JST: Smart Systems and Devices
Volume 32, Issue 1, January 2022, 017-024

Development of Real-Time Traffic-Object and Traffic-Sign Detection Models Applied for Autonomous Intelligent Vehicles

Xuan-Ha Nguyen*[*], Thanh-Tung Ngo, Duy-Anh Nguyen
Hanoi University of Science and Technology, Hanoi, Vietnam
[*]Email: ha.nguyenxuan@hust.edu.vn

Abstract

Technologies for detecting traffic objects are an essential requirement for any applications in autonomous intelligent vehicles. In this work, models for detecting traffic objects were developed. Based on the existing datasets and the pre-trained models, fine-tuning techniques were applied to achieve trained models with higher accuracies even for the very challenging test data. The traffic object detection was developed based on the pre-trained YOLOv5s model. Two approaches were introduced for the traffic sign detection task. The so-called tiling technique incorporated with the YOLOv5s model was exploited in the first approach. In the second approach, a combination of the RetinaFace model for the localization and the MobileNetV1-SSD for the classification was employed. The experimental results show that all developed models release a very high rate of accuracy with a maximum AP_{50} of 75.9% for object detection and mAP_{50} of 64.2% for sign detection. Models developed via the second approach have twofold advantages in terms of accuracy and computational efficiency, which allows to deploy practical applications.

Keywords: Autonomous intelligent vehicles, deep learning, embedded hardware, object detection.

1. Introduction

In recent years, autonomous intelligent vehicles have attracted considerable interests due to their high potential for practical applications. To work effectively on real-world roads, the vehicles need a fast and accurate image processing system to gain perception of the environment. The last few years have witnessed a huge growth in image processing by taking advantage of deep neural networks. Therefore, there have been many studies focusing on exploiting deep neural networks in autonomous intelligent vehicles.

According to SAE [1], there are six different levels of an autonomous vehicle system as shown in Fig. 1. Nowadays, a majority of the vehicles using among the community are at level 0, which requires the full control of humans. At level 1, several specific systems, for example, cruise control, or automatic braking, could be separately controlled by the driving assistant system. The autonomous vehicles at level 2 offer at least two simultaneous automatic functions such as steering and acceleration/deceleration, with the expectation that the human drivers do all remaining driving tasks. In these lowest autonomous driving levels, drivers have to monitor the driving situations continuously to perform interventions. In contrast, from level 3 on, the autonomous driving system takes over the control of the vehicle in certain conditions, but drivers must intervene in the control system if it is not able to handle complex traffic situations. At level 4, a fully automated system provides all required operations of driving behaviors for vehicles. While no human interventions are required, the driver is able to take over the control and drive the vehicle manually. At the highest level, vehicles are fully capable of autonomy in all situations. The vehicles do not need any human interaction anymore and could work independently.

Two essential tasks of an image processing system in an autonomous intelligent vehicle are the traffic object detection and the traffic sign detection. Several studies have been conducted on the traffic object detection [2-4]. Investigations on the traffic sign detection are also focused [5]. Common characteristics of these works are the combination of traditional computer vision techniques with advanced deep neural networks for the image processing. Many optimizations have been introduced in order to improve the accuracy, the computational efficiency as well as the robustness again dynamic changing environments. Optimizations can be: i) the use of the multi-sensor fusion to obtain multispectral images, which allows to robustly detect various types of objects such as cars, people, and bicycles in various conditions such as daytime and nighttime [2]; ii) the exploitation of lightweight deep neural networks to achieve computational efficiencies for real-time applications [3]; iii) the re-design of the network architecture to accurately detect objects under complex scenarios including diversified object and background appearance, motion blur, adverse weather conditions, and complex interactions among objects [4].

ISSN 2734-9373
https://doi.org/10.51316/jst.155.ssad.2022.32.1.3
Received: April 16, 2021; accepted: August 25, 2021

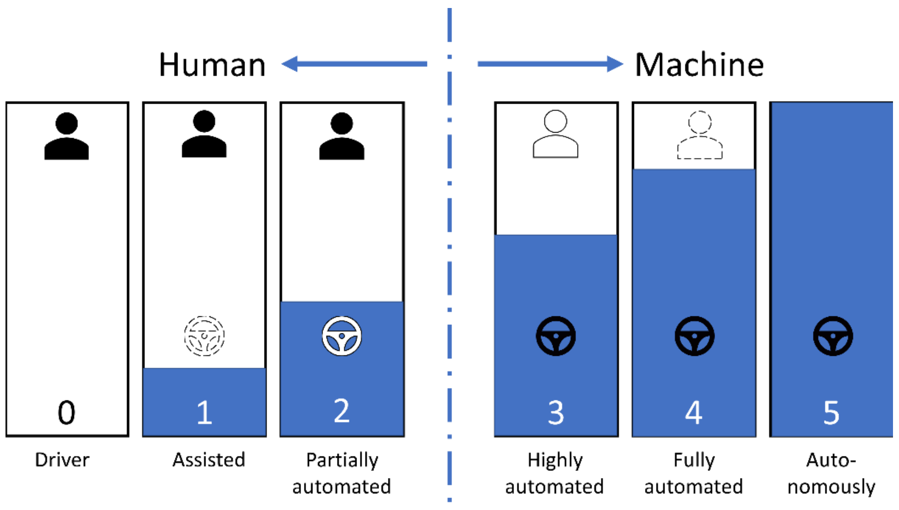

Fig. 1. Autonomous driving system levels

It is visible that three important characteristics of the traffic object/sign detection are the high accuracy, the robustness again challenging environments, and the fast response time under the condition of the limited-resource hardware. These characteristics are really challenging to obtain, especially for the bad traffic infrastructure/condition as it is in Vietnam. Therefore, the development of traffic object/sign detection models under such kind of condition is essentially needed.

In this work, deep learning models for the two crucial tasks of an image processing system in autonomous vehicles, including traffic object detection and traffic sign detection are proposed. In the case of traffic object detection, the transfer learning technique was applied to re-train a state-of-the-art pre-trained real-time object detection model with a dataset of traffic objects, namely the Cityscapes [6]. To deal with the traffic sign detection problem, two different approaches were developed. In the first approach, the fine-tuning technique on a pre-trained deep neural network, namely YOLOv5s [7], was combined with the tiling technique [8]. Two pre-trained models were fine-tuned and incorporated in the second approach, where one model, namely RetinaFace [9], is used to localize the signs and the other, namely MobileNetV1-SSD [10], is applied for the classification. The developed models were fine-tuned and evaluated based on a very challenging dataset, namely Zalo [11], which consists of images having very tiny traffic signs inside. In addition, the models for the two traffic sign detection approaches were deployed into limited-resource embedded hardwares, with the method similar to the processing block in [12]. The performance of the embedded hardware when running the models of the two approaches was measured in order to verify whether they are feasible for real-time applications in autonomous intelligent vehicles.

2. Methods

2.1. Fine-Tuning Deep Learning Models

In this work, the transfer learning approach was chosen since the datasets for each problem have only thousands of high-quality labeled images. Fig. 2 illustrates the process of our approach for transfer learning object detection models. As the first step, an open dataset is selected for each specific problem. In this step, a few pre-processing techniques are employed to convert the dataset into a suitable format for an object detection network. The dataset is then divided into subsets such as the training, the validation and the test. In the second step, a pre-trained object detection model is chosen depending on the advantages and disadvantages of the network and the requirements of the problem. The pre-trained model is trained with the prepared dataset in the third step. After finishing the training, the model is evaluated on the test set of the dataset. If the precision of the model on the test set is not good enough, the trained model must be improved by selecting another network or changing the structure of the network.

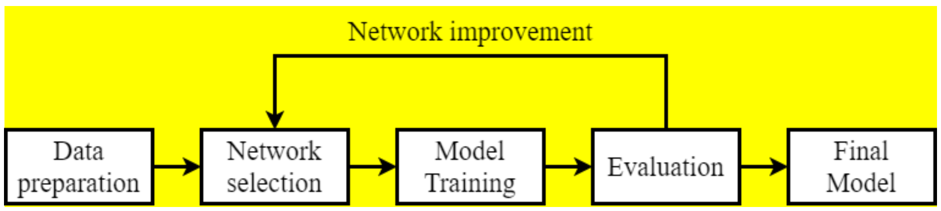

Fig. 2. Workflow of transfer learning

2.1.1. Traffic object detection

For the data preparation, the so-called Cityscapes dataset was chosen because it is one of the most diverse 2D traffic object detection datasets, compared to other published open datasets such as the KITTI dataset, and the Apollo Scapes dataset. The Cityscapes dataset consists of a huge number of video sequences of the daily life traffic activities in fifty different cities in Germany. The videos were recorded at all seasons of the years as well as under different weather conditions. Additionally, objects in the dataset also have a variety of shapes and sizes. Five thousand images of the dataset were manually selected from twenty-seven cities that have high-quality instance-level semantic annotations. For the training purpose, these images were divided into 3 sets: the training set (2975 images), the validation set (500 images), and the test set (1525 images). Since the annotation format of the Cityscapes dataset is a polygon, it was converted to the format suitable for training the YOLO network.

Table 1. Transfer learning configurations of YOLOv5s models in two applications

Parameters	Traffic object detection	Traffic sign detection
Model	YOLOv5s	YOLOv5s
No. of images	2975	4000
Image size	1024x1024	640x640
Batch size	30	160
Epochs	50	50
Learning rate	0.001	0.005
No. of classes	7	7

YOLOv5s, the latest version of the YOLO series, was chosen for the traffic object detection task because it is a state-of-the-art real-time 2D object detection model. According to [7], YOLOv5s outperforms the fastest and most accurate real-time object detection models in both speed and accuracy. The model could reach the mean average precision of 54.4% on MS COCO dataset [13]. A transfer learning process on a pre-trained YOLOv5s model with the Cityscapes dataset was implemented. Seven classes of the interested traffic objects of the Cityscapes dataset were chosen for detection, including the car, the bicycle, the person, the rider, the traffic signs, the pole, and the traffic light. The configuration parameters for the transfer learning task are set according to the values of Table 1.

2.1.2. Traffic sign detection

For the traffic sign detection task, the so-called Zalo AI Challenge dataset was chosen. This dataset is very challenging but very suitable for traffic sign detection applications in the traffic condition of Vietnam. The dataset consists of 4500 images with sizes of 1622x626, which were crawled from Google Street View Map in various places in Vietnam. The task of the problem is detecting seven common types of traffic signs, including no entry, no parking/waiting, no turning, max speed, other prohibition signs, warning, and mandatory. The dataset is very challenging since the majority of interested objects are extremely small (<10x10). Furthermore, it has a wide range of scenes from urban areas to rural areas, from exceedingly crowded streets to secluded highways. The dataset was split into two subsets including the training set (4000 images), and the test set (500 images). The dataset was annotated in COCO format, so its annotation was converted into YOLO format.

It can be known that although the size of images of the Zalo dataset is large, traffic signs are very small in comparison to the whole image because the signs are very far from the camera. Furthermore, as an input for the training of the model, the images must be resized to match the dimensions of the network. This leads to the fact that the traffic signs will be very small, which extremely difficult to detect with a good enough accuracy. An efficient model, which can detect such small traffic signs as well as be able to deploy on very lightweight hardware, is needed. Therefore, two approaches were implemented.

As the first approach, YOLOv5s was selected for the traffic sign detection task. YOLOv5s, an open pre-trained model of the YOLOv5s, was fine-tuned using the Zalo dataset. During the training process, the cross-validation technique was applied to improve the accuracy of the model with the limited number of images in the dataset. Table 1 shows the configuration of the parameters of the training operation. Because there are a considerable number of small objects in the dataset, a tiling-based technique was exploited for pre-processing the dataset. The technique reduces the detail loss of the object detection process in both the training phase as well as the inference one. The workflow of the method is shown in Fig. 3. In the first step, the input image was cropped into 160x160 images using the tiling technique with a 40-stride sliding window. In the next stage, the cropped images were filtered into an image set which includes all images containing traffic signs and three times as many as that number of random images that not containing traffic signs. After that, these filtered images were resized to 640x640 because it is the input image size standard of the YOLOv5s model. Then, the YOLOv5s model was fine-tuned with the resized images.

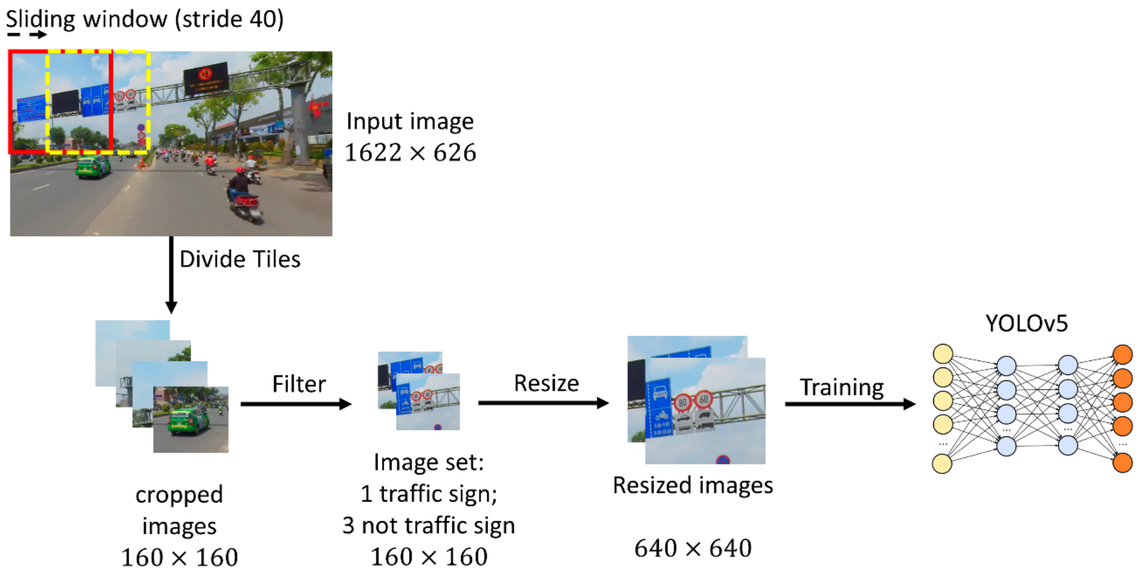

Fig. 3. Processing pipeline in the first approach for the traffic sign detection: YOLOv5s combined with the tiling technique

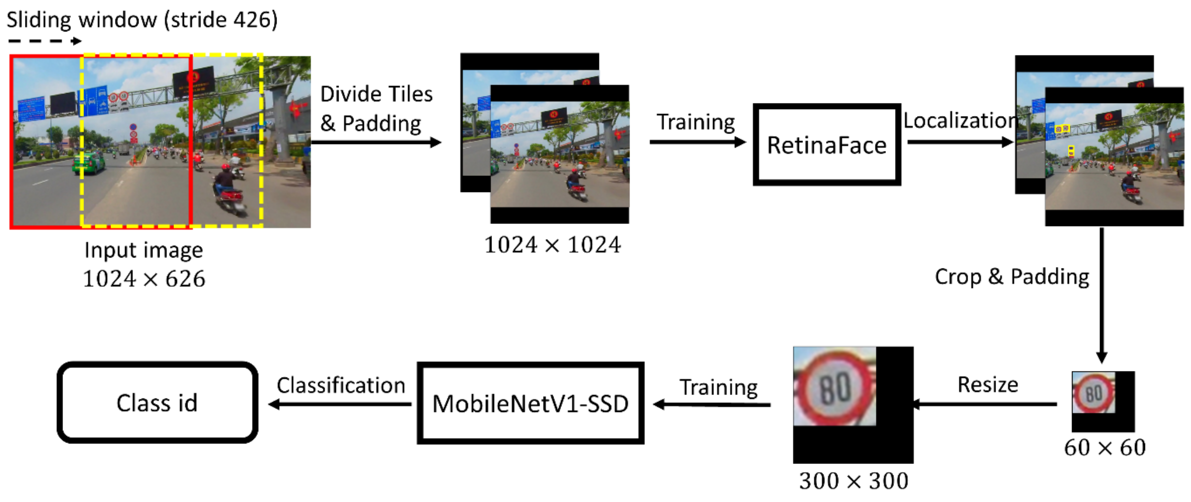

Fig. 4. Processing pipeline in the second approach for the traffic sign detection: RetinaFace [9] combined with the MobileNetV1-SSD [10]

In the second approach, utilization of the so-called RetinaFace [9] as a localization model and MobileNetV1-SSD [10] as a classification model was proposed. The approach consists of two main phases, which are illustrated in Fig. 4. In the first phase, the RetinaFace was customized for use with traffic signs localization. Similar to the first approach, the tiling technique [8] was implemented with the sliding window of 1024x626 in size and stride 426. After that, the cropped images were padded to 1024x1024 to match the input dimension of the RetinaFace. Then these images were used to train the RetinaFace model. In the second phase, the traffic signs in localized areas of the image were classified into seven different classes by using MobileNetV1-SSD. Each localized traffic sign from the first phase was padded to a 60x60 image. Then, the image was resized to 300x300 in order to satisfy the input image size condition to train the MobileNetV1-SSD model. This approach is expected to get a better result than the approach using the YOLOv5s and the tiling technique. The reason is that the approach keeps the original size of the objects, which allows the classification model to get higher accuracy. Another reason is that the two selected deep neural networks are excessively fast and lightweight compared to other deep learning models in their fields. As a consequence, the method would be able to deploy efficiently on an embedded computer. Combining the two above phases, the tiny traffic signs could be detected with an exceedingly high speed and accuracy. The two models were fine-tuned with the Zalo dataset. Training configurations of the two models are shown in Table 2.

Table 2. Configurations for training RetinaFace [9] and MobileNetV1-SSD [10]

Models	RetinaFace	MobileNetV1-SSD
Image size	1024x1024	300x300
Epochs	250	200
Batch size	32	48
Learning rate	0.001	0.01

2.2. Model Deployment in Embedded Hardware

In order to implement deep learning models for an autonomous vehicle, the models would be deployed into an embedded computer. An embedded computer is suitable for autonomous vehicle applications since it has a lightweight, compact size and low energy consumption. In our approach, the NVIDIA Jetson Nano developer kit was chosen because it is one of the most affordable and powerful embedded computers designed for deep learning applications. The kit has 4GB RAM and could reach 472 GFLOPS of computer performance with a 128-core NVIDIA Maxwell GPU and a 64-bit Quad-core Arm A57 CPU. Although the kit provides a high computational capacity, it consumes only 5 watts of power. The workflow of our method to deploy deep learning models into the embedded computer is shown in Fig. 5.

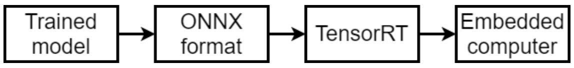

Fig. 5. The workflow of deploying deep learning models into embedded hardware

First, a deep learning model is trained with a custom dataset as described in section 2.1. The trained model is saved as a weight file at the end of this step. After that, the weight file is converted into ONNX format. ONNX is a standard format to describe machine learning models built from manifold different frameworks such as PyTorch or TensorFlow. The software or hardware able to run an ONNX model can load deep learning models developed in a variety of frameworks. In the next stage, the model is transformed from ONNX format to TensorRT format, which assists the model to run efficiently on the Jetson Nano Kit. NVIDIA TensorRT is an SDK for high-performance deep learning inference. It provides APIs to do inference for deep learning models on an embedded computer and optimizes the inference process. TensorRT allows developers to focus on creating innovative artificial intelligent models rather than the performance adjusting inference deployment. In our experiments, the environment installed on the kit is *Ubuntu18.04/cuda10.2/tensorrt7.1/opencv4.5.0*. We applied this process to deploy the two presented traffic sign detection models on the Jetson Nano developer kit.

3. Results and Discussion

3.1. Evaluation of the Traffic Object Detection Model

Table 3. Obtained AP_{50} of the traffic object detection model using Yolov5s (%)

Class	car	bicycle	person	light
AP_{50}	75.9	47.4	58.3	49.9
Class	rider	sign	pole	
AP_{50}	57.8	50.7	28.5	

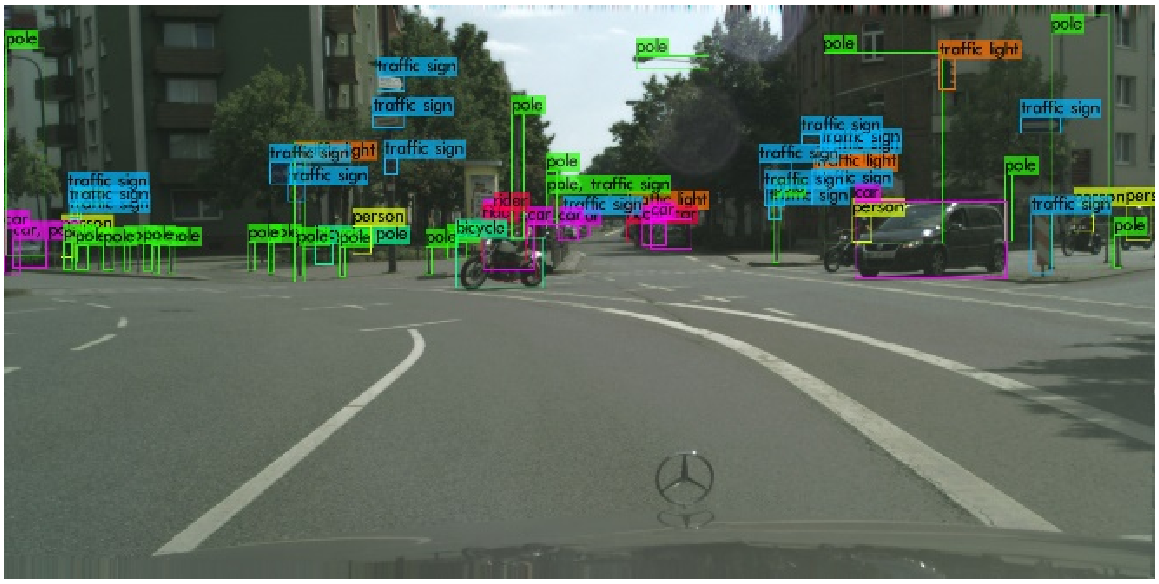

Fig. 6. Illustration of the traffic object detection using the obtained model

The metric Average Precision (AP) with an Intersection over Union (IoU) threshold of 0.5 (AP_{50}) was used to evaluate the precision of the traffic object detection model. The mean average precision for all seven object classes on the test set of the Cityscapes dataset is 52.6%. Table 3 shows the AP_{50} result of individual traffic object classes. It is critically noticed that six of seven classes reach very high AP_{50} (more than 40%), including the car, the bicycle, the person, the rider, the trafic light and the traffic sign The obtained results show a high potential of using the model for practical applications where traffic object detection and localization are needed. Nevertheless, the AP_{50} for the pole class is still low because the poles have very few unique features. Also, the diversity in the shape and the size of the poles is an additional reason. As an illustration, Fig. 6 shows an example of the output frame after running our traffic object detection model. It is clearly seen that all of the interested object classes can be detected efficiently.

3.2. Evaluation of the Traffic Sign Detection Models

Similar to section 3.1, the AP_{50} was used to evaluate the accuracy of the two traffic sign detection approaches using the Zalo dataset. The detailed results of the two methods on each class of the Zalo dataset are shown in Table 4. With the first approach, in which the fine-tuned YOLOv5s model and the tiling technique [8] were used, the mAP_{50} is 53.6%. This result is slightly lower than the result of the second approach, where the fine-tuned RetinaFace [9] model and MobileNetV1-SSD [10] model are combined. A mAP_{50} of 64.2% is achieved. For all classes, the two approaches release very high accuracy, where the highest AP_{50} is 77.2% for the “no parking” class. Detections for interested classes of the dataset all reach the AP_{50} over 50% in the case of combining RetinaFace [9] and MobileNetV1-SSD [10]. Compared to the best result of the Zalo AI Challenge [11], in which the model is trained from scratch, the result of our second approach is higher (64.2% vs. 60.53%).

As our expectation described in section 2.1.2, the second approach improves the accuracy compared to the first approach. This can be explained that, in the second approach, the input image for the classification model (MobileNetV1-SSD) is cropped based on the output bounding boxes of the RetinaFace model. Each bounding box contains only an individual traffic sign. This ensures that the cropped image has only one traffic sign having a maximal dimension as it is in the original frame. Thus, the best quality of the cropped image is achieved. Subsequently, the MobileNetV1-SSD [10] can output better results. Fig. 7 shows an illustration of detecting the traffic signs of the Zalo dataset [11] using the second approach. It can be seen that an excellent result is obtained where the signs are detected correctly with a high rate of confidence.

Table 4. Obtained AP_{50} of the traffic sign detection models (%)

ID	Classes	Tiling technique + YOLOv5s	RetinaFace + MobileNetV1-SSD
1	No entry	46.3	51.9
2	No parking/ waiting	67.2	77.2
3	No turning	56.3	70.9
4	Max speed	64.2	69.0
5	Other prohibition signs	54.3	59.1
6	Warning	47.3	65.6
7	Mandatory	39.6	55.6
	mAP_{50}	**53.6**	**64.2**

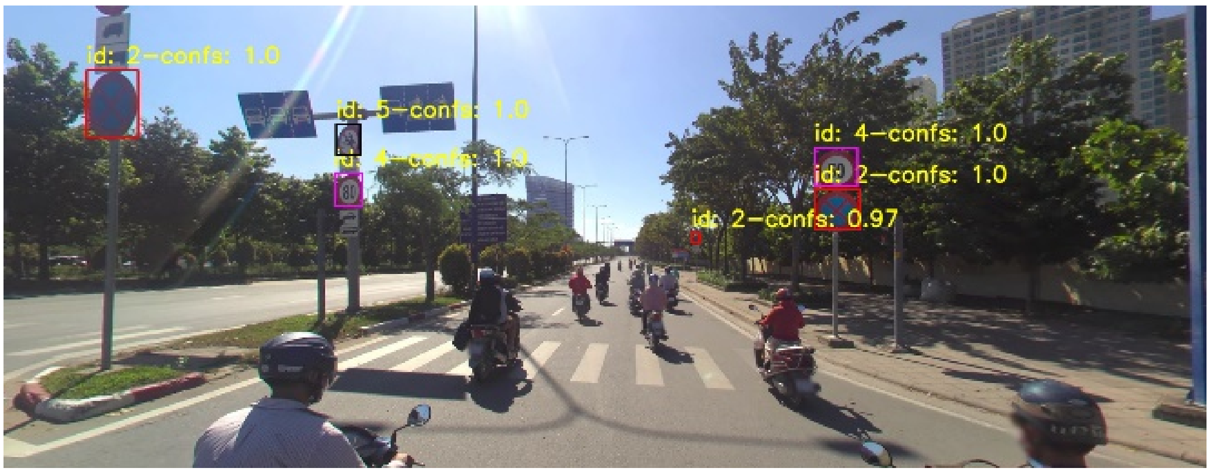

Fig. 7. Illustration of the traffic sign detection using the obtained models

3.3. Computational Efficiency

Computational requirements are very important when deploying the developed models for applications of autonomous vehicles. These applications always require low-power consumption and lightweight hardware with a real-time response. Therefore, the computational efficiency of the models of the two approaches on the very lightweight hardware Jetson Nano Developer Kit of Nvidia [14] was evaluated. Table 5 shows the results where important parameters of the two approaches are presented. It is seen that the second approach has better computational efficiency, where computing time is 9.5 times faster than the first one (0.2s vs. 1.9s). Similarly, the use of the CPU and RAM is less than that of the first approach leading to a reduction in power consumption.

Table 5. Results of deploying models on Jetson Nano Developer Kit [14]

Model	YOLOv5s (traffic object detection)	Tiling technique + YOLOv5s	RetinaFace + MobileNetV1-SSD
Batch size	1	1	1
Mode	FP16	FP16	FP16
Image dimension	640x640	640x640	1024x1024
Computing time	0.08 s	1.92 s	0.20 s
mAP$_{50}$	52.6%	53.6 %	64.2 %
RAM usage	1.4 GB	1.4 GB	1.1 GB

Table 6. Results of deploying models on Jetson Xavier Development Kit [15]

Model	YOLOv5s (traffic object detection)	Tiling technique + YOLOv5s	RetinaFace + MobileNetV1-SSD
Batch size	1	1	1
Mode	FP16	FP16	FP16
Image dimension	640x640	640x640	1024x1024
Computing time	0.01 s	0.24 s	0.02 s
mAP$_{50}$	52.6 %	53.6 %	64.2 %
RAM usage	3 GB	3 GB	1.2 GB

Several analyses are considered. In the second approach, since the RetinaFace model [9] was used to localize traffic signs of the original frame and drop them for further classifications, only a few inferences must be performed. In contrast, in the first approach, the original frame was split into 24 smaller frames and then resized to the dimension fitted to the input requirement of the YOLOv5s [7]. Consequently, 24 inferences must be performed for the classification task. Furthermore, the architecture and the weight of YOLOv5s [7] is heavier than the MobileNetV1-SSD [10]. All of these reasons make the first approach less computational efficiency and accuracy in comparison to the second one. On such lightweight hardware, the second approach can reach a computational capacity of 5 FPS. This shows very high potential for practical applications where stronger hardware can be exploited.

Implementation of the developed models was also deployed on stronger hardware, namely the Jetson Xavier Development Kit (GPU: 512-core Volta GPU with Tensor Cores; CPU 8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3) [15]. The results are shown in Table 6. It is nicely detected that the computational time is much less than the case using Jetson Nano where a value of 100 FPS (0.01s) for the traffic object detection and 50 FPS (0.02s) for the traffic sign detection are achieved. This shows very high potential for practical applications.

4. Conclusion

In this work, models for detecting traffic objects and traffic signs have been developed successfully. Based on the existing datasets and the pre-trained models, the fine-tuning techniques were applied to achieve the trained models with a higher accuracy. The traffic object detection model can be deployed for practical applications since it has high accuracy with a mAP$_{50}$ of more than 40%. The traffic sign detection models also have a high accuracy. The use of the tiling technique incorporated with the YOLOv5s should be avoided since it is not computationally efficient. The combination of the RetinaFace model for localizing the signs in images and the MobileNetV1-SSD model for the classification has twofold advantages in terms of accuracy and computational efficiency (100 FPS on Jetson Xavier). This approach has a very high potential of applications in autonomous intelligent vehicles where a low-power consumption, limited-resource hardware, a real-time response, and a high rate of accuracy are absolutely required. It must be emphasized that the test data from the Zalo dataset is very challenging in which the signs are very far from the camera and thus have very tiny dimension. However, this challenge is solved nicely.

This work still has some limitations. Several shapes and size-specific traffic objects, for example, the pole, cannot be detected accurately. The computational time should be more reduced. In the future, a model which can at the same time localize and classify traffic signs is intended to be developed.

Acknowledgements

Thanh-Tung Ngo was funded by Vingroup Joint Stock Company and supported by the Domestic Master Scholarship Programme of Vingroup Innovation Foundation (VINIF), Vingroup Big Data Institute (VINBIGDATA), code VINIF.2020.ThS.26.

References

[1] Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems, SAE Standard J3016, 2014.

[2] T. Karasawa, K. Watanabe, Q. Ha, A. Tejero-De-Pablos, Y. Ushiku, and T. Harada, Multispectral object detection for autonomous vehicles, in Proc. Themat. Work. ACM Multimed. 2017, Mountain View, CA, USA, 2017, pp. 35–43.

[3] J. Varagula, P. A. N. Kulproma, and T. Itob, Object detection method in traffic by on-board computer vision with time delay neural network, in Proc. KES2017, Marseille, France, 2017, vol. 112, pp. 127–136. https://doi.org/10.1016/j.procs.2017.08.185

[4] H. Zhang, K. Wang, Y. Tian, C. Gou, and F. Y. Wang, MFR-CNN: Incorporating multi-scale features and global information for traffic object detection, IEEE Trans. Veh. Technol., vol. 67, no. 9, pp. 8019–8030, Sept. 2018, 10.1109/TVT.2018.2843394. https://doi.org/10.1109/TVT.2018.2843394

[5] Á. Arcos-García, J. A. Álvarez-García, and L. M. Soria-Morillo, Evaluation of deep neural networks for traffic sign detection systems, Neurocomputing, vol. 316, pp. 332–344, 2018, 10.1016/j.neucom.2018.08.009. https://doi.org/10.1016/j.neucom.2018.08.009

[6] M. Cordts et al., The cityscapes dataset for semantic urban scene understanding, in Proc. CVPR, Las Vegas, NV, USA, 2016, pp. 3213–3223. https://doi.org/10.1109/CVPR.2016.350

[7] G. Jocher, A. Stoken, J. Borovec, C. NanoCode012, L. Changyu, H. Laughing, ultralytics/yolov5: v4.0, Ultralytics, Jan. 5, 2021. [Online]. Available: https://github.com/ultralytics/yolov5.

[8] F. Ö. Ünel, B. O. Özkalayci and C. Çiğla, The power of tiling for small object detection, in Proc. CVPRW, Long Beach, CA, USA, 2019, pp. 582-591. https://doi.org/10.1109/CVPRW.2019.00084

[9] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, S. Zafeiriou, RetinaFace: single-stage dense face localisation in the wild, arXiv: 1905.00641, 2019. [Online]. https://doi.org/10.1109/CVPR42600.2020.00525

[10] Single Shot MultiBox Detector Implementation in Pytorch. [Online]. Available: https://github.com/qfgaohao/pytorch-ssd, Accessed on: Mar. 28, 2021.

[11] Zalo AI Challenge. [Online]. Available: https://challenge.zalo.ai/portal/traffic-sign-detection, Accessed on: Mar. 28, 2021.

[12] N. H. Dung, Application of convolution neural network in design and fabrication of robots for transporting goods in factories, J. Sci. Technol., vol. 147 (2020), pp. 51-58, Nov. 2020. https://doi.org/10.51316/30.8.9

[13] T. Y. Lin et al., Microsoft COCO: Common objects in context, Lect. Notes Comput. Sci., vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014 https://doi.org/10.1007/978-3-319-10602-1_48

[14] Jetson Nano Developer. [online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[15] Jetson AGX Xavier Developer. [online]. Available: Kithttps://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit