Abstract

Object Detection of street-level objects like human-beings, cars and potholes is an important step in the making and development of self-driving cars. The use of computer vision tools can help us to establish a cost-effective as well as accurate and precise solution to this problem. Thus, it becomes very important for us to analyse properly and choose from the variety of object detection models available to us. In this research paper, we focus on the YOLO object-detection model and compare the performances of various YOLO models that have been released over the years; YOLOv3, YOLOv4 and YOLOv7 for street-level objects that any self-driving vehicle may encounter. The best performing model can be used to detect human-beings, cars and potholes and assist self-driving cars. For the analysis part, we used a custom-dataset and divided it into training and testing sets. The training set was used for training YOLOv3, YOLOv4 and YOLOv7. On the other hand, the testing set was used for calculating the accuracy of the model. The model performances are then evaluated, compared, and analysed for object detection and classification.

Keywords

Self-Driving Cars, Computer Vision, You Only Look Once (YOLO), Object Detection, Machine Learning, Depth Analysis

1.Introduction

As technology is advancing very fast nowadays, therefore, whatever is relevant to the market today becomes irrelevant for tomorrow. Self-Driving Cars are the future and a very convenient and easy-to-use mode of transportation, however, the only reason that refrains people from adopting the same is its safety parameter. The concern includes if the car will be as safe as a traditional car and secondly, it can detect the threats accurately as well as alert the system proactively. In response to these questions, a substantial amount of research has been carried out in this field focusing on various approaches.

Accidents have been on a surge in recent years and are constantly increasing over the years as the number of automobiles is increasing. The WHO in its report published in June 2021, mentions some key facts which include that road accident deaths are around 1.4 million, GDP loss is in the range of 3% and the most vulnerable age group is 5 to 29. Moreover, India has suffered a loss of between 1.17 lakh crores to 2.91 lakh crores in 2019 due to road accidents. India, which owns nearly 1% of global vehicles, contributes to around 11% of annual deaths in accidents. Thus, having an automated system of vehicles that follow all the safety guidelines can be an effective solution to this problem as well.

Our objective is to compare and establish generic object detection models against custom-trained YOLO models, that can be used for Self-driving cars. We intend to compare custom-trained YOLO models with other available models for a comprehensive comparison between the models. The problem will be analysed by taking into consideration various factors and varying the parameters such as depth estimation to achieve the best possible accuracy of the model. Adopting such a model will allow better response to threats in the real-world scenarios using neural networks and computer vision. In future, with more improvements, we can use the model to deploy a system on cars that works in real-time and can detect accidents and provide indications that could help in reducing fatal injuries. The main contribution of this paper is:

- Drawing a comparison of performances between YOLO and other image classification models for a multiclass image classification.
- Creating a method for Depth analysis of objects in the image.

In the following sections of this paper, we shall dwell deeper into the technologies used and the literature review of recent and related work. After that we discuss about our methodologies in the work done in using the YOLO models and training along with parameters and constraints used. In the final section we explore the result and analysis of various models against our own custom-trained ones. The final section will conclude this research paper.

2.Literature Review

2.1 Object Detection: Object detection is a computer vision technique for locating instances of semantic objects in digital images or videos. It combines object proposals with classifiers to detect and recognize the objects within an image or video. Object detection has many real-world applications, such as self-driving cars, security systems, and augmented reality. One of the key benefits of object detection is its ability to localize objects in an image, not just classify them. This allows for more nuanced analysis and understanding of the visual content, leading to improved accuracy and efficiency compared to traditional image classification techniques.

In self-driving cars, object detection can be used to identify and track vehicles, pedestrians, road signs, and other objects in real-time video feeds from cameras mounted on the car. This information is then used by the autonomous vehicle's decision-making system to navigate the roads safely and efficiently. For example, the car can use object detection to detect and avoid other vehicles, anticipate traffic signals, recognize road signs and lane markings, and detect and respond to pedestrians, among other tasks. By incorporating object detection, self-driving cars can make better decisions and navigate complex road scenarios, ultimately leading to safer and more reliable autonomous driving.

There are several state-of-the-art object detection models that can be used for self-driving cars, including but not limited to:

1. YOLO (You Only Look Once): YOLO is a fast and accurate object detection model that uses a single convolutional neural network (CNN) to detect objects in real-time.
2. Faster R-CNN: Faster R-CNN is a popular object detection model that uses a region proposal network (RPN) to generate object proposals and a separate network to classify the objects.
3. Single Shot MultiBox Detector (SSD): SSD is a fast and simple object detection model that uses a single CNN to perform both object proposal and classification.
4. RetinaNet: RetinaNet is a single-shot object detection model that uses a focal loss function to address the class imbalance problem in object detection.
5. Mask R-CNN: Mask R-CNN is an extension of Faster R-CNN that adds instance segmentation to object detection, allowing for the precise localization of objects and the ability to segment each object in an image.

These are just a few examples of the many object detection models that can be used for self-driving cars. The choice of model will depend on the specific requirements and constraints of the self-driving car system, including computational resources, accuracy needs, and real-time requirements.

2.2 YOLO Model: YOLO (You Only Look Once) was introduced in 2015 by Joseph Redmon et al. It is a single deep convolutional neural network that is trained end-to-end to detect objects in an image. The YOLO architecture divides an image into an SxS grid and predicts a fixed number of bounding boxes (also called anchors) along with their confidence scores, class probabilities, and coordinates within each grid cell. The predicted boxes with high confidence scores and high-class probabilities are considered as detections. YOLO is designed to be fast and efficient, making it suitable for real-time applications, such as video surveillance or autonomous vehicles.

Also, Darknet, which is an open-source framework used to implement and train YOLO. It is written in C and CUDA and it is designed to be fast and efficient, making it an ideal choice for real-time object detection applications. The framework also includes pre-trained weights for several popular object detection tasks, as well as a demo implementation of YOLO. Additionally, it provides a flexible architecture that allows users to experiment with different network configurations and loss functions. YOLO (You Only Look Once) has undergone several improvements and upgrades since its first release in 2015. Some of the newer versions of YOLO and their improvements are:

1. YOLOv2: This version introduced anchor boxes to improve the detection of small objects and to provide better handle on the aspect ratios of detected objects. It also made several improvements to the network architecture, such as the use of batch normalization, to improve detection accuracy.
2. YOLOv3: This version introduced several key improvements, such as a deeper network architecture, better handling of scale variations, and the use of three scale-specific detection heads. YOLOv3 also introduced the concept of multi-scale training, where the network is trained on different scales of the input image to improve its ability to detect objects at different scales.
3. YOLOv4: This version introduced a new architecture called SPP-Net (Spatial Pyramid Pooling Network) that allows the network to process different scales of the input image simultaneously. YOLOv4 also introduced several new techniques, such as Mosaic data augmentation, that further improve detection accuracy.

The main advantage of YOLO over other object detection algorithms is that it processes an entire image in one forward pass, instead of sliding a window over the image and running a classifier for each window. This results in faster object detection compared to algorithms like R-CNN, Fast R-CNN, and Faster R-CNN. However, the trade-off is that YOLO may have lower accuracy compared to these algorithms because it makes some sacrifices to achieve its fast-processing speed. Nevertheless, recent improvements to the YOLO architecture, such as YOLOv7, have shown promising results in terms of accuracy and speed, making YOLO a popular choice for real-time object detection.

2.3 Faster R-CNN Model: Faster R-CNN is a two-stage object detection model that combines the strengths of both region-based convolutional neural networks (R-CNN) and single-shot object detectors (SSD). It consists of three main components: a deep convolutional neural network (CNN), a region proposal network (RPN), and a classifier.

The first component, the CNN, takes an input image and generates a feature map, which is then passed to the RPN. The RPN generates a set of anchor boxes, which are candidate regions that are likely to contain objects. The anchor boxes are then fed into the second component, the classifier, which performs object classification and bounding box regression. The classifier outputs a set of predicted object class labels, object confidence scores, and refined bounding box locations.

In the second stage of the Faster R-CNN pipeline, the predicted bounding boxes are passed through a non-maximum suppression (NMS) step to eliminate overlapping boxes and select the most confident predictions. The remaining boxes are then used to crop feature maps from the CNN and pass them to fully connected layers for final classification and bounding box refinement.

It is designed to be fast and accurate, with a two-stage pipeline that balances speed and precision. The use of anchor boxes and RPN allows for the efficient generation of high-quality object proposals, while the shared feature extractor in the CNN allows for efficient computation. These features make it a popular choice for many real-world object detection tasks.

2.4 Single Shot MultiBox Detector: Single Shot MultiBox Detector (SSD) is a deep learning-based object detection model that was introduced to address the need for fast and accurate object detection. SSD is a single-shot object detection model, meaning it processes the entire image in a single forward pass of the network, rather than requiring multiple passes as in some other models. This makes SSD fast and efficient, making it well-suited for real-time applications such as self-driving cars and security systems.

SSD uses a single convolutional neural network (CNN) to perform both object proposal and classification. The network is trained to predict the locations and class probabilities of objects within an image, using anchor boxes and predicted offsets. The anchor boxes are pre-defined bounding boxes of various aspect ratios and scales that are placed at regularly spaced locations throughout the image. The predicted offsets are used to refine the position and shape of the anchor boxes to match the objects more accurately within the image.

One of the notable advantages of SSD is its capability to manage objects of varying sizes within a single image, rendering it appropriate for projects that require detection of objects of diverse scales. Additionally, through the use of anchor boxes and predicted offsets, SSD can detect multiple objects within a single image, making it ideal for object detection tasks with high density.

2.5 RetinaNet: RetinaNet is a deep learning-based object detection model that was introduced to address the issue of class imbalance in object detection. In many object detection datasets, the number of background (negative) samples is much larger than the number of object (positive) samples. This leads to a high false positive rate and low precision in object detection models that are trained on these datasets. It solves this problem by introducing a new loss function, called focal loss, that down-weights the contribution of easy examples in the loss calculation. This allows the model to focus on the hard, foreground object samples and improve its detection accuracy.

It is a single-shot object detection model that uses a combination of anchor boxes and a feature pyramid network (FPN) to efficiently detect objects in an image. The model is designed to address the imbalance between foreground objects and background pixels that is common in object detection tasks, and to improve the accuracy and efficiency of object detection.

Overall, it is a powerful and effective object detection model that has achieved state-of-the-art performance on a variety of benchmark datasets, making it well-suited for real-time applications such as self-driving cars and security systems. Its combination of high accuracy and efficiency make it a popular choice for many real-world applications.

2.6 Mask R-CNN: Mask R-CNN is a deep learning-based object detection and segmentation model. It is an extension of the popular Faster R-CNN model, adding the ability to perform instance segmentation in addition to object detection.

It consists of two main components: a region proposal network (RPN) that generates object proposals and a separate network that performs object classification and instance segmentation. The RPN is trained to predict the locations of objects in an image, while the second network is trained to classify the objects and generate a binary mask for each object, indicating its precise location within the image.

One of the key benefits of Mask R-CNN is its ability to perform both object detection and instance segmentation in a single forward pass of the network, making it more efficient and effective compared to other models that require separate models for each task. Additionally, the model is capable of handling instances of objects with arbitrary shapes, making it well suited for tasks that require detailed object information. It is a powerful and versatile object detection and segmentation model that has seen widespread use in a variety of computer vision applications, including self-driving cars, medical imaging, and satellite imagery analysis.

3. Related Work

Over the past several years, numerous studies have compared the effectiveness and efficiency of various object detection algorithms in specific contexts and environments. However, research has yet to fully explore the capabilities of newer algorithms such as YOLOv7. There is currently a scarcity of research that compares YOLOv7 with its predecessors or other object detection algorithms, particularly in the domain of detecting multiple objects on roads.

Multiple studies have evaluated object detection algorithms that are tailored to detecting a particular type of object on roads. For example, Choyal and Singh's research paper [1] compared Faster-RCNN and SSD MobileNet in detecting traffic signs. Their findings indicated that Faster-RCNN is more precise but takes longer to train than MobileNet SSDs.

Detecting humans is a critical task in road object detection. Kim et al. [2] conducted a comparative study of various object detection algorithms, including Faster-RCNN, YOLO, SSD, and R-FCN algorithms. The results of their study showed that YOLOv3 is the most suitable algorithm for detecting people as it provides reasonably accurate results in a reasonable amount of time.

Another object that often appears on the road, namely potholes, has been raised as a topic of object detection by P.Ping et al. [3]. Her paper found that YOLOv3 is the most efficient algorithm because its speed and detection results are more reliable than HOG, SSD, and Faster-RCNN. In addition, other objects such as trees also often appear on the roadside. A comparison of YOLOv3-SPP and YOLOv3 was carried out by Z. Yinghua et al. [4]. The result is that YOLOv3-SPP is claimed to be the most suitable for detecting trees.

Based on the studies, YOLO object detection model has been selected as the preferred object detection algorithm or primary research subject. Where previous versions of the YOLO model have achieved great performance and lower inference time than Faster-RCNN and other object detection algorithms. Numerous studies have been conducted to compare YOLO object detection algorithms. It is worth noting that the majority of comparative studies on object detection in road cases have focused on YOLOv3, Faster-RCNN, and SSD MobileNet. While there are limited studies utilizing the YOLOv5 algorithm, there are even fewer studies that have explored the performance of the YOLOv7 algorithm in comparison to its predecessors, particularly in detecting multiple objects on roads.

4. Methodology

This section provides a more detailed explanation of the YOLO object detection algorithms. It is then followed by a discussion of the dataset and how it was split, as well as the pre-processing and augmentation stage used for images in the dataset. Lastly, each object detection algorithm is trained and compared.

4.1 YOLO Object Detector Model:

The YOLO (You Only Look Once) object detection algorithm is unique compared to previous detectors, as it uses a single convolutional neural network. It approaches object detection as a regression problem by predicting bounding boxes and associated class probabilities directly from full images in a single evaluation. This differs from traditional methods that rely on multiple stages to identify objects in an image.[5] The design comprises several components, which can be divided into three main parts. Firstly, there is the input, which refers to the collection of training images that are fed into the network and processed in parallel batches using the GPU. Secondly, there is the Backbone and Neck, which are responsible for extracting and aggregating features. Finally, there is the Object Detector, which consists of the Detection Neck and Detection Head. Together, these components form the Object Detector.

The YOLO algorithm divides any given image into a grid of S×S cells. Each cell in the grid predicts a fixed number of anchor boxes for an object. For each anchor box, the network outputs four offset values (b_x, b_y, b_h, b_w), one confidence value p_c, and C conditional class probabilities. The coordinates (b_x, b_y) represent the centre of the bounding box relative to the bounds of the grid cell in the input image. The b_h and b_w represent the height and width of the box, respectively. The confidence value p_c is the probability that a box contains an object and how accurate the boundary box is, which is equivalent to $Pr(object) * IoU$. C conditional class probabilities, $Pr(class_i \mid object)$, are the probabilities that the object belongs to a particular $class_i$ given that an object is present.

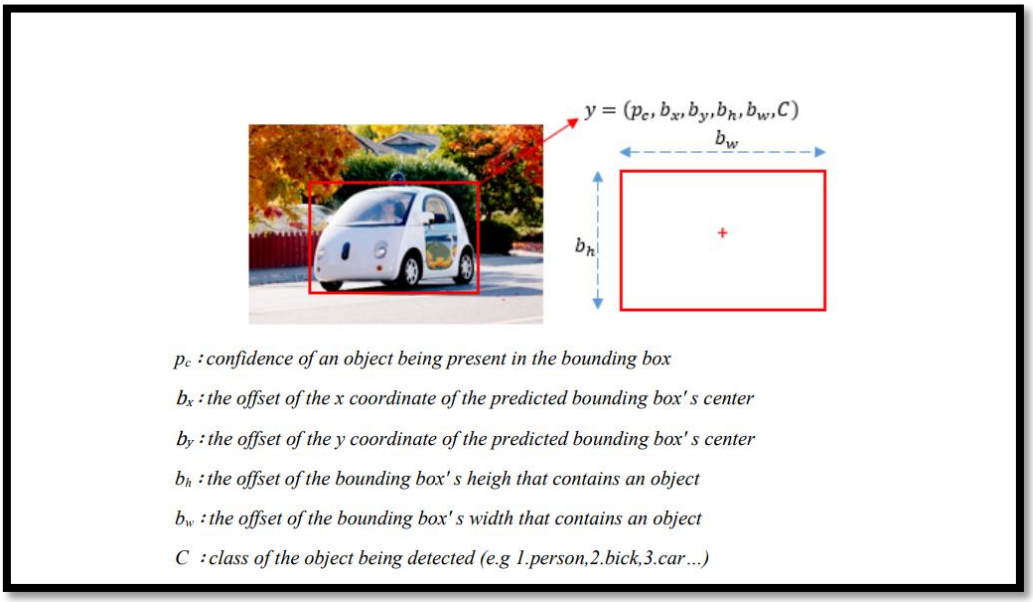


Fig 1

Normalisation: To normalize the bounding box's width and height, they are divided by the image's width and height, respectively. This ensures that the values of width and height fall between 0 and 1. Similarly, the x and y coordinates of the bounding box are offset with respect to the grid cell location, so they also fall within the range of 0 and 1 [].

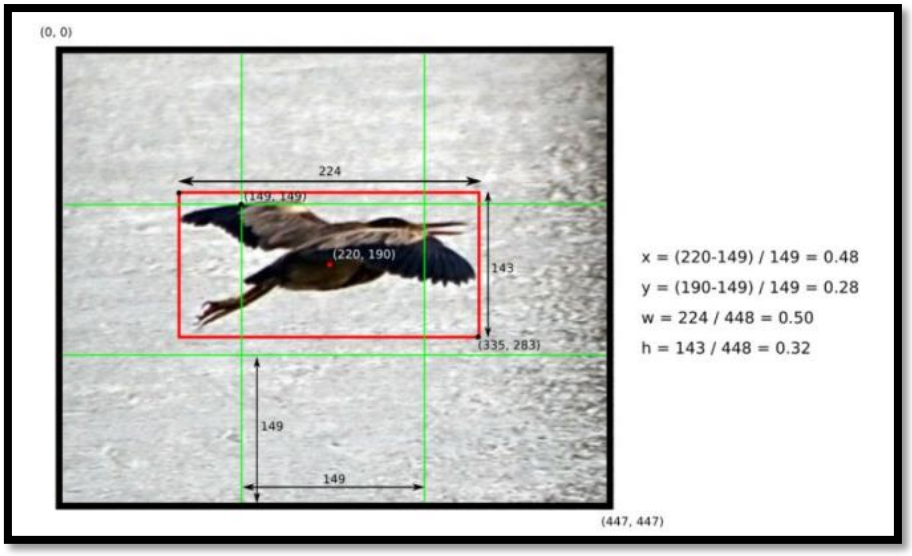
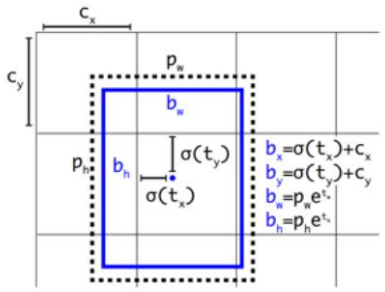


Fig 2

YOLO utilizes the sigmoid function to predict the coordinates of the box's center relative to the location of filter application, as well as the width and height of the box as offsets from cluster centroids.



$b_x=\sigma(t_x)+c_x$
$b_y=\sigma(t_y)+c_y$
$b_w=p_w\, e^{tw}$
$b_h=p_h e^{th}$
$Pr(object) * IoU(b,object) = \sigma(t_0)$
Where
t_x, t_y, t_w, t_h are predictions made by YOLO.
c_x, c_y is the top left corner of the grid cell of the anchor.
p_w, p_h are the width and height of the anchor.
c_x, c_y, p_w, p_h are normalized by the image width and height.
b_x, b_y, b_w, b_h are the predicted boundary box.
$\sigma(t_0)$ is the box confidence score.

Fig 3

Non-Maximal Suppression: A potential issue with the algorithm is when it predicts multiple bounding boxes for one class. To address this, a non-max suppression algorithm is used. The algorithm works by first selecting the box with the highest probability for a particular class. Then, it compares this box to all other boxes of the same class using intersection over union (IoU), which is a metric that measures the overlap between two bounding boxes. The box with the highest probability is kept, and any overlapping boxes with an IoU above a certain threshold (usually 0.5) are discarded. This process is repeated for all classes and all remaining boxes until no more boxes are left or all boxes have been processed. The result is a set of non-overlapping bounding boxes with their associated class probabilities [].

If the IoU between two boxes is higher than the predefined threshold (e.g., 0.5), then the box with the lower probability is excluded since it is likely to be a duplicate prediction of the same object. This process is repeated until there are no more boxes left to consider or all boxes have been processed. The remaining boxes are then considered as the final object predictions.

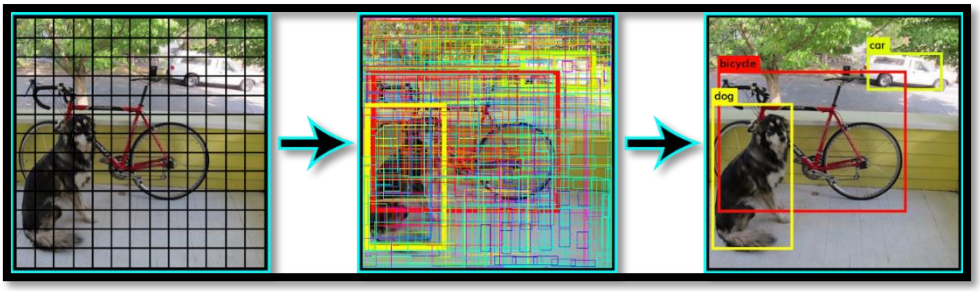


Fig 4

Vector Generalisation: YOLO predicts one object per grid cell, but this is not a hard rule. The original paper actually used two boxes per cell, each of which could only belong to one class. This means that the vector representing each cell contains two sets of probabilities and dimensions for the boxes. If we want to detect more classes, we simply increase the number of classes in the vector. However, this can affect the algorithm's accuracy. To detect more objects, we can also increase the size of the grid. In general, each cell can be represented by a vector with dimensions of B x 5 + C, where B is the number of bounding boxes and C is the number of classes. If the image is divided into an S x S grid, the entire image can be represented as an S x S x (B x 5 + C) tensor [].

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
1x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64 x 64
2x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
8x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
8x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
4x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

YOLOv3: In YOLOv3, there are three different scales, and at each scale, three anchor boxes are predicted. For our custom dataset, the tensors at each scale are of size N x N x [3 x (4 + 1 + 3)], where 4 represents the bounding box offsets, 1 represents the objectness, and 3 represents the class predictions []. To determine the priors (anchors), YOLOv3 applies the k-means cluster. Nine clusters are pre-selected, with widths and heights of (10x13), (16x30), (33x23), (30x61), (62x45), (59x119), (116x90), (156x198), and (373x326), which are split evenly across the three scales. Each group is assigned to a specific feature map for detecting objects.

In addition, YOLOv3 uses a new network called Darknet-53 for feature extraction. The network uses successive 3x3 and 1x1 convolutional layers, with some shortcut connections that are relatively larger than those in YOLO 9000 with 19 layers, and YOLOv1 with 24 layers. Darknet-53, with its 53 convolutional layers, performs more accurately in extracting features than Darknet-19 adopted in YOLOv2 or other networks like ResNet [].

YOLOv4: The architecture of YOLOv4 includes several components, such as CSPDarknet53 as the backbone, a spatial pyramid pooling module, a PANet path-aggregation neck, and a YOLOv3 head. CSPDarknet53 is a new type of backbone that has been designed to improve the learning ability of convolutional neural networks. By combining these different components, YOLOv4 is able to achieve better performance in object detection tasks [].

The additions to YOLOv4 include Bag of Freebies (BoF) & Bag of Specials (BoS). The Bag of Freebies (BoF) is a set of techniques that can enhance the performance of a neural network without increasing its inference time. These techniques primarily consist of data augmentation methods, which involve creating various versions of a single image to make the network more adaptable and capable of making accurate predictions. By using BoF techniques, the neural network can become more robust and effective in its ability to detect and classify objects in images []. The Bag of Specials (BoS) refers to a set of strategies that can significantly enhance the performance of a neural network, although they may slightly increase inference time. Unlike the Bag of Freebies (BoF) which focuses on data augmentation techniques, BoS methods add specialized layers and modules to the neural network architecture. These additions can lead to a significant improvement in the network's ability to detect and classify objects in images. While there may be a slight increase in inference time due to the added complexity, the performance gains are usually well worth it [].

YOLOv7: Several architectural improvements have been introduced in YOLOv7 to enhance both its speed and accuracy. Like Scaled YOLOv4, YOLOv7 uses backbones that are not pre-trained on ImageNet. Instead, they are trained exclusively on the COCO dataset. This is not surprising, given that YOLOv7 was developed by the same authors as Scaled YOLOv4, which is an extension of YOLOv4. The YOLOv7 paper outlines several significant changes that have been made to the architecture, which we will discuss in detail []. The architectural reforms include E-ELAN (Extended Efficient Layer Aggregation Network) and Model Scaling for Concatenation-based Models. The YOLOv7 backbone includes a computational block called E-ELAN. This block is inspired by prior research on improving network efficiency and has been specifically designed to consider various factors that can affect both speed and accuracy [].

4.2 Dataset

The dataset that we use in this paper is a Custom Dataset gathered from a variety of sources, like Kaggle [] and Roboflow [], that contains 5,635 images with a total of 45,655 annotations. We have used LabelImg tool for creating yolo-format annotations from the images. The classes along with their images and annotations is mentioned in the following Table 1.

Class	Images	Annotations
Human	2,500	26,401
Car	2,500	18,179
Potholes	635	1,075

Table 1: Class Balance

Splitting of dataset: The experiment is being carried out on the dataset mentioned earlier, with the aim of comparing different algorithms for object detection. The dataset is split into three sets of data: Train, validation, and test data, which are shown respectively.

Class	Distribution
Train	3,965
Test	1,067
Validation	383

Table 2: Dataset Distribution

4.3 Pre-processing of dataset

The YOLO models use rescaling and cropping to adjust the images to a specified aspect ratio based on the input layer sizes. After rescaling, the dataset undergoes data augmentation, which allows for different transformations such as geometric and colour changes, such as scaling, transformation, and colour transformation, to be applied to the images. The images are pre-processed by being resized to 416 × 416 pixels while maintaining the original aspect ratio and auto-orientation.

4.4 Model tuning with Hyperparameters

The paper presents a comparison of different object detection algorithms based on frame data. The evaluation criteria used to compare the models include precision, recall, F1-score, mAP values, and inference time. For this experiment, YOLOv3, YOLOv4, and YOLOv7 were implemented using the darknet framework and trained for 97 epochs using the Stochastic Gradient Descent (SGD) optimizer. A comprehensive outlook of the hyperparameters is in the following Table 3.

Hyperparameters	
Batch Size	64
Optimizer Momentum	0.9
Weight Decay	0.0005
Learning Rate Scheduler	Cosine
Base Learning Rate	0.001
Maximum Batches	6000
Filters	24

Table 3: Hyperparameters for training

5. Performance Metrics

Various performance metrics can be employed to assess the effectiveness of an object detector, with accuracy (measured by mean average precision, mAP). To evaluate accuracy, different methods can be used, but mAP is the primary metric. This paper uses precision, recall, F1-score and mAP as evaluation criteria for object detection models. However, before analysing these metrics, it is important to understand basic concepts such as confidence score, Intersection over Union (IoU), precision and recall.

5.1 mAP as a metric for model evaluation

The confidence score is a probability value that represents the likelihood of an anchor box containing an object, and it is usually predicted by a classifier in object detection algorithms. The ground truth bounding box is the expected output of an algorithm on a given input, such as a manually labelled bounding box in the testing set that identifies where objects are located in the image. The predicted bounding box is a rectangular region generated by the model detector that indicates the predicted

location of the object. Intersection over Union (IoU) is a performance metric used to measure the overlap between the predicted bounding box and the ground truth bounding box. It quantifies the area encompassed by both the predicted and ground-truth bounding boxes.

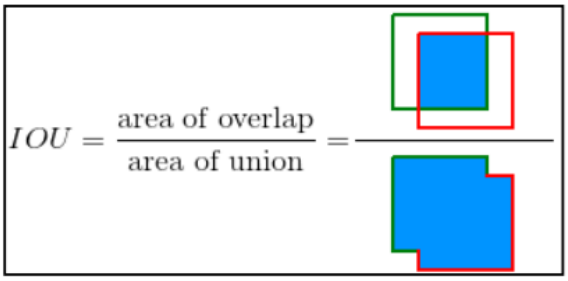


Fig 6

The threshold principle involves predefining an IoU limit, such as 0.5, to distinguish between true positives and false positives in object detection. A true positive test result indicates that the condition being tested for is present and has been correctly detected. A true negative test result indicates that the condition is absent and has been correctly identified as such. A false positive test result occurs when the condition is absent but is mistakenly detected, while a false negative test result occurs when the condition is present but is missed or not detected by the test. On the basis of these terms, we can go on to define the various evaluation metrics, that we undertake in this paper.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

Fig 7

Precision is a performance metric that calculates the ratio of accurate predictions to the total number of predicted instances, including both true and false positives. It measures the accuracy of the model's predictions.

Recall is a performance metric that measures the ability of an algorithm to detect all positive cases. It quantifies the fraction of true positives found by the model out of the total number of actual positive instances in the dataset.

The F1-Score is a performance metric that reflects the trade-off between precision and recall. It is a single value between 0 and 1 that balances the precision and recall values of the model. A higher F1-Score indicates that the model has achieved a better balance between precision and recall.

Thus, we have covered enough ground to properly define mAP. The mean average precision (mAP) is a performance metric used to evaluate the overall performance of an object detection algorithm for all object classes. It is calculated by taking the average of the average precision values across all object classes. In this experiment, different Intersection over Union (IoU) values were used to calculate mAP. IoU measures the overlap between the predicted box and the ground-truth box by computing the

percentage match area of intersection. mAP@.5 indicates that the threshold of IoU used for computing mean average precision is 0.5.

6. Result and Discussion

Here we evaluate and compare the different models that we have talked about in our paper with the focus on YOLO models. We can see that YOLOv4 has strangely outperformed all the other competitor object detection models with a mAP value of 72.97%. Although YOLOv7 the latest state-of-the-art object detection out of all of them, is able to achieve the maximum precision of 0.753 in our evaluation, still it failed to beat YOLOv4 in terms of accuracy and has a mAP value of 56.65%, below YOLOv3 and YOLOv4. It is important to note that the reported performance metrics are dependent on various factors, such as the specific hardware and software configuration, input image size, and batch size used during training and testing, all of which have been mentioned in the above section.

Measure	YOLOv3	YOLOv4	YOLOv7	Faster RCNN	SSD	RetinaNet	Mask RCNN
Precision	0.671	0.692	0.753	0.377	0.354	0.447	0.378
Recall	0.467	0.660	0.519	0.362	0.228	0.419	0.355
F1-Score	0.553	0.673	0.610	0.369	0.277	0.507	0.363
mAP@.5	57.38%	72.97%	56.65%	36.42%	31.52%	53.84%	35.13%

Table 4: Evaluation of various Object Detection models

It is also worth noting that all the YOLO models outdid all the other models. RetinaNet is a state-of-the-art model and has produced expected results with a mAP value of 53.84%. SSD, although a fast detection model, once again suffers with poor accuracy and has a low mAP value of 31.52% and this issue has persisted for a long time. Faster R-CNN and Mask R-CNN have also produced expected results with mAP scores of 31.52% and 35.13%.

7. Depth Analysis

In this section, an experiment was performed to establish a relationship and derive an equation between the distance of the car and any obstacles detected in the frame data. We have selected two parameters for our experiment, which are:

- Pixel - this represents the area that the object occupies in the image
- Distance - this represents the distance between the object and the car.

To gather data for our experiment, we positioned a car at different distances from the camera, captured pictures, and created nine sets of photos. We then used YOLOv4 to detect the car in each image and obtain its coordinates. From these coordinates, we drew a bounding box around the car and calculated its area, which we displayed. We took pictures of a parked car at different distances from another car and captured 9 sets of photos. Then, we used YOLOv4 to detect the car and get its coordinates. We drew a bounding box around it and calculated its area.

Area (pixel2)	Distance (m)
440378	2
239600	3
137134	4
96661	5
67623	6
47297	8
33628	10
25482	11
12167	16

Table 5: Pixel-area and respective distance values

We plotted a graph with the area on the x-axis and the distance on the y-axis. For now, we will be using a power function to generate an equation ($Y=4319.3x^{-0.589}$) that best fit our observed values. We will now use this equation to judge the distance when an object is detected. We will simply calculate its pixel-area and substitute that value into the equation to get its distance from the car. This should solve our main objective.

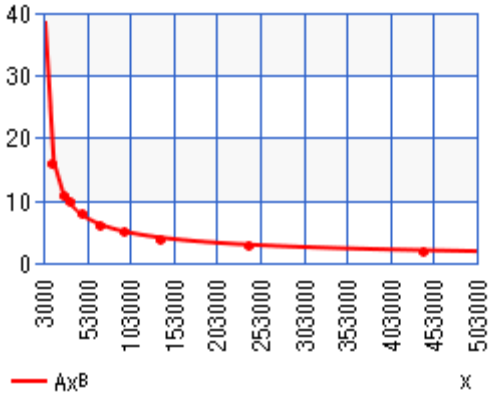


Fig 8

8. Conclusion and Future Work

The experimental results showed that the YOLOv4 deep learning algorithm is a viable solution for detecting street-level objects such as cars, humans, and potholes for use in self-driving cars. It emerged as the best detection algorithm according to our hyper-parameterisation for multiple objects for self-driving cars.

In this research paper, several object detection algorithms were compared for their ability to detect street-level objects such as humans, cars, and potholes in the context of self-driving cars. The algorithms evaluated included SSD, YOLOv3, YOLOv4, YOLOv7, ResNet, Mask R-CNN, and Faster R-CNN, using a custom dataset prepared from various sources, and augmented with rescaling, auto-orientation, and hue shifting. The results showed that YOLOv4 was the most accurate algorithm among the tested models. Future work may involve evaluating new object detection models and comparing them to existing ones, using a larger dataset for training and testing, incorporating additional evaluation metrics such as inference time or FPS, and optimizing the depth analysis functionality to improve output. Overall, there is room for improvement and further research in this area.

9. References

[1] Samiksha Choyal and Ajay Kumar Singh. An Acoustic based Roadside Symbols Detection and Identification using Faster RCNN and SSD. In 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), pages 1–4, Lakshmangarh, India, February 2020. IEEE.

[2] Chloe Eunhyang Kim, Mahdi Maktab Dar Oghaz, Jiri Fajtl, Vasileios Argyriou, and Paolo Remagnino. A Comparison of Embedded Deep Learning Methods for Person Detection. 2018. Publisher: arXiv Version Number: 2.

[3] Ping Ping, Xiaohui Yang, and Zeyu Gao. A Deep Learning Approach for Street Pothole Detection. In 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService), pages 198–204, Oxford, United Kingdom, August 2020. IEEE.

[4] Zhao Yinghua, Tan Yu, and Liu Xingxing. Urban Street tree Recognition Method Based on Machine Vision. In 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), pages 967–971, Xi'an, China, April 2021. IEEE.

[5] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You only look once: Unified, real-time object detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[6]

[7]

[8]

[8]

[10]

[11]

[12]

[13]

[14]

[15]