



Traffic Signs Detection for Real-World Application of an Advanced Driving Assisting System Using Deep Learning

Riadh Ayachi¹ · Mouna Afif¹ · Yahia Said^{1,2} · Mohamed Atri¹

Published online: 19 September 2019

© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

Recent advanced driving systems are used as luxury tools to handle a difficult or repetitive task. One of the most important tasks is traffic signs detection that provides the driver with a global view of traffic signs on the road. A traffic signs detection application should be able to detect and understand each traffic sign. To develop a robust traffic sign detection application, we propose to use the deep learning technique to process visual data. The proposed application is used for an embedded implementation. To solve this task, we propose to use the deep learning technique based on convolutional neural networks. As known, a convolutional neural network needs a big amount of data to be trained. To solve the problem, we build a dataset for traffic signs detection. The dataset contains 10,500 images from 73 traffic signs classes. The images are captured from the Chinese roads under real environmental conditions. The proposed application achieves high performance on the proposed dataset with a mean average precision of 84.22%. Also, the proposed application can be easily used for embedded implementation because of its lightweight model size and its fast inference speed.

Keywords Advanced driving assisting system · Deep learning · Convolutional neural networks · Embedded implementation

1 Introduction

Deep learning has been deployed in a variety of applications. The deep learning models were successfully used to solve applications problems based on artificial intelligence. Deep learning and neural networks were used in different fields, like computer vision. As an example of application, we cite image processing for place recognition [1], face pose estimation [2], traffic sign recognition [3], indoor object recognition [4] and Image

✉ Riadh Ayachi
riadhayachi@yahoo.fr

¹ Laboratory of Electronics and Microelectronics (EμE), Faculty of Sciences of Monastir, University of Monastir, Monastir, Tunisia

² Electrical Engineering Department, College of Engineering, Northern Border University, Arar, Saudi Arabia

Retrieval [5] for search engines enhancement. On the other hand, many techniques were proposed to enhance the performance of deep learning models like dimension reduction using local deep-feature alignment [6] and replacing pooling layers with strides convolutions for memory and accuracy efficiency [7]. Deep learning models prove its efficiency for many applications and can be used to solve more applications problems like traffic signs detection for Advanced driving assisting system (ADAS). ADAS has existed for a long time. They are developed more and more to improve the driving experience and to ensure a safer environment. ADAS provides essential information about the driving environment to increase the safety of all the road users. ADAS are combined from a large number of sensors and cameras. The data provided by those elements are processed to provide an overview of the driving conditions. One of the most important tasks of an ADAS is to understand traffic signs used to notify humans about the driving rules. By analyzing the visual data, a traffic sign detection application can spot an interrupt traffic sign. In this paper, we propose a deep learning-based method to develop the traffic signs detection application. The priority of this work is to develop an application suitable for embedded implementation. To handle this challenge, we propose the use of the convolutional neural networks [8], a deep learning model commonly used to solve computer vision problems. In the few past years, convolutional neural networks are deployed successfully for image processing tasks like object recognition [9] and image detection [10]. Generally, a convolutional neural network is composed of 6 type of layers, an input layers which is an image in our case, a convolution layers, a pooling layers, fully connected layers and output layers. For the output layer, two types can be considered, the first is the classification output where the network provide predictions about the class of the input and the second type is the regression output where the network provides a prediction about the value (for example the value of the x, y coordinate). The convolutional neural networks are the best deep learning model for visual data processing. The traffic sign detection application is combined from 2 modules. The first module is the traffic localization, where it needs to determine the coordinate of the traffic sign in the image. The second module is used to identifier the traffic sign by predicting its class. So, the traffic sign detection application needs to provide two outputs; the traffic signs coordinate and class. To handle this task, we propose to use a single convolutional neural network with 2 heads. To determine the traffic sign coordinate, we use a linear regression output. For the classification problem, the normalized exponential function (softmax) which provide the probability of each class. So, the class with the highest probability is the predicted class. To build the traffic sign application, we propose to use a convolutional neural network called 'MobileNet' [11]. The MobileNet models are developed for embedded purpose thanks to its lightweight model size. Also, the MobileNet provides a perfect balance between inference speed and accuracy. Based on the advantages of the MobileNet for embedded implementation, we deploy it in the traffic signs detection application for the classification module. For the localization module, we introduce an all convolutional neural network with two outputs, a binary classifier to predict that there is a sign or not and a regressor to produce bounding boxes of positive classified regions. The localization module is integrated inside the classification network to work as a single network with two outputs. As known, The performance of the deep learning models increases with the training data amount. To train the proposed convolutional neural network, we propose a dataset for traffic signs detection benchmark. The dataset contains 10500 images from 73 traffic signs classes. The dataset was divided into training and testing data to train and evaluate the performance of the proposed model. The evaluation of the proposed model was based on the pascal voc [12]

evaluation metric. The proposed model achieves high performance in traffic sign detection with 84.22% of mean average precision and an inference speed of 0.1 s per image.

The rest of the paper is organized as follow, in Sect. 2, we provide an overview of related works to the traffic signs detection applications. Section 3 will be reserved to detail the proposed model and the dataset. In Sect. 4, experimental results are provided and discussed. Section 5 will conclude the paper.

2 Related Works

The traffic signs detection application was and still a hard challenge to solve. Many works are proposed to attend a high performance. The deep learning techniques achieve brilliant performance in comparison with old techniques in the field of computer vision and visual data processing. In the general contest, numerous deep learning models are proposed for object detection. The R-CNN [13] model was the first deep learning-based model for multi-class object detection. The R-CNN model uses the selective search technique [14] for region proposal and a convolutional neural network for features extraction and uses support vector machines as a classifier. The main idea of the R-CNN model is to classifier each region proposed by the selective search technique. The R-CNN model achieves better performance than the old technique, but it was prolonged, with more than 50 s per image. Many improvements were applied to the R-CNN model to enhance performance. Fast R-CNN [15] model propose to process the whole image and then extract proposed regions. Also, it eliminates the support vector machines and uses softmax classifiers. The proposed changes reduce the inference speed to 2 s and enhance accuracy. The Faster R-CNN [16] apply some changes to the fast R-CNN by replacing the selective search technique by a convolutional neural network called 'Region Proposal Network (RPN)' [16] to generate a region of interest. The RPN uses the features extractor of the classification network. It proposes an alternative training process. First, it trains the RPN. Second, it uses the optimized parameters to train the classier. Third, it retrains the RPN and fine-tunes it using the optimized parameters of the classifier. Finally, it uses the optimized parameters to fine-tune the classifier. The proposed changes improve the accuracy and reduce the inference time to 0.2 s. The R-CNN family achieve its performance limit with the Faster R-CNN model. In order to achieve better performance, other deep learning models are proposed. The single-shot multi-box detector (SSD) [17] was proposed to achieve a balance between accuracy and speed. The SSD model eliminates the regions proposal technique and replaces it with a set of default bounding boxes from different scales and aspect ratio. The network combines predictions from different levels on different features of maps dimensions. As output, the model computes the probability of all the object classes in each default bounding box and adjust the parameters of each bounding box. The SSD model achieves high performance on the pascal voc [5] object detection challenge with a mean average precision of 76.9%. Another well-known model for object detection is YOLO (You Only Look Once) [18–20], which is developed with speed priority. The main focus of the YOLO model is accelerating the inference speed and achieving acceptable accuracy. The basic idea of the YOLO model is to divide the image into a grid of size $S \times S$ dimension, for each grid cell a bounding boxes (B), a confidence score and a class probabilities (C) were provided. The YOLO model solves the detection task as a regression problem. The model output is a tensor with $S \times S \times (B \times 5 + C)$ dimension. The YOLO model achieves high performance in term of inference speed with 0.02 s per image and 63% of mean average precision. Many of those

models were used for traffic sign detection. As an example, the work proposed in [21] performs the traffic sign detection by using the SSD (single shot multibox detection) model. A number of modifications were applied to the used SSD model to achieve high performance in the traffic signs detection on high-resolution images. The proposed method was used to detect small traffic signs in high-resolution images. To do that, a small-object-sensitive-CNN technique was applied on image pyramid. Then, all image levels are projected on the original image scale to provide predictions. Zhang et al. [22] propose an end-to-end convolutional neural network based on the YOLO model [19]. The proposed end to end CNN achieve real-time implementation and acceptable accuracy. In addition, many custom models are developed for traffic signs detection. Zhu et al. [23] proposed an end-to-end CNN for traffic sign detection and classification. The proposed method achieves high performance when evaluated on the proposed dataset called ‘Tsinghua-Tencent 100 K’ [23] by achieving 88% of accuracy in the classification and 84% in the detection. All the proposed models for traffic signs detection achieve good performance, but none of those models developed for embedded implementation.

3 Proposed Model and Dataset

In this section, we will detail the proposed model for the traffic sign detection application and the used dataset for training and testing. The proposed model is an end to end the convolutional neural network with two outputs. It was based on the MobileNet [4] as a classifier and an integrated all convolutional neural network for traffic sign localization. The MobileNet model is designed for implementation on mobile devices with a focus on the inference speed and small size. The MobileNet model introduces the use of the Depthwise Separable convolutional blocks. The main idea of using those blocks is to build deep convolutional neural networks with lightweight model size. Generally, a depthwise separable convolution block is combined from 2 main blocks. The first is a depthwise convolution followed by batch normalization and a rectified linear unit 6 (ReLU6). The second is a pointwise convolution followed by batch normalization and a rectified linear unit 6 (ReLU6). Figure 1 presents the depthwise separable convolution blocks used in the MobileNet model.

The depthwise convolution is the same regular convolution layer where a kernel was applied to all of the input image channels. The kernel slides across the image and performs a weighted sum of the input pixels covered by the kernel across all input channels at each step. However, the main difference is that the depthwise convolution operation combines the values of all the input channels. In effect, if the input feature map has three channels, then running a single depthwise convolution across this feature map results in an output feature map with only one channel. So, for each input feature map, no matter how many channels it has, the depthwise convolution new output is a feature map with only a single channel. The depthwise convolution purpose is to filter the input feature maps. Also, the MobileNet introduces the use of a custom activation function which is the ReLU6. The ReLU6 is regular activation function, but it prevents activations from becoming too high by limiting the max activation value to 6. Assuming x is an input and y is output, the ReLU6 is computed as (1).

$$y = \min(\max(0, x), 6) \quad (1)$$

Another innovation proposed by the Mobilenet model is the pointwise convolution with is a regular convolution layer with 1×1 kernel size. The main function of the

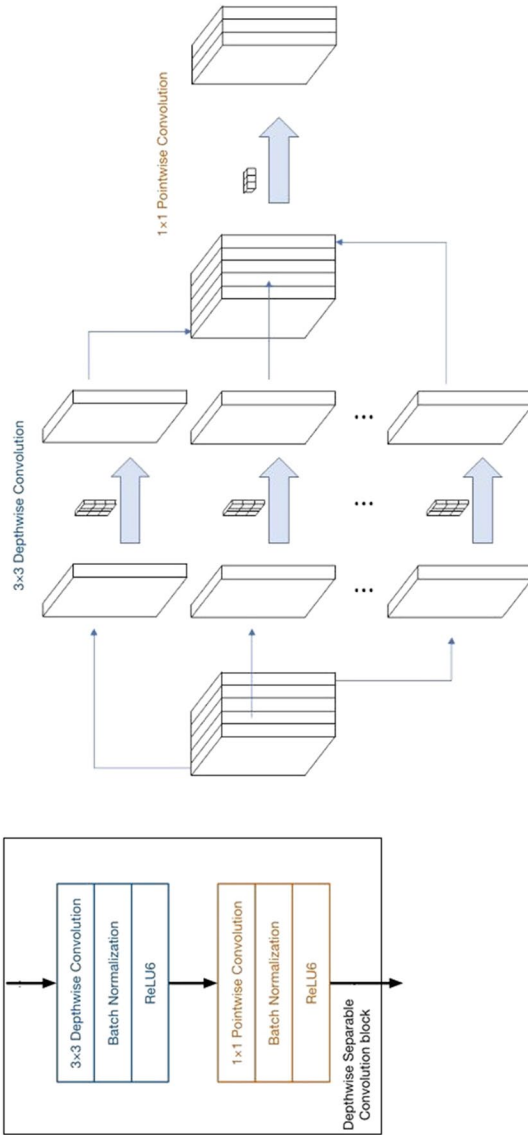


Fig. 1 MobileNet depthwise separable convolution blocks

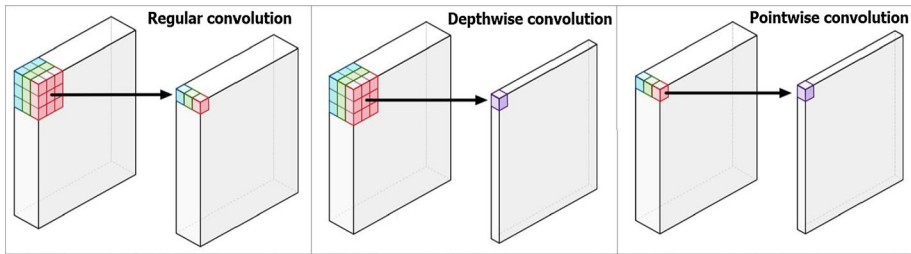


Fig. 2 Different convolution types used in MobileNet

pointwise convolution is to combine the output feature maps of the depthwise convolution to create new features. Figure 2 presents the difference between regular convolution, depthwise convolution and pointwise convolution. The combination of the depthwise convolution and the pointwise convolution provides the depthwise separable convolution block, unlike regular convolution that filter and combine features simultaneously, separate the filtering and the combining operations. Both regular and depthwise separable convolution blocks filter and combines features to generate new ones, but a regular convolution consumes more computation resources and learns more weights.

The depthwise separable convolution block does the same function of the regular convolution layer but faster. Depthwise convolution with 3×3 kernel size can be computed nine times faster than the regular convolution. Also, the MobileNet model uses regular convolution as the first hidden layer just after the input image. The MobileNet features extraction part ends with an average pooling layer. For the classification, a fully connected layer was deployed, followed by a softmax layer. To summarize, the MobileNet model is composed of a regular convolution layer followed by 13 depthwise separable convolution blocks and an average pooling layer then ends with a fully connected layer and a softmax layer for the classification. The MobileNet model is presented in Fig. 3.

The MobileNet model is used for features extraction and classification. To solve the detection task, a localization module needs to be added. For this purpose, we propose a fully convolution network with two outputs. The first output is a binary classifier to predict if there is a sign or not and the second output is regressor to predict bounding boxes parameters. This network is connected to the last feature map of the Mobilenet feature extraction part. So, it uses the same features of the classifier. By sharing the feature maps between the classifier and the localization network, we reduce the network complexity. The localization network is composed of 3 convolution layer, softmax layer and a linear regression layer. The first convolution layer is connected to the last feature map of the MobileNet. 3×3 kernel was slide across the feature maps with a stride of 1 to collect useful information. Then, two separated convolutions are used. The first convolutional layer is followed by a softmax layer to predict the selected region if it is a sign or not. The second convolution layer is followed by a linear regression layer where different anchor scales are tested on the selected region to generate bounding boxes. Since traffic signs have circle and triangle shapes, we propose to use square anchors with three different scales. After that, the binary classier output was combined with the regressor output and each bounding box was judged if it contains a sign or not.

To train the localization binary classifier, the tested anchor bounding box is projected on the input image and calculating the IoU factor. The IoU factor is the division of the

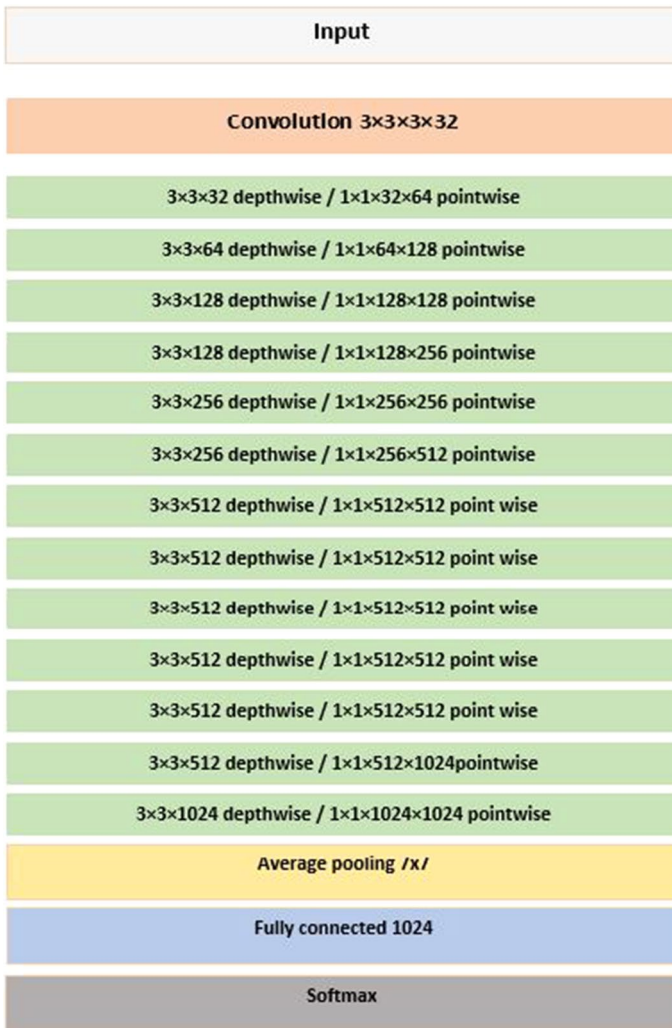


Fig. 3 MobileNet model for classification

intersection between the proposed bounding box (tested anchor) and the ground truth bounding box by the union of the same bounding boxes. If the IoU is under 0.3 the network classifies the region as false (no sign on it) and if the IoU is equal or more than 0.7, the region is classified as true (there is a sign). In the other side, a regression layer is used to determine the bounding box parameters (x, y coordinate, height and width). If the selected region was classified as true, the bounding box parameters are sent to the detector to predict the traffic sign class and fine-tune the bounding box parameters. The localization network is illustrated in Fig. 4.

As results, the localization network was able to predict if there is a sign or not in the image and generate only bounding boxes which contain signs to reduce the processing time at the detection stage. In effect, instead of processing the whole image at the detection stage, it processes only the proposed regions to predict the class and fine-tune the bounding

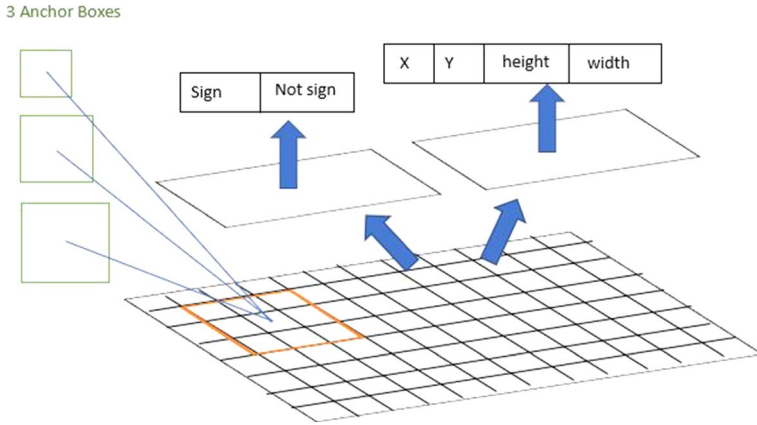


Fig. 4 Localization network

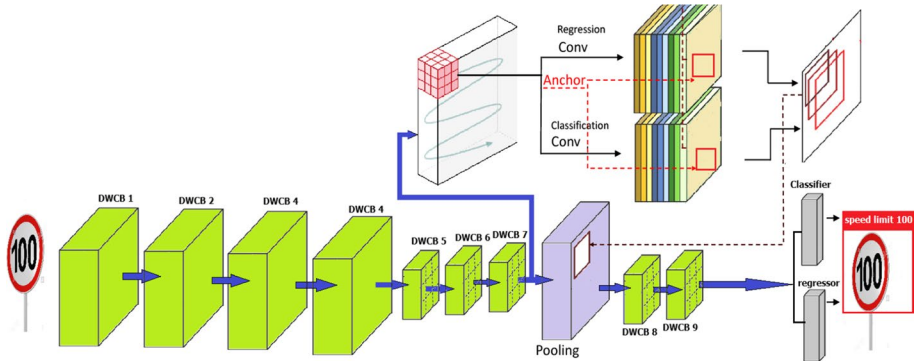


Fig. 5 Proposed traffic signs detection model

boxes parameters. A special pooling layer is deployed to connect the localization network output with the detection stage. This pooling layer fixes the dimension of the output feature maps to connect them with the next stage. The proposed network for traffic signs detection is composed of 9 depthwise separable convolution blocks (DWCB), a localization network and two separated outputs, an output for class prediction and output for bounding boxes prediction. Seven depthwise convolution blocks are used before the localization network and the two others are used after. Figure 5 presents the proposed model for the traffic signs detection application. The non-maximum suppression technique [24] was deployed to choose the perfect bounding box. The main difference between the proposed model and existing models is the use of an end-to-end convolutional neural network with two outputs to perform the detection of the traffic signs in real images and the use of a fully convolutional neural network to generate proposed regions that only contain traffic signs and that accelerate the inference time. In addition, the proposed model is suitable for embedded implementation because it provides a good balance between speed and accuracy. An existing model like the Fast R-CNN use and external module to provide region proposals then it processes the whole image using a convolutional neural network, extracts the proposed

regions and classifier each of those regions to determine that its class. This methodology is costly in term of speed and accuracy since the external module propose random regions and the classifier need to process all those regions. The best inference speed of the fast R-CNN is 2 s per image. Also, the YOLO model solves the object detection task as a regression task. Yolo achieves a good inference speed of 0.02 s per image, but its accuracy is limited and it struggles in detecting small object and objects situated in the same receptive field (objects close to each other). In our case traffic signs are tiny objects in comparison of the total image size where it occupies less than 15% of the total size in pixels. So, the proposed model can be perfectly used to solve the problem.

To train the proposed model, we propose a traffic signs detection dataset. In this work, we build traffic signs dataset where the images are collected under real-world conditions. All the appeared traffic signs classes were considered in this dataset and we end up by 73 traffic sign classes. The images were captured from the Chinese roads by [25] and it was initially used for three supper classes classification. The three superclasses cannot be used for an ADAS because each sign of the supper class have a different mean except for the danger superclass, which all mean warning. In effect, in our work, we labeled each traffic sign and assigned it with the independent class. Figure 6 presents the new dataset classes. All the dataset images are converted to the jpg format because the original dataset contains PNG format and BMP format. In addition, more images are added to balance the number of instances per class and to ensure more than 100 instances per class. The dataset was annotated manually using the LabelImg tool [26]. The generated annotation files were on the XML format like those used in pascal voc (The PASCAL Visual Object Classes) challenge [27]. The labeling process was made by drawing a bounding box around the traffic signs and assign each bounding box with the label of the traffic sign. To make dataset perfect for real condition evaluation the images were captured under real-world conditions like day and night illumination, a different point of view of the traffic signs, occlusion and the effect of environmental factors (rain, sun, dust ...).

Until today, the maximum number of classes in a dataset for traffic sign detection is 45. It was proposed by Tsinghua-Tencent 100 K benchmark [28]. This dataset was collected specially for traffic signs detection benchmark. The dataset contains 10,000 positive examples (images contain traffic signs) captured from 300 Chinese cities and the high ways between them. In the proposed dataset, the number of classes was raised to 73 class to give the intelligent vision system more information about all the traffic signs that can spot. The number of classes is an important factor for the evaluation of the performance of an ADAS.

The existence of the new Nvidia platforms for autonomous cars makes it easy to deploy a deep learning model in the vehicle computer. As examples, the Nvidia drive PX2 and AGx, which are two hardware designed specifically for autonomous vehicles. In our test, a hybrid system composed of an Intel CPU and an Nvidia GPU with approximately the same performance of the Nidia drive AGx. In the next section, more details will be provided about the user system to test the performance of the proposed model.



Fig. 6 Dataset classes

4 Experiment and Results

The experimental environment of our work is composed of an MSI PRO Series desktop with an Intel i7 processor and an Nvidia GTX960 general-purpose graphics processing unit. The TensorFlow deep learning framework was used to develop the proposed model and to build the traffic signs detection application. To train and evaluate the proposed model, the dataset was divided into two parts. The first part is the training data, which is 80% of the data and the second part is the testing data, which is 20% of the data. The dataset was splinted according to the most used protocol in deep learning techniques. As known, the deep learning models improve their performance if more data was used in the training process [29]. In this work, a data augmentation technique was deployed to increase the training data by reusing the proposed bounding boxes as input data. The use of data augmentation has enhanced the model performance in a significant way. All the original

Table 1 Comparison between the proposed dataset and state-of-the-art datasets

Dataset	Number of images	Number of classes
GTSDDB [31]	900	43
Tsinghua-Tencent 100 K [23]	10,000	45
CTSD [25]	10,000	3
Sweden dataset [32]	3777	45
Ours	10,500	73

and generated images using the data augmentation are resized to 1000×600 pixel with three channels RGB color space.

To train the proposed model, the back-propagation technique [30] was deployed. In particular, the gradient descent algorithm was used to optimize the model parameters. To optimize the parameters, a loss function need to be optimized. The loss function is the difference between the model output and the desired output. In the training process, the model is fitted with images where the signs are labeled. The label for the detection task contains the class and the parameters of the bounding box. The learning algorithm defines a mapping function using the feed-forward operation and computes the error using the backpropagation. As a gradient descent technique, we use Adam optimizer because of its advantages in comparison with other optimizers. Adam optimizer can achieve faster and better convergence. Also, it generates its learning rate and optimizes it while optimizing the loss function. The learning rate defines the speed and quality of the convergence.

The proposed dataset contains 10,500 images and presents 73 traffic signs classes. Table 1 presents a comparison between the proposed dataset and existing datasets in term of number of images and number of classes. The new dataset is considered the most significant datasets in term of amount of data with 10,500 images when compared against state-of-the-art datasets. Also, it presents the largest number of traffic signs classes with 73 class.

To determine the performance of the proposed model, an evaluation metric need to be used. Both accuracy and precision can be used to evaluate the performance of a deep learning model. Accuracy is the sum of the true positive examples and the true negative examples devised by the total number of examples. Accuracy can give us an immediately idea whether a model is being trained correctly and how it may perform generally. However, it does not give detailed information regarding its application to the problem. The problem with using accuracy as a main performance metric is that it does not do well when the dataset presents a huge number of imbalanced classes. So, the use of precision can solve the problem. Precision is the true positive divided by the sum of the true positive and false positive. The precision can tell us how often the prediction is correct when the model predicts true positive. The Recall is used when the cost of false negatives is high. The recall is the true positive divided by the sum of a true positive and false negative. The F1 score can summarize both precision and recall. The F1 score is two times the multiplication of the precision and recalls divided by the some of precision and recall. The proposed dataset presents a big number of classes. So, the mean average precision is used as an evaluation metric to evaluate the performance of the model. The mean average precision is the average of the maximum precision of different classes. Table 2 shows the obtained average precision (AP) for each class, the mean average precision (mAP) of the proposed model, the recall and the F1 Score.

Table 2 Evaluation of the proposed model

sign									
AP	85.61%	79.22%	95.92%	37.56%	92.16%	32.13%	52.55%	86.77%	81.02%
sign									
AP	89.21%	90.11%	90.46%	85.33%	80.87%	80.94%	84.84%	94.67%	93.20%
sign									
AP	89.86%	75.67%	90.19%	90.91%	81.42%	88.68%	89.73%	100%	100%
sign									
AP	85.89%	90.65%	58.81%	87.71%	74.78%	77.60%	73.01%	100%	87.29%
sign									
Ap	90.51%	32.34%	93.76%	70.84%	69.92%	86.52%	84.10%	88.59%	97.35%
sign									
Ap	90.66%	76.14%	83.75%	92.21%	79.80%	84.85%	62.12%	98.27%	57.66%
sign									
Ap	77.11%	49.13%	37.23%	100%	60.24%	100%	85.32%	100%	75%
sign									
Ap	100%	78.21%	60.43%	90.70%	72.51%	64.52%	100%	100%	45.80%
sign									
AP	90.50%								
mAP					84.22%				
recall					89.45%				
F1 Score					86.42%				

The reported results prove the efficiency of the proposed model for traffic signs detection. The proposed model achieves 84.22% of mean average precision on 73 traffic sign classes. The number of classes is an essential factor that has a direct influence on model performance. Since our dataset has 73 class, we compare our performance against state of the art model with taking into account the number of classes used to train and test each model. Table 3 presents a comparison of the proposed model against state-of-the-art models.

On the other hand, the model is developed for embedded implementation. As shown in Table 4, the model size is only 70 MB, which is a very light size in comparison with other detection models. The light size ensures that the proposed model can be easily implemented in an embedded device.

Table 3 Comparison against state-of-the-art models in term of performance

Model	Performance	Number of classes
SOS-CNN [21]	90% of accuracy	45
Modified YOLO v2 [22]	98% of precision	3
End-to-end convnet	88% of accuracy	54
Ours	84.22% of mAP	73

Table 4 Comparison against existing detection models

Model	Size (MB)
Yolo v3	237
SSD	182
Faster R-CNN	167
Ours	70

**Fig. 7** Application output

The proposed model has been implemented in a traffic signs detection application. This application was tested on new images and proved its performance with a detection precision of more than 80% of confidence per object. Figure 7 presents the output image of the traffic signs detection application. In the other side, the application achieves an inference speed of 0.07 s per image or 15 frames per second. The achieved inference speed allows the application to run for real-time detection.

5 Conclusion

Advanced driving assisting systems have become an essential option in today cars. One of the most important tasks of an ADAS is to assist the driver to respect traffic rules. In this paper, we propose a traffic sign detection application based on the deep learning technique. The proposed application achieves high performance in both accuracy and speed. In addition, the application was developed for embedded implementation thanks to the lightweight size of the deep learning model use for traffic sign detection. As future works the proposed model will be implemented on an embedded Nvidia platform and test in real application.

References

1. Yu J, Zhu C, Zhang J, Huang Q, Tao D (2019) Spatial pyramid-enhanced NetVLAD with weighted triplet loss for place recognition. *IEEE Trans Neural Netw Learn Syst.* <https://doi.org/10.1109/TNNLS.2019.2908982>
2. Hong C, Yu J, Zhang J, Jin X, Lee K-H (2018) Multi-modal face pose estimation with multi-task manifold deep learning. *IEEE Trans Ind Inform* 15(7):3952–3961
3. Ayachi R, Said Y, Atri M (2019) To perform road signs recognition for autonomous vehicles using cascaded deep learning pipeline. *Artif Intell Adv* 1(1):1–10
4. Afif M, Ayachi R, Said Y, Pissaloux E, Atri M (2018). Indoor image recognition and classification via deep convolutional neural network. In: International conference on the sciences of electronics, technologies of information and telecommunications. Springer, Cham, pp 364–371
5. Yu J, Tao D, Wang M, Rui Y (2014) Learning to rank using user clicks and visual features for image retrieval. *IEEE Trans Cybern* 45(4):767–779
6. Zhang J, Yu J, Tao D (2018) Local deep-feature alignment for unsupervised dimension reduction. *IEEE Trans Image Process* 27(5):2420–2432
7. Ayachi R, Afif M, Said Y, Atri M (2018) Strided convolution instead of max pooling for memory efficiency of convolutional neural networks. In: International conference on the sciences of electronics, technologies of information and telecommunications. Springer, Cham, pp 234–243
8. LeCun Y, Kavukcuoglu K, Farabet C (2010) Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE international symposium on circuits and systems. IEEE, pp 253–256
9. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint [arXiv:1409.1556](https://arxiv.org/abs/1409.1556)
10. Ren S, He K, Girshick R, Sun J (2015) Faster r-CNN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
11. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint [arXiv:1704.04861](https://arxiv.org/abs/1704.04861)
12. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. *Int J Comput Vis* 88(2):303–338
13. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
14. Uijlings JR, Van De Sande KE, Gevers T, Smeulders AW (2013) Selective search for object recognition. *Int J Comput Vis* 104(2):154–171
15. Girshick R (2015) Fast r-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
16. Ren S, He K, Girshick R, Sun J (2015) Faster r-CNN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
17. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: single shot multi-box detector. In: European conference on computer vision. Springer, Cham, pp 21–37
18. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
19. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271

20. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint [arXiv:1804.02767](https://arxiv.org/abs/1804.02767)
21. Meng Z, Fan X, Chen X, Chen M, Tong Y (2017) Detecting small signs from large images. In: 2017 IEEE international conference on information reuse and integration (IRI). IEEE, pp 217–224
22. Zhang J, Huang M, Jin X, Li X (2017) A real-time Chinese traffic sign detection algorithm based on modified YOLOv2. *Algorithms* 10(4):127
23. Zhu Z, Liang D, Zhang S, Huang X, Li B, Hu S (2016) Traffic-sign detection and classification in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2110–2118
24. Rothe R, Guillaumin M, Van Gool L (2014) Non-maximum suppression for object detection by passing messages between windows. In: Asian conference on computer vision. Springer, Cham, pp 290–306
25. Lai Y, Wang N, Yang Y, Lin L (2018) Traffic signs recognition and classification based on deep feature learning. In: ICPRAM, pp 622–629
26. Tzutalin (2015) Labeling. Git code. <https://github.com/tzutalin/labelImg>. Accessed 1 Sept 2019
27. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. *Int J Comput Vis* 88(2):303–338
28. Zhu Z, Liang D, Zhang S, Huang X, Li B, Hu S (2016) Traffic-sign detection and classification in the wild. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2110–2118
29. Hynes N, Cheng R, Song D (2018) Efficient deep learning on multi-source private data. arXiv preprint [arXiv:1807.06689](https://arxiv.org/abs/1807.06689)
30. LeCun Y, Touresky D, Hinton G, Sejnowski T (1988) A theoretical framework for back-propagation. In: Proceedings of the 1988 connectionist models summer school, vol 1. Morgan Kaufmann, CMU, Pittsburgh, PA, pp 21–28
31. Houben S, Stallkamp J, Salmen J, Schlipsing M, Igel C (2013) Detection of traffic signs in real-world images: the German traffic sign detection benchmark. In: The 2013 international joint conference on neural networks (IJCNN). IEEE, pp 1–8
32. Larsson F, Felsberg M (2011) Using Fourier descriptors and spatial models for traffic sign recognition. In Proceedings of the 17th Scandinavian conference on image analysis, SCIA 2011, LNCS 6688, pp 238–249

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.