Complex YOLOv1:

Real-time 3D Object Detection

A

report submitted in partial fulfillment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Ashish Kumar Singh : 2019BCS-010

Under the Supervision of

Professor Manisha Pattanaik

Department of Electrical and Electronics Engineering

ABV-INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
AND MANAGEMENT GWALIOR

GWALIOR, INDIA

DECLARATION

I hereby certify that the work, which is being presented in the report/thesis, entitled Complex YOLOv1: Real-time 3D Object Detection, in fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering and submitted to the institution is an authentic record of my/our own work carried out during the period January-2022 to May-2023 under the supervision of Professor Manisha Pattanaik. I also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

Dated: **Signature of the candidate**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dated: **Signature of supervisor**

Acknowledgements

I want to express my profound gratitude to my Supervisor, Professor Manisha Pattanaik, for her guidance. She challenged me, help increase my curiosity towards the subject and gave me the complete liberty to experiment with my ideas. The resources provided by her is responsible for my work presented in the project report.

I would also like to thank Nupur Srivastava Ma'am and Alok Tiwari Sir for helping me with their vast knowledge and guiding me at every step.

Finally, I want to express my gratitude to the Institution and my colleagues for their unwavering support in helping me complete this job. Their consistent encouragement helped revive my spirit and concentrate my attention and energy.

Ashish Kumar Singh

Abstract

The number of edge computing devices like Cameras, Mobile Phones have significantly increased in recent years. These devices carry out computation on the go. All of this has been made possible by smaller processors with increased computation power.

Object detection is a fundamental computer vision problem because it can help address many real-world issues and assist in the automation of various tasks. Most 2D object detection algorithms like Faster Region based Convolutional Neural Network(Faster R-CNN) and You Only Look Once(YOLO) cannot be directly extended to 3D. 3D Object Detection is a relatively new field.

The utilization of LiDAR technology for detecting 3D objects is essential for autonomous driving since it directly contributes to understanding the surroundings, which serves as the foundation for making predictions and planning motion. The ability to interpret highly dispersed 3D data in real-time poses a challenging issue for various other fields, such as industrial automation, personal robotics, augmented reality and vehicle automation.

This project proposes changes to Complex YOLO an already existing popular 3D Object Detection algorithm which uses 3D point cloud data collected via LiDAR sensor to predict 3D bounding boxes.

Our work describes a network, a fast and accurate 2D object detector for RGB images followed by orientation prediction to estimate multi-class 3D bounding boxes in a 3 dimensional Cartesian space. We have used Cross Stage Partial Connection Networks in the backbone of our model, and introduced modified versions of Dual Channel Attention Modules and Batch Normalization Channel Attention Modules for feature extraction. Also, Complex YOLO used Leaky ReLU activation function, instead we used Mish Activation Function.

Our work outperforms current leading methods for 3D object detection on KITTI Dataset specifically in terms of accuracy especially for Cars and Cyclist class and have achieved computational speed which makes our model capable of being deployed for Real time 3D Object Detection.

Keywords: 3D Object Detection, Complex YOLO, KITTI Dataset, Mish, 3D Point Clouds, Euler Region Proposal Network, LiDAR

Contents

List of Figures

List of Tables

1

Introduction

Supervised Learning defines learning as a function that can map input data to known target based on new, unseen data based on training performed on a given labelled dataset. In supervised learning, the data set includes both input and output data, and the algorithm learns by identifying patterns and relationships between the input and output data. The algorithm then uses these patterns to make predictions on new data. For example, a supervised learning algorithm can be trained on a dataset of images labeled with the objects they contain, such as "man" or "woman." The algorithm would then learn the features that distinguish cats from dogs and use this knowledge to accurately classify new images. Supervised learning algorithms can be used in a variety of applications, such as image recognition, speech recognition, natural language processing, and predictive modeling. Common algorithms used in supervised learning include decision trees, random forests, neural networks, and support vector machines. Object detection is a supervised learning problem. It tries to identify and locate a particular object instance in an image or video frame. The classified objects(objects of interest) are characterized by bounding boxes. In the case of object detection, the labeled data would include images and the bounding boxes that identify the location of objects in the image. During training, the object detection algorithm uses this labeled data to learn to recognize the features of objects and their locations in images. Once trained, the algorithm can then be used to detect objects in new, unseen images.

Object Detection in two dimensional space has achieved groundbreaking results with respect to how accurate they are and their computation speed that they can match or even outperform human vision. 2D object detection is a computer vision task that involves detecting objects of interest within an image or a video stream and drawing bounding boxes around them. The goal is to accurately identify the location and class of the objects within the scene. This technology has applications in fields such as surveillance. 2D object detection is used for monitoring security cameras and detecting suspicious behavior or objects. It is used in airports, train stations, and other public places to ensure safety and security. Figure 1.1 shows the use of 2D Object Detection in Safety Helmet(Hardhat)

Figure 1.1: Hardhat Detection

Detection System at construction sites for worker safety.

2D Object Detection however, is insufficient to completely understand our 3D world. The 2D Object Detection techniques have limitations in terms of accurately giving a measure of an object's distance and size. Capturing an object from different viewpoints changes its appearance and its perceived size. It is also not able to give us information about the orientation of an object in a given 2D frame. Also it is not able to detect objects that are occluded or partially hidden.

3D Object Detection techniques can be used to predict the location and size of the objects consistent to their real world dimensions. The orientation of the object is also calculated. Also 3D object detection can help overcome occlusion and partial visibility problems by using depth information to estimate the complete shape of an object.

1.1 Background

3D object detection is the process of localizing and identifying objects in a three-dimensional space from 2D images or point clouds. It is a crucial task in many fields, including robotics, autonomous driving, and augmented reality. The main aim of 3D object detection is to accurately estimate the position, orientation, and size of objects in a 3D environment. This is typically achieved by processing sensor data, such as LiDAR point clouds or cam-

era images, and applying deep learning techniques to extract relevant features and make predictions. Information obtained can be used in wide variety of applications like to enable robots to interact with objects in their environment or to allow self-driving cars to navigate safely.

The process of 3D object detection involves several steps, including sensor data acquisition, data preprocessing, feature extraction, and object detection. Some popular approaches for 3D object detection include multi-view-based methods, which use information from multiple camera or LiDAR viewpoints, and single-view-based methods, which use a single image or point cloud to infer 3D object properties. Other techniques, such as fusion-based methods, combine information from multiple sensors to improve detection accuracy. In general, 3D sensors such as LiDAR (Light Detection and Ranging) or stereo cameras are used to capture 3D data from the environment. The data is then preprocessed to remove noise and extract useful features. One of the primary challenges of 3D object detection is the complexity of the data, which includes multiple modalities and a large number of dimensions.

Feature extraction is a crucial step in 3D object detection, as it involves identifying patterns and characteristics in the data that can be used to differentiate objects from their surroundings. Various techniques can be used for feature extraction, including point cloud processing, image-based methods, and deep learning approaches.

Object detection is the final step in the process, where objects are identified and their attributes (position, orientation, size, etc.) are estimated. This can be done using a variety of methods, including traditional machine learning algorithms such as SVMs (Support Vector Machines) or more recent deep learning techniques such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs)

3D object detection is a rapidly developing field, and there is ongoing research aimed at improving the accuracy and efficiency of object detection algorithms. Recent advances in 3D object detection have been driven by the development of large-scale datasets and the use of deep learning models such as convolutional neural networks (CNNs) and recur-

rent neural networks (RNNs) for feature extraction and prediction. Recent advancements in deep learning have led to significant improvements in object detection accuracy, and there is a growing interest in developing real-time and robust 3D object detection systems for a wide range of applications The field continues to evolve rapidly, with ongoing research focusing on improving detection accuracy, efficiency, and robustness in challenging real-world environments.

1.2 Motivation

The LiDAR assists in measuring the distance of objects which would ultimately help in accurate prediction of the objects in 3D. LiDAR and stereo cameras used to gather spatial information about the surroundings have become much more efficient and accurate. This has opened up new avenues for research in 3D object detection, as researchers can leverage this rich data to develop more accurate and robust detection algorithms.

With the arrival of Intel's Atom and AMD's Geode processor, computers have become smaller in size, energy-efficient and powerful enough to handle substantial amount of computing. Recent advancements in technology have made it now possible to achieve groundbreaking results in terms of accuracy and speed with respect to 3D Objection. It has helped reduce the training time for a given 3D Object Detection model.

Advances in deep learning have enabled researchers to develop more powerful models for 3D object detection. As a result, there is a growing interest in this field, as researchers seek to push the boundaries of what is possible with these new techniques.

3D Object Detection is an exciting field. Ability to implement 3D object detection and make fast predictions can help in automation of vehicles, planes, drones etc. Most of the work with respect to Object Detection has been carried out with respect to 2D Object Detection. This is a relatively new field in which much work hasn't been carried out as of now.

3D object detection is a challenging research problem that requires a deep understanding of computer vision, machine learning, and geometry. Solving this problem requires devel-

oping new algorithms and techniques that can effectively process 3D data. Despite recent advancements, 3D object detection is still a challenging task as the need to detect and track objects in Real-time is there in numerous automation applications. There isn't any 3D Object Detection so far out in the open that strikes a right balance between accuracy and speed and is deployable on edge computing devices for real-time applications. This presents an exciting opportunity for researchers to develop new techniques and algorithms to overcome these challenges and improve the performance of 3D object detection systems.

The development of more accurate and efficient 3D object detection methods has numerous practical applications that can improve safety, convenience, and efficiency in a wide range of industries. It has its use case even for performing tasks such as biomedical imagery. Performing 3D Object Detection on CT and MRI scans can help detect and localize tumors or other abnormalities. 3D object detection can be used in industrial automation to enable robots to detect and manipulate objects in a 3D space, improving efficiency and reducing the need for human intervention. Also it can be used to create immersive augmented reality experiences, where virtual objects can be placed and interacted with in real-world environments.

1.3 Report Organization

Chapter 2 describes the previous works that have been carried out in the field of 3D Object Detection which he have studied in order to gain an all round knowledge of the subject we are working upon. It also highlighted the limitations of work done in this field and formulated the research objective.

Chapter 3 describes the approach we have taken. It also contains the architecture of our modified network, newer activation functions which we have used and also provide an explanation for why they have been chosen.

Chapter 4 provides our implementation details and discusses our results obtained when our algorithm is run on a system containing RTX 5000 graphics card for 300 epochs. It

describes our algorithm's accuracy, efficiency and average loss and compares it with other state of the art existing work in this field.

Chapter 5 describes the limitations of the work done by us as well as gives us a glimpse about future work that can be done in this field.

2

Literature Review

3D object detection is a computer vision field that aims to identify objects and their 3D locations in the real world using sensors such as cameras and LiDAR. The research on computer vision and image processing started in the 1960s, but in the 1980s, researchers began exploring various techniques for 3D object recognition, including model-based recognition and stereo matching. With the introduction of 3D sensors in the 1990s, such as LiDAR and structured light cameras, significant advancements were made in 3D object detection. In the early 1990s, researchers began exploring the use of 3D object recognition techniques, such as point cloud matching, for object detection. These early approaches relied on hand-crafted features and often required manual tuning. In the 2000s, researchers began using machine learning techniques to improve 3D object detection accuracy. One popular approach was to use support vector machines (SVMs) for object classification. Other researchers explored using decision trees, neural networks, and other machine learning algorithms. In the 2010s, researchers began developing feature-based methods for 3D object detection. These approaches relied on features such as 3D edges, surface normals, and histograms of oriented gradients (HOG). One notable feature-based method is the 3D object proposal generation (3DOP) algorithm. Researchers started incorporating 3D information from LiDAR sensors and depth cameras into deep learning models. Large-scale autonomous driving datasets in the 2010s enabled significant progress in 3D object detection. In the mid-2010s, deep learning algorithms, particularly convolutional neural networks (CNNs), started to gain popularity in the field of 3D object detection. These approaches often used point clouds or voxel grids as input data and produced 3D bounding boxes as output. Some notable deep learning-based 3D object detection algorithms include Frustum PointNets[7], and VoxelNet[13]. The use of data from multiple sensors and modalities, such as LIDAR, RGB cameras, and radar for 3D Object Detection also gained popularity during this time. Fused data from multiple sensors was used to improve detection accuracy. More recently, researchers have been exploring hybrid approaches that combine feature-based and deep learning-based methods for improved accuracy. These approaches often use deep learning to extract features from the data and then apply

traditional computer vision techniques for object detection. Ongoing research aims to improve accuracy and efficiency and expand applications beyond autonomous driving, such as robotics and augmented reality. Overall, 3D object detection algorithms have come a long way in the past decade, and there is still much ongoing research and development in this area.

2.1 Key Related Research

The KITTI dataset[1] is a well-known computer vision benchmark for tasks such as object detection, tracking, and 3D scene understanding. It is named after the Karlsruhe Institute of Technology (KIT), where the dataset was initially collected. It was created by the Karlsruhe Institute of Technology and the Toyota Technological Institute at Chicago, and first released in 2012. The KITTI dataset[1] includes a variety of data from sensors mounted on a car, including color and grayscale stereo images, 3D LiDAR point clouds, GPS/IMU data, and object detection annotations. The dataset is often used to develop and evaluate algorithms for tasks such as 3D object detection and tracking, visual odometry and road segmentation.

Figure 2.1: Setup for collecting KITTI Dataset[1]

The data was collected using a specially equipped car that was driven around urban and

rural areas in Karlsruhe, Germany. The car was equipped with a Velodyne HDL-64E LIDAR sensor, which provides a 360-degree view of the environment, as well as two grayscale stereo cameras and a color camera, which provide RGB images and depth maps. The data collection process involved driving the car on various routes, including urban, rural, and highway scenarios. The dataset contains data from different weather conditions, such as sunny, cloudy, and rainy days, as well as different lighting conditions, such as day and night scenes. Figure 2.1 shows the car setup for collecting KITTI dataset. The dataset includes,

- RGB images which are captured from a stereo camera setup mounted on the car, these images are used to provide visual information about the surrounding environment.
- LIDAR point clouds which are captured using a Velodyne LIDAR sensor, these point clouds provide a 3D representation of the surrounding environment, including the locations of objects such as cars, pedestrians, and buildings.
- GPS/IMU data which is captured using a GPS and inertial measurement unit (IMU), this data provides information about the car's motion and orientation.
- 3D bounding box annotations which provide ground-truth information about the locations and sizes of objects in the scene.

Figure 2.2: An Instance of 3D Point Cloud in KITTI Dataset[1]

The KITTI dataset includes several different subsets of data, each with its own specific task for evaluation. These tasks include,

- 3D Object detection, given an RGB image or LIDAR point cloud, the goal is to detect and classify all objects of interest in the scene, and provide their 3D bounding boxes.
- Object tracking, given a sequence of RGB images or LIDAR point clouds, the goal is to track the movement of all objects of interest over time, and provide their trajectories.
- Semantic segmentation, given an RGB image, the goal is to classify each pixel into one of several semantic classes, such as road, building, or vehicle.

Figure 2.2 shows an instance of 3D Point Cloud in KITTI dataset[1].
Frustum PointNets[7] approach uses a frustum-based 3D detection pipeline to detect and classify objects in RGB-D data. First, 2D object detection method is carried out to identify objects and its bounding box dimensions in an RGB image. This 2D box is lifted to a frustum that defines a 3D search space. Now, given a 2D image region and its corresponding 3D frustum , it is used to find out the 3D bounding box center. After that for each point cloud in our frustum, we calculate the probability of it belonging to our object of interest. The parameters for the 3D bounding box include its center coordinates in a 3D space, its height, width and length and heading angle. The heading angle is typically calculated by taking the angle between the projection of the 3D bounding box onto the ground plane and the positive x-axis of the camera coordinate system. Figure 2.3 shows the workflow of Frustum PointNets[7].
VoxelNet[13] uses 3D Point Clouds as input. 3D point cloud is a collection of points in 3D space that is captured by a LiDAR(Light Detection and Ranging) sensor. Each point in the 3D point cloud represents a location in 3D space and has associated properties such as intensity, reflectivity, and color. Each point cloud is divided into a set of voxels. Each voxel is then represented using a feature vector. These feature vectors are then used to

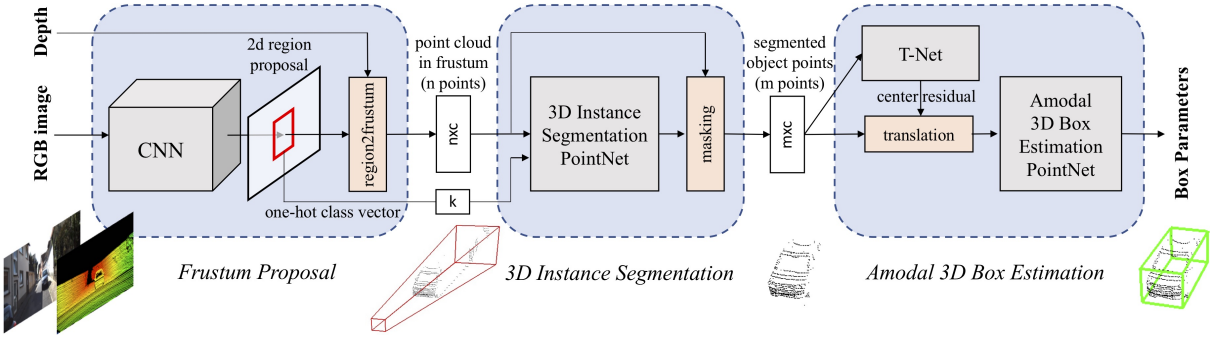

Figure 2.3: Frustum PointNets[7]

generate 3D bounding boxes. Figure 2.4 shows the workflow of VoxelNet[13].

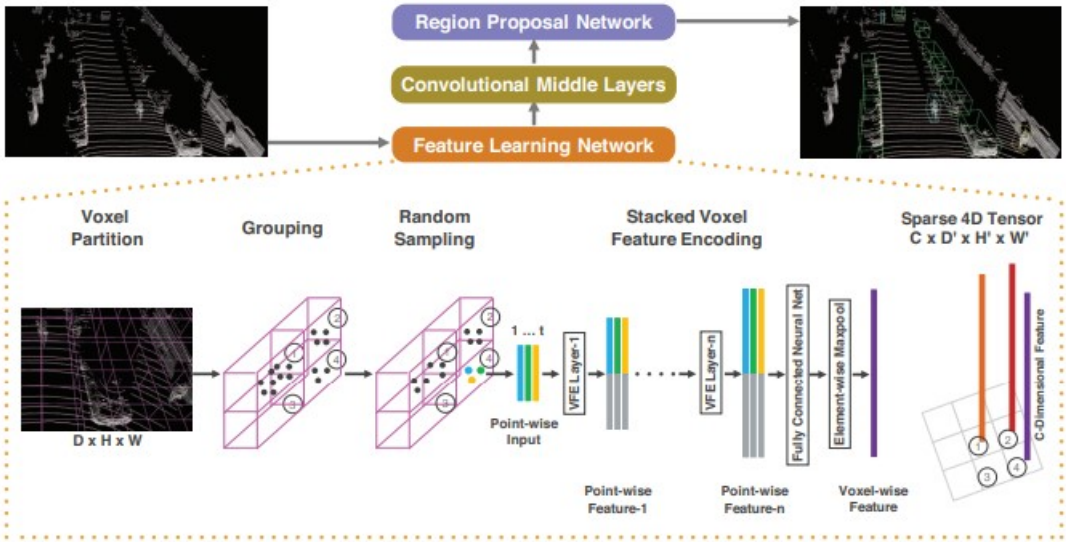

Figure 2.4: VoxelNet[13]

VoxelNet[13] was one of the first papers to use 3D Point Clouds as input for performing 3D Object Detection.

Complex-YOLO[10] is a state-of-the-art real-time 3D object detection algorithm. The algorithm is designed to detect objects in point cloud data generated by LIDAR sensors. It incorporates complex Euler-region-proposals to improve the accuracy of object detection. The algorithm is based on a neural network architecture that takes as input a point cloud and outputs a set of 3D bounding boxes that enclose the objects in the scene.

First a bird eye view RGB map of each frame is created using 3D Point-Clouds. 2D image thus obtained is then fed to a neural network for object detection. This neural network

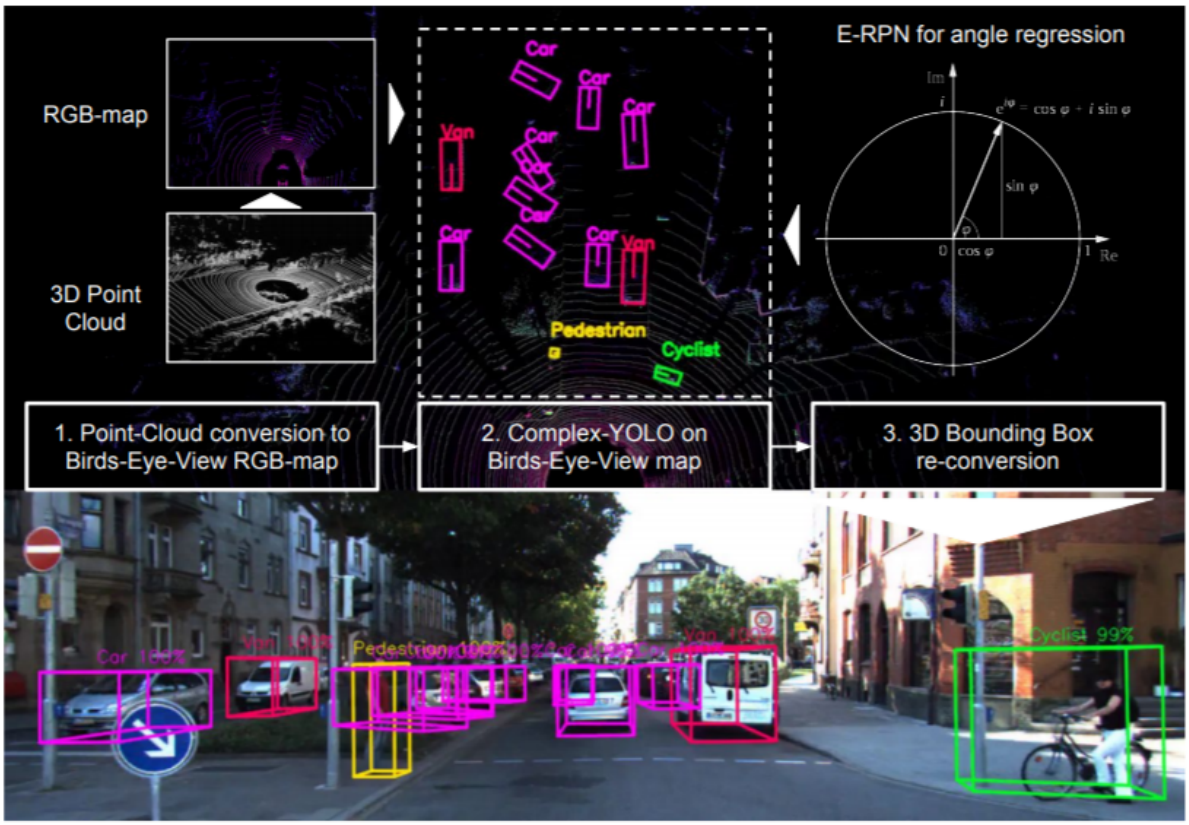

Figure 2.5: Working of Complex YOLO[10]

has a Euler Region Proposal Network attached to the end of it to predict the correct orientation of the multi class 3D object. The steps described in Complex-YOLO[10] are shown in Figure 2.5. The network details of Complex YOLO are shown in Table 2.1.
In Table 2.1,

- conv denotes Convolutional layer.
- reorg denotes Reorganization layer.
- route denotes Routing layer.
- max denotes Max Pool layer.
- E-RPN denotes Euler Region Proposal Network.

The Euler Region Proposal Network(ERPN) is a component of the Complex YOLO[10] model that is used to predict the potential bounding box based on the feature map which

Table 2.1: Complex-YOLO Specifications[10]

layer	filters	size	input	output
conv	24	$3 \times 3/1$	$1024 \times 512 \times 3$	$1024 \times 512 \times 24$
max		$2 \times 2/2$	$1024 \times 512 \times 24$	$512 \times 256 \times 24$
conv	48	$3 \times 3/1$	$512 \times 256 \times 24$	$512 \times 256 \times 48$
max		$2 \times 2/2$	$512 \times 256 \times 48$	$256 \times 128 \times 48$
conv	64	$3 \times 3/1$	$256 \times 128 \times 48$	$256 \times 128 \times 64$
conv	32	$1 \times 1/1$	$256 \times 128 \times 64$	$256 \times 128 \times 32$
conv	64	$3 \times 3/1$	$256 \times 128 \times 32$	$256 \times 128 \times 64$
max		$2 \times 2/2$	$256 \times 128 \times 64$	$128 \times 64 \times 64$
conv	128	$3 \times 3/1$	$128 \times 64 \times 64$	$128 \times 64 \times 128$
conv	64	$3 \times 3/1$	$128 \times 64 \times 128$	$128 \times 64 \times 4$
conv	128	$3 \times 3/1$	$128 \times 64 \times 64$	$128 \times 64 \times 128$
max		$2 \times 2/2$	$128 \times 64 \times 128$	$64 \times 32 \times 128$
conv	256	$3 \times 3/1$	$64 \times 32 \times 128$	$64 \times 32 \times 256$
conv	256	$1 \times 1/1$	$64 \times 32 \times 256$	$64 \times 32 \times 256$
conv	512	$3 \times 3/1$	$64 \times 32 \times 256$	$64 \times 32 \times 512$
max		$2 \times 2/2$	$64 \times 32 \times 512$	$32 \times 16 \times 512$
conv	512	$3 \times 3/1$	$32 \times 16 \times 512$	$32 \times 16 \times 512$
conv	512	$1 \times 1/1$	$32 \times 16 \times 512$	$32 \times 16 \times 512$
conv	1024	$3 \times 3/1$	$32 \times 16 \times 512$	$32 \times 16 \times 1024$
conv	1024	$3 \times 3/1$	$32 \times 16 \times 1024$	$32 \times 16 \times 1024$
conv	1024	$3 \times 3/1$	$32 \times 16 \times 1024$	$32 \times 16 \times 1024$
route	12			
reorg		/2	$64 \times 32 \times 256$	$32 \times 16 \times 1024$
route	22 20			
conv	1024	$3 \times 3/1$	$32 \times 16 \times 2048$	$32 \times 16 \times 1024$
conv	75	$1 \times 1/1$	$32 \times 16 \times 1024$	$32 \times 16 \times 75$
E-RPN			$32 \times 16 \times 75$	

the backbone has predicted. It parses the incoming feature map's position, width, length, probability, class scores, and orientation. Each grid cell of the feature map predicts 5 bounding boxes. 2D to 3D conversion of object is carried out by using predefined height for each class. Working of ERPN is shown in Figure 2.6.

Complex-YOLO[10] was one of the first papers to perform Real-time 3D Object Detection.

Mish activation function[6] is a relatively new activation function for artificial neural networks and has gained attention due to its good performance in various machine learning tasks.

Equation(2.1) shows the Mish Activation Function[6].

$$f(x) = x \cdot \tanh(\ln(1 + e^x)) \tag{2.1}$$

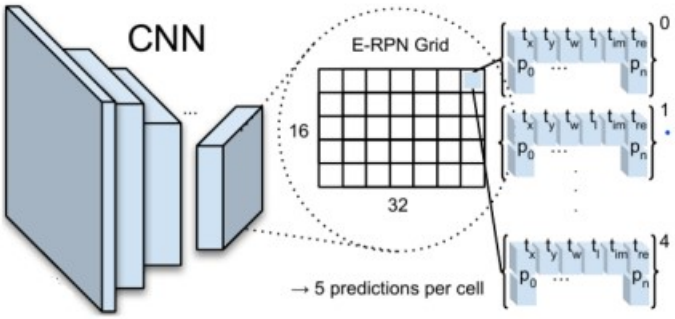

Figure 2.6: ERPN Grid[10]

The Mish function[6] combines the best properties of two other popular activation functions: the Rectified Linear Unit (ReLU) and the Hyperbolic Tangent (tanh) function. Like ReLU, it has a non-negative derivative, which can speed up the learning process in some cases. Like tanh, it is smooth and differentiable, which makes it well-suited for gradient-based optimization algorithms. The Mish function[6] is defined on the entire real line and can take any input value. Its output ranges from approximately -0.31 to infinity. Figure 2.7 shows the graphical plot of Mish Activation Function.

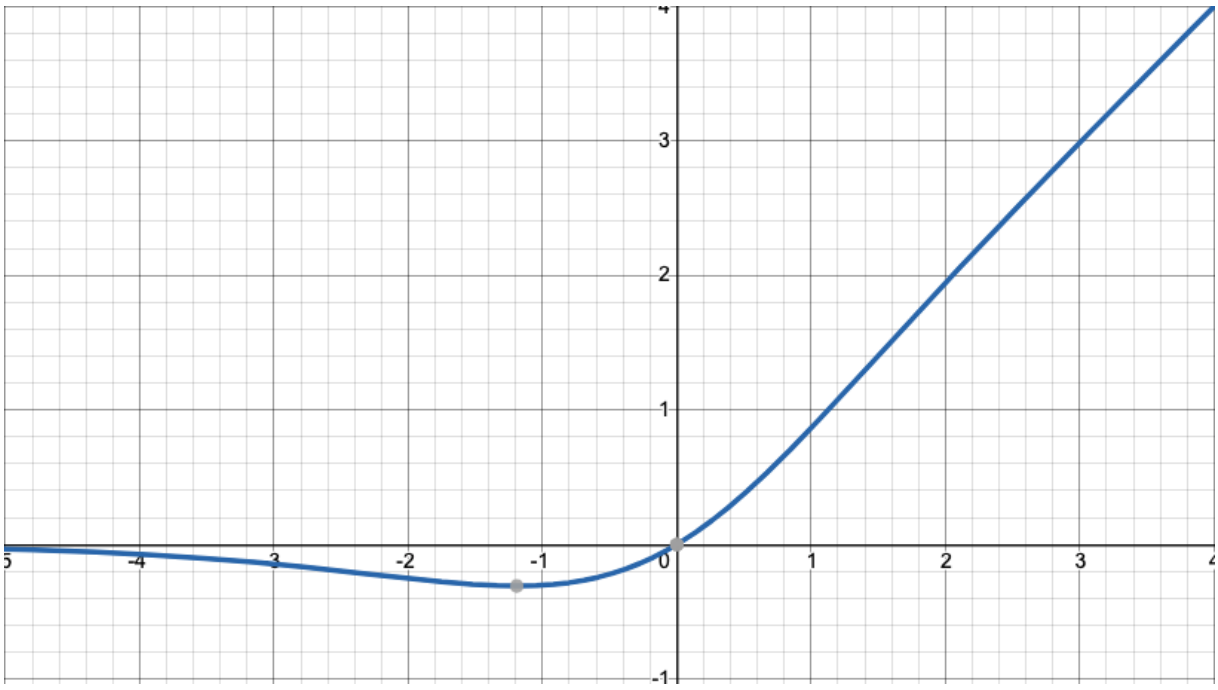

Figure 2.7: Mish Activation Function[6]

Table 2.2: Comparing Activation Functions for Image Classification on CIFAR10 dataset using a SqueezeNet for 23 Runs[6]

Activation	μ_{acc}	μ_{loss}	σ_{acc}
Mish	87.48%	4.13%	0.3967
Swish	87.32%	4.22%	0.414
ReLU	86.66%	4.398%	0.584
Leaky ReLU	86.85%	4.112%	0.4569
TanH	82.72%	5.322%	0.5826

Table 2.2 shows the comparison between Mish[6] and other popular activation functions for image classification on CIFAR10 dataset using a Squeeze Net for 23 runs. In table 2.2, μ_{acc} signifies mean accuracy in percentage, μ_{loss} signifies mean loss in percentage and σ_{acc} signifies standard deviation of accuracy.

Equation(2.2) shows the formula for calculating mean accuracy.

$$\mu_{acc} = \frac{1}{N}\sum_{i=1}^{N} \text{acc}_i \tag{2.2}$$

In Equation(2.2) μ_{acc} signifies mean accuracy, N signifies total number of images in the CIFAR10 test dataset, acc_i is the accuracy of the SqueezeNet model on i^{th} image.

Equation(2.3) shows the formula for calculating accuracy of i^{th} image where acc_i signifies accuracy of i^{th} image sample.

$$\text{acc}_i = \begin{cases} 1, & \text{if predicted class for image } i = \text{ground truth label for image } i \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

Equation(2.4) shows the formula for calculating mean loss where loss_i indicates the loss of i^{th} image sample and N indicates total number of images in the CIFAR10 test dataset.

$$\mu_{\text{loss}} = \frac{1}{N}\sum_{i=1}^{N} \text{loss}_i \tag{2.4}$$

Equation(2.5) shows the formula for calculating the loss of i^{th} image sample.

$$\text{loss}_i = -\sum_{k=1}^{K} y_{i,k} \log(p_{i,k}) \tag{2.5}$$

In Equation(2.5) K is the number of classes, $y_{i,k}$ is a binary indicator variable that is equal to 1 if the ground truth label for image i is k, and 0 otherwise and $p_{i,k}$ is the predicted probability of the SqueezeNet model for image i belonging to class k.

Equation(2.6) shows the formula for calculating the standard deviation of accuracy, where N is the total number of accuracy values, acc_i is the i^{th} accuracy value and μ_{acc} is the mean accuracy value.

$$\sigma_{acc} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - \mu_{acc})^2} \tag{2.6}$$

Equation(2.7), Equation(2.9), Equation(2.10) and Equation(2.11) depict the formula for Swish, ReLU, Leaky ReLU and TanH activation functions respectively.

$$f(x) = x \cdot \sigma(x) \tag{2.7}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.8}$$

$$f(x) = \max(0, x) \tag{2.9}$$

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \tag{2.10}$$

In Equation(2.10) α is a constant.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.11}$$

The CIFAR-10 dataset[5] is a popular image classification dataset that consists of 50,000 training images and 10,000 testing images. Each image is a color image with dimensions. There are 10 classes in CIFAR-10 dataset[5]. The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. For each class there are 6000 images. CIFAR-10 dataset[5] is widely used for machine learning research and benchmarking image classification algorithms. It is often used as a benchmark to evaluate the performance of various deep learning architectures, such as convolutional neural networks (CNNs). Many researchers use this dataset to test their models and compare them to other state-of-the-art

approaches.

SqueezeNet[4] is a deep neural network architecture for image classification that was introduced in 2016 by researchers at UC Berkeley. SqueezeNet[4] achieves comparable accuracy to other state-of-the-art deep neural networks, but with a much smaller number of parameters, making it more memory-efficient and faster to train and execute. SqueezeNet[4] has been widely adopted in various applications, including mobile and embedded systems, where memory and computational resources are limited.

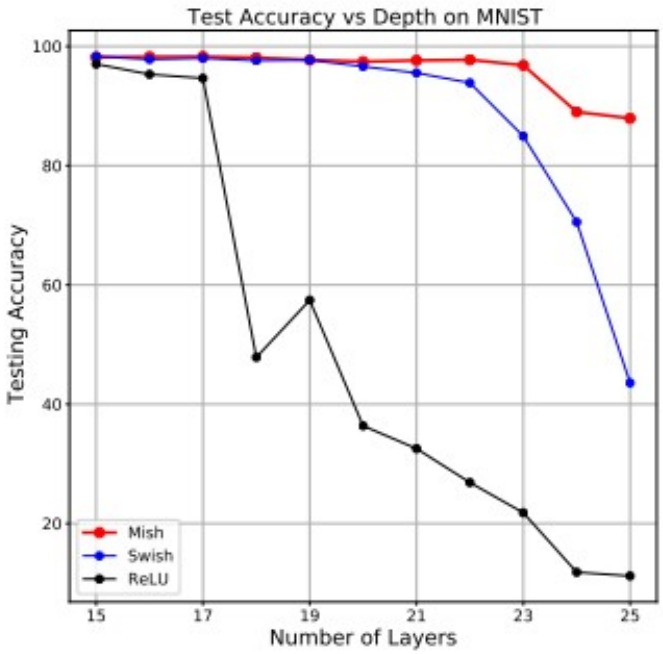

Figure 2.8: Comparison between ReLU, Swish and Mish activation functions in terms of test accuracy with increasing depth of the neural network on the MNIST dataset[6]

Figure 2.8 shows the comparison in performance of Mish activation function with Swish and Relu activation functions with an increase in number of layers of the neural network model on MNIST dataset[6].

MNIST dataset is a collection of images of handwritten dataset, often employed as a reference dataset for image classification algorithms in machine learning. It includes 60,000 images for training and 10,000 for testing, with each image having dimensions of 28 by 28 pixels. The objective is to accurately categorize each image into one of ten possible classes

representing digits 0 through 9. The MNIST dataset has gained widespread recognition in the research community, and its popularity has stimulated the creation of numerous models and approaches for image recognition.

2.2 Research Gaps

There doesn't exist any 3D Object Detection technique which strikes a right balance between speed and accuracy. The techniques discussed so far give satisfying result for either speed or accuracy but not both. Some of the techniques require use of multiple cameras for capturing input data which makes them computationally expensive. Also the activation functions used in the corresponding model network of the techniques discussed so far prevents the model from updating its weights after some epochs which in turn limits accuracy of our model. There doesn't exist any single technique which is deployable on edge computing devices for Real-time 3D Object Detection applications and is economically viable.

2.3 Objectives

- To achieve accuracy and speed of our 3D Object Detection algorithm comparable to other state of the art techniques on our KITTI Dataset.
- To perform 3D Object Detection in Real-time.

3

Methodology

We have changed the network described in Complex-YOLO[10]. We have designed our network by using Cross Stage Partial Connection Blocks in the backbone of our network. This is followed by the use of attention modules at two different stages in the backbone of our network followed by feature extraction. Results are obtained at two different sizes of the grid map by using multiple YOLO heads.

3.1 Overall flow

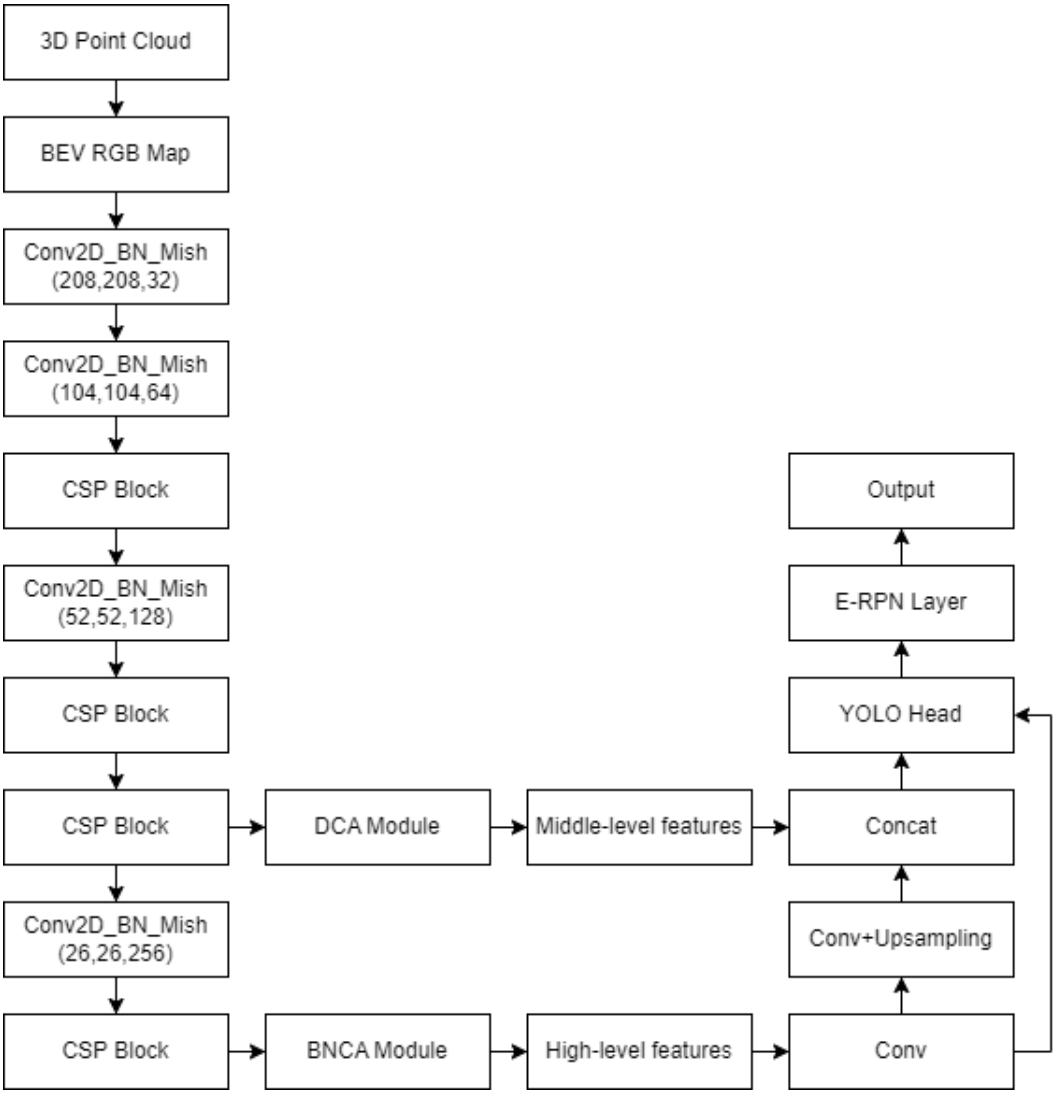

Figure 3.1: Complex-YOLOv1 Proposed Framework

Steps involved are as follows,

- First a bird's eye view RGB map of each 3D Point Cloud frame is created.
- 2D image thus obtained is then fed to our network to predict possible objects.
- The last layer of our network is a Euler Region Proposal Network(E-RPN) Layer which is used to predict the correct orientation of the multi class 3D object. Each grid cell of the E-RPN grid predicts 5 bounding boxes.
- These bounding boxes contain information about object's position, width, length and their orientation.
- Complex number representations are used for orientation angle of our object, containing real and imaginary part, the paper[10] has used it so that it can be directly used to calculate the loss during learning process.
- 2D to 3D conversion of object is carried out by using predefined height for each class.

Figure 3.1 shows the proposed framework for Complex YOLOv1.
3D Point Cloud covers a very large area of 80m width and 40m length. A 3D Point Cloud is a set of points in three dimensional space that represent the surface of an object or environment. These 3D Point Clouds are then converted to a Bird's eye view RGB map. The R channel refers to height, G channel refers to Intensity and B channel refers to Density of the point data. The size of the grid map used is of 416*416 resolution. The calibration data obtained from KITTI dataset is used to encode points to respective grid cell of the RGB map. Each grid cell has maximum height, intensity and density for that particular grid cell encoded to it.
The input for Complex-YOLOv1 network is the encoded Bird's eye view image which can be processed as a normal RGB image.
In the initial layers of our network Cross Stage Partial Connection Blocks(CSP Block) are used. CSP Blocks are followed by Max Pooling layer which reduce the incoming input feature map's width and height by 2.

We have also used Dual Channel Attention Module and Batch Normalization Channel-wise attention Module in our network for middle and high level feature extraction.
Regular convolutional layers are used across different layers of our Complex-YOLOv1 network.
Also, we have used Mish Activation Function across different layers of our network followed by Batch normalisation to speed up training, reduce overfitting and for better generalization.
Multiple YOLO heads are used to predict object bounding boxes and class probabilities at different scales and resolutions.

3.1.1 Cross Stage Partial Connection Block(CSP Block)

The main advantage of the Cross Stage Partial Connection Block or CSP block is that it reduces the computational cost of the network while maintaining a high level of accuracy. By splitting the feature map into two parts, the CSP block reduces the number of parameters and computation required in each convolutional layer. Furthermore, the concatenation of the feature maps from both pathways allows the network to learn more diverse representations of the input data, leading to improved accuracy. Figure 3.2 shows the architecture of CSP Block.
First, the input feature map is split into two halves along channel dimension. The second halve first undergoes convolution. The resulting output from the convolution of second halve undergoes convolution again and is concatenated with the output of that convolution, which is followed by another convolution. At the end concatenation across channel dimension with the first halve of input feature map to the CSP Block is carried out.

3.1.2 Dual Channel Attention Module(DCA Module)

Dual channel attention module or DCA Module helps obtain the global relationship between each pixel and not only local domain relationships as performed by traditional convolution. It helps capture middle level features.

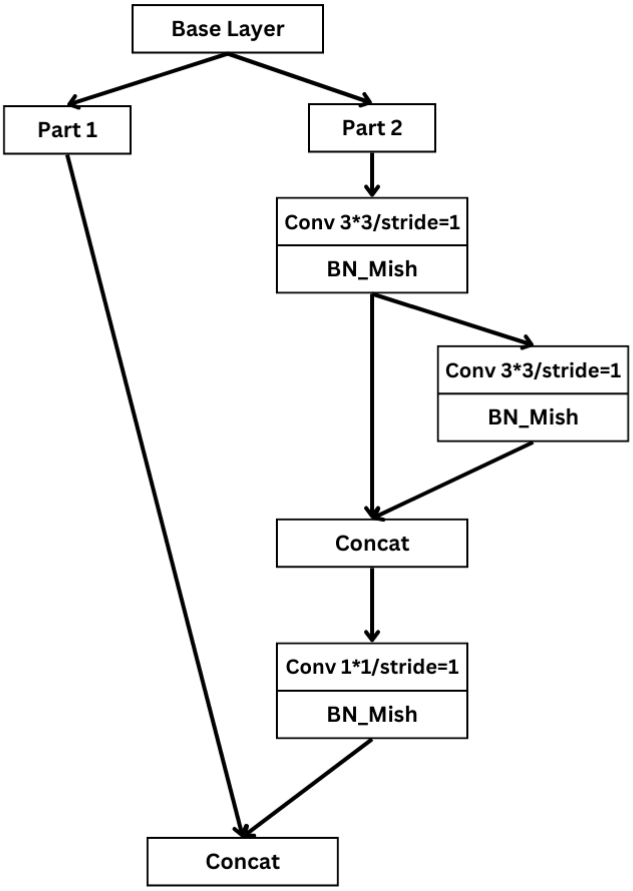

Figure 3.2: Cross Stage Partial Connection Block

Architecture of the module is shown in Figure 3.3.

First, the Global Average Pooling(GAP) operation is carried out to obtain global information. In the original paper[2], Dual channel attention module used two fully connected layers after GAP operation. However upon testing we came to the conclusion that removing these two fully connected layers will not affect our accuracy much but would however significantly increase the efficiency of our algorithm. The spatial attention section undergoes multiple convolutions and at the end stages number of channels are reduced to 1. After obtaining the spatial and channel weights, we adopt an element-wise addition

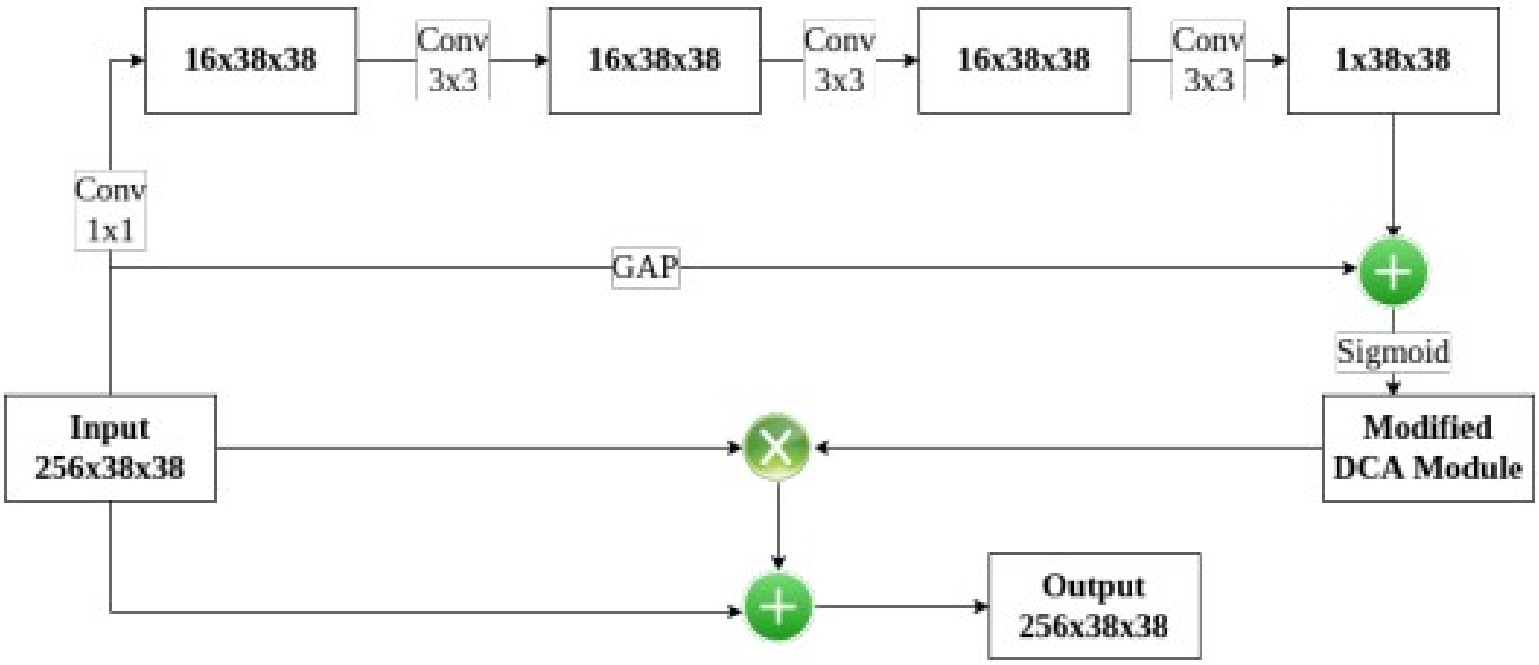

Figure 3.3: Dual Channel Attention Module

method for integration, followed by Sigmoid activation function. The output thus obtained is element wise multiplied with our input to DCA module is followed by element wise addition of our input to DCA with the resulting output.

3.1.3 Batch Normalization Channel-wise Attention Module(BNCA Module)

Batch Normalization Channel-wise attention module or BNCA Module is used for high level feature extraction. First, Global average pooling is carried out to generate characteristics of the channel. After that every value in each channel is then multiplied by its corresponding channel average. The corresponding output thus obtained then undergoes Sigmoid Activation Function followed by Batch Normalization. This output is then again element wise multiplied by original incoming feature layer. The same is shown in Figure 3.4.

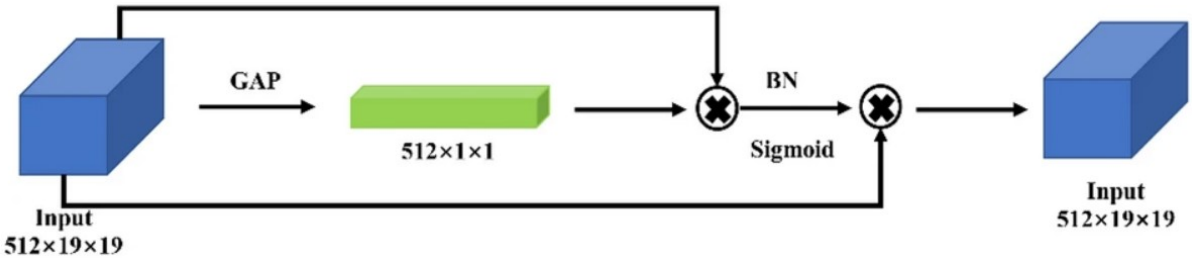

Figure 3.4: Batch Normalization Channel-wise Attention Module[2]

3.1.4 Mish Activation Function

In place of Leaky ReLU function used in Complex-YOLO[10], we have used Mish Activation Function. Details about Mish Activation Function are present in Literature Review Part.

Mish is a self regularized and non-monotonic activation function. This helped our model to learn more complex representations of the input data, improving its ability to generalize to new data.

Also it's smoothness, ensures that it has a continuous first derivative everywhere. It helps avoid the issue of "dying ReLU" in our model where some neurons get stuck and stop updating their weights during training.

Also, we tested for first 20 epochs for Leaky ReLU, ReLU, TanH, Mish and Swish activation functions, and Mish activation function gave us the lowest loss and highest Mean Average Precision Score amongst them all.

3.2 Methodology

3.2.1 3D Point Cloud Conversion

The KITTI dataset[1] contains 3D point clouds which have been captured using the Velodyne HDL64 laser scanner, which are then converted into bird's-eye view RGB map. The RGB map thus obtained covers an area of 80 metres by 40 metres in front of the LiDAR sensor. The grid size is 1024 by 512. Three feature channels are calculated for the whole point cloud in one frame in the area Ω(say); Equation(3.1) shows the points taken into

consideration,

$$P_\Omega = \{P = [x, y, z]^T \mid x \in [0\text{m}, 40\text{m}], y \in [-40\text{m}, 40\text{m}], z \in [-2\text{m}, 1.25\text{m}]\} \tag{3.1}$$

$z \in [-2\text{m}, 1.25\text{m}]$ expects highest objects to be 3.25 metres above the ground. The mapping function S_j used to map points to each cell maps each point to grid cell in RGB map shown by Equation(3.2).

$$P_{\Omega i \to j} = \{P_{\Omega j} = [x, y, z]^T \mid S_j = f_{PS}(P_{\Omega i}, g)\} \tag{3.2}$$

Velodyne Intensity $I(P_\Omega)$ is used to calculate the channel of each pixel.

$$z_g(S_j) = max(P_{\Omega_{i \to j}} \cdot [0, 0, 1]^T) \tag{3.3}$$

$$z_b(S_j) = max(I(P_{\Omega_{i \to j}})) \tag{3.4}$$

$$z_r(S_j) = min(1, \log(N + 1)/64) \qquad N = |P_{\Omega_{i \to j}}| \tag{3.5}$$

In Equation(3.5) N denotes the number of points that will be mapped from the area to grid cell S_j. In Equation(3.4) z_b consists of points having maximum intensity, in Equation(3.3) z_g consists of points having maximum height and in Equation(3.5) z_r consists of points with normalized density. All points are mapped to a particular cell.

3.2.2 Bounding Box Prediction

Figure 3.5 shows how our final 3D Bounding Box is calculated. Our final 3D bounding box is based on YOLO[8] regression parameters from our incoming feature map as well as complex angle for box orientation. 5 bounding boxes containing YOLO regression parameters for each grid cell of our feature map are provided as input to our Euler Region Proposal Network layer. A predefined height is used for each class for converting our 2D predictions to 3D.

Euler Region Proposal Network Layer is the last layer of our network which is used to predict the potential bounding boxes. It parses the 3D object position, width, length

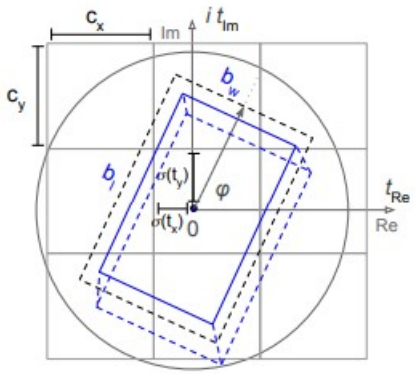

Figure 3.5: 3D Bounding Box Prediction[10]

and finally its orientation from incoming feature map. Equation(3.6), Equation(3.7), Equation(3.8) and Equation(3.9) for calculating the coordinates of the bounding boxes are as follows,

$$b_x = \sigma(t_x) + c_x \tag{3.6}$$

$$b_y = \sigma(t_y) + c_y \tag{3.7}$$

$$b_w = p_w + e^{t_w} \tag{3.8}$$

$$b_l = p_l + e^{t_l} \tag{3.9}$$

In Equation(3.6) and Equation(3.7) $b_{x,y}$ represents the object coordinates, in Equation(3.8) b_w represents the object's width and in Equation(3.9) b_l represents object's length. Orientation of bounding box is calculated using the Equation(3.10),

$$b_\phi = arctan_2(t_{Im}, t_{Re}) \tag{3.10}$$

The complex number is added to make them account for orientation. The use of complex number representation for our orientation allows us to use them directly in our loss function without any modification and hence the whole network predicts outcomes in one

go.

3.2.3 Loss Function

Loss function used is a summation of two loss functions. This is shown in Equation(3.11).

$$\mathcal{L} = \mathcal{L}_{YOLO} + \mathcal{L}_{Euler} \tag{3.11}$$

$\mathcal{L}_{YOLO}$ in Equation(3.11) uses standard mean squared error loss.
Euler part of the loss is depicted using Equation(3.12) and Equation(3.13),

$$\mathcal{L}_{\text{Euler}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} | \left(e^{ib_\phi} - e^{i\hat{b}_\phi} \right)^2 | \tag{3.12}$$

$$\mathcal{L}_{\text{Euler}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [\left(t_{im} - \hat{t}_{im}\right)^2 + \left(t_{re} - \hat{t}_{re}\right)^2] \tag{3.13}$$

In Equation(3.12) and Equation(3.13), λ_{coord} is a scaling factor to ensure stable convergence in early phases and 1_{ij}^{obj} denotes that the jth bounding box predictor in cell i has highest intersection over union (IoU) compared to ground truth for that prediction.
Euler part of the loss is used to account for orientation. Comparison of the predicted complex number of unit 1 and actual complex number of unit 1, their squared error is minimized to get Euler loss.

3.3 Summary

This chapter describes the methodology which we have adopted to convert 3D Point Clouds to Bird's eye view maps, the procedure for calculating our loss function(accounting for the loss of change in predicted object's position, size and orientation). Details about how we have designed our proposed network has also been given.

4

Results and Discussion

4.1 Experiment Design

The proposed algorithm is trained and tested on a system containing NVIDIA Quadro RTX 5000 graphics card.

We used KITTI Dataset[1] for model training. Our dataset includes 7481 training samples and 7518 test samples. Our training dataset is split into two parts where 6000 training samples have been assigned for training purpose and rest for validation purpose.

Adam Optimization based gradient descent is used to train our model using momentum of 0.9 and weight decay of 0.0005 using a batch size of 4, and we have trained our model for upto 300 epochs.

The given system of equations(from Equation(4.1) to Equation(4.5)) is used to depict the update of weights using Adam Optimization based gradient descent.

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.1}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.2}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.3}$$

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \tag{4.4}$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \tag{4.5}$$

For the given system of equations(from Equation(4.1) to Equation(4.5)) θ_t indicates weight at instance t, α denotes weight decay, β_1 denotes momentum, g_t denotes the first moment of gradient of loss function with respect to model weights, g_t^2 denotes second moment of gradient of loss function with respect to model weights. We have set β_2 in Equation 4.5 to its default value 0.999.

4.2 Results

The results and comparison depicted in this section are for when our model has been trained on our dataset for 300 epochs. Our model took 21 hours 42 minutes to do the

same.

Table 4.1 shows different performance metrics of our algorithm for different classes for Intersection over Union(IoU) threshold of 0.5 on our validation dataset.

IoU is calculated using Equation(4.6).

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \tag{4.6}$$

IoU is a commonly used metric in computer vision tasks, especially for applications like Object Detection. It measures the overlap between actual bounding box and ground truth bounding box. Generally, a threshold value for IoU is usually set.

In table 4.1,

Table 4.1: Evaluation Matrix on our Validation dataset with IoU of 0.5

Class	Precision	Recall	AP	f1
Car	0.9338	0.9684	0.9595	0.9508
Pedestrian	0.5248	0.6481	0.4538	0.5800
Cycle	0.7445	0.8645	0.8116	0.8000

Precision for a class is calculated using Equation(4.7).

$$P = \frac{TP}{TP + FP} \tag{4.7}$$

In Equation(4.7), TP stands for True Positives, FP stands for False Positives and P stands for Precision for the class for which Precision is calculated respectively.

True Positive is the number of positive instances that were correctly identified as positive by the model. False Positive is the number of negative instances that were incorrectly identified as positive by the model.

With respect to Object Detection, if the IoU between the ground truth bounding box and predicted bounding box is greater than or equal to our set threshold value and the class label for our predicted bounding box is same as that of ground truth bounding box it is considered as true positive for the predicted class label.

If for a predicted class there is no ground truth box having IoU greater than the threshold value and having the same class label, then the prediction is considered a false positive for the predicted class label. False positive signify incorrect detections.

In Object Detection, Precision for a class is a measure of how accurately our system identifies objects of a class in an image and ignore irrelevant objects in an image or background.

Recall for a class is calculated using Equation(4.8).

$$R = \frac{TP}{TP + FN} \tag{4.8}$$

In Equation(4.8), R stands for Recall, TP stands for True Positives and FN stands for False Negatives for the class for which Recall is calculated respectively.

A false negative is an outcome where the model does not correctly predict the negative class.

If for a ground truth object for a particular class there is no bounding box having IoU greater than threshold value and having the same class label, it signifies a False Negative for that particular class. False negatives signify missed detections in Object Detection.

Recall for a class is a measure of how many objects of that given class our model is able to detect.

AP in Table 4.1 stands for Average Precision. AP for a class is calculated by calculating the area under Precision Recall curve for that particular class for different confidence threshold values. AP for a class captures the balance between precision and recall for that particular class and enhances the impact of both evaluation measures. It is the widely used metric for comparing the accuracy of detection of different Object Detection models.

f1 score for a class is calculated using Equation(4.9).

$$f = \frac{2}{\frac{1}{P} + \frac{1}{R}} \tag{4.9}$$

In Equation(4.9), f stands for f1 score, P stands for Precision and R stands for Recall for the class for which f1 score is calculated respectively.

f1 score combines both precision and recall, making it a good aggregated indicator. A high F1 score in object detection tasks implies that the algorithm is capable of detecting a large proportion of the objects of a specific class with a low number of false positives. This suggests high recall and high precision of the algorithm. On the other hand, a low F1 score indicates that the algorithm may miss a significant number of objects in a specific class, resulting in low recall or may produce many false positives, indicating low precision.

Figure 4.1: Average loss vs Number of steps taken by the algorithm

A graph of Average loss vs Number of steps taken by the algorithm is depicted in Figure 4.1. As we can see there is a significant decrease in average loss of our model as the optimization process goes on. The step wise gradient descent is performed for every batch. The batch size is 4 and number of training samples is 6000 which gives 1500 steps per epoch and 300 epochs are performed which gives us a total of 450,000 steps.

Table 4.2 shows performance comparison between our model and other state of the art 3D Object Detection Models. FPS in table 4.2 stands for Frames per Second. It is a measure of number of frames(images) detected by an algorithm in a second.

From table 4.2 we can see that our model outperforms other state of the art algorithms for car and cyclist class. For Pedestrian class however our model has been outperformed by Frustum PointNets[7], SVGA-Net[14] and EPNet++[15], however, they perform de-

Table 4.2: Average Precision's in percentage and Frames per second for our model on KITTI dataset compared to other leading methods

Method	FPS	Car	Pedestrian	Cyclist
Frustum PointNets[7]	5	83.76	70.00	77.15
VoxelNet[13]	4.3	77.47	39.48	61.22
Complex-YOLO[10]	50.4	67.72	41.79	68.17
SA-SSD[3]	25	88.75	-	-
CIA-SSD[12]	32	89.59	-	-
SVGA-Net[14]	16.13	87.33	48.48	78.58
EPNet++[15]	10	91.37	52.79	76.15
SPG[16]	11.11	90.50	-	-
Complex-YOLOv1(proposed)	44.71	**95.95**	45.38	**81.16**

tections at rate of 5 Frames per Second, 16.13 Frames per Second and 11.11 Frames per Second respectively which makes them unfeasible for deployment for Real-time applications(the required FPS for real-time applications deployment depends on the specific use case, but a minimum of 30 FPS is usually required for a smooth experience). Still the improvement observed for Pedestrian class from Complex-YOLO[10] is lesser than we expected. This could be owing to the fact that 3D Point Clouds which are provided as input to our model are converted to Bird's eye view maps. Bird's eye view makes the perceived size for pedestrian class very small which makes this class harder to capture and detect. Also, Complex-YOLO[10] performs detection at 50.4 FPS which is faster compared to our method which performs detection at 44.71 FPS. However, our model beats Complex-YOLO[10] by considerable margin for Average Precision across all the classes.

Our method for 3D Object Detection outperforms other state of the art across several metrics. Thus we have achieved our objective of developing a 3D Object Detection algorithm which has accuracy and speed comparable to other state of the art algorithms on our KITTI dataset[1].

Which 3D Object Detection technique to chose depends on our application, how we prioritise for accuracy on different classes, whether we want to deploy our method for performing

detections in real time, speed vs accuracy trade-off etc.

As discussed earlier our model didn't perform as per our expectations for pedestrian class however it outperformed for Car and Cycle class at computational speed of 44.71 FPS. This makes our model ideal for deployment for applications such as Real-time autonomous Vehicle tracking systems, traffic monitoring and sports analysis applications like tracking the movement of cyclists during races or cars during motorsport events. Also our model was trained on KITTI dataset which focuses on cars, cyclists and pedestrians. For applications involving other object classes our model would require retraining or fine-tuning for that particular application. However, our model's performance may degrade for applications which involve complex scenes such as crowded urban environments. Also, it is not suitable for high-precision applications such as medical imaging or scientific analysis where accuracy is critical.

Figure 4.2 shows an example of 3D bounding boxes obtained using our method on bird's eye view RGB map obtained from 3D Point Clouds obtained using LiDAR and its corresponding 3D Bounding Box on camera image. In figure 4.4, yellow coloured bounding box is used for car, red coloured bounding box is used for pedestrian and blue coloured bounding box is used for cycle.

4.3 Summary

This chapter describes the results for our model obtained and shows its comparison with other state of the art 3D Object Detection algorithms. Details about initial model setup used to obtain our results has also been given.

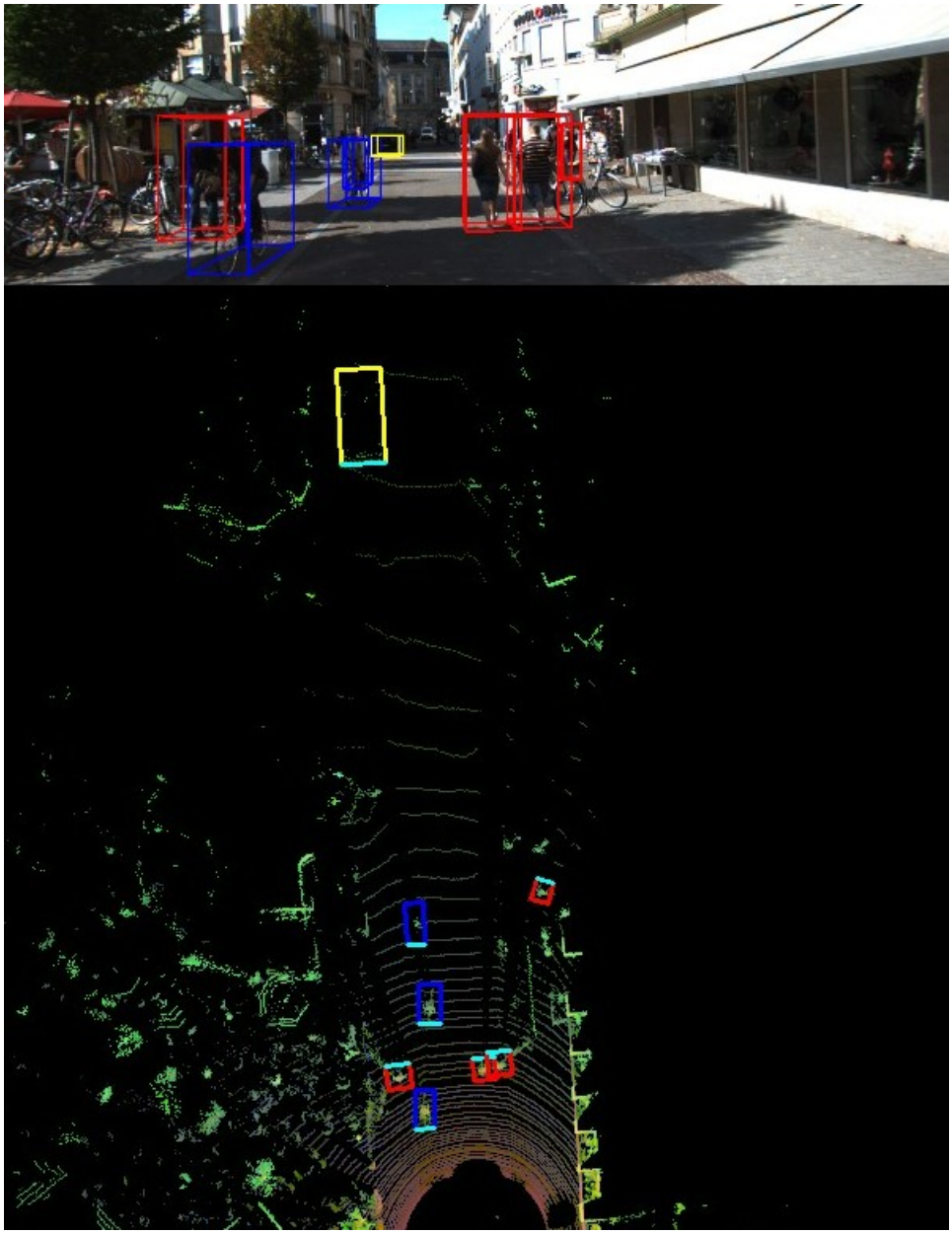

Figure 4.2: 3D Bounding Boxes on Bird's eye view map and its corresponding camera image

5

Conclusion and Future Scope

5.1 Conclusion

3D Object Detection is a complex problem that involves detecting objects for a given input frame and finding the object's location, height, width, length and orientation in our 3D space. We have performed changes to the network described in Complex-YOLO[10] to obtain a model that strikes a right balance between speed and accuracy for 3D Object Detection tasks. The work presented in this report is a step forward in the field of 3D Object Detection.

The results obtained by us are comparable to other state of the art existing 3D Object Detection algorithms.

Also, we have achieved our objective of performing detections in real-time.

Our model can be deployed for real world scenarios for applications like Real-time vehicle tracking and traffic monitoring on edge computing devices provided enough information is captured and provided to it in form of 3D Point Clouds. Our model can even be deployed for sports analysis applications like tracking the movement of cyclists during races or cars during motorsport events. Also, our model could be retrained and fine tuned for 3D Object Detection tasks that are concerned with object classes other than the ones described in our KITTI dataset(cars, pedestrians, cyclists).

Also, the modified network introduced by us for performing 2D Object Detection step and getting corresponding feature map for 2D Object could further be tried for different 2D Object Detection applications.

Our approach for 3D Object Detection can be used in areas with low lighting conditions.

5.2 Future Scope

The approach presented in this paper has addressed the current state of the art in machine learning-based Real-time 3D Object Detection. However, there are several areas that can be explored further in the future to improve the efficiency and accuracy of these techniques.

Our method uses predefined heights for different classes. In order to correctly output accurate bounding boxes we need to go away with this assumption.

We can try and implement attention modules which we have used for feature extraction namely Batch Normalization Channel-wise attention module and Dual Channel Attention module with different combinations of fully connected layers and Convolutional layers to find the right balance between speed and accuracy.

Our method uses 3D Point Clouds as input which are captured using LiDAR sensors. LiDAR sensors though accurate are still expensive. Alternative approaches which doesn't require the use of LiDAR sensors for capturing input data should be explored.

We should try to implement Complete Intersection Over Union to further increase the accuracy of our algorithm.

We should try newer and unexplored open source 2D Object Detection method techniques for implementing 2D Object Detection in our algorithm and getting corresponding feature maps and check for if they would give a better overall result in terms of speed and accuracy for 3D Object Detection.

Bibliography

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1231–1237, 2013.

[2] C. Guo, J. Yang, and T. Guo, "Improved yolov4-tiny network for real-time electronic component detection," *Scientific Reports*, vol. 11, no. 1, pp. 1–13, 2021.

[3] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3d object detection from point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 873–11 882.

[4] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[5] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[6] D. Misra and D. Mishra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[7] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[10] M. Simon, S. Milz, K. Amende, X. Dang, U. Franke, and S. Behnke, "Complex-yolo: Real-time 3d object detection on point clouds," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6362–6369.

[11] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Csp-net: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1467–1476.

[12] W. Zheng, G. Zhou, S. Huang, W. Xiang, and Y. Dai, "Cia-ssd: Confident iou-aware single-stage object detector from point cloud," *arXiv preprint arXiv:2012.03015*, 2020.

[13] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4490–4499.

[14] Q. He, Z. Wang, H. Zeng, Y. Zeng, and Y. Liu, "Svga-net: Sparse voxel-graph attention network for 3d object detection from point clouds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 870–878.

[15] Z. Liu, T. Huang, B. Li, X. Chen, X. Wang, and X. Bai, "Epnet++: Cascade bi-directional fusion for multi-modal 3d object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[16] Q. Xu, Y. Zhou, W. Wang, C. R. Qi, and D. Anguelov, "Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 446–15 456.

BTP 2023

ORIGINALITY REPORT

13%	9%	11%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	1%
2	www.ncbi.nlm.nih.gov Internet Source	1%
3	"Computer Vision – ECCV 2022", Springer Science and Business Media LLC, 2022 Publication	1%
4	webmafia.pk Internet Source	1%
5	hdl.handle.net Internet Source	1%
6	"Computer Vision – ECCV 2018 Workshops", Springer Science and Business Media LLC, 2019 Publication	1%
7	deepai.org Internet Source	<1%
8	repository.tudelft.nl Internet Source	<1%

9	Li Wang, Xinyu Zhang, Ziying Song, Jiangfeng Bi, Guoxin Zhang, Haiyue Wei, Liyao Tang, Lei Yang, Jun Li, Caiyan Jia, Lijun Zhao. "Multi-modal 3D Object Detection in Autonomous Driving: A Survey and Taxonomy", IEEE Transactions on Intelligent Vehicles, 2023 Publication	<1 %
10	oa.upm.es Internet Source	<1 %
11	Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, Leonidas J. Guibas. "Frustum PointNets for 3D Object Detection from RGB-D Data", 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018 Publication	<1 %
12	Chongben Tao, Shiping Fu, Chen Wang, Xizhao Luo, Huayi Li, Zhen Gao, Zufeng Zhang, Sifa Zheng. "F-PVNet: Frustum-level 3D Object Detection on Point-Voxel Feature Representation for Autonomous Driving", IEEE Internet of Things Journal, 2022 Publication	<1 %
13	Xinkai Wang, Xu Jia, Miyuan Zhang, Houda Lu. "Object Detection in 3D Point Cloud Based on ECA Mechanism", Journal of Circuits, Systems and Computers, 2022 Publication	<1 %

14	fdotwww.blob.core.windows.net Internet Source	<1%
15	www.porikli.com Internet Source	<1%
16	"Pattern Recognition and Computer Vision", Springer Science and Business Media LLC, 2021 Publication	<1%
17	link.springer.com Internet Source	<1%
18	open.metu.edu.tr Internet Source	<1%
19	rodmont.com Internet Source	<1%
20	Yusheng Xu, Xiaohua Tong, Uwe Stilla. "Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry", Automation in Construction, 2021 Publication	<1%
21	cupdf.com Internet Source	<1%
22	web.archive.org Internet Source	<1%

23	"Computer Vision – ECCV 2020", Springer Science and Business Media LLC, 2020 Publication	<1 %
24	Mohammed Baziyad, Ismail Shahin, Tamer Rabie, Ali Bou Nassif. "Maximizing embedding capacity for speech steganography: a segment-growing approach", Multimedia Tools and Applications, 2021 Publication	<1 %
25	S. Abbas, M. Mosbah, A. Zemmari. "Distributed Computation of a Spanning Tree in a Dynamic Graph by Mobile Agents", 2006 IEEE International Conference on Engineering of Intelligent Systems, 2006 Publication	<1 %
26	dspace.dtu.ac.in:8080 Internet Source	<1 %
27	elib.dlr.de Internet Source	<1 %
28	utomorezadwi.medium.com Internet Source	<1 %
29	Hee - Mun Park, Jin - Hyun Park. "Multi - lane recognition using the YOLO network with rotatable bounding boxes", Journal of the Society for Information Display, 2023 Publication	<1 %

30	Priya M V, Dhanya S Pankaj. "3DYOLO: Real-time 3D Object Detection in 3D Point Clouds for Autonomous Driving", 2021 IEEE International India Geoscience and Remote Sensing Symposium (InGARSS), 2021 Publication	<1 %
31	Duarte Fernandes, António Silva, Rafael Névoa, Cláudia Simões et al. "Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy", Information Fusion, 2021 Publication	<1 %
32	www.igi-global.com Internet Source	<1 %
33	"Computer Vision – ECCV 2018", Springer Nature America, Inc, 2018 Publication	<1 %
34	Addie Ira Borja Parico, Tofael Ahamed. "Chapter 11 Real-Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT", Springer Science and Business Media LLC, 2023 Publication	<1 %
35	Nohyoon Seong, Kihwan Nam. "Forecasting price movements of global financial indices using complex quantitative financial networks", Knowledge-Based Systems, 2021 Publication	<1 %

36	Rogerio Feris, James Petterson, Behjat Siddiquie, Lisa Brown, Sharath Pankanti. "Large-scale vehicle detection in challenging urban surveillance environments", 2011 IEEE Workshop on Applications of Computer Vision (WACV), 2011 Publication	<1 %
37	www.hindawi.com Internet Source	<1 %
38	Darren Tsai, Julie Stephany Berrio, Mao Shan, Stewart Worrall, Eduardo Nebot. "See Eye to Eye: A Lidar-Agnostic 3D Detection Framework for Unsupervised Multi-Target Domain Adaptation", IEEE Robotics and Automation Letters, 2022 Publication	<1 %
39	Mohammad Muntasir Rahman, Yanhao Tan, Jian Xue, Ke Lu. "Recent Advances in 3D Object Detection in the Era of Deep Neural Networks: A Survey", IEEE Transactions on Image Processing, 2020 Publication	<1 %
40	sol.sbc.org.br Internet Source	<1 %
41	Chang Nie, Zhiyang Ju, Zhifeng Sun, Hui Zhang. "3D Object Detection and Tracking Based on Lidar-Camera Fusion and IMM-UKF	<1 %

Algorithm Towards Highway Driving", IEEE Transactions on Emerging Topics in Computational Intelligence, 2023
Publication

42	Putri Wahyu Herlina, Suryo Adhi Wibowo, Agus Pratondo. "Performance Analysis of The Effect Euler Regression on Complex YOLOv4 Model for Autonomous Driving Applications", 2022 International Conference on Electrical and Information Technology (IEIT), 2022 Publication	<1 %
43	Yutian Wu, Shuwei Zhang, Harutoshi Ogai, Hiroshi Inujima, Shigeyuki Tateno. "Realtime Single-Shot Refinement Neural Network with Adaptive Receptive Field for 3D Object Detection from LiDAR Point Cloud", IEEE Sensors Journal, 2021 Publication	<1 %
44	hcisj.com Internet Source	<1 %
45	openaccess.thecvf.com Internet Source	<1 %
46	opus4.kobv.de Internet Source	<1 %
47	www.coursehero.com Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		