Go, change the world

RV Educational Institutions ®
RV College of Engineering ®

Autonomous
Institution Affiliated
to Visvesvaraya
Technological
University, Belagavi

Approved by AICTE,
New Delhi, Accredited
By NAAC, Bengaluru
And NBA, New Delhi

Lane Detection and Traffic Sign Detection for Autonomous Vehicles Using Carla Simulator

A MINOR PROJECT-18TE64 REPORT

Submitted by

Samudyata A	**1RV19ET050**
Hithaishi Surendra	**1RV19ET064**
S Bhuvana	**1RV19ET065**

Under the guidance of

Mr. Mohana
Assistant Professor
Department of Electronics and Telecommunication Engineering
RV College of Engineering

Submitted in partial fulfillment for the sixth semester examination of

Bachelor of Engineering
in
Electronics and Telecommunication Engineering

2021-2022

RV COLLEGE OF ENGINEERING®, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

CERTIFICATE

Certified that the minor project work titled ***"Lane Detection and Traffic Sign Detection for Autonomous Vehicles Using Carla Simulator"*** is carried out by **Samudyata A (1RV19ET050), Hithaishi Surendra (1RV19ET064) and S Bhuvana (1RV19ET065)** who are bonafide students of RV College of Engineering, Bengaluru, submitted in partial fulfilment for the sixth semester examination of **Bachelor of Engineering** in **Electronics and Telecommunication Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2021-2022. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the minor project report deposited in the departmental library. The minor project report has been approved as it satisfies the academic requirements in respect of minor project work prescribed by the institution for the said degree.

Signature of Guide	**Signature of Head of the Department**
Mr. Mohana	**Dr. K. Sreelakshmi**
Assistant Professor	**Professor and Head,**
Dept. of ETE, RVCE	**Dept. of ETE, RVCE**

External Viva

Name of Examiners	**Signature with Date**
1.	
2.	

RV COLLEGE OF ENGINEERING®, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

DECLARATION

We, **Samudyata A, Hithaishi Surendra and S Bhuvana,** students of sixth semester B.E., Department of Electronics and Telecommunication Engineering, RV College of Engineering, Bengaluru, hereby declare that the minor project titled ***"Lane Detection and Traffic Sign Detection for Autonomous Vehicles Using Carla Simulator"*** has been carried out by us and submitted in partial fulfilment for the sixth semester examination of **Bachelor of Engineering** in **Electronics and Telecommunication Engineering** during the year 2021-2022.

Further we declare that the content of the dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Names **Signature**

1. **Samudyata A (1RV19ET050)**
2. **Hithaishi Surendra (1RV19ET064)**
3. **S Bhuvana (1RV19ET065)**

ACKNOWLEDGEMENT

We are indebted to our guide**, Mr. Mohana,** Assistant Professor**, Dept of Electronics and Telecommunication Engineering** for his wholehearted support, suggestions and invaluable advice throughout our project work and also helped in the preparation of this thesis.

Our sincere thanks to **Dr. K Sreelakshmi**, Professor and Head, Department of Electronics and Telecommunication Engineering, RVCE for her support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for his appreciation towards this project work.

We thank all the **teaching staff and technical staff** of Department of Electronics and Telecommunication Engineering, RVCE for their help.

Lastly, we take this opportunity to thank our **family** members and **friends** who provided all the backup support throughout the project work.

Student Names
Samudyata A
Hithaishi Surendra
S Bhuvana

Abstract

Deep learning is a rapidly expanding machine learning method for perceiving and comprehending vast volumes of data. The idea of autonomous driving has drawn a lot of interest and discussion in recent years because it is thought to have numerous advantages for both individuals and society as a whole, including improved road safety, decreased traffic congestion, accidents, and fatalities, as well as reduced time and pollution associated with commuting.

However, the safety of autonomous vehicles and the reduction of traffic accidents are still crucial concerns. Lane identification and traffic sign detection is the most challenging and promising problem for self-driving cars since statistics reveal that unintentional lane departure and ignorance of traffic signs are major contributing factors to motor vehicle collisions around the world. To tackle this problem in this study, we aim at helping autonomous vehicles to detect both lane and traffic signs. Due to the remarkably dynamic environment and inability to test the system in the vast range of scenarios it may meet after deployment, designing a controller for autonomous vehicles that can provide appropriate performance in all driving scenarios is tough. Hence the environment is set up virtually with the help of the CARLA simulator. Deep learning techniques such as Semantic Segmentation and Object Detection hold considerable potential for generalizing previously acquired rules to novel circumstances in addition to offering good performance for complicated and non-linear control issues. With aid of Semantic segmentation which is the pixel-wise classification of images and object detection techniques, the lane and traffic signs were successfully detected. The applications are implemented using a variant of CNN known as SegNet which is a semantic segmentation model and YOLO which is a state of art object detection algorithm. For recognition of text present on the traffic signs, Tesseract OCR was deployed. Both the models were tested under various constraints such as towns, vehicles, and weather conditions. It was observed that the lane detection model performed well in dynamic weather conditions resulting in an average accuracy of 93.66% and performed consistently in classifying the left lane and right lane. Further, the traffic sign detection model was trained to detect four classes of speed signs namely 30km, 60km, 90 km and STOP signs. The model was tested and it showed an overall 92.58% accuracy with the best-observed case corresponding to the STOP sign.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADAS	-	Advanced Driver Assistant System
API	-	Application Programming Interface
CARLA	-	Car Learning to Act
CNN	-	Convolutional Neural Network
CNYK	-	Cyan Magenta Yellow Key
FCL	-	Fully Connected Layers
GPU	-	Graphic Processing Unit
HSV	-	Hue Saturation Value
HUD	-	Heads up Display
MSER	-	Maximally Stable Extremal Regions
OCR	-	Optical Character Recognition
RCNN	-	Region Based Convolutional Neural Network
ReLU	-	Rectified Linear Units
RGB	-	Red Green and Blue
SCNN	-	Sparse Convolutional Neural Network
TSDR	-	Traffic Sign Detection and Recognition
VGG	-	Visual Geometry Group
VP	-	Vanishing Point
YOLO	-	You Only Look Once

Chapter 1

Introduction

1.1Background

Over the last ten years, several attempts have been undertaken to improve algorithmic performance and provide datasets for semantic segmentation. Recently, substantial progress has been made in this topic, which is a subset of visual scene interpretation, owing to the contribution of deep learning approaches. Here, some of the most extensively used and open urban semantic segmentation datasets for self-driving car applications are explored and identified.

Semantic segmentation is the process of giving a class label to each pixel in an image from a collection of predetermined categories. Deep learning's immense success has had a significant influence on semantic segmentation algorithms, significantly enhancing their accuracy. Many technical and academic disciplines that demand high-end computer vision capabilities have taken notice of this potential breakthrough. Self-driving vehicles, for instance, must comprehend their surrounding environment, which includes other automobiles, pedestrians, road lanes, traffic signs, and traffic signals. Due to the exceptional accuracy of deep neural networks in detection and multi-class identification tasks, semantic segmentation based on deep learning is a significant choice for achieving this aim. A semantic segmentation model is displayed in Fig.1.1 containing the input image, prediction and ground truth of an urban street.

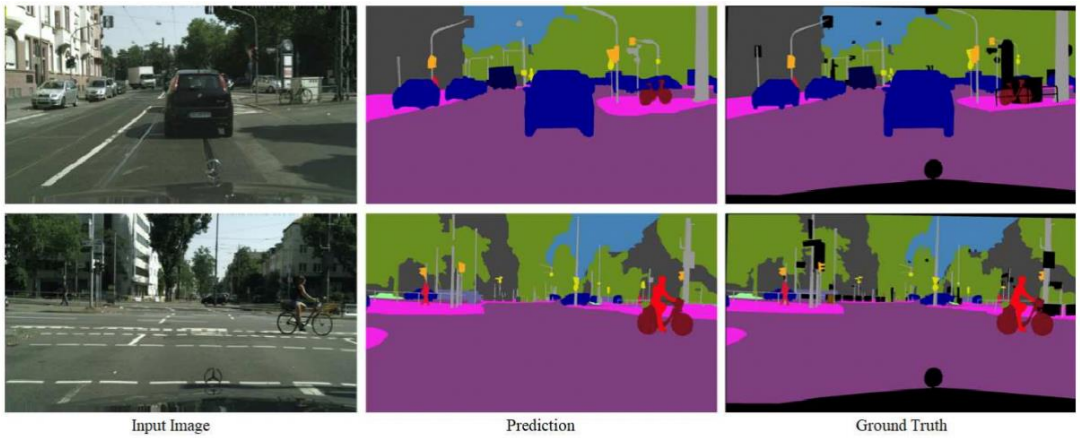

Fig.1.1: Semantic Segmentation [3]

In the near future, more research works can be concentrated on developing end-to-end

models to perform both instance and semantic segmentation simultaneously. More concentration on detecting smaller objects, removal of unnecessary small objects and other miscellaneous objects can be given. Moreover, improving the performance of panoptic segmentation models and widening their applications are among the relevant future orientations, especially in digital health, real-time self-driving, scene reconstruction and 3D/4D Point Cloud Semantic Segmentation, as it is explained in the following subsections.

According to a recently conducted survey, roughly 1.3 million people die each year as a result of traffic accidents, or nearly 3287 each day. Human error and careless driving are the main causes of many collisions. Advancement of technology can play a critical role in preventing more tragedies and saving lives.

By removing the human aspect of driving, the goal is to eliminate unforeseen errors in order to further preserve human life. The Society of Automotive Engineers (SAE) defines 6 levels of driving automation ranging from 0 (fully manual) to 5 (fully autonomous). Currently, only Level 2 is available in the market.

Annotating real-time datasets is a tedious and time consuming process, which may be cut down by using synthetically generated datasets. Hence, by making strides in this field of study, it is possible to improve the autonomous driving technologies. Since the goal of cutting-edge technology is to provide maximum user convenience, the development of self-driving cars will help achieve this.

1.2 Literature Survey

Qusay Sellat *et al.,* [1] The paper proposes designing of an accurate and real-time semantic segmentation system for self-driving cars using two main deep learning architectures Convolutional Neural Networks(CNNs). A hybrid model based on the concepts of feature pyramid networks (FPNs) and bottleneck residual blocks is built. Proposed model is trained and tested using CamVid common dataset and also used data augmentation techniques to increase the training quality. Proposed model is implemented using Python frameworks TensorFlow, Keras, and Albumentations. The code was run on

NVIDIA Tesla P100-PCIE-16 GB. Better performance on the Mean Intersection over

Union (mIoU) metric can be a topic of future research. Other models different from MobileNetV2 can be made a backbone for the designed models.

Maksims Ivanovs *et al.*, [2] .The goal of study was to investigate whether it is possible to improve the quality of semantic segmentation by augmenting real-world images of urban scenes with synthetic images generated with the open-source driving simulator Car Learning To Act(CARLA). Cityscapes dataset was used as a source dataset and two experiments were performed with DEEP Neural Networks(DNN) architectures from the Deeplab library, MobileNetV2 and Xception. In the first series of experiments, the Cityscapes dataset was augmented with synthetic images from MICC-SRI dataset but because of low degree of photorealism of synthetic images, the augmentation led to the deterioration. In the second series of experiments, synthetic images from a more recent version of CARLA were used for augmentation which achieved better accuracy. Best result was achieved on CCM-50 split, whereas in the case of Xception, best result was achieved on CCM-25 split. This indicates for future work on more elaborate methods of dataset augmentation with synthetic data.

Mohammad Hosein Hamian *et al.*,[3] The proposed method is divided into four units: Semantic Segmentation Unit (SSU), Classical Segmentation Unit (CSU), Boundary Correction and Merge Unit (BCMU), and conditional random field unit (CRFU). In this paper Cityscapes dataset is used for fine-tuning the SSU. For the next version of DeepLab, DeepLab v2 which takes into account the problem of objects of different scales as much as possible, Atrous Spatial Pyramid Pooling (ASPP) is used at the end of the architecture. DeepLab v3+, has been used and the results were improved by using classical segmentation to modify object's boundaries with the help of the information of the regions obtained from K-means. Using this post-processing method and one of the conventional post-processing, Conditional Random Field (CRF) for the two backbones, Xception and MobileNetV2, good results were achieved with 81.73% and 76.31% mIoU, respectively.

Omar Elharroussa *et al.*, [4] In this paper, existing panoptic segmentation techniques was reviewed. Standard panoptic segmentation benchmarks, including Cityscapes, Common Objects in Common(COCO), Mapillary Vistas, Pascal VOC 2012, ADE20K, Karlsruhe Institute of Technology and Toyota Technological Institute(KITTI), and Semantic KITTI

were used. Firstly, extensive critical review of the panoptic segmentation technology was proposed. Accordingly, the background of the panoptic segmentation technology was first presented. Next, a taxonomy of existing panoptic segmentation schemes was performed based on the nature of the adopted approach, type of image data analysed, and application scenario. Datasets and evaluation metrics were discussed and the most pertinent works were tabulated for a clear comparison of the performance of each model. The Performance Qualification(PQ) of 69% on the Cityscapes dataset and 50% on the COCO dataset were the best results from all the models which indicated significant improvement in performance.

Ilias Papadeas *et al.*, [5] In this paper, an overview of the best methods and architectures for real-time semantic segmentation has been presented. The datasets used are: Cityscapes, CamVid, MS COCO—Common Objects in Context, KITTI, KITTI-360, SYNTHIA, Mapillary Vistas, ApolloScape, RaidaR. Furthermore, tables listing the most accurate and efficient state-of-the-art real-time models are provided. The selection of the chosen models is based on experiments made using “Cityscapes” and “CamVid”datasets, depending on their mIoU, FPS and inference time they achieved. For studying purposes, a list of extensively used datasets and metrics of real-time semantic segmentation was described. Finally, this survey discusses current issues, thus showing areas of improvement in real-time semantic segmentation regarding the needs of autonomous driving and other high-end technological fields.

Jack Stelling *et al.,* [6] the aim of this paper is to mitigate colour bias from semantic segmentation models trained on urban driving scenes. In this paper, colour bias unlearning scheme was applied to highly variable images of urban road scenes as an iterative learning process during training. The contribution empirically shows that semantic segmentation architectures do overfit to the colour within training data, and they struggle to generalise to unseen test data. In the worst case, when validating on a set with a colour invert transformation, reductions of 85.50% were observed. Observing a 62% increase in mIoU score for colour invert; When neglecting the result for colour invert, it is still observed as an average increase of 1.5% over all validation set manipulations tested. Furthermore, an average increase of 14.5% is observed for the “human” class, enhancing pragmatic performance in a safety-critical application such as autonomous

driving. This paper can be positioned to push towards robust, trustworthy technology alleviating algorithmic bias.

Çağrı Kaymak *et al.*,[7] in this paper an application about semantic image segmentation is carried out in order to help autonomous driving of autonomous vehicles. This application is implemented with Fully Convolutional Network (FCN) architectures obtained by modifying the (CNN) architectures based on deep learning. Experimental studies for the application have utilized 4 different FCN architectures named FCN-AlexNet, FCN-8s, FCN-16s and FCN-32s. For application, firstly, FCNs are trained separately and the validation accuracy of these trained network models is compared on the SYNTHIA-Rand-CVPR16 dataset. In addition, image segmentation inferences in this paper are visualized to see how precisely the used FCN architectures can segment objects. Maximum validation accuracies of 92.4628%, 96.015%, 95.4111% and 94.2595% are achieved with FCN-AlexNet, FCN-8s, FCN-16 and FCN-32s models trained using weights in pre-trained models, respectively. When training times of FCN models are compared, it is seen that the training time spent on FCN-AlexNet is about one-fourth of the other FCN models with very close training times. Therefore, it can be easily stated that the most suitable model for the application is FCN-8s. Obtained experimental results show that FCNs from deep learning approaches are suitable for semantic image segmentation applications.

Laura von Rueden *et al.,* [8] this paper proposes an approach of street map based validation of semantic segmentations which offers a new way to support the goal of safe artificial intelligence in autonomous driving. In this approach the segmentation in bird's eye view is combined with field of view in street map and uses this overlay to compute validation errors. It has introduced two new validation metrics called Intersection over Segmentation (IoS) and Intersection over Map (IoM) that it used to identify segmentation masks with false positive and false negative road segments. Furthermore, it has developed an algorithm that can correct inaccurate vehicle positions by finding the best overlap with ground truth segmentation. It has performed an experimental study with the Cityscapes dataset and OpenStreetMap, and showed that road segmentation errors can indeed be detected by the proposed validation procedure. Besides the validation approach, a method to correct the vehicle's GPS position is presented. Lastly, it has presented

quantitative results on the Cityscapes dataset indicating that validation approach can indeed uncover errors in semantic segmentation masks and even better results can be achieved with high definition maps.

Kuo-Kun Tseng *et al.*,[9] this paper proposes a fast one-stage multi-task neural network for instance segmentation. A one-stage method is proposed to do object detection with the regression module and use a concurrent segmentation task to perform the segmentation in the same framework. Proposed new model intends to fuse one-stage object detection to realise instance segmentation and is named Fast instance segmentation with One-stage Multi-task neural Network (FastOMNet). It employs a new feature extraction, which can return the position of the detection frame according to feature map while classifying object target. Module of regression box generation maps features into a fixed-size bounding box, which performs integrated ROI (Region of Interest) Align and Mask Prediction operations on the feature maps. For experiment, algorithms are verified with two public datasets, the COCO and CityScapes datasets. In summary of this work, combining the YOLOv3 and Mask-RCNN algorithms, a fast instance segmentation algorithm is proposed: (1) FastOMNet and Advanced FastOMNet are proposed with Average Precision(AP) as 22.3 and 24 and with Frames per second(FPS) as 26.1 and 23, respectively, which proves that the method can speed up the mainstream instance segmentation with 5 times speed performance. Experiments with different combinations of modules have verified the effectiveness, accuracy and that algorithm can speed up the inference speed of the model.

Tejas Mahale Chaoran Chen *et al.,* [10] proposed a model for lane detection using semantic segmentation using Global Convolution Networks (GCN) model and it can address both classification and localization issues for semantic segmentation of a lane. A real time video transfer system is built to get video from car, get the model trained in edge server (which is equipped with GPUs), and send the trained model back to the car. Around 3000 images were gathered from the CARLA simulator as the dataset. Environment used here is distribution.Conda environment along with tensor flow gpu v1.10, Keras gpu v2.2.2, and python version 3.5.For sending the real time video a Picamera on Raspberry Pi was being used. In order to measure its performance various parameters such as Minimum Square Error and Mean Absolute Error were measured.

Minimum Square Error was found to be 57.5875 and the Mean Absolute Error was about 2.2104. Overall performance measure achieved from the proposed model was 57.5875%. In order to achieve state-of-art performance a residual-based boundary refinement and Adam optimization techniques were employed.

Jiaxing Sun *et al.,* [11] designed a work that uses shallow neural networks to achieve semantic segmentation for intelligent transportation systems. Evaluation Metric used is mIoU which is used to evaluate the model and compare it with other advanced methods on the Cityscapes and CamVid datasets. Experiments show that this method achieves high accuracy and comparable speed on the Cityscapes and CamVid datasets. Major advantages using this model are that firstly it is based on multi-feature fusion, showing high accuracy on the Cityscapes and CamVid datasets and second is that it is lightweight and has relatively good speed. Experimental results show that the proposed model achieved a better balance between speed and accuracy than the existing semantic segmentation methods. On the Cityscapes dataset, this method can achieve 75.0% mIoU, which is 0.2% higher than the better-performing BiSeNet.

Shruti Jadon *et al.,* [12] proposed a system that introduces SemSegLoss, a python package consisting of some well-known loss functions widely used for image segmentation. It is developed with intent to help researchers in the development of novel loss functions and perform an extensive set of experiments on model architectures for various applications. Also ease-of-use and flexibility of the presented package have reduced development time and increased evaluation strategies of machine learning models for semantic segmentation. Furthermore, different applications that use image segmentation can use SemSegLoss because of the generality of its functions. Future Work to include addition of more advanced loss function techniques such as Correlation Maximized, Structural Similarity Loss, and Distance map derived loss penalty term for medical-based image segmentation.

Mehmet Aygun *et al.,* [13] objective is to create a unified space-time perspective to the task of 4D LiDAR panoptic segmentation, and pose detection/segmentation/tracking jointly as point clustering which can effectively leverage the sequential nature of data and process several LiDAR scans while maintaining memory efficiency. Also a point-centric evaluation protocol is adapted that fairly weights semantic and association aspects of this

task and summarizes their final performance with a single number. Furthermore, a test bed for this task is established, which is used to thoroughly analyze the model's performance and the existing LiDAR panoptic segmentation methods used in conjunction with a tracking-by-detection mechanism. SemanticKITTI LiDAR dataset is used to conduct the experiments. Experimental results show that both 2 and 4 scan versions models can track the instance correctly, but due to slight difference in semantic segmentation predictions, Multiple object tracking and Segmentation Accuracy (MOTSA) scores differ drastically. Also it was observed that 4-scan variant performs better than the 2-scan variant in terms of LSTQ.

Javier Corrochano *et al.*, [14] proposed a technique that involves automating the segmentation and semantic-labeling process based on combination of scenarios with chromas and superposition of images in indoor scenarios. The dataset for this task was collected while driving with the circuit placed in the target scenario. Various advantages of proposed model concern the decomposition of learning task into two tasks: feature extraction and driving learning. Further, development of the system is more flexible, making it possible to approach the work incrementally and can reduce required training time. In validation tests, the driving model was found to be robust to changes in lighting on stage, as well as to shadows or to the presence of obstacles on the track. In the presence of an occlusion on one of the lines, the driving model is able to follow the other line. In addition, it was found that the car was able to follow a different circuit than the one used for training. Future work include developing a more elaborate proposal based on a mixed reality system, using generative methods to induce variations in lighting and perspective, and including more elements (road signs, for example) in the semantic segmentation which would allow the merging of the training process in both simulators and real environments, retaining the advantages of both.

Dongkyu Lee *et al.*, [15] paper proposes an autonomous vehicle driving camera calibration system that is low cost and utilizes low infrastructure. The proposed network predicts human-like poses from a single image. In this study MORAI Sim Standard as a simulator was used to acquire the dataset. Segmentation backbone-based features were used during the study and the performance difference before and after attachment were analyzed by adding on a pose regressor. Various performance metrics employed are the

seg loss, LUT and pose losses. Furthermore the performance was improved through the combination with a pose regressor. All tests were referenced in the NVIDIA GTX 1080 environment, and three losses were evaluated. It was observed that pose regressor to the end of the encoder needs to be connected to obtain the direction of the entire network. Future improvements include conducting research using actual camera data (or actual + synthetic data) in this network and try to reduce aliasing by improving the network. In addition, to supplement the characteristics of a single camera, which makes it difficult to estimate scale, it is intended to produce a real distance-based BEV with an explicit unit rather than a relatively real distance through combination with a ToF sensor or other odometry methods, as with adjacent frames of a single camera.

Libo Wang *et al.,* [16] proposes a novel bilateral structure composed of convolution layers and transformer blocks for understanding and labeling very fine resolution urban scene images. It employs linear attention to reduce the fitting residual and greatly improves the generalization of fused features. Performance of BANet on the ISPRS Potsdam dataset is evaluated using the overall accuracy (OA), the Mean Intersection over Union (mIoU), and the F1 score (F1). Experiments conducted on the three large-scale urban scene image segmentation datasets, i.e., ISPRS Vaihingen dataset, ISPRS Potsdam dataset, and UAVid dataset, demonstrate the effectiveness of BANet. Specifically, a 64.6% mIoU was achieved on the UAVid dataset. Here, main application scenario of the method is urban scene segmentation using remotely sensed images captured by satellite, aerial sensors, and UAV drones. This model considers both accuracy and complexity, revealing enormous potential in illegal land use detection, real-time traffic monitoring, and urban environmental assessment. In future, they aim to continue to study the hybrid structure of convolution and Transformer and apply it to a wider range of urban applications.

Jean-Jacques Ponciano *et al.,* [17] research compares a machine learning approach developed by IPM and a knowledge-based approach called KnowDIP , developed by i3mainz. The DL approach is a supervised deep learning solution that uses a VGG6 Encoder with an FCN8 Architecture on pure RGB images. The KB approach uses standard semantic web technologies such as SPARQL and OWL2. Experimental results have been assessed using quantitative metrics that include recall, precision, F1, and IoU

scores. These deep learning and knowledge-based approaches produced a semantic segmentation with an average F1 score of 0.66 and 0.78, respectively. Therefore, it was observed that the KB approach showed strength in the semantic segmentation of objects with distinct characteristics such as streetlights. In contrast, the DL approach detected less structured objects like bushes effectively .Future work includes extending this study to more data and objects to confirm the results, improving the DL approach by reducing the projection error and improving the KB approach by adding more data features such as color and exploring an approach that combines knowledge and deep learning.

Kunyu Peng *et al.*,[18], in this paper, the concept of MASS - a Multi-Attentional Semantic Segmentation framework is introduced. Unlike previous works which mostly focus on sparse segmentation of the LiDAR input, this paper deals with dense top-view segmentation. MASS is primarily comprised of three algorithms: (1) Multi-Attention (MA) mechanisms, (2) a pillar feature net (PFN), and (3) a modified UNet (M-UNet) utilized for dense top- view semantic segmentation. For experimentation purposes, the proposed model is first evaluated on the SemanticKITTI dataset and then validated on the nuScenes-LidarSeg dataset. Results show that MASS surpasses the existing state-of-the-art methods. Future work includes investigating cross-dimensional semantic mapping along with unsupervised domain adaptation and dense contrastive learning strategies for uncertainty-aware driver behavior and holistic scene understanding.

Sheng Lu *et al.*, [19], due to the unsatisfactory results from existing lane detection methods, this paper proposes a new system which is fast and robust by making use of a combination of semantic segmentation network and an optical flow estimation network. The whole model consists of three sub-networks or algorithms, namely: (a) a semantic segmentation network responsible for detecting lane robustly (b) an optical flow estimation network to find out the spatio-temporal information and track lanes fast and finally (c) an adaptive scheduling network to schedule the optical flow estimation network and the segmentation network adaptively. In addition to these, a density-based spatial clustering of applications with noise (DBSCAN) was utilised to provide feedback for automatic driving. In, terms of experimentation and results, the model proved to be three times faster with its accuracy reduced by 2% at most. The relative error is about 3%. Future entails reducing fitting error of the developed model.

Christos Sakaridis *et al.,* [20], **s**ince most of the existing datasets on semantic segmentation consist of images from clear weather conditions, this paper aims to address this issue by creating a large-scale driving dataset specialised to adverse conditions (fog, nighttime, rain and snow). The dataset supports uncertainty-aware semantic segmentation along with the standard semantic segmentation. A specialised annotation protocol and a dedicated performance metric, termed average uncertainty-aware IoU (AUIoU) is introduced for the separation of labeled pixels into invalid and valid is encoded in a binary mask. For experimentation purposes, four main directions is considered: evaluation of models pre-trained on normal conditions, supervised learning in adverse conditions, unsupervised and weakly supervised normal-to-adverse domain adaptation, and evaluation of uncertainty-aware semantic segmentation baselines and oracles. Roads in Switzerland is considered for the purpose of data collection and the process is accomplished using a 1080p GoPro Hero 5 camera. Finally, about 4006 images are collected successfully. The aim of this ACDC is to enable researchers to jointly use this dataset along with existing normal-condition datasets for training in order to regularise models better and improve their performance both under normal and adverse conditions.

Lidia Fantauzzo *et al.*,[21], **t**his paper proposes a new machine learning paradigm known as FedDrive due to the absence of research explicitly addressing the challenges of federated learning in semantic segmentation. It focuses on the real world challenges of statistical heterogeneity and domain generalisation. Here, state-of-the-art federate learning methods and algorithms are introduced and explained in great detail such as FedAvg. Two popular and widely used datasets Cityscapes and IDDA is employed for implementation and experimentation purposes. A BiSeNet architecture is used for implementation of the concept. A comparative analysis of both datasets is performed and the results are tabulated in extensive detail. The aim of this paper is to make FedDrive publicly available to the research community in order to encourage more research in the subject.

Daniel Maturana *et al.*,[22], **m**ost of the research work in autonomous vehicles focuses on urban driving. This paper, however, proposes a semantic mapping system, for off-road driving with All-Terrain Vehicles (ATVs). Having observed that relying primarily on geometric information leads to disappointing results for autonomous navigation in off-

road environments, this paper aims to counter the problem by building a semantic map which has a representation of the vehicle's surroundings encoding both geometric and semantic information. The map is stored as a 2.5D grid centred on the vehicle frame and is continuously updated as new sensor data is acquired. A custom Convolutional Neural Network (CNN) is built for this purpose and trained with a novel dataset of labelled off-road imagery. RGB imagery and LiDAR point cloud data are two primary sensor streams employed. The hardware platform makes use of an NVIDIA GT980M GPU and all computers run Ubuntu Linux. C++ and Python are used for implementation using CUDA. The two datasets used are DeepScene Dataset and Yamaha-CMU Off-Road Dataset. Qualitative and Quantitative results are observed and field experiments are performed as well. The future scope of this paper includes discovering alternatives to manual labelling, such as self-supervision and inverse reinforcement learning.

Sampo Kuutti *et al.*,[23], as the title of the paper suggests, this research paper provides a very extensive and thorough survey of the existing Deep Learning applications pertaining to autonomous vehicle control. The focus of the paper is on vehicle control, rather than the wider perception problem which includes tasks such as semantic segmentation and object detection. Lateral Control Systems, Longitudinal Control Systems and Simultaneous Lateral & Longitudinal Control Systems are the three approaches explored in detail. The paper also emphasises how much of the current research is only limited to simulation. While testing in simulation is useful for feasibility studies and initial performance evaluations, extensive testing and training in the field will be required before these systems are ready for deployment. The primary challenges identified by this survey include computation, selecting the appropriate architecture, adequate goal specification, adaptability and generalisation, safety and verification. The future work aims to design autonomous vehicles which can understand the rules of the road and the behaviour of other road users.

Biao Gao *et al.*,[24], unlike most of the other available papers on this subject, this particular research paper focusses on fine grained off-road semantic segmentation based on active and contrastive learning. It proposes a model with three distinct features, namely: (i) No pixel-wise annotated datasets (ii) No predefined semantic categories and (iii) Adaptation to new scenes actively. To summarise, an off-road dataset is developed in

this research containing three subsets of different scenes with a total of 8000 image frames and the public DeepScene dataset. Four major experiments were conducted to test the performance of the intended research outcomes. They are as follows: (a) Contrastive Learning for Feature Representation (b) Adaptive Category Modelling (c) Results of Active Learning and (d) Results of Active Learning across Datasets. The extensive experiments show improved results when compared to existing datasets. The future work will be focused on preventing models from catastrophic forgetting after adapting to new scenes.

Pengchuan Xiao *et al.*,[25], this paper introduces PandaSet, a multimodal open source dataset for training autonomous driving machine learning models. It provides a complete kit of high-precision sensors covering a 360◦ field of view. The dataset features 28 different annotation classes for each scene as well as 37 semantic segmentation labels for most scenes. It is designed to assist drivers during challenging driving conditions with upto level 4 and 5 driving autonomy. It also provides LiDAR-only 3D object detection, LiDAR-camera fusion 3D object detection, and LiDAR point cloud segmentation. For the purpose of data collection, a Chrysler Pacifica minivan mounted with a sensor suite of six cameras, two LiDARs, and one GNSS/IMU device is used to collect data in Silicon Valley. A frame-based data structure is used to encapsulate point cloud and image data. The future objectives of the paper is to design evaluation metrics and build a public leaderboard to track the research progress in 3D detection and point cloud segmentation, and add map information to the dataset.

P. Shunmuga Perumal *et al.,* [26], focus of this paper is on road safety aspect of Advanced Driver Assistance Systems (ADAS) by exploring Crash Avoidance and Overtaking Advice (CAOA) subsystems. It highlights the issues in human's driving intelligence and provides insights on resolving them. Challenges such as perception errors, driver's blind spots and inattention are cited and many algorithms are suggested in order to overcome them. Optical flow algorithm, image processing techniques to monitor eye closure frequencies and sleepy driving behaviour of drivers, Strategy Selection and Transition (SSAT) approach and fuzzy logic are some of them. The constraints and obstacles of operating environments is explained and functional architecture of a typical CAOA system is suggested. Sensors like camera, LiDAR, RADAR, IMU, GPS

and speedometer are used in this architecture. Certain simulations and experiments are conducted in MATLAB and other platforms in order to measure the performance metrics. As a whole, this article throws light on various aspects of CAOA subsystems to assist the researchers, who work on CAOA systems for road safety.

Inference from literature survey

After reviewing the above-mentioned papers in extensive detail, it is fair to conclude that the use of semantic segmentation in recent works has shown positive results, particularly in the field of autonomous vehicle driving. [4][5][7] and [24] provide useful information with respect to understanding the concepts behind semantic segmentation. [2][11] and [20] in particular give a deep insight into possible algorithms which can be utilised for the achieving the objectives of this project. To summarise, after conducting the literature survey, a better knowledge of the field was obtained.

The key takeaway from the literature review includes the possible semantic segmentation algorithms and performance metrics considered for observing the results of these algorithms. Algorithms such as Bird's Eye View and Optical Flow Estimation are mentioned. However, Convolutional Neural Networks are used more extensively and have proved to show better results.

It was observed from the literature surveyed that there is a lack of information with respect to parametric analysis of Semantic Segmentation algorithms on simulators such as Carla. Since most of the research makes use of simulators to obtain datasets, there is a need of better analysis for the algorithms developed.

1.3 Motivation

Self-driving cars, also known as autonomous vehicles, are currently one of the most promising upcoming technologies and a thriving study field. In the recent decade, self-driving car research and prototype development have accelerated, with a growing emphasis on data collection and processing technology. Currently, progress with respect to Computer Vision and Artificial Intelligence is one of the most important areas of research in autonomous vehicles. Also, the development of better and more efficient algorithms for lane detection is vital for autonomous vehicles. Increasing the accuracy of

these algorithms could greatly benefit the ongoing research of autonomous vehicles since it is essential to have a very low margin of error. In addition to this, contribution to the ADAS research community is another motivational factor for pursing this area of research.

All of these criteria are the motivation for the development of a viable semantic segmentation model for self-driving vehicles that can function in real-time and is accurate enough for the driverless car to rely on its results to comprehend the surroundings.

1.4 Problem Definition

To implement Lane Detection and Traffic Sign Detection for autonomous vehicles using CARLA simulator.

1.5 Objectives

The objectives of the project are as follows:

- To study the fundamental concepts, mathematical models behind semantic segmentation algorithms and its implementation for autonomous vehicles.
- To understand CARLA simulation platform to implement semantic segmentation algorithms.
- Design and implementation of lane detection and traffic sign detection algorithms.
- To deploy lane detection and traffic sign detection models on CARLA simulator for autonomous vehicles.
- Comparison, performance analysis and evaluation of lane and traffic sign detection algorithms to validate models.

1.6 Methodology

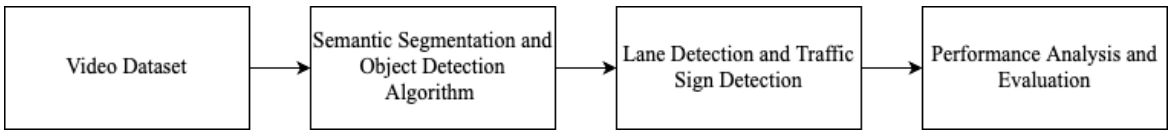

Fig.1.2 Block diagram of Methodology

The architecture is outlined in the Fig. 1.2. In the present study, the aim is to improve the accuracy of semantic segmentation and object detection of urban scenes by augmenting video datasets with synthetic images generated on the open-source driving simulator CARLA (Car Learning to Act).The acquired datasets are pre-processed and trained using semantic segmentation and object detection algorithms. The generated neural network will be then used for implementing the applications of Lane Detection and Traffic Sign Detection. Experiments are then performed and models are evaluated using various performance metrics. Further, the obtained results are analysed and recorded in detail.

Software Components

- CARLA 0.9.8
- Python 3.7
- Pygame 1.9.6
- OpenCV 4.2.0.34
- Numpy 1.18.3
- Tensorflow - 1.15.0
- Pillow 7.1.2
- Tesseract 4

Hardware Components

- Windows 11
- 64-bit Operating System, x64 based processor
- Hard drive - 2 TB

1.7 Organization of the Report

Chapter 1 discusses the overview of the project, including the introduction to Semantic Segmentation and Object detection along with the concepts of Deep learning and Neural Networks. Also, the summary of the literature survey, motivation, objectives and methodology of the project are explained.

Chapter 2 discusses the concepts behind the working of Carla platform. The idea of Neural Networks is explained in detail in this section along with its working.

Chapter 3 outlines the design and implementation of Lane Detection and Traffic Sign Detection on Carla simulator. It also shows the structure of the detailed design of the

project pipeline along with the constraints and assumptions. Various performance metrics are also defined in this section.

Chapter 4 demonstrates the results of the implementation of the project and the analysis of how it impacts the driving scenarios. The results of both Lane Detection and Traffic Sign Detection are compared with surveyed papers to evaluate the outcome of the project.

Chapter 5 talks about the conclusions that can be drawn from the results of the project. It also highlights the immense potential of the project and elaborates on the future scope in this field of application. The learning outcomes of the project are also discussed here.

Chapter 2

Theory and Fundamentals of Lane Detection and Traffic Sign Detection

This chapter covers the theory and fundamentals associated with the implementation of lane and traffic sign detection. It also explains in detail the architecture and algorithms used to achieve this in CARLA simulator. Here, SegNet and YOLO architectures and their working are explored.

2.1 Introduction

Computer vision and robotics research is primarily focused on fully autonomous vehicles, both in academic and commercial settings. In each scenario, the objective is to fully comprehend the environment surrounding the car by utilizing a variety of sensors and control modules. In order for autonomous vehicles to understand the road situation, lane detection is a paramount technology. In order for the car to position it correctly within the traffic lanes, camera-based lane recognition is a crucial first step toward such environmental perception.

A car can recognize traffic signs placed on the road, such as "speed limit," "children," or "turn ahead," using a technique called traffic sign detection and recognition (TSDR). This technology alerts the driver of any impending signs. With the help of this Advanced Driver Assistance Systems (ADAS) application, drivers will no longer face the problem of understanding what the sign says.

2.2 Car Learning To Act (CARLA) Simulator

CARLA is an open-source, Unreal Engine-based simulator that has flexible API's and is used for research on autonomous vehicles that enables users to alter and control certain simulation-related elements. It is a powerful simulator that includes tools for creating, honing, and validating systems in predetermined situations. The simulation is run on realistic cityscapes, allowing users to interact with the environment and make informed decisions. CARLA is constantly evolving since it serves a sizable community of users and scholars. Some of the diverse towns and weather conditions are shown in Fig. 2.1.

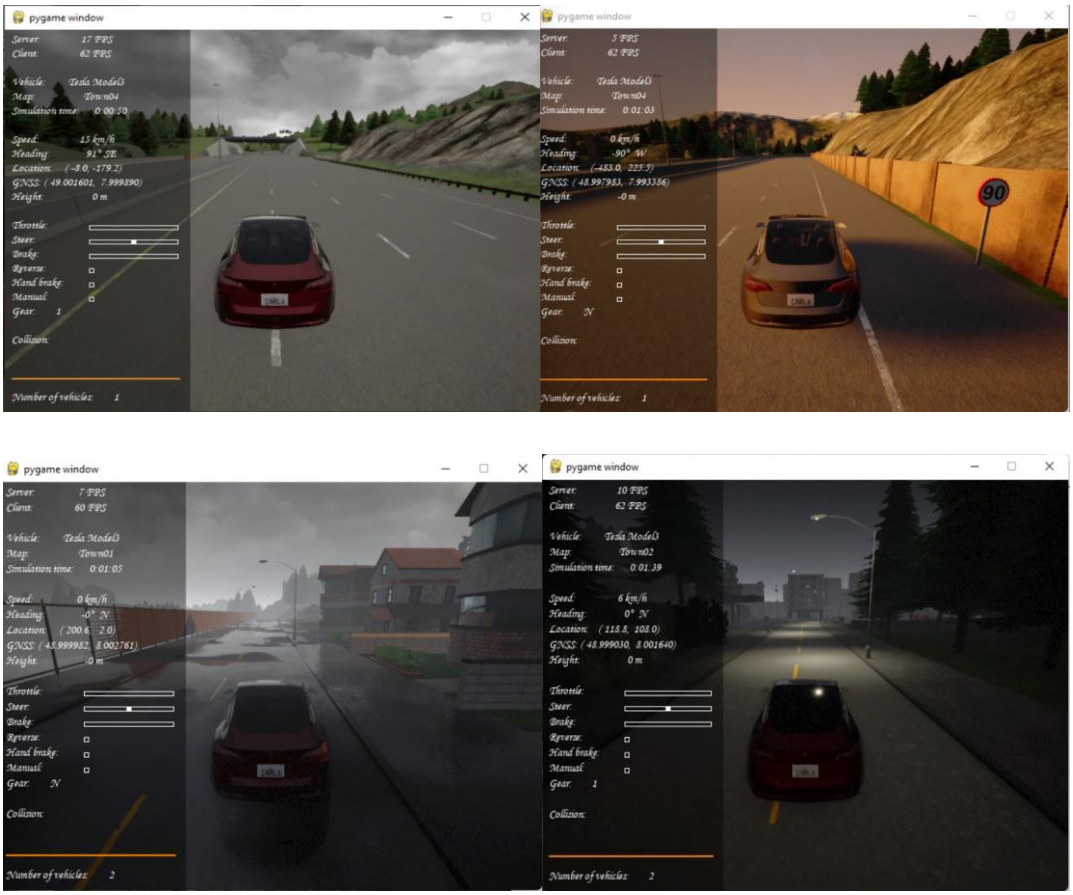

Fig.2.1: CARLA towns and streets with different weather presets

The client-server design of the CARLA simulator is scalable as shown in Fig.2.2. Everything pertaining to the simulation itself must be handled by the server, including rendering sensor data, doing physics calculations, updating information about the world-state and its actors, and much more. Running the server with a dedicated GPU would be ideal because it aspires for realistic results, especially when working with machine learning. The client side is made up of a collection of client modules that govern the world conditions and the logic of the actors in the scene. This is accomplished by making use of the CARLA API (written in Python or C++), a layer that serves as a mediator between the server and the client and is constantly developing to offer additional functions.

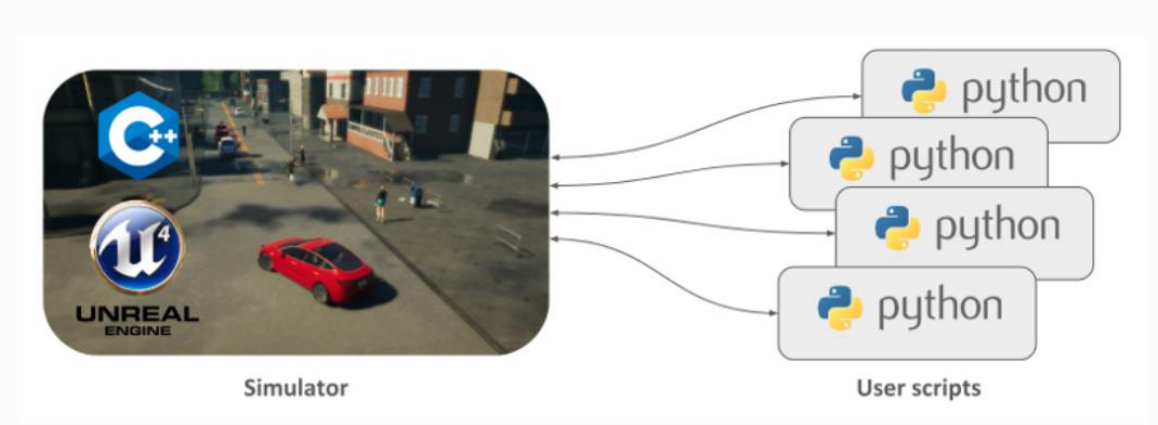

Fig.2.2: Client server architecture of CARLA

Recently released versions include 0.9.12 and 0.9.13 of CARLA. CARLA 0.9.7 was selected as the stable version for use in the implementation of lane detection and traffic sign detection.

The car and the sensors are both designated actors in CARLA, which has a large number of sensors including RGB cameras, Lidar, and some sensors that are only present in CARLA, like Semantic sensors. However, since we're working with Yolo on this project, we are using RGB cameras so that we can obtain snaps of each CARLA frame.

2.3 Convolutional Neural Network (CNN)

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, give importance (learnable weights and biases) to various aspects/objects in the image, and be able to distinguish one from the other. Comparatively, a ConvNet requires substantially less pre-processing than other classification techniques. A ConvNet's architecture takes its cues from the way the Visual Cortex is set up and is similar to the connectivity network of neuronal cells in the human brain. The Receptive Field, a constrained area of the visual field, is the sole area in which individual neurons are stimulated. The entire visual field is covered by a series of such fields that overlap. An RGB image that has been divided into its three colour planes—Red, Green, and Blue—can be seen in the Fig. 2.3. Images can be found in a variety of different colour spaces, including gray scale, RGB, HSV, CMYK and so on.

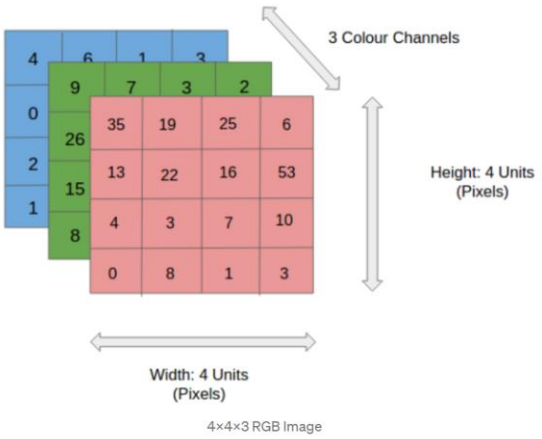

Fig.2.3: RGB Image [27]

The three main components of ConvNet designs are:

- Convolution Layers
- Pooling Layers
- Fully Connected Layers

Convolution is the mathematical process of combining two functions to create a third function. Kernel convolution is a fundamental component of numerous different Computer Vision methods in addition to CNNs. A small number matrix, referred as the kernel or filter, is used in the process to alter the image based on the values from the filter. The input image is represented by the letter f and our kernel by the letter h in the following formula, which is used to generate subsequent feature map values. The result matrix's indexes for the rows and columns are denoted by the symbols m and n, respectively. The equation (2.1) represents the convolution equation as follows:

$$\boldsymbol{G[m,n] = (f * h)[m,n] = \sum_{j}\sum_{k} h\,[j,k] f[m-j, n-k]} \tag{2.1}$$

Pooling's goal is to down-sample an input representation (such as an image or hidden-layer output matrix), which reduces its dimensions and enables assumptions to be made about the characteristics present in the resulting sub-regions. Fully connected layers (FCL) in a neural network are those layers where each activation unit in the layer above is connected to each input in the layer above it.

A typical rule for ConvNet architectures is to apply Convolutional Layers to the input in

succession, periodically down sample the spatial dimensions, and then use Pooling Layers to reduce the amount of feature mappings. Feature Maps: The result of one filter applied to the preceding layer is the feature map. The output of each layer, then, is the feature map.

The ConvNet's role is to condense the images into a format that is simpler to analyze without sacrificing elements that are essential for obtaining an accurate forecast. A feed-forward neural network with up to 20 or 30 layers is known as a convolutional neural network. The convolutional layer is a unique kind of layer that gives convolutional neural networks its power.

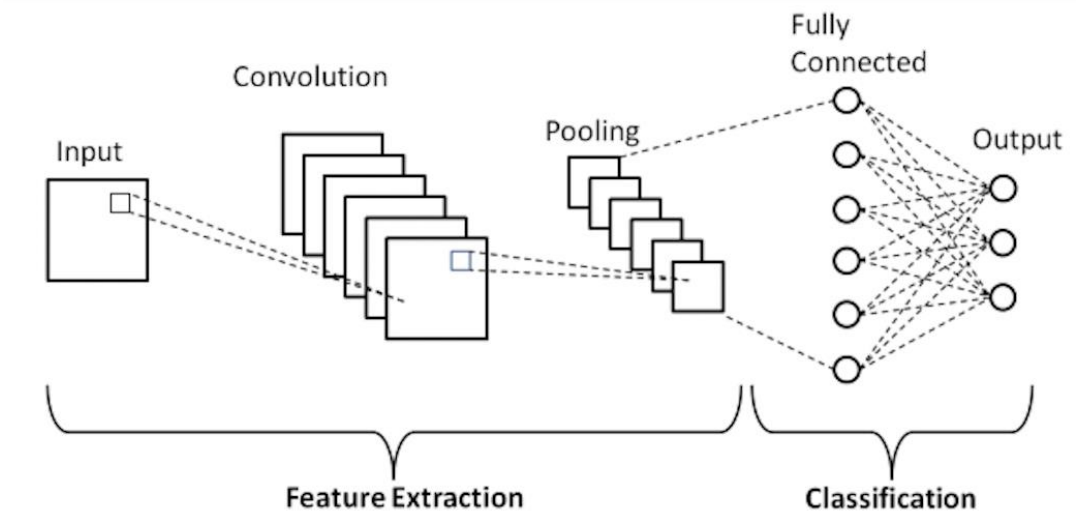

Fig.2.4: CNN architecture [28]

A Convolutional Neural Network's architecture Fig.2.4 is a multi-layered feed-forward neural network created by sequentially stacking numerous hidden layers on top of one another. Convolutional Neural Networks can learn hierarchical features because of their sequential design. Convolutional layers are frequently followed by activation layers, some of which are then followed by pooling layers, as the hidden layers.

2.4 SegNet

SegNet is a semantic segmentation model. An encoder network, a related decoder network, and a pixel-wise classification layer make up the core trainable segmentation engine.

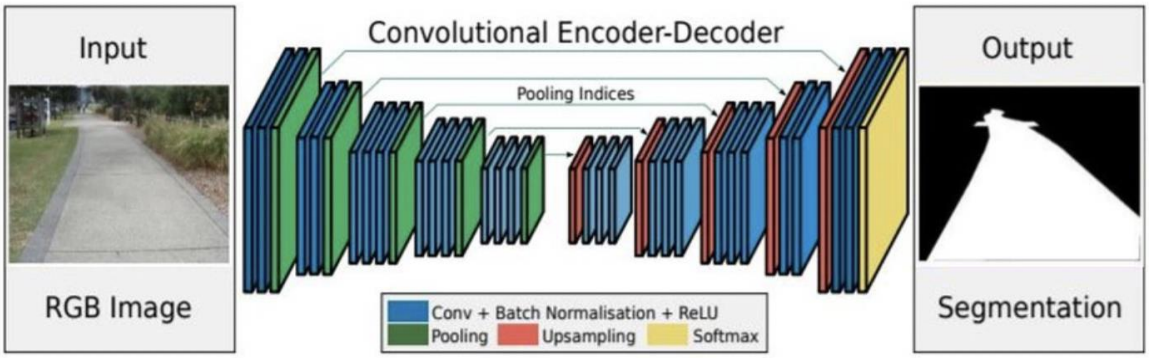

Fig.2.5 SegNet Architecture [29]

The encoder network consists of 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG16 network [1] designed for object classification. The SegNet architecture is represented in Fig 2.5. The decoder network contains 13 layers since there is a corresponding decoder layer for each encoder layer. A multi-class soft-max classifier is fed the final decoder output to generate class probabilities for each pixel separately. For the purpose of creating a collection of feature maps, each encoder in the encoder network convolutions with a filter bank [1]. After that, these are batch normalized. Next, a max (0, x) element-wise rectified linear nonlinearity (ReLU) is used. The output is then sub-sampled by a factor of two once max-pooling is completed with a 2 x 2 windows and stride 2 (non-overlapping window). To provide translation invariance over minute spatial alterations in the input image, max-pooling is used. Each pixel in the feature map has a large input image context (spatial window) as a result of subsampling. Although using many layers of max-pooling and subsampling might increase translation invariance for robust classification, this results in a loss of spatial resolution for the feature maps.

For segmentation, where boundary delineation is essential, the increasingly lossy (border detail) image representation is not beneficial. Therefore, prior to performing sub-sampling, boundary information must be recorded and stored in the encoder feature maps. After subsampling, all of the encoder feature maps can be kept if memory during inference is not limited. Here, we are just storing the maximum pooling indices, which means that for each encoder feature map, we have memorized the positions of the maximum feature values in each pooling window. This is theoretically possible with 2 bits for each 2 × 2 pooling window, making it

considerably more efficient to store than learning feature map(s) with float precision.

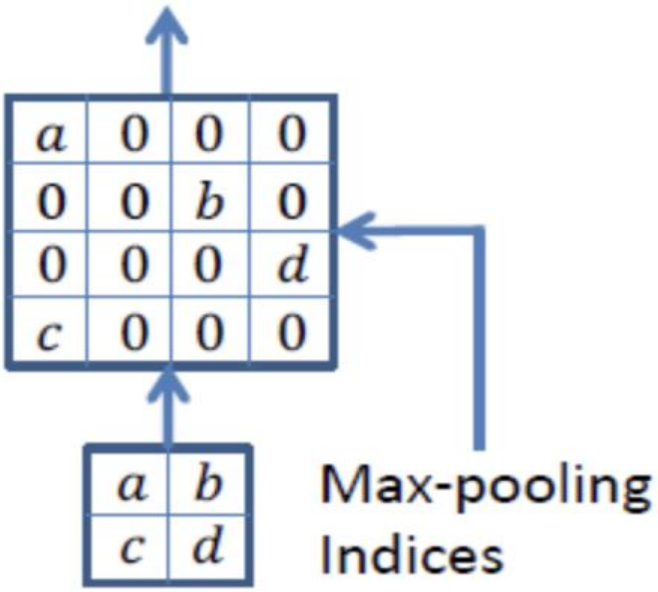

Fig.2.6 Decoder network [30]

The appropriate decoder in the decoder network up samples the input feature map(s) using the max-pooling indices from the matching encoder feature map(s). Fig. 2.6 illustrates this SegNet decoding method in action. Dense feature maps are then created by convolving these feature maps with a trainable decoder filter bank. Each of these maps is subsequently subjected to a batch normalizing procedure. It should be noted that despite having a 3-channel encoder input, the decoder corresponding to the first encoder (which is closest to the input image) outputs a multi-channel feature map (RGB).

Contrary to other network decoders, this one creates feature maps with the same number of channels and sizes as the encoder inputs. A trainable soft-max classifier receives the high dimensional feature representation at the output of the final decoder. Each pixel is classified separately by this soft-max. The soft-max classifier produces a K channel probability picture, where K is the number of classes. The class with the highest probability at each pixel corresponds to the segmentation that is expected. Scene understanding applications were the main driving force behind SegNet. As a result, it is designed to be effective during inference both in terms of memory usage and calculation time. Additionally, compared to other competing architectures, it has a substantially lower number of trainable parameters and can be taught from beginning to end using stochastic

gradient descent.

2.5 YOLO (You Only Look Once)

YOLO is a state-of-the-art object detection algorithm. Yolo is a unified structure that frames detection as a regression problem and obtains a pipeline that is not overly complex, allowing the system to be very quick: it processes streaming video in real-time with less than 25 milliseconds of latency while processing images in real-time at a rate of 45 frames per second. It makes more localization errors than other cutting-edge detection systems, but it is less prone to predict false positives on background. When generalizing from natural images to other domains like artwork or unexpected inputs, it outperforms other detection algorithms because it learns very broad representations of objects. The architecture of YOLO model is shown below in fig 2.7.

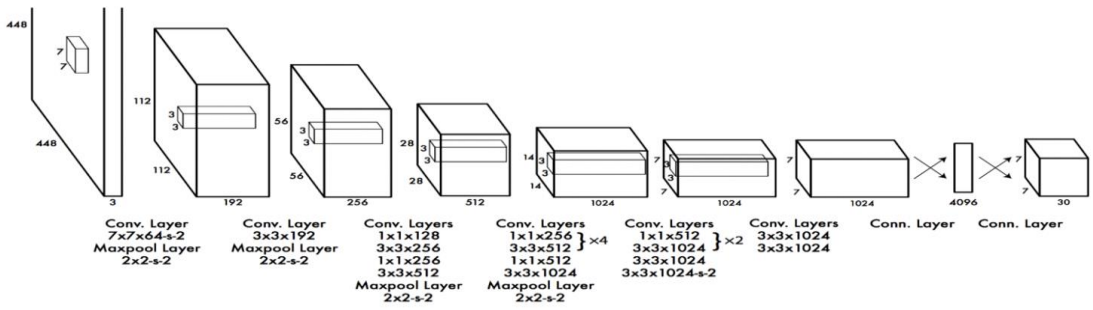

Fig.2.7 YOLO Architecture [31]

In-depth analysis of the Yolo network's architecture reveals that it was modelled after the GoogLeNet model for image classification; however it includes 24 layers instead of 22 and 2 completely linked layers. 1 x 1 convolutional layers are switched between layers to condense the features space from previous layers. During the training and testing phases Yolo views the whole image wherein it will implicitly encode the contextual information about their classes as well as their appearances and avoids doing some object detection errors in the background of the images. This is in contrast to some top detection methods, like R-CNN, which use a sliding window over several locations. A single convolutional neural network (CNN) predicts simultaneously B numerous bounding boxes for each grid cell, their confidence value, and their classes probability for each region to carry out the detection method. The system divides the input image into an S x S grid. When objects near the borders of several cells are localized by these near-cells, the B bounding boxes

are weighted by the expected probability and the non-maximal suppression technique is used to fix multiple detections. The YOLO machine learning algorithm has three versions, with the third version being a more accurate version of the original ML algorithm. Versions 1-3 of YOLO were developed by Joseph Redmon and Ali Farhadi. Initially, the YOLOv3 algorithm divides an image into a grid. Each grid cell foretells the presence of a specific number of boundary boxes (also known as anchor boxes) around items that perform well in the aforementioned predetermined classes.

Only one object is detected by each boundary box, which has a corresponding confidence score indicating how correct it expects that prediction to be. The ground truth boxes' dimensions from the original dataset are clustered to identify the most typical sizes and shapes before being used to create the border boxes.

2.6 Optical Character Recognition

Optical Character Recognition is a software technology that electronically recognizes text (hand written or printed) within an image file or physical document, such as a scanned document, and transforms it into a machine-readable text form for use in data processing. It is also termed as text recognition.

Simply put, optical character recognition software facilitates the transformation of physical or digital materials into searchable formats. Text extraction tools, PDF to.txt converters, and Google's picture search feature are a few examples of OCR. To accomplish OCR as a procedure as accurately as feasible, it typically consists of multiple sub-processes. The subsequent steps are:

- Preprocessing of the Image
- Text Localization
- Character Segmentation
- Character Recognition
- Post Processing

A. Scanning the Image

The primary OCR stage involves connecting to a scanner and scanning the page. Since scanning the document/image standardizes the inputs, it reduces the number of factors

that need to be taken into consideration while developing OCR software. Additionally, by guaranteeing the precise alignment and dimension of the particular document, this phase specifically improves the efficiency of the overall process.

B. Refining the Image

The components of the document that must be captured are improved by the optical character recognition software in this step. To produce a text that is simple and easy to read, any flaws like dust specks are removed, and borders and pixels are smoothed. With the help of this phase, the programmer can more easily "see" the supplied words without obstructions like smudges or irregular dark patches.

C. Binarization

The next step is to turn the refined picture document into a bi-level document image with only black and white colors, with dark or black areas being recognized as characters. White or light areas are recognized as background at the same time. This step applies segmentation to the document in order to clearly distinguish the foreground text from the background and to enable the best possible character recognition.

D. Recognizing the Characters

The black spots are further processed in this stage to reveal letters or numerals. An OCR typically concentrates on a single character or block of text at a time. One of the two algorithms listed below is used for character recognition:

- *Pattern recognition:* OCR software is used to input text in various fonts and formats as part of the pattern recognition method. The characters in the scanned document are then compared to and recognized using the customized software.
- *Feature detection:* OCR software employs rules taking into account the characteristics of a specific letter or number to recognize characters in the scanned document using the feature detection method. The quantity of angled lines, crossed lines, or curves used to compare and distinguish characters are a few examples of features. Simple OCR software finds the closest match by comparing each scanned letter's pixels to those in an existing database. To compare and match physical characteristics with corresponding letters, more advanced OCR

techniques break down each character into its component parts, such as curves and corners.

E. Checking for Accuracy

After successful character recognition, the OCR software's internal dictionaries are used to cross-reference the findings to confirm correctness. The result of an OCR analysis is compared to the content of the original version to determine the correctness of the OCR. There are two typical methods for analyzing the accuracy of OCR software:

- Character-level accuracy, measured by the number of successfully recognized characters.
- Word-level accuracy, measured by the percentage of properly identified words.

OCR has many benefits, but in particular:

- It makes office work more productive and efficient.
- Instantaneous content search is quite helpful, especially in an office scenario where there is a lot of scanning or a lot of document input.
- OCR is rapid, preserving the document's content while also saving time.
- Employee productivity increases as a result of reduced time spent on manual labor and their ability to complete tasks more quickly and effectively.

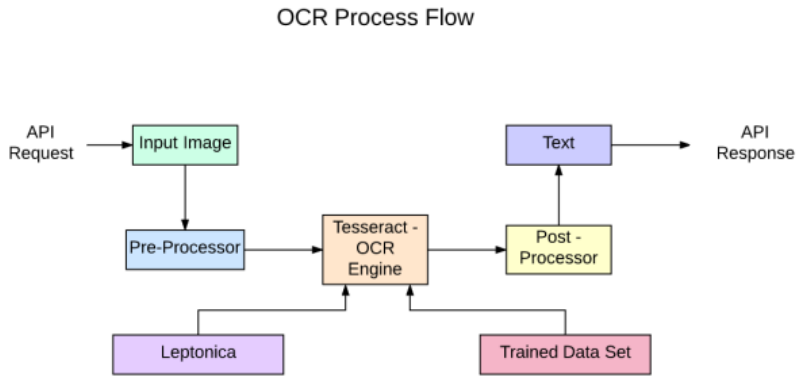

Fig.2.8: Work Flow of OCR [32]

The OCR process flow is described in Fig 2.8. A picture is simply a matrix of RGB

channels made up of numbers in a computer, and the goal is to convert this matrix into data that can be used by CARLA. OCR functions as a pattern identifier, thus it will take the image, pre-process it (rotation, correcting angles, using filters), search through each image for letter patterns, match the ones with the best accuracy, and save the results in a document that can be edited.

Tesseract is an open source text recognition (OCR) engine that can be used to extract printed text from photos either directly or (for programmers) via an API. Many different languages are supported by it. Consider a scenario where the automobile is unable to connect to the internet. In this case, Tesseract is used to construct the solution inside the car, which minimizes connectivity issues and response times. For the implementation of our project Tesseract 4 was used as its accuracy was high compared to its lower versions.

Summary

To summarize, a better understanding of CARLA simulator where the semantic segmentation and object detection models are deployed is acquired. More knowledge with regard to the inner working of models being utilized is obtained. The steps that need to be followed in order to deploy these for lane and traffic sign detection are explained.

Chapter 3

Design and Implementation of Lane Detection and Traffic Sign Detection

In this chapter, the block diagrams and flowcharts associated with the working models is provided and explained. Specifications of the two datasets created are also mentioned along with the performance metrics being considered for the analysis of results. Since the project is being deployed in a virtual environment, there are certain assumptions and constraints that need to considered. These factors are also clearly noted in this chapter.

3.1 Block Diagram of Lane Detection

The implementation of lane detection is inspected in the Carla environment using Carla simulator. Fig 3.1 shows the pipeline of the project. This system is trained to detect the lane for the vehicles to advance on the road.

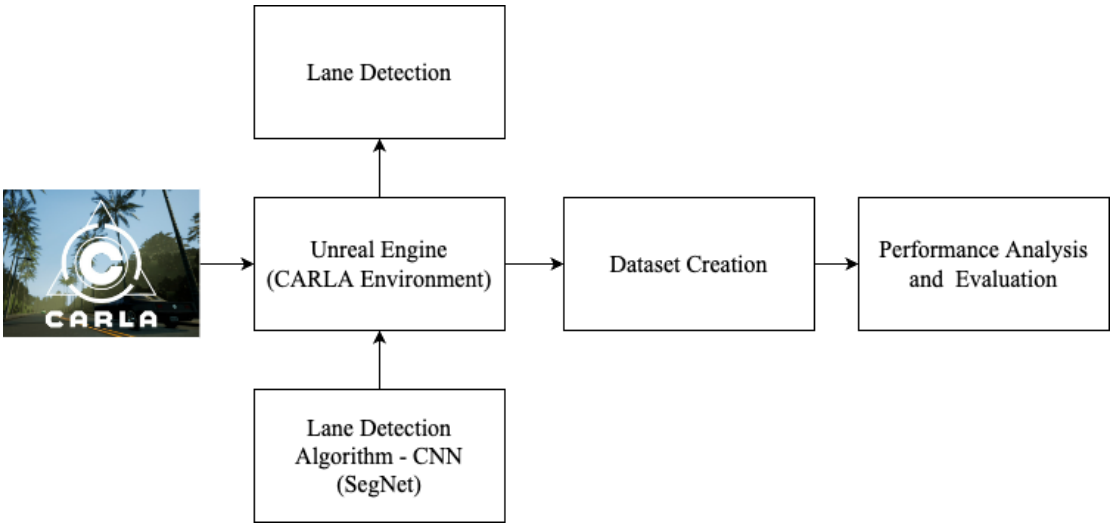

Fig.3.1: Block diagram of Lane Detection

The Carla environment is spawned with various actors and pedestrians. The Carla simulator source code is then modified and trained using a variant of CNN called SEGNET. This lane detection algorithm performs semantic segmentation on the Carla video to determine the lanes. Once the lane is determined it is divided into three planes: center plane, left plane and right plane. These planes enable the vehicle to move in the required path and stay in the lane precisely.

As a next step the dataset of total 612 images are created and detailed analysis is performed to obtain the results for evaluation.

3.2 Block Diagram of Traffic Sign Detection

The system is implemented in the Unreal engine (Carla environment). Fig 3.2 shows the pipeline of the project and the main agents that take part in it: the Speed Traffic Sign Detection Module (Yolo), RGB sensor, Tesseract OCR and the Carla Simulator. Open source optical character recognition software called Tesseract makes it easier to extract text from photos.

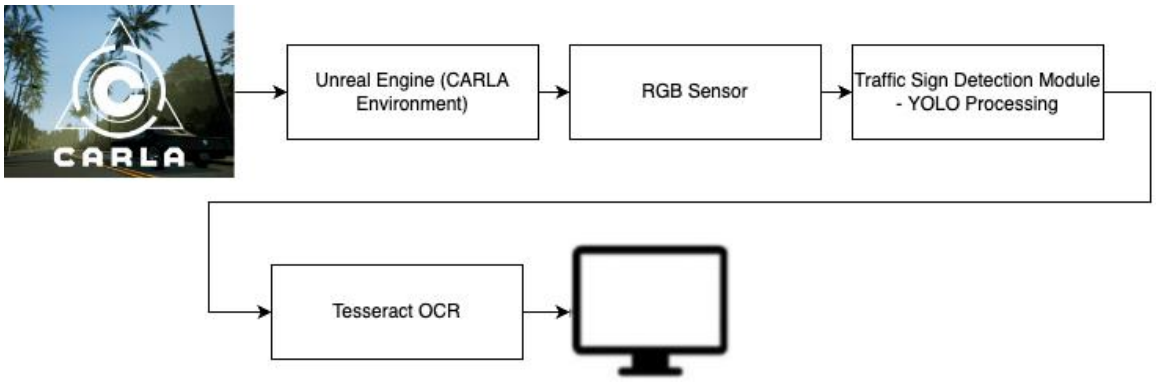

Fig.3.2. Block diagram of Traffic Sign Detection

The system involves two main steps: detection and recognition. In the detection phase the structure of the scene in Carla environment is explored using RGB camera mounted on the dashboard of the car. The YOLO detection module determines the size and location of the frame and examines the regions in the scene to detect the road signs. Once the scene's vanishing point (VP) and consequently the ground plane are identified, these regions are defined. Then, only within these scene search regions are candidate locations for road signs determined by integrating MSERs with threshold, saturation, and hue colour values (HSV). Based on the movement of these regions in connection to the camera and the scene structure, time information is combined with these regions over subsequent frames to further reduce the detected zones.

Once the potential traffic signal has been identified by YOLO, the algorithm then moves on to try to recognize the text in the area. To vertically align the text's characters, the region is first given a rough perspective transformation. The optical shelf character recognition (OCR) tool is then used to detect potential text character components in the

area and classify them into potential lines of text. OCR data from numerous frames are integrated by combining individual words between frames and utilizing a weighted histogram results in order to increase recognition accuracy. The recognized text is then displayed on the monitor indicating the speed sign as to whether it is speed sign 30, 60 90 or stop sign.

Subsequently, the dataset comprising of 256 images were created to analyze the various performance parameters and perform evaluation.

3.3 Flowchart of Lane Detection

The Carla Simulator source code is modified to detect lane in Carla environment. Fig.3.3 shows the working of lane detection algorithm in Carla.

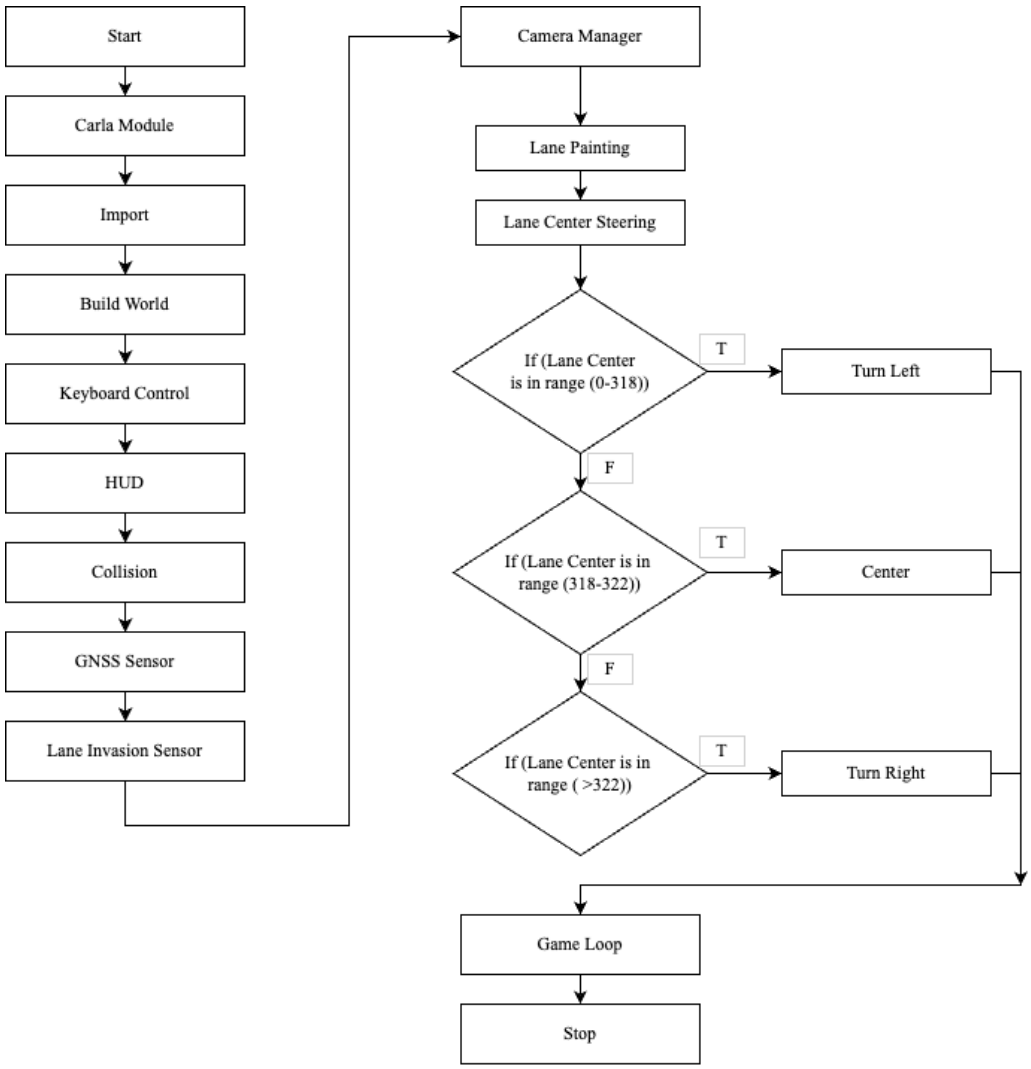

Fig.3.3: Flowchart for Lane Detection

Initially the Carla simulator is setup to begin. The Carla module is then imported and all other various modules required are also imported. Then the virtual world is built and various manual controls to operate the vehicle are provided using the keyboard. Further numerous sensors such as collision sensor, global navigation satellite system(GNSS) sensor, lane invasion sensor, camera manager and many others sensors are defined.

The heads-up-display (HUD) is a method which displays the vital information regarding the collision history, speed, brake, location and other parameters with respect to the vehicle. Then the lane painting takes place indicating as left lane (turn left) if its lane center is in range (0-318) or right lane (turn right) if its range is in (318-322) or center if it is (greater than 322).Thus the vehicle can move ahead in accordance to the predicted lane. In this way the game loop occurs and finally it can be terminated.

3.4 Flowchart of Traffic Sign Detection

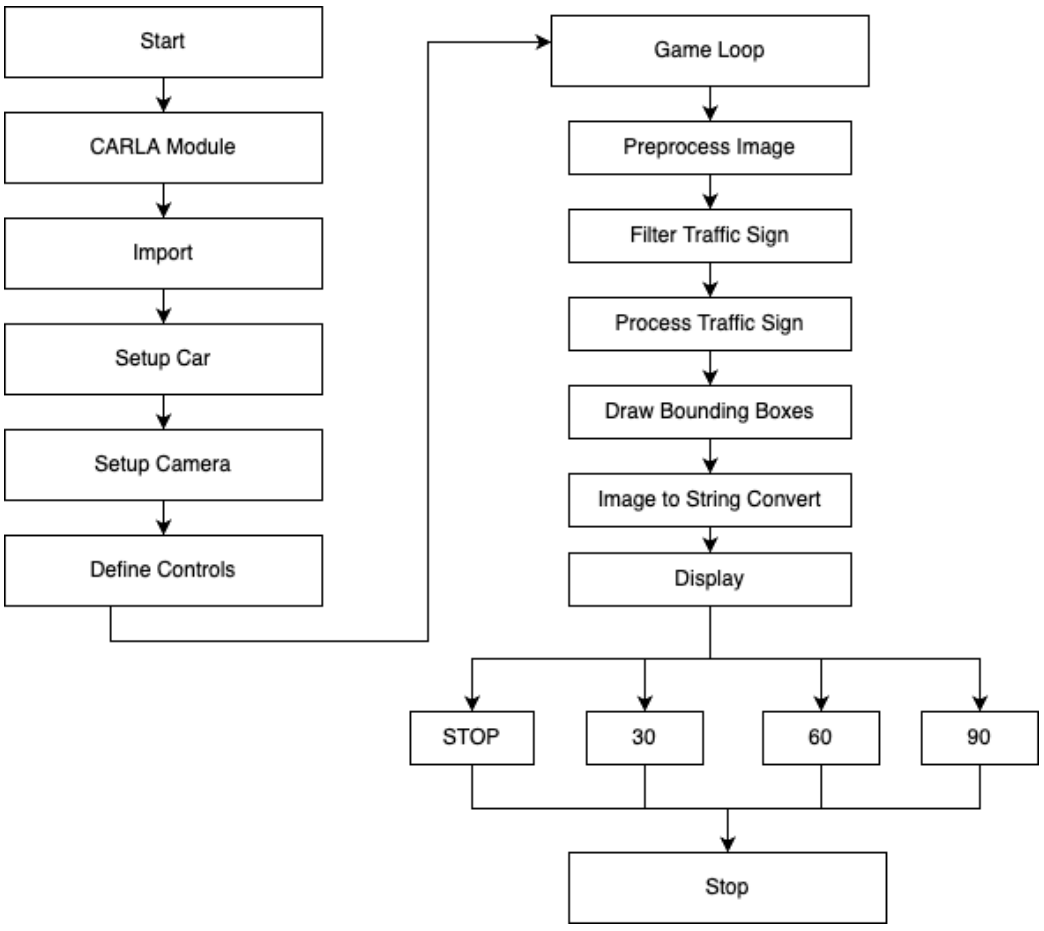

Fig.3.4. Flowchart of Traffic Sign Detection

In the first instance the source code of Carla simulator is reformed to detect speed signs and recognize them. Fig 3.4 represents the working flow of the system.

At first, the Carla environment is built by importing Carla module and all other required modules are also imported. The vehicle is then defined with manual controls to operate in the world with the help of keyboard. An RGB camera is mounted on the dashboard of the car to capture the frames. The captured images are then preprocessed by the YOLO object detection module that filters out and tries to determine all the signs located in the frames. Once the sign is detected a bounding box is drawn to indicate it as the sign detected by the module. The images are then converted to string and passed through Tesseract OCR to read the text from the detected box of the sign. Further the recognized text is then displayed on the screen as either 30, 60, 90 or stop depending on the sign detected.

3.5 Dataset Specifications for Lane Detection

The model was provided with video dataset that is of Carla simulator to run. In order to perform analysis of the model a dataset containing total of 612 images of 1920 x 1080 pixels were created and extracted at different instances. These images were manually captured in various towns ranging from town 1 to town 5, multiple weather conditions, by spawning a number of vehicles and pedestrians, different traffic conditions with some of the images containing heavy traffic flow and others containing vacant streets. Some images were captured through manual control of the Carla vehicle and others were captured through the autopilot mode available in Carla simulator. All the images were captured during multiple sessions.

3.6 Dataset Specifications for Traffic Sign Detection

The model was executed in towns containing abundant traffic signs. Next to perform analysis of the model a dataset containing total of 256 images 1920 x 1080 pixels were created and extracted from the Carla simulator while driving in varied weather conditions, towns, with pedestrians and other vehicles and with vacant streets. It was observed that towns 1, 3 and 5 consisted of more number of signs and hence the images for the dataset were extracted from the same. Under different weather presets such as rainfall or fog, it was found that the signs did not appear clearly and hence were taken more under normal daylight as well as night time conditions. Due to non-visibility or bad light some of them had to be filtered out and the final set consisted of 256 images. Four classes of traffic

signs were identified for the purpose of object detection. The classes are:

- Speed Limit 30 km/hr traffic sign
- Speed Limit 60 km/hr traffic sign
- Speed Limit 90 km/hr traffic sign
- Stop traffic sign

These images were further compared and analyzed to determine the model's performance.

3.7 Performance Metrics

The effectiveness of Machine Learning algorithms, classification algorithms, and regression algorithms can be assessed using a variety of metrics. The datasets obtained from both the models of lane detection and traffic detection respectively has been analyzed using the performance metrics below. The simplest approach to gauge how well a classification problem is performing when the output can include two or more different types of classes is using a confusion matrix. A confusion matrix consists of a table having two dimensions, Actual and Predicted, along with True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for each dimension, as shown below in Table. 3.1

Table 3.1: Confusion Matrix

Confusion Matrix		Actual Class	
		1	0
Predicted Class	1	True Positives (TP)	False Positives (FP)
	0	False Negatives (FN)	True Negatives (TN)

.

- True Positives (TP) occur when a data point's actual class and projected class are both 1.
- True Negatives (TN) are when a data point's actual class and anticipated class are both 0.
- False Positives (FP) occur when a data point's real class is 0 and its predicted class is 1, which is known as a false positive.
- False Negatives (FN) occur when a data point's real class is 1 and its predicted class is 0.

A confusion matrix was obtained using the parameters mentioned in Table. 3.2.

Table 3.2: List of performance metrics

Performance Metric	Abbreviations	Formulae
Accuracy Rate	ACC	(TP+TN)/(N+P)
Sensitivity - True Positive Rate	TPR	TP/(TP+FN)
Specificity - True Negative Rate	TNR	TN/(TN+FP))
False Positive Rate	FPR	FP/(FP+TN)
False Negative Rate	FNR	FN/(FN+TP)
Positive Pred Value - Precision	PPV	TP/(TP+FP)
F1 Score	F	2*((PPV*TPR)/(PPV+TPR))
Error Rate	ERR	(FP+FN)/(N+P)

- Accuracy is given by total number of correct classifications (either True Positive or True Negative) divided by total number of classifications done.
    - Accuracy = (TP + TN)/ (TP +FP +FN+TN)
- Sensitivity: The sensitivity of a model measures the proportion of TP examples or positive cases which were correctly classified. It is measured as
    - Sensitivity = TP/ (TP+FN)
- If Intersection of Union (IoU) ≥0.5, classify the object detection as True Positive (TP)
- If IoU < 0.5, then it is a wrong detection and classify it as False Positive (FP)
- When a ground truth is present in the image and model failed to detect the object, classify it as False Negative (FN).

- True Negative (TN): TN is every part of the image where we did not predict an object.
- Specificity: Specificity of a model measures the proportion of negative examples which have been correctly classified.
    - Specificity = TN/ (TN+FP)
- Precision and Recall: Precision gives the proportion of positive predictions which are truly positive.
- Precision indicates the reliability of a model in predicting a class of interest
    - Precision = TP/ (TP+FP)
- Recall indicates the proportion of correct prediction of positives to the total number of positives.
- Recall gives the proportion of TP cases over all actually positive cases
    - Recall = TP/ (TP + FN)
- F-score is measure of model performance which combines the precision and recall. It takes the harmonic mean of precision and recall
    - F-score = 2*(precision*recall)/ (precision + recall)
- Error Rate: The percentage of misclassifications is indicated using error rate which is measured as
    - Error rate = (FP + FN)/ (TP+TN+FP+FN)

3.8 Constraints and Assumptions

The lane detection and traffic sign detection models were performed under certain predefined constraints and assumptions in the CARLA environment.

For Lane Detection, the weather conditions considered were - Cloudy Noon, Cloudy Sunset, Hard Rain Noon, Mid Rain Sunset, Wet Cloudy Sunset. These weathers were configured in 4 different Urban Towns namely - Town 01, Town 02, Town 03, Town 04. The vehicle used to navigate the environment is Tesla Model 3. For the purpose detection, various sensors – Collison, Lane Invasion, GNSS and Camera sensors – were utilised.

In CARLA, actors refer to the pedestrians and vehicles present on the streets. For the application of lane detection, about 40 to 80 pedestrians and 60-100 vehicles are spawned. It was also carried out in empty streets with no pedestrians.

For Traffic Sign Detection, the classes considered are STOP, Speed limit (30km/hr), Speed limit (60km/hr), Speed limit (90km/hr). Here, only 3 towns are considered as they have more number of traffic signs namely - Town 01, Town 03, and Town 05. The weather conditions here ranges from Sunny Noon to Heavy Rainfall Sunset.

Summary

Separate block diagrams are created to represent the concepts and systems in a higher level with a less detailed overview for each application. The flow of control is clearly displayed through flowcharts for lane and traffic sign detection. Two different datasets were created for the purpose of testing the semantic segmentation and object detection model containing 612 and 256 images respectively. The assumptions and constraints while testing the models in CARLA environment are mentioned. Finally, the performance metrics used for evaluation of the trained models are explained along with their formulas.

Chapter 4

Simulation Results and Analysis

In this section, the Lane and Traffic Sign Detection application results were obtained and evaluated using the performance metrics stated in the previous chapter. All the cases for each application is documented and presented with appropriate figures and explanations. A qualitative and quantitative analysis is performed to validate the models being implemented.

4.1 Lane Detection Results

4.1.1 Case 1: Turn Left

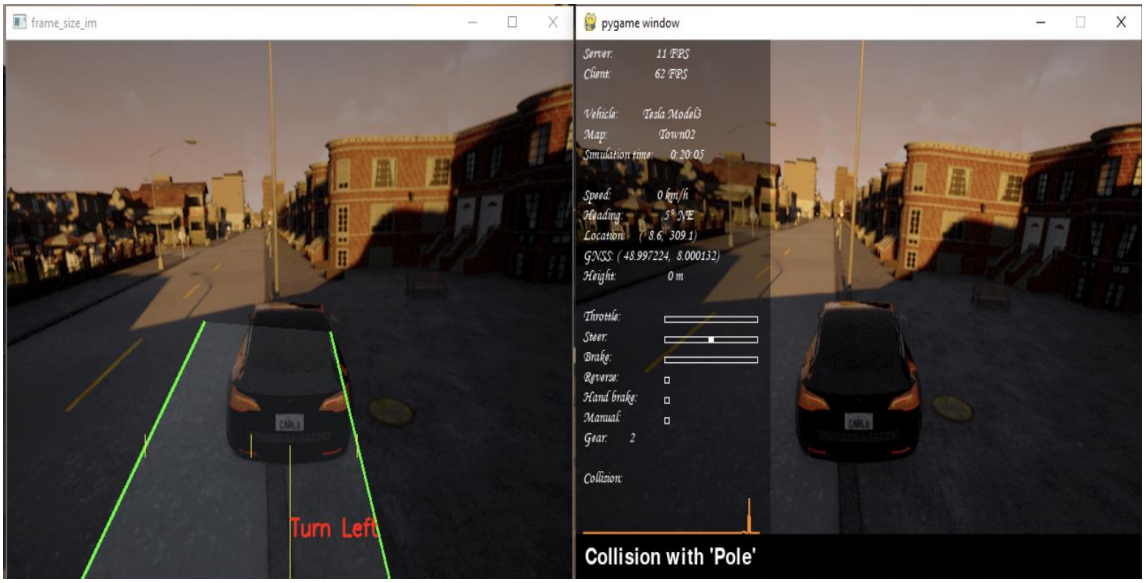

Fig 4.1: Lane detection for turn Left

Initially, when the car tends to deviate from its appropriate path and traverses along the footpath which may result in a collision with a pole, the lane detection algorithm deployed shows a sliding window which pulls each lane from the binary segmentation map after it has been processed to split the lanes, creating the lane instance segmentation image Fig.4.1. The lanes are binary segmented using an encoder-decoder deep learning architecture. Hence the detected lane is shown on the other screen with an indication to turn left for the car in order to avoid collision with the pole. This result not only shows the lane of the CARLA Simulator detected but also a warning with a statement about

what to do next for collision avoidance. On the dataset captured, this strategy was validated and produced results that were competitive. The results were compared for several scenarios in which the automobile was deliberately driven into barriers to see how accurately the lane detection algorithm would identify and suggest a left turn.

4.1.2 Case 2: Turn right

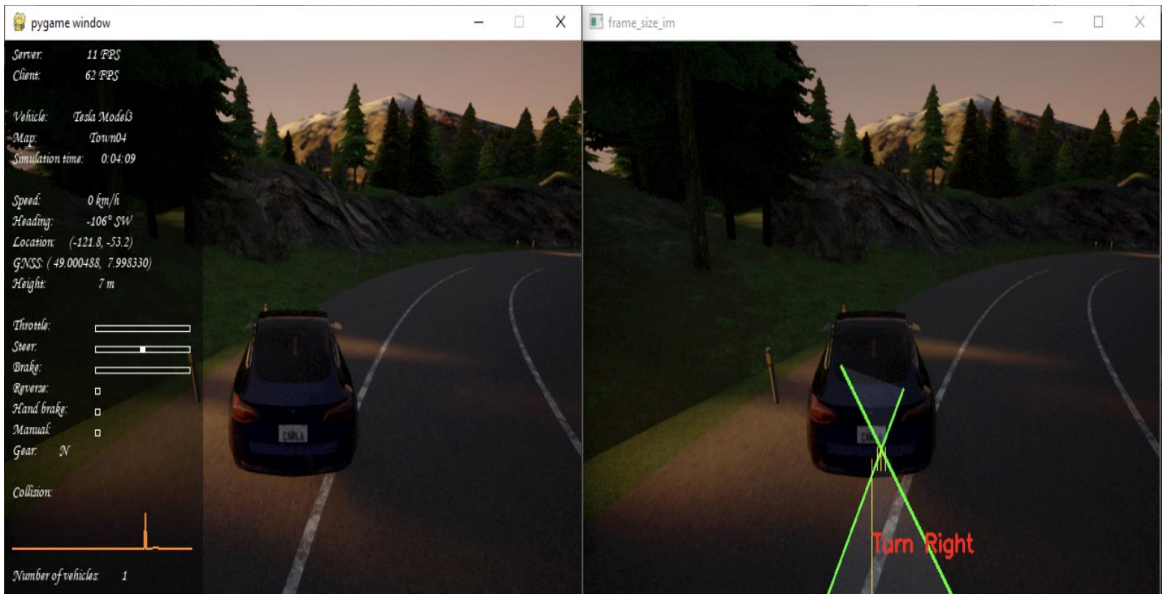

Fig 4.2: Lane detection for turn Right

Similar to what was observed in case 1, sliding window pulls each lane from the binary segmentation map after it has been processed to split the lanes, creating the lane instance segmentation image when the car starts straying from its proper path and travels along the side of the road. Utilizing an encoder-decoder deep learning architecture, the lanes are binary segmented. As a result, the detected lane is displayed on the opposite screen along with a warning for the automobile to turn right Fig.4.2 in order to move along the curve of the road. This was also verified on the obtained dataset. To determine how precisely the lane detection algorithm would recognize and advise a right turn, the results were compared for a number of scenarios in which the car was purposefully pushed into barriers

4.1.3 Case 3: Center

Fig 4.3: Lane detection to stay in the Center

In this case, Center is displayed as indication that the vehicle is moving properly within the borders of the lane and does not need to make any adjustments. As seen in Fig.4.3, the car on the left is moving perfectly inside the lane. Hence, the semantically segmented image on the right shows 'Center'. Basically, if the lane detection algorithm is displaying 'Center', the automobile is being driven accurately along the road. This was also tested in various conditions and the results of the dataset obtained were utilized for the performance analysis of the model.

4.2 Performance Analysis of Lane Detection

Confusion Matrix is a representation of the differences between actual and predicted values. It has a table-like shape and measures how well our machine learning classification model performs. It displays the various contrasts between actual and predicted values. True Positives (TP) are values that both turned out to be positive and were expected to be positive. False Positives, or FPs, are values that were genuinely anticipated to be negative but turned out to be positive. known also as a Type I error. False Negatives, or FNs, are values that were projected as negative but were actually positive. Error of the same type as Type II. True Negative values are those that were both genuinely negative and projected to be negative.

4.2.1 Confusion Matrix for Lane Detection

Table 4.1: Confusion Matrix for Lane Detection

Confusion Matrix		**Actual Class**		
		Class LEFT	**Class CENTER**	**Class RIGHT**
Predicted Class	**Class LEFT**	167	14	7
	Class CENTER	2	41	6
	Class RIGHT	5	6	383

Here three classes have been considered: Class Left, Class Center and Class Right. The actual class represents the obtained results and the predicted class represents the results that were supposed to occur or predicted. As observed from Table 4.1 the diagonal elements are all True Positive indicating absolute true results. 7 indicates Right and incorrectly classified, 1 indicates Center and incorrectly classified and 2 indicates Left and incorrectly classified. The row elements indicate False Positive and the column elements indicate False Negative. In the similar way the other classes have been designed, implemented and compared with each other to obtain the Confusion Matrix for an efficient performance analysis.

4.2.2 Performance analysis of Lane Detection

Table 4.2: Performance analysis of Lane Detection using results from Confusion Matrix

Parameter/Class	Class LEFT	Class CENTER	Class RIGHT
Accuracy	95.63%	94.07%	94.7%
Error Rate	4.47%	5.93%	5.3%
Precision	89%	84%	95%
Recall	96%	58%	97%
F1 Score	92%	68%	96%
True Positive Rate (Sensitivity)	88.83%	83.67%	97.21%
False Positive Rate	11.17%	16.33%	2.79%
True Negative Rate (Specificity)	96.45%	67.21%	96.72%
False Negative Rate	3.55%	31.79%	3.28%

The performance of the algorithms in the CARLA Environment is analyzed using the performance Table 4.2 where Accuracy, Error rate, Precision, Recall, F1 Score, True Positive Rate(Sensitivity), False Positive Rate, True Negative rate, False negative rate have been calculated for the classes: Left, Right and Center. As observed, an accuracy of 95.63% has been obtained in Class Left indicating better lane detection in this particular class where the model is better reinforced in indicating and detecting the left lane. The maximum Recall, F1 Score and Precision has been obtained in class Right. The false positive rate is least observed in Class Right and least False Negative Rate in class right. An overall accuracy of 92.58% has been obtained and an average precision of 92.25% is observed.

4.3 Lane Detection Performance Evaluation

The semantic segmentation model developed for lane detection using SegNet gives a precision of 92.25%. This is an improvement when compared to existing models such as Sparse Convolutional Neural Networks (SCNN) which has a precision of 76.13% and PointLaneNet which has a precision of 86.33% [33]. There is considerable improvement of about 20.31% and 6.86% in other parameters such as F1 score as well. Recall shows an improvement of 26.93% and 10.76% respectively.

4.4 Traffic sign Detection Results

4.4.1 Detection of speed limit 90 km/hr traffic sign

Fig 4.4: Traffic sign detection for speed limit of 90km/hr.

The YOLO model employed is able to recognize the 90 km/hour traffic sign as seen in Fig.4.4, which is a significant advantage for autonomous driving vehicles since the speed detected on the traffic signs will allow the car to either slow down or speed up depending on the number and ensure a safe driving. Based on a regression system that gives classes and bounding boxes for photos of traffic lights, traffic signs, cars, people, and objects at speed during the execution of the algorithm, the Yolo model is capable of detecting items in real time. The CARLA vehicle was equipped with an RGB camera sensor, which

collected environmental data in each frame and prepared them for analysis by the trained model. Following the creation of the trained model, every vehicle navigating the lanes of the CARLA Environment is capable of recognizing traffic signs for the specific range of speeds. The speed ranges from 30 km/h to 60 km/h to 90 km/h, and is separated into three categories here for the regression model's intended identification.

4.4.2 Detection of speed limit 60 km/hr traffic sign

Fig 4.5: Traffic sign detection for speed limit of 60km/hr..

In this case, the speed limit 60 km/hr traffic sign is detected and displayed on the bottom left corner of Fig. 4.5. Several more of these signs were found and detected in various Carla environments to check the validity of the object detection model.

4.4.3 Detection of speed limit 30 km/hr traffic sign

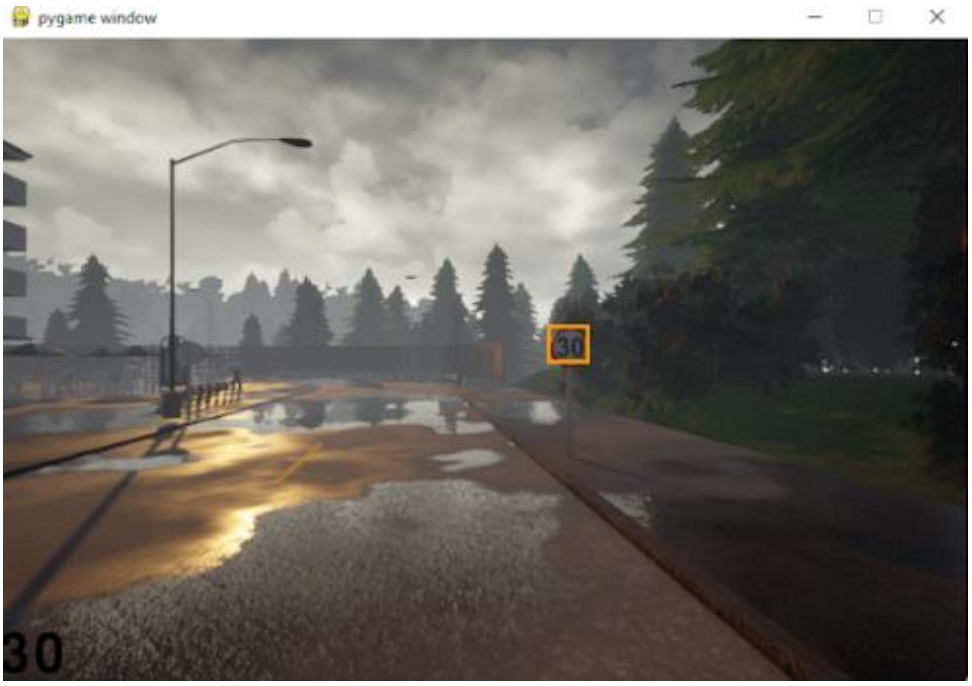

Fig 4.6 : Traffic sign detection for speed limit of 30km/hr.

Similar to cases 1 and 2, here traffic signs displaying speed limit 30 km/hr are detected using the object detection model and OCR.

4.4.4 Detection of Stop traffic sign

Fig 4.6: Traffic sign detection for STOP sign

In this case, a STOP sign is detected in clear weather in the CARLA environment. This process was repeated for several different conditions and the training dataset was obtained. The model was tested against this to obtain the results.

4.4.5 Confusion Matrix for Traffic Sign Detection

Table 4.3 Confusion Matrix for Traffic Sign Detection

Confusion Matrix		Actual Class			
		Class Speed Limit 90 km/hr	**Class Speed Limit 60 km/hr**	**Class Speed Limit 30 km/hr**	**Class STOP sign**
Predicted Class	**Class Speed Limit 90 km/hr**	84	3	0	0
	Class Speed Limit 60 km/hr	6	75	2	1
	Class Speed Limit 30 km/hr	1	3	48	0
	Class STOP sign	3	0	0	30

An illustration of the discrepancies between actual and predicted values is a confusion matrix. It assesses how effectively our machine learning classification model works and has a table-like shape. The numerous comparisons between actual and anticipated values are shown. Values that both turned out to be positive and were anticipated to be positive are known as True Positives (TP). Values that were actually expected to be negative but ended up being positive are known as false positives, or FPs. also referred to as a Type I mistake. Values that were projected as negative but were actually positive are known as false negatives, or FNs. the same kind of error as Type II. Values that were both actually negative and expected to be negative are known as true negative values. Here, classes have been taken dividing them into Class Speed Limit 90, Class Speed Limit 60, Class Speed Limit 30 along wit the Class Stop sign. The Perdicted Class represents the results

that were expected to happen or were predicted, whereas the Actual Class reflects the results that were really received. The diagonal elements in the table, as can be seen, are all True Positives, indicating absolutely true results. False Positive is indicated by the row elements, and False Negative by the column elements. Here, 6 represents 90km/hour incorrectly classified and 2 reprsents 30km/hour incorrectly classified. The Confusion Matrix was created for a successful performance analysis by designing, implementing, and comparing the other classes in a similar manner.

4.5 Performance analysis of Traffic Sign Detection

Table 4.4: Performance analysis of Traffic Sign Detection using results from Confusion Matrix

Parameter/Class	Class speed limit 90 km/hr	Class speed limit 60 km/hr	Class speed limit 30 km/hr	Class STOP sign
Accuracy	94.92%	94.1%	97.66%	98.44%
Error Rate	5.08%	5.9%	2.34%	1.56%
Precision	97%	89%	92%	91%
Recall	89%	93%	96%	97%
F1 Score	93%	91%	94%	94%
True Positive Rate (Sensitivity)	96.55%	89.28%	92.20%	90.9%
False Positive Rate	3.45%	10.72%	7.8%	9.1%
True Negative Rate (Specificity)	89.36%	92.59%	96%	96.77%
False Negative Rate	10.64%	7.49%	4%	3.23%

Using the performance Table 4.4, which compares the performance of the algorithms in the CARLA Environment The following metrics have been calculated for the speed limit 90, 60, 30 km/hour classes : accuracy, error rate, precision, recall, F1 Score, true positive

rate (sensitivity), false positive rate, true negative rate, and false negative rate. As seen, Class 30 achieved an accuracy of 97.66 percent, demonstrating improved traffic sign detection for 30km/hour in this particular class where the model is more strongly supported. Class speed limit 90km/hour has achieved the highest levels of Precision and class 30 speed limit has achieved more F1 Score, and Recall. In Class 90km/hour, the False positive rate is also the lowest and the False negative Rate is the lowest. Overall precision of 92.25 percent and an accuracy of 96.28 percent have been attained.

4.6 Traffic Sign Detection Performance Evaluation

The object detection model developed for traffic sign detection using YOLO gives a precision of 89.33%. This shows an improvement of 12.33% when compared to an existing model which also uses YOLO architecture and has a precision of 77%. There is an upgrade in recall by 0.75% and in F1 score by 8% [34].

Another point to note is that the existing model does not detect the individual speed limit sign values. This is an added advantage of the model developed

Summary

Lane and Traffic Sign Detection applications are successfully implemented in CARLA simulator using SegNet and YOLO. After training and testing the models against the datasets created, it is observed that both models are providing good and efficient results with Lane Detection showing an overall accuracy of 92.58 percent and Traffic Sign Detection showing an overall accuracy of 96.28 percent. These results are also compared to existing models to verify that there is an improvement in the developed model.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

Unreal environment of CARLA was used to deploy Semantic Segmentation and Object Detection models for the detection of lanes and traffic signs. These applications for autonomous vehicles were successfully implemented using SegNet and YOLO architecture respectively.

For the Lane and Traffic Sign Detection algorithms implemented in the CARLA virtual environment, accuracy and precision of the models were observed to be the most crucial parameters for Advanced Driver Assistance Systems (ADAS), as even the smallest mistakes might result in unfavourable and catastrophic circumstances. For the purpose of testing these models, two separate datasets were created and analysed.

Along with other performance indicators such as recall and specificity, lane detection's results show an accuracy of 93.66 percent and a precision of 89.33 percent. It is observed that the detection of the right lane is providing very good results. Hence, by enhancing the detection of the centre and left lanes and evaluating this against a solid training dataset, the error margin of 6.44 percent can be further reduced.

The object detection model is used to detect 4 classes of traffic signs. It is detected with an average accuracy and precision of 92.58 and 96.28 percent, respectively. By taking into account more training datasets and increasing the number of images it is tested against, the error rate of 3.72 percent of the model can be minimised. A model that can recognise lanes and traffic signs for autonomous vehicles would be produced by combining the two.

5.2 Future scope

The Semantic Segmentation model and Object Detection model trained on Carla can be used to test real world video and images to improve the accuracy of the Lane Detection. To further develop the Object Detection model a wider variety of signs and dynamic objects such as changing traffic signal lights can be included. A hybrid model of the

implemented models as an ADAS application can be created. The scope of lane detection in the future will include complicated surroundings that take into consideration many variables, such as the weather when there is fog, mist, sunny, bright day light, darker, or when there is an occurrence roadblocks like Humps and Speed Limiters. The goal is to create a system for automatically detecting and recognising traffic signs that is more accurate, resilient to failures, and strong in a variety of challenging conditions. The YOLO technique will therefore be used in additional work to improve object detection utilising an improved framework and libraries. In a subsequent step, traffic marks can be recognised and categorised using single-stage decoders. Utilizing an additional traffic-sign classification network can be avoided with the use of this technology.

5.3 Learning Outcomes

Understanding of the theory and basics underlying the object detection models with different improvements made to the setting, familiarity with the CARLA simulated environment was felt. TensorFlow's practical implementation was built in order to improve the procedure. It was developed and analysed to have knowledge of numerous performance measures and their implications in comparison to other regular outcomes achieved by other studies. The creation and utilisation of the dataset that the academic community can utilise for a variety of additional autonomous driving research initiatives was made. The CARLA Simulator's features and technical information were learned. Writing code for distinct path alterations and environmental detection necessitated the development of coding skills. Coding and debugging abilities were improved as needed. With all the intricate concepts learned, a very positive exposure to technical and analytical skills was made.

References

[1] Qusay Sellat, SukantKishoro Bisoy, Rojalina Priyadarshini, Ankit Vidyarthi, Sandeep Kautish, Rabindra K. Barik, "Intelligent Semantic Segmentation for Self-Driving Vehicles Using Deep Learning", *Computational Intelligence and Neuroscience, vol. 2022, Article ID 6390260,pp. 1-8, 2022,*DOI: 10.1155/2022/6390260

[2] Ivanovs, M.; Ozols, K.; Dobrajs, A.; Kadikis, R. "Improving Semantic Segmentation of Urban Scenes for Self-Driving Cars with Synthetic Images". *Sensors 2022, 22, 2252.*

[3] M. Hosein Hamian, A. Beikmohammadi, A. Ahmadi and B. Nasersharif, "Semantic Segmentation of Autonomous Driving Images by the Combination of Deep Learning and Classical Segmentation," *2021 26th International Computer Conference, Computer Society of Iran (CSICC), 2021, pp. 1-6,* DOI: 10.1109/CSICC52343.2021.9420573.

[4] Elharrouss, Omar, Somaya Ali Al-Maadeed, Nandhini Subramanian, Najmath Ottakath, Noor Almaadeed and Yassine Himeur. "Panoptic Segmentation: A Review." *ArXiv abs/2111.10250 pp.1-24 (2021).*

[5] Papadeas, I.; Tsochatzidis, L.; Amanatiadis, A.; Pratikakis, I. "Real-Time Semantic Image Segmentation with Deep Learning for Autonomous Driving": *A Survey. Appl. Sci.pp.1-19 2021, 11, 8802.*

[6] Jack Stelling, Amir Atapour-Abarghouei, "Just Drive": Colour Bias Mitigation for SemanticSegmentation in the Context of Urban Driving, *2021 IEEE International Conference on Big Data (IEEE BigData 2021),pp.3-9,* DOI :Thu, 2 Dec 2021.

[7] ÇKaymak, Çagri and Ayşegül Uçar. "A Brief Survey and an Application of Semantic Image Segmentation for Autonomous Driving." *Handbook of Deep Learning Applications, pp. 14-34(2019).*

[8] Von Rueden, Laura, Tim Wirtz, Fabian Hueger, Jan David Schneider, Nico Piatkowski and Christian Bauckhage. "Street-Map Based Validation of Semantic Segmentation in Autonomous Driving." *2020 25th International Conference on Pattern Recognition (ICPR)* (2021): 10203-10210.

[9] Kuo-Kun Tseng, Jiangrui Lin, Chien-Ming Chen, Mohammad Mehedi Hassan, "A fast instance segmentation with one-stage multi-task deep neural network for autonomous driving" *Volume 93, 2021, 107194, ISSN 0045-7906,*

[10] Zhang, Wenhui & Mahale, Tejas, "End to End Video Segmentation for Driving : Lane Detection For Autonomous Car*", Dec 2018,* DOI :org/10.48550/arXiv.1812.05914

[11] Jiaxing Sun, Yujie Li, "Multi-feature fusion network for road scene semantic segmentation, Computers & Electrical Engineering", *vol. 92, April 2021,pp.1-3* DOI: 10.1016/j.compeleceng.2021.107155

[12] Jadon , Shruti., "SemSegLoss: A python package of loss functions for semantic segmentation, Software Impacts*",vol. 9, May. 2021, pp.1-2* DOI :10.1016/j.simpa.2021.100078.

[13] M. Aygun, *et al*., "4D Panoptic LiDAR Segmentation",*in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2021 pp. 5523-5533,* DOI: 10.1109/CVPR46437.2021.00548

[14] J. Corrochano, J. M. Alonso-Weber, M. P. Sesmero, and A. Sanchis, "Lane following Learning Based on Semantic Segmentation with Chroma Key and Image Superposition," *Electronics, vol. 10, no. 24, p. 3113, Dec. 2021,* DOI: 10.3390/electronics10243113.

[15] D. Lee, W. P. Tay, and S.-C. Kee, "Birds Eye View Look-Up Table Estimation with Semantic Segmentation," *Applied Sciences, vol. 11, no. 17, p. 8047, Aug. 2021,* DOI: 10.3390/app11178047.

[16] L. Wang, R. Li, D. Wang, C. Duan, T. Wang, and X. Meng, "Transformer Meets Convolution: A Bilateral Awareness Network for Semantic Segmentation of Very Fine Resolution Urban Scene Images," *Remote Sensing, vol. 13, no. 16, p. 3065, Aug. 2021,* DOI: 10.3390/rs13163065.

[17] J.-J. Ponciano, M. Roetner, A. Reiterer, and F. Boochs, "Object Semantic Segmentation in Point Clouds—Comparison of a Deep Learning and a Knowledge-Based

Method," *ISPRS International Journal of Geo-Information, vol. 10, no. 4, pp. 256, Apr. 2021* DOI : 10.3390/ijgi10040256.

[18] Peng, Kunyu & Fei, Juncong & Yang, Kailun & Roitberg, Alina & Zhang, Jiaming & Bieder, Frank & Heidenreich, Philipp & Stiller, Christoph & Stiefelhagen, Rainer. (2022). "MASS: Multi-Attentional Semantic Segmentation of LiDAR Data for Dense Top-View Understanding. IEEE Transactions on Intelligent Transportation Systems." *pp.1-17. 10.1109/TITS.2022.3145588.*

[19] Lu, S.; Luo, Z.; Gao, F.; Liu, M.; Chang, K.; Piao, C. "A Fast and Robust Lane Detection Method Based on Semantic Segmentation and Optical Flow Estimation." *Sensors, pp.8-12,2021, 21, 400.*

[20] Sakaridis, Christos et al. "ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding." *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021): 10745-10755.

[21] Lidia Fantauzzo, Eros Fani', Debora Caldarola, Antonio Tavera, Fabio Cermelli, Marco Ciccone, Barbara Caputo:FedDrive: "Generalizing Federated Learning to Semantic Segmentation in Autonomous Driving." *CoRR pp.1-7abs/2202.13670 (2022)*

[22] Maturana, Daniel, Po-wei Chou, Masashi Uenoyama and Sebastian A. Scherer. "Real-Time Semantic Mapping for Autonomous Off-Road Navigation." *FSR* (2017).

[23] Kuutti, Sampo & Bowden, Richard & Jin, Yaochu & Barber, Phil & Fallah, Saber. (2019). "A Survey of Deep Learning Applications to Autonomous Vehicle Control." *IEEE Transactions on Intelligent Transportation Systems. pp. 10.1109/TITS.2019.2962338.*

[24] Gao, Biao & Zhao, Xijun & Zhao, Huijing. (2022). "An Active and Contrastive Learning Framework for Fine-Grained Off-Road Semantic Segmentation." DOI : 2021

[25] Pengchuan Xiao, Zhenlei Shao1, Steven Hao, Zishuo Zhang, Xiaolin Chai, "PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving". *pp 1-6,* DOI Dec 2021.

[26] Perumal, P. & Sujasree, M. & Chavhan, Suresh & Gupta, Deepak & Mukthineni, Venkat & Shimgekar, Soorya & Khanna, Ashish & Fortino, Giancarlo. (2021). An insight into crash avoidance and overtaking advice systems for Autonomous Vehicles: A review, challenges and solutions. "*Engineering Applications of Artificial Intelligence". 104. 104406. 10.1016/j.engappai.2021.104406.*

[27] Varghese, Lijo & Jacob, Suma & Chinnasamy, Sundar & I, Jacob. (2021). "Design and Implementation of a Machine Learning Assisted Smart Wheelchair in an IoT Environment".DOI:10.21203/rs.3.rs-490123.

[28] Phung, & Rhee,. (2019), "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets." *Applied Sciences. 9. 4500. 10.3390/app9214500.*

[29] Ghosh, Tonmoy & Li, Linfeng & Chakareski, Jacob. (2018). "Effective Deep Learning for Semantic Segmentation Based Bleeding Zone Detection in Capsule Endoscopy Images". 3034-3038. 10.1109/ICIP.2018.8451300.

[30] Badrinarayanan, Vijay & Kendall, Alex & Cipolla, Roberto. (2017). "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". DOI:10.17863/CAM.17966 10.17863/CAM.17966.

[31] Wu, Jiatu. (2018). "Complexity and accuracy analysis of common artificial neural networks on pedestrian detection". *MATEC Web of Conferences*. Doi:10.1051/matecconf/201823201003232.01003.10.1051

[32] Dhandapani, Karthikeyan & P., Arumbu & Surendhirababu, K. & Selvakumar, K. & Divya, P. & Suhasini, P. & Palanisamy, R.. (2021). Sophisticated and modernized library running system with OCR algorithm using IoT. *Indonesian Journal of Electrical Engineering and Computer Science.* DOI:10.11591/ijeecs.v24.i3.pp1680-1691 24

[33] Sato, Takami & Chen, Qi Alfred. (2022). "Towards Driving-Oriented Metric for Lane Detection Models".

[34] R Singh, M Danish, V Purohit, A Siddiqui. (2021). "Traffic Sign Detection using YOLOv4". *IJCRT Volume 9, Issue 5 May 2021, ISSN: 2320-2882*. DOI: 5 May 2021, ISSN: 2320-2882.

CO – PO Mapping

Mapping of **"Lane Detection and Traffic Sign Detection for Autonomous Vehicles Using Carla Simulator"** to Program Outcomes (PO)

Program Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
Lane Detection and Traffic Sign Detection for Autonomous Vehicles Using Carla Simulator	H	L	M	M	H	M	M	H	H	H	H	H

Mapping of **"Lane Detection and Traffic Sign Detection for Autonomous Vehicles Using Carla Simulator"** to Course Outcomes (CO)

Course Outcomes	CO1	CO2	CO3	CO4
Lane Detection and Traffic Sign Detection for Autonomous Vehicles Using Carla Simulator	H	H	M	L

Student Names **Signature**

Samudyata A

Hithaishi Surendra

S Bhuvana

Guide Signature