Object Detection of Self-driving Vehicles

Bilal Zeidan
W Booth School of Engineering Practice and Technology
McMaster University
Hamilton, Canada
zeidanb@mcmaster.ca

***Abstract*—Autonomous vehicles offer a promising solution to safer driving through monitoring and utilizing real-time sensing data. Information collected from different sensors, including LiDAR's, ultrasonic sensors, cameras, radars, and GPS, can be processed, and analyzed to operate the vehicle. Computer vision played an essential role in advancing this technology, especially object detection, which aims to locate and classify objects in a scene. This paper will discuss autonomous vehicle technology and investigate the different object detection, neural network models, architecture, and performance.**

Keywords—Computer Vision; Object Detection; Autonomous Vehicles; self-driving; Neural Networks

I. INTRODUCTION

Autonomous vehicles (AVs) have become a reality, thanks to the increase in computation power and decrease in the cost of sensing technology. According to a 2017 McKinsey report, autonomous driving in urban areas could reach $1.6 trillion a year in 2030, which, if put into perspective, is around two times the combined revenues of Ford, General Motors, Toyota and Volkswagen during 2017 [1]. As of 2021, there are no level 5 cars available to the public; by level 5, we mean entirely autonomous, i.e. no human interaction is required. Many companies are developing a level 5, and it is undergoing many tests to overcome the challenges. Some companies claim that their car is level 5 ready, but it will still be challenging to introduce them in the real world. Usually, in real traffic situations, we are interested in detecting other vehicles, pedestrians, animals, road signs, traffic lights and many other objects that a vehicle might encounter. Also, different weather conditions and road conditions need to be accounted for, making the problem more challenging. Many large companies, including Tesla, BMW, Volkswagen, General Motors and Ford, have invested billions of dollars in this technology and are helping advance it and overcome the challenges.

II. CHALLENGES

A. *Large & Complex Software System*

Due to the complexity of tasks carried in AVs, the software runs with around 1 billion lines of code. This can be compared to the Boeing 787 jetliner's 14 million lines of code to understand how large and complex the AV software can be. This large codebase can introduce many bugs that can cause terrible accidents, and if a company with millions of cars on the street decided to apply a patch or update, one could imagine how hard it can get be to manage [2].

B. *Model Generalization*

A typical problem that all neural network models face is a model generalization. The model must detect and interpret objects under different conditions and changes in the background. For instance, different road and weather conditions should not affect the prediction accuracy [2]. That also includes the countless possible situations the vehicle could encounter like jaywalking, people lying on the ground or small objects on the road [3].

C. *Speed & Accuracy*

Solving an optimization problem in under a second is an essential requirement for AVs as they rely on real-time detection to take fast reactions. In such problems, it becomes infeasible to use image pre-processing, which helps in boosting detection performance to save on time. This restricts the input image while demanding the same performance accuracy [4].

D. *Network Depth*

The deeper the neural network becomes; the more parameters must be stored. Car computing systems are generally memory hungry, making it impossible to use detectors with lots of parameters [4].

III. CLASSIFICATION VS. REGRESSION

The problem at hand is a combination of classification and regression. We are trying to detect if the scene contains an object or not; in our data, it is a car or nothing. Also, we need to know the car's location in the scene, which is described by finding the four coordinates of the object's bounding box. Below are some differences encountered when solving a classification problem vs. a regression one.

A. *Activation Function Output*

In the context of neural networks, a classification problem requires a non-linear activation function output such as softmax at the end of the network to get the class probability as output. On the other hand, regression uses a linear model on the output neurons [5].

B. *Loss Function*

The cross-entropy function is the default loss function in classification, while we usually use the least-squares cost function in regression [5].

C. *Output*

In a regression problem, the output is continuous. However, in a classification problem, the goal is to identify discrete output variables, which can be labels or categories [6].

IV. The Dataset

The data set used in the project is taken initially from the TJ machine learning club website. The training and testing data consists of JPEG images (380x676 pixels) taken using a side-view-mounted camera on a moving car. The label for the training data is either a car or none. Also, the bounding box data is given in a .csv file which includes for each image the bounding box coordinates xmin, ymin, xmax, ymax with the image ID. Some images have multiple bounding boxes, and if an image does not contain a car, then no bounding box exists. The total size of the training data is 1001 images and 175 for testing. About 35% of the images contain cars, and 65% do not contain any cars.

A. *Pre-processing*

No null values were detected in the bounding box .csv file during pre-processing. Since the images are RGB and pixel values range from 0 to 255, we need to normalize the data to work better with neural networks by dividing it by 255 to obtain values between 0 and 1. After that, the data can be split into training/validation with a ratio of 80/20, resulting in 801 training samples and 200 validation samples.

B. *Image Size*

To reduce computational time, images can be resized to (236x420 pixels) using OpenCV without compromising much accuracy.

C. *Data Visualization*

To get a sense of the data we are dealing with, we used OpenCV to draw a sample image with the bounding box, as shown in fig. 1.

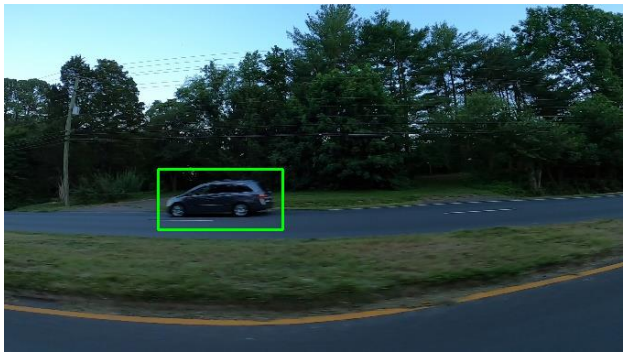

Fig. 1. Sample image from the training data

V. Model Selection

As mentioned before, we must deal with two different problems in object detection: image classification and object localization. We want to classify the images as to whether they contain a car or not and at the same time to determine the bounding box around one or more objects in an image. Combining both tasks will give us an object detection problem, and in our dataset, we only have one class, "car." However, this can be taken to the next level to include tracking the objects across different frames, which will allow us to know if it is the same object or a different object has appeared on the next frame. This will also allow us to differentiate between cars and treat them as objects, i.e. "car1" and "car2". While object tracking is out of the scope of this paper, but it remains an excellent material for future research to expand on this paper.

We will discuss some neural network models that made breakthroughs in object detection. Almost all of those models share an initial stage that includes a CNN to extract the feature map. Many CNN models have been developed, including the VGG, ResNet, Inception, and MobileNet, which can be used with object detection models. These CNN models can solve many similar image classification problems by utilizing a technique called transfer learning. This can be very useful when the data size is small, as we can start training with initialized pre-trained weights.

A. *R-CNN*

When the R-CNN paper was released in 2013, it made a breakthrough in object detection. Although the model now is considered very slow, it is essential to discuss to understand the models that came after, including the Faster R-CNN model.

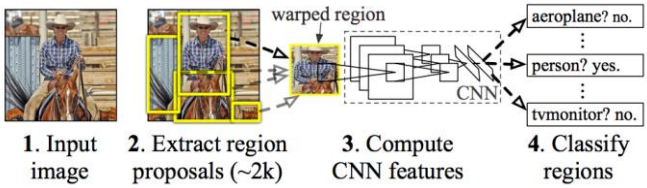

Fig. 2. R-CNN Architecture [7]

It scans the input image for possible objects using a selective search algorithm. Then feeds the regions of proposal into a CNN to compute feature maps. Finally, it uses an SVM to classify the regions if they contain an object with labelling and a linear regressor to adjust the bounding box of the obtained object. Generally, R-CNN suffers from being slow due to the large number of parameters computed based on the number of proposals found by the selective search (~2,000 regional proposals) [8].

B. *Fast R-CNN*

It came out a year after the R-CNN architecture and introduced two significant changes that improved the model's speed. First, it applied feature extraction to the image before proposing regions instead of applying a CNN on the ~2000 overlapping regions. Finally, it replaced the SVM used in R-CNN with a softmax layer. It also uses an RoI pooling layer after obtaining the feature maps [8].

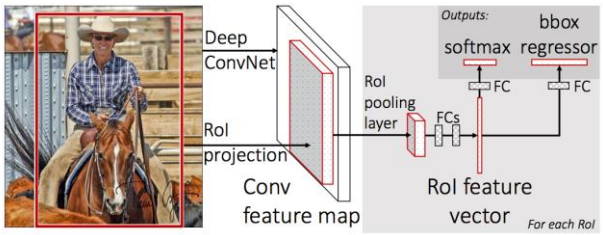

Fig. 3. Fast R-CNN Architecture [9]

a) RoI Pooling: Max pooling layers are applied on different-sized inputs to obtain a fixed-size feature map. RoI layer takes a fixed-size feature map and a list of regions of interest described by the coordinates of the proposed bounding boxes. One of the major benefits of RoI pooling is that it reuses the feature map detected from the previous convolutional network stage [9].

Although faster than R-CNN, it still suffers from the slow selective search algorithm.

C. *Faster R-CNN*

Instead of using selective search to determine the regions of interest, Faster R-CNN introduced the region of proposal network (RPN), making it faster and more accurate than the Fast R-CNN. The remaining stages in Faster R-CNN are very similar to the Fast R-CNN, where we use an RoI pooling layer to obtain fixed-size feature maps. The final layers are a softmax for classification and a bounding box regressor for determining the four coordinates of the bounding box [8].

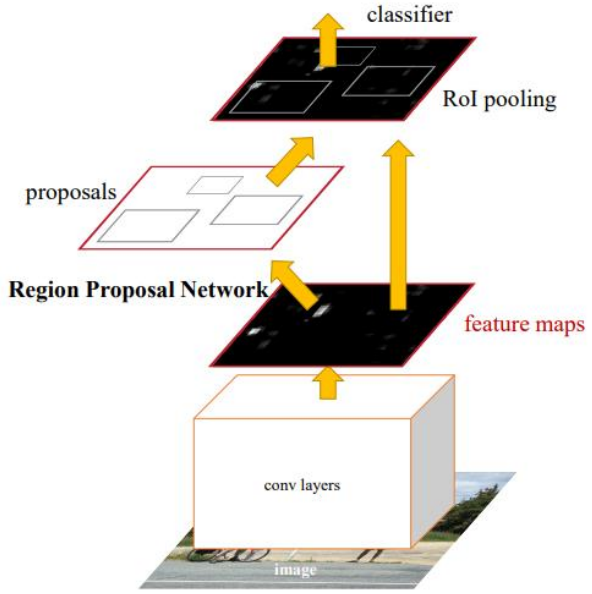

Fig. 4. Faster R-CNN Architecture [9]

a) Region Proposal Network (RPN): It is a fast neural net that uses reference boxes (anchors) to find suitable regions, i.e. bounding boxes which may contain objects. It takes the feature maps obtained from the CNN and suggests different size boxes centred around each location on the feature map. Each proposed region has two outputs associated with them. First is the objectness score, i.e. if we have an object or not, as we do not care about the class label at this stage. The other output is the coordinates of the bounding box that helps in obtaining a better fit for the predicted objects. A threshold is applied to eliminate regions with a lower objectness score [9].

It consists of 2-stages, a regional proposal network stage and a classification stage. Speed is dependent on the number of proposed regions

D. *R-FCN*

This model is considered more accurate and faster than Faster R-CNN. It is fully convolutional, which can benefit from sharing computations in the network. It uses a convolutional layer to produce a position-sensitive score map from each class's input feature maps, including the background. Then uses the same RPN technique used in Faster R-CNN to generate the RoI's. Next, it performs RoI pooling using the obtained bank of score maps. Finally, the result is classified using softmax [9].

E. *SSD*

Considered faster as it uses a single shot detector, it performs object detection and bounding box prediction simultaneously. Thus avoiding duplicate computations. It generally can predict larger objects accurately and fast. However, it performs poorly in detecting small objects [10].

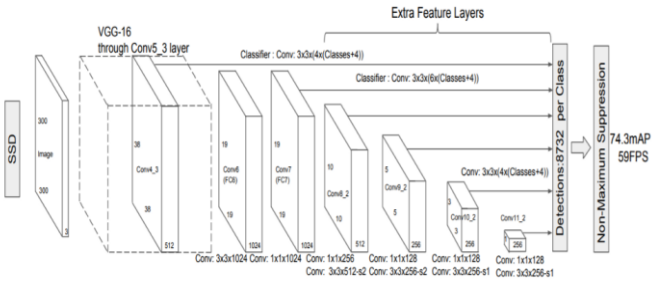

Fig. 5. SSD Architecture [11]

VI. Speed vs. Accuracy

Fig. 6 shows a comparison between the Faster RCNN, R-FCN and SSD object detection models with different CNN models on different datasets. It is tough to say that one model outperforms the other in all situations. Picture size, speed and accuracy are vital elements in deciding which model to choose. It is worth noting that the variation in results can also be due to one model being better in detecting certain objects compared to others.

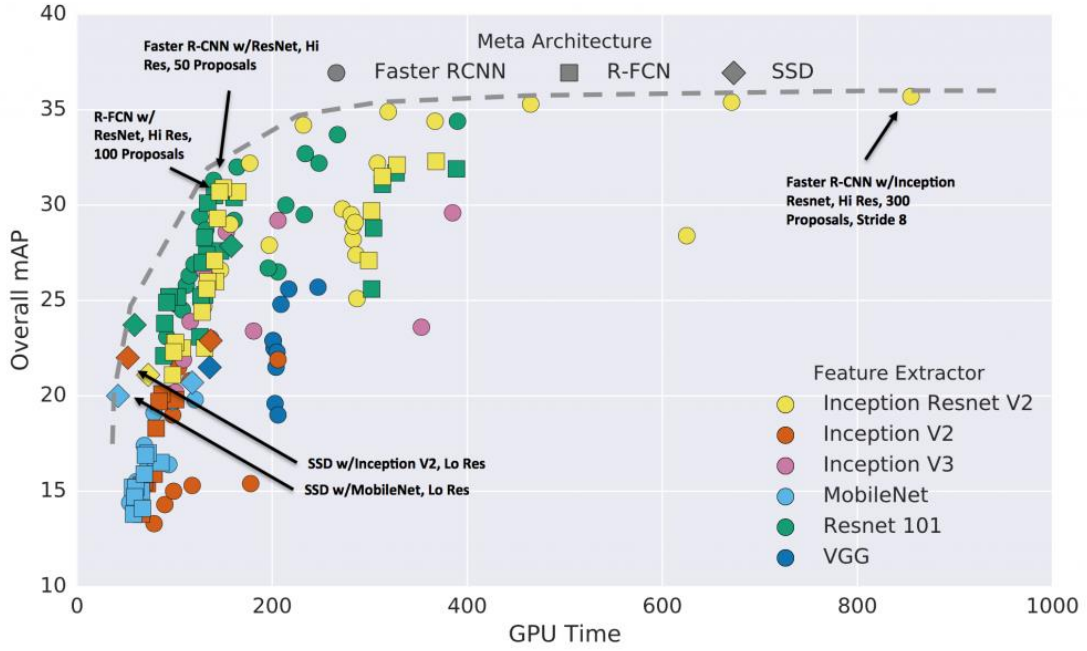

Fig. 6. mAP vs GPU Time for different models [12]

VII. Conclusion

Autonomous vehicles process complex data that come from their sensors. We looked at data that can be received from a camera sensor on an AV. However, object detection problem in AVs is more complex and involves feedback data from different sensors, including LiDAR's, GPS and radar. This paper discussed some of the major models used in object detection. However, some other models were not investigated and are worth mentioning, including the famous YOLO model and RetinaNet, which Facebook AI Research released to overcome dense detection problems.

References

[1] J. R. Nerad, "Autonomously Driven Electric Cars Will Change The World, McKinsey Says," forbes, 08 July 2019. [Online]. Available: https://www.forbes.com/sites/jacknerad2/2019/07/08/autonomously-driven-electric-cars-will-change-the-world-mckinsey-says/?sh=15737a614119. [Accessed 01 December 2021].

[2] T. Zhang, "Toward Automated Vehicle Teleoperation: Vision, Opportunities, and Challenges," *IEEE*, vol. 7, no. 12, pp. 11347 - 11354, 2020 .

[3] R. Sell, A. Rassõlkin, R. Wang and T. Ott, "Integration of autonomous vehicles and Industry 4.0," *PROCEEDINGS OF THE ESTONIAN ACADEMY OF SCIENCES*, vol. 68, no. 4, pp. 389-394, 2019.

[4] G. Lewis, "Object Detection for Autonomous Vehicles," 2016. [Online]. Available: https://web.stanford.edu/class/cs231a/prev_projects_2016/object-detection-autonomous.pdf. [Accessed 01 December 2021].

[5] G. Dreyfus, Neural Networks Methodology and Applications, Berlin: SpringerNature, 2005.

[6] S. Gupta, "Regression vs. Classification in Machine Learning: What's the Difference?," springboard, 06 October 2021. [Online]. Available: https://www.springboard.com/blog/ai-machine-learning/regression-vs-classification/. [Accessed 01 December 2021].

[7] M. R. B. Girshick , J. Donahue and T. , "Rich feature hierarchies for accurate object detection and semantic," *CoRR*, vol. abs/1311.2524, 2018.

[8] R. Shanmugamani, Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras, Packt Publishing, 2018.

[9] J. Xu, "Deep Learning for Object Detection: A Comprehensive Review," towardsdatascience, 11 September 2017. [Online]. Available: https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9. [Accessed 01 December 2021].

[10] M. G. Hyams and D. , "The Battle of Speed vs. Accuracy: Single-Shot vs Two-Shot Detection Meta-Architecture," clear.ml, 08 March 2020. [Online]. Available: https://clear.ml/blog/the-battle-of-speed-accuracy-single-shot-vs-two-shot-detection/. [Accessed 01 December 2021].

[11] R. Khandelwal, "SSD : Single Shot Detector for object detection using MultiBox," towardsdatascience, 30 November 2019. [Online]. Available: https://towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca. [Accessed 01 December 2021].

[12] M. J. Huang , V. Rathod and C. S, "Speed/accuracy trade-offs for modern convolutional object detectors," arXiv, 2017.

[13] Y. Pang and J. Cao, Deep Learning in Object Detection and Recognition, Singapore: Springer, 2019.

[14] P. S. Drew , A. H. and D. X, "Perception, Planning, Control, and Coordination for Autonomous Vehicles," *Machines*, vol. 5, no. 6, 17 February 2017.

[15] L. P. Sermanet , D. Eigen and X. Z, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," arXiv, 2014.