred to this layer as a ”Dense layer” as this is the terminology used by Keras