Deep Neural Networks for Road Scene Perception in Autonomous Vehicles Using LiDARs and Vision Sensors

Sabeeha Mehtab

A thesis submitted to the Auckland University of Technology

in the partial fulfilment of the requirements for the degree of Doctor of Philosophy

2022

School of Engineering, Computer & Mathematical Sciences

Abstract

Accurate object detection in road scenes is one of the most essential requirements of autonomous vehicles. A plenty of research work has been carried out in the two-dimensional field to support traffic surveillance, vehicle counting, and object tracking. However, less emphasis has been put forward on 3D object detection complications. Based on our findings, the existing solutions for autonomous vehicles rely on expensive 64 beams three-dimensional LiDAR (i.e., Light Detection and Ranging) point clouds for positioning the objects in real-world that highly raises the cost of autonomous vehicles and imposes the biggest barrier to their adaption.

Considering the limitations of existing solutions, in this thesis, we aim to give accurate three-dimensional object detection using sparse LiDAR point clouds in support with camera images. A unified detection framework is proposed for road scene perception including cars, pedestrians, and cyclists for optimizing the accuracy of 2D object detection based on the existing hardware and datasets. However, for final three-dimensional object detection, this research work is narrowed down to vehicle detection only due to time and resources constraints of academic research.

In 2D road scene perception precision, a flexible deep neural network is proposed by using the end-to-end detection approach named FlexiNet. The dynamic architecture of this network allows network scaling to obtain the best results based on the available resources. The proposed 3D vehicle detection algorithm exploits the two-dimensional bounding boxes extracted by using FlexiNet. The algorithm does not rely on dense point clouds, rather based on the prime fact that the three-dimensional center of a vehicle is the translation of its two-dimensional center in the form of world coordinates. The model performance is also analyzed based on 64, 32 and 16 beams density point clouds.

Keywords: Deep neural network, 2D vehicle detection, 3D vehicle detection, autonomous vehicles, self-driving car, LiDAR, point clouds, fusion

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature: Date: 26/05/2022

Acknowledgement

First of all, I would like to thank my primary supervisor Wei Qi Yan for his consistent support and motivation in my journey of Doctor of Philosophy. His kind supervision helped me to accomplish my research goals and kept me moving forward. He has been a strong source of inspiration and guidance to me in my academic journey.

My special thanks go to my other supervisor Ajit Narayanan, for his strong guidance, valuable suggestions and motivational remarks. I have been fortunate enough to get benefitted from his expertise and knowledge. I will always be grateful to him for the support he has provided. I highly appreciate his time and concerns.

I wish all the best for Karishma Bhatt, she has been kind and helpful to me in all these years. I thank all the AUT administrative and technical staff for their continuous support; everyone has been cooperative and helpful throughout this journey of mine.

I am very much thankful to my family, my husband, who remained very considerate and supportive towards me to accomplish my goals and worked with me to resolve any problem I faced. My sons understood the need of time and cooperated with me in the best possible ways. I would like to give special thanks to my mother and all my sisters and brother, who have been a pillars of strength to me constantly.

Sabeeha Mehtab

Auckland, New Zealand

May 2022

Table of Content

List of Figures

List of Tables

Acronyms

2D: Two Dimensional

3D: Three Dimensional

3DOP: 3D Object Proposal

AV: Autonomous Vehicles

Adam: Adaptive Moment Estimation

ACS: Automated Control System

AVs: Autonomous Vehicles

BN: Batch Normalization

BCEL: Binary Cross-Entropy with Logits Loss

BEV: Bird Eye View

BS3D: Bounding Shape SSD

CIoU: Complete Intersection over Union

CNN: Convolutional Neural Network

CSPNet: Cross-Stage-Partial-connections network

DNN: Deep Neural Network

DPM: Deformable Part-based Model

DenseNet: Densely Connected Network

DIoU: Distance Intersection over Union

FReLU: Flexible ReLU

FCN: Fully Convolutional Network

GIoU: Generalized Intersection over Union

GPS: Global Positioning System

GPU: Graphics Processing Unit

HD maps: High-Definition maps

HOG: Histogram of Oriented Gradients

IMU: Intertial Measurement Unit

IoU: Intersection over Union

LiDAR: Light Detection and Ranging

LBP: Local Binary Pattern

MAE: Mean Absolute Error

MSE: Mean Squared Error

MLP: Multi-Layer Perceptron

MV3D: Multi-View 3D

NMS: Non-maximum Suppression

RGB: Red Green Blue

ReLU: Rectified Linear Unit

ROI: Region of Interests

RPN: Region Proposal Network

ResNet: Residual Network

SSD : Single-Shot Multibox Detector

SAE: Society of Automotive Engineers

SPP: Spatial Pyramid Pooling

SGD: Stochastic gradient descent

SVM: Support Vector Machine

YOLO: You Look Only Once

Chapter 1

Introduction

This chapter is composed of six parts. In the first part, we introduce the rationale and significance of the study. The second part includes the motivation followed by the research background. In the third part, we present the research questions in follow up with contributions and the structure of this PhD thesis.

1.1 Rationale and Significance of the Study

According to the World Health Organization (WHO) global status report on road safety 2020 (WHO, 2020), road traffic crashes take the lives of nearly 1.3 million people every year and injure 20–50 million. most of these cases, people are driving under the influence of drugs, drowsiness, fatigue or distracted attention. On the other hand, with artificially intelligent vehicles, we expect they will always follow road rules and will never be tired off with proper maintenance. Moreover, it will also be of great use to the aged or physically challenged people in addition to saving the driver's time. Self-driving vehicles will potentially improve safety, reduce pollution, and provide mobility solutions for otherwise underserved sectors of the population with their successful technology.

However, there are innumerable challenges to face for an AV, e.g., identifying road obstacles accurately is a complex task for the difference in lighting conditions, high degree occlusion, truncation of objects present in the images (Hai Wang et al. 2019, Gu et al., 2017). Many times, there are strong shadows of objects over others. On the shiny surface of vehicles or glass windows, reflections of surrounding objects also create a dilemma. Other major challenging conditions are extreme weather (snow, foggy, and heavy rainfall) and night timings. These are the practical road scene challenging conditions that AVs needs to deal with. A fundamental requirement to AVs technical core side is the ability to perceive the 3D road scene accurately in real-time.

With the goal of providing common terminology for automated driving, the Society of Automotive Engineers (SAE) classifies automotive driving, as shown in Fig. 1.1, in five levels based on their automation capacity and task requirements, for example, lane driving or closed campus driving (SAE, 2021). Briefly, we could summarize them as "level-0" that provides no automation, and "level-1" refers to hands-on driving where gears and/or pedal could be auto-controlled. "level-2" adds steering control automation over the previous levels; however, the driver needs to be vigilant to take over control in any critical circumstances. "level-3", the driver is allowed to disorient his work. However,

the automation system of the vehicle may ask the driver to take control over exceptional circumstances, e.g., traffic jams, so the driver must be present at the driver seat during the course. “level-4”, AV allows for fully automated control but in limited areas and can pull over to a safe place when necessary. The “level-5” is presumably fully automated without any need for human intervention; hence steering wheel would not be required at all with complete trust of AV accuracy and robustness.

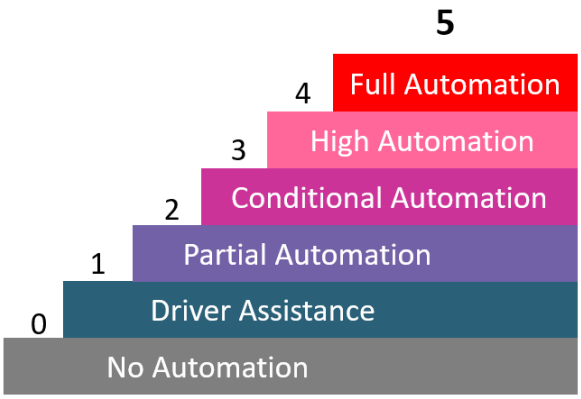


Fig. 1.1: SAE international Identifies six levels of driving automation from “no automation” to “full automation”.

Multiple sensors are incorporated in the AVs to collect maximum surrounding information, including camera, LiDAR, radar, and supplement sensors such as GPS, odometry and IMU (Asvadi et al., 2018). Over the past few years, many methods have been proposed to tackle the problem of 2D and 3D object detection from monocular images, stereo cameras or LiDAR 3D point clouds. In this information plays a vital role for Automated Control System (ACM) in perceiving the world, understanding the actions of objects around and deciding the safe manoeuvre. However, another essential requirement of AVs is to reduce the whole perception and reaction time to a minimum to make a safe move or stop of the AVs in case of an emergency.

To quote, the latest accident of Aug-2021, where Tesla Model S autopilot drove past a stop sign and a flashing red light before hitting a parked car (Gregory, 2021), and other

critical accidents are of July-2020 and Dec-2019 when autopilots couldn't take the right decision at the right time and had led to casualties. These accidents have revealed the scope of improvement in the field of AV system technology to the world. On the other hand, the latest development in AVs is robotaxi, owned by Waymo that are "level-4" self-driving cars and allowed to move in parts of Phoenix, Arizona, since 2020. Robotaxi can do safe pull over in case of emergency or ask humans to take over (Litman 2020). These vehicles are in an auto-learning mode, i.e., they learn new road driving possibilities based on the experience received during travel.

However, Waymo cars are equipped with multiple cameras, five LiDARs, two radars, and many other sensors that make their budget unapproachable for most of us. 2017, a Velodyne 64-beams LiDAR cost was 75,000US dollars that was cut off later by 50% in 2018 (Aijazi et. al., 2020). However, with the advent of Solid State LiDAR (SSL), it is assumed that LiDAR prices will slash down with a big difference in the coming years with their high production. The SSL utilizes a micro-lidar array with 8-beams giving a static 120-degree field of view (Aijazi et. al., 2020).

1.2 Motivation

Visual perception plays an integral part in autonomous driving to estimate the positions of surrounding obstacles on the road. Vehicle and pedestrian detection using aerial images have gained tremendous attention in traffic monitoring, surveillance, and military applications (Mo, 2020). The aerial view of vehicles shows less or no occlusion and shadows as compared to the ground-based images, whereas in the case of AV, it has to deal with the ground-based field of view only. Thus, extensive work is needed for object detection with the ground-based field of view to accelerate AVs success. Considering the consequences of a minor error in an AV perception and reaction generated requires highly sophisticated and robust algorithms. However, complex algorithms generally require heavy computation, subsequently leading to excessive run time.

In driving scenarios, perception-break reaction time (PRT) is defined as the time

lapse between the hazard seen and the reaction applied, ranging between 0.7 to 1.5 seconds depending on the driver's age, visual equity, and cognitive load (Green, 2000). Other than PRT, the car will take a while to come to a stop depending on the current speed, tyers tread depth, road friction and brake efficiency. Therefore, with the objective of human safety and reliable transport, an AV should be capable of detecting an object 20-60 meters beforehand with a 50-70 km/hr speed (Lantos and Márton, 2011).

In this research project, we aim to uplift 2D road scene perception accuracy for ground-based images by proposing a unified framework for detecting cars, pedestrians, and cyclists objects present in front of AV. Considering the fact that 2D detection is not sufficient for AV because that gives the resulting information in image coordinates only and no knowledge about the distance and shape of the objects present in front of AV. The 3D detection of road objects is a necessary requirement to understand their absolute positioning. At the same time, the objective is to find a solution that is not based on point clouds density and can efficiently work with sparse point clouds. the present research journey, we have targeted at optimizing 2D road scene perception accuracy of AV based on available hardware and dataset; and positioning front lying vehicles in the 3D world accurately based on the RGB images of the camera sensor and 3D sparse point clouds.

1.3 Research Background

An AV system is composed of three major technological components. Primary - sensing and perception, secondary - localization and mapping, and thirdly – application for driving policy (De Silva, Roche, and Kondoz 2018). In this research project, we have considered the sensory and perception system that is responsible for understanding the surrounding environment. Human primarily takes use visual, auditory and cognitive senses to drive, whereas AV perceives the world with multiple sensors to overcome the shortcomings of individual ones (Mehtab et. al., 2021). A myriad of sensors can be used to collect surrounding information: Passive sensors, such as monocular, stereo, and infrared cameras; in contrast to active sensors including GPS (i.e., Global Positioning

System), LiDAR, IMU (Inertial Measurement Unit), radar and sonar (Kuutti et al., 2018). However, in recent years, most 3D object detection algorithms have preferred to use monocular/RGB cameras and LiDAR sensors.

A monocular camera outputs digital images rich in texture and reveals the shape of the objects in the form of pixel values, such as RGB, YUV, or other colour systems (Podpora, Korbaś, and Kawala-Janik, 2014). On the other hand, LiDAR detects object location accurately irrespective of its visibility, however in the range of certain meters only because of radiation limitation (Wei et al., 2018). Unlike monocular cameras, LiDARs cannot discriminate the objects based on texture and colour (Wei et al., 2018). LiDAR data has been utilized in multiple ways in AV systems, for example, front-view 2D projected form (Z. Wang, Zhan, and Tomizuka, 2018), top-view or BEV projected form (Ku et al., 2018), voxel form, i.e., 3D grid cell (Zhou and Tuzel, 2018), or raw form only (Qi et al., 2017). Based on the facts mentioned above, it is justified to integrate the information from LiDAR and cameras to complement each other.

In recent years, many advanced DNN models have been proposed for AVs for 2D and 3D road objects detections. Despite considerable success in the accuracy of 2D detection of individual road objects, there is much scope for improvement in their parallel execution. On the other hand, 3D road scene object detection still has a marginal difference in accuracy compared to 2D detection. the case of 3D object detection, LiDAR has shown significant importance, whereas the fusion of camera and LiDAR sensors has given some reliable results (Liang et al., 2019; Ku et al., 2018). Based on our investigation, maximum accuracy achieved on medium complexity KITTI benchmark datasets (Geiger et al., 2013) for 3D detection to date is up to 87.47% (Liang et al., 2019) using densely fused two-stream multi-sensor; however, for road safety aspect it has a considerably less and demands an improvement.

1.4 Research Method

As a research discipline, we have adopted design science methodology (Peffers et al.,

2007) that follows iterative procedure of implementation. It allows to repeatedly improve the network model based on the evaluation results to achieve the desired accuracy. The exploited research methodology is depicted in Fig. 1.2 that identifies the research problems based on literature reviewed, derives objectives of solutions, and proposes model design. Furthermore, model simulation is performed that is followed by quantitative evaluation. For quantitative evaluation, precision, mean average precision, recall, accuracy and inference time metrics are considered. Various approaches are adopted based on deep learning techniques to optimize the results iteratively. After achieving the desired results, work is published in relevant media.

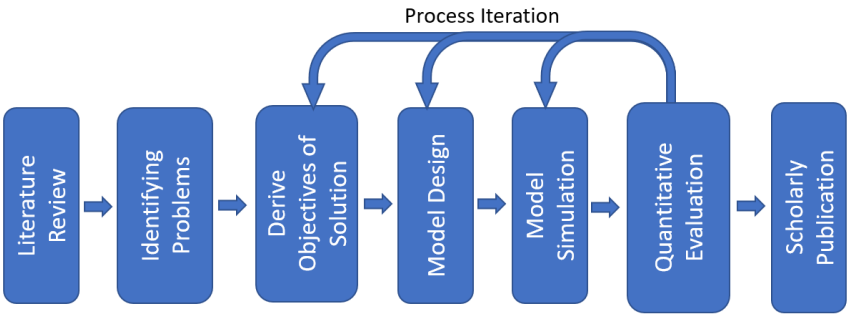


Fig. 1.2: Research Methodology

Fig. 1.3 shows the overall flow of model design. After comprehensive survey, two open source self-driving car datasets were selected. After performing data augmentation and data distribution, the research conducted 2D road scene perception using deep neural network (DNN) based on RGB images. However, 3D vehicle detection exploited the results generated through 2D detection and it was based on RGB images as well as LiDAR point clouds.

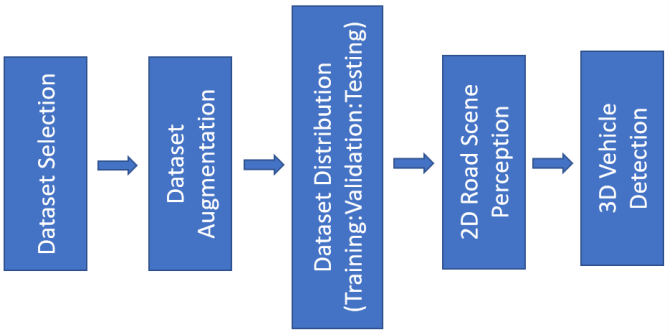


Fig 1.3: Overall flow of model design

1.5 Research Problems

In 2D road scene perception, extensive work has been carried out for individual objects like cars, pedestrians, and cyclists detection using machine learning and DNN models; however, comparatively, very few efforts have been put into a unified framework of detection. Moreover, pertaining to humans pose variation and occlusion, detecting pedestrians and cyclists with high precision has remained challenging. In the case of 3D road scene perception, most of the recent developments have considered 64 beam dense LiDAR point clouds to get the world coordinates; however, that results in an overall unapproachable price of AV causing the biggest barrier to its adoption. For above mentioned reasons, the research problems can be stated as follows:

Problem 1. How to increase the accuracy of a unified framework in 2D road scene perception for AVs that can efficiently deal with different road users simultaneously?

Problem 2. What should be the architecture of DNN to obtain optimum results in 2D road scene perception based on the existing dataset and hardware resources?

Problem 3. How to give an accurate yet cost-effective model for 3D object detection for AVs based on sparse LiDAR point clouds and camera images?

1.6 Research Contributions

In the proposed research project, we have firstly emphasized on the accuracy improvement of 2D road scene perception exploiting the DNN potential with the input provided in the form of camera RGB images. In the late section, the focus is on 3D object detection, in this area, our research work is narrowed down to vehicle detection only considering the time and resources constraints of academic research. We have utilized the 3D point clouds knowledge grasped from LiDAR with 2D detection results obtained to predict the vehicles 3D positions present in front of AV.

1.6.1 2D Road Scene Perception

For 2D perception, we aim to detect and classify cars, pedestrians, and cyclists in real-time with high accuracy in complex road scenes using a unified DNN framework. The proposed PyTorch-based framework attains the least inference time that allows ACM to make manoeuvre decisions in a safe amount of time. The network architecture is inspired by YOLOv5 (Jocher et al., 2020), based on a single regression network for object classification and detection via region anchors to make object detection robust. In the proposed solution, we investigate a flexible neural network to generate an optimized DNN architecture (Mehtab and Yan, 2021). The contribution in the proposed work can be summarized as follows:

- Based on YOLO5 framework (Jocher et al., 2020), a dynamic scaling neural network is proposed to get the required size of DNN that achieves the best result for detection based on the available hardware and dataset.

- Based on existing optimization techniques (Ian, Yoshua, and Aaron, 2016) for DNN, the network is fine-tuned by using different batch sizes, loss functions, activation functions, optimization functions and intermediate checkpoints to achieve the desired accuracy.

- On benchmark KITTI dataset with medium complexity, we have obtained 95.01%, 93.32%, and 99.33% recall for vehicles, pedestrians, and cars respectively.

The network performance is compared with the state-of-the-art 2D detection methods that depict the superior performance of the proposed work by yielding the lowest missing rate with the highest detection speed.

Based on the literature surveyed, it was found that extensive work has been carried out for the car, pedestrian, and cyclist detections independently; however, comparatively, fewer efforts have been put collectively into their 2D object localization. The proposed FlexiNet model provides a unified framework for the detection of multiple classes together with high inference speed. Our results are analysed based on precision and recall metrics based on the detection dataset. The result analysis reveals that detection models such as Faster R-CNN (Ren et al. 2017) and EfficientDet (Tan, Pang, and Le 2020) faced difficulty differentiating features between cyclists and pedestrians, leading to low recall and precision. In contrast, FlexiNet achieved desirable performance with the lowest miss rate based on the KITTI dataset (Mehtab and Yan, 2021). YOLOv4 achieved comparable results, although it demands more computational resources as compared to the proposed model.

1.6.2 3D Road Scene Perception

Based on the results achieved from the 2D detection method, we have proposed 3D vehicle detection using camera RGB images along with LIDAR point clouds. In this piece of work, a cost-effective solution for 3D vehicle detection based on sparse point clouds in the context of AVs is proposed. A novel framework to estimate 3D bounding boxes and orientations of the front lying cars is proposed based on the success of 2D object detection using DNN (Mehtab et. al., 2021). The contribution of this work can be summarized as follows:

- The proposed solution leverages from existing MobileNetV2 (Sandler et al., 2018) architecture for feature extraction. The last fully connected layer of original network is replaced with three branches pertaining to 3D box size, orientation, and confidence score.

- The novel 3D vehicle detection algorithm gives regard to the fact that the 3D centre of a car in world coordinates is the translation of the predicted 2D centre with the referenced AV field of view.

- Relying on the factual information of LiDAR beams, car distance from AV is estimated using trigonometrical geometry based on predicted size, orientation, and estimated pose of car 3D bounding boxes.

- A solution is proposed to mitigate the occlusion problem in real-world scenarios exploiting relative distancing between the front visible and occluded cars using point clouds information.

- To test the network performance over sparse point clouds, KITTI 64 data of beam point clouds is transformed into 32 and 16 beam formats. The results depict consistent performance of the proposed solution with sparse point clouds along with higher detection speed.

Our experimental results unfold that images and LiDAR point clouds early fusion leads to poor detection of 3D box size as compared to images only (Mehtab et. al., 2021). Rather in the proposed solution, LiDAR point clouds are employed later in 2D detection windows to estimate the distances of vehicles based on their active information. Benchmark KITTI dataset (Geiger et al., 2013) that contains a variety of challenging conditions of the roads is applied to test and evaluate network performance that is further verified by using the Waymo dataset (Sun, Henrik, and Xerxes 2020). In this research project, we present a novel idea of testing network performance over different densities point clouds without having other LiDARs.

1.7 Structure of This Thesis

The structure of this report is described as follows:

In Chapter 2, an exhaustive literature review of relevant research done in the field of AV for road scene perception is conducted. In this section, we present our introductory discussion about different sensors and open datasets used in the AV research. Numerous 2D and 3D road object detections techniques are considered and compared based on sensors and techniques used. In the light of the literature reviewed, critical analysis is made, leading to research gap findings.

Chapter 3 has shared some of the basics of neural networks, including Feedforward and Backpropagation, Gradient Descent algorithms, Convolution Neural Networks (CNNs), the advanced techniques to optimize neural network performance. Hereinafter, we have also discussed the basic principle of region proposal-based object detection and YOLOv3 to understand some of the concepts of the end-to-end paradigm.

In Chapter 4, the details of the research methodology are utilized to increase the accuracy of 2D road scene object detection are covered. A flexible deep neural network named FlexiNet is proposed to set the network's shape to find the most promising structure based on the dataset and available resources. To make the network potentially generalizable, the auto-anchor technique generates anchor sizes using the k-means clustering algorithm by using IoU features based on the training dataset information. In the late Section, the network performance is optimized by using various neural networking tools.

In Chapter 5, the research is narrowed down to vehicle detection exclusively. 3D vehicle detection is accomplished based on the success of 2D detection results by using additional 3D world coordinates information from LiDAR point clouds. The proposed solution can be summed up in two sections, where the first section predicts the size and orientation of 3D bounding boxes. On the other hand, the second section projects LiDAR

point clouds on the detected 2D bounding boxes to transform the 2D centres of the cars into 3D coordinates form (Mehtab et. al., 2022).

In Chapter 6, we have conducted result analysis and discussion based on the experimentation performed using both the methods proposed. In this chapter is comprised of three sections. In the first section discusses the results achieved in 2D road scene perception, the second section focuses only on 2D vehicle detection, whereas the third section focuses on 3D vehicle detection results with their summary (Mehtab and Yan, 2021).

In Chapter 7, we have summarized the subject and methods applied in this course of research with their outcomes. Light is reflected on the research areas that give a new direction according to the result and insufficiency of the experiment, preparing for future work. A look at what went wrong in the initial course of research is also discussed.

Chapter 2

Literature Review

In this chapter, we highlight the literature surveyed for developing a sound understanding of the state-of-the-art object detection techniques in autonomous driving. In the chapter covers three major sections pertaining to sensor analysis, data analysis, and research methods explored for the object detection used in recent years based on camera and LiDAR sensors. In the third section, we give the details of traditional object detection methods based on conventional machine learning and computer vision techniques, e.g., histogram of oriented gradients (HOG), support vector machine (SVM) etc., and DNN-based 2D object detection and 3D object detection methods. Clear segregation is carried out among the 3D detection methods based on the type of sensors used and the approach exploited to fuse the information from different modalities. In the last section, we discuss the literature surveyed and present the gaps existing in the state-of-the-art object detection methods.

2.1 Sensors and Datasets Analysis

Although human primarily takes use of visual, auditory systems and cognitive senses in driving, AV perceives the world with the help of multiple sensors to overcome the shortcomings of individual ones. There are a wide range of sensors in autonomous vehicles: Passive sensors, such as monocular, stereo cameras and infrared cameras etc., and active sensors, including GPS, LiDAR, IMU, radar and sonar (Yan, 2019). However, most of the object detection methods have relied on cameras and LiDAR sensors in recent years, so it becomes necessary to discuss these sensors in detail (Mehtab et. al., 2021).

2.1.1 Camera

A monocular camera provides rich textured images and reveals the shape of the objects in the form of pixel values in RGB, Grayscale or YUV and many other colour coding schemes (Podpora, Korbaś, and Kawala-Janikc, 2014). In the shape and texture information can be used to identify different objects, traffic signs and lane geometry (Zakaria et al. 2018) etc. A significant disadvantage of monocular cameras is the lack of depth information required for accurate object size and distance estimation (Arnold et al., 2019).

2.1.2 Laser Detection and Ranging (LiDAR)

LiDAR is a remote sensing device used in AV for estimating the distance of the road objects from AV by calculating the roundtrip time of emitted laser rays. LiDAR detection system gives 3D world coordinates and the intensity of the reflected beam in the form of point clouds (Wei et al., 2018). Market dominating mechanical LiDARs come with varying numbers of laser beams (i.e., 8-128) with an eye-safe technology in 360-degree horizontal and 10 to 40-degrees vertical field of view coverage. Based on the beam density, The latest LiDARs can empower AVs to locate the surrounding objects in the range between 50-250 meters along with their speed estimation. Being an active sensor, LiDAR

has many benefits over the camera by collecting 3D world coordinates of objects in the form of point clouds. It's immune to reflection, weather and lighting conditions. LiDAR can estimate velocity, orientation and the objects distance accurately in certain meters range.

The primary ways for formatting the point clouds in object detection methods are listed as follows:

- **2D Front-view Projected Form:** In some research (Minemura et al., 2018; Zhou et al., 2019), the 360-degree rotating mechanical LiDAR point clouds are converted into cylindrical coordinates to get a front view 2D projection. If the given 3D point coordinate is $P_1(x, y, z)$, then its corresponding cylindrical coordinates P_2 (r, c) in the frontal-view map can be calculated as given in Eq. (2.1) and (2.2).

$$c = atan2(y, x)/\partial\theta \tag{2.1}$$

$$r = atan2(z, \sqrt{x^2 + y^2})/\partial\emptyset \tag{2.2}$$

where $\partial\theta$ and $\partial\emptyset$ are the horizontal and vertical resolutions of the point clouds, respectively. Fig. 2.1 illustrates the front view projection of point clouds.

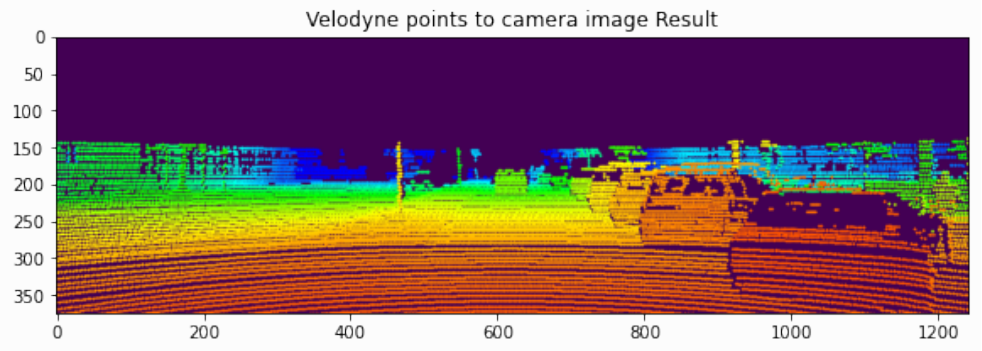


Fig. 2.1: Front view of LiDAR point clouds, 3D points are converted into cylindrical coordinates to project front view on a 2D plane.

- **Top View or Bird's Eye View (BEV) Projected Form:** The BEV view over the front view of point clouds is shown while generating projections (Yi et al., 2020; Ku et al., 2018; Chen et al. 2017). BEV represents the top view of point clouds, as the name implies from a bird's field of view, thereby removing the occlusion up to a certain extent. a BEV projection, the whole point clouds are distributed into an $N \times N$ centimetres horizontal grid with multiple vertical band slices, as shown in Fig 2.2. For every grid cell, BEV stores multiple channels holding the maximum point height in every vertical index, with the final channel having the overall density of points in that grid.

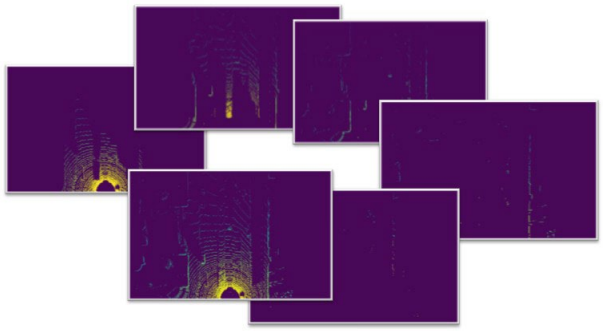


Fig. 2.2: BEV point cloud mapped in six channels based on different height levels and points' density.

- **Voxel form:** LiDAR point clouds are exploited in the form of 3D voxels (Yi et al., 2020; Zhou and Tuzel, 2018) to get high-performance detection results.
- **Raw point clouds**: PointNet and its variants are proposed based on raw point clouds that didn't consider changing the point clouds format to make them regular such as voxels or image projections but respected the perturbation and corruption (Qi et al. 2017; 2017; 2018) to give high accuracy in autonomous driving.

2.1.3 LiDAR and Camera Fusion

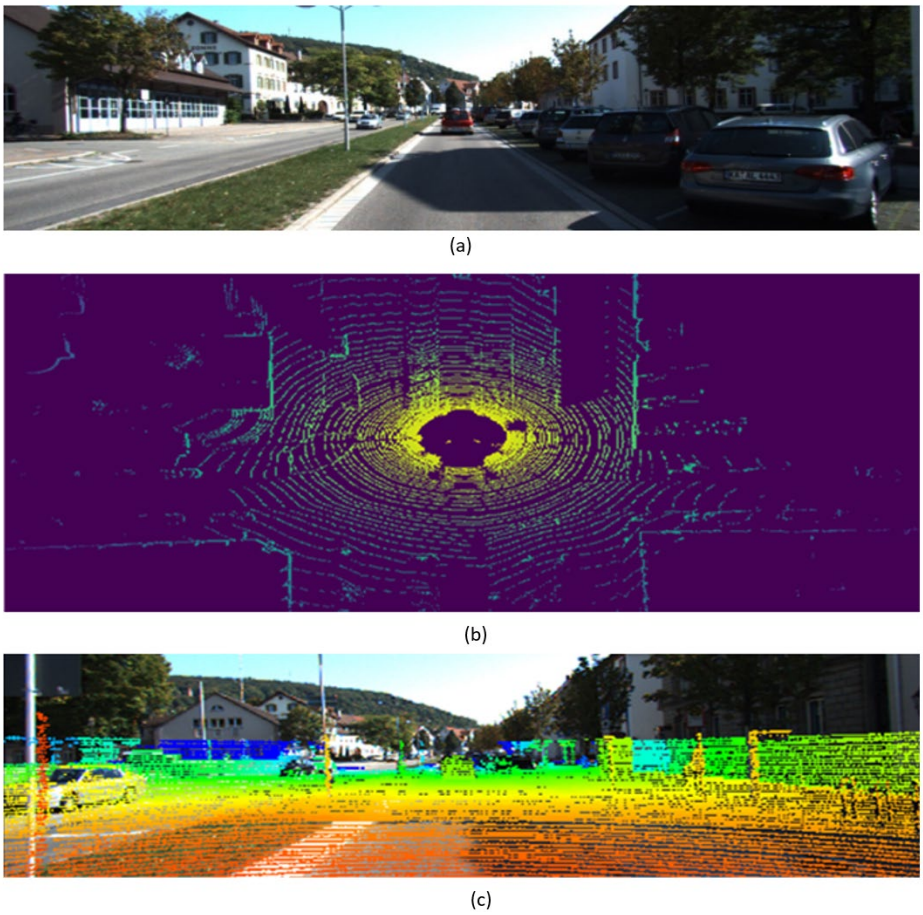


Fig. 2.3: (a) Camera View (b) LiDAR 360-degree BEV point cloud projection. (c) Early fusion of LiDAR and Camera Fusion

To deal with the individual limitations of camera and LiDAR sensors, multimodal fusion is employed to combine information from various sensors that lead to better decisions (Vielzeuf et al., 2018). In the fusion of these sensors helps in resolving the problem of the incompleteness of both the sensors.

In artificial neural networks, three approaches that were widely harnessed for multimodal fusion named “Early Fusion”, “Late Fusion”, and “Hybrid Fusion/multistage fusion” that differ in the way the information is integrated, features or decisions yielded from different sensors (Snoek, Worring, and Smeulders, 2005).

The early fusion approach feeds unimodal" raw data or features in a unified representation at the initial stage of the neural network to retrieve their semantic concepts. In this paradigm, the different sensor information is integrated at the early stages without transforming; early fusion attains actual multimodal features extraction (Snoek, Worring, and Smeulders, 2005). Fig. 2.3 presents the projection of 3D point clouds on an image, where (a) presents a visual scene, (b) shows the 360-degree 3D point clouds in BEV form. Fig 2.3(c) displays the early stage fusion where point clouds are superimposed on the image, aligning their field of views as a unified input source. Colour coding of point clouds depicts the distance of objects from LiDAR. In this kind of model is privileged by single-stream network processing.

On the other hand, the late fusion relies on the power of individual modalities and finds the detection score based on each sensor; and fuses processed information at a detection stage of the network (Asvadi et al., 2018; Wang, Zhan, and Tomizuka, 2018).

2.2 Dataset Analysis

Many organizations are facilitating open autonomous driving datasets for research purposes. One of the most used datasets in the self-driving context is KITTI (Geiger et al., 2013). KITTI has primarily been used as a benchmark for performance comparison among models. KITTI dataset provides right and left view cameras colour images, LiDAR's 360-degree point clouds and GPS coordinates, all synchronized in time. Scenes have been recorded from well-structured highways, complex urban areas and narrow countryside roads. In the dataset contains 15,000 frames: 7,500 labelled and 7,500 unlabeled. In the calibration of different sensors and labelling information of 2D and 3D bounding boxes is provided for each frame, along with the orientation.

The KITTI benchmark dataset has been categorized based on complexity using "Easy", "Medium", and "Hard" terms based on objects' size, occlusion, and truncation levels. However, the dataset has a few limitations, such as all scenes have been captured in the daytime and mostly on sunny days. Dataset is biased toward cars; less than 20% of

the dataset holds cyclist or pedestrians information (Geiger et al., 2013). Moreover, the dataset lacks twists and turns of roads and lanes, thereby reducing reliability in real-world applications.

Other than KITTI, the "nuScene" open dataset offered by APTIV (C, B, and H, 2019) and "Waymo" dataset (Sun, Henrik, and xer×es, 2020) also provide large, annotated information for autonomous driving cars through using multiple LiDARs and cameras information. As our analysis, the nuScene dataset offers a variety of road scenes; however, it lacks considering different weather conditions. On the other hand, Waymo facilitates diverse weather conditions and night-time scenes to the researchers. Another advantage of using Waymo is its availability at Google Colab (that makes it easy for researchers to take benefit of GPUs directly). Other organizations also provide dense pixel annotation of high-resolution RGB videos for multiple object classes. These include "APOLLOSCAPE" by Baidu (Wang, Peng, and Haung, 2020), "BDD100K" by Berkeley Deep Drive (Yu, Chen, and Wang, 2018) and "CITYSCAPES".

On the other hand, data from the virtual world environment may also be taken into consideration, such as the virtual KITTI 3D dataset (Gaidon et al., 2016). Simulators provide Flexibility to create an environment with different lighting, weather and traffic conditions. SIM4CV (Müller, Casser, and All, 2018) and CARLA (Dosovitskiy and Ros, 2017) are open-source simulation tools for autonomous driving, allowing flexible environmental setup and sensor configuration. Using virtual data during training can enhance the performance of detection models in natural environments (Arnold et al., 2019).

2.3 Methods Used for Object Detection

2.3.1 Traditional Object Detection

Computer vision methods have strongly relied on handcrafted features for features

extraction and semantic understanding based on images (Mehtab and Yan, 2022). Regarding feature extraction, the HOG (Histogram of Oriented Gradient) descriptor has been applied successfully in visual object detection using conventional machine learning methods along with the SVM classifier (Freeman and Roth, 1994; Dalal and Triggs, 2005). For pedestrians detection, low-level image features have been employed exhaustively to produce Region of Interest (ROI) by using different algorithms (Ahmed et. al., 2019). These methods have exploited HOG features and other methods such as decision trees or Local Binary Pattern (LBP) (Wang et. al., 2009).

Visual features of HOG have been exploited to detect cars (Zhang et al., 2018; Hou et al., 2018). A variant of the hierarchical HOG symmetrical feature was applied to different sides of the vehicle (Zhang et al., 2018). Another modified version of HOG (Hou et al., 2018) was introduced to cover gradient information from different angles. A two-step vehicle detection algorithm (Yun et al., 2019) was proposed based on combined results of HOG and Haar-such as features that focus the changes on intensities in the horizontal, vertical and diagonal directions to detect the object. Using RGB images (Zakaria et al., 2018), a HOG variant was proposed in combination with nonlinear SVM for vehicle detection. In the method was based on compass gradients to collect the features from multiple angles rather than horizontal and vertical restrictions. A lightweight vehicle detection method was experimented with using various colour schemes along with a HOG descriptor and SVM classifier with a sliding window over the selected ROI (Farag, 2020).

Object detection methods based on deformable parts models (DPM) is based on a sliding window approach for object detection (Felzenszwalb et al, 2010). DPM is based on a disjoint pipeline to draw static features of an image, and bounding box prediction and object classification. Based on the DPM, (Ghosh, 2017) proposed pedestrian detection and (Cho, Rybski, and Zhang, 2010) proposed cyclist detection. However, in comparison with state-of-the-art algorithms, DPM proved to be much slower due to excessive calculations (Redmon et al., 2016). Using ROI extraction and handcrafted features of monocular images, (Tian and Lauer, 2015) proposed cyclist detection.

However, these methods based on specified features of conventional machine learning are susceptible to occlusion and other complex environmental conditions (Mehtab et. al., 2021). Moreover, they cannot be applied to real-time scene understanding, whereas current deep learning methods allow the network to produce high-level and complex features of objects in addition to high speed of computing (Hu et al., 2017; Shah and Kapdi, 2018).

2.3.2 2D-Object Detection Methods Based on DNN

Owing to the advancement of Graphics Processing Unit (GPU), its costs have been reduced remarkably. On the other hand, with easy accessibility of extensive training, open-access datasets and advanced detection deep learning-based algorithms, Artificial Intelligence (AI) has significantly improved accuracy in road scene perception over traditional computer vision and machine learning-based methods. Object detection speed has also been considerably enhanced with persistently improving software tools such as PyTorch and TensorFlow that retrieve maximum benefits with the parallel processing GPU hardware.

Based on the literature reviewed, we have categorized DNN-based road scene understanding methods in two sections: 2D object detection and 3D object detection. In this section, we focus on the 2D detection methods proposed for autonomous driving based on DNN.

A myriad of popular 2D detection algorithms is based on the region proposal approach. However, Faster R-CNN (Ren et al. 2017) had a significant change in the way ROI should be selected over R-CNN based on existing methods, which is a state-of-the-art algorithm for object detection. Faster R-CNN took the concept of anchor boxes to propose candidates regions. Faster R-CNN, a shallow CNN named Region Proposal Network (RPN) is proposed to replace selective search that had remained the bottleneck in the previous versions and attained a speed of 5 fps using GPU with remarkable accuracy (Ren et al. 2017).

Another popular 2D detection approach is single-stage or end-to-end object detection with representative YOLO (i.e., You Look Only Once), SSD (i.e., Single Shot Multibox Detector), and RetinaNet algorithms (Hai Wang et al. 2019). These algorithms have not intermediate candidate region proposal layers and predict classification and bounding box detection results using a single regression network.

In the latest version of YOLO, the detection layer extracts feature maps from three different stages based on their scale. In the whole detection process is completed by using a single regression network that leads to speedy execution. In the YOLO series, the latest published work is YOLOv4, which divides the complete network into Backbone, Neck and Head Sections. YOLOv4 introduced bags of freebies and bags of special features to make different design specifications. Based on empirical testing, the backbone network produces the best features extraction using CSPNet (i.e., Cross-stage-partial-connections network) with efficient gradient propagation. In the neck section was designed using Feature Pyramid Network (FPN) based on upsampling operations. These different level features were used by the head network targeting the detection of varying size objects. Some working modules of YOLOv5 are discussed in (Thuan, 2021).

Acknowledging the limitations of the camera sensor, LiDAR and camera fusion have been conducted in early, late, and hybrid fashion (Caltagirone et al., 2019). Data from both modalities were fused and fed into an FCN network for object detection. Hybrid fusion achieves maximum 2D road/lane detection accuracy with around 96.06% results on the KITTI dataset (Caltagirone et al., 2019). Two processing branches from each sensor concatenate their outputs in a cross pattern after every convolution layer. Another multistage fusion approach (Yu et al., 2019) also achieved around *96.86%* accuracy for 2D road detection. These results depict that 2D road detection approaches have achieved a level of maturity even after many challenging lane conditions (Mehtab and Yan 2022).

The highest successful rate has been achieved for vehicle detection compared to pedestrian and cyclist detection in terms of road objects. Faster R-CNN was applied to video frames so as to get the desired accuracy (Tourani et al., 2019), though the algorithm

limits the computing speed of visual object detection, improving the basic structure of SSD (Jingwei et al., 2020) was proposed by adding inception blocks and feature fusion layers in the original network so as to detect distant car objects accurately. In the deep MANTA (Chabot et al., 2017) was designed based on the principle of RPN to find ROIs that were followed by two convolutional layers and fully connected layers to get 2D bounding boxes and key parts of vehicles. Targeted at blind-spot monitoring, a DNN using multiple cameras in autonomous driving (Shen and Yan, 2018) was proposed. For vehicle detection, MSVD_SPP (Kim et al., 2019) modified YOLOv3 (Redmon and Farhadi, 2018) by using five particular pyramid pooling blocks in the feature extraction net. YOLOv2 was modified by (Sang et al., 2018**)** by combining a *k*-means++ clustering algorithm to generate best-fit anchor boxes. Faster R-CNN (Wang, Ye, and Weiwen, 2019) was exploited by including multi-shape receptive field and anchor generating optimizations (Mehtab et. al., 2021).

Infrared cameras have performed better recognition for human detection because of the temperature sensing capabilities. Thermal images were employed to detect pedestrians and cyclists more efficiently in fewer visible conditions than RGB images. In the late fusion was conducted using thermal and RGB images with two parallel SSD detection streams (Li et al., 2018); Faster R-CNN was exploited to adaptively merge both modalities by a subnetwork of gated fusion (Li et al. 2019). A unified framework based on Fast R-CNN was employed for pedestrian and cyclist detection via multilevel feature fusion (Wang and Zhou, 2019).

Car and pedestrian detections have attracted a group of researchers (Song et al. 2018; Yang, Jun and Huiyun 2018). Car and pedestrian detections were conducted by using SSD architecture with MobileNet backbone subnet for a faster detection ratio (Yang, Jun and Huiyun, 2018), the visual object detection algorithms were proposed based on YOLOv2 replacing *k*-means clustering algorithm of anchor generation with a priori knowledge of objects in the database.

The recently proposed network (Condat, Alexandrina, and Abdelaziz, 2020) took

multimodal data (RGB, Depth from Stereo, Optical Flow, LIDAR Point Cloud) into consideration to detect road objects successfully. On the other hand, a lightweight network (Liu, Cao, Lasang, 2019) was proposed to detect on-road objects using limited computing resources while preserving accuracy using a modular structure. A CentreNet-based anchor-free approach (Li et al., 2020) was proffered by using Atrous Spatial Pyramid Pooling (ASPP) to extract features from multiple scales with low computational cost.

2.3.3 3D-Object Detection Methods

In the following section, 3D road object detection methods are classified based on the modalities: Image-based 3D object detection, point cloud-based 3D object detection, and fusion-based 3D object detection (Mehtab et. al., 2021).

2.3.3.1 Image-based 3D Object Detection

A spate of image-based approaches start from generating a 2D bounding box for predicting the 3D coordinates of objects using a monocular camera only (Mousavian et al., 2017; Chabot et al., 2017). In the 3D bounding box fits tightly within the 2D detection window of objects (Mousavian et al., 2017). To regress orientation loss, a novel hybrid discrete-continuous loss was proposed to discretize the orientation angle and divides it into *n* overlapping bins same as anchor boxes do for regressing bounding boxes in SSD and Faster R-CNN (Ren et al., 2017). In the proposed algorithm couldn't estimate the vehicles distances from AV up to the desired accuracy.

Based on 2D region proposal algorithms, a 3D object proposal (3DOP) (Pham, Cuong, and Jae, 2017) was proposed for 3D object detection in autonomous driving. They primarily took advantage of stereo RGB cameras to generate disparity maps. Regarding 3DOP generation, the slanted plane smoothing algorithm was utilized and was inspired by (Yamaguchi, McAllester, and Urtasun, 2014). Influenced with 3DOP (Chen et al., 2018), Mono3D (Monocular 3D) algorithm was proposed for visual object proposal.

Rather than applying a sliding window to all over the image for the region proposal Mono3D related to hand-engineered shape features, semantics, context and location priors based on a monocular camera for proposal generation. Top proposals were scored and regressed by using the Fast R-CNN network (Girshick, 2015) to determine confidence scores and 3D bounding boxes. Mono3D outperforms the results of 3DOP despite of using monocular images only.

3D Voxel Pattern (3DVP) (Xiang et al., 2015) jointly encoded the key properties of objects in the images, including appearance, 3D shape, viewpoint, occlusion and truncation, to deal with challenging conditions of visibility. 3DVP, a novel approach was proposed to represent the 3D shape of objects as a set of voxels and occlusion masks through RGB colours intensities. However, a fixed set of 3DVPs extracted during training became the bottleneck towards its generalization for random pose possibilities.

The success of Faster R-CNN (Ren et al., 2017) is not limited to 2D detection but is used frequently in many 3D detection algorithms also. RPN has remained crucial for reliable object detection. SubNet (Xiang et al., 2017) was an extended model of 3DVP that proposed a CNN to explore the classification and detection of objects at the RPN level. In this network was trained for both 2D and 3D object detection using 2D images and 3DVPs, respectively. Through 3DVPs subcategories for pedestrian, cyclist and vehicle classes, the model (Xiang et al., 2015) recovered 3D shape, pose and occlusion patterns, but limited voxel patterns became a constraint of this model.

In Deep MANTA (Chabot et al., 2017), an algorithm was proposed for 3D vehicle detection using monocular images without voxel patterns. In the second stage, a refinement operation was performed based on the proposals obtained from RPN using 3D templates. In another detection method entitled Bounding Shape SSD (BS^3D) (Gahlert et al., 2019), a novel framework was proposed for generating 3D bounding boxes for vehicle detection using 2D key points. In this model, rather than regressing the 3D coordinates of the bounding box, four visible 2D key points of the targeted object were regressed by using standard 2D detection approaches (e.g., SSD, hence named BS3D). Later, using the

camera projection matrix, 2D points were transformed into 3D bounding box coordinates.

A solution (Garanderie, Abarghouei and Breckon, 2018) was proposed to cover the blind spot in the driving scenario. In the proposed work, 360-degree round object detection was taken into account by using a panoramic view. However, LiDAR also provides a 360-degree view of the environment. Still, when LiDAR and the camera's view are integrated, LiDAR point clouds are generally trimmed to align with the camera field of view. In the proposed method, the researchers adapt KITTI, CARLA (Dosovitskiy and Ros, 2017) and Mapillary datasets (Neuhold et al. 2017) using style and projection transformations (Atapour-Abarghouei and Breckon, 2018) (because of unavailability of a panoramic labelled dataset). They estimated dense depth maps of panoramic images and adapted standard object detection methods for the equirectangular representation, and provided benchmark detection results on a synthetic dataset.

In ROI10D (Manhardt, Kehl and Gaidon, 2019), 3D object detection was proposed by using monocular images using a monocular depth network. In the proposed end-to-end paradigm-based network, 2D ROIs were lifted up based on generated depth maps. ROI10D performed a regression of all required components for the estimation of 3D boxes. However, the network remained susceptible to the distances, giving higher accuracy to the closed objects.

In M3D-RPN (Brazil and Liu, 2019), 3D detection was proposed for AV using DenseNet-121. A special 3D RPN architecture was proposed. In the network was comprised of two parallel end-to-end networks connected in the late stage to combine the results. By using stereo images (Li, Chen, and Shen, 2019) and stereo R-CNN, the network was an extension of Faster R-CNN based on proposed regions in both left and right images, in the stereo boxes, keypoints, dimensions, and viewing angles were regressed. In the extracted prior information was later combined with corresponding region-based photometric alignment to produce 3D bounding boxes. A solution (Wang et al. 2019) was proffered to overcome the depth limitations of monocular/stereo cameras. A spatial CNN was proposed to convert the depth maps into pseudo-LiDAR

representation. In this pseudo-LiDAR representation could be processed in the PointNet, the state-of-the-art 3D object detection algorithms in autonomous driving field lead to stereo-image-based approaches. Table 2.1 shows a summary of investigated Image-based 3D object detection techniques.

Table 2.1: The summary of investigated image-based 3D object detection methods

REFERENCES	APPROACHES	LIMITATIONS
(Mousavian et al., 2017)	Authors exploited the fact that the perspective projection of a 3D bounding box should fit tightly within its 2D detection window. They proposed a novel hybrid discrete-continuous loss using multi-bins for object orientation prediction.	3D bounding box projection relies on the exactness of dimensions prediction and orientation accuracy of objects.
(Pham, Cuong, and Jae, 2017)	Proposes a 3D object proposal paradigm using monocular images and depth maps for generating class independent proposals. Proposals are re-ranked again using RGB images before passing into the detection network.	Depth map calculations increased the complexity of the model, leading to slow inference speed.
(Xiang et al., 2017)	Proposes a dictionary of 3DVP by collecting possible patterns of different classes of objects and training a classifier for each pattern. SubNet (Xiang et al. 2017) introduces a multistage pyramid for feeding images	A fixed set of 3DVPs extracted during training becomes a constraint for generalizing arbitrary object poses.

	at the feature extraction level to efficiently deal with mini-sized objects.	
(Garanderie, Abarghouei, and Breckon, 2018)	Proposes 360-degree 3D object detection to cover the blind spots around AV using a panoramic view. Due to the lack of labelled panoramic datasets, researchers adapted standard datasets using style and projection transformations.	The proposed model failed when the object was too close to the camera.
(Gahlert et al., 2019)	BS3D (Gahlert et al. 2019) proposes a novel framework for generating a 3D bounding box for vehicle detection using a set of 4 visible 2D key points.	Can predict the orientation but not the exact direction of the vehicle.
(Manhardt, Kehl and Gaidon, 2019),	ROI10D: proposed monocular depth network to generate fine depth maps and raised the height of the proposed regions based on the depth map.	The solution remained susceptible to distances.
(Brazil and Liu, 2019)	M3D-RPN: Proposed use of CSPNET for feature extraction and a special 3D RPN based on 3×3 convolution to extract depth aware features using images. Generated 2D and 3D detection results.	Regarding camera limitations, M3D-RPN couldn't achieve comparable results.

(Li, Chen, and Shen, 2019)	Stereo R-CNN: Proposed Faster R-CNN based CNN streams using left and right views and extracted keypoints and dimensions of objects. In the detection network, a 3D bounding box was recovered using a region-based photometric alignment.	Car-keypoints specific techniques can be applied to the car only. Give a need for other detection methods of a different class of objects.
(Wang et al., 2019)	They proposed a CNN-based model to combat poor image-based depth estimation to convert image-based depth maps to pseudo-LiDAR representations. That can give 3D world information at the cost of the camera.	The proposed solution is not suitable in real-time taking 1-second inference time per image.

2.3.3.2 Point Clouds-Based 3D Object detection

By considering the limitations of vision sensors, point clouds data was exploited in different representations; projection-based, voxel representation and raw point clouds representation. In the following Section, each category has been discussed in brief.

(a) Projection-Based 3D Object Detection

In this project, the leverage was gained from existing, proven, and standard 2D CNN architectures by projecting the point clouds on the ground plane (Su et al., 2015), cylindrical (Zhang and Xia, 2016) or on spherical surfaces (Iandola et al. 2016) to get different projection views from AV. Later, 3D bounding boxes got recovered with position and dimension regression. In the approach, a cylindrical projection of point clouds (Zhang and Xia, 2016) was propounded (due to point clouds angular dispersion pattern around the ego vehicle) to detect road vehicles in 3D bounding box form using FCN (Shelhamer,

Long and Darrell, 2017). In the input image resulting from the projection has channels encoding the points height and horizontal distance from the LiDAR as given above in Eq. (2.1) and (2.2) in Section 2.1.2.

BEV or top-view over cylindrical and spherical view of point clouds was taken into account to generate 3D proposals. BEV presents a clearer view of the scene with no occlusion behind. BirdNet (Beltrán et al. 2018) makes use of three channels for height, intensity and density of point clouds to make BEV maps. In the approach (Yu et al. 2017), Faster R-CNN (Sun et al. 2017) was selected as a method to train the BEV maps with an additional refinement of 3D orientation detection in the detection phase. However, the model remained unsuccessful in giving good precision over high Intersection over Union (IoU) for refining regional proposals. In LMNet (Minemura et al., 2018), the frontal view of LiDAR point clouds was exploited. In the proposed network encoded point clouds data into five different channels: height, side, forward, range, and reflection to extract 3D information of objects in front of AV. In the proposed architecture was based on an end-to-end detection approach and FCN architecture.

Like the YOLO architecture, Complex-YOLO (Simon et al., 2018) aimed for high speed per frame at execution time but 3D localization. In the point clouds are converted into 3 BEV channels representing RGB maps, where R, G, and B were encoded with point clouds height, intensity, and density. In this allows Complex-YOLO to achieve a 3D detection rate of 50 frames per second while the performance remained a little inferior to previous YOLO used in 2D detection. In another BEV point clouds based approach (Feng, Rosenbaum, and Dietmayer 2018), the focus is on finding the exact confidence score considering its importance for AVs. In the confidence score estimated in every other model is generally done with softmax normalization. In the softmax function works for the sum of probabilities to unity; it does not necessarily reflect the absolute confidence in the prediction.

On the other hand, HDNet (Yang, Liang and Urtasun, 2018) proposed a new approach using high definition maps. High-Definition (HD) maps contain geometric and

semantic information with centimetre level accuracy. Usually, these maps are assisted with GPS that, in general, are used for motion planning. In the proposed work, online and offline HD maps were exploited to incorporate geometric ground information with discretized point clouds. U-Net inspired the network architecture. HDNet results show the state-of-the-art performance; however, due to LiDAR range limitations, network performance degraded after the 40-meter range. In PIXOR (Yang, Luo and Urtasun, 2018), an unconventional pixel-wise prediction approach was proposed for exploiting the BEV point clouds representation. PIXOR takes into account of the fact that all objects of interest lie on the same ground.

(b) Voxel-based Object Detection

Voxel-based 3D convolutional networks gained our attention to enhance the retention of 3D information when processing point cloud LiDAR data. In this section, we discuss the approaches where point clouds are represented in the form of a 3D grid of voxels.

VoxelNet (Zhou and Tuzel, 2017) presented a generic 3D detection framework in an end-to-end fashion. It learns discriminative features of point clouds voxels and transformed points representation of vectors into shape characterization. Special Voxel feature encoding (VFE) layers are employed to extract complex pointwise features. In the obtained features were passed into 3D convolution layers to abstract local voxel features. In the final RPN layer in the network produced results using volumetric information. In this tends to close the gap between point set feature learning and RPN for 3D detection tasks. Vote3Deep network (Engelcke et al., 2017) proposed a voting scheme based on voxel features to implement a sparse convolution matrix. It allows different sizes of the kernel in CNN for the classes of objects, all models were allowed to run in parallel. It was noted that a larger receptive field helped the model to learn significantly from the sparsity present in the point clouds representation.

3D FCN (Li, 2017) extended the previous work in 2D detection (Li, Zhang, and Xia 2016). 3D FCN transplants the standard FCN to 3D convolution operation. Point clouds

data is fed into the network in 3D voxel form to predict the confidence score and shape of the objects directly. In the output of the segmentation network predicts the ROIs and bounding box coordinates. Because of the heavy computation of the 3D convolutional network, its average running time was 1s per frame which is not suitable for run-time application in autonomous driving.

VoxelNet was refined in SECOND (Yan, Mao and Li, 2018). In the SECOND architecture introduced a sparse middle feature extractor, which gradually performed dense operations using multiple submanifold convolutional layers. Later, featured maps got transformed into 2D data such as an image. Another voxel-based SARPNet (Ye et al., 2020) was the third winner in the nuScenes 3D detection challenge of CVPR2019, workshop on Autonomous Driving (WAD). Like SECOND, SAPRNet generated point clouds voxels but overlapping with the neighbouring grids. SARPNet harnessed a fixed sampling scheme rather than random sampling, followed by previous algorithms (Yan, Mao and Li 2018; Lang et al., 2019) to optimize the results. In the proposed VFE layer in SAPRNet consists of only fully connected layers followed by batch normalization and ReLU activation to extract point-wise features.

In SegVoxelNet (Yi et al., 2020), a model was proposed for 3D vehicle detection exploiting points voxel to incorporate semantic Context information and LiDAR point cloud sparsity with respect to distance. A special BEV semantic mask was considered in the proposed work with additional depth-aware heads to learn distinctive depth-aware features. In the network was trained using the fully convolutional network.

(c) Raw Point Cloud-based Object Detection

In this section, two main architectures are detailed: PointNet (Qi et al., 2017) and PointNet++ (Qi et al., 2017) that are targeted directly by using the raw point clouds so as not to lose any relevant information in transformation. It performs feature transformation and aggregation of high dimensional local features that are learned from multi-layer perceptron (MLP) from each point using the max-pooling layer. PointNet takes use of a

Special Transformation Network to make point cloud rotation invariant. Since the MLP only has extracted the local features of each point and ignored the connections between points, PointNet fails to represent the local features of neighbouring points, thus limit its performance in complicated scenes (Liu et al., 2019).

The networks treated all points in the clouds independently (without forming any relationship) to maintain permutation invariance; however, this property neglects the geometric relationship and global features among points (Wang et al., 2019). Dynamic Graph CNN (Wang et al., 2019) proposed a simple solution called "EdgeConv" to deal with this problem. Influenced by FrustumNet, 3D object detection was proposed and named as FVNet using 3D raw point clouds based on their front view (Zhou et al., 2019).

Another approach using raw point clouds was F-ConvNet (Wang and Jia, 2019) which divided the point clouds frustum into multiple slices based on their front ranges. In the slices of spatial point clouds passed through parallel PointNets to aggregate local point-wise features. Later, these features were normalized and transformed into 2D maps to feed into FCN for estimating 3D boxes and classes of objects. In the proposed algorithm, a variant of FCMs were employed that extracted multi-resolution frustum features. F-ConvNet compared the results with the state-of-the-art networks based on the KITTI dataset. F-ConvNet also experimented with the SUN-RGBD dataset (Song, Lichtenberg, and Xiao, 2015) contained RGB-D images of 10 object categories. These depth images were transformed into point clouds to test the F-ConvNet performance. However, SUN-RGBD point clouds were not segregated, such as KITTI. Fabricated Point Clouds density remained a constraint plus that did not consider the orientation of vehicles algorithm is susceptible to 180-degree rotation.

In PointR-CNN (Shi, Wang and Li, 2019), a completely different approach was followed with conventional frustum-based detectors by using raw point clouds. Firstly, they segmented the whole point clouds into the foreground and background parts to generate high-quality 3D proposals along with semantic features. Secondly, a pooling operation was executed based on these 3D proposals in a sub-network. For better local

spatial features, understanding the processed point clouds and their semantic information was transformed into canonical coordinates. Regarding network optimization, multiple bins have been adopted for 3D bounding box regression. In the proposed method claimed to give the state-of-the-art detection accuracy based on LiDAR sensor only. However, LiDAR point cloud density remained a bottleneck towards the algorithm success.

PillarNet is a raw point clouds-based algorithm (Lang et al., 2019), a novel encoder that utilizes PointNet to learn a representation of point clouds organized in vertical columns or pillars. While the encoded features can be employed with any standard 2D convolutional detection architecture that converts the whole point clouds into multiple pillars emphasizing the vertical density of points on the horizontal *xy* grid. Based on the preserved index values of points, the extracted feature was reverted back to height and width values of image coordinate converting point features in 2D pseudo image in follow up of convolutional layer. PillarNet also leveraged the SSD detection head at the detection layer following losses used in the SECOND (Yan, Mao and Li, 2018) approach. For the benchmark, the KITTI dataset PointPillar achieved state-of-the-art precision for car and cyclist detection; however, results showed some limitations for pedestrian detection.

Table 2.2: Summary of Investigated Point Clouds based 3D Object Detection Techniques

References	Approaches	Limitations
Projection-Based Detection Methods		
(Li, Zhang, and Xia, 2016)	Cylindrical projection of point clouds is fed into FCN to detect the 3D localization of objects.	Poor localization accuracy as compared to state-of-the-art due to sparsity of point clouds.

(Beltrán et al., 2018)	BirdNet: Generates BEV maps based on height, intensity, and density of point clouds for feeding into CNN after density normalization.	Intensity information does not always give intended information, sometimes misleads.
(Simon et al., 2018)	Complex YOLO: Focuses on faster performance. BEV maps are passed into Special Euler-RPN to predict five 3D anchor boxes per grid cell; boxes are regressed to detect objects' locations.	Gives inferior accuracy compared with parallel 2D detection versions.
(Feng, Rosenbaum, and Dietmayer, 2018)	Special Bayesian Neural Network is used to predict the class and 3D bounding box after ROI pooling. Epistemic uncertainty is used to determine the uncertainty in the model in conjunction with penalizing noise.	Require multiple forward passes for uncertainty estimation that limits real-time performance.
(Minemura et al., 2018)	LMNet: FCN-based architecture using the front view of point clouds in terms of five different channels.	Unsatisfactory accuracy of results.
(Yang, Liang and Urtasun, 2018)	HDNet: HDmap with BEV point clouds. Researchers exploited U-Net to regress the results.	After 40 meters of distances, network performance starts degrading.
(Yang, Luo and Urtasun, 2018)	PIXOR: A novel approach based on pixel-wise prediction. In the input data was in the form of BEV point clouds projection. In the network was influenced by FCN using points reflectance into account.	Projection caused the loss of critical information.

Voxel-Based Detection Methods		
(Zhou and Tuzel, 2017)	VoxelNet: Feeds voxels into the VFE layer to generate point-wise concatenated features that are passed into RPN to predict 3D localization.	For every class, a specific model has to run in parallel that degrades performance over the 3D convolutional network.
(Engelcke et al., 2017)	Vote3Deep: The network proposes a voting scheme based on features to implement a sparse convolution matrix. L1 regularization and Rectified Linear Unit (ReLU) function is used to maintain the sparsity of the convolutional layers.	Fixed-size of the bounding box for each class limits precision detection.
(Li, 2017)	3D FCN transplants the standard FCN to 3D convolution operation.	Due to 3D convolution operation detection speed slows down.
(Yan, Mao and Li, 2018)	SECOND: SECOND was proposed as a sparse convolution method to deal with slow inference speed and poor performance orientation estimation. Researchers introduced SmoothL1(sin($\partial\theta$)) angle loss regression to improve the orientation estimation.	The network illustrated lower performance for pedestrian and cyclist detection.
(Ye et al., 2020)	SARPNet: Network was influenced by SECOND; however, it emphasised on the shape of objects using 3D priors based on the top view and front view of point clouds.	It Couldn't achieve comparable pedestrian detection.

(Yi et al., 2020)	Segvoxelnet: Semantic context and depth exploration using voxel features for 3D Vehicle Detection from Point Cloud.	It was tested over car data only. Shows a reasonable margin of accuracy.
Raw Point Clouds Based Detection Methods		
(Qi et al., 2017)	PointNet: Works on point cloud segments. Uses MLP that learns local features from each point. Spatial Transformation Network is used to make point cloud rotation invariant.	Generates features of independent points, thus limiting the performance without showing any relationship with the neighbourhood.
(Qi et al., 2017)	PointNet++: Construct class pyramid features on the local neighbourhood of selected point clouds by PointNet.	Slow due to sampling of the neighbourhood and running multiple PointNets parallelly.
(Wang et al., 2019)	Dynamic Graph CNN: EdgeConv captures the local geometric structure of points while maintaining permutation invariance by updating graphs at each layer of CNN.	Deals with point clouds' segments only, thus cannot handle global features.
(Wang and Jia, 2019)	FVNet: 2D proposals were later transformed into 3D frustum based on radial distance. In the proposal estimation network, final 3D regression took place.	It Couldn't compete with the state-of-the-art.
(Shi, Wang and	PointRCNN: A novel approach to segmenting point clouds into the	Performance heavily

Li, 2019)	foreground and background partitions. In the late network, 3D regression took place.	relied on point density.
(Lang et al., 2019)	Pointpillar: A novel approach to converting point clouds into point pillars for extracting unique features. These features were later transformed into a BEV pseudo image to leverage reliable CNN architecture.	Pointpillar Performance degrades with an increase in distance.

The summary of investigated point clouds-based detection methods is presented in Table-2.2.

2.3.3.3 Sensors Fusion Based Methods

We have already discussed how sensor fusion can integrate multiple modalities in different ways in Section 2.1.3. In this section, we considered the recent work done in the autonomous driving field for 3D object detection using multimodality (Mehtab et. al., 2021).

In MV3D, i.e., Multi-View 3D (Chen et al., 2017), LiDAR's point clouds BEV and front view maps with RGB images are proposed as the input. In this was a network with three parallel CNN streams with different information, extracting multi-dimensional features. In the 3D region proposals were generated from multi-channel BEV maps and fused with features extracted from all three streams. In the late section, these integrated features were passed through individual ROI pooling layers to generate fixed-sized proposals. order to produce robust results, these proposal specific features were fused in object detection with a cascading order. At the final layer, visual object classification and bounding box regression were performed. MV3D, by considering high-resolution images to deal with small-sized objects, that may result in poor performance and dense beam LiDAR results in high cost.

AVOD, i.e., Aggregate View Object Detection Network (Ku et al., 2018), was inspired by MV3D, and Faster R-CNN considered LiDAR point clouds and RGB camera images as input. As ROI pooling, AVOD preferred crop and resize operation to obtain the fix-sized proposals. In the filtered proposals further passed through fully connected layers to regress confidence score and 3D bounding boxes using "cross-entropy and smooth L_1" losses. However, one of the constraints with AVOD was the fusion of ROI features at a higher level only (Liang et al., 2019), resulting in the ignorance of special relationships occurring at low-level features.

The multistage pipeline PointFusion method (Xu, Anguelov and Jain, 2018) was employed for the late-fusion approach. PointFusion allows images to pass through CNN and raw points data into PointNet. At the final fusion stage, both streams got combined by using spatial 3D anchors. Frustum PointNet (Qi et al., 2018) leverages the established 2D detection networks and raw points based PointNet architecture for 3D detection. In this model, 2D bounding boxes were firstly obtained using a 2D detection network with their class. These 2D bounding boxes were projected on the 3D point clouds collecting all points in the extended frustum to form a frustum point cloud or frustum proposal.

In another approach (Du et al., 2018), a general pipeline was proposed for the 3D detection vehicles. An estimated 2D bounding box was projected onto the BEV point clouds, resulting in subsets of points. Based on three generalized 3D car models (i.e., SUV, SEDAN, and VAN), a generalized model-fitting algorithm was executed based on the subsets of points that filtered the car surface points in a subset removing all inside-outside points of the vehicle. These surface points were further passed through the CNN to find the confidence score and 3D bounding box of cars.

A multi-task fusion approach (Liang et al., 2018) was proposed that preferred projecting camera features information onto BEV image and fused both pieces of knowledge with the convolution layers in a 3D-based detector. In the most significant limitation of this approach remained the sparsity of LIDAR point clouds (Liang et al., 2019). In the continuous convolution approach (Liang et al., Liang et al., 2019), another

end-to-end model integrating the ground estimation module was proposed by considering the special geometry of road objects from the ground. In this approach also performed convolution level fusion; however, an extra channel to support the front view of LiDAR was included with image features that improved multimodal fusion. In the design of multi-sensor architecture performed pointwise and ROI-wise feature fusion. In the special depth completion is proposed to gain dense point-wise feature fusion.

The recent work on 3D detection (Zhu et al., 2021) utilized binocular images and corresponding LiDAR point cloud as input. In the proposed method, input images were fed into the FPN with ResNet-50 backbone architecture for features extraction. On the other hand, PoinNet++ (Qi et al., 2017) was utilized for extracting point cloud features. Point-wise feature fusion was used in four sub-abstraction layers to estimate 2D and 3D proposals jointly. To make robust and discriminative detection results, 2D and 3D proposals were fused together based on ROIs. Aggregated loss of 2D and 3D detection were utilized to optimize the model. Table 2.3 shows the summary of investigated sensors fusion-based 3D object detection techniques.

Table 2.3: The summary of investigated fusion-based 3D object detection methods

Reference	Approach	Limitation
(Chen et al., 2017)	MV3D is related to a fusion of LiDAR BEV; front view maps with RGB images to detect 3D bounding boxes using the deep fusion approach in CNN.	MV3D remained unsuccessful in detecting small objects and ambiguous in the direction of the object.
(Ku et al., 2018)	AVOD takes use of LiDAR BEV maps with RGB images using the Faster R-CNN framework fusing data in 3D anchors based on RPN and the detection network. Deploy an extra feature pyramid network as a feature extractor to deal with small objects	The proposed algorithm is based on Dense LiDAR point clouds, thereby leading to cost.

	detection problems.	
(Qi et al., 2018)	FrustumNet was influenced by 2D detection CNN results and raw point clouds based PointNet. Finds frustum proposals in point clouds based on 2D bounding box obtained from 2D detection. PointNet regressed these proposals to transform the frustum point clouds in rotation-invariant forms and finally regressed the 3D bounding box.	FrustumNet Performance is limited by 2D bounding box detection. Does not consider the whole point cloud at the same time but in segments.
(Du et al., 2018)	A generic model-fitting algorithm was proposed for matching with a subset of point clouds considering three cars templates. In the extracted surface points of the model within the subset were passed through CNN for predicting 3D detection.	Model performance remained limited by 2D bounding box detection. Three general models restrict the performance of the network.
(Xu, Anguelov and Jain, 2018)	PointFusion: Two independent streams was proposed based on images and point clouds for feature extraction. In the image and point cloud features were combined in the dense fusion network to regress the final results based on input 3D points.	Results showed a margin of improvement in the detection accuracy.
(Liang et al., 2018)	A multi-task fusion approach was proposed that preferred projecting camera features onto BEV maps and fusing both modalities information with the convolution layers in a 3D-based detector.	The network was based on dense point clouds to achieve good results, leading to expensive installation costs.
(Liang et al., 2019)	The work was a refinement of the author's previous work considering the ground distance of objects. In this network, camera	The network was based on dense point clouds to achieve good results but

	images were transformed into pseudo-LiDAR point clouds for the dense fusion of modalities information.	remained a bottleneck.
(Zhu et al., 2021)	Model exploited binocular images along with LiDAR point clouds as input sources. In the first stage in the model performed point-wise features fusion aiming to produce 3D proposals, and their dense fusion was processed in the second stage using camera image transformed pseudo-LiDAR points.	The network was based on dense point clouds to achieve good results resulting in a high hardware installation cost.

For 3D object detection, fusion detection methods lead to unimodal-based detection approaches; however, in recent years, few remarkable improvements have been depicted using LiDAR point clouds only. LiDAR and camera give complementary information together, thereby improving the performance.

2.4 Discussion

In this section, we highlight the noticeable gaps during the entire course of the literature survey. Based on the findings, we have finalized the specific areas to pay specific attention to and briefly introduced the solutions proposed to deal with the gaps found.

2.4.1 Gaps found in the Literature Reviewed

Based on the WHO 2018 road safety report, nearly half of the casualties on the road pertain to vulnerable road users. Therefore, to give reliability and assurance of autonomous driving, it becomes essential to pay great attention to detecting and protecting these vulnerable road users besides vehicles, however, the literature reviewed doesn’t give enough proof of it.

Most 2D detection work has targeted individual road objects. Consequently, multiple pipelines need to run in parallel for individual classes of interest, leading to increased operations and time complexity (Mehtab and Yan, 2022). On the other hand, the general outcome of reviewed unified detection networks results in poor pedestrian and cyclist accuracy compared to vehicle detection. An apparent reason for the legging accuracy of pedestrians and cyclists is the challenges attached to them, e.g., different poses, sizes and outfits. Discussed findings can be verified from the results achieved in Frustum ConvNet (Wang and Jia, 2019) and GFD-Retina (Condat, Rogozan, Bensrhair, 2020) for 2D road scene perception in autonomous driving. In this shows a remarkable precision in Car (≈ 88.54 %) *Vs* pedestrians and cyclists (≈ 60.43%) on the medium complexity KITTI dataset. That indicates a considerable margin to cover in pedestrians and cyclists detection accuracies.

Another significant gap is the difference between 2D and 3D detection accuracy. Despite multiple road scene challenges, 2D object detection has achieved remarkable accuracy in autonomous driving. Average Precision (AP) achieved based on KITTI

medium complexity dataset is more than 93.85% for 2D car localization whilst only 83.19% for car 3D car positioning (Zhu et al., 2020). 2D methods play a vital role in road scene perception, though a good estimate of objects distance and shape is mandatory in autonomous driving success. In the existing 2D and 3D detection gaps demand more research in 3D detection for autonomous driving (Mehtab and Yan, 2022).

By considering the latest research for 3D detection, it is proved that LiDAR is an inevitable sensor for reliable road scene perception. LiDAR directly connects with native world information that builds the base for making the next manoeuvre (Simon et al., 2019). However, the proposed 3D techniques generally rely on expensive 64 beam Velodyne LiDAR (Arnold et al., 2019). Further work is required to give robust 3D detection even with sparse LiDAR point clouds. Table 2.4, we present the comparisons of 2D and 3D object detection in autonomous driving.

Table 2.4: 2D vs 3D object detection for autonomous cars

Attributes	2D Object Detection	3D Object Detection
Positives	• Highly accurate and efficient detection architectures are developed. Exhibits good results using a monocular camera only. • Makes a strong base for further objection detection research. • Well established 2D labelled dataset is available.	• 3D bounding box provides objects' size, shape and positioning in real-world coordinates form. • The information allows better scene understanding to make a manoeuver of AV. • LiDAR gives 3D point clouds that help in estimating objects' location. A stereo camera also provides depth estimation or may be used to generate pseudo-LiDAR point clouds. • With the advent of time, increasingly 3D labelled datasets have been available for research purposes.
Negatives	• Doesn't give information about real-world coordinates of targeted objects, about their depth or occluded portion. • Not suitable in actual world driving applications.	• Model complexity increases with extra dimension regression. • 3D object detection has not achieved enough precision to apply in the actual world of autonomous driving applications successfully.

Another noticeable fact is, that most work done in 3D road scene perception does not give sufficient weightage to the specialties of LiDAR v*s* camera sensors and treats them in a similar way while performing data fusion. Additional research work is required by considering the limitations and strengths of individual sensors for doing multimodal data fusion.

By making a comparative analysis, the benchmark KITTI dataset was taken into account which is a good source of analysis considering its complexity. However, one of the noticeable limitations of the KITTI dataset is its weather conditions which are mostly sunny and daylight scenes. In the significant work has not been tested based on extreme weather conditions and night timings except a few (Ku et al., 2018). Further research work can be conducted to evaluate the effects of such conditions. To deal with this, simulators such as CARLA (Dosovitskiy and Ros, 2017) can also be used where scenes can be customized accordingly. Waymo/BDD100k datasets also claim to have included diverse weather conditions, which could also be considered while training and testing neural networks.

2.4.2 Solutions Proposed

To fill the above-mentioned gaps, a unified framework for the detection of cars, pedestrians, and cyclists is proposed for 2D road scene perception. In the proposed network is based on the end-to-end detection framework which provides a dynamic approach for changing the network shape to optimize the results based on existing hardware and database. In the network exploits a feature pyramid network to deal with multi-scale objects, CSPNet-based backbone network allows the network to learn complex features in the scene and gives high accuracy to pedestrians and cyclists detections along with cars.

In this thesis, we also deal with 3D object detection, however, due to academic

research limitations the scope is narrowed down to vehicles only. By taking the strengths and weaknesses of LiDAR and camera sensors into consideration, the proposed solution localizes the front lying vehicles in the 2D bounding boxes firstly with the help of camera images. Furthermore, the 3D point clouds of LiDAR are projected into 2D detection windows to transform the detected 2D centre of bounding into 3D world coordinates.

This research work has not relied on the dense LiDAR point clouds. In the proposed solution exploits the behavioural pattern of sparse LiDAR point clouds that always sweeps in the horizontal directions and the gaps among points lie in the vertical plane only as presented in Fig. 6.28. In the proposed solution has relied on images for estimating the dimensions and orientations of vehicles and for finding 3D centres it takes advantage of horizontal 3D point clouds stream and integrates detected 2D bounding box results with point clouds information (Mehtab et. al., 2021).

By considering the limitations of KITTI datasets, we have tested the proposed models performance on the Waymo dataset also that contains night and extreme weather scene complexities.

Chapter 3

Basics of Convolutional Neural Networks

By considering the strengths and promising outcomes of DNN from the literature reviewed, we have dug down into the convolutional neural networks to design an efficient and robust object detection algorithm for self-driving cars. In this chapter, we present the knowledge of essential blocks of the DNN with clear pictures. Popular DNN nets like VGG, ResNet and DenseNet are discussed to understand the specialties and working principles of their frameworks. Light is drawn on the 1×1 convolution network usage and transfer learning method benefits for training a CNN. In the framework of the benchmark DNN feature extraction and object detection model is also considered. In the focus of last section of this chapter especially is on standard object detection approaches by using DNN.

A DNN or ConvNet is a special class of artificial neural networks mainly applied in visual data analysis to make out patterns (Valueva et al., 2020). In the DNNs are simply neural networks that make use of convolution operation in general matrix multiplication in a few layers (Ian, Yoshua, and Aaron, 2016). Fig. 3.1 illustrates the layout of DNN consisting of an input layer, multiple hidden layers and an out layer. Hidden layers are responsible for extracting the input features using different convolutional, activation, pooling layers ad fully connected layers; in the following section, we discuss all the hidden layers in detail.

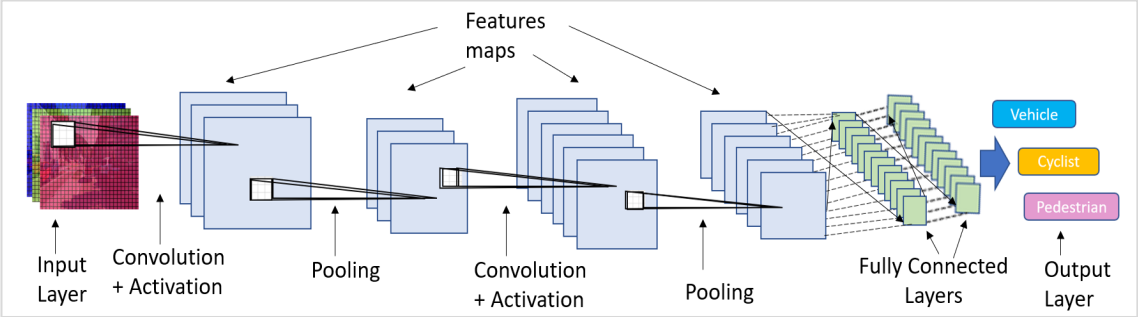


Fig. 3.1: A DNN model comprises an input layer, multiple convolution layers followed by activation, pooling layers, fully connected and output layers.

3.1 Basic Blocks of DNNs

Input Layer:

The input layer of ConvNet provides 2D images in tensor format to the neural network software environments. a coloured image, there exist three channels at the input layer.

Convolutional Layer:

As the name suggests, conventional layers perform the convolution operation on the input and find image features such as vertical edges, horizontal edges, tilted edges, corners, contours, circles, squares and other complex features using different filters. As the input

passes through each layer, features maps keep evolving to understand more abstract and complex shapes of the input provided.

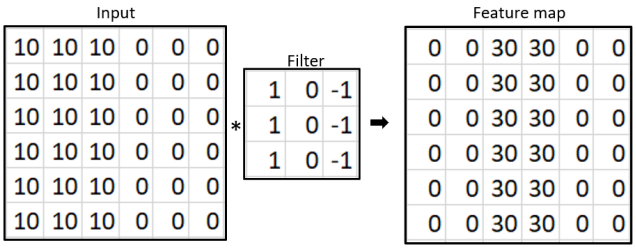


Fig. 3.2: The vertical edge detection applying convolution filter over the entire input vector

As shown in Fig. 3.2, the input convolves with the given filter to extract specific feature maps. These filter values hold learnable parameters and are tuned based on the loss that occurred during network training. In the convolution operation works in a sliding window manner over the input and shrinks the receptive field. For this reason, the very deep neural network could result in losing fine-grained features of the image. order to deal with this problem, padding zeros can be applied to the input before convolution operation (Dumoulin and Visin 2016), as shown in Fig 3.3.

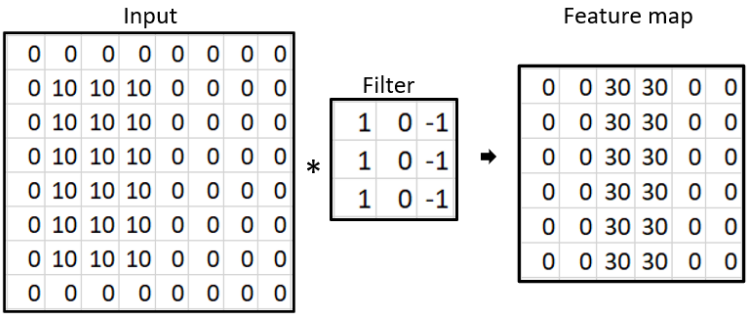


Fig. 3.3: Adding padding number before the convolution operation results in lossless features extraction

There are two options, “Valid” or “Same” for padding and no padding, respectively. Generally, the number of padding rows is determined by $(f-1)/2$, where $f \times f$ is the filter

size. Another parameter to be considered while making a convolution layer is stride '*s*' that determines how many columns to filter will slide while conducting convolution operation.

Activation Layer:

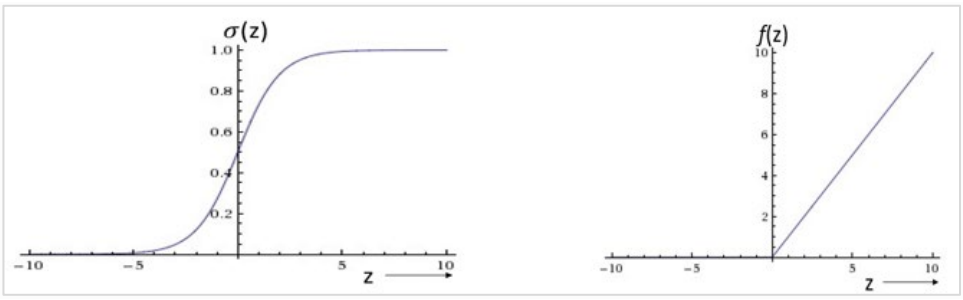


Fig. 3.4: Activation functions (a) Sigmoid activation curve (b) ReLU activation curve

However, the convolution process produces only linear outputs that are further manipulated by an activation function on the pixel basis to find non-linear/complex features of the input. If z is the input value, sigmoid function $\sigma(z) = 1/(1 + e^{-z})$ that is generally considered analogous to brain processing produces output between 0~1 as depicted in Fig. 3.4(a). However, there are two major problems with the sigmoid function. Firstly, the sigmoid saturates the large negatives or positive output gradient to zero, resulting in a vanishing gradient during backpropagation. Secondly, it is not centred at zero which indirectly generates undesirable dynamics in gradient updates.

These problems are resolved by using Rectified Linear Unit (ReLU) function, $f(z) = \max(0, x)$ that is depicted in Fig. 3.4(b). A constant gradient of ReLU results in faster learning; moreover, ReLU is computationally inexpensive compared with the sigmoid. However, in negative values of z, ReLU also produces zero gradients; this problem can be solved by using more advanced activation functions discussed in Section 4.2.2.

Pooling Layer:

In order to reduce the computational complexity, it is required to reduce the spatial dimension of feature maps. That is achieved by using pooling layers after the activation operation (Yingge, Ali, and Lee, 2020). It is a process of sample discretization that can be achieved by using max pooling or average pooling, as shown in Fig. 3.5. In the pooling window makes the sliding window moving over the output produced by the activation layer with a stride value. In the resultant value of each window represents the whole; generally, these windows are not overlapped while sliding.

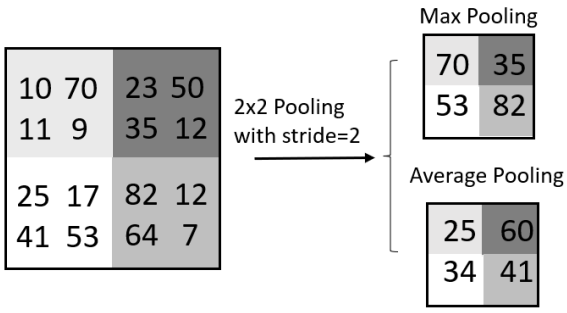


Fig. 3.5: The max-pooling picks the maximum value in that window while average-pooling finds their average

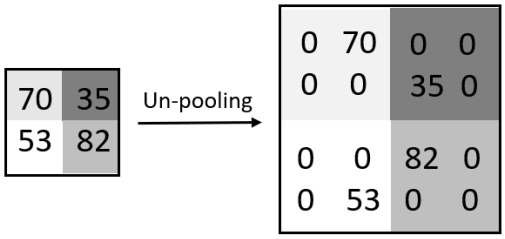


Fig. 3.6: Max un-pooling operation

However, the advanced CNN architecture includes the un-pooling operation that performs the reverse pooling function (Li, Johnson, and Yeung, 2017). In the reason to perform un-pooling is to increase the resolution of the feature maps; however, the lost information cannot be retrieved. In the state-of-the-art techniques implement pooling or

downsampling in the initial layers, in follow up perform un-pooling or up-sampling in the advanced layers. Like pooling, un-pooling can also be performed in various ways; for instance, Fig. 3.6 shows an example of the max un-pooling method (Krizhevsky, Sutskever, and Hinton, 2012; Shelhamer, Long, and Darrell, 2017), which also remembers the indexing value of the maximum number.

Deconvolutional Layer:

For the same reason as upsampling, deconvolution operation is preferred, also known as dilation or transposed convolution, in the late layers of CNN to increase the receptive field of intermediate feature maps. In the resultant features are expanded and trained, unlike un-pooling operations (Shelhamer, Long, and Darrell, 2017). Deconvolution applies the same method as convolution, albeit with some extra padding inserted in the original feature maps, as shown in Fig. 3.7. Therefore, the size of resultant receptive field becomes more significant than the original.

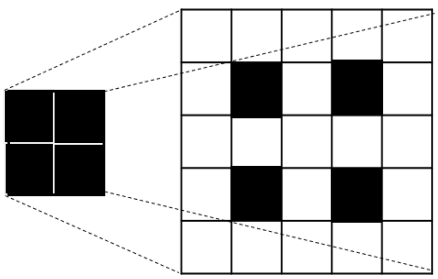


Fig. 3.7: Deconvolution operation

Fully Connected Layer:

In CNN, the last few layers are fully connected (FC) layers that work as original feed-forward neural networks. Before feeding data into the FC layers, it is flattened into a 1D array. Every input of the FC layer holds learnable weight parameters that are added with bias value at the node followed by non-linear transformation through an activation function. In the output from every node is connected to all nodes in the next FC layer.

Output Layers/Loss Layers:

The output or loss layer specifies how the training penalizes the difference between the ground truth and predicted values to minimize the cost/loss. Different loss functions can be employed depending on the task, such as the softmax function gives the probabilities of occurrences in the range *zero* to *one* that is interpreted as the scores of classes.

Mean Squared Error (MSE) estimates squared differences between predicted and ground-truth values as given in Eq (3.1) for regression purposes. Inbuilt library tools are available in PyTorch and Tensorflow libraries to perform MSE, which has been exploited in our 3D box prediction to train the network, *w.r.t*, dimensions loss as discussed in Section 5.3.

$$loss = \frac{1}{m}(square(y - \hat{y})) \tag{3.1}$$

where *m* is the batch size, $y\ and\ \hat{y}$ are the ground truth and predicted values.

On the other hand, Mean Absolute Error (MAE) calculates the means of the absolute differences between the ground truth and the predicted values as defined by Eq. (3.2).

$$loss = \frac{1}{m}(absolute(y - \hat{y})) \tag{3.2}$$

One of the most popular classification losses is cross-entropy which calculates the score of the average difference between the ground-truth and predicted probability distributions for predicting classes. In the score is minimized, a perfect cross-entropy value is *zero*. Python supports two different cross-entropy functions, namely, Binary Cross-Entropy (BCE) for binary classification loss and Categorical Cross-Entropy (CCE) for multi-class classification. Eq. (3.3) represents the CCE loss.

$$loss = -\frac{1}{N}\sum_{i=1}^{N}[y_i(\log(\hat{y_i}) + (1 - y_i)log(1 - \hat{y_i}))] \tag{3.3}$$

where *N* is the number of prediction classes, the overall loss is the average loss over all classes. At the final stage, the class with maximum probability is finalized.

In the proposed solution, a variant of BCE named "BCEWithLogitsLoss" is used which is discussed in Section 4.2.3. We have discussed the loss functions used in our experiments in Section 4.2.3. Once the feedforward pass is completed, DNN starts training its parameters based on the loss that occurred at the output layer through backpropagation traversing.

1×1 Convolution Layer:

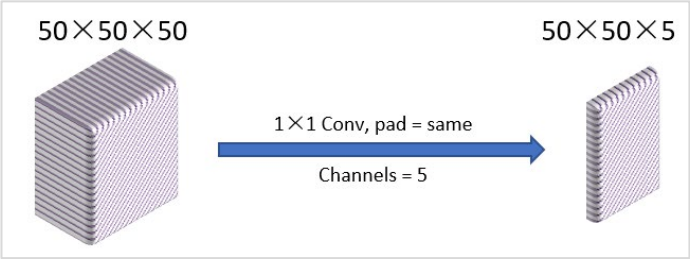


Fig. 3.8: Dimensionality reduction as a result of 1×1 convolution operation

As we know that DNN pooling layers perform downsizing operations over feature maps, it is desirable to increase the number of channels with increasing network depths to learn complex features. As a result, with increasing depth, the computational requirement of a network also increases exponentially. To address this problem, 1×1 convolution layers are used that offer channel-wise contraction (Szegedy, Vanhoucke, and Shlens, 2014), as shown in Fig. 3.8. In this simple method, generally referred to as dimensionality reduction, performs a features pooling operation; on the other hand, the same 1×1 convolution operation can also be used for increasing the number of feature maps. ReLU or other activation functions always follow these 1×1 convolutions. Another significant aspect of 1×1 convolution is projecting multiple feature maps to retain important information using an activation function.

3.2 Popular DNN Architectures

This section presents the standard DNNs and the strengths of their frameworks.

3.2.1 VGG Network

VGG (Simonyan and Zisserman, 2015) is one of the most fundamental object recognition models in DNN which was among the top finalist in ILSVRC-2014.

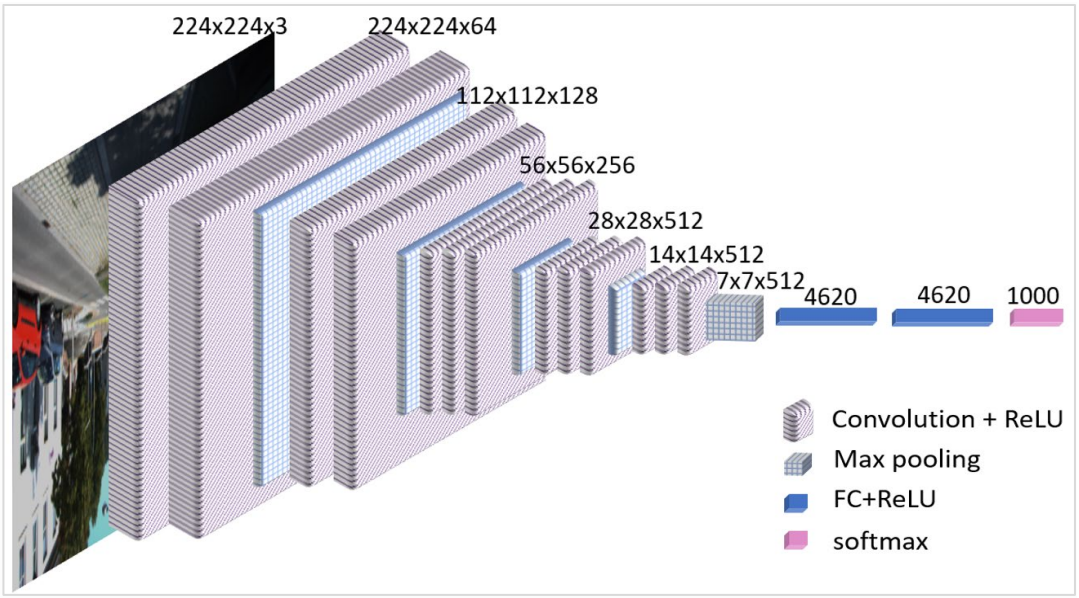


Fig. 3.9: VGG model with RGB image input and 16 hidden layers consisting of convolution layers with ReLU activation and repeatedly followed by max-pooling layers, two fully connected layers at the end with a softmax output layer.

VGG achieved 92.70% accuracy on the ImageNet dataset with 1,000 classes, demonstrating the effect of increased depth in DNN on the model accuracy. VGG took use of fixed-size 224×224 RGB images with 3×3 convolutional kernel size (i.e., the smallest possible size, which still captures left/right and up/down pixel values). VGG also exploits 1×1 convolution filters for linear transformation of the input followed by a

ReLU unit. VGG allowed fixed stride to preserve the resolution during convolution. VGG exploited three fully-connected layers at the detection stage; the first two have 4,096 channels each, the third has 1,000 channels, 1 for each class.

This network is mainly employed as a feature extractor to facilitate visual object detection or classification using DNN (Ren et al., 2017; Zhang et al., 2018; Mousavian et al., 2017). Fig. 3.9 shows the VGG-16 as a variant of deep net with sixteen layers.

3.2.2 Residual Network

Despite having given potential outcomes, DNN performance does not always lead to better features extraction with very deep neural networks as a result of information loss in convolution and pooling operations (Tan, Pang, and Le, 2020). As the size of the network grows, the existence of small objects or fine-grained features starts vanishings. ResNet (He et al. 2016) handled this problem by reformulating the sequence of the layers for learning residual features with reference to the layer inputs using skip connections as shown in Fig. 3.10.

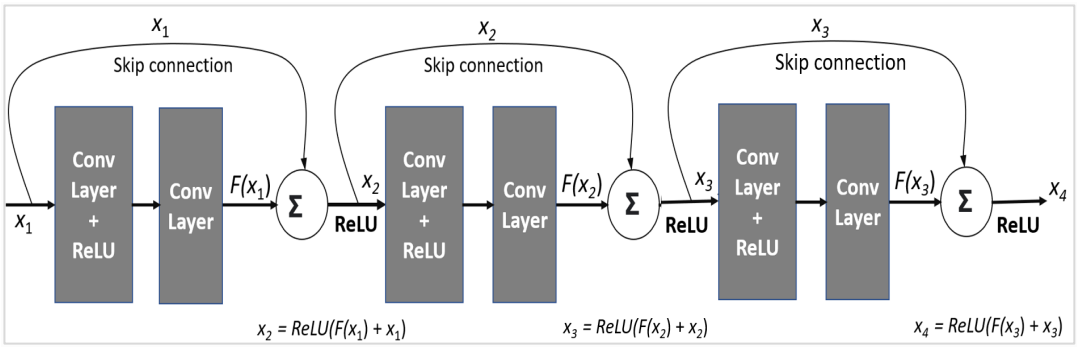


Fig. 3.10: Three residual blocks: Input features x_i maps are stacked with the successive feature $F(x_i)$ maps using element-wise addition.

ResNet adopted residual learning to every stacked layer by using Eq. (3.4) as,

$$x_{i+1} = ReLU(F(x_i) + x_i) \tag{3.4}$$

where the first term $F(x_i)$ is the weighted sum of convolutional layers to perform the non-linear transformation in a single residual block and second term x_i is the identity function that bypasses the transformation using a shortcut connection. Here, the '+' operation performs element-wise addition; however, feature maps from both the streams need to have the same dimensionality. In residual networks, shortcut connections don't introduce any extra parameters or computational complexity in the network. Based on empirical testing, it was found that using the residual connection; it is easier to optimize and achieve higher accuracy using DNNs (He et al., 2016).

3.2.3 DenseNet

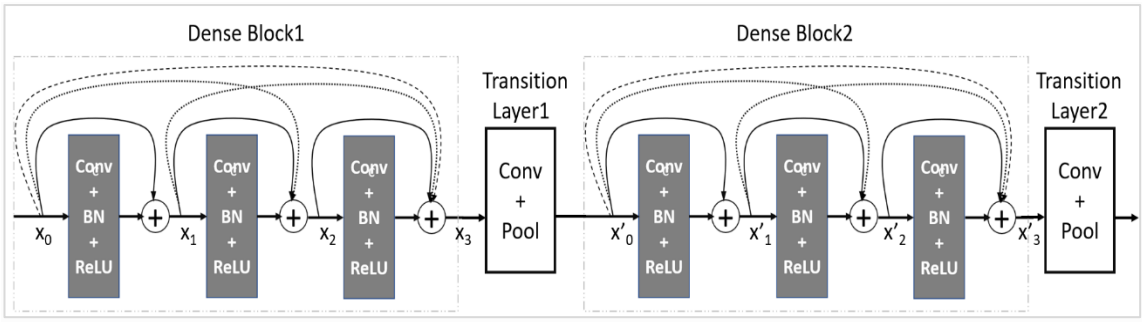


Fig. 3.11: Presentation of two dense blocks in DenseNet. Each layer in the block comprises of convolution, Batch Normalization (BN), ReLU activation functions. Output feature maps of every layer in the block are carried forward using shortcut connections to all successive layers in the local dense block. There is a transition layer between two dense blocks units.

Another advancement using skip connections was DenseNet (Huang et al. 2017) which preferred concatenation of feature maps rather than elementwise addition to carry forward residual information in the advanced layers. DenseNet carried forward feature maps generated in the intermediate layers to all their successive layers in a dense block. Fig. 3.11 represents the general architecture of DenseNet using two dense blocks, where each layer performs convolution, batch normalization and ReLU activation operation. a dense block, the i^{th} layer receives the feature maps of all previous layers, $x_0, x_1, \ldots, x_{i-1}$, that can

be represented by Eq. (3.5).

$$x_i = F(x_0, x_{1,} \ldots x_{i-1}) \tag{3.5}$$

where $(x_0, x_{1,} \ldots x_{i-1})$ refers to the concatenation of the feature maps produced in layers 0, 1, … , $i-1$. Thereby, at the l^{th} layer, when a feed-forward network holds one input from the previous layer, a dense block holds $l\cdot(l\text{-}1)/2$ inputs. However, a dense block restricts the exponential flow of redundant information at the last layer in the dense block by using the transition layer that compresses stacked feature maps using a multiple coefficient ϕ; ($0<\phi<1$). In the most significant advantages of DenseNet are dealing with the vanishing-gradient problem and efficient feature propagation.

3.3 Transfer Learning in DNN

Our objective in this research projet is to train a DNN to identify objects in images. One of the popular solutions of this is using pre-trained models through using transfer learning to accelerate the performance. Instead of training a DNN from scratch for a task, the transfer learning takes advantage of an already-trained network on another dataset. In this whole process accelerates the training speed and makes the model performance robust even with small datasets leveraging the pre-trained model with basic features abstraction efficacy. Models such as VGG-16, ResNet-50 and others are proven to give good accuracy using the ImgeNet dataset, containing 1.2 million images with 1,000 classes. There are two ways that transfer learning can be applied to train a DNN on a dataset:

(1) Load in a pre-trained DNN model trained on a large dataset. Freeze parameters (e.g., weights, biases) in the lower layers of the model for these layers correspond to 1,000 visual object classification.

(2) Replace the final deciding layers of the pre-trained model with custom layers to detect or classify with trainable parameters to model as per requirement.

(3) Train the detection and classifier layers on training data available for the task. It is

to note that number of frozen layers can also be finetuned to get the desired outcome.

Another approach is applied by replacing and retaining detection layers and training the complete model from scratch. In this approach relies only on the model framework and finetunes the weights of network during backpropagation to generate data specific features from the initial stage.

However, a combination of the two methods is also practiced and very initial layer layers that hold basic features such as edges, gradients or colour blobs are left frozen, leaving only late layers that are more specific to the details of data features are trained. Transfer learning is a practice in visual perception deep learning research for its fast and robust performance. Deep learning frameworks like TensorFlow and PyTorch hold open libraries of several pre-trained models of shared network weights.

Chapter 4

2D Road Scene Perception

This chapter covers the methodology for the robustness of 2D road scene perception using a unified DNN. In the first section, the details of the proposed end-to-end detection network are discussed with clean diagrams of all building blocks. A flexible network is proposed to find the most promising results based on the datasets and hardware resources available. In the auto-anchor generator gives custom anchor boxes with k-means clustering algorithm by using IoU features to make the network potentially generalisable based on the training dataset information. In the next section, the network performance is enhanced with various optimization tools, e.g., gradient descent optimizers, activation functions, regression loss functions, and early stopping. In the third section, evaluation methods for performance analysis of the proposed network are discussed.

4.1 Network Architecture

In Chapter 3, we have uncovered many black boxes of DNN and focused on different approaches for visual object detection. A diversity of architectures represent that the blocks can be combined, the dataflow among them can be oriented in multiple constructive ways to design a deep learning model. In the following sections, we have described the proposed unified end-to-end flexible DNN and uplifted the 2D road scene perception performance for AV.

4.1.1 FlexiNet: Flexible Neural Network

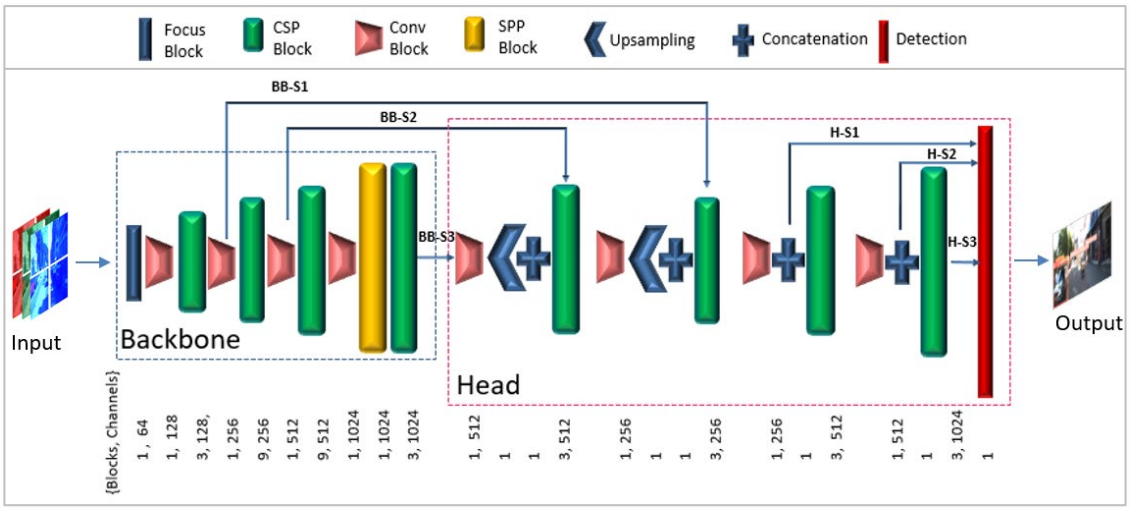


Fig. 4.1: Baseline architecture of FlexiNet network, where CSPNet is serving as a fundamental building block with dynamic scaling. Note: from (Mehtab and Yan, 2022) In Multimedia Tools and Applications

To improve the 2D road scene perception, we have investigated a DNN that provides a high level of flexibility to optimize the performance of the detection network, named FlexiNet. In the proposed architecture is influenced by YOLOv5 (Jocher et al., 2020), designed using the end-to-end detection paradigm discussed in Section 3.4.2. As shown in Fig. 4.1, the network comprises two parts with a feature extraction section called

"Backbone subnet", and another section called "Head subnet" dedicated to detection operations. However, both sections are front and end parts of the same regression stream.

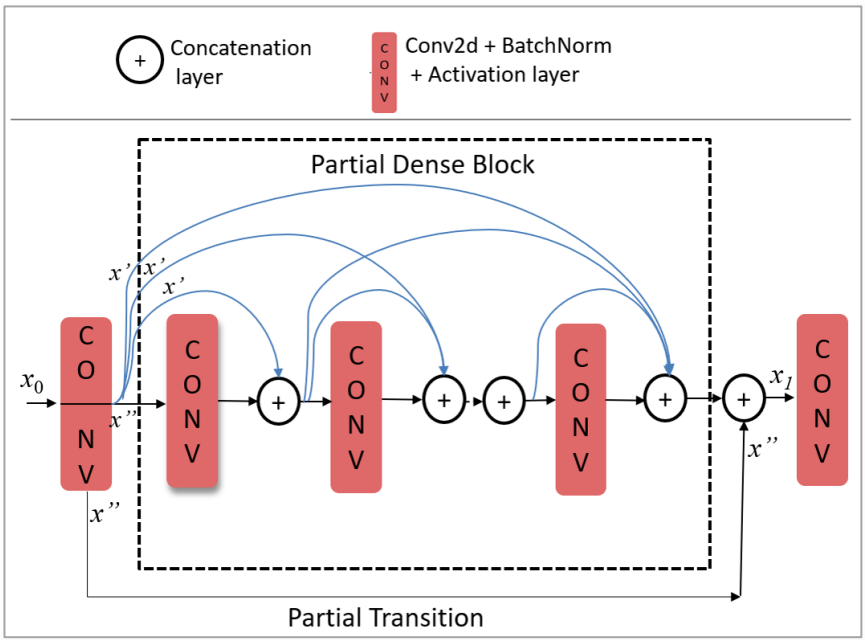


Fig. 4.2: CSPNet working as the building block of FlexiNet backbone network with dynamic scaling, Note: from (Mehtab et. al. 2021) In: *ICCCV*

The architecture and complexity of deep neural networks have shown powerful ability in image feature extraction (Krizhevsky, Sutskever, and Hinton, 2012; Simonyan and Zisserman 2015; Szegedy et al. 2015); however, it is only suitable for costly hardware components. Moreover, naively increasing the network depth results in overfitting and vanishing gradient problems (Tan, Pang, and Le, 2020; He et al., 2016). On the other hand, a wider network captures fine-grained features more precisely (Komodakis 2016). However, as shown in Fig. 6.5, our empirical results depict that going too wide also leads to decreased accuracy.

Thus, we propose a FlexiNet model that allows to dynamically define a network structure by using the depths and widths as attributes of the baseline architecture based on the available hardware resources. Although the strength of the network lies in efficient

feature extraction irrespective of its size, having a strong baseline network is of prime importance. Fig. 4.1 shows the FlexiNet baseline architecture (Mehtab and Yan 2021), the final size of the net is evaluated concerning the parameters *depth_multiple* and *width_multiple*. Eq. (4.1) represents the formation of each block in a flexible neural network based on the assigned multiples.

$$\begin{cases} Final\_layers\_in\_block = no\_of\_layers\_in\_block \times depth\_mutiple, \\ Final\_channels\_in\_block = no\_of\_channels\_in\_block \times width\_mutiple \end{cases} \quad (4.1)$$

In Section 3.2, to avoid losing residual spatial information with very deep networks, ResNet models were proposed with the skip connections (He et al. 2016), PANet was based on adaptive feature pooling (Liu et al. 2018), whereas DenseNet (Huang et al. 2017) and CSPNet (Wang et al. 2020) were proposed with cross-stage hierarchy, whereas CSPNet was influenced by DenseNet with considerable refinements and successfully applied in YOLOv4 (Bochkovskiy, Wang, and Liao, 2020).

Influenced by the performance of YOLOv4 and SocialDeep, the proposed FlexiNet model exploits CSPNet as the basic features extraction block. CSPNet has comprised of two blocks, namely, a partial dense block and a partial transition layer, as shown in Fig. 4.2. a partial dense block, the feature maps of the input layer are split into two parts through channel $x_0 = [x', x'']$, where x'' has a direct connection with the partial transition block, and x' goes through partial dense block stacking compound gradient information. In the end, the partial transmission layer breaks the compound gradient flow and concatenates the first half part x'' of the input with refined features of x_0 to generate output x_1. Mutually exclusive information of both the streams makes the gradients features strong.

In this way, CSPNet leverages feature reuse, however, truncating the gradient flow breaks the flow of excessively amount of redundant gradient information. CSPNet has been proven to converge faster with no extra storage cost (Huang et al., 2017; Bochkovskiy, Wang, and Liao, 2020). In the implementation details of a single CSPNet unit are discussed, with the details of every module implemented in the FlexiNet

architecture.

In the following sections, all components of FlexiNet are discussed with fine detailing. In addition, the auto-anchor generation method is applied to create custom anchor sizes based on the training dataset based on *k*-means clustering exploiting the IoU features to uplift network performance.

Focus Module:

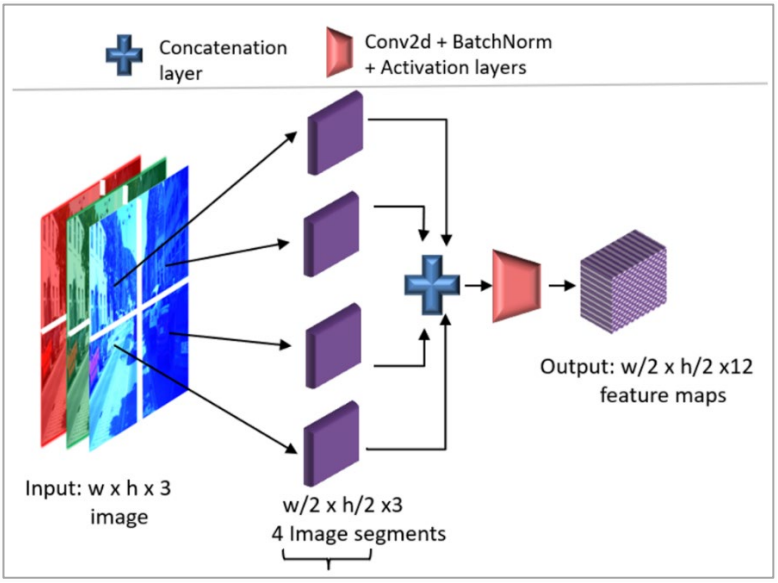


Fig. 4.3: The focus module slices the input image into four equal parts and concatenates them together in a depth-wise manner.

This module mainly targets at accelerating the fast execution and reducing the operational complexity of the advanced networking layers. In the focus module, the RGB input image of size $3\times w\times h$ is divided into four equal parts of size $3\times w/2\times h/2$ by using a slicing operation (Wang, 2018). These four parts are stacked together in the form of different channels with the help of a concatenation operation, as shown in Fig. 4.3. Subsequently, the information is passed through the ConvNet module that comprises of 1×1 convolutional layer in Section 3.5.4 followed by batch normalization in Section 3.3.8 and activation function (Elfwing, Uchibe, and Kenji, 2018). As a result, the focus block

transforms the $w \times h \times 3$ input information into $w/2 \times h/2 \times 12$ dimensions, accelerating GPU utilization.

In all successive sections, the ConvNet block has been referred to the composition of 2D convolutional, batch normalization, and activation layers. As discussed in appendix A.3.7, batch normalization stabilizes the distribution of the parameters layer by layer (over a mini-batch) for faster convergence and reparametrizes the underlying parameters imbalanced to make its gradient landscape significantly smoother (Santurkar, Tsipras, and Ilyas, 2018).

CSPNet Unit Operation in Partial DenseNet:

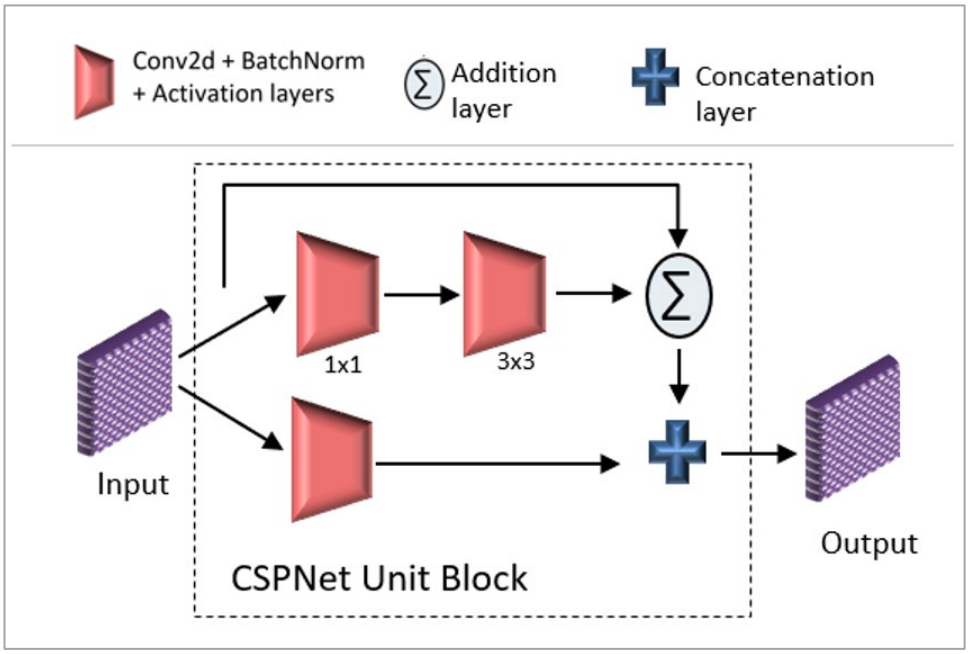


Fig. 4.4: Unit operation in CSPNet module

CSPNet is the basic building block in the FlexiNet. Fig. 4.4 represents the single operational unit of the CSPNet module with the details of every operation executed. In this module, input information flows in two streams; the first stream directs the information to the ConvNet block with dimension 1×1. In the second stream passes the

same information through the bottleneck layer with a consecutive addition performed with the original input. Hereinafter, the bottleneck module performs data compression through dimensionality reduction in the corresponding 1×1 Conv layer.

In the subsequent operation, outputs from both streams are concatenated to attain high-level gradient information. In the number of units in a CSPNet block in the FlexiNet architecture is based on the depth and width multiples assigned in execution. At the final stage of CSP, a single convolutional block called the partial transition block performs a hierarchical feature fusion mechanism, as shown in Fig. 4.2.

In each CSPNet module, the output of a k layers block is expressed as follows:

$$y = F(x_0) \Rightarrow x_k = H_k(x_{k-1}, H_{k-1}(x_{k-2}), H_{k-3}(x_{k-3}) \ldots . H_1(x_0)\, x_0) \quad (4.2)$$

where F is the mapping function from input x_0 to target k^{th} layer, which is also the model of the entire CNN. As for H_k, it is the operation function of the k^{th} layer of the CSPNet. Usually, H_k is composed of a set of convolutional layers and a non-linear activation function (Lin et al., 2017).

The architecture design of CSPNet makes the k^{th} layer pass the gradient information to all $k-1$, $k-2$,..., 1 layers and uses it to update the weights, which causes repeated redundant learning information. order to truncate the gradient flows of H_1, H_2, ..., H_k, we see intermediate ConvNet blocks among CSPNet blocks in Fig. 4.2, which work as partial transition blocks in the FlexiNet performing truncation of recurrent operation. That makes CSPNet converge faster with no extra storage costs (Wang et al., 2020; Bochkovskiy, Wang, and Liao, 2020).

Spatial Pyramid Pooling (SPP) Module:

Another important module in the backbone subnet is SPP which works against the constraint of fixed size image restriction in visual object detection (He et al., 2015). SPP was successfully adopted in many end-to-end detection architectures (Redmon, Farhadi,

2018). In this module, a max-pooling (refer: Section 3.1) operation using three kernel sizes {*5×5, 9×9* and *13×13*} is performed to extract the different levels of features abstraction as shown in Fig. 4.5. In succession, the abstracted features are arranged in a fixed-length representation with the increased receptive field using a concatenation operation.

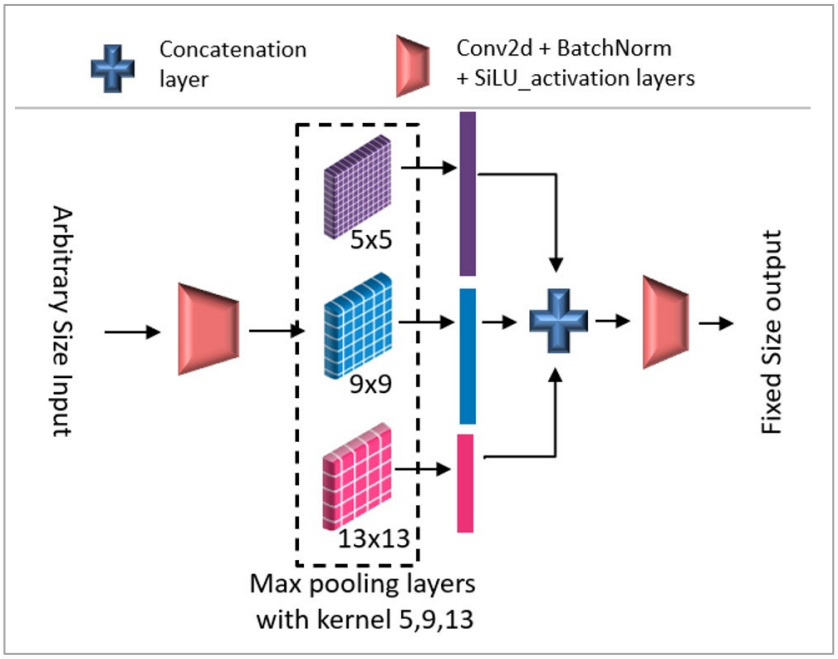


Fig. 4.5: In SPP block, three kernels are employed for pooling the same features received, the outputs are concatenated to produce a fixed-sized feature map.

This pyramid structured pooling operation of the SPP module not only improves the gradient flow in DNN but also transforms the varying receptive field information into a fixed-size format.

4.1.2 Learning with Multiscale Features

In DNN, every convolutional layer result in rich and sophisticated features maps; however, there is a continuous reduction in the receptive field. In order to combat this problem, the

proposed network takes use of Feature Pyramid Network (FPN) (Lin et al., 2017) which was exploited in YOLOv3 and the following versions. FPN employs bottom-up and top-down pathways to integrate features extracted from different levels in the form of a pyramid.

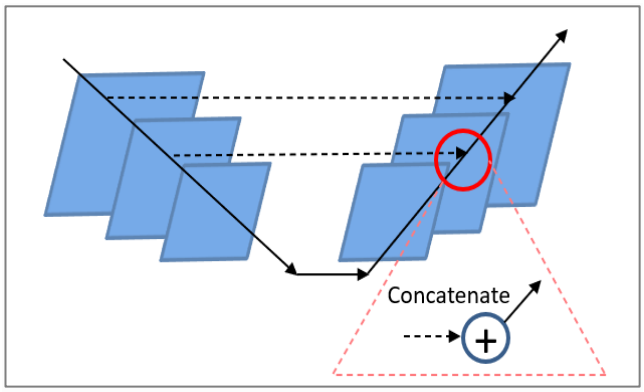


Fig. 4.6: Feature Pyramid Network block illustrating lateral connections between bottom-up and top-down pathways

In FPN, the stages are defined in the feature maps hierarchy of the network for features retrieval. As shown in Fig. 4.6, at the initial bottom-up pathway, where feature map downsampling operation is performed, hierarchical features are fetched at the end of every defined stage. On the other hand, at the top-down pathway, feature map upsampling is performed to attain spatially coarser but semantically more robust features. For multiscale detection using FPN, a bridge is formed between bottom-up and top-down pathways using skip connections of same sized feature maps, as shown in Fig. 4.6. Using concatenation operation, obtained same sized features are integrated into three hierarchical levels for final multiscale detection.

As illustrated in Fig. 4.1, the proposed solution extracts feature maps from three different stages BB-s1, BB-s2, and BB-s3, in the backbone subnet (Mehtab and Yan, 2022). These feature maps are carried forward in the head module using skip connections. In addition, the feature maps in the head section are dilated at three stages using

upsampling operation. With reference to the receptive field size, the feature maps from BB-s1, BB-s2, and BB-s3 stages are concatenated with feature maps in the head section. Furthermore, the detection layer predicts object bounding boxes at multiple scales by using anchor boxes with various sizes. Visual object detection is accomplished at three stages in the head Section, namely, H-s1, H-s2, and H-s3, targeting at visual objects with various sizes. However, multistage detection results in various outcomes of multiple bounding boxes of the same object. While using non-max suppression removes these extra boxes, we retain the ones with the highest confidence score.

4.1.3 Auto-anchor Generation

In the proposed network, the final success of the network is firmly based on the anchor boxes defined. However, the aspect ratio and size of visual objects vary as per their class and distance, respectively, as clearly visible in Fig. 4.7, the anchor size should be changed. However, manual finetuning of the anchor sizes applies a limitation to the algorithm's success, especially when the algorithm is targeting multiclass detection. Fig. 4.8 illustrates scatter plots drawn w.r.t the width and height of cars, pedestrians and cyclists present in the KITTI dataset. All visual objects have multiple aspect ratios that indicate the need for different anchor boxes for all classes in multiple scales.

Fig. 4.7: Bounding boxes drawn around cars, pedestrians, and cyclists illustrate that every class holds a different aspect ratio, and their sizes vary w.r.t distances.

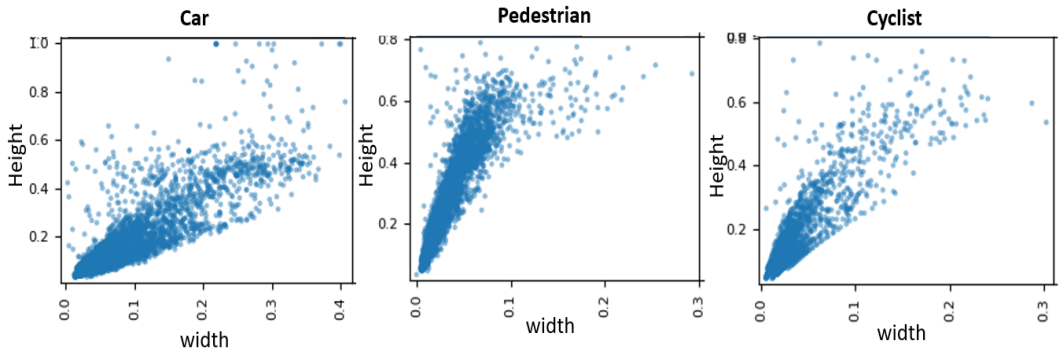


Fig. 4.8: Scatter plots drawn w.r.t widths and heights of cars, pedestrians and cyclists objects based on the KITTI dataset. All classes have different aspect ratios that indicate the need for different anchor boxes for all classes at multiple scales.

In order to give a high level of flexibility, an auto-anchor generating method is proposed that is based on *k*-means clustering algorithm, however, method is modified based on the application requirement (Mehtab et. al, 2022). In our *k*-means variant that is influenced by YOLOv2, the IoU metric is deployed in place of the Euclidian distance, which is biased towards small bounding boxes as compared to large ones and gives lower loss value for small boxes even at the significant difference in shapes as depicted in Fig. 4.9. However, the criteria of prior selection are better for IoU scores, independent on box sizes.

The proposed algorithm firstly takes account of default Anchor_size [*n*, 3] for *n* classes on three scales. In the anchor_box refinement process, every ground-truth bounding box (GT_BB) is associated with a current anchor_size based on their IoU score to form clusters. Regarding the mean of box size in each cluster, anchor_sizes are iteratively refined. In this cluster reforming and anchor_size refinement process is repeated until no change is acquired in two consecutive stages. In the pseudocode for the auto-anchor generation algorithm is drawn in Algorithm 1.

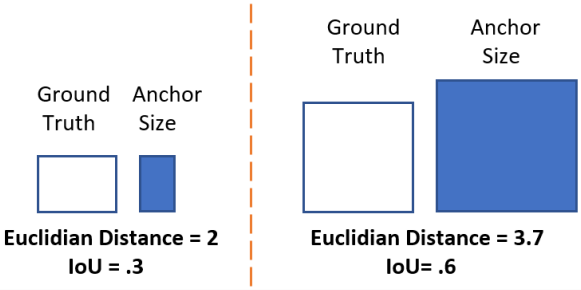


Fig. 4.9: Left side represents the small ground truth box and its prior, whereas the right illustrates the large ground truth box and its prior. Although left boxes show a major difference in shapes, their difference in terms of euclidian difference is lesser compared to right boxes, which is unexpected in finding object similarity. Thus, we propose to exploit IoU to find the loss during anchor box generation.

Algorithm 4.1: Auto-anchor generation algorithm

```
input: data GT_BBs,n
initialize Anchor_Size[n×3] with base_values
no_change = False
repeat
  # Refine n×3 clusters
  for i in n:
   for j in Gt_BBs[i]:
     associate Gt_BB[i,j] with an in AnchorSize[i] based on min(IoU)
  # Claculate mean_AnchorSize[3×k] of new clusters formed
  for i in range(n):
    for j in range (3):
      find mean_AnchorSize[i,j] based on GT_BBs in cluster[i,j]
```

```
    if Anchor_size[i] == mean_AnchorSize[i]:
        no_change = True
    else:
        Anchor_size[i] = mean_AnchorSize[i]
until no_change== True:
Output: Anchor_Size[n×3]
```

4.2 Network Optimization

The FlexiNet is perfectly aligned to innovation needs, and pipeline – optimizer function, activation function, and loss function selections are considered a significant part of the methodology for efficient neural network building. In this section, we will discuss different methods investigated to improve the performance of FlexiNet.

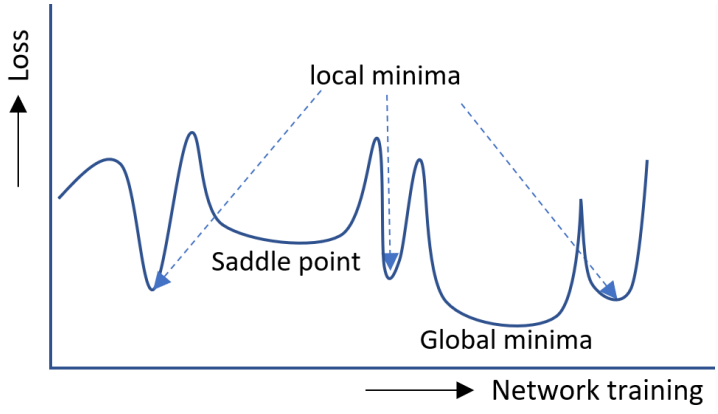


Fig. 4.10: Asymmetric gradient plot depicting wide flat surfaces and multiple local minimas encountered during the neural network optimization process

4.2.1 Optimization Functions

Optimization algorithms are responsible for changing the network parameters to reduce

the resultant losses and make network convergence fast. Based on the strategy defined in the optimization algorithm, weights, biases or other parameters are tweaked in the neural network. An optimization algorithm should be smart enough to deal with the asymmetric gradient curve, as shown in Fig 4.10, which may consist of multiple local minimas and some saddle points where the gradient is almost zero. Fast training optimizer sometimes results in poor generalization than a wide-flat minimizer; on the other hand, a saddle point may also confuse the wide flat minimizer.

It is thus of prime importance while designing a deep neural network to select the optimization algorithm that gives the optimal solution by escaping local minimas and the saddle points. However, the performance of the optimizer also gets influenced by the model to be trained on. So, empirical testing is required before fixing the final optimizer for the network. In the choice of optimizer selection becomes critically important to extract the high performance of the network. In the proposed FlexiNet architecture, we have emphasised on the investigation of the most popular Stochastic Gradient Descent (SGD) with momentum and Adam optimizers for achieving the desirable accuracy (Mehtab and Yan, 2022).

There has been a long debate on SGD *vs* Adam for the generalized output performance. Appendix A.3.3, despite the simplicity and effectiveness of SGD, it takes use of a single learning rate for all gradient coordinates and could suffer from unsatisfactory convergence results and stay on the saddle region, sometimes referred to as a wide flat optimizer. However, the additional momentum term gives flexibility in the algorithm to make the convergence stabilized and at the same time allows it to start with a high learning rate. Another contributing factor in the performance of the SGD algorithm is the mini-batch size; results obtained by (Ohzeki et al. 2018) claim that a smaller batch size helps in dealing with the saddle points as well as local minima problems effectively (Mehtab and Yan, 2022). Eq. (4.3) represents the SGD with momentum '$\propto$' using a batch size *m*.

$$\boldsymbol{\Delta\omega}_{new\_batch} = -\eta \cdot \frac{1}{m}\sum_{i=1}^{m} \frac{\boldsymbol{\partial \text{cost}}}{\boldsymbol{\partial \omega_{old\_batch}}} + \propto \cdot \boldsymbol{\Delta\omega}_{old\_batch} \quad (4.3)$$

Given optimization algorithm tweaks the network parameters $\boldsymbol{\Delta\omega}$ based on the average gradient descent in the last batch, including the momentum term $\propto \cdot \boldsymbol{\Delta\omega}_{old\_batch}$ to make stable convergence.

On the other hand, adaptive gradient learning algorithms are becoming much more popular for training deep neural networks due to the fast convergence in the initial epochs. Algorithms such as AdaGuard and Adam adjust the learning rate based on the exponentially decaying average of past gradients. Adam in Appendix A.3.6, adjusts the momentum and variance of the gradient so as to adapt to local changes in the gradient geometry. If m_t and $v_t$are the mean and variance at point *t,* final updated weights can be presented as given in Eq. (4.4).

$$\boldsymbol{\Delta\omega_{t+1}} = -\frac{\eta}{\sqrt{\widehat{v_t} + \epsilon}} . \widehat{m_t} \quad (4.4)$$

where ϵ is the summation coefficient to avoid division by zero condition. In the details of m_t, v_t terms are presented in Appendix A.3.6.

In the investigation of the best performing optimizer, we would be testing our proposed DNN on the discussed SGD with momentum and Adam optimization algorithms. We will test the results based on different sized FlexiNet architecture, with datasets of varying levels of complexity.

4.2.2 Activation Functions

In DNN architecture, convolutional layers are followed by activation layers. In the activation function triggers a neuron based on the threshold value. In the main role of the activation layer is to convert the linear output value of a neuron into a nonlinear value that enables the neural network to extract complex/nonlinear input features of images that finally leads to successful object classification, object recognition and object

segmentation results. Therefore, the selection of activation function plays a key role in the training dynamics and network performance. Based on the literature surveyed, we found novel activation functions that claim promising and consistent performance. In the network optimization phase, we empirically evaluate the most promising activation functions to raise the accuracy of the FlexiNet results.

In Section 3.1, sigmoid $\sigma(x) = 1/(1 + e^{-x})$ was the introductory activation function used in the neural network architecture; however, it results in saturations at large positive and negative neuron values, giving almost zero gradient value. Appendix A.2.3, in the backpropagation process of network training, the gradient is employed for parameter optimization; however, if the gradient is too small, it leads to a vanishing gradient problem.

Rectified Linear Unit (ReLU) (Nair and Hinton, 2010) has been a classic choice for effectively dealing with the vanishing gradient problem that also offers low computational cost. ReLU transforms the linear function by using the function max(0, x) as shown in Fig. 3.4 (b), which primarily results in better convergence and faster speed as compared to pre-existing function because of its computational simplicity. However, ReLU suffers from gradient information loss by collapsing the negative inputs to zero. A ReLU neuron comes in a dead state forever if it never reaches the negative region; it is also impossible for the neuron to recover back. Leaky ReLU (Maas, Hannun and Ng, 2013) is considered a variant of ReLU, allows a slight positive gradient "*ax"* if the unit is not active, as shown in Fig. 4.11(b). In spite of overcoming ReLU limitation, Leaky ReLU is not considered a preferred choice as it demands one more parameter, "*a*", to tune.

Recently, many advanced activation functions have been proposed based on learnable parameters that overcome the dying condition of ReLU and provide defense against vanishing gradient problem and heavy computational requirements. These activation functions have been investigated individually for the final activation selection in FlexiNet architecture to improve the performance of deep neural networks.

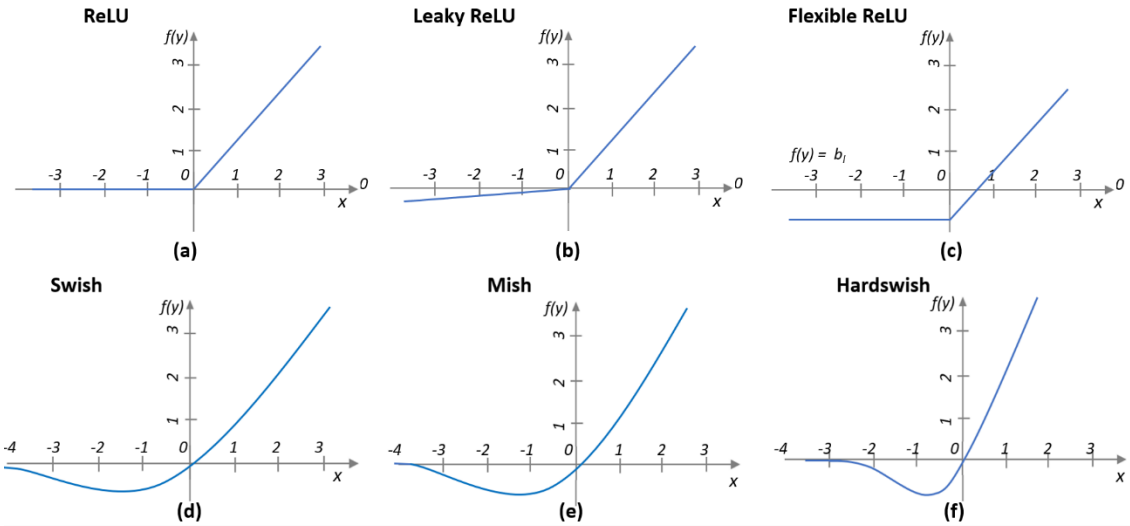


Fig. 4.11: Plot of popular activation functions such as ReLU, Leaky ReLU, Flexible ReLU, Swish, Mish and Hardswish. All monotonic functions are represented in the first row, whereas the second row illustrates the latest trend of non-monotonic activation functions.

- **Flexible ReLU (FReLU)** (Qiu and Cai, 2018) aims to overcome that dying state of ReLU with additional flexibility on horizontal and vertical axes, as shown in Fig. 4.11(c). FReLU is expressed in Eq. (4.5), where b_l is the layer-wise learnable parameter.

$$frelu(x) = \begin{cases} x + b_l & if x > 0 \\ b_l & if x \leq 0 \end{cases} \tag{4.5}$$

Although the x parameter has a hidden variable a by effective $(x+a)$ weight, where a can be trained together with the bias of the preceding convolutional/linear layers.

- **Swish** (Ramachandran, Zoph, and Le, 2017), as shown in Fig. 4.11(d), is a smooth continuous activation function, unlike previously discussed piecewise linear activation functions. In the region of the negative weight, Swish allows a slight gradient to flow without sticking the network. In the non-monotonic and smooth transition property of swish makes it increasingly important in deep neural networks.

Swish does not incorporate any input variable and allows the trainable parameter to be better tuned by activation function to maximize information propagation, making smoother gradients. It is unbounded at ceiling and bounded at floor such as ReLU, whereas smooth, non-monotonic and continuously differentiable. Swish ($x\cdot \sigma(\beta x)$) is represented by Eq. (4.6)

$$swish(x) = \frac{x}{1+e^{-\beta x}} \tag{4.6}$$

where β is a learnable parameter or a constant, in the PyTorch implementation, β is considered as 1.0.

- **Mish** activation function (Misra, 2019) was inspired by the self-gating property of swish; however, it depicts an increased gradient for the same weights compared to swish as shown in Fig. 4.11(e). Mish generates an unbounded gradient at the positive side which is a desirable property to avoid vanishing gradient or network saturation problems. Although, it is bounded below that also plays a crucial role in regularisation effects and reduces overfitting. Mish is represented by Eq. (4.7),

$$mish(x) = x \cdot tanh(ln(1 + e^{x})) \tag{4.7}$$

- **Hardswish** is a piece-wise linear analog activation function (Avenash and Viswanath, 2019) which is specially designed for quantization networks. Hardswish activation function performs $2x.Hardsigmoid(\beta x)$, represented in PyTorch in a more simplified form by using Eq. (4.8).

$$hardswish(x) = \begin{cases} 0 & if\ x \leq -3, \\ x & if\ x \leq +3, \\ x.\frac{x+3}{6} & otherwise \end{cases} \tag{4.8}$$

The activation curve of Hardswish is shown in 4.11(f). In the non-monotonic bump in the negative region is the significant difference between Hardswish and other non-monotonic functions when x is less than 0. In the negative region of the bump ($-2.5 \leq x \leq 0$), a high percentage of weights and biases falls, which leads to better convergence and

improved accuracy.

4.2.3 Loss Functions

Classification and bounding box regression are two main pillars in visual object detection field. Object localization relies on a bounding box regression principle to position the objects using rectangular bounding boxes. It aims to refine the location of a predicted bounding box. Our model takes use of various loss functions for class probability and bounding box regression, respectively. In the final loss of each iteration is the sum of both constituent losses that eventually tend to maximize the accuracy.

Class Loss Function:

Advanced Binary Cross-Entropy with Logits Loss (BCEL) is directly supported by PyTorch, which is utilized in the proposed architecture for classification and objectness score. BCEL combines binary cross-entropy and sigmoid function in a single class and is thus much more stable numerically. By integrating two functionalities together, BCEL takes advantage of the log-sum-exp trick for numerical stability and computes batch-wise loss that can be defined by using Eq.4.9 for a complete epoch:

$$\text{Loss} = \{l1, \dots, lN\}^{\top}, ln = -wn[yn \cdot \log\sigma(xn) + (1 - yn) \cdot \log(1 - \sigma(xn))], \qquad (4.9)$$

where *N* is the batch size.

Bounding Box Regression Loss:

To improve the results, bounding box regression exploits the overlapping area between the predicted and the ground truth bounding boxes, referred to as Intersection over Union (IoU). IoU plays a prime role in the final non-maximum suppression (NMS) of bounding boxes, which seems the most promising metric for box regression. However, IoU cannot give any gradient for nonoverlapping boxes as represented in Fig. 4.12, no measure to reduce the loss. An ideal optimization algorithm tends to reduce distances between the

predicted and overlapping boxes.

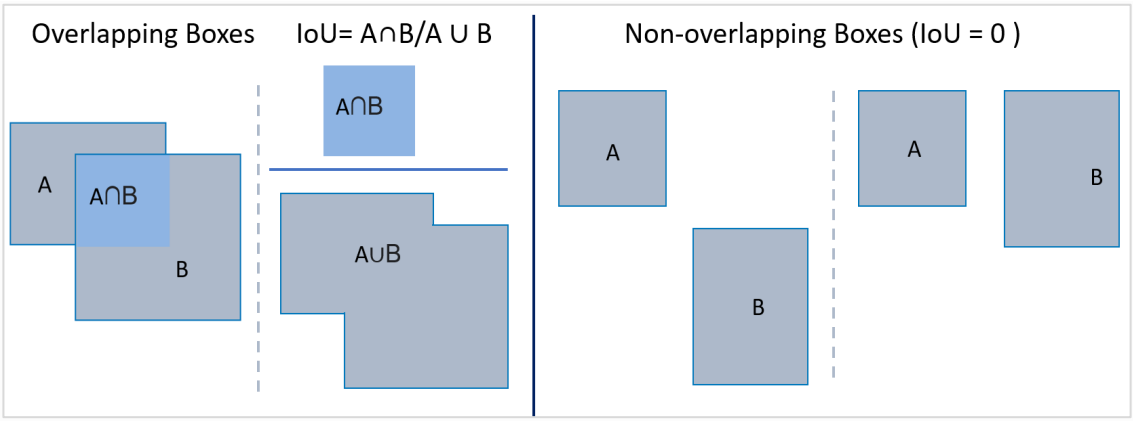


Fig. 4.12: (Left) Representation of IoU metric that only considers the overlapping region of boxes, as illustrated, (Right) For non-overlapping boxes, IoU results in zero output irrespective of the different distances between the boxes.

In recent years, IoU variants have been proposed regarding bounding box regression, namely Generalized-IoU (GIoU) (Rezatofighi et al., 2019), Distance-IoU (DIoU), and Complete-IoU (CIoU) (Zheng et al., 2020). In the proposed method, we would investigate these IoU losses for improving road scene perception accuracy. Unlike the basic IoU, the focus of these loss functions is not only on overlapping regions but also on other non-coincident regions, which better reflect the gradient between the predicted and ground truth bounding boxes. These loss functions are as follows: A and B represent ground truth and the predicted bounding boxes in the following explanations, respectively.

Generalized Intersection over Union (GIoU) (Rezatofighi et al., 2019) is the first IoU based bounding box regression algorithm that considers non-coincidental regions $\frac{|C\setminus(A\cup B)|}{|C|}$ or $\frac{|C-A\cup B|}{|C|}$ as shown in Fig. 4.13. In evaluating the loss, GIoU not only maximizes the IoU ratio between the overlapping regions, but also penalizes the loss that occurred due to the non-overlapping region. Thus, GIoU has a gradient in all possible overlapping and non-overlapping cases and makes it suitable to use as an objective function for measuring the loss in object detection. GIoU loss can be formulated as given

in Eq. (4.10).

$$\mathcal{L}_{GIoU} = 1 - \frac{|A \cap B|}{|A \cup B|} + \frac{|C \setminus (A \cup B)|}{|C|} \tag{4.10}$$

where C is the minimal closer area of bounding boxes A and B, $\frac{|C \setminus (A \cup B)|}{|C|}$ is the non-coincidental region of A and B in the coverage of C.

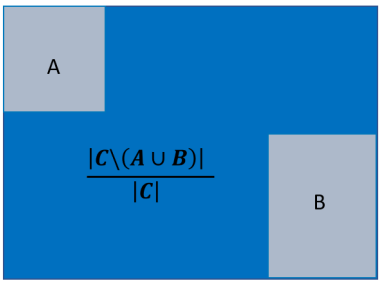


Fig. 4.13: Representation of non-coincidental region using $\frac{|C \setminus (A \cup B)|}{|C|}$, where C is the area of the smallest enclosing box covering the two bounding boxes A and B.

Distance Intersection over Union (DIoU):

(Zheng et al., 2020) is another bounding box regression loss function considered in this research project.

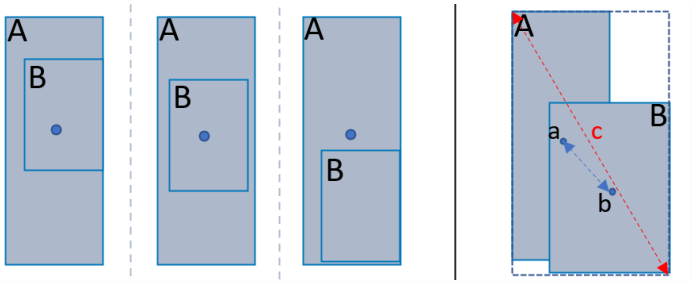


Fig 4.14: Left: All three states show the same GIoU value based on the ground truth box A and predicted box B despite the significant differences in the predicted box positions *w.r.t* the centres. Right: A and B boxes placed at a euclidian distance $\rho(a, b)$ of their

centres, c is the diagonal length of the smallest enclosing box covering the two bounding boxes. DIoU penalizes the IoU by adding a score of $\rho^2(a,b)/c^2$.

As shown in Fig. 4.14, all three states of ground truth and predicted bounding boxes give the exact value of GIoU; however, that is not desirable to achieve optimum accuracy. Due to heavily relying on the coincidental regions, GIoU primary target is to bring the boxes closer and make them overlap, as shown in Fig. 4.14, but it doesn't find any difference between concentric and non-concentric boxes. However, it is required to bring their centres closer to improve the accuracy of position estimation. In order to fix this issue, DIoU loss simply adds a penalty term on IoU loss by adding the euclidian distance term to minimize the distance between the centres of two boxes. In the DIoU regression loss function is defined as Eq. (4.11),

$$\mathcal{L}_{DIoU} = 1 - \frac{|A \cap B|}{|A \cup B|} + \frac{\rho^2(a,b)}{c^2} \tag{4.11}$$

where a and b are central points of ground truth and predicted boxes as shown in Fig. 4.14 (right), $\rho(\cdot)$ is Euclidean distance between a and b; the term c defines the diagonal length of the smallest enclosing box covering the two bounding boxes.

Complete Intersection over Union (CIoU) (Zheng et al., 2020) was considered a refinement over DIoU and has also been investigated in the proposed methodology. It considers aspect ratios of the ground truth and predicted bounding boxes as important geometric factors other than IoU and central points. CIoU loss is defined by using Eq. (4.12).

$$\mathcal{L}_{CIoU} = 1 - \frac{|A \cap B|}{|A \cup B|} + \frac{\rho^2(a,b)}{c^2} + \alpha\upsilon \tag{4.12}$$

where υ measures the consistency of the aspect ratio which is defined by Eq. (4.13).

$$\upsilon = \frac{4}{\pi^2}\left(\arctan\frac{w^A}{h^A} - \arctan\frac{w^B}{h^B}\right)^2 \tag{4.13}$$

where α is a trade-off parameter that can be presented by Eq. (4.14):

$$\alpha = \frac{v}{(1+IoU)+\partial v} \tag{4.14}$$

where ∂v is the first derivative of v *w.r.t w* and *h*. CIoU loss as represented in Eq. 4.15, all the three terms are invariant of scale and normalized in the range 0 ~ 1.

4.3 Performance Evaluation

In the experiments, the test results encapsulate precision, recall, mAP to demonstrate the capacity of the proposed model. These functions are summarized by using Eq. (4.15), Eq. (4.16) and Eq. (4.17).

$$\text{precision} = \text{true\_positive}/(\text{true\_positive} + \text{false\_positive}) \tag{4.15}$$

$$\text{recall} = \text{true\_positive}/(\text{true\_positive} + \text{false\_negative}) \tag{4.16}$$

$$\text{mAP} = \frac{1}{\text{n}} * \sum_{\text{i}=1}^{\text{n}} \text{precision}_{\text{i}} \tag{4.17}$$

where precision, recall rates and mAP are calculated based on true_positive, false_positive, and false_negative, respectively. Parameters under consideration are defined as follows, which are decided based on the IoU of the predicted bounding boxes with respect to the ground truth:

True-positives: Network identified positives which are also similarly identified positives by a human.

False-positives: Network identified positives that have been identified as negatives by humans.

False-negatives: Network identified negatives that have been identified as positives by a human.

True-negatives: Network identified negatives that have been identified as negatives

by a human.

The detection outcomes are finally arranged in the form of a confusion matrix, as shown in Fig. 4.15.

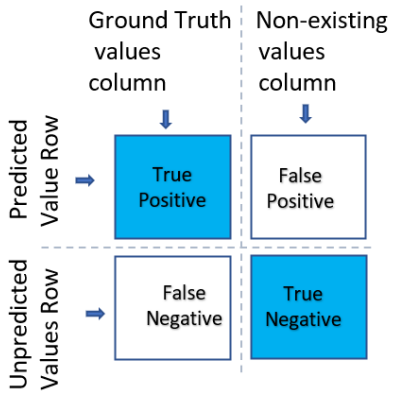


Fig 4.15: Confusion matrix of predicted results, *w.r.t* ground truth values representing true-positive, false-positive, false-negative and true-negative results

The algorithm describes the procedure applied for evaluating the performance of the FlexiNet with different metrics. As shown in Figure 4.12 (left), A is the ground truth bounding box (GT_Box) whilst B is the predicted bounding box (Pred_Box). In the IoU value, i.e., $\frac{|A \cap B|}{|A \cup B|}$ gives a measure of the overlapping area ratio of the predicted bounding box over the ground truth bounding box. In the terms clearly notify the relevance in the algorithm. Different IoU values are taken into account to generate the results. In the given algorithm finds the recall and precision for a single class of objects. However, for multiclass detection, a few of changes are made in false-positive variables based on different classes of interests. At the same time, in the calculation of performance metrics, only the false positives belonging to different classes are considered as the general standard. In the given algorithm, the IoU is set to 0.50, though we have also tested network performance over 0.20 and 0.70 IoU during the result analysis.

Algorithm 4.2: Metrics Evaluation based on the ground Truth and Predicted results

```
input: GT_Boxes, Pred_Boxes
Initialize true_positive, false_ negative, false_ positive = 0,0,0
repeat for each Gt_Boxes:
   if GT_Box.class_id in (classes_defined):
      best_box_iou = 0
      box_id = 0
      #Compare each GT_Box with all Pred_Boxes
      repeat for each pred_Boxes with index i:
          #IoU calculation
           IoU = (Pred_box ∩ Gt_Box)/(Pred_box ∪ Gt_Box)
           #Threshold check
           if IoU > 0.5:
                #Check for better prediction box
               if IoU > best_box_iou:
                   best_box_iou = IoU
                   box_id = i
      #Delete the selected bounding box from pred_boxes list
      if best_box_iou <> 0:
           delete Pred_boxes[box_id]
           true_ positive += 1       #Pred_box match with GT_Box
      else:
           false_negative += 1   #No prediction match with GT_Box
```

```
 # Now remaining items in pred_boxes are false_postive
false_ positive = len(Pred_boxes) #Extra boxes found in Pred_Boxes
   #Calculate final metrices
   precision = true_positive /(true_ positive + false_ positive)
   recall = true_ positive /(true_ positive+false_negative)
output: precision, recall
```

The benchmark KITTI dataset gives three different complexity levels for visual object detection, performance of the proposed network is compared with the state-of-the-art detection models based on these conditions.

Easy: Min. bounding box height: 40 pixels; Max. occlusion level: Fully visible; Max. truncation: 15 %

Moderate: Min. bounding box height: 25 pixels; Max. occlusion level: Partly occluded; Max. truncation: 30 %

Hard: Min. bounding box height: 25 pixels; Max. occlusion level: Difficult to see; Max. truncation: 50 %

4.4 Summary

In this chapter, a flexible deep neural network based on the YOLO framework is proposed that can be tested with various sizes to test the network efficiency based on *depth-multiple* and *width-multiple* coefficients. CSPNet has played a major role in providing fine and complex features to the multiscale detection network. We have taken leverage of different optimization, activation and loss functions to increase the accuracy of 2D road scene perception. Finally, multiple approaches for evaluating the network performance are

discussed, that would be exploited while comparing the network performance with state-of-the-art networks in the results analysis section in Chapter 6. Nevertheless, the unified architecture may perform the detection of multiple road objects in a single stream without causing additional time and computational complexity to the existing resources.

In the proposed research, the results achieved from 2D road scene perception have been further utilized for 3D object detection with the aim of getting the exact positioning of objects in the 3D world with additional point clouds information from LiDAR.

Chapter 5

3D Vehicle Detection

The content of this chapter is to answer the research question of reducing the cost of 3D detection while maintaining accuracy in autonomous driving. However, in this phase, considering the academic research limitations, the scope is narrowed down to vehicle detection exclusively. In the proposed model is based on the fact that the 2D center estimation of an object is nothing but the projection of a real-world 3D center on an image. We propose a simple yet effective approach that is based on the success of 2D vehicle detection to estimate the 3D positions of cars in front of an AV. A lightweight MobileNetV2-based DNN architecture is leveraged to predict 3D box size and orientation of cars. 3D point clouds are projected on 2D bounding boxes to map 2D car centres to 3D world coordinates using basic trigonometry. In the proposed solution exploits top-mounted LiDAR point clouds to combat occlusion conditions. In the model fulfils the proposed objective to give a cost-effective solution for 3D vehicle detection using sparse point clouds and RGB images from digital cameras.

In autonomous driving, it is essential to know the exact distance and 3D shape of front lying vehicles to avoid accidents and make a successful manoeuvre of AVs. As the literature survey, LiDAR in combination with camera sensors has been a dominant choice of researchers in 3D vehicle detection in the autonomous driving field for the complementary nature. While the camera efficiently gives us the rich textured and strong visual perception of the world in the form of image coordinates, it is susceptible to lighting conditions. In contrast, LiDAR provides us active information about visual objects surroundings AV without getting influenced by night or sun shining; however, its rays become sparse over long distances and cannot be relied on entirely. Following the same trend to achieve the best possible outcome, we have also relied on the camera and LiDAR in combination, so as to achieve optimum accuracy cost-effectively and promote the AVs acceptance in the market.

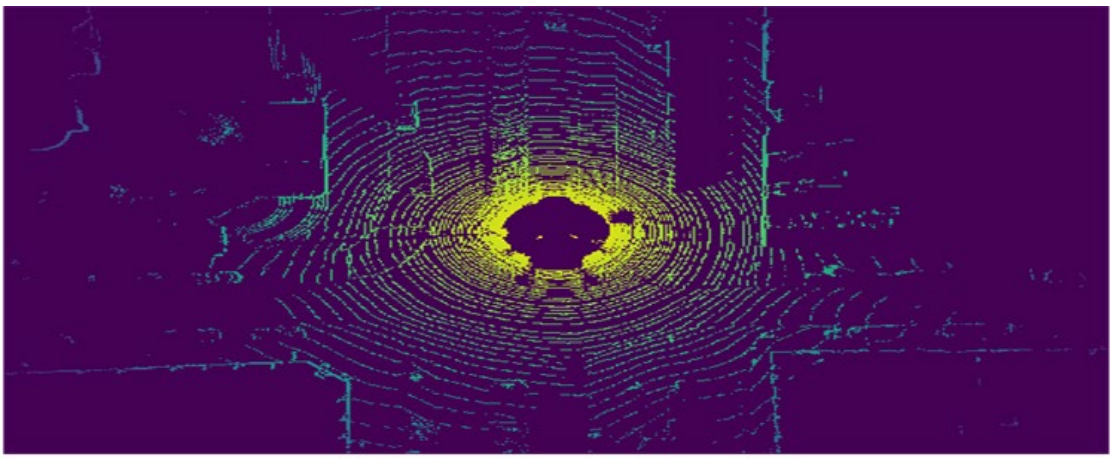


Fig. 5.1: 3D point cloud generating bird eye view (BEV) from LiDAR point clouds installed at the top of an autonomous vehicle

We are much familiar with the RGB camera and its data representation in pixel format. Let us review the working principle of LiDAR briefly before starting the discussion of 3D vehicle detection. LiDAR, i.e., Light Detection and Ranging device, is essentially a distance measuring equipment that detects the distance of the surrounding

objects by using round trip time of emitted laser beams. When this process is repeated multiple times per second, it creates a precise, real-time 3D point clouds map of the environment, as shown in Fig. 5.1.

An onboard software application can utilize a point clouds map for safe navigation. In the density of LiDAR point clouds depends on the number of rays in the beam emitted, which could be 8,16,32,64, or 128 laser rays together. In the cost of LiDAR varies immensely depending on the number of rays the beam consists of. LiDAR is use of low-intensity laser pulses that are safe for the human eyes, which allow LiDAR to know the distance to an object to within a few centimetres, up to 250 meters using the latest LiDAR technology. That's the reason why it has been the favourite choice in the field of AV. In the strength of LiDAR is measured in terms of beam density and its range; however, the relative cost of LiDAR in AVs remains a bottleneck towards its successful usage.

5.1 3D Dataset Description

Pertaining to implementing the 3D vehicle detection proposal, we have again preferred the public KITTI dataset because it provides accurate 3D ground truth information with Velodyne 64 beam LiDAR as well as the images from left and right cameras (Geiger et al., 2013). If a LiDAR is mounted at the top of a car, it is able to capture more surrounding information that is occluded from the front view. Fig. 5.2 shows an example of camera view and LiDAR for the same scene with the labelling information.

Regarding 3D vehicle detection research, the KITTI dataset has been employed as the benchmark based on the investigation. However, a thorough understanding of calibration files is mandatory to use 3D point clouds and camera coordinates together. We aim at camera and LiDAR fusion in this method; as a result, LiDAR coordinates to camera coordinates and camera coordinates to LiDAR coordinate conversion are required at many stages. Fig. 5.3 represents the different coordinates systems in the dataset.

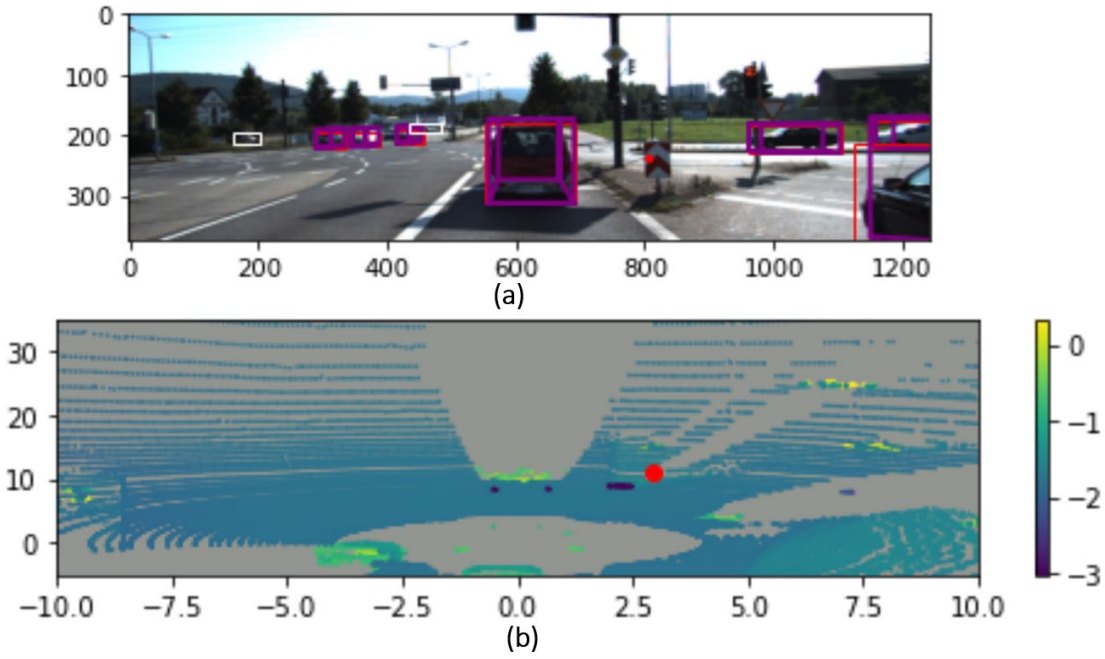


Fig. 5.2: (a) A KITTI dataset sample image shows 3D and 2D labelling of vehicle objects with a specially selected red solid point. (b) LiDAR point clouds for the exact figure with red solid point spotting same camera coordinate on LiDAR coordinates.

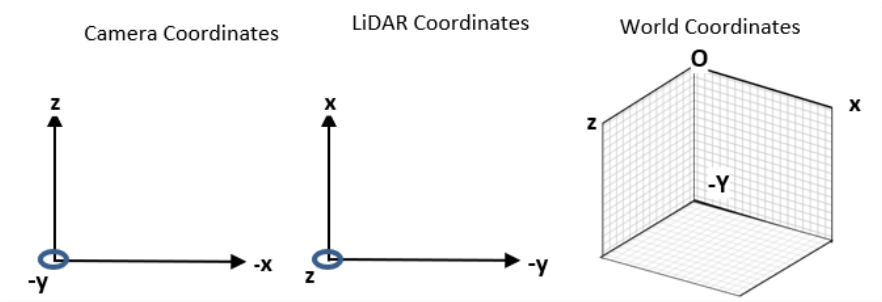


Fig 5.3 : (a) Camera coordinates system (b) LiDAR coordinates system (c) Object coordinates system that depicts real-world rotation at different axes.

We have utilized four different types of files for any single frame from the KITTI 3D Objection Detection dataset as follows:

- camera_2 image (.png),

- camera_2 label (.txt),
- calibration (.txt),
- velodyne point cloud (.bin).

The image files are .png files that can be displayed. In the label files contain the bounding box for objects in 2D and 3D in text form, as listed in Table 4.1. For 3D bounding boxes, height, width, and length of each object are provided with 3D centre coordinates. In the size (height, weight, and length) is in the world coordinate, and the centre on the bounding box is in the image coordinate. LiDAR point clouds give x, y, and z coordinates information in the form of a 3D array.

The calibration file contains six projection matrices for different sensors used— P_0, P_1, P_2, R_0_rect, *Tr_velo_to_cam*, and *Tr_imu_to_velo*. There are left, right, and reference cameras along with a Velodyne LiDAR, all sensors are synchronized with the reference camera. KITTI considers Camera_0 as a reference for the other right and left cameras (named as Camera_1, Camera_2) and LiDAR. While working with multiple sensors, a rectification process is required to integrate information with different modalities. Hereinafter, every *Px* matrices give mapping coefficients for projecting a point to *camera_x* image coordinate from rectified camera coordinate. R_0_rect is the rectifying rotation matrix to map a point from world coordinate to reference coordinate and *Tr_velo_to_cam* matrix maps a point in point cloud coordinate to reference coordinate; an inverse operation works for mapping camera to Velodyne. *Tr_velo_to_cam* matrix is the composition of rotation and translation matrices from the Velodyne to the reference camera coordinate.

5.2 Methodology

It is the simple discoveries that make the most significant differences. In this part of the project, the focus is made on the inherent key component between 2D and 3D vehicle

detection to minimize the gap between the two in the field of autonomous driving (Mehtab et. al., 2021). In the prime fact is to acknowledge that the 2D centres of the predicted vehicles are the projection of their 3D centres on an image. In the proposed 3D detection work takes leverage of well developed 2D detection for improving 3D detection accuracy. By considering the visibility limitation of cameras, the solution doesn't support camera-based 3D detection, moreover encourages to use LiDAR-based active world information that is not susceptible to lighting conditions. However, the most recent research in 3D vehicle detection is based on dense 64 beams of LiDAR point clouds to show their performances. As a result, the cost of LiDAR remains a bottleneck in the practical acceptance of AVs. In the proposed model aims to give consistent performance over sparse point clouds, a low-cost solution for the 3D vehicle detection in AVs.

The model extracts 3D world coordinates of cars in 2D detection windows of image planes using LiDAR point clouds. Tensorflow-based platform is exploited for designing the proposed neural network. In the model first regresses the size and orientations of 3D bounding boxes of cars using MobileNetV2-based DNN with a transformation in the detection extremities. For better accuracy, the network is trained from scratch based on the custom dataset. Secondly, LiDAR point clouds are employed to extract 3D centre coordinates of vehicles based on their 2D centres in image coordinates.

Although the KITTI dataset is exploited based on the availability and benchmarking criteria, existing point clouds are transformed into three different formats of 64, 32 and 16 beam density for performance analysis with point clouds sparsity. In the results are further verified by using a Waymo dataset to show the model behaviour with extreme weather scenes. In the ranging is the biggest hurdle while working with LiDAR; therefore, the results are analysed, *w.r.t* distances to get a better understanding of the results.

5.3 Estimation of Size and Orientation for 3D Bounding Boxes

Our aim in 3D object detection of cars is not only to solve the problem of correctly predicting the 3D coordinates of bounding boxes but also to give a real-time performance. MobileNetV2 (Sandler et al., 2018) is 11.7 times smaller in size than VGG16 net with comparable performance, which makes it worth using in mobile devices with low computational cost and high speed for image features extraction purposes (Sandler et al. 2018).

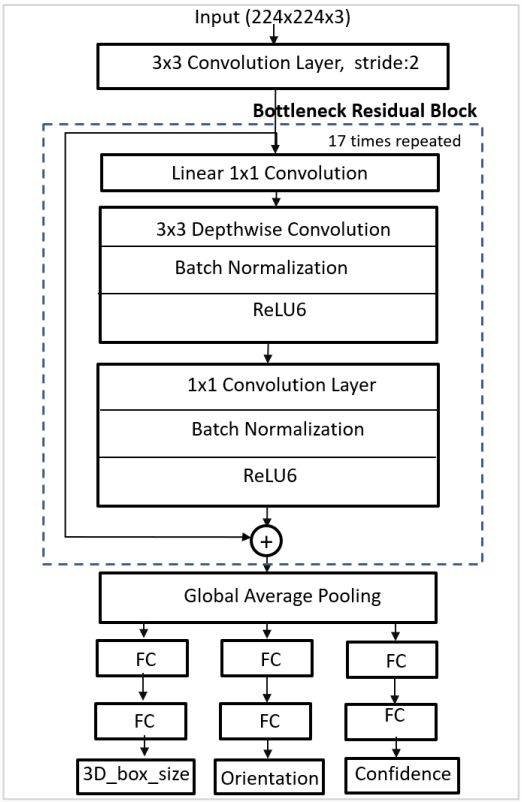


Fig.5.5: The proposed network architecture to predict the size, orientation, and confidence of the 3D bounding box based MobileNetV2 as a features extractor. Note: from (Mehtab et. al., 2021) In: *ICVNZ*

In MobileNetV2, the regular convolutional operation is replaced with depthwise separable convolutions to make the architecture low dimensional. In the basic idea is to use

a factorized version of a convolutional layer that divides the convolutional layer into two separate layers. In the first layer that performs depth-wise separable convolution does lightweight filtering by applying a single convolutional filter per input channel. a standard convolutional layer, where, to transform $h_i \times w_i \times d_i$ channels into $h_i \times w_i \times d_j$ with $k \times k$ kernel, it costs $h_i \cdot w_i \cdot d_i \cdot k \cdot k \cdot d_i \cdot d_j$ operations, MobileNetV2 performs the same in much reduced $h_i \cdot w_i \cdot d_i \cdot (k \cdot k + d_j)$ operations. In the follow-up, the second layer is a 1×1 convolution, which takes the responsibility of extracting features by performing a linear combination of all input channels. At the same time, 1×1 convolution performs a bottleneck function to reduce the number of channels. MobileNetV2 performs the same function as the convolutional layer, however splitting into two lightweight layers with reduced dimensionality.

Based on the analysis of the empirical results (Sandler et al., 2018), it was found that using linear layers is crucial in CNN performance as nonlinearity produced by the activation function destroys the high dimensional information. By considering this insight, MobileNet inserts a linear convolution layer bottleneck block, as shown in Fig 5.5. In the bottleneck blocks in MobileNet appear the same as in the residual nets, where an intermediate 1×1 convolution layer reduces the dimensionality which is followed by an expansion. MobilNetV2 applied shortcuts between the bottlenecks to improve the gradient features and carries forward residual information of gradient in the successive layers. However, the bottleneck block performs an inverted residual operation that is considerably more memory efficient as well as showed better results in the empirical testing.

One interesting property of the architecture is its design, input and output passes through repetitive bottleneck blocks and a linear convolution layer that plays a vital role in the bottleneck block. These features influenced us to study the MobilNetV2. Moreover,

the depthwise separable convolution makes an interestingly important network to research and evaluate its capacity.

Fig. 5.5 shows the proposed DNN architecture consisting of 17 bottleneck modules followed by a global average pooling layer (Mehtab et. al., 2021). It is to note that the number of input and output channels in every bottleneck module are fixed as per the MobileNetV2 convention. However, the last layers of MobileNet are replaced with three detection branches of specific functions based on fully connected layers.

The first branch is responsible for the estimation of 3D bounding box size using mean squared error. In the second branch conducts orientation prediction using L_2 loss, as discussed in Section-3.4, it plays a crucial role in final vehicle position detection. In the third branch regresses the confidence of car orientations using the softmax function. In the literature reviewed, it is found that the orientation information of cars is lost in general (Ku et al., 2018). Our architecture remedies this problem by considering two proposals in the intervals of (0, -179-degree) and (1, 180-degree) to predict the car orientation and confidence score; the one with the highest confidence score is selected (Mehtab et. al., 2021). In the net loss of the network is calculated using the weighted sum of all branches as given in Eq. (5.1), where $\alpha \, and \, \beta$ are multiple coefficients of orientation and confidence loss, respectively.

$$L_{net} = L_{size} + \alpha \cdot L_{orient} + \beta \cdot L_{conf.} \tag{5.1}$$

5.4 Estimation of Centre Coordinates of 3D Bounding Boxes

We estimate the positions of cars in the 3D world by using LiDAR point clouds in the 2D box windows. By considering the LiDAR height from the ground, firstly, we removed the ground points from the point clouds. Point clouds were projected onto the camera coordinate by using the calibration parameters while preserving the depth information in the form of added channel.

To convert 3D point $X = (x, y, z)^T$ into corresponding camera coordinate $Y = (p, q,$

$r)^T$, the operations such as translation, rotation and projection are summarized as Affine transformation (Weisstein, 2004) in Eq.(5.2).

$$Y = P_{rect} \cdot RT \cdot X, \tag{5.2}$$

Where,

$$P_{rect} = \begin{pmatrix} f_u & 0 & c_u & -f_u b_x \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \; RT = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

where (f_u, f_v) and (c_u, c_v) denote focal length and optical parameters of the camera across the x-y axes, respectively, b_x stands for the baseline, *w.r.t* the reference camera (Geiger et al., 2013). $In\ RT$, r_{ij} represent rotation parameters and (t_x, t_y, t_z) is translation across the *x, y* and *z* axes. Furthermore, *Y* is converted to 2D image coordinate (u, v) as presented in Eq. (5.3),

$$\begin{cases} u = p/r \\ v = q/r \end{cases} \tag{5.3}$$

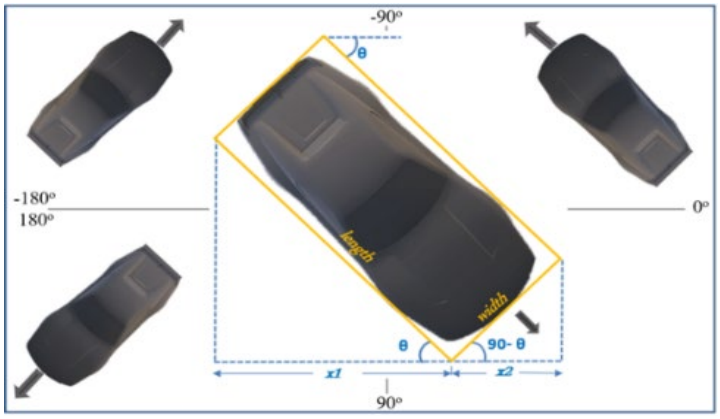


Fig. 5.6: The pose estimation of the size and orientation of 3D bounding box of cars, car pose of longitudinal or front/back sides depends on its orientation and size as displayed. Note: from (Mehtab et. al., 2021) In: *ICVNZ*

As shown in Fig. 5.6, we detect the poses of cars by using the predicted size and

orientation of 3D bounding boxes. In the case of the central car shown, if $x_1>(x_1+x_2)/2$, then the car pose is a longitudinal side, or else it poses from the frontal side, the same principle is applied to all directions. Fig. 5.6, the cars heading in different, forward directions are illustrated. In the blue dots represent the predicted 2D centre (t_x, t_y) of the cars, whereas the yellow dots stand for the outermost 3D point (l_x, l_y, l_z) across the 2D centre vertical axis. Finally, (t_x, t_y, l_z) is considered the projected centre point on the 3D bounding box surface; however, in order to get the exact depth estimation, we need to go further.

Fig. 5.7: The blue dots represent 2D centres of predicted 2D bounding boxes whilst the yellow dots refer to the 3D outermost point on the central vertical axis on the car's surface. In the right arrow shows the reference direction. Note: from (Mehtab et. al., 2021) In: *ICVNZ*

Fig. 5.8 shows the estimation of 3D car centres through 2D bounding box centres. In the depth value of the 3D centre from the surface point is based on estimated pose, orientation, and size of 3D boxes. In the figure illustrates the 2D centres (blue circle with yellow border line) projection on 3D centre (red circle) of cars for different poses by using the predicted size of the 3D bounding box and orientation, *w.r.t* the reference direction. In the trigonometric geometry for calculating car centre depth is given in Table 5.1, the final distance estimation of the car from AV is deduced by using Eq. (5.4).

$$t_z = l_z + abs(depth) \tag{5.4}$$

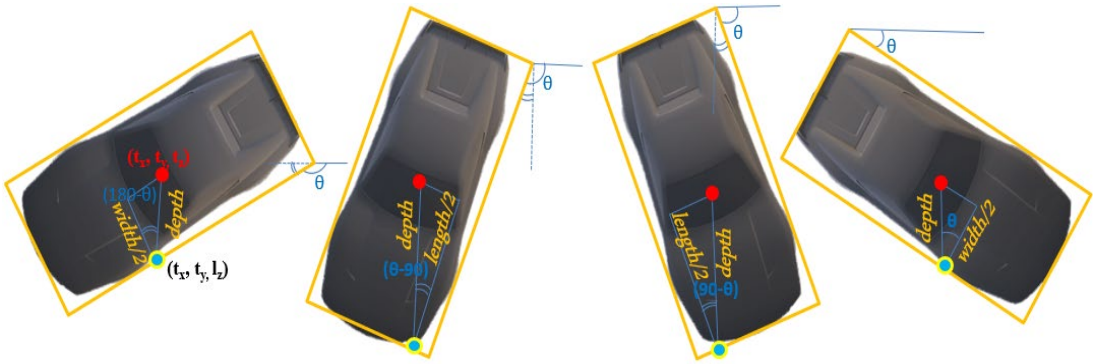


Fig. 5.8: The top view of cars oriented in different directions with the 3D centres represented in red circles. Note: from (Mehtab et. al., 2021) In: *ICVNZ*

Table 5.1: The calculation of depth estimation of 3D car centres based on the orientation, size and pose of 3D bounding boxes predicted.

Orientation	Pose	Depth calculations
	Longitudinal Side	$\cos(\theta) = \frac{width/2}{depth} \Rightarrow \text{depth} = \text{width}/2\cos(\theta)$
	Front/Back	$\cos(90-\theta) = \frac{length/2}{depth} \Rightarrow \text{depth} = \text{length}/2\sin(\theta)$
Orientation	Front/Back	$\cos(\theta-90) = \frac{length/2}{depth} \Rightarrow \text{depth} = -\text{length}/2\sin(\theta)$
	Longitudinal Side	$\cos(180-\theta) = \frac{width/2}{depth} \Rightarrow \text{depth} = -\text{width}/2\cos(\theta)$

5.5 Occlusion Handling

In a great deal of road scenarios, there exists a high degree of occlusion among cars that are tackled upon an extension using LiDAR point clouds if the LiDAR is mounted at the top of AV. Fig. 5.9 (a), we see the detailed information of point clouds retrieved; it shows the occluded cars more clearly than the camera view. By considering the fact, in the proposed algorithm, we firstly identify point clusters in each 2D detection window based on depth difference among points after removing outliers. We find the closest point of each cluster from AV. Furthermore, we arrange all cars' indexing in ascending order of closest points and identify the immediate front cars for all the occluded ones.

(a)

(b)

Fig. 5.9: A sample frame from the KITTI dataset. (a) BEV point clouds after ground points removal. (b) Image of the same frame. Occluded cars that cannot be seen in images are detectable in point clouds.

Hereinafter, we have considered the occluded cars only to the ones whose centres

are hidden behind others as we cannot obtain the direct laser distance of their centre points on the surface of the cars. In order to calculate the 3D centre of occluded cars, the gap of the closest points between the immediate front car and the car under consideration is utilized to calculate relative distancing. Fig. 5.10 shows the complete flowchart of the proposed algorithm for finding the 3D box information of fully visible as well as occluded cars in road scene perception. To give a precise idea about the working of the algorithm, the pseudocode is presented.

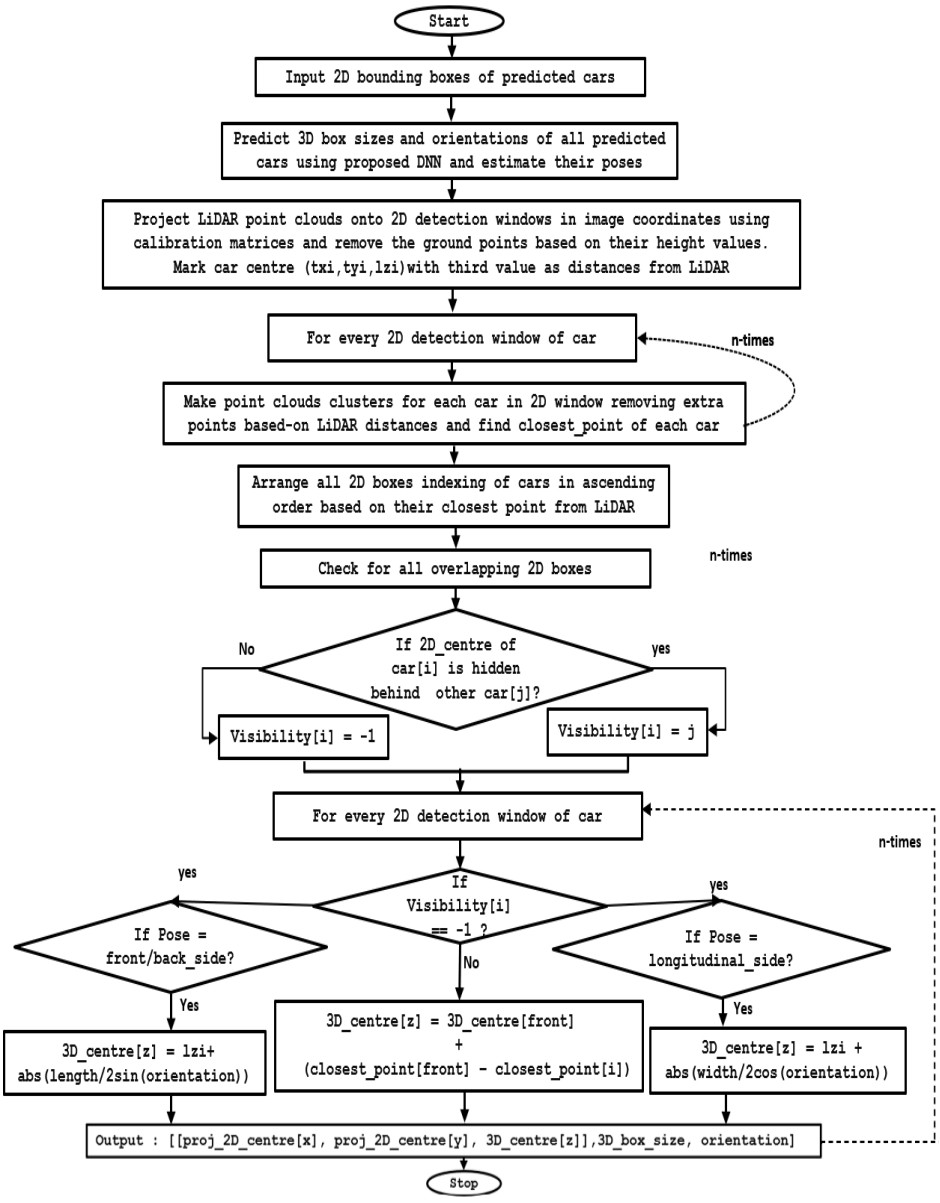


Fig. 5.10: The flowchart of the proposed algorithm for finding the 3D Box information of visible or partly visible cars based on camera RGB images and LiDAR point clouds

---

Algorithm: 3D Vehicle detection exploiting 2D detection results with LiDAR point clouds

---

1. **Input:** 2D bounding boxes of predicted vehicles based on FlexiNet.

2. Predict 3D box sizes and orientations of all predicted cars using proposed DNN and estimate their poses.

3. Based on LiDAR height from the ground, remove the ground points from the point clouds

4. Project extracted point clouds onto predicted 2D boxes preserving z-depth as a third channel.

#for each 2D detection window

5. **for** i = 1 to n, repeat:

6. Find outermost point (lXi,lyi,lzi) of across 2D centre(t_{Xi},t_{yi}) of car_i on y-axis.

7. set temp_cp_3Dbox_surface[i] = (t_{Xi},t_{yi},l_{zi})

8. Find the points cluster around (tXi,tyi,lzi) based on their z-gap-threshold

9. Find the closest[i] point from LiDAR in the cluster

10. arrange all closest[i] points in increasing order of distance from AV

#Mark the visibility of the car w.r.tits front front car

11. **for** i = 1 to n, repeat:

12. **for** j =1 to n-1, repeat:

13. **if** car[i] is hidden behind car[j]:

14. visibility[i] =j

15. **else**: #if the car is visible

16. visibility[i]=-1

#Estimate distance 3D centre of cars

```
for i = 1 to n, repeat:
  if visibility[i]==-1:
    if pose == Longitunal_side:
      c_depth[i] = l_zi + abs(width[i]/2cos(orientation[i]))
    else:
      c_depth[i] = l_zi + abs(length[i]/2sin(orientation[i]))
    endif:
  else:       #for occluded cars
    front = visibility[i]
    c_depth[i] = c_depth[front]+(closest[front]-closest[i])
  endif:
  t_zi= c_depth[i]
  3Dbox_centre_i = (t_Xi,t_yi,t_zi)
endfor:
Output: 3D_box_size, orientation, 3Dbox_centre
```

5.6 Performance Evaluation

For the evaluation of the proposed network, the performance of the proposed method was tested over KITTI as well as on the Waymo dataset. As the essential requirement of the network, firstly, 2D vehicle detection was performed using FlexiNet. 2D vehicle detection, all optimization steps were tested to get the best possible results in Section 4.3. In the second phase, based on 2D detection boxes, the size and orientations of 3D bounding boxes were predicted by using the proposed MobileNetV2-based DNN. In the network was finetuned using multiple learning rates and momentum values. Early stopping was preferred to avoid overfitting the network and stopped training if the results could not

come up with any improvement over the last ten epochs. In the performance analysis of the proposed neural network is evaluated in the range of distances.

The centres of 3D bounding boxes are detected using the proposed algorithm. As mentioned before, we have emphasised on giving a cost-effective solution for 3D vehicle detection, test the network based on sparse point clouds. In order to experiment with sparse point clouds, KITTI points are made sparse by removing the number of points in the 360-degree rotation of a single beam (Mehtab et. al., 2021). With different point clouds densities, 3D centre point estimation is evaluated, *w.r.t* distance ranges as well. In the last section, the overall inference speed of 3D vehicle detection is evaluated based on the sparsity of point clouds.

5.7 Summary

In this chapter, we have discussed the methodology to get the exact positioning of vehicles in the 3D world with the aim to give a cost-effective solution. In the workflow passes through four major steps, starting from the detection of 2D box windows using FlexiNet, we have estimated the size and orientation of 3D bounding boxes of vehicles based on RGB images of the scene using MobileNetV2. After the 3D centres of the vehicles were determined based on the predicted 2D centres and 3D point clouds information, in the final step, the performance of neural network was analysed at various distances using the sparsity of point clouds.

Chapter 6

Experimental Setup and Result Analysis

This chapter demonstrates the experimental setup, data preparation and analysis of results obtained based on the proposed methods in previous sections. In the main highlights of this chapter are experiments for 2D road scene perception, 2D vehicle detection, and 3D vehicle detection. In the 2D detection and classification results of the proposed unified FlexiNet framework are presented for cars, pedestrians, and cyclists (Mehtab and Yan, 2022). In the proposed network has attained high accuracy with 95.86% recall@0.5IoU on medium complexity KITTI dataset with a 10ms inference speed. In the scope is narrowed down for 3D object detection using cars and vans only. All experiments in this section are primarily run on the benchmark KITTI dataset and further tested on the Waymo dataset. In the performance of deep learning models is measured by using standard metrics of object detection. 3D vehicle detection, the effectiveness of deep learning models is presented using 16, 32 and 64 beams density point clouds for testing its capacity on low-cost hardware as well. In the proposed solution obtained 83.54% and 77.39% average accuracies for the size and orientation of 3D bounding boxes, respectively, with an approximate .2 second time between 20-50 meters of range based on the KITTI dataset.

In this thesis, we primarily target at improving 2D road scene perception accuracy for AV based on a unified DNN framework that can provide optimum performance by exploiting available resources. Secondly, the thesis fills the accuracy gap between 2D and 3D vehicle detection using sparse LiDAR point clouds and camera images.

The first section in this chapter covers the experimental setup and data preparation, followed by the results obtained using a unified detection model named FlexiNet for 2D road scene perception to localize cars, pedestrians and cyclists. However, in the second section, FlexiNet only targets at vehicles detection to support the final target of 3D vehicle detection research.

6.1 Experimental Setup

Table 6.1: Dependencies installed for the experimental setup

Python Tools	Version
Python	≥ 3.7
Torch	≥3.2.2
Torchvision	≥0.7.0
TensorBoard	≥2.2
Tqdm	≥4.41.0
PyYAML	≥5.3
Numpy	-
Cython	-
Matplotlib	≥3.2.2
Pillow	-
Scipy	1.4.1
OpenCV	4.1.2

The proposed DNNs are trained on Google Colab GPU machines with a batch size

of 16 images, the allocated GPU is mentioned in the corresponding sections. Dependencies for experimental setup are mentioned in the following Table 6.1. In the experiments, the results are compared based on precision, recall, mAP, and loss metrics to measure the performance of the model. In the details of these metrics are depicted in Section 4.3.

6.2 2D Road Scene Perception

Fig. 6.1 illustrates the workflow of 2D road scene perception execution using the proposed flexible neural network. Different finetuning strategies are followed to improve the network performance based on the normalization, network scaling, gradient descent optimizer, loss functions, and checkpoint selection.

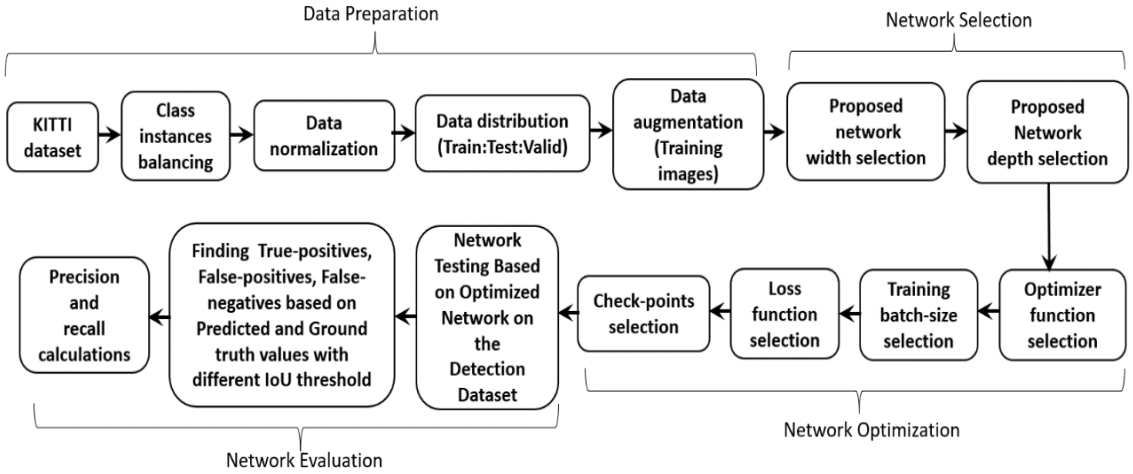


Fig. 6.1: The workflow of the proposed 2D road-scene perception experiments. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

6.2.1 Dataset Preparation

With the variety of data the neural network is trained, the better it is expected to perform on the test data (Gupta, 2018). In the road scene perception, the dataset quality refers to

the scaling of road users in the view, variation in environment and luminous conditions (Henrik, and Xerxes, 2020). As discussed in the literature survey, we have explored multiple public datasets such as KITTI, Waymo, and nuScene other than simulation datasets. However, among AV system researchers, the KITTI dataset has been the predominant choice, mostly the results are compared based on specific complexity criteria set by the KITTI. However, many datasets were created as well for training and testing purposes. Based on the evaluation, self-created datasets lack road scene complexity and other required challenges. Despite leading to high accuracy in results, the datasets would not be suitable in many practical implementations.

By considering the popularity and challenges of the KITTI dataset, we have primarily focused on the KITTI dataset in our experiments and the Waymo dataset for results verification on the night and rainy scenes. Among different camera images, we have selected the left camera images dataset (12GB) that are captured at 10Hz speed. There are 7,481 labelled images in the KITTI dataset and 7,500 unlabeled images of trajectories with 1350×350 average image resolution. In the selected dataset, we have considered four classes of interest: Car, van, pedestrian, and cyclist. In the car and van objects are put into the same category and assigned the same class, “Car”, as the difference would not significantly contribute to AV research; in fact, it would create ambiguity.

Fig. 6.2(a) shows the data distribution of different instances in the KITTI complete training dataset. Out of 7,481 images, only 2,486 images have pedestrians and cyclists instances with object count proportions of 18:3:1 for cars, pedestrians, and cyclists. As we know that data distribution plays a key role in the classification algorithm, we have conducted the filtering while finalizing images based on the different instances present. We have chosen 4,000 images covered all pedestrians and cyclists available in the dataset and additional images to include enough cars (Mehtab and Yan, 2022). Fig. 6.2(b) shows the finalized selection of images taken into consideration for our experiments. In the dataset is split into 8:2:2 ratio for training, testing, and validation in road scene detection experiments.

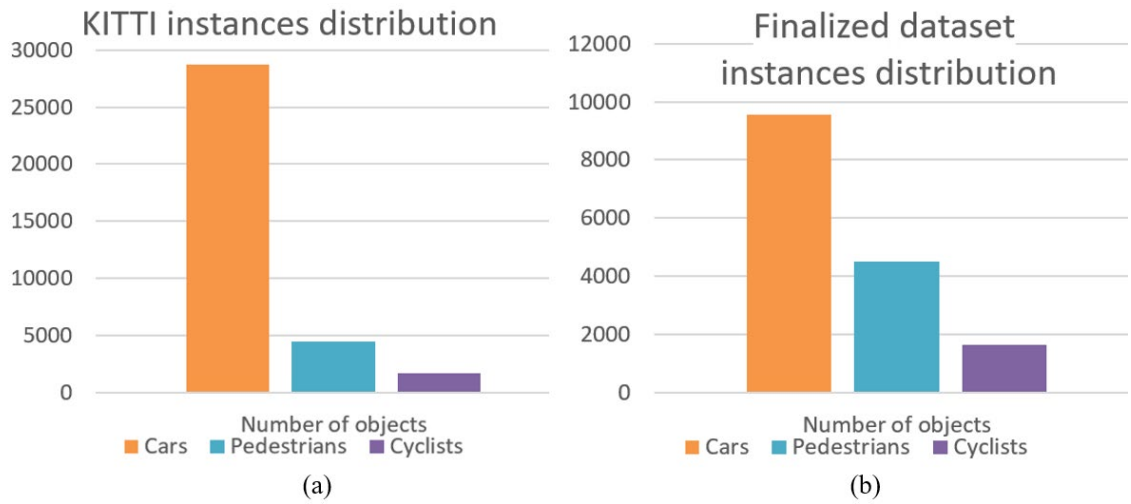


Fig. 6.2: The sample distribution (a) The labelled KITTI dataset with the ratio of instances distribution (b) The sorted KITTI dataset in the experiments includes all images containing pedestrians or cyclists with additional images. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

Fig. 6.3 shows the image samples taken from the KITTI finalized dataset. Here, we see a high degree of occlusion and truncation among objects and different scaling and lighting conditions. KITTI labelling format is given in Table 6.2. In the labelled information, the truncation complexity of objects is defined by 0,1 and 2 levels, while the occlusion is determined by using 0,1,2, and 3 levels. Alpha is the angle of observation of an object from AV. In the 2D bounding box is defined by using the left top and right bottom (x, y) coordinates object in the image plane, whereas for 3D bounding boxes, object dimensions: Height, width, and length in meters and their centres (x, y, z) in camera coordinates. Rotation angles of 3D boxes on X and Y planes are also provided.

Table 6.2: KITTI dataset formatting and calibration information

Seq#	Data Types	Descriptions
1	Object Type	“Car”, ”Van”, ”Pedestrian”, ”Cyclist”, ”Tram”, Person_sitting”, “Misc” or “DontCare”
2	Truncated Integer	(0,1,2) indicating the level of truncation.
3	Occluded Integer	(0,1,2,3) indicating occlusion state.
4	Alpha	Observation angle of the object, ranging [-Pi; Pi]
5-8	Bbox 2D	(0-based) the bounding box of the object: Left, top, right, bottom of image coordinates
9-11	Dimensions 3D	object dimensions: Height, width, length.
12-14	Location 3D	object location x,y,z in camera coordinates system.
15	Rotation_y	Rotation around Y-axis in-camera coords. [-Pi; Pi]
Calibration Information		$P2 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Camera Planes		x = right, y = down, z = forward
Velodyne Planes		x = forward, y = left, z = up
World planes		x = right, z = forward, y = down

Fig. 6.3: The samples from the KITTI dataset

In the proposed 2D object detection network named FlexiNet, standard YOLO format is employed for object labelling, where each object defined as [*class_id, centre_x, centre_y, width, height*] in the label file. Hereinafter, every object is identified with *class_id*, the remaining four parameters notify the bounding box location and dimension in the form of *x and y* coordinates. All parameters are normalized in the range [0, 1] by performing a simple division operation based on the size of images along the *x-y* axes. We again notify that we have considered only cars, vans, pedestrians, and cyclists among all classes.

Data Augmentation

Influenced by YOLOv5 and YOLOv4, FlexiNet performs CutMix (Yun and Corp, 2019) and Mosaic data augmentations on the training dataset to get exposed to a wider range of semantic variation (Bochkovskiy, Wang, and Liao, 2020).

(a) (b) (c)

Fig. 6.4: Data augmentation in the proposed network, (a) Mosaic data augmentation where four images are merged into one frame, (b) Cutmix augmentation, where a part of the image is cut for regularization, (c) Final image produced with the combination of Mosaic and cutmix augmentation.

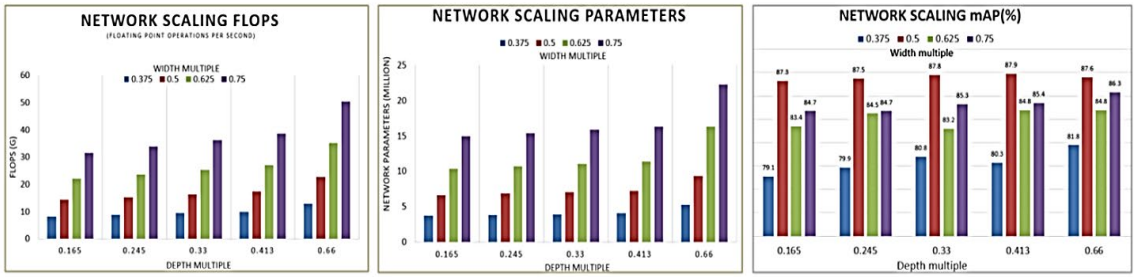
Multiple images are combined in the mosaic data augmentation as shown in Fig. 6.4(a), enhancing the detection outside their normal context, whereas cutmix

augmentation replaces one patch of the image with some blank information to avoid overfitting the model as shown in Fig. 6.4(b). Fig. 6.4(c), the combination of mosaic and CutMix data augmentation is illustrated. FlexiNet, we have resized all images to 640×640 resolution to achieve optimum accuracy with available hardware resources

The dataset particularly aims to push forward the development of computer vision and robotic algorithms for autonomous vehicles. In the KITTI dataset, there are 7,481 labelled images with an average resolution of 1350x350. Basic classes of interests are taken into consideration, including car, van, pedestrian, and cyclist. In the number of instances of car objects in KITTI are much higher than pedestrians and cyclists. Taken a balance between the class instances into account, 4,000 images were adopted, including all pedestrians and cyclists images for training, validation, and test with a proportion of 8:2:2. All images were scaled to 640×640 resolution, labels are normalized. Motivated by the latest progress YOLOv4 and YOLOv5, we have exploited CutMix and the image mosaic method as the augmentation based on the training dataset with a wider range of semantic variations.

6.2.2 Network Optimization

Firstly, we have identified the optimized network to achieve the best mAP@0.5IoU based on the existing hardware. Fig. 6.5 (a), (b), and (c) show Floating Point Operations (FLOPs), involved network parameters and final mAP@0.5IoU results respectively for sets of width and depth multiples based on FlexiNet baseline architecture. Interestingly, there is a considerable difference in FLOPs and network parameters as the network goes deeper and wider; however, the trends are not the same with precisions achieved. These statistics indicate that overparameterization leads to overfitting, the models become too complex to be generalized. It is to note that GPU assigned by the Colab for these 2D road scene perception is Tesla P100-PCIE with 16 GB memory.



(a) (b) (c)

Fig. 6.5. Network scaling results. (a) FLOPs executed at different sized networks (b) Parameter stored at different sized networks (c) mAP@0.5IoU obtained at different sized networks (GPU memory- 16GB). Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

As shown in Fig. 6.5(c), increasing the width of the network renders a significant improvement in the precision initially; however, it eventually results in sinking the performance. On the other hand, going deeper into the network improves the results at first, whereas later comes to saturation carrying excessive computational and storage overhead (Mehtab and Yan, 2022).

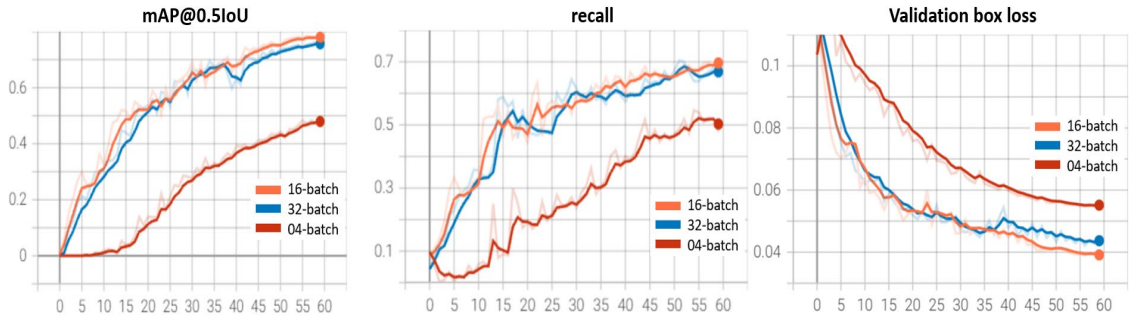
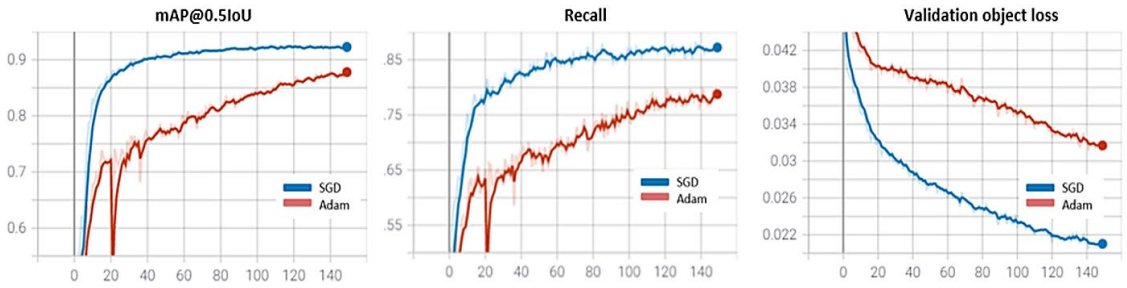


Fig. 6.6: FlexiNet performance on various batch sizes based on Adam Optimizer with finalized network size. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

For the given dataset, FlexiNet attained 87.8%mAP@0.5IoU with the *width_multiple* 0.50 and *depth_multiple* 0.33, exploiting the minimum hardware with

Adam optimizer (Kingma, Diederik, and Jimmy 2015) by using the GIoU bounding box regression loss function (Zheng et al., 2020). In the investigation of batch size, we have tested the network performance on various batch sizes; however, FlexiNet gave the best results on 16 batch sizes, as shown in Fig. 6.6.



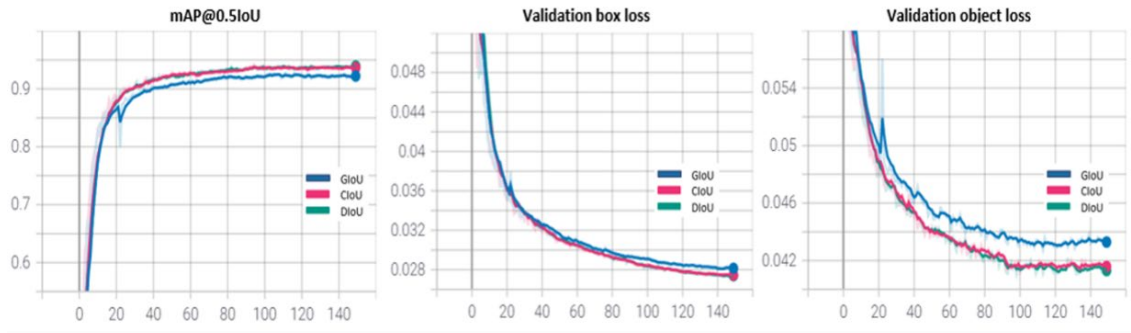
(a) (b) (c)

Fig. 6.7. Training and validation results based on Adam and SGD optimizer functions. (a) mAP@0.5IoU curves obtained based on training dataset (b) obtained recall values based on the training dataset and (c) objectness loss curves based on the validation dataset. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

In Appendix A.3, optimization algorithms make changes in the network parameters based on the gradient descent during backpropagation. However, the changes are considered after processing a number of images based on the batch size. In the prime consideration is to choose a batch size that is neither too small to make the output noisy nor too large to make the convergence slow. We have shown the results of three outcomes 4, 16, and 32 batches tested. Fig. 6.6 illustrates that batch 16 is performing the best on the proposed network in all three metrics considered, i.e., mAP, recall and validation loss. addition, the performance was further tested based on 64 and 128 batches, though initial results showed a lower performance as compared to 16 and 32 batches.

Optimizer selection plays a vital role in the performance of deep learning pipeline. In order to improve the accuracy of the results, we have investigated the effect of the SGD optimizer with respect to Adam on our final network with hyperparameters set based on

empirical testing (learning rate 0.01, momentum 0.94 and weight_decay 0.05×10^{-1}). Fig. 6.7 depicts the results of the SGD with momentum *vs* Adam optimizer, illustrating the lead of SGD over Adam by achieving 92.70% mAP and 87.4% recall based on the given training dataset. However, Adam started a fast convergence in the initial training phase but couldn't lead to the desired outcome eventually (Mehtab and Yan, 2022). These results further validate the work of (Choi et al. 2019; Wilson et al. 2017), adaptive methods are prone to getting influenced by spurious features that limit the neural networks finding out-of-sample generalization. However, it is also claimed that with a manually selected learning rate, SGD is guaranteed to converge to a local minimum, but the convergence remains slow; however, it overweights the remaining optimizer in overall performance if the momentum term is also included with SGD (Wilson et al., 2017).

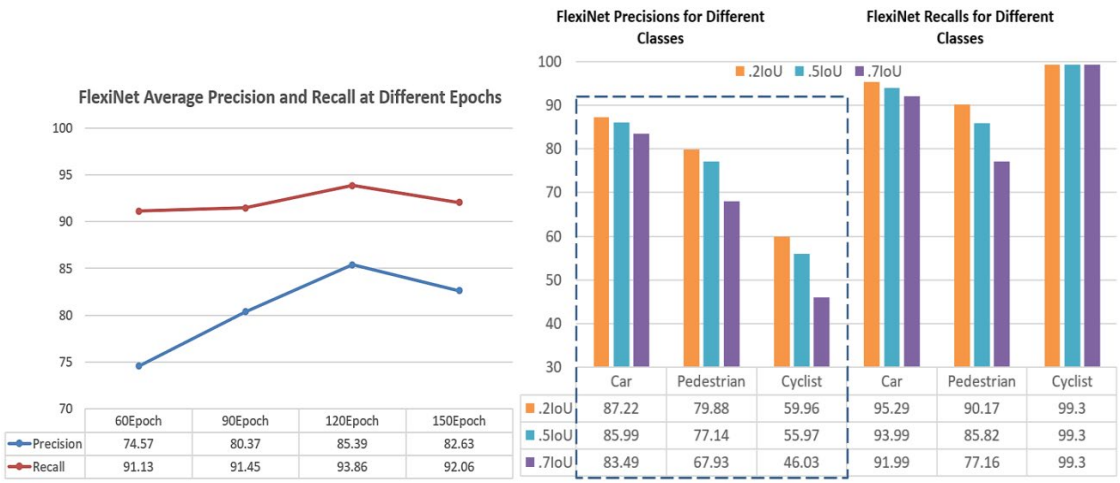


(a) (b) (c)

Fig. 6.8. Training and validation results based on GIoU, DIoU and CIoU loss functions. (a) mAP@0.5IoU curves based on the training dataset (b) the loss curves of the bounding box, and (c) the loss curves based on the validation dataset. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

In Section 4.2.3, IoU loss has institutive significance in the bounding box regression. In this is the final metric for non-max suppression in finalizing detection boxes. Based on their relevance, three different IoU loss functions are investigated; GIoU (Rezatofighi et al., 2019), DIoU and CIoU (Zheng et al., 2020).

Fig. 6.8 shows the results obtained by using different IoU loss functions. GIoU was the first major introduced IoU-based loss algorithm. It is the most popular one that focuses on the overlapping region of ground truth and predicts bounding boxes, and considers their non-incidental areas. Although it is much clear from the three figures that DIoU and CIoU losses converge faster compared to the GIoU loss function, thereby give better accuracy. In the DIoU and CIoU produced 1~2% better results because of the special cases if the predicted box directly fits inside the ground truth box completely; however, a size difference exists between them (Mehtab and Yan, 2022).



(a) (b)

Fig. 6.9. FlexiNet results based on the detection dataset (a) Obtained precision and recall values at 0.5IoU threshold at different intermediate checkpoints (b) Precision and recall results over the cars, pedestrians and cyclists objects by exploiting the best checkpoint. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

In Fig. 6.9, the FlexiNet results obtained based on the detection dataset are depicted at different epochs taking three IoU thresholds into consideration regarding individual classes,

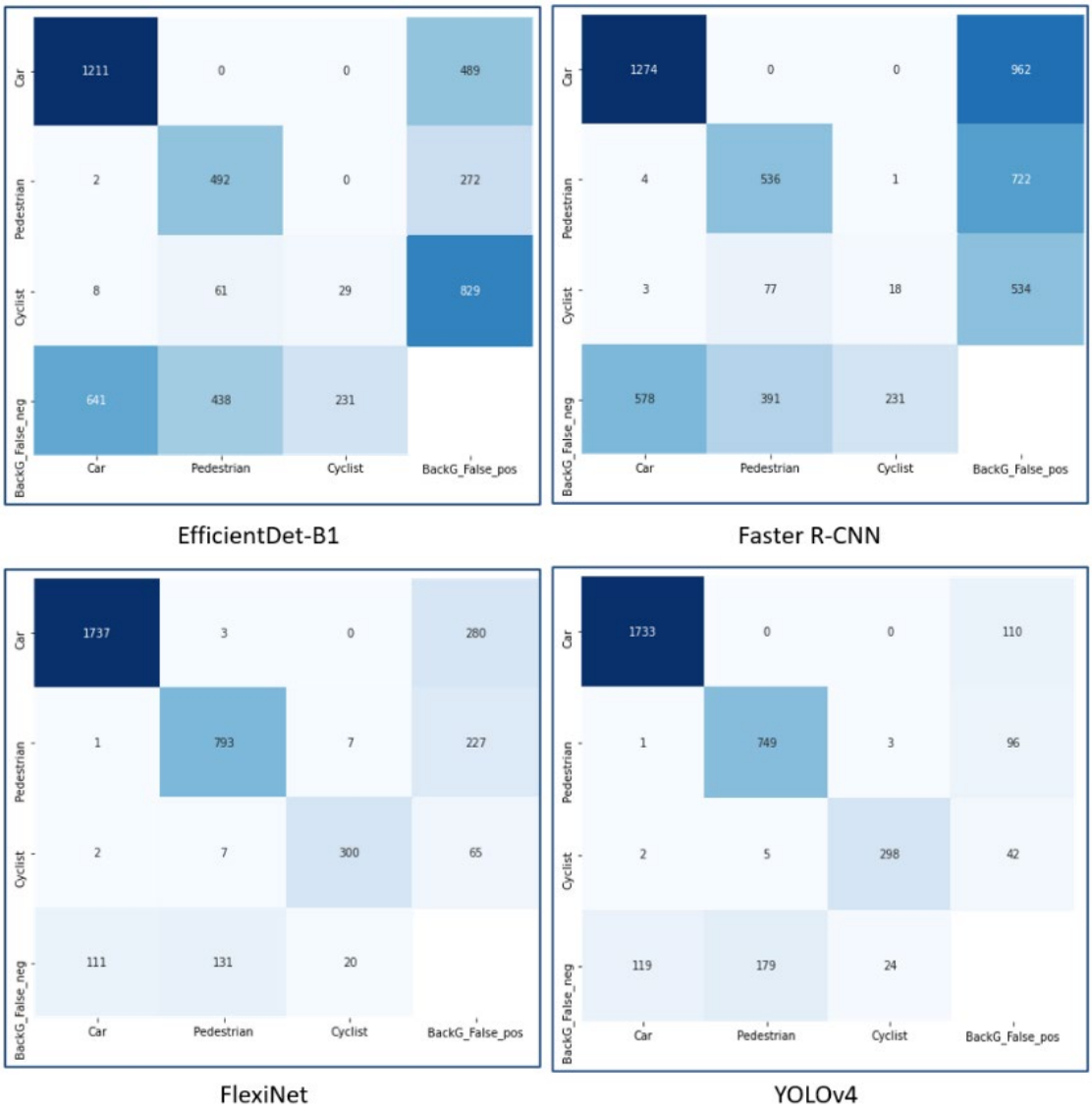


Fig. 6.10: Confusion matrix of the models taken into consideration for result evaluation based on the detection dataset for multiple classes, i.e., car, pedestrian, and cyclist. All results are evaluated on the same platform with 0.50 IoU threshold.

taking DIoU loss, and SGD optimizer with momentum into account. order to check the overfitting of the model, the model performance is tested at the intermediate checkpoints (Mehtab and Yan, 2022). Fig. 6.9(a) indicates that at 120 epoch, the model achieves the best performance with 86.39% average precision and 93.86% recall with 0.50 IoU;

however, its trend towards overfitting the model needs further training. Fig. 6.9(b) reveals the results of cars, pedestrians, and cyclists classes with respect to 0.70, 0.50 and 0.20 IoU thresholds (Mehtab and Yan, 2022). Section 4.6, the precision and recall values are directly proportional to true-positive; whereas precision is inversely proportional to false-positive rates and recall to false-negative ratio.

The proposed network proves to be efficient in achieving high recall, whereas slight low precision indicates the existence of false-positives, i.e., false detection of non-existing objects in the results. Fig. 6.9(b) shows that over cyclist class, the model has attained the poorest precision though with the best recall; on the other hand, car objects attained 87.22% precision and 95.29% recall at 0.70 IoU. Although the precision results give a margin of improvement in the proposed solution, the results obtained with 0.2IoU thresholds are promising compared to 0.50 IoU and 0.70 IoU as expected for obvious reasons; however, the cyclist class achieves 99.3% recall in all consideration of IoU.

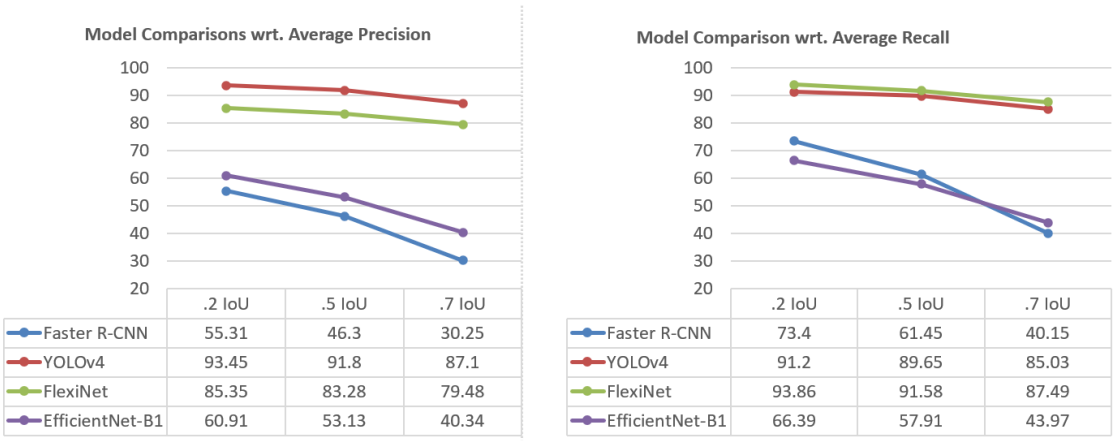
6.2.3 Performance Evaluation

In this section, the results are compared with the state-of-the-art object detection models. All models are executed on the same platform by using the same datasets. In the benchmark KITTI dataset gives three complexity criteria, named “Easy”, “Medium”, and “Hard”, for visual object detection based on the size, occlusion level, and truncation score in Section 4.3.

Confusion matrices of detection results obtained are presented in Fig. 6.10, depicting total true-positive, false-positive, and false-negative number. EfficientDet has shown the poorest performance by attaining the highest false-negative number in all instances and worst for cyclists. Given insight into it, we found that EfficientDet and Faster R-CNN couldn’t generate enough discrepancy between cyclists and pedestrians and also got confused with parked cycles as cyclists and thereby led to poor results.

Based on the different IoU thresholds, we evaluated the true-positive, false-positive

and false-negative numbers based on the prediction results and ground-truth values. That was employed to find final precision and recall values as per equations given in Section 4.3. Confusion matrices of detection results obtained are presented in Fig. 6.10, depicting overall true-positive, false-positive, and false-negative number obtained by using various models. EfficientDet has shown the poorest performance by retrieving the highest false-negative in all occurrences and worst for cyclists objects. Given the insights, we found that EfficientDet and Faster R-CNN couldn't make the decision between cyclists and pedestrians very clearly and got confused with parked cycles, also considering them as cyclists and thereby leading to poor results. evaluating the precision and recall calculation, false-positive numbers are considered only to the ones notified in different classes as per general standard (Mehtab and Yan, 2022).



	.2 IoU	.5 IoU	.7 IoU
Faster R-CNN	55.31	46.3	30.25
YOLOv4	93.45	91.8	87.1
FlexiNet	85.35	83.28	79.48
EfficientNet-B1	60.91	53.13	40.34

	.2 IoU	.5 IoU	.7 IoU
Faster R-CNN	73.4	61.45	40.15
YOLOv4	91.2	89.65	85.03
FlexiNet	93.86	91.58	87.49
EfficientNet-B1	66.39	57.91	43.97

(a) (b)

Fig. 6.11: The comparison of FlexiNet model with the state-of-the-art object detectors based on test dataset (a) Average precision results at different IoU thresholds (b) Recall results at different IoU thresholds. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

Fig. 6.11 shows the two-dimensional comparison of the detection models in terms of precision and recall at the different IoU threshold values based on the test dataset. Results depict that FlexiNet and YOLOv4 (Wang et al. 2020) outperform Faster R-CNN

(Ren et al. 2017) and EfficientDet-B1 (Tan, Ruoming and Quoc 2020) at both the metrics. Fig. 6.11(b) indicates that FlexiNet achieved the best recall rate over other models. Although, YOLOv4 proves better in terms of precision that reveals lower false-positive numbers for prediction. In the results also show the trend of decline in the models performance with an increasing IoU threshold in general; however, FlexiNet and YOLOv4 produced consistent output altogether.

Table 6.3 shows the FlexiNet comparison with the state-of-the-art detection models using recall, size and fps as attributes. In the KITTI criteria of easy, medium and hard challenges as discussed in Section 4.3 is considered. All results are based on 0.5 IOU threshold value. Table 6.3 shows the decrease in accuracy with increased challenges over all the detectors. From the results, we see that FlexiNet shows a better recall rate than other detectors for all three challenging conditions over the classes of cars, pedestrians, and cyclists in general.

A clear view of the models performance is seen in Fig. 6.12 considering two detection images based on each model. Fig. 6.12, the results are arranged in a row-wise manner in the sequence of ground-truth, EfficientDet, Faster R-CNN, YOLOv4 and FlexiNet, respectively. It was found that in the case of car objects, Faster R-CNN (Ren et al., 2017) and EfficientDet-B1 (Tan, Ruoming and Quoc, 2020) achieved reasonable accuracies; on the other hand, the same models depicted weak performances in identifying pedestrians, cyclists. Especially for the cyclists objects, Faster R-CNN and EfficientDet showed difficulty differentiating cyclists from stand-alone cycles or pedestrians.

Table 6.3: The comparisons of the FlexiNet model with other popular detectors based on the KITTI dataset with Easy, Medium, and Hard levels of complexities at 0.5 IoU threshold, recall is considered the prime metric to be considering the importance miss ratio in autonomous driving scenarios; however average precision, model size and inference speed are also taken into consideration. Note: from (Mehtab and Yan, 2022) In *Multimedia Tools and Applications*

Model	**Cars (recall%)**			**Pedestrians (recall%)**			**Cyclists (recall%)**			**Average Precision (%)**	**Model size (MB)**	**fps**
	Easy	Medium	Hard	Easy	Medium	Hard	Easy	Medium	Hard			
Faster R-CNN	96.12	76.33	68.68	80.59	66.72	43.58	12.43	9.13	6.76	46.33	7	5
EfficientDet-B1	96.42	72.62	68.80	80.01	77.23	46.18	17.53	16.32	5.92	53.09	27	15
YOLOv4	99.42	93.81	90.28	96.38	71.41	74.19	99.99	98.57	83.89	91.78	244	25
FlexiNet (Our)	99.68	95.01	91.18	98.28	93.32	83.22	99.33	99.32	99.33	83.18	54	100

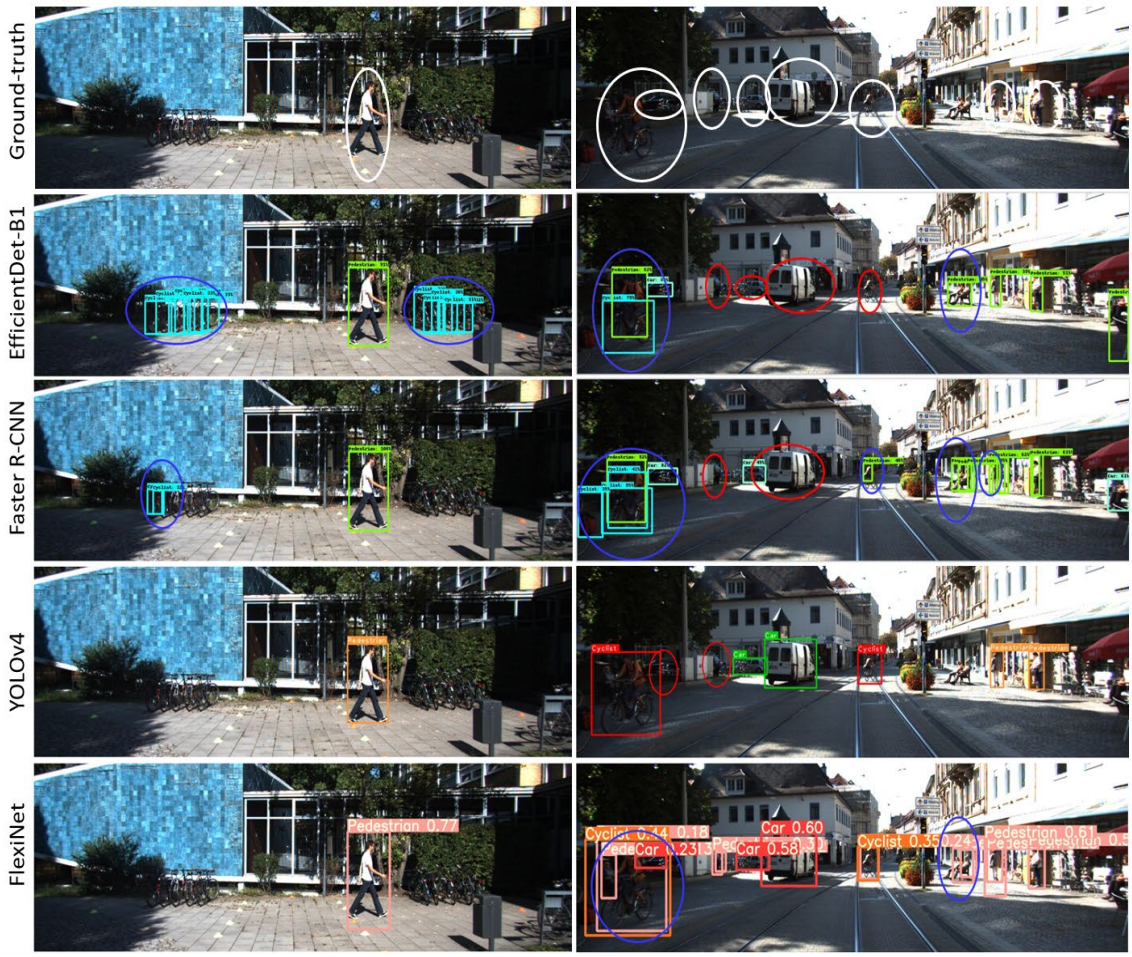


Fig. 6.12: Detection results of four models with three classes, i.e., “Car”, “Pedestrian”, and “Cyclist” are depicted..

Fig. 6.13: The detection results of car, pedestrian, and cyclist instances by using the proposed FlexiNet architecture based on the KITTI dataset, from (Mehtab and Yan, 2021) In *Multimedia Tools and Applications*

6.2.4 Summary

Based on the results of 2D road scene perception, it is justified that network size plays a

crucial role in the neural network-based visual object detection method. In the flexible architecture of the network is a promising approach to find the optimum sized network. In the results show that SGD with momentum performs much better than Adam optimizer based on the dataset considered despite much popularity of Adam. Among IoU-based bounding box losses, DIoU escalates the network performance by 1~2% precision as compared to GIoU. Model capturing is performed at various intermediate checkpoints to investigate the overfitting curve, the most promising performance of the model at epoch 120 was finally employed for detection purposes.

The proposed model is compared with the state-of-the-art object detection models. Albeit FlexiNet remained unsuccessful in achieving the best precision compared to YOLOv4 that can be seen in Table 6.3 and Fig. 6.12. However, it gives the lowest false-negative number, which is the most accountable factor in autonomous driving. In addition to that, FlexiNet gave four times faster speed (i.e., 100fps) compared to YOLOv4. Another considerable fact is that training of the YOLOv4 model requires five times more memory (around 250MB) as compared to FlexiNet (approximately 50MB) that is a considerable measure while executing the model on edge devices with limited resources. A few detection results captured by FlexiNet are depicted in Fig. 6.13, where the bounding boxes of three classes of interests are shown in various colours with the confidence scores (Mehtab and Yan, 2022).

6.3 2D Vehicle Detection Results

After successful 2D road scene perception, we narrowed down our research for vehicle detection in autonomous driving with the aim of finally dive into 3D vehicle detection. In the focus of this section is on finetuning the results of the proposed FlexiNet for 2D vehicle detection only. Regarding experimental evaluations of 2D vehicle detection, we have followed the same approach of 2D road scene perception in Section 6.2 by using the KITTI dataset while considering “Car” and “Van” objects alike for practical reasons.

During experiments, 2,400 random images were employed for training and validation with a proportion of 8:2:2, respectively (Mehtab et. al., 2021). In the GPU allocated for these experiments was Tesla X GPU with 15GB memory.

Fig. 6.14: Image samples of downloaded Waymo segments, the dataset shows varying weather and daytime conditions that make it suitable for DNN training in autonomous driving.

In vehicle detection, the second dataset is the Waymo dataset. In the details of this KITTI dataset are already discussed in Section 6.2.1. However, the Waymo dataset is briefly introduced in this section. As shown in Fig. 6.14, Waymo contains a variety of scene images to train the DNN for all weather and lighting conditions. All images in the Waymo dataset have resolution 1980×1280. Unlike KITTI, the Waymo dataset is

published in the form of *tfrecords* files, with continuous track labelled frames. Waymo has provided an enormous amount of data in the form of 900 segments, with each segment of around 1GB size. We downloaded 35 segments and converted them into KITTI labels to be compatible with the existing processing standards of our experiments. It was found to contain the night, rainy as well as foggy scenes with mostly car objects in the downloaded data segments. Random frames were picked from these collections to make our dataset general. Amid the training process, variations in functions such as activation, optimization, and loss functions are employed to observe the network performance.

6.3.1 Network Optimization

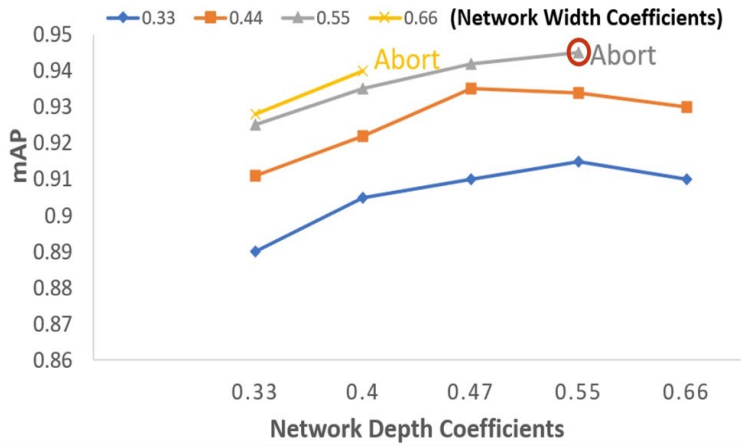


Fig. 6.15: The performance of the proposed scaling network for a set of widths and depths based on FlexiNet architecture. Note: from (Mehtab et. al., 2021) In *ICCCV*

For the sake of different sets of data and GPU in 2D vehicle detection experiments, the structure balancing of FlexiNet was required to get the best performance based on available resources. With dynamic neural network scaling, a continuous increase in the performance was observed with the width expansion based on the training dataset. However, as the depth of the network grew, initially, the performance improved at a fast pace, but after going further deep, the performance started degrading and eventually led to the abortion of the experiment because of the high GPU storage requirement. It is to

be noticed that network scaling requirement varies with the dataset selection. Fig. 6.15, FlexiNet attained the best performance at 0.55 *depth-multiple* and 0.55 *width-multiple* with the assigned training dataset, all experiments ahead were run on this network configuration (Mehtab et. al. 2021).

Table 6.4: The influence of various activation functions on mAP of 2D vehicle detection

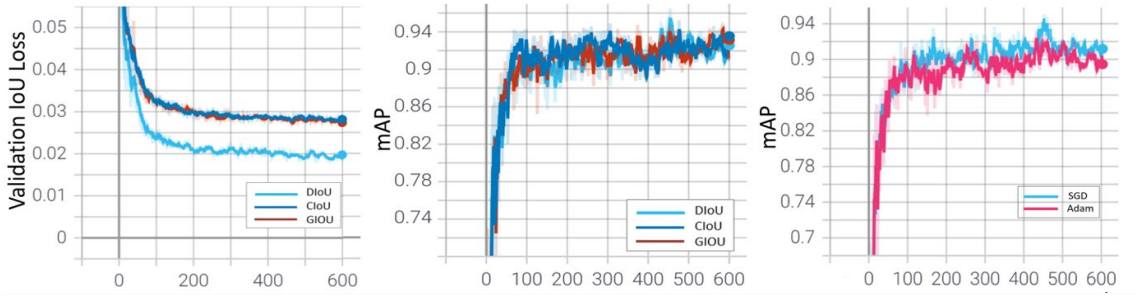
Activation Function	mAP@0.5IoU(%)	Recall(%)
FReLU	93.38	96.39
Swish	92.22	91.67
Mish	92.91	92.23
Hardswish	94.52	96.38

Note: from (Mehtab et. al., 2021) In *ICCCV*

With regard to the activation function fitness in the optimized FlexiNet architecture (Mehtab et. al., 2021), FReLU (Qiu and Cai, 2018) and Hardswish (Avenash and Viswanath, 2019) outperformed other functions in Table 6.4. Based on the literature surveyed, it was found that ReLU-based activation functions give inconsistent results (Ramachandran, Zoph and Le, 2017). However, regarding the dataset considered and the proposed network, the FReLU activation function has shown promising performance. On the other hand, Hardswish, which is a piece-wise linear analog and non-monotonic activation function, provides a bump in the negative gradient regions that makes it different from swish and mish activations. Based on empirical results (Avenash and Viswanath, 2019), it was found that a high percentage of pre-activation weights and biases falls in the negative region of bump ($-2.50 \leq x \leq 0$) leading to better convergence in the case of Hardswish. Our results validate the previous findings by giving a similar performance.

In the selection of bounding box regression loss, the performance of DIoU, CIoU and GIoU (Zheng et al., 2020) losses were examined, with results shown in Fig. 6.16 (a). Fig. 6.16 (b) shows the corresponding mAP curves; DIoU gave 1% better mAP based on

the validation dataset over GIoU loss functions. Pertaining to vehicle detection, 1% growth in precision is a noteworthy achievement for avoiding road accidents. Although CIoU gave a comparable performance to DIoU, it causes extra computation with no accuracy improvement based on the results achieved (Mehtab et. al., 2021).



(a) (b) (c)

Fig. 6.16: 2D vehicle detection training results based on FlexiNet (a) Validation loss curves with GIoU, CIoU, and DIoU functions, (b) Network mAP with respect to different IoU losses, (c) Network mAP with respect to SGD and Adam optimizers. Note: from (Mehtab et. al., 2021) In *ICCCV*

Pertaining to further performance enhancement, the selection of gradient descent optimizer for the existing dataset was considered. Fig. 6.17 (c), the results illustrate that Adam optimizer (Kingma, Diederik, and Jimmy, 2015) has demonstrated faster convergence at the initial stages; however, SGD with momentum proves to be a better choice for giving the maximum accuracy whilst keeping a slow learning rate as shown in Fig. 6.17(c). In the best results are obtained at a learning rate of 0.01, with the momentum of 0.94 using the SGD optimizer. These findings align with the results obtained in Section 6.1.1.

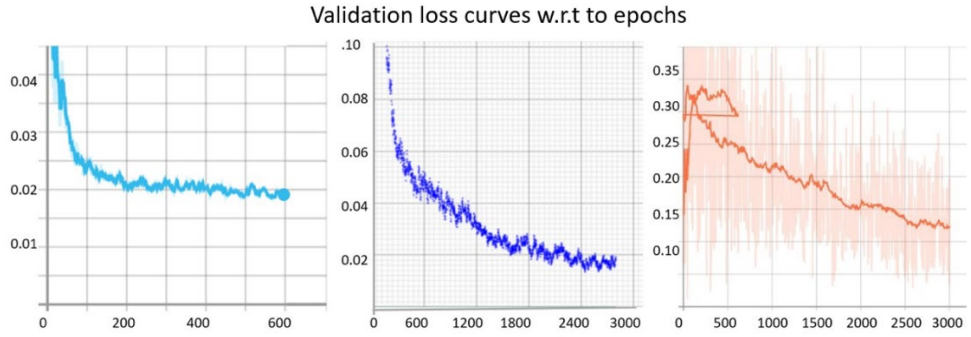
6.3.2 Performance Evaluation

Optimized FlexiNet results are compared with the state-of-the-art detection networks on the same platform for solely 2D vehicle detection. Table 6.5, it is clear that SSD (Wang et al., 2019), YOLOv3 (Redmon, Farhadi 2018), and EfficientDet-B2 (Tan, Ruoming and Quoc, 2020) were relatively unsuccessful in providing promising results in the vehicle

detection dataset due to its high occlusion, truncation, scaling and lighting conditions present in the KITTI dataset. Although Faster R-CNN achieves 82.9% mAP@0.5IoU based on the same platform, its speed is much slower than YOLOv4 and FlexiNet, which is not suitable for real-time vehicles detection.

Table 6.5: Comparisons of the proposed FlexiNet with the state-of-the-art detection methods for 2D vehicle detection, Note: from (Mehtab et. al., 2021) In *ICCCV*

Model	mAP@0.5IoU (%)	Recall (%)	fps
EfficientDet-B2	31.89	32.12	8
Faster R-CNN	82.92	56.33	7
SSD	22.23	12.71	40
YOLOv3	70.28	22.93	35
YOLOv4	92.51	-	25
FlexiNet (our)	94.52	96.41	72



(a) (b) (c)

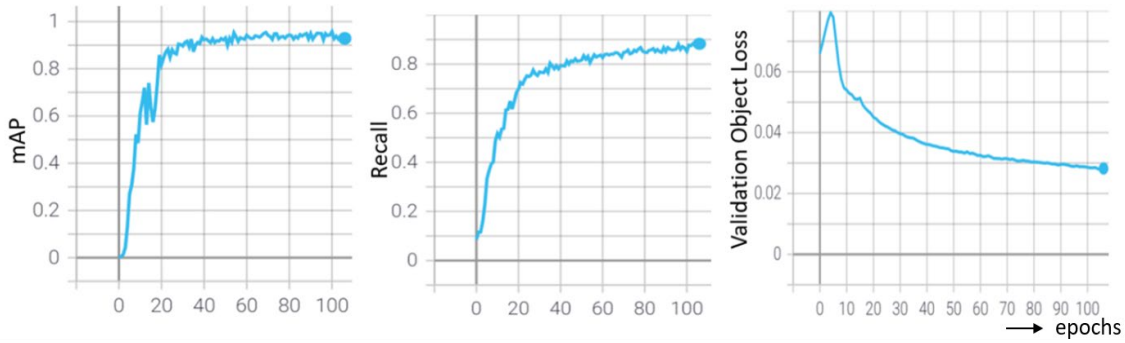
Fig. 6.17: FlexiNet validation loss analysis with multiple networks, (a) FlexiNet converges to 1.8% loss at 600 epochs; (b) YOLOv4 converges to 1.8% loss at 2600 epochs; (c) Faster R-CNN converges to 12% loss at 3000 epochs. Note: from (Mehtab et. al., 2021) In *ICCCV*

The respective loss curves of FlexiNet, YOLOv4, and Faster R-CNN are shown in Fig. 6.17.

Fig. 6.18: Vehicle detection results based on test images of the KITTI dataset by using FlexiNet

FlexiNet converges to 1.8% loss in 600 epochs, YOLOv4 achieved a comparable loss in 2,600 epochs, and Faster R-CNN shows continuous improvements in the loss but could not go beyond 12%. Between YOLOv4 and FlexiNet, there is not much variance in the final precision; however, it was observed that FlexiNet generated 94.5% mAP and 96.4% recall after 4,50 epochs, whereas YOLOv4 gives comparable accuracy after 1,300 epochs. While testing YOLOv4 model, the batch size could not be increased by more than 16 with image resolution 640 x 640 based on our 15GB GPU. On the other hand, FlexiNet could run efficiently with batch size 64 (Mehtab et. al., 2021).

To further test the performance, FlexiNet training and detection on the Waymo dataset (Sun, Henrik, and Xerxes, 2020) were conducted. However, despite night and rainy images in the Waymo dataset, FlexiNet resulted in 97.25% mAP, 84.92% recall, and 2.84% object loss, as shown in Fig. 6.19, which is better than the KITTI dataset-based performance. One of the reasons for this increased performance is the higher occlusion and truncation scenes presence in KITTI over the Waymo dataset.



(a) (b) (c)

Fig. 6.19: FlexiNet results based on Waymo dataset using DIoU loss, Hardswish activation and SGD optimizer functions for 2D vehicle detection in autonomous driving (a) mAP curve (b) Recall curve (c) Object loss curve, Note: from (Mehtab et. al., 2021) In *ICCCV*

Fig. 6.18 shows the detection results of FlexiNet on test images based on the KITTI

dataset. Fig. 6.20 shows FlexiNet detection results based on the Waymo test dataset. Hence, we see that the network gives consistent performance in extreme weather and night time conditions also (Mehtab et. al., 2021).

Fig. 6.20: Vehicle detection results based on test images of Waymo dataset by using FlexiNet

6.3.3 Summary

Pertaining to 2D vehicle detection, FlexiNet is trained based on KITTI and Waymo datasets. Our experiments showed that network scaling resulted in the best performance of the proposed network at 0.55 width and 0.55 depth multiples based on the baseline framework. In the network achieved the best performance with SGD optimizer using Hardswish activation function with DIoU bounding box regression functions. FlexiNet showed outstanding results compared to Faster R-CNN, EfficientDet, SSD and YOLOv3 by a remarkable gap (Mehtab et. al., 2021). Although YOLOv4 gave comparable precision with real-time inference speed, considering the training time and computing complexity of YOLOv4, it is justified to say FlexiNet is the best possible solution for 2D vehicle detection based on the dataset used. In the results obtained in 2D vehicle detection would be further utilized for 3D detection with additional LiDAR point clouds information in Section 6.4.

6.4 3D Vehicle Detection Results

In this section, we focus on the results obtained using the proposed 3D vehicle detection method in Chapter 5. Hereinafter, we would analyse the results based on 3D box size (dimensions) and orientation accuracy achieved by using the proposed DNN and the efficiency of the 3D centre extraction method based on the sparsity of LiDAR 3D point clouds.

For training and testing of the 3D vehicle detection model, we have utilized the benchmark KITTI dataset (Geiger et al., 2013) and Waymo datasets (Sun, Henrik, and Xerxes, 2020), already discussed in Section 6.2 and Section 6.3. For each dataset, 2,400 images were taken into account for the experiments that were split into the ratio of 8:2:2 for training, validation, and testing purposes (Mehtab et. al., 2021). In the proposed method is based on existing 2D vehicle detection results that have been predicted using FlexiNet in Section 6.2.

In the first part of the experiment, 3D box sizes and orientations of vehicles were estimated by using the proposed DNN (Mehtab et. al. 2021). As discussed in Section 4.3, MobileNetV2 has been exploited as a features extractor with the replacement of the fully connected layers for our specific requirement. 2D object detection results were cropped into the 224×224 windows so as to train the proposed DNN.

After predicted the 3D box size and orientation of cars, the centre detection was targeted. By projecting LiDAR point clouds on the 2D detection windows of car objects, the 2D centres were transformed into 3D using trigonometrical geometry, taking the size and orientation into account, as discussed in Section 5.4, the results obtained by using each module are presented and analysed.

6.4.1 DNN Performance

Learning rate is the most important hyperparameter to be considered during neural network training. Learning rate makes the change in the network weights based on the loss occurred as given in Eq. (3.7). However, learning rate selection is a crucial factor. A too-large learning rate can skip the global minima; on the other hand, a too-small learning rate may get the network stuck in local minima, given a false impression of global minima. Also, a small learning rate causes an increase in the training time enormously. Therefore, empirical testing is performed to find the most suitable learning rate, which resulted in 0.001 with the SGD optimizer (Mehtab et. al., 2021).

Another term that plays a vital role in network optimization is for skipping the local minima and making convergence fast is “momentum”. Momentum in combination with the learning rate makes the convergence smooth and increases the accuracy. a neural network, momentum tracks the exponentially weighted average of previous gradients and stabilizes the convergence as discussed in Appendix-A. Ideally, the momentum lies in the range [0, 1]. our experiments, the momentum value is considered 0.9, prioritising the latest gradient. Analogously, the momentum can be considered as a marble rolling down the hill towards its valley. Although marble may fall into a small dip at the intermediate

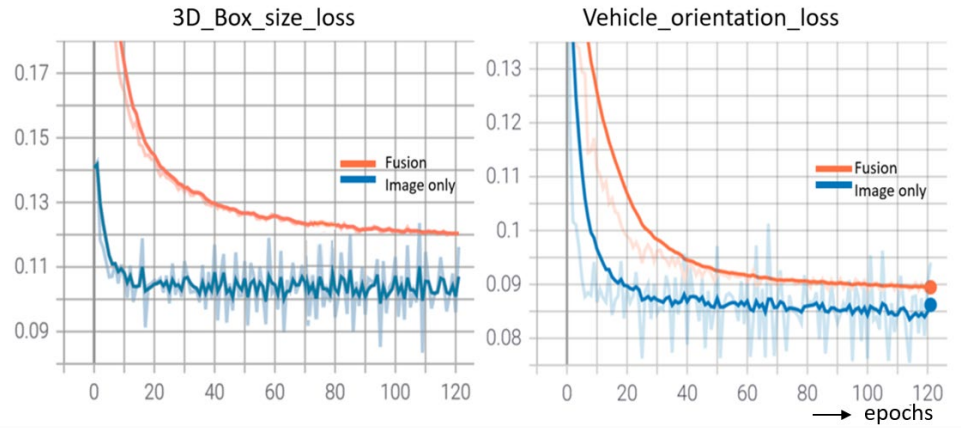
stage, due to its momentum successfully jumps out; however in the case of global minima, it loses sufficient kinetic energy and falls back into the valley.

6.4.1.1 Result analysis based on the KITTI dataset

For predicting the 3D box sizes and orientations of cars, we tested the proposed DNN on image datasets as well as early fused datasets (point clouds projection on image).

(a) (b) (c)

Fig. 6.21: KITTI Dataset single frame (a) LiDAR point cloud (b) Image of the same scene (c) Early fusion of point clouds and image by projecting point clouds onto the image. Note: from (Mehtab et. al., 2021) In *ICVNZ*

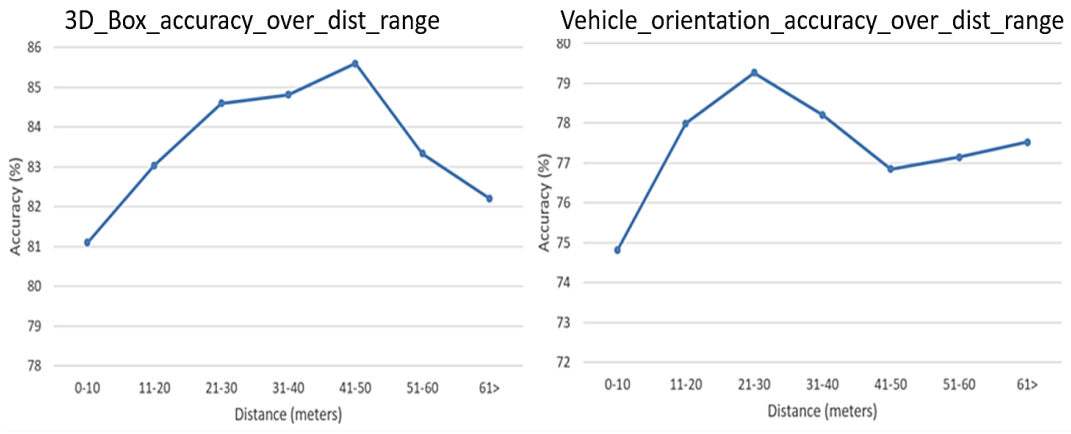


(a) (b)

Fig. 6.22: The comparison of validation loss results of the proposed DNN for early fused vs image only input data formats based on KITTI datasets (a) accuracy achieved for predicted 3D bounding boxes size (b) accuracy achieved for vehicle orientation prediction. Note: from (Mehtab et. al., 2021) In *ICVNZ*

Fig. 6.21(a) shows an example of LiDAR BEV point clouds of the KITTI dataset. Fig. 6.21(b) displays the RGB image of the same frame. On the other hand, Fig. 6.21(c) depicts early-stage fusion (Mitchell, 2007) of image and point clouds. Our experimental results based on the KITTI validation dataset are shown in Fig. 6.22; it was found that the images only are better for features extraction without extra computational cost, as in the case of early fusion dataset (Mehtab et. al., 2021).

Fig. 6.23 represents the detection results of the proposed DNN with respect to distances from AV. Based on the experiments, the network performance is most promising in the 20~50 meters range using camera images. In the proposed network gives 83.54% average accuracy of 3D box size and 77.39% average accuracy over orientation. our experiments, the loss function coefficients α and β were set to 6.0 and 3.0 respectively based on empirical test results. Although the results showed a marginal gap of accuracy for closed range vehicles in the range of 0~10 meters, the reason lies in the complexity of the dataset, where there is a high degree of truncation and occasion in appearances that caused some degradation of results.



(a) (b)

Fig. 6.23: The detection results of the proposed DNN over distance range based on the KITTI test dataset, (a) 3D box size prediction accuracy w.r.t range (b) orientation prediction accuracy w.r.t range. Note: from (Mehtab et. al., 2021) In *ICVNZ*

6.4.1.2 Result analysis based on the Waymo dataset

For further verification, we checked the network performance on the Waymo dataset (Sun, Henrik, and Xerxes, 2020) by taking into account the night and rainy scenes. Fig. 6.24 shows the validation results obtained through different extremities of the network based on the Waymo dataset. Compared to the KITTI dataset, using the Waymo dataset, the proposed network achieved better results, 99.5% accuracy in terms of orientation, as shown in Fig. 6.24(a). However, the prediction loss of the size of 3D bounding box converged at 15%, as shown in Fig. 6.24(b).

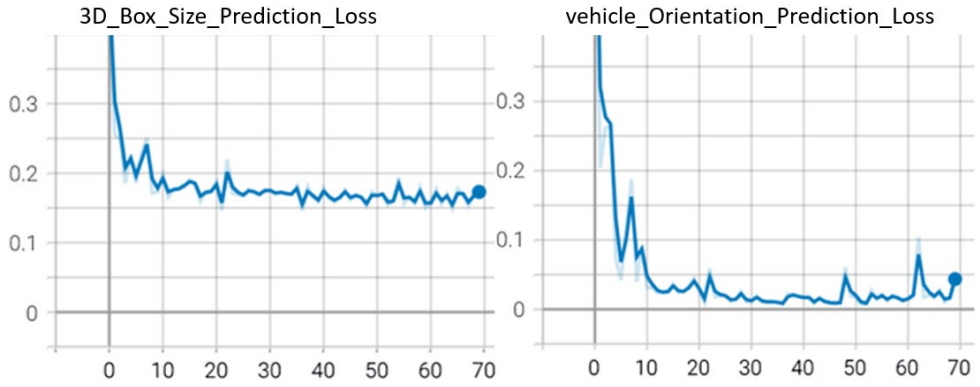


Fig. 6.24: The validation loss results of the proposed DNN based on Waymo datasets (a) 3D box size prediction accuracy w.r.t epochs (b) orientation prediction accuracy w.r.t epochs. Note: from (Mehtab et. al., 2021) In *ICVNZ*

6.4.2 Estimation Results of 3D Bounding Boxes

Fig. 6.25 illustrates the results of intermediate steps of the proposed algorithm. Fig. 6.25(a) shows that 2D bounding boxes were obtained with their confidence scores by using our previous 2D vehicle detection based on FlexiNet. Each proposal was fed into the proposed DNN net to yield the size and orientation of 3D bounding boxes around car objects. Fig. 6.25(b) displays the projected point clouds on the image (Mehtab et. al., 2021). During

projection, only the point clouds within the 2D detection windows were considered, trimming the ground points based on the LiDAR vertical distance from the ground. In the remaining point clouds gave a clear view of the car objects with their depth values as an added information channel. Points colours in the cloud notify the distancing from AV. Fig. 6.25(c) shows the 2D centre of detected cars by using small circles (i.e., p_1) with the depth values estimated by using projected point clouds, whereas the outermost 3D points based on the car surface using 2D central vertical-axis are represented with big circles (i.e., p_2). Integration of p_1 and p_2 gives 3D car centres projection on the 3D bounding box surface.

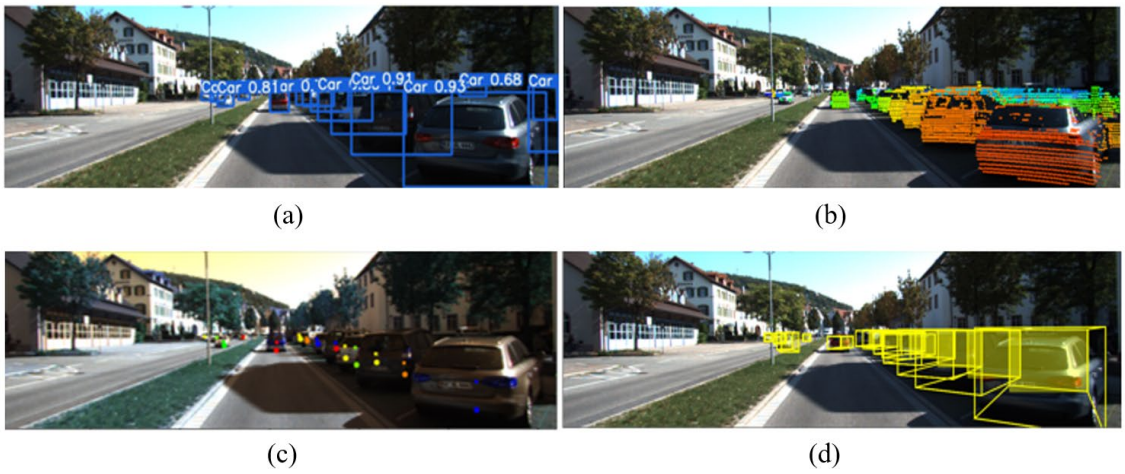


Fig. 6.25: (a) The example of predicted 2D bounding boxes based on the KITTI dataset (b) The projected LiDAR point clouds onto 2D detection windows of image after ground points removal (c) The small dots show the centres of 2D bounding boxes whilst the big dots depict the maximum bulged out 3D surface points across y-axis of the 2D centres (d) Based on estimated 3D centres, sizes, orientations and poses of 3D bounding boxes of cars. Note: from (Mehtab et. al., 2021) In *ICVNZ*

These centre projections were further extended inside the car to estimate the inner 3D centre distance by exploiting the orientation and dimensions predicted information along with the cars pose. Furthermore, these 3D centre points were converted back into world coordinates by using inverse projection and inverse rotation-translation matrices. In the finally positioned 3D bounding boxes of cars are presented in Fig. 6.25(d).

Fig. 6.26: The test results of 3D car detection based on the KITTI dataset by using the proposed model

Fig. 6.27: The test results of 3D car detection based on the Waymo dataset by using the proposed model

Fig. 6.26 depicts sample 3D vehicle detection results with the proposed algorithm based on the KITTI test dataset, including extreme weather and lighting conditions, Fig. 6.27 illustrates 3D vehicle detection results based on the Waymo test dataset. In the object detection results show that even in extreme weather conditions, the network shows good 3D detection results (Mehtab et. al., 2021).

6.4.3 The Experiments with Sparse Point Clouds

Pertaining to the proposed solution on spars point clouds, we have converted the KITTI point clouds into 32 and 16 beams keeping one set in the original 64 beams format as shown in Fig. 6.28. On the left side of the figure, raw point clouds with different densities are depicted, whereas the right side presents the projection of the point clouds on the area under consideration in the images.

A noticeable factor in these images is the pattern of sparse point clouds that give continuous information on the horizontal axis and misses information on the vertical plane. Taken advantage of this fact, the proposed solution relied only on the closest points from AV in the cars clusters and the outermost points on the cars surface closed to the 2D central vertical line to get their 3D central distances. In the reason for seeking a point on the vertical axis is to extract the outermost point on the 3D box surface.

The comparison of accuracies achieved over distances with different point clouds densities, i.e., 16, 32 and 64 are shown in Fig. 6.29. In the experimental outcomes show that sparse point clouds also give a remarkable performance with the proposed solution up to 40 meters range. On the other hand, Table 6.6 represents the overall inference time for positioning vehicles in 3D space. Presented results depicts that inference speed is inversely proportional to the point clouds density for involved in less computation than denser point clouds.

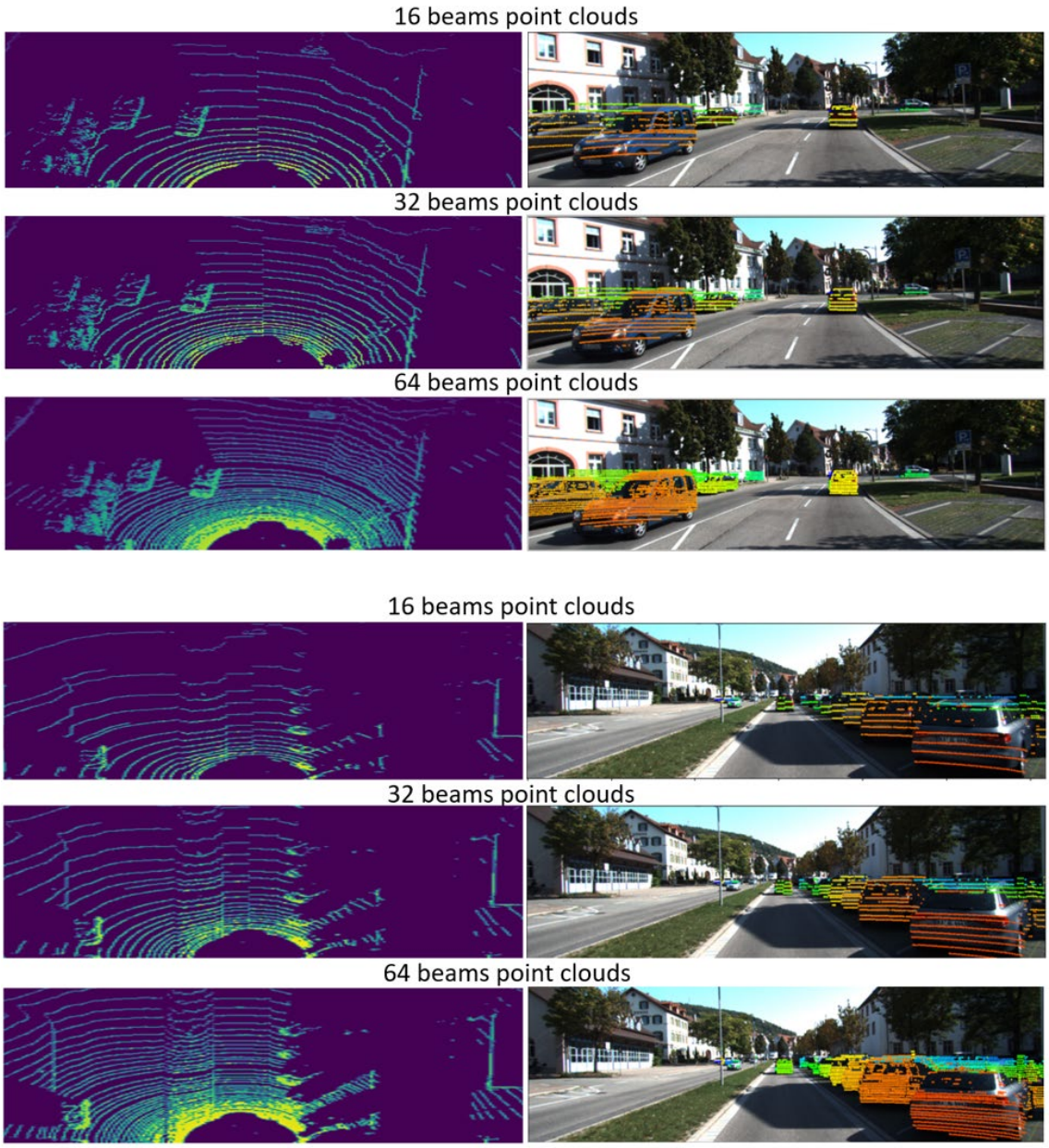


Fig. 6.28: Two sample frames of KITTI point clouds presented into 64 beams, 32 beams and 16 beams density LiDARs form for model testing. Left images are the raw point clouds in BEV presentation, and the right images are projected point clouds onto image coordinates in 2D detection windows with ground points removed. Note: from (Mehtab et. al., 2021) In *ICVNZ*

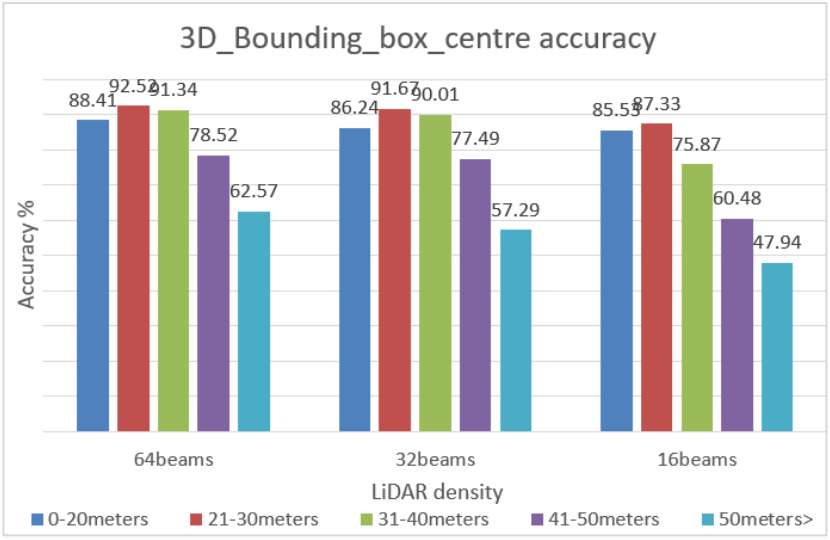


Fig. 6.29: The evaluations of the proposed model performance by using 64, 32 and 16 beam density point clouds over distances based on the KITTI dataset, Note: from (Mehtab et. al., 2021) In *ICVNZ*

Table 6.6: Analysis of inference speed of the proposed model with 64, 32, and 16 beams point clouds based on the KITTI dataset.

Beam Density	Inference Time (seconds)
64	.224
32	.212
16	.206

Note: from (Mehtab et. al., 2021) In ICVNZ

6.4.5 Summary

In this section, we summarize the 3D vehicle detection results of the proposed MobileNetV2-based DNN by using camera RGB images and LiDAR point clouds based on 2D detection results. In the proposed algorithm has exploited camera images to predict the 3D box size and orientation of cars in 2D detection windows and LiDAR point clouds to estimate their central distances from AV in real-world coordinates. In the most

promising aspect of this solution is that the algorithm does not heavily rely on the density of point clouds but leverages the continuous horizontal streams of LiDAR emission that are always present with sparse point clouds (Mehtab et. al., 2021).

Although the algorithm doesn't provide the best accuracy, it offers a cost-effective solution over the existing methods that rely on costly LiDARs to achieve high performance. Fig. 6.30 depicts the reliability of the network results up to 40 meters range, although results fell drastically after 50 meters distances with Velodyne LiDAR. With the latest Velodyne and Luminar latest solid-state LiDAR that provides sparse point clouds but up to 250 meters range accuracy with 120 degrees horizontal and 30 degrees vertical field of view (Aijazi et. al., 2020). Using these new solid-state LiDAR, results are expected to show high improvements over range constraint with the proposed algorithm. Two or more LiDAR can be assembled in the system to make the solution robust.

Chapter 7

Conclusion and Future Work

In this chapter, we have summarized the subject and methods in this course of research with the outcomes. In the light is shed on the research areas that give a new direction according to the results achieved and the insufficiency of the experiment, preparing for future work.

AVs have relied on multiple sensors, conventional machine learning, computer vision methods, advanced deep learning-based algorithms, and powerful GPUs to improve the autonomous driving experience. Although multi-perceptron neural networks have been known for a long time, the parallel processing power of GPUs has supported the parallel execution of the neural networking, thereby making the solutions practically acceptable in real-time scenarios. GPU works with thousands of cores designed to handle multiple tasks simultaneously.

On the other hand, recent libraries of neural networks keep image-based inputs, outputs, and intermediate parameters information in the form of multidimensional tensors. In the software architecture of these libraries takes advantage of GPU parallelism to make execution faster and allow implementation in real-time scenarios. In the most significant development in the field of computer vision based on neural network practices is CNN. CNN is a class of deep learning models for processing data that has grid patterns, such as images, that iteratively learns spatial features with the help of progressive abstraction at every layer by using kernel patterns. In the proposed solutions, PyTorch and Tensorflow based CNN libraries are employed, which are low-level APIs that perfectly fit with the python platform. In the following sections, the conclusions are drawn based on the results achieved by using proposed 2D and 3D detection methods.

7.1 2D Road Scene Perception

In 2D road scene perception, we explored the power of flexible neural networks based on CNN (Mehtab et. al., 2021). In the proposed architecture provides a unified framework for the detection of car, pedestrian and cyclist objects. In the network model has been coded on PyTorch-based framework, which is a high-performance library with optimization support for scientific computing operations using Python tools (Chollet, 2017). In the architecture is designed by using a baseline CNN model that can be finalized in size based on the *depth_multiple* and *width_multiple* attributes. In the main idea behind

the proposed solution is to allow flexibility in model selection and investigate the results based on the different number of layers and channels in the CNN. In the proposed architecture is designed using CSPNet as the building block in the network, which has been successfully exploited in YOLO latest versions. In the CSPNet not only carries forward the spatial gradient information using skip connections but also controls the enormous amount of parameters based on the partial transition blocks.

Influenced by YOLO, FlexiNet architecture is comprised of backbone and head modules. At the features extraction level, three-stage extraction is drawn to retrieve features from different levels of abstraction of convolution layers. In the head part of the network is based on a feature pyramid network to detect multiscale objects at multiple levels of features. In the detection layer works for the promising objects using different sized anchor boxes. An auto-anchor generation method is exploited in the proposed algorithm based on *k*-means clustering to generalise the network for different datasets. Multilevel detection resulted in multiple detections for the same object that was finally removed by using a non-max suppression algorithm that keeps the ones with high object scores.

Result evaluation is conducted in comparison with the state-of-the-art detectors such as YOLOv4, EffiecientDet-B2, and Faster R-CNN. FlexiNet is leading other models in terms of recall, inference rate, and storage size with the scaling factors of width at 0.50 and depth at 0.33 as shown in Fig. 6.5, exploiting the minimum hardware with batch-size 16, SGD with momentum optimizer and DIoU loss function, based on Tesla P100-PCIE-GPU with 16 GB memory. However, YOLOv4 has shown the highest precision pertaining to fewer false positives, however, for an AV prediction, an extra non-existing object is less hazardous than non-predicting an existing one at all. In the proposed model outweighs other algorithms by achieving an average recall of 95.86% based on the detection dataset as presented in Table 6.3, where other algorithms faced differentiating pedestrians and cyclists, FlexiNet has attained good accuracy and real-time performance.

7.2 2D Vehicle Detection

In 2D vehicle detection, road scene perception problem is narrowed down to vehicles only with the aim of further using these results in 3D vehicle detection. 2D vehicle detection, the baseline architecture of FlexiNet was investigated with the dataset-based on vehicle objects and hardware available (Mehtab and Yan, 2022). In the architecture was investigated at different widths and depths to achieve optimized results. In the scaling results showed that increasing the width and depth to a great extent naively is not always the best solution to achieve the optimum accuracy and leads to high computational complexity with an extra storage cost. Based on the TESLA X GPU with total_memory approximating 15GB, fixing a batch size of 16, the network showed the best outcomes at 0.55 *depth_multiple* and 0.55 *width_multiple* as shown in Fig. 6.15. In the network was further finetuned by using different optimizations.

The results showed superior performance in comparison to the state-of-the-art visual object detection algorithms with SGD optimizer, Hardswish activation and DIoU bounding box regression loss. We have attained 94.5% mAP based on the KITTI dataset while 97.5% on the Waymo dataset using 0.50 IoU threshold values as shown in Section 6.3.1. With FlexiNet converging to 1.8% loss in 600 epochs, YOLOv4 achieved a comparable loss in 2,600 epochs, Faster R-CNN shows continuous improvements in the loss but could not go beyond 12.0% as shown in Section 6.3.2. There was not much variance between YOLOv4 and FlexiNet in the final precision; however, there was a rich assortment of computational difficulties while executing YOLOv4.

7.3 3D Vehicle Detection

The proposed 3D vehicle detection is a simple yet effective method for predicting 3D bounding boxes of cars and vans based on the information received from front view of RGB images and top-mounted LiDAR 3D point clouds (Mehtab et. al., 2021). In the

proposed solution utilizes the basic fact that the 3D centre of a car is the translation of the 2D centre in the world coordinates and takes advantage of well-developed 2D vehicle detection. In the proposed work is a two-step solution that firstly regresses the size of 3D bounding box and the orientation of vehicles by using a MobileNetV2-based DNN. Subsequently, the 3D centres are estimated by using LiDAR point clouds. Basic trigonometric geometry is applied to exploit car predicted pose, size and orientation of 3D bounding box to make a distance estimation using 3D point clouds. In the algorithm takes advantage of horizontal streams of sparse point clouds and finds the closest points and centre points in world coordinates form for each car in the AV field of view. In order to tackle the occlusion problem, a relative distancing between occluded and front cars is exploited based on the available closest points.

For result analysis, KITTI point clouds were transformed into three density patterns of 64, 32, and 16 beams point clouds as shown in Section 6.4.3. In the performance was further tested by using the Waymo dataset that contains extreme weather and challenging night scenes. Based on the experiments, we found that the performance is most promising in the 20~50 meters range using camera images, achieved 85.7% accuracy of size and 79.7% accuracy of orientation of 3D bounding boxes based on the KITTI dataset as shown in Fig. 6.23. On the other hand, the network obtained even better results on the Waymo dataset, namely, 99.50% accuracy in terms of orientation and 84.90% accuracy for box size prediction as shown in Fig. 6.24. In this work contributes most in the direction of achieving a low-cost solution for 3D vehicle prediction in autonomous driving using sparse LiDAR and camera sensors.

7.4 Future Work

Based on the results achieved, the scope of further improvement was noticed in the proposed methods that can be focused on in future for increasing the accuracy of detection ahead. In the following sections, the areas are recommended aiming at the research

directions for progressing in the given field of research.

7.4.1 Improving precision of 2D Object Detection

Based on the results achieved by using FlexiNet architecture, we have successfully attained desired recall over different classes, i.e., car, pedestrian, and cyclist. However, the models detect false-positive numbers that are misleading and demand refinement in the existing network. In future, we will work on improving precision in the proposed network. At present, the backbone network is carrying forward with unnecessary information that would have been better dropped off. We can have changes in the feature extraction network with additional pooling or kernel-size/stride change in the convolutional layers. Investigation can also be performed with the shortcut connections in the FPN at the detection level.

7.4.2 Perspective Transformation for Long-range Objects

Although LiDAR gives accurate distance estimation, its performance diminishes as the objects stand farther from it. In the proposed 3D detection results have shown high accuracy in the range of 40 meters; however, mathematical perspective transformation can take over as an alternative measure for long-distance objects where laser rays become sparse and not strong enough to estimate object distance directly. Perspective transformation is a simple and intuitive approach based on the relative distancing and size of visual objects. In order to use the perspective transformation, we need to have prior information on the accurate distance of objects in a similar pose and in a closed range of AV where laser rays are capable of generating accurate results. To make it simple, the pose standards can be determined in advance to estimate the distance of long-range objects.

7.4.3 3D Road Scene Perception

In the currently proposed methods, 3D detection is restrained to vehicles only, excluding pedestrians, cyclists and scooterists. In future work, we would include these vulnerable

road users also in 3D road scene perception. In the present proposal, a MobileNetV2-based feature extractor was deployed for estimating size and orientation of 3D bounding box. However, with the CSPNet-based backbone network, we would expect to retrieve more delicate features, thereby improve the robustness of 3D bounding box sizes. addition, the proposed solution would effectively comply with promising solid-state LiDAR, which claims to give accurate results within 120 meters of range with spars point clouds at a reasonable cost (Aijazi et. al., 2020). In the experiment could be performed using these low-cost LiDARs and synchronized RGB images to verify the findings.

7.4.4 Integrating 2D and 3D Networks Together

Extensive research work can be completed for merging 2D and 3D vehicle detection architecture to exploit the same features of images for both purposes. As we know that CSPNet has proved itself in retrieving high level and fine-grained features of images, networks integration would benefit the results in either way. In addition, networks integration will accelerate 3D vehicle detection speed immensely. PyTorch-based framework would provide much flexibility for network designing and optimization.

The proposed research work would not only improve the accuracy of results but also accelerate its detection speed and resources usability. We look forward to much improved results by using the latest solid-state LiDARs to validate the efficiency of the proposed method.

7.5 A Lookback on What Went Wrong

In the whole course of the study, multiple approaches were tested for attaining good detection results; however, they would have been dropped off for not fulfilling expectations.

We conducted an investigation of the YUV colour coding scheme over RGB. It encodes a colour image taking human perception into account (Podpora et al., 2014).

Human eyes have minor sensitivity to colours, while the accuracy of the brightness or luminance information has a substantial impact on the image recognition, Where Y stands for the luminance component, U and V are the chrominance (colour) component. In the YUV colour scheme, U and V signals are significantly compressed than Y luminance. YUV can be more illumination independent, which means that the shadows and illuminations changes don't affect it much. In order to work with YUV, we firstly converted RGB images provided by KITTI into YUV using Python tools before feeding them into the network for any processing. However, the proposed DNN results did not show up any major benefits at the expense of an additional computation cost, so the idea was dropped.

Spatiotemporal images have been employed in human action recognition and data mining tasks successfully in many applications. We also worked on the investigation of spatiotemporal images for the fact that foreground objects would have strong spatial changes as compared to background objects, which could result in the better training of the neural network in the detection of foreground objects. However, our experimental results didn't present any additional benefits for object detection based on the KITTI dataset; in fact, that resulted in excessive computational and storage costs. One of the reasons for this outcome might be large-time laps between consecutive frames present in the KITTI dataset.

2D road scene perception was investigated by using front-view projection images of LIDAR point clouds; however, the results led to decay in precision compared to camera RGB images. Experiments were also performed to test the early fusion performance based on our network; LiDAR point clouds were projected on image coordinates using affine transformation and passed into the proposed DNN; however, this also showed downside performance with computational overheads.

Influenced by the high reflectance of cars surfaces, the intensity values of LiDAR point clouds were also considered for vehicle detection; however, the results were not

very promising and suggested the presence of cars in mat colours as well and thereby misleading to the detection.

7.6 Novelty and Advancement in This Thesis

In terms of the novelty of this thesis, we found that a fixed DNN size does not tend to achieve the best results on any hardware and dataset, however a fine tuning of net size based on available resources is a promising approach to extract optimized results. In the proposed FlexiNet detection architecture that is based on the CSPNet backbone structure, has shown exclusive results on car, pedestrian, and cyclist road scene objects in a single stream, where the state-of-the-art networks faced difficulty in distinguishing pedestrians and cyclists, FlexiNet has shown promising results that are seen in Section 6.2.3.

In this thesis, we present a simple and effective solution to move from 2D to 3D vehicle detection. It primarily acknowledges the fact that a 2D centre of a vehicle is the projection of 3D world coordinates on an image. If we have 2D centres of vehicles on an image then through reverse calculation, we can retrieve 3D centre coordinates also. In the proposed solution, firstly, we predicted the 2D centres of vehicles by using FlexiNet and in the late section, available LiDAR point clouds were projected on the 2D detection windows to map their 3D centres. To move from vehicle surface to inner 3D centres, trigonometric geometry concepts has been exploited based on the predicted dimensions, orientations and pose of vehicles using images as presented in Table 5.1.

Based on the literature reviewed, the existing 3D vehicle detection methods have relied on expensive 64 beam dense LiDAR point clouds. On the other hand, the proposed solution looks for the optimum solution using sparse point clouds and minds the pattern of LiDAR point clouds that always sweeps in the horizontal direction as shown in Fig. 6.28. In sparse LiDAR point clouds, we see the gap in the vertical plane, however, the horizontal points are always continuous. In the proposed solution can successfully retrieve 3D centres of vehicles up to 40 meters in range by using sparse point clouds as

shown in Section 6.4.3.

Reviewed literature depicts that most 3D object detection has been tested on the benchmark KITTI dataset, despite having a high level of complexity KITTI dataset lacks night and rainy scenes and for practical reasons, it is necessary to train and test AV detection applications for such scenarios. In this thesis, another open dataset namely Waymo is taken into account which possesses many nights and foggy scenes in addition to the variety of environments and promotes the researchers for training the network with different environments.

References

Ahmed, S., Huda, M. N., Rajbhandari, S., Saha, C., Elshaw, M., & Kanarachos, S. (2019). Pedestrian and cyclist detection and intent estimation for autonomous vehicles: A Survey. *Applied Sciences 9(11),* pp.1–38, Switzerland

Aijazi, A., Malaterre, L,Trassoudaine, L.,Checchin, P. (2020) Systematic evaluation and characterization of 3D solid state lidar sensors for autonomous ground vehicles. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43, pp.199-203.

Al-Tairi, Z., Rahmat, R., Saripan, M. & Sulaiman, P. (2014). Skin segmentation using YUV and RGB color spaces. *Journal of Information Processing Systems, 10(2),* pp.283-299.

Arnold, E., Al-Jarrah, O.Y., Dianati, M., Fallah, S., Oxtoby, D. & Mouzakitis, A. (2019). A survey on 3D object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems, 20(10),* pp.3782-3795.

Asvadi, A., Garrote, L., Premebida, C., Peixoto, P., & Nunes, U. (2018). Multimodal vehicle detection: Fusing 3D-LiDAR and color camera data. *Pattern Recognition Letters,* 115, pp.20-29.

Atapour-Abarghouei, A., & Breckon, T. (2018). Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2800-2810.

Avenash, R., & Viswanath, P. (2019). Semantic segmentation of satellite images using a modified CNN with hard-swish activation function. *VISIGRAPP (4: VISAPP),* pp. 413-420.

Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., & Baskurt, A. (2011). Sequential deep learning for human action recognition. *International Workshop on Human Behavior Understanding.* pp. 29-39, Springer, Berlin, Heidelberg

Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., Garcia, F., & De La Escalera, A. (2018). BirdNet: A 3D object detection framework from lidar information. *International Conference on Intelligent Transportation Systems (ITSC).* pp. 3517-3523. Maui, Hawaii, USA.

Bochkovskiy, A., Wang, C. Y., & Liao, H. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arxiv preprint arXiv:*2004.10934.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *COMPSTAT, Springer*, pp.177–186.

Brazil, G. & Liu, X. (2019). M3D-RPN: Monocular 3D region proposal network for object detection. *IEEE/CVF International Conference on Computer Vision*. pp. 9287-9296.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., & Beijbom, O. (2020). Nuscenes: A multimodal dataset for autonomous driving. *IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 11621-11631. Seattle, WA, USA.

Caltagirone, L., Mauro B., Lennart S., & Mattias W. (2019). LIDAR–camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems,* 111, pp.125–31.

Caltagirone, L., Samuel, S., Lennart, S., & Mattias, W. (2017). Fast LiDAR-based road detection using fully convolutional neural networks. *IEEE Intelligent Vehicles Symposium (iv)* , pp.1019–24.

Cao, J., Song, C., Song, S., Peng, S., Wang, D., Shao, Y. & Xiao, F. (2020). Front vehicle detection algorithm for smart car based on improved SSD model. *Sensors, 20(16),* pp. 4646.

Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C. & Chateau, T. (2017). Deep manta: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 2040-2049.

Chen, X., Ma, H., Wan, J., Li, B. & Xia, T. (2017). Multi-view 3D object detection network for autonomous driving. *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1907-1915.

Chollet, F. (2017, July). Xception: Deep learning with depthwise separable convolutions, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258.

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S. & Urtasun, R., (2016). Monocular 3D object detection for autonomous driving. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147-2156.

Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S., & Urtasun, R. (2017). 3D object proposals using stereo imagery for accurate object class detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*40(5):1259-72.

Cho, H., Rybski, P. E., & Zhang, W. (2010). Vision-based bicyclist detection and tracking for intelligent vehicles. *IEEE Intelligent Vehicles Symposium*, pp. 454–461. USA.

Choi, D., Shallue, C.J., Nado, Z., Lee, J., Maddison, C., & Dahl, G. (2019). On empirical comparisons of optimizers for deep learning. arxiv preprint arXiv:1910.05446.

Chollet, F. (2017). Deep Learning with Python. *Simon and Schuster.*

Condat, R., Rogozan, A. & Bensrhair, A. (2020). GFD-Retina: Gated fusion double RetinaNet for multimodal 2D road object detection. *IEEE International Conference on*

Intelligent Transportation Systems (ITSC). pp. 1-6. Rhodes, Greece

Cross, S., Harrison, R., & Kennedy, R. (1995). Introduction to neural networks. In the *Lancet*, 346(8982), pp.1075-1079.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 886–93.

De Silva, V., Roche, J., & Kondoz, A. (2018). Robust fusion of LiDAR and wide-angle camera data for autonomous mobile robots. *Sensors,* 18(8), 2730.

Dieterle, T., Particke, F., Patino-Studencki, L. & Thielecke, J. (2017). Sensor data fusion of LIDAR with stereo RGB-D camera for object tracking. *IEEE Sensors.* pp. 1-3. IEEE.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. *Conference on Robot Learning.* PMLR, pp. 1-16.

Du, K. L., & Swamy, M. N. (2006). Neural Networks in A Softcomputing Framework. *Springer*

Dumoulin, V. & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arxiv preprint arXiv:1603.07285.*

Elfwing, S., Uchibe, E. & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107, pp.3-11.

Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H. & Posner, I. (2017). Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks. *IEEE International Conference on Robotics and Automation (ICRA).* pp. 1355-1361. Singapore

Felzenszwalb, P. F. , Girshick, R. B. , McAllester, D. & Ra- manan. D. (2010). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern*

Analysis and Machine Intelligence, 32(9): pp. 1627–1645

Feng, D., Rosenbaum, L., & Dietmayer, K. (2018). Towards safe autonomous driving: Capture uncertainty in the deep neural network for LiDAR 3D vehicle detection. *IEEE Confere*nce on Intelligent Transportation Systems , ITSC. pp. 3266–73.

Freeman, W. T., & Roth, M. (1995). Orientation histograms for hand gesture recognition. *Proceedings of International Workshop on Automatic Face and Gesture Recognition.,*12, pp. 296-301.

Gählert, N., Wan, J. J., Weber, M., Zöllner, J. M., Franke, U. & Denzler, J. (2019). Beyond bounding boxes: Using bounding shapes for real-time 3D vehicle detection from monocular RGB images. *IEEE Intelligent Vehicles Symposium (IV).* pp. 675-682.

Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340-4349.

Garanderie, D., Abarghouei, G., & Breckon, T. (2018). Eliminating the blind spot: Adapting 3D object detection and monocular depth estimation to 360 panoramic imagery. *European Conference on Computer Vision (ECCV).* pp. 789-807.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. In the *International Journal of Robotics Research*, 32(11), pp.1231-1237.

Ghosh, S., Amon, P., Hutter, A., & Kaup, A. (2017). Reliable pedestrian detection using a deep neural network trained on pedestrian counts. *IEEE International Conference on Image Processing (ICIP),* pp. 685–689. Beijing, China

Girshick, R. (2015). Fast R-CNN. IEEE International Conference on Computer Vision (ICCV), pp.1440–48.

Green, M. (2000). How long does it take to stop? Methodological analysis of driver

perception-brake times. *Transportation Human Factors*, 2(3), pp. 195-216.

Gu, Q., Yang, J., Kong, L., Yan, W., & Klette, R. (2017). Embedded and real-time vehicle detection system for challenging on-road scenes. *Optical Engineering* 56 (6), pp. 63102

Gupta, O. (2018). Unlocking the Potential of Neural Networks in Resource and Data Constrained Environments (Doctoral Thesis), *Massachusetts Institute of Technology, USA.*

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 770-778.

He, K., Zhang,X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 37(9). pp. 1904–1916.

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. *Neural Networks for Perception*, pp. 65-93. Academic Press

Hemmati, M. (2019). Adaptive Embedded Systems for Autonomous Vehicles (Doctoral Dissertation)*, The University of Auckland*, Auckland, New Zealand.

Hochreiter, S. (1998). In the vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(2): pp.107–116.

Hou, Y.L., Song, Y., Hao, X., Shen, Y., Qian, M. & Chen, H. (2018). Multispectral pedestrian detection based on deep convolutional neural networks. *Infrared Physics and Technology,* 94, pp.69-77.

Hu, Q., Wang, P., Shen, C., Van den Hengel, A. & Porikli, F. (2017). Pushing the limits of deep CNNs for pedestrian detection. *IEEE Transactions on Circuits and Systems for Video Technology,* 28(6), pp.1358-1368.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. (2017). Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4700-4708.

Ian, G., Yoshua, B., Aaron, C. (2016). Deep Learning. In the *MIT Press.*

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50×fewer parameters and < 0.5 MB model size. *arxiv preprint arXiv:1602.07360.*

Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*. pp. 448-456.

Jocher, G., Stoken, A., Borovec, J. (2020). YOLOv5. ultralytics/yolov5: 3983579 3.0 (Zenodo. http://doi.org/10.5281).

Kang, J.K., Hong, H.G., & Park, K.R. (2017). Pedestrian detection based on adaptive selection of visible light or far-infrared light camera image by fuzzy inference system and convolutional neural network-based verification. *Sensors, 17(7),* p.1598.

Kim, K. J., Pyong K. K., Yun, S. C., & Doo, H. C. (2019). Performance enhancement of YOLOv3 by adding prediction layers with spatial pyramid pooling for vehicle detection. *IEEE International Conference on Advanced Video and Signal-Based Surveillance.* pp. 1–6.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing* Systems, pp.1097–1105.

Krogh, A. & Hertz, J. (1992). A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4, pp. 950--957.

Ku, J., Mozifian, M., Lee, J., Harakeh, A. , & Waslander, S., (2018). Joint 3D proposal generation and object detection from view aggregation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 1-8.

Kuutti, S., Fallah, S., Katsaros, K., Dianati, M., Mccullough, F., & Mouzakitis, A. (2018). A survey of the state-of-the-art localization techniques and their potential for autonomous vehicle applications. *IEEE Internet of Things Journal, 5(2),* pp.829-846.

Lantos, B., & Márton, L. (2010). Nonlinear Control of Vehicles and Robots, *Springer Science and Business Media.*

Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, pp. 12697-12705.

Li, B. (2017). 3D fully convolutional network for vehicle detection in point cloud. *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS),* pp. 1513–18.

Li, B., Zhang, T., & Xia., T. (2016). Vehicle detection from 3D LiDAR using fully convolutional network. *Robotics: Science and Systems,*

Li, C., Dan, S., Ruofeng, T., & Min, T. (2019). Illumination-aware Faster R-CNN for robust multispectral pedestrian detection. *Pattern Recognition, 85,* pp.161–171.

Li, G., Xie, H., Yan, W., Chang, Y., & Qu, X. (2020). Detection of road objects with small appearance in images for autonomous driving in various traffic situations using a deep learning based approach. *IEEE Access, 8,* pp.211164-211172.

Li, P., Chen, X., & Shen, S. (2019). Stereo R-CNN based 3D object detection for

autonomous driving. *IEEE/CVF Conference on Computer Vision and Pattern Recognition.* pp. 7644-7652.

Liang, M., Yang, B., Wang, S., & Urtasun, R. (2018). Deep continuous fusion for multi-sensor 3D object detection. *European Conference on Computer Vision (ECCV),* pp. 641-656.

Liang, M., Yang, B., Chen, Y., Hu, R., & Urtasun, R. (2019). Multi-task multi-sensor fusion for 3D object detection. *Proceedings of the IEEE/CV*F *Conference on Computer Vision and Pattern Recognition*. pp. 7345-7353.

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 2117-2125.

Litman, T., (2020) Autonomous vehicle implementation predictions: Implications for transport planning, *Traffic Technology International*, pp. 36-42.

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 8759-8768.

Liu, W., Sun, J., Li, W., Hu, T., & Wang, P. (2019). Deep learning on point clouds and its application: A survey. *Sensors*, *19(19*), pp.4188.

Liu, X., Neuyen, M., & Yan, W. (2019). Vehicle-related scene understanding using deep learning. In *Asian Conference on Pattern Recognition, Springer*, pp. 61-73.

Liu, Y., Cao, S., Lasang, P., & Shen, S. (2019). Modular lightweight network for road object detection using a feature fusion approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51(8),* pp. 4716-4728.

Luo, Q., Ma, H., Tang, L., Wang, Y., & Xiong, R. (2020) 3D-SSD: Learning hierarchical

features from RGB-D images for amodal 3D object detection. *Neurocomputing,* 378, pp.364-374.

Manhardt, F., Kehl, W., & Gaidon, A. (2019) ROI-10D: Monocular lifting of 2D detection to 6D pose and metric shape. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 2069-2078.

Maas, A., Hannun, A., & Ng, A. (2013). Rectifier nonlinearities improve neural network acoustic models. *International Conference of Machine Learning, 30(1).* Pp. 3.

Mehtab, S., & Yan, W. Q. (2022). Flexible neural network for fast and accurate road scene perception. *Multimedia Tools and Applications, 81(5),* pp.7169-7181.

Mehtab, S., Sarwar, F., & Yan, W.Q. (2021). FlexiNet: Fast and accurate vehicle detection for autonomous vehicles. *International Conference on Control and Computer Vision*, pp. 43-49. Macau, China.

Mehtab, S., Yan, W.Q., & Narayanan, A. (2021). 3D vehicle detection using cheap LiDAR and camera sensors. *International Conference on Image and Vision Computing New Zealand (IVCNZ),* pp. 1-6.

Minemura, K., Hengfui L., Abraham M., & Shinpei K. (2018). LMNet: Real-time multiclass object detection on CPU using 3D LiDAR. *Asia-Pacific Conference on Intelligent Robot Systems,* pp. 28--34.

Misra, D. (2019). Mish: A self regularized non-monotonic neural activation function*. arXiv preprint arXiv:1908.08681, 4(2),* pp.10-48550.

Mita, T., Kaneko, T., & Hori, O. (2005). Joint haar-like as features for face detection. *IEEE International Conference on Computer Vision,* 2, pp. 1619-1626.

Mitchell, H. (2007). Multi-Sensor Data Fusion: An Introduction. *Springer Science & Business Media.*

Mousavian, A., Anguelov, D., Flynn, J., & Kosecka, J. (2017). 3D bounding box estimation using deep learning and geometry. *IEEE conference on Computer Vision and Pattern Recognition.* pp. 7074-7082.

Mo, N., & Yan, L. (2020). Improved Faster R-CNN based on feature amplification and oversampling data augmentation for oriented vehicle detection in aerial images. *Remote Sensing 12(16),* pp. 2558

Müller, M., Casser, V., Lahoud, J., Smith, N., & Ghanem, B. (2018). Sim4CV: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision, 126(9),* pp.902-919.

Nair, V., & Hinton, G. (2010). Rectified linear units improve restricted Boltzmann machines. *International Conference on Machine Learning*.

Neuhold, G., Ollmann, T., Rota Bulo, S., & Kontschieder, P. (2017). In the mapillary vistas dataset for semantic understanding of street scenes. *IEEE International Conference on Computer Vision.* pp. 4990-4999.

Ohzeki, M., Okada, S., Terabe, M., & Taguchi, S. (2018). Optimization of neural networks via finite-value quantum fluctuations. *Scientific Reports*, 8(1), pp.1-10.

Pascanu, R., Tomas M., & Yoshua B. (2013). On the difficulty of training recurrent neural networks razvan. *Phylogenetic Diversity: Applications and Challenges in Biodiversity Science (2).* pp. 41–71.

Pelletier, C., Webb, G.I., & Petitjean, F. (2019). Temporal convolutional neural network for the classification of satellite image time series. *Remote Sensing,* 11(5), pp.523.

Pfeiffer, D., & Franke, U. (2010). Efficient representation of traffic scenes by means of dynamic stixels. *IEEE Intelligent Vehicles Symposium.* pp. 217-224.

Pham, C. & Jeon, J. (2017). Robust object proposals re-ranking for object detection in

autonomous driving using convolutional neural networks. *Signal Processing: Image Communication*, *53*, pp.110-122.

Podpora, M., Korbas, G. & Kawala-Janik, A. (2014). YUV *vs* RGB-Choosing a color space for human-machine interaction. *FedCSIS (Position Papers)*, pp. 29-34.

Prechelt, L. (2012). Early stopping - but when? *Neural networks: Tricks of the trade*. pp. 55-69. Springer, Berlin, Heidelberg.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems, 30.*

Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum PointNets for 3D object detection from RGB-D data. *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 918–27.

Qi, C.R., Su, H., Mo, K. & Guibas, L. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 652-660.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms acknowledgements. *Neural Networks 12(1).* pp. 145–151.

Qiu, S., Xu, X., & Cai, B., (2018). FReLU: Flexible rectified linear units for improving convolutional neural networks. *International Conference on Pattern Recognition (ICPR).* pp. 1223-1228.

Ramachandran, P., Zoph, B., & Le, Q. (2017). Searching for activation functions. *arxiv preprint arXiv:1710.05941.*

Redmon, J., & Farhadi, A., (2017). YOLO9000: Better, faster, stronger. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 7263-7271.

Redmon, J., & Farhadi, A. (2018). YOLOv3 : An Incremental Improvement. *arxiv preprint arXiv:1804.02767.*

Ren, S., Kaiming, H., Girshick, R., & Sun., J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39(6).* pp. 1137–1149.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized interSection over union: A metric and a loss for bounding box regression. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 658-666.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention.* pp. 234-241.

Ruder, S. (2016). An overview of gradient descent optimization. *arxiv preprint arXiv*:*1609.04747.*

Sang, J., Wu, Z., Guo, P., Hu, H., Xiang, H., Zhang, Q., & Cai, B. (2018). An improved YOLOv2 for vehicle detection. *Sensors, 18(12),* pp.4272.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018) MobileNetV2: Inverted residuals and linear bottlenecks. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 4510-4520.

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization?. *Advances in neural information processing systems,* 31.

Shah, M., & Rupal, K. (2018). Object detection using deep neural networks. *International Conference on Intelligent Computing and Control Systems*, pp. 787–790.

Shelhamer, E., Jonathan, L., & Trevor, D. (2017). Fully convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39(4).* pp. 640–51.

Shen, Y., & Yan, W. Q. (2018). Blind spot monitoring using deep learning. *International Conference on Image and Vision Computing.* pp. 1-5.

Shorten, C., & Khoshgoftaar., T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6(1).* pp. 1-48

Shi, S., Wang, X. & Li, H. (2019). PointRCNN: 3D object proposal generation and detection from point cloud. *IEEE/CVF Conference on Computer Vision And Pattern Recognition,* pp. 770-779.

Simon, M., Stefan M., Karl A., & Horst, M. (2018). Complex-YOLO: An Euler-region-proposal for real-time 3d object detection on point clouds. *European Conference on Computer Vision (ECCV) Workshop*, pp. 197–209.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv*:*1409.1556.*

Sivaraman, S., & Trivedi, M. (2013). Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Transactions on Intelligent Transportation Systems, 14(4),* pp.1773-1795.

Snoek, C., Worring, M. & Smeulders, A. (2005). Early versus late fusion in semantic video analysis. *ACM International Conference on Multimedia,* pp. 399-402.

Song, H., Choi, I.K., Ko, M.S., Bae, J., Kwak, S., & Yoo, J. (2018). Vulnerable pedestrian detection and tracking using deep learning. *International Conference on Electronics, Information, and Communication (ICEIC).* pp. 1-2.

Song, S., Lichtenberg, S.P., & Xiao, J. (2015). Sun RGB-D: A RGB-D scene understanding benchmark suite. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 567-576.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014).

Dropout: A simple way to prevent neural networks from overfitting. In the *Journal of Machine Learning Research, 15(1),* pp.1929-1958.

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3D shape recognition. *IEEE International Conference on Computer Vision.* pp. 945-953.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., ... & Anguelov, D. (2020). Scalability in perception for autonomous driving: Waymo open dataset. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 2443–2451.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 1-9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *In IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2818-2826.

Tan, M., Pang, R., & Le, Q. (2020). EfficientDet: Scalable and efficient object detection. *In IEEE/CVF Conference on Computer Vision and Pattern Recognition.* pp. 10781-10790.

Thuan, D. (2021). Evolution Of YOLO algorithm and YOLOv5: The State-of-the-Art Object Detection Algorithm (Bachelor's Thesis). *Oulu University of Applied Sciences.*

Tian, W., Lauer, M. (2015). Fast and robust cyclist detection for monocular camera systems. International Joint Conference on Computer Vision Imaging and Computer Graphics Theory and Applications (VISIGRAPP), pp. 11–14.

Tourani, A., Soroori, S., Shahbahrami, A., Khazaee, S., & Akoushideh, A. (2019). A robust vehicle detection approach based on Faster R-CNN algorithm. *International Conference on Pattern Recognition and Image Analysis (IPRIA),* pp. 119-123. Tehran,

Iran

Tumas, P., Jonkus, A., & Serackis, A. (2018). Acceleration of HOG based pedestrian detection in FIR camera video stream. In the *Open Conference of Electrical, Electronic and Information Sciences (eStream).* pp. 1-4.

Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G., & Chervyakov, N. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177, pp.232-243.

Vielzeuf, V., Lechervy, A., Pateux, S., & Jurie, F. (2018). Multilevel sensor fusion with deep learning. *IEEE Sensors Letters, 3(1),* pp.1-4.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*.

Wang, X., Han, T., & Yan, S. (2009). An HOG-LBP human detector with partial occlusion handling. *IEEE International Conference on Computer Vision,* pp. 32–39.

Wang, C.Y., Liao, H., Wu, Y., Chen, P., Hsieh, J., & Yeh, I. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. pp. 390-391.

Wang, H., Yu, Y., Cai, Y., Chen, X., Chen, L., & Liu, Q. (2019). A comparative study of state-of-the-art deep learning algorithms for vehicle detection. *IEEE Intelligent Transportation Systems, 11(2),* pp.82-95.

Wang, H. (2018). Real-Time Face Detection and Recognition Based on Deep Learning (*Masters Thesis), Auckland University of Technology,* Auckland, New Zealand.

Wang, K., & Zhou, W. (2019). Pedestrian and cyclist detection based on deep neural network Fast R-CNN. *International Journal of Advanced Robotic Systems,* 16(2), p.1-10

Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., & Yang, R. (2019). In the apolloscape open dataset for autonomous driving and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(10*), pp.2702-2719.

Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., & Weinberger, K. (2019). Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. *IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 8445-8453.

Wang, Y., Liu, Z., & Deng, W. (2019). Anchor generation optimization and region of interest assignment for vehicle detection. *Sensors, 19(5),* pp.1089.

Wang, Y., Sun, Y., Liu, Z., Sarma, S., Bronstein, M.M., & Solomon, J. (2019). Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics* (tog*), 38(5),* pp.1-12.

Wang, Z., & Jia, K. (2019). Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* pp. 1742-1749.

Wang, Z., Zhaon, W., & Tomizuka, M. (2018). Fusing bird's eye view LiDAR point cloud and front view camera image for 3D object detection. *IEEE Intelligent Vehicles Symposium (IV),* pp. 1-6.

Wei, P., Cagle, L., Reza, T., Ball, J., & Gafford, J. (2018). LiDAR and camera detection fusion in a real-time industrial multi-sensor collision avoidance system. *Electronics, 7(6),* p.84.

Wei, Y., Tian, Q., Guo, J., Huang, W., & Cao, J. (2019). Multi-vehicle detection algorithm through combining Harr and HOG features. *Mathematics and Computers in Simulation, 155*, pp.130-145.

Weisstein, E. (2004). Affine Transformation. *Mathworld.*

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017): The marginal value of adaptive gradient methods in machine learning. arxiv preprint arXiv:1705.08292.

Wu, T., Tsai, C., & Guo, J. (2017). LiDAR/camera sensor fusion technology for pedestrian detection. In the *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC).* pp. 1675-1678.

Wulff, F., Schäufele, B., Sawade, O., Becker, D., Henke, B., & Radusch, I. (2018). Early fusion of camera and lidar for robust road detection based on U-Net FCN. *IEEE Intelligent Vehicles Symposium (IV)*. pp. 1426-1431.

Xiang, Y., Choi, W., Lin, Y., & Savarese, S. (2017). Subcategory-aware convolutional neural networks for object proposals and detection. *IEEE Winter Conference on Applications of Computer vision (WACV).* pp. 924-933. Santa Rosa, CA, USA

Xiang, Y., Choi, W., Lin, Y., & Savarese, S. (2015). Data-driven 3D voxel patterns for object category recognition. *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1903-1911.

Xin, D., Ang, M. H., Karaman, S., & Rus, D. (2018). A general pipeline for 3D detection of vehicles. *IEEE International Conference on Robotics and Automation.* pp. 3194–3200.

Xu, D., Anguelov, D., & Jain, A. (2018). PointFusion: Deep sensor fusion for 3D bounding box estimation. *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 244-253.

Yamaguchi, K., McAllester, D., & Urtasun, R. (2014). Efficient joint segmentation, occlusion labelling, stereo and flow estimation. *European Conference on Computer Vision.* pp. 756-771.

Yan, W. (2021). Computational Methods for Deep Learning. *Heidelberg: Springer.*

Yan, W. (2019) Introduction to Intelligent Surveillance: Surveillance Data Capture. *Transmission, and Analytics. Springer.*

Yan, Y., Mao, Y., & Li, B. (2018). SECOND: Sparsely embedded convolutional detection. *Sensors, 18(10),* pp.3337.

Yang, B., Liang, M., & Urtasun, R. (2018). HDNet: Exploiting HD maps for 3D object detection. In the *Conference on Robot Learning,* pp. 146-155.

Yang, B., Luo, W., & Urtasun, R. (2018). PIXOR: Real-time 3D object detection from point clouds. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 7652-7660.

Yang, Z., Li, J., & Li., H. (2018). Real-time pedestrian detection for autonomous driving. *International Conference on Intelligent Autonomous Systems*, pp. 9–13.

Ye, Y., Chen, H., Zhang, C., Hao, X., & Zhang, Z. (2020). SARPNET: Shape attention regional proposal network for LiDAR-based 3D object detection. Neurocomputing, 379, pp.53-63.

Yi, H., Shi, S., Ding, M., Sun, J., Xu, K., Zhou, H., Wang, Z., Li, S., & Wang, G. (2020). SegVoxelNet: Exploring semantic context and depth-aware features for 3D vehicle detection from the point cloud. *IEEE International Conference on Robotics and Automation (ICRA),* pp. 2274-2280.

Yingge, H., Ali, I., & Lee, K. (2020). Deep neural networks on chip - A survey. *IEEE International Conference on Big Data and Smart Computing*, pp. 589–92.

Yu, D., Xiong, H., Xu, Q., Wang, J., & Li, K. (2019). Multi-stage residual fusion network for LiDAR-camera road detection. *IEEE Intelligent Vehicles Symposium (IV)* (pp. 2323-2328).

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020) BDD100K: A diverse driving dataset for heterogeneous multitask learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 2636-2645.

Yu, S. L., Westfechtel, T., Hamada, R., Ohno, K., & Tadokoro, S. (2017). Vehicle detection and localization on bird's eye view elevation images using convolutional neural network. *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR),* pp. 102-109.

Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., & Yoo, Y. (2019,). CutMix: Regularization strategy to train strong classifiers with localizable features. *IEEE/CVF International Conference on Computer Vision.* pp. 6023-6032.

Zagoruyko, S., & Komodakis, N., (2016). Wide residual networks. *arxiv preprint arXiv:1605.07146.*

Zakaria, Y., El Munim, H.E.A., Ghoneima, M., & Hammad, S. (2018). Modified HOG based on-road vehicle detection method. *International Journal of Pure and Applied Mathematics, 118(18),* pp.3277-3285.

Zhang, X., Gao, H., Xue, C., Zhao, J., & Liu, Y. (2018). Real-time vehicle detection and tracking using improved histogram of gradient features and Kalman filters. *International Journal of Advanced Robotic Systems, 15(1)*, pp.1-9

Zhang, Y., Qiu, Z., Yao, T., Liu, D., & Mei, T. (2018). Fully convolutional adaptation networks for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition.* pp. 6810-6818.

Zhang, Z., Zhang, M., Liang, Z., Zhao, X., Yang, M., Tan, W., & Pu, S. (2020). MAFF-Net: Filter false_positive for 3D vehicle detection with multi-modal adaptive feature fusion. *arxiv preprint arxiv:2009.10945.*

Zhao, Z.Q., Zheng, P., Xu, S.T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems, 30(11),* pp. 3212-3232.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU loss: Faster and better learning for bounding box regression. *AAAI Conference on Artificial Intelligence, 34(7),* pp. 12993-13000. New York, USA.

Zhou, J., Tan, X., Shao, Z., & Ma, L. (2019). FVNet: 3D front-view proposal generation for real-time object detection from point clouds. *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI),* pp. 1-8.

Zhou, Y., and Oncel, T. (2017). VoxelNet: End-to-End learning for point cloud based 3D object detection. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499.

Zhu, M., Ma, C., Ji, P., & Yang, X. (2021). Cross-modality 3D object detection. *IEEE/CVF Winter Conference on Applications of Computer Vision.* pp. 3772-3781.