DOI 10.47176/sjis.4.4.1

Original Article

SRPH

SJIS, 2022; 4(4): 1-7

Designing and Implementing of Real-Time Intelligent System with the Ability to Identify and Classify Different Topics in AutonomousVehicle

Seyed Parsa Mirfasihi[1], Sina Ghazali[2] and Shahriar Baradaran Shokouhi*[3]

[1,2,3] Faculty of Electrical Engineering, Iran University of Science and Technology, Iran.

***Corresponding Author:**

Baradaran.sh@gmail.com

Received: 23 August, 2022
Accepted: 25 November, 2022
Published: 15 December, 2022

ABSTRACT

Deep learning is a branch of the machine learning and artificial intelligence and a set of algorithms, which tries to model high-level abstract concepts using learning at different levels and layers. Due to high accuracy and efficiency, these algorithms have been used in self-driving cars. Many car accidents are caused by human errors. One of the reasons for paying attention to self-driving cars is to prevent these accidents by eliminating or reducing human interference and consequently reducing the loss of life and property. Also, save and efficient use of time is another factor that has made self-driving cars one of the topics of interest. One of the things that happen in self-driving cars is collecting information, forecasting and making decisions based on available data. In this article, we represent, review, compare and implement networks based on SSD and Mask RCNN models with two types of Inceptions and Resnet architecture in the form of TensorFlow, which is an open-source library. At the end, we have provided a table comparing these two models and representing their characteristics. To do this, we have used the Coco dataset, which is one of the most popular databases for using in mobile robotics as well as automatic driving. The SSD model with the inception v2 architecture has faster speed in identifying objects, which reaches 2fps, but in the other model, this amount is 25% less. Also, the Mask RCNN model is more accurate than the SSD model, which the observations and the results show that with more than 88% probability, it predicts the desired object correctly. All calculations have been tested on the Nvidia P1000 Quadro graphics card.

Keywords: Autonomous vehicle, Machine learning, Deep learning, Artificial intelligence

Introduction

In recent years, deep learning has been actively used in various fields including computer vision and independent driving. The development of sensors and processors, along with deep learning algorithms, has accelerated research into AI-based vehicles. A self-driving vehicle without automatic driver intervention must accurately detect vehicles, pedestrians, traffic signs, traffic lights, and other parameters to ensure safe and correct control decisions. To detect such objects, various sensors such as cameras, light detection sensor (lidar) and radio range detection sensor (radar) are usually used in self-driving vehicles. First, we will briefly describe the history of self-driving cars, and then we will explain the networks used with the above models based on inception and ResNet architectures in the form of TensorFlow. Finally, we will see the results of simulation and implementation on the car.

History

Since the 1930s, science fiction writers dreamed of a future with self-driving cars, and building them has been a challenge for the AI community since the 1960s. By the 2000s, the dream of autonomous vehicles became a reality in the sea and sky, and even on Mars, but self-driving cars existed only as research prototypes in labs.

Driving in a city was considered to be a problem too complex for automation due to factors like pedestrians, heavy traffic, and the many unexpected events that can happen outside of the car's control. Although the technological components required to make such autonomous driving possible were available in 2000—and indeed some autonomous car prototypes existed predicted that mainstream companies would be developing and deploying autonomous cars by 2015. During the first Defense Advanced Research Projects Agency (DARPA) "grand challenge" on autonomous driving in 2004, research teams failed to complete the challenge in a limited desert setting.

But in eight short years, from 2004-2012, speedy and surprising progress occurred in both academia and industry. Advances in sensing technology and machine learning for perception tasks has sped progress and, as a result, Google's autonomous vehicles and Tesla's semi-autonomous cars are driving on city streets today. Google's self-driving cars, which have logged more than 1,500,000 miles (300,000 miles without an accident), are completely autonomous no human input needed. Tesla has widely released self-driving capability to existing cars with a software update. Their cars are semi-autonomous, with human drivers expected to stay engaged and take over if they detect a potential problem. It is not yet clear whether this semi-autonomous approach is sustainable, since as people become more confident in the cars' capabilities, they are likely to pay less attention to the road, and become less reliable when they are most needed. The first traffic fatality involving an autonomous car, which occurred in June of 2016, brought this question into sharper focus [1].

Suggested Methods

In this section, we describe the two main proposed methods and the architectures of each one.

Mask-RCNN model with Resnet50 architecture in the form of TensorFlow

Mask R-CNN, or Mask RCNN, is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation and instance segmentation. Mask R-CNN was developed on top of Faster R-CNN, a Region-Based Convolutional Neural Network.

The function of Mask R-CNN

Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding-box offset, Mask R-CNN is the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object [2].

Mask R-CNN is an extension of Faster R-CNN and works by adding a branch for predicting an object mask (Region of Interest) in parallel with the existing branch for bounding box recognition.

- Simplicity: Mask R-CNN is simple to train.
- Performance: Mask R-CNN outperforms all existing, single-model entries on every task.
- Efficiency: The method is very efficient and adds only a small overhead to Faster R-CNN.
- Flexibility: Mask R-CNN is easy to generalize to other tasks. For example, it is possible to use Mask R-CNN for human pose estimation in the same framework.

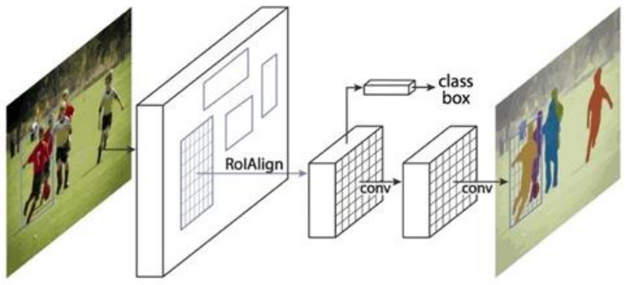

Figure 1. Mask R-CNN, Mask R-CNN framework for sample segmentation [2]

ResNet

ResNet stands for Residual Network. It is an innovative neural network that was first introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their 2015 computer vision research paper titled 'Deep Residual Learning for Image Recognition'. This model was immensely successful, as can be ascertained from the fact that its ensemble won the top position at the ILSVRC 2015 classification competition with an error of only 3.57%. Additionally, it also came first in the ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation in the ILSVRC & COCO competitions of 2015 [3].

ResNet has many variants that run on the same concept but have different numbers of layers. Resnet50 is used to denote the variant that can work with 50 neural network layers.

[DOI: 10.47176/sjis.4.4.1]

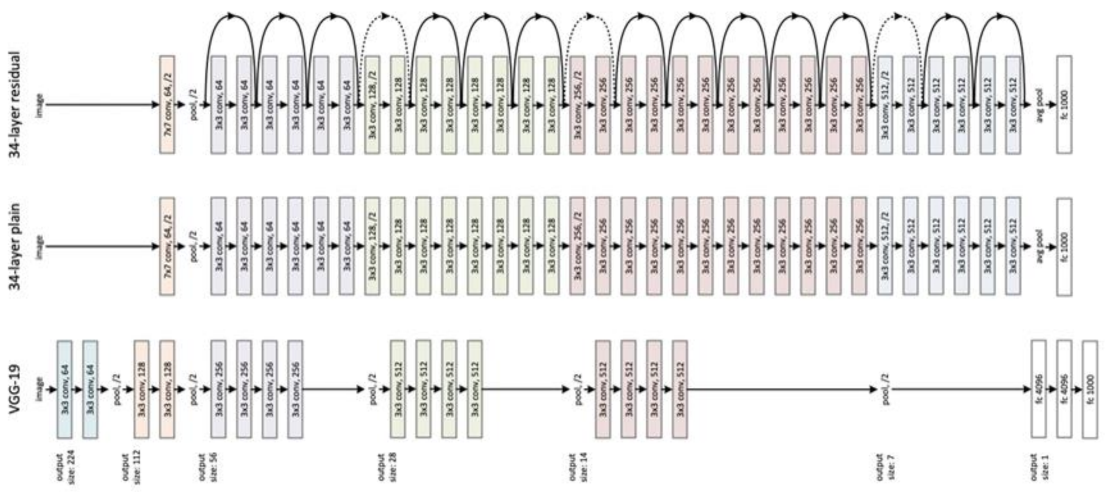

Figure 2. General structure of a ResNet [3]

Resnet50 Architecture

While the Resnet50 architecture based on Figure 2, there is one major difference. In this case, the building block was modified into a bottleneck design due to concerns over the time taken to train the layers. This used a stack of 3 layers instead of the earlier 2.
Therefore, each of the 2-layer blocks in Resnet34 was replaced with a 3-layer bottleneck block, forming the Resnet 50 architecture. This has a much higher accuracy than the 34-layer ResNet model. The 50-layer ResNet achieves a performance of 3.8 bn FLOPS.

SSD model with Inception version 2 architecture in the form of TensorFlow

SSD model

SSD is a single-shot detector. It has no delegated region proposal network and predicts the boundary boxes and the classes directly from feature maps in one single pass. To improve accuracy, SSD introduces: small convolutional filters to predict object classes and offsets to default boundary boxes.
SSD has two components: a backbone model and SSD head. Backbone model usually is a pre-trained image classification network as a feature extractor. This is typically a network like ResNet trained on ImageNet from which the final fully connected classification layer has been removed. We are thus left with a deep neural network that is able to extract semantic meaning from the input image while preserving the spatial structure of the image albeit at a lower resolution. The SSD head is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers activations [4].
In the figure below, the first few layers (white boxes) are the backbone, the last few layers (blue boxes) represent the SSD head.

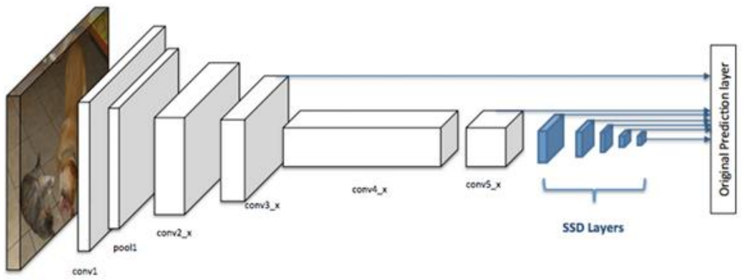

Figure 3. Architecture of a convolutional neural network with an SSD detector

Inception version 2 Architecture

The architecture was introduced in 2015 and was an important turning point in the development of CNN classifiers. Prior to the development of this architecture, most popular CNNs only deepend the covolutional layers with the hope that they would perform better, but on the other hand, the Inception architecture was complex, meaning that it was highly engineered and made of tricks. Many used it to increase its performance, both in terms of speed and accuracy, which is why several versions of it have been released over time [5].

[DOI: 10.47176/sjis.4.4.1]

In this article, we will use the second version, which is not much different from the third version. In this architecture, to improve the speed of 5×5 convulsions, they are divided into two 3×3 convolution operations. Convolution 5×5 is approximately 2.7 times more expensive than 3×3, but two 3×3 convolutions are faster (Figure 4-a). Also, n × n filters are divided into a combination of 1 × n and n × 1 filters, which reduces costs by 33% (Figure 4-b). Finally, instead of going deeper, the filters have been expanded to help fix the bottleneck. If each convolution deepened instead of expanding, it would cause information loss as well as excessive dimensionality (Figure 4-c).

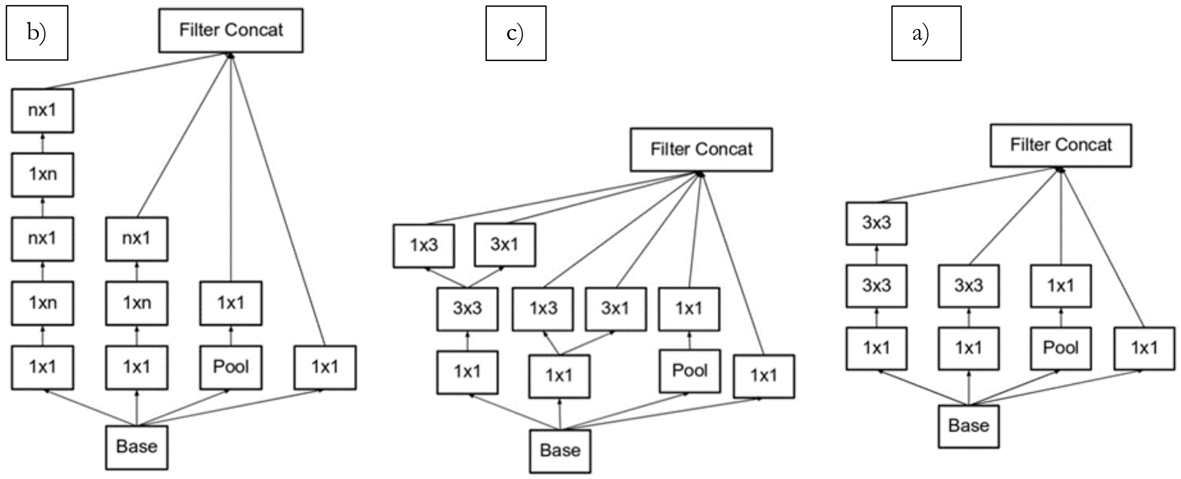

Figure 4. a) Resizing convolutions **b)** Converting n×n filters to a combination of 1×n and n×n filters **c)** Initial modules in which each 5×5 convolution is replaced by two 3 × 3 convolution [5]

TensorFlow

Machine learning is a complex discipline. But implementing machine learning models is far less daunting and difficult than it used to be, thanks to machine learning frameworks such as Google's TensorFlow that ease the process of acquiring data, training models, serving predictions, and refining future results. Created by the Google Brain team, TensorFlow is an open-source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training [6].

Dataset

A data set called COCO has been used to train the network. MS COCO Dataset is a large-scale identification, segmentation, and subtitle dataset released by Microsoft. Machine learning and computer vision engineers typically use COCO datasets for a variety of computer vision projects.

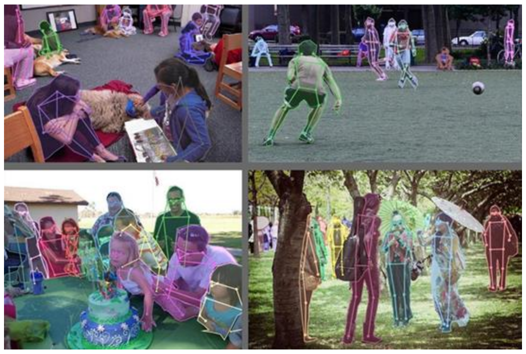

Figure 5. Identification of the key point for estimating the position in the COCO dataset [7]

[Downloaded from sjis.srpub.org on 2026-01-02]
[DOI: 10.47176/sjis.4.4.1]

COCO stands for Common Objects in Context, as the image dataset was created with the goal of advancing image recognition. The COCO dataset contains challenging, high-quality visual datasets for computer vision, mostly state-of-the-art neural networks.
For example, COCO is often used to benchmark algorithms to compare the performance of real-time object detection. The format of the COCO dataset is automatically interpreted by advanced neural network libraries.

Results and Discussion

This section includes the outputs of the models as well as a comparison between their speeds in the input image processing. Finally, we have stated the issues that have been misidentified and their causes.

Outputs after installation on the car

To install the desired system on the car, we must first set up the system connection to the camera wireless network that is based on the web. To do this, we first connect the mobile phone camera to the network. In this regard, there are various software, but in this project, we have used IP Camera software because it provides various capabilities to the user that make the work easier. In the next step, we install the mobile phone in a suitable place in the front part of the car and at the bottom of the front mirror, and we make sure that it is stable. Then we start the simulation and moving at a constant speed of 10 km/h.

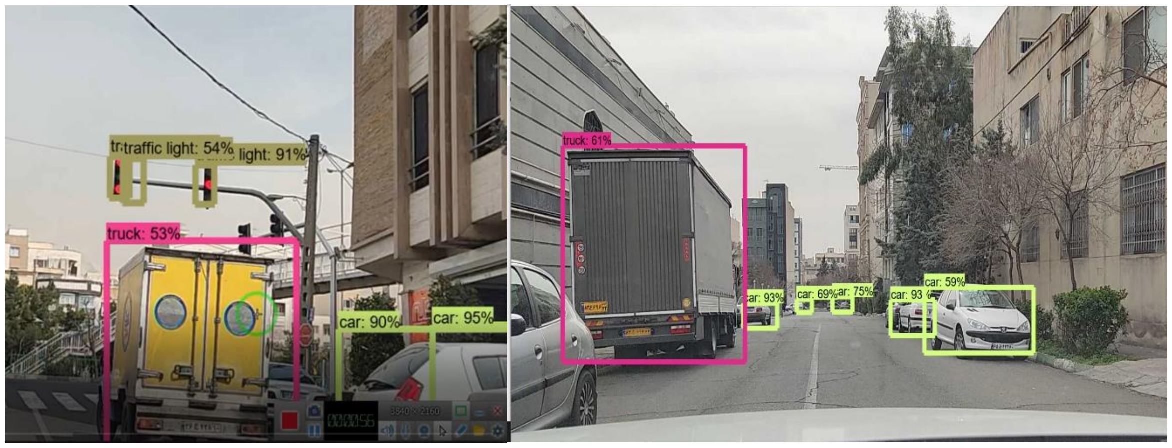

Figure 6. SSD model outputs

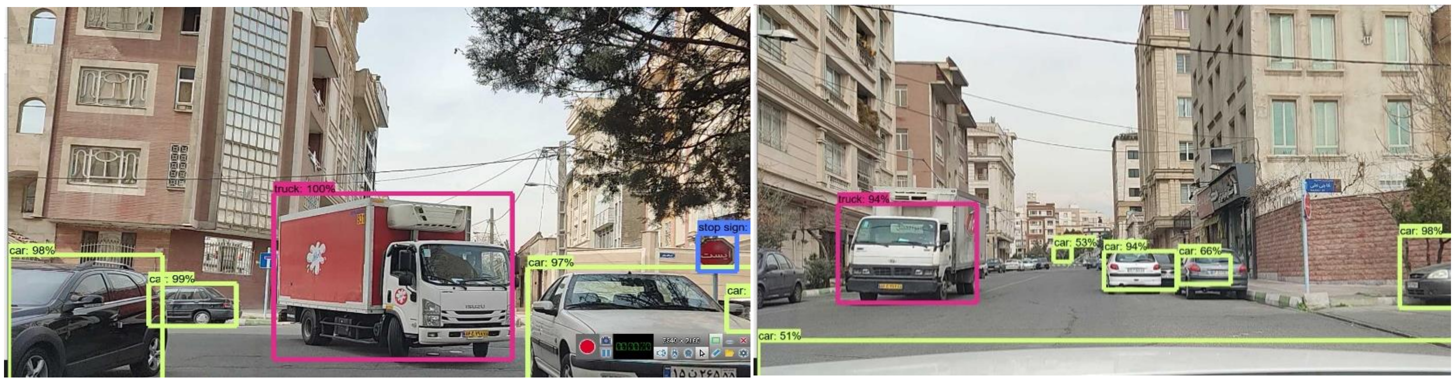

Figure 7. Mask model outputs

Differences between the two models in terms of speed and accuracy

To evaluate a particular algorithm in sample segmentation, you usually see the terms mentioned in the diagram, i.e. AP (average accuracy), which is obtained by comparing the amount of average accuracy with other models presented in different years.

[DOI: 10.47176/sjis.4.4.1]

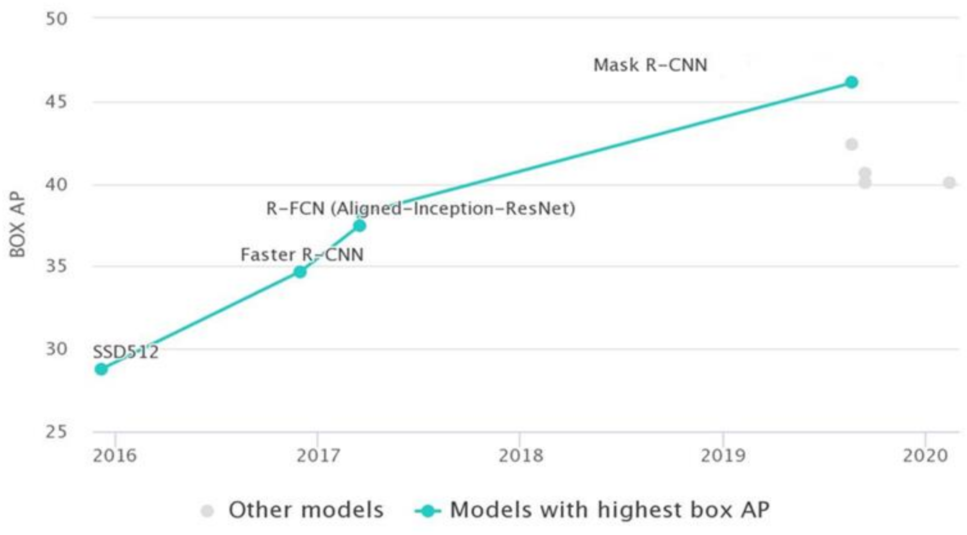

Figure 8. Comparison of average accuracy

From the figure above, we can see that as time goes by and the models become more engineered (not necessarily deeper), the accuracy of the output associated with these models also increases.

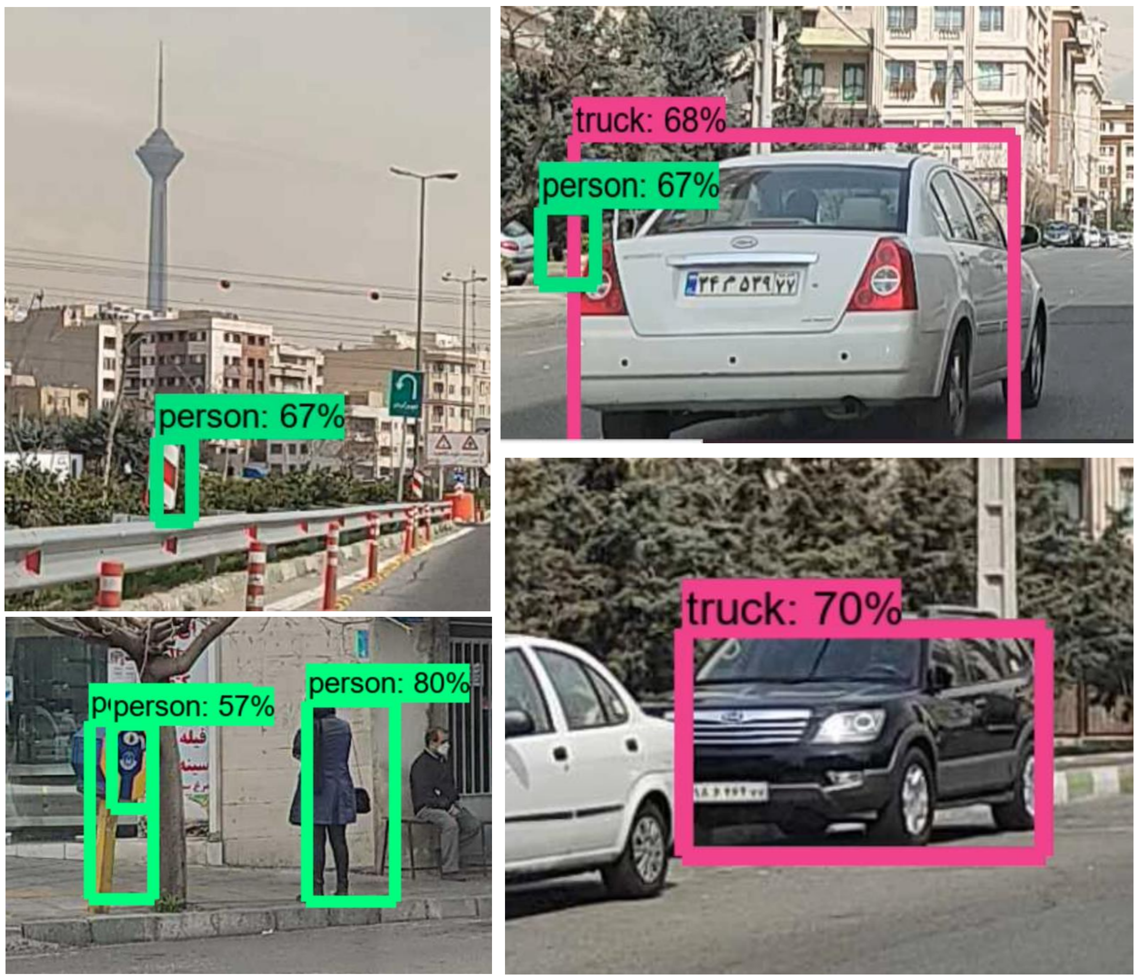

Figure 8. Examples of incorrectly identified topics

[DOI: 10.47176/sjis.4.4.1]

Table 1
Speed of processing systems inputs

Method	Duration of one frame	FPS
SSD	0.5 sec	2
Mask	4 sec	0.25

From the table above, it can be seen that on home graphics cards, the SSD model has a relatively faster processing speed of the desired video, which can be explained by the type of architecture.

False recognition

In general, issues that are misidentified have four different causes:

- Location: Here the subject is properly categorized and has the correct label, but the overlap of the predicted box with the correct box is less than 0.5. In fact, the network could not predict the exact location of the issue.
- Similarities: Here the network has correctly identified the location of the subject, but has not been able to categorize the subject correctly. In fact, the subject is in the group of similar classes. Similar classes include classes that are similar in appearance and may not be able to be separated by the network. There are two similar groups for this dataset:

{Truck, Car}
{Bicycle, Motorcycle, Pedestrian, Sign}

- Other: Here the network correctly identifies the location of the subject, but can not categorize the subject correctly and recognize its class, and the subject class is not in the same class group.
- Background: Here the network categorizes the subject as the background, in fact the network could not recognize the subject class.

Conclusion

In this paper, COCO datasets with two SSD and MASK networks were taught. The AP was 28% for the SSD network and 47% for the Mask network, and the speed of each photo on the SSD network was 2 frames per second, and for the Mask network, 2 frames per 4 seconds. As a result, the SSD network is much faster than the Mask network for detecting vehicles and on this data set, but it is also much less accurate.
In addition, as shown in the paper, the ResNet architecture on small classes performed well and relatively compared to the Inception architecture.

References

1. Chair BJG, Altman R, Horvitz E, Mackworth A, Mitchell T, Mulligan D, Shoham Y. Artificial intelligence and life in 2030. *Stanford University, CA,* 2016.
2. He K, Gkioxari G, Doll´ar P, Girshick R. Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision,* 2017; 2961-2969.
3. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016; 770-778.
4. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC. SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision,* 2016; 21-37.
5. Szegedy C, Vanhoucke V, Ioffe S, Shlens J. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016; 2818-2826.
6. Abadi M, Barham P, Chen J, Chen Z. {TensorFlow}: A System for {Large-Scale} Machine Learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16),* 2016; 265-283.
7. Lin T-Y, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, Perona P, Ramanan D, Zitnick CL, Doll´ar P. Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision,* 2014; 740-755.

Citation: Mirfasihi SP, Ghazali S, Baradaran Shokouhi S. Designing and Implementing of Real-Time Intelligent System with the Ability to Identify and Classify Different Topics in Autonomous Vehicle. SJIS, 2022; 4(4): 1-7.

https://doi.org/10.47176/sjis.4.4.1