SEMANTIC SEGMENTATION SELF DRIVING CARS

[1]M. NARENDRA, [2]M. VENKATSHIVA, [3]CH.BHANU PRAKASH, [4]N. TEETU REDDY, [5]M. VIGNESHKUMAR

[1,2,3,4]Students, [5]Professor
Dept. Computer Science and Engineering,
Bharath Institute of Higher Education and Research,
Chennai, India

Abstract- **Semantic segmentation was once thought to present a formidable prediction challenge in computer vision. Deep learning has made strides in recent years, making automated driving solutions a viable prospect. The majority of currently used semantic segmentation algorithms weren't created with automated vehicle operation in mind; instead, they were created for general-purpose image processing. We describe a dependable technique for semantic segmentation in autonomous cars in this article. A precise and current semantic segmentation system must be created right away for autonomous vehicles to operate in a safe and efficient manner.**

The foundation of the project is the idea of autonomous driving or self-driving cars. Without the ability to recognize and react to risks like pedestrians, other vehicles, traffic lanes, etc., autonomous vehicles cannot operate.

INTRODUCTION

Semantic segmentation was regarded as a challenging computer vision issue a few years ago. Recent developments in deep learning have made solutions for its application in automated driving more practical. The vast majority of semantic segmentation algorithms in use today were developed for generic images and do not take into account the pre-existing structure or end goals because of self-driving cars. According to our findings in this article, because of self-driving cars. According to our findings in this article, we describe an effective technique for semantic segmentation for self-driving cars. The development of an accurate and real-time semantic segmentation system for self-driving automobiles is urgently required.

In the recent years, a wide range of tech companies and academic institutions have demonstrated a significant interest in autonomous vehicles (or autonomous cars), investing enormous amounts of technological and monetary resources in their development. .Many of them are engaged in extensive study at the moment. They include General Motors' Cruise, Tesla Motors' Model S, BMW's partnership with BAIDU, Ford's Argo AI, etc., as well as NVIDIA's Pilot Net, Delphi's MIT-based start-up autonomy, and a slew of others. As a result, widespread adoption of driverless vehicles is on the horizon if certain conditions are met.

To do semantic segmentation, we must identify each pixel in an image with a category that best describes its content. Although this could be viewed as a pixel classification problem in a picture, the process required to accurately anticipate the label of the complete image is somewhat more involved. The massive influence of deep learning's achievement has improved the accuracy of semantic segmentation approaches significantly. Many areas of technology and study that now lack adequate computer vision skills have become interested in this area as a result of these promising breakthroughs. One such use case is autonomous driving, which requires the vehicles to be aware of and react appropriately to their immediate surroundings, including other vehicles, pedestrians, and roadway features such as signs and signals.

Applications

The cost of processing these algorithms is still fairly low in applications like autonomous driving, where low latency operations are crucial. One such use is autonomous driving, which frequently necessitates making quick decisions. As a result, it's critical to improve the segmentation model design in order to create effective structures that work well in real time.

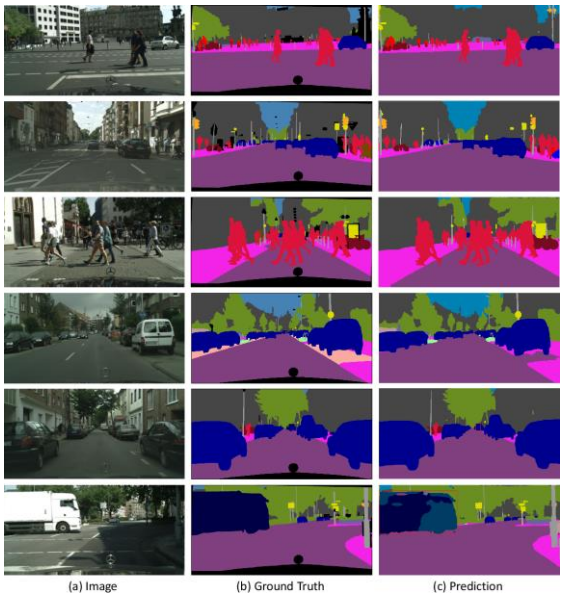

Figure 1: Real time image segmentation

LITERATURE SURVEY

[1] We should expect a dramatic shift in how we get around cities thanks to self-driving cars since they will make greener, safer, more convenient, and less congested modes of transportation available to more people. Using AI for autonomous vehicles presents a number of obstacles, including the inability to accurately recognise pedestrians, traffic signals, signs, and ambiguous lane lines. The ubiquitous availability of Graphics Processing Units (GPUs) and cloud platforms, as well as recent advancements in Deep Learning and Computer Vision, make it possible to tackle and solve these issues. In this study, we offer a deep neural network model for traffic signal identification and recognition that was created and improved via transfer learning. This method uses a quicker Inception V2 region-based convolutional network model from TensorFlow for transfer learning (R-CNN). The dataset used to train the model includes a variety of images of traffic lights that were consistent with Indian traffic signals. The model is successful in its endeavor because it associates the correct class type with the traffic signal.

[2] To overcome the limitations of using a single sensor during harsh weather, a radar and camera information fusion sensing technique is applied. The radar is the primary sensor in our fusion system, while the camera provides extra data. At the same time, the Mahala Nobis distance is employed to compare the observed values to the desired sequence. the use of a joint probability function to integrate data. Using sensor data from a working automobile, we were able to investigate how people perceive their surroundings in the real world. Positive results from experiments reveal that radar and camera fusion algorithms beat single sensor environmental perception in adverse conditions, which is good news for the safety of autonomous cars. The fusion method strengthens the environment perception system, delivering more reliable data for the autonomous vehicle's decision-making and control systems.

[3] Predictions provided by area proposal algorithms, which try to place items in an image, are significantly relied upon by most state-of-the-art two-stage object recognition networks. However, it is widely acknowledged that region proposal techniques are the bottleneck in the majority of two-stage object recognition networks, making them unfit for real-time applications like autonomous driving vehicles due to the increased processing time required for each image. For the purpose of object identification in autonomous cars, we provide RRPN, a method for proposing regions in real time. RRPN converts radar detections into image coordinates, allowing for the creation of anchor boxes at each detection site. When the distance between the vehicle and the object is known, the anchor boxes are enlarged and altered to make more precise recommendations for the identified things. We test our approach using the recently published NuScenes dataset and the Fast R-CNN object identification network. In comparison to the Selective Search object proposal method, our model is almost a hundred times quicker while also improving detection accuracy and recall.

[4] Because of the lack of currently accessible Vehicle-to-Infrastructure (V2I) communication in transportation systems, Traffic Light Detection (TLD) is still seen as an essential component of autonomous cars and Driver Assistance Systems (DAS). To remedy the drawbacks of both heuristic vision-based algorithms and deep learning-based techniques, we introduce a low-power, real-time traffic light detector for the autonomous vehicle platform. To locate all potential traffic signals, we employ a heuristic candidate region selection module and a lightweight Convolution Neural Network (CNN) classifier. Offline simulations utilizing the acquired data and a wide variety of publicly accessible datasets demonstrate that our strategy yields greater average accuracy and consumes less time. In order to evaluate the capabilities of our full-scale system for autonomous driving, we drove a car and a bus equipped with it together with ordinary traffic while integrating our detector module on NVidia Jetson TX1/TX2. Our model achieves a 99.3% average detection accuracy (mRttld) at 10Hz on TX1, and a 99.7% average accuracy (mRttld) at 99.8Hz (Rttld). When combined with other autonomous driving modules, our traffic light detecting module has been shown in on-road trials to reach; + 1:5m errors at stop lines.

[5] Recent years have seen several improvements in our daily lives thanks to technology based on deep learning. Image recognition technology is employed in the rapidly growing field of autonomous driving to identify the roadways, white lines, and approaching vehicles. The production cost is quite high, however, because automated cars are controlled by gathering data about the position of the vehicle and its surroundings mostly via image sensors and cameras. The objective of this project is to create image sensor-free autonomous driving system that only uses an on-board visual camera. In Unity's virtual world, automatic parking is implemented utilizing reinforcement learning. It is possible to obtain high accuracy autonomous parking by using the input image as a segmentation image.

[6] Neural networks are a type of artificial intelligence (AI). Deep convolutional neural networks excel at multidimensional signal processing (CNNs). The term "deep learning" is used to describe the processes involved in training networks with a "few" to several dozen or more convolution layers to automatically learn their functional parameters through the analysis of instances in the user's issue domain of choice. To automatically learn features from (often enormous) data bases and to generalize responses to scenarios not encountered during the learning phase, CNNs are finding widespread use. The learnt characteristics can be put to use in a variety of applications, such as signal classification using a CNN. This "Lecture Notes" piece aims to do two things: (a) give a high-level overview of how convolutional neural networks (CNNs) are constructed; and (b) utilize a computational example to show how convolutional neural networks (CNNs) are trained and applied to real-world issues.

[7] In this research, a new method is proposed for automatically locating and separating the desired object or region from a satellite image. This color-based strategy relies on the histogram threshold and the HSV color space. Satellite-downloaded multispectral pictures are commonly represented in the RGB color space. However, the human visual system is more attuned to intensity than it is to color information (hue or saturation). The proposed technique involves converting the satellite image from RGB color space to HSV color space, and then splitting the resulting image into three parts (channels) based on the intensity and hue of the original image. The HSV color space attempts to reflect human vision as accurately as possible. Each of the three variables is then used to generate a histogram (hue, saturation, and value). The threshold is then applied separately to each of the three variables. As a last step, morphological operations including masking, filtering, and smoothing are used to extract the desired area. Consistent experimental evidence supports the proposed approach as successful. Experimental findings demonstrate that the proposed method outperforms the current gold standard for extracting and segmenting a target region or item from satellite photos.

[8] Unchecked, biases might seep into AI systems. Accuracy is generally prioritized in powerful deep learning networks. Here, we apply an iteratively trained unlearning method to a realistic, in-car, urban driving situation in an effort to counteract the biases present in semantic segmentation models. It has been proven that convolutional neural networks rely more on color and texture than on geometry. Consequently, problems may develop while testing safety-critical applications, such autonomous vehicles, and seeing images with covariate shift during the testing period, as a consequence of things like seasonality or fluctuations in illumination. On straightforward datasets like MNIST, the idea of bias unlearning has been established. However, this approach has not yet been used for pixel-level semantic segmentation from extremely heterogeneous training data, such as urban landscapes. Testing trained models for both the baseline and bias unlearning schemes on color-manipulated validation sets revealed a discrepancy of up to 85.50% in mIoU from the original RGB images, confirming that segmentation networks heavily rely on the color information in the training data for their classification.

[9] The development of AI is changing the face of modern society (AI). In order to make self-driving cars a reality, scientists and engineers are making concerted efforts to implement deep learning-based technologies into the automotive industry. However, before being used in mass-produced vehicles, a neural network must undergo a rigorous functional safety examination. The study examines both the pros and cons of using deep learning in autonomous vehicles.

Fig 2: Image segmentation

METHODOLOGY

Module 1: Data Collection and Gathering

We will download the correct dataset for the yolov5 model. This data is already preprocessed as yolov5 is a pretrained model. Ultralytics, who also developed the Pytorch edition of YOLOv3, announced their most recent object detection model, YOLOv5, in June of 2020. Each of the four variants of YOLOv5, s, m, l, and x, provides a unique level of detecting precision and performance.

Module 2: Module Making and Training

As we are using the Yolov5 model we will import the YOLOv5 model and define its parameters and backbone. When one of the pre-defined networks doesn't work for us, YOLO v5 also lets us design our own unique architecture and anchors. We must create a custom weights configuration file for this. The yolov5s.yaml is used in this example. This is how it seems. We will explain it in detail below:

Defining Parameters

We define the following parameters in our model:

- **Number of Classes**

Although there is support for additional pre-trained models, Ultralytic's default model was pre-trained over the COCO dataset. COCO (Common Object Classification and Object Segmentation) is a dataset for object recognition and segmentation that contains images of everyday scenes. The programme already has its various levels set up. Processes have already been applied to it.

By giving if we give our model the name of a pre-trained COCO model in the 'weights' option, it will be initialized with the weights from that model. It's automated, so the pre-trained model will get downloaded as soon as it's available.

- **Adding the Final Layer**

Part of the model called Yolo. In the last phase of detection, the use of the head is essential. After applying anchor boxes to the features, the system produces final output vectors including class probabilities, objectness scores, and bounding boxes.

The YOLO V5 has the same model head as the V3 and V4. It creates small, medium, and large-scale feature maps to facilitate multiscale prediction. This enhances the model's capacity to make predictions over a range of sizes.

Module 3: Creating CLI Based Program

1) Webcam Input

We start by opening the file for our project and writing the code into it. After that, we imported our necessary packages, which included OpenCV and NumPy. Arguments passed via the command line are processed during the current runtime, which enables us to modify the parameters of our script directly from the terminal. We pass four command line arguments:

- Image: Indicates the location of the picture to be read in. Using YOLO, we will search for things contained within this image.
- YOLO: The base path to the YOLO directory, specified with the —yolo option. After that, in order to carry out the object detection procedure on the image, our script will load the necessary YOLO files.
- Confidence: The threshold probability used to filter out insufficient detections. Although I've set this to have a default value of 50% (0.5), you are more than welcome to play with with different values for it.
- Threshold: This is the threshold for non-maxima suppression, and its default value is 0.3.

At long last, we load our class labels and assign each one a colour chosen at random. After that, we load YOLO from disc after having derived the paths to the YOLO weights and configuration files. After loading the image and transmitting it over the network, we then initialize the following lists:

- boxes: These are the bounding boxes that we have drawn around the object.
- confidences are defined as the value of trustworthiness that YOLO bestows upon an object. When the confidence levels for an item are low, it suggests that the object might not be what the network believes it to be. Keep in mind that we will filter out any objects that don't reach the 0.5 criterion as stated in the previous section about our command line options.
- Class IDs: are the object's class label that was discovered.

Because YOLO does not automatically apply non-maxima suppression for us, we are need to manually apply it. When non-maxima suppression is used, highly overlapping bounding boxes are eliminated, and only the most confident of the remaining bounding boxes are retained. In addition to this, NMS checks to make sure that we do not have any redundant or unnecessary bounding boxes. After that, we proceed to add the class text and the boxes to the image. Finally, we get the image that looks like the following:

2) Video Input

To start, we'll go over our imports and the options for the command line. The image argument that was present in the previous script is not present in this one. As a replacement for it, we now have two arguments that are connected to video:

- input: The directory where the input video file is located.
- output: The location of the video file that was created.

In light of these considerations, we now have the option of using either the videos that we capture using our smartphones or the videos that we locate online. After that, we process the video file, which results in an output video with annotations. It is also feasible for us to use our webcam to process a live video stream.

First, we load the labels, then we produce colors, and last, we load our YOLO model and decide what the output layer names will be. After that, we are prepared to begin processing the frames one at a time. Following that, we will carry out a forward pass of YOLO using the input that is now being provided by our frame. Then, we follow the same steps as for the webcam input and we derive a function for the same.

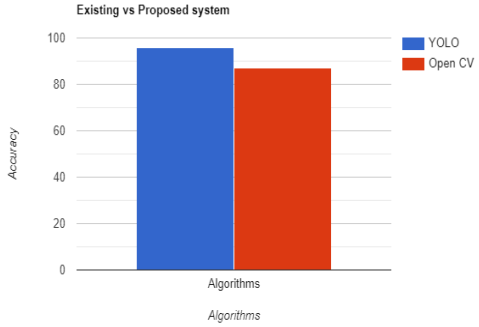

Figure 3: Accuracy comparison of existing and proposed system

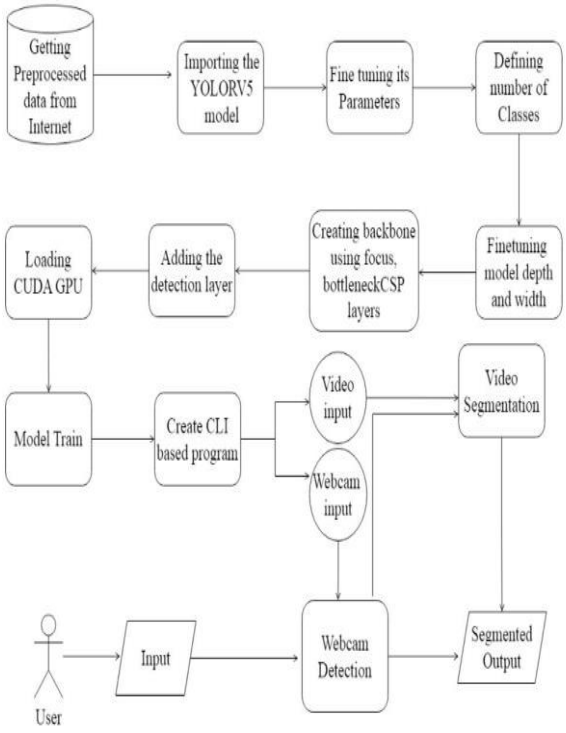

Figure 4: System Architecture

COMPARISON OF EXISTING AND PROPSED SYSTEM

EXISTING SYSTEM:

The technology in place only focuses on image detection. During the testing and training phases, there is not a suitable test-train divide in place. The current approach uses a very small number of photos, which makes the dataset training quite ineffective. The current system also lacks a solid architecture basis.

PROPOSED SYSTEM:

The suggested method can be applied immediately and is extremely accurate. The projected system's production version makes use of the ICNet network. The proposed approach can discriminate between up to six different categories of objects, including people, cars, buildings, barriers, etc. The recommended system accepts input from both images and videos and uses semantic segmentation.

CONCLUSION

Real-time picture segmentation is essential for many uses of autonomous vehicles. To make it practical for embedded systems and autonomous driving, the computational load of semantic segmentation must be reduced. Semantic segmentation's end goal is to provide a category name to every single image pixel. The discussed models offer realistic options for meeting real-world requirements, including essential applications like road scene comprehension, environmental awareness, and self-driving rules.

FUTURE WORK

The future work includes deploying this project in a cloud-based environment and also improving the accuracy of the algorithms by doing more epoch training. Multiple algorithms can also be tested with the dataset for improving performance and efficiency of the project.

REFERENCES:

1. Ruturaj Kulkarni, Shruti Dhavalikar, Sonal Bangar, "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning", Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2019
2. Ze Liu, Yingfeng Cai, Hai Wang, Long Chen, Hongbo Gao, Yunyi Jia, Yicheng Li, "Robust Target Recognition and Tracking of Self-Driving Cars With Radar and Camera Information Fusion Under Severe Weather Conditions", IEEE Transactions on Intelligent Transportation Systems (Volume: 23, Issue: 7), 2022
3. Ramin Nabati, Hairong Qi, "RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles", IEEE International Conference on Image Processing (ICIP), 2019
4. Zhenchao Ouyang, Jianwei Niu, Yu Liu, Mohsen Guizani, "Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles", IEEE Transactions on Mobile Computing (Volume: 19, Issue: 2), 2020
5. Rikuya Takehara, Tad Gonsalves, "Autonomous Car Parking System using Deep Reinforcement Learning", 2nd International Conference on Innovative and Creative Information Technology (ICITech), 2021
6. Malvi Mungalpara, Priyanka Goradia, Trisha Baldha, Yanvi Soni, "Deep Convolutional Neural Networks for Scene Understanding: A Study of Semantic Segmentation Models", International Conference on Artificial Intelligence and Machine Vision (AIMV), 2022
7. A.A. Mahersatillah, Z. Zainuddin, Y. Yusran, "Unstructured Road Detection and Steering Assist Based on HSV Color Space Segmentation for Autonomous Car", 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2021
8. Jack Stelling, Amir Atapour-Abarghouei, "'Just Drive': Colour Bias Mitigation for Semantic Segmentation in the Context of Urban Driving", IEEE International Conference on Big Data (Big Data), 2022
9. Tuan Pham, "Semantic Road Segmentation using Deep Learning", Applying New Technology in Green Buildings (ATiGB), 2021
10. Sanchit Gautam, Tarosh Mathuria, Shweta Meena, "Image Segmentation for Self-Driving Car", 2nd International Conference on Intelligent Technologies (CONIT), 2022