Autonomous Driving Using YOLO

Varun Reddy Chaykam
Artificial Intelligence
SRM Institute of Science and Technology
Chennai, India
cc4243@srmist.edu.in

Venkata Sanjeevi Sai Prasanna Emandi
Artificial Intelligence
SRM Institute of Science and Technology
Chennai, India
ee5151@srmist.edu.in

Satya Koushik Manchikanti
Artificial Intelligence
SRM Institute of Science and Technology
Chennai, India
mm3433@srmist.edu.in

Sai Rupesh Naidu Relli
Artificial Intelligence
SRM Institute of Science and Technology
Chennai, India
sr6932@srmist.edu.in

Abstract **-Autonomous driving has emerged as a transformative technology with the potential to revolutionize the automotive industry. This paper explores the integration of the YOLO (You Only Look Once) object detection algorithm into autonomous vehicles to enhance their perception and decision-making capabilities. YOLO, renowned for its real-time object detection efficiency, enables vehicles to rapidly identify and track objects in their surroundings. The paper delves into the architecture of YOLO and its adaptation for autonomous driving scenarios. It discusses the algorithm's ability to process high-resolution images in real-time, enabling swift and accurate detection of pedestrians, vehicles, and obstacles. The integration of YOLO contributes to the development of robust and responsive autonomous systems, enhancing safety and efficiency on the road. Furthermore, the paper explores the challenges associated with deploying YOLO in real-world driving conditions, addressing issues such as occlusion, varying lighting conditions, and complex traffic scenarios. Insights from practical implementations and case studies illustrate the effectiveness of YOLO in autonomous driving applications. This research contributes to the ongoing discourse on advancing autonomous technologies, providing a foundation for future developments in enhancing the perception capabilities of self-driving vehicles.**

KeyWords: You only Look Once (YOLO), BottleNeck Cross-Stage Partial (CSP), Autonomous.

I. INTRODUCTION

Autonomous driving, a paradigm-shifting technology, has revolutionized the automotive industry by integrating artificial intelligence (AI) and computer vision systems. One of the pioneering technologies powering autonomous vehicles is the You Only Look Once (YOLO) algorithm.

The concept of autonomous vehicles traces its roots back to the early 20th century, with visionary inventors foreseeing a future where machines could navigate without human intervention. However, it wasn't until recent decades that significant strides were made, thanks to advancements in computing power, AI, and sensor technologies. Autonomous driving relies heavily on the ability of vehicles to perceive and interpret their surroundings accurately—a task where computer vision plays a pivotal role.

Computer vision empowers machines with the capability to interpret and make decisions based on visual data. In the context of autonomous driving, this involves recognizing and understanding elements of the environment, such as pedestrians, vehicles, traffic signs, and road markings. YOLO, a groundbreaking object detection algorithm, has emerged as a key player in enhancing the precision and efficiency of these tasks.

Developed by Joseph Redmon and Santosh Divvala, YOLO represents a departure from traditional object detection methods. Instead of processing an image multiple times at different scales, YOLO processes the entire image in a single forward pass through a neural network. This unique approach allows YOLO to achieve real-time object detection, a critical

requirement for the dynamic and fast-paced environment of autonomous driving.

The significance of YOLO in autonomous driving lies in its ability to accurately and swiftly identify objects in each scene. Traditional object detection methods often struggle with balancing accuracy and speed, making them less suitable for real-time applications like self-driving cars. YOLO's efficiency stems from its ability to simultaneously predict multiple bounding boxes and class probabilities for each object in the image.

In addition to its technical prowess, YOLO's open-source nature has fostered a collaborative community of developers and researchers. This collaborative spirit has led to continuous improvements and adaptations of the YOLO algorithm, making it versatile and applicable to a range of autonomous driving scenarios.

This study investigation endeavours to augment our comprehension of diverse iterations of YOLO, elucidating their distinct and singular attributes. The models fashioned during this study pursuit have been operationalized on assorted images capturing traffic scenarios, thereby facilitating a nuanced exploration of disparities among various renditions of YOLO.

II. LITERATURE SURVEY

In our pursuit of the best Yolo model, we encountered various scientific researchers and papers. Our conclusions, drawn from these studies, provided valuable insights into the strengths and nuances of different Yolo models.

The first paper by Dong et al. introduces a lightweight vehicle detection network model based on YOLOv5s. The incorporation of C3Ghost and Ghost modules into the YOLOv5 neck network addresses the challenges of high FLOPs, large model files, and slow computational speed. Additionally, the paper introduces the Convolutional Block Attention Module (CBAM) to enhance critical information selection, improving detection accuracy. The utilization of the Complete Intersection over Union Loss (CIoU_Loss) as a regression loss function further enhances bounding box regression speed and accuracy.

Rajib Ghosh's paper focuses on on-road vehicle detection under varying weather conditions using Faster R-CNN with multiple Region Proposal Networks (RPNs). The novel approach considers challenging weather scenarios, and the introduction of multiple RPNs of varying sizes within the Faster R-CNN structure enhances accuracy in detecting vehicles of different sizes.

Yingfeng Cai, Ze Liu, Xiaoqiang Sun, Long Chen, Hai Wang, and Yong Zhang propose a deep vehicle detection algorithm based on the dual-vehicle deformable part model to address false positivity in partially obscured vehicles. The deep learning framework, involving feature extraction, deformation processing, occlusion handling, and classifier training using backpropagation, proves effective in enhancing detection accuracy, particularly in scenarios involving occlusion.

Ying Guo, Rui-lin Liang, You-kai Cui, Xiang-mo Zhao, and Qiang Meng present a domain-adaptive method for vehicle target detection in foggy weather. Their contributions include the construction of a foggy dataset, the development of an improved generative confrontation network (CPGAN) for image style transfer, and the formulation of a YOLOv4 target detection framework based on domain adaptation. The proposed method achieves real-time detection, addresses the lack of foggy datasets, and exhibits higher accuracy and speed compared to traditional defogging detection methods.

These papers collectively contribute to advancing object detection in varying scenarios, incorporating novel modules, addressing specific challenges, and introducing innovative approaches to enhance detection accuracy and speed.

III. YOLO ARCHITECTURE

Yolo's strength lies in its straightforward architecture, enhanced by intricate mathematical equations that contribute to its remarkable accuracy, robustness, and speed. While its topology is simple, the complexity of the underlying mathematical computations is the key to Yolo's impressive performance, making it adept at real-time object detection with precision and efficiency.

1) YOLOv5

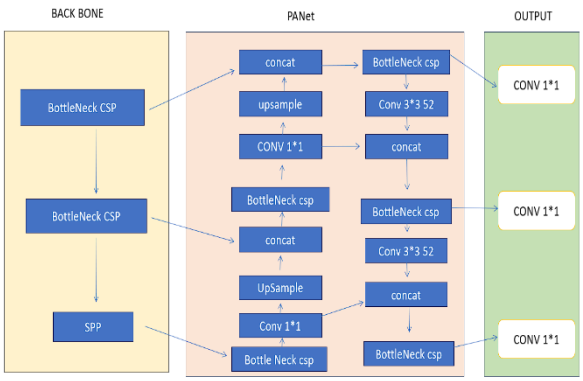

Fig. 1. Architecture of Yolov5

1.1) Backbone

YOLOv5 adopts CSP-Darknet53 as its backbone and incorporates the CSPNet strategy to combat the vanishing gradient problem. This involves the use of residual and dense blocks to optimize information flow.

By partitioning the base layer's feature map, CSPNet minimizes redundant gradients and efficiently merges them through a cross-stage hierarchy. The resulting model, with reduced parameters, exhibits enhanced computational efficiency, translating to a significant boost in inference speed. This strategic combination is pivotal for real-time object detection in YOLOv5, showcasing its capability to achieve effective and rapid processing of visual data in diverse applications.

1.2) Neck of YOLOv5

YOLOv5 introduces significant changes to its model neck, incorporating a modified version of Spatial Pyramid Pooling (SPP) and adapting the Path Aggregation Network (PANet) with the integration of BottleNeckCSP. PANet, a feature pyramid network utilized in prior YOLO versions like YOLOv4, is designed to enhance information flow and facilitate accurate pixel localization for mask prediction tasks. YOLOv5 enhances PANet by applying the CSPNet strategy, as illustrated in the network architecture figure, contributing to improved feature aggregation and overall performance in tasks such as object detection and segmentation.

The Spatial Pyramid Pooling (SPP) block in YOLOv5 aggregates information from its inputs, generating a fixed-length output. This design offers the advantage of significantly expanding the receptive field and extracting crucial contextual features without compromising network speed. Previously employed in YOLO versions like yolov3 and yolov4, the SPP block separated vital features from the backbone. In YOLOv5, the SPPF variant is introduced, optimizing network speed while maintaining the benefits of enhanced context feature extraction. This modification contributes to the model's efficiency, ensuring that it processes information rapidly without sacrificing the ability to capture relevant contextual details.

1.3) Output

YOLOv5 outputs classes, bounding boxes, and objectness scores. It employs Binary Cross Entropy loss (BCE) for classes and objectness loss, and Complete Intersection over Union (CIoU) for location loss. The final loss formula integrates these components, ensuring a comprehensive optimization strategy for accurate object detection, classification, and localization in diverse scenarios.

$Loss=\lambda_1 * L_{cls}+\lambda_2 * L_{obj}+\lambda_3 * L_{loc}$

2)Yolov3

2.1) Input

The YOLOv3 feature detector architecture drew inspiration from prominent models such as ResNet and Feature Pyramid Network (FPN). YOLOv3 feature detector Darknet-53, it featured 52 convolutions with ResNet-like skip connections and 3 prediction heads akin to FPN. This design allowed YOLOv3 to efficiently process images at varying spatial resolutions.

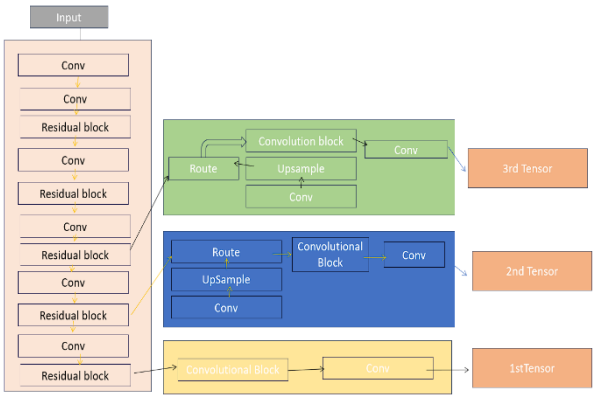

Fig. 2. Architecture of Yolov3

Feature Pyramid Network (FPN), introduced by Facebook Artificial Intelligence Research (FAIR) in 2017, is a pivotal component in YOLOv3. It involves a topology where a feature map gradually decreases and then increases in spatial dimension, being concatenated with previous feature maps of corresponding sizes. YOLOv3 employs three detection heads, each processing differently sized feature maps. The detection filter's size is 1 x 1 x (B x (4 + 1 + C)), where B is the number of predicted bounding boxes (3 in YOLOv3), "4" denotes bounding box attributes, "1" represents objectness predictions, and C is the number of class predictions (80 for COCO dataset). This yields a filter size of 1 x 1 x 255 in YOLOv3.

2.2) Upsampling and concatenation

In the transformative step of upsampling and concatenation, the feature map is selectively extracted from the prior two layers and subjected to a 2x upsampling process. This encompasses the crucial upsampling aspect. Additionally, a feature map originating from an earlier segment of the network is intricately merged with the upsampled features, constituting the concatenation process. This strategic methodology closely mirrors the architecture of an encoder-decoder model.

The application of upsampling and concatenation proves instrumental in capturing a more nuanced understanding of the input, combining both semantic and fine-grained information effectively. To further refine this amalgamated knowledge, several additional convolutional layers are introduced. These layers process the encapsulated feature map, ultimately predicting a tensor that is twice the size of the original, showcasing the model's intricate feature extraction and refinement capabilities.

2.3) Input resolution augmentation

YOLOv3 adopts a fully convolutional network design, distinguishing itself from networks employing fully connected layers for classification. This architecture allows YOLOv3 to process images of varying sizes. However, this approach could lead to an abundance of network parameters, addressed through resolution augmentation. YOLOv3 strategically employs 10 different resolution steps, ranging from 384 x 384 to 672 x 672 pixels. These resolutions alternate randomly within training batches, fostering generalization across various image resolutions. Notably, in YOLOv3, each spatial cell in the output layer predicts three boxes, known as anchors, centerd within the cell, facilitating effective object localization.

The loss for every box prediction is based on the following terms.

Coordinate Loss: Penalizes inaccurate bounding box predictions, refining object localization.

Objectness Loss: Penalizes inaccurate predictions of box-object Intersection over Union.

Classification Loss: Ensures precise class predictions within the detected bounding box.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$
$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2$$
$$+ \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

Fig. 3 Loss Function of Yolov3

IV. METHODOLOGY

This paper explores the implementation of Yolov3 and Yolov5 models for automating vehicle-related tasks, aiming to identify the superior performer in this technological domain. Our methodology aligns with standard machine learning approaches, providing a comprehensive analysis of both models to determine their effectiveness in enhancing vehicle automation capabilities.

i) Data Collection:

The dataset for our investigation, acquired from Kaggle, consisted of an extensive 22,000 images and an accompanying CSV file detailing bounding box information for each image across five distinct classes namely cars, truck, pedestrian, light and bicyclist. To streamline our training process, we judiciously reduced the image count to 9,000 without compromising the study's integrity. Subsequently, our data preprocessing involved a meticulous curation phase. Duplicate instances were systematically identified and removed, contributing to a refined and non-redundant dataset. Furthermore, a deliberate decision was made to exclude data associated with bounding boxes lacking corresponding image captures. This strategic refinement not only optimized computational efficiency but also ensured the dataset's fidelity, laying the groundwork for a more focused exploration of object detection nuances within our chosen classes.

ii) Feature Engineering

During this phase, we aligned our dataset with the requirements of the YOLO model, which necessitates bounding boxes specified as the x and y coordinates of the center, along with the height and width dimensions. The original dataset presented bounding box information in the form of corner coordinates. To address this disparity, a recalibration process was implemented. This involved computing the center coordinates, height, and width of each bounding box, ensuring compatibility with the YOLO model specifications. This transformation facilitated seamless integration of our dataset with the model, laying the groundwork for more effective and accurate object detection during subsequent training and evaluation phases.

Calculated as:

$X_{center} = (X_{max+}X_{min}\ /2)$/width of image

$Y_{center} = (Y_{max}\text{-}Y_{min}\ /2)$/height of image

$Width_{bbox} = (X_{max}\text{-}X_{min})$/height of image

$Height_{bbox} = (Y_{max}\text{-}Y_{min})$/height of image

iii) Data Visualization:

Diverse visualization techniques have been deployed to unveil latent insights within the dataset, allowing an exploration of the distribution of images across different classes. This process involves assessing the frequency and representation of images associated with each specific class, shedding light on the dataset's composition and aiding in the identification of potential imbalances or focal points within the classification framework.

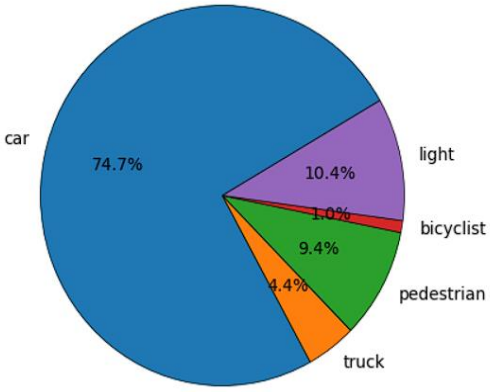

Fig 3. Bounding Boxes of each class

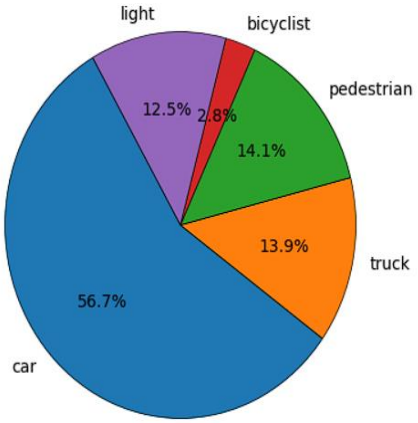

Fig 4. Images of each class

iv) Data Splitting:

The dataset underwent segmentation into requisite training and validation subsets tailored to the YOLO model's specifications. Following this, for YOLO v5, a YAML file was meticulously constructed, encapsulating essential information such as class names and the respective paths to the training and validation images. In contrast, YOLO v3 demanded a two-fold approach. It necessitated the creation of a NAME file, enumerating class names, and a DATA file specifying the paths of the images along with the associated NAME file. These configuration files played a pivotal role in facilitating seamless integration and alignment of the datasets with the distinct requirements of YOLO versions 3 and 5.

v) Training YOLO:

The training process involved cloning GitHub repositories from Ultralytics for YOLOv5 and Darknet for YOLOv3. These repositories encompassed the requisite model codes and configuration files essential for the training regimen. Upon completion of model training, mean average precision (mAP) values were computed and saved alongside the corresponding model weights. Insightful visualizations capturing hidden patterns and nuances during the training phase were generated and stored as images. These visualizations played a crucial role in our comparative study, offering valuable perspectives on the divergences and similarities between the two models, enriching our understanding of their respective performances.

vi) Evaluating the Model:

In the final phase, the model's efficacy was evaluated through object detection on images, employing the OpenCV framework. The output of bounding boxes encompassed information on the detected classes, with probabilities assigned to each class. The results included the likelihood of an image belonging to a specific class, expressed as a probability score. This comprehensive evaluation process provided a practical assessment of the model's performance in real-world scenarios, aiding in the validation and refinement of the object detection capabilities across diverse visual inputs.

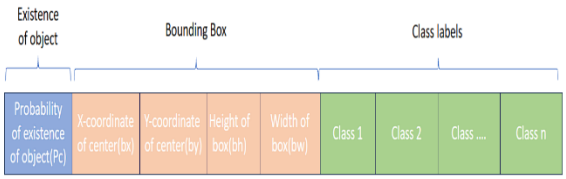

Fig. 5. Output of Yolo Model

V. RESULTS AND DISCUSSIONS

Our comparative study between YOLOv3 and YOLOv5 aims to discern the superior version. Employing diverse preprocessing techniques, we tailored the data to optimize model compatibility. Our focus on nuanced data refinement provides valuable insights into each model's efficacy. Through this investigation, we aim to shed light on the preferable version, offering guidance based on performance within the context of our specifically tailored preprocessing methodologies.

Training YOLOv3 demanded a substantial time investment, especially when compared to YOLOv5. This was exacerbated by the necessity to augment the number of epochs for YOLOv3, as stipulated by the authors who recommended setting the number of epochs to 2000 times the number of classes.

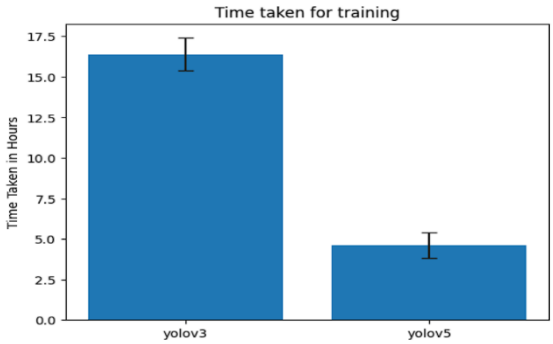

Fig. 6. Time taken to Tarin

The F1 score, calculated at various confidence levels for the bounding boxes of each given class, was determined to be as follows:

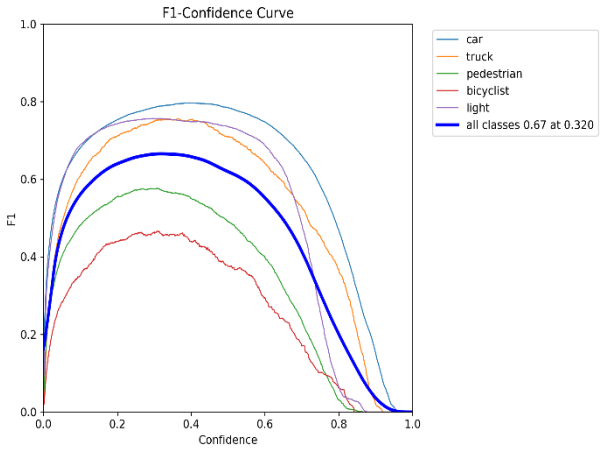

Fig. 7. F1 confidence curve

To comprehensively examine the alignment between ground truth and predicted bounding boxes, an in-depth analysis was conducted using a confusion matrix. This matrix provides a detailed breakdown of the original and predicted class associations, shedding light on the accuracy and discrepancies in the classification process. The results of this analysis are presented below

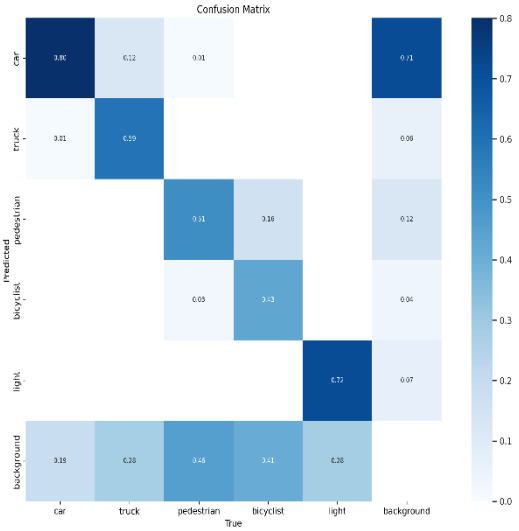

Fig. 8. F1 Confusion Matrix

Precision and recall stand as essential benchmarks in the evaluation of classification models, providing insights into the model's ability to make accurate positive predictions and capture all relevant instances of a class, respectively. The precision-recall values, detailed across varying confidence thresholds, offer a nuanced perspective on the model's performance, allowing for a comprehensive analysis of its precision and recall trade-offs at different levels of confidence.

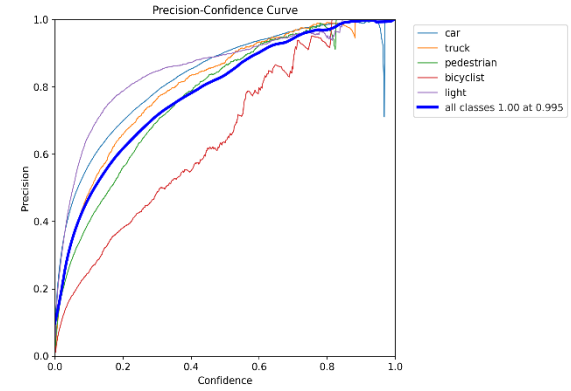

Fig. 9. Precision confidence curve

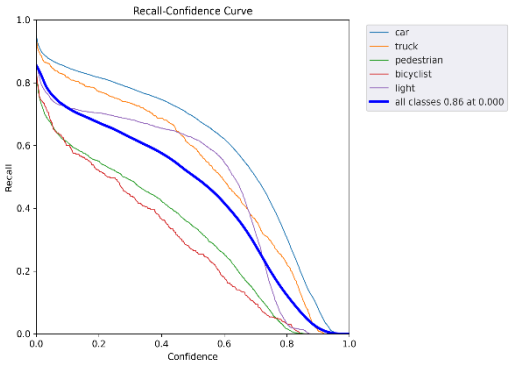

Fig. 10. Recall confidence curve

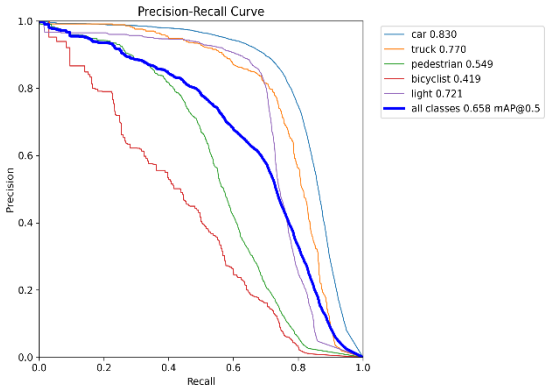

Fig. 11. Precision Recall confidence curve

We subjected the models to rigorous evaluation using the Mean Average Precision (mAP) metric, a widely accepted and standard approach for assessing the performance of models in the realm of object detection. This metric is recognized for its ability to comprehensively gauge the precision and recall of a model, providing valuable insights into the overall effectiveness of the object detection capabilities.

The Mean Average Precision (mAP) values for the YOLOv5 model are as follows.

Class	Instance	P	R	mAP50
all	22603	0.74	0.6	0.65
car	15595	0.81	0.77	0.83
truck	1017	0.78	0.72	0.77
pedestrian	2518	0.72	0.46	0.54
bicyclist	370	0.5	0.41	0.41
light	3102	0.85	0.67	0.72

Table. 1. Average Precision vs Recall values

After conducting a thorough comparison between the two models, it was observed that YOLOv5 consistently outperformed the other model in a majority of scenarios.

VI. CONCLUSION

In this comparative study between YOLOv3 and YOLOv5 for object detection, a meticulous evaluation of both models revealed noteworthy distinctions in performance. YOLOv5 consistently demonstrated a marginally superior performance across a spectrum of evaluation metrics. The model exhibited a nuanced edge, particularly noticeable in precision, recall, and Mean Average Precision (mAP) values.

While YOLOv3 remains a robust and well-established model for object detection tasks, YOLOv5's architecture enhancements and optimization strategies contributed to its incremental advantage. The improvements in detection accuracy, coupled with streamlined training processes, underscore YOLOv5 as a viable evolution in the YOLO series.

The nuanced outperformance of YOLOv5, though subtle, has practical implications for applications demanding precise and efficient object detection. The findings of this research advocate for considering YOLOv5 in scenarios where subtle improvements in detection performance can have a tangible impact. However, it's essential to weigh the computational costs associated with these advancements in performance when determining the most suitable model for specific use cases.

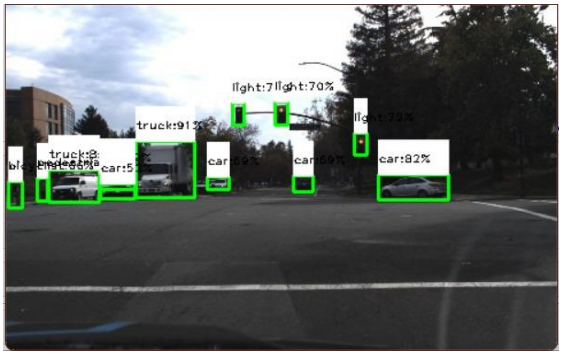

Fig. 12. YOLOv5 Prediction on training data

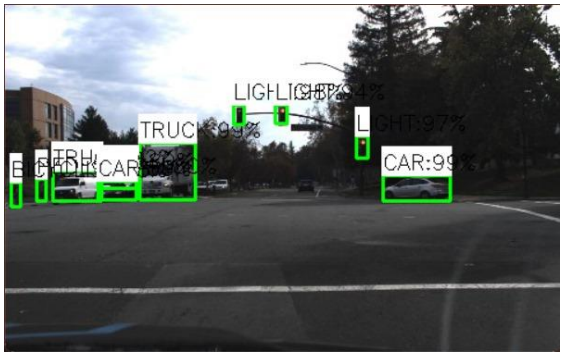

Fig. 13.YOLOv3 Prediction on training data

Fig. 14. YOLOv5 Prediction on unseen data

Fig. 15. YOLOv3 Prediction on unseen data

Overall, this study highlights the iterative advancements within the YOLO architecture, with YOLOv5 showcasing a modest yet meaningful stride forward in the realm of object detection.

REFERENCE

[1] https://www.frontiersin.org/articles/10.3389/fnbot.2023.1058723/full

[2] https://www.frontiersin.org/articles/10.3389/fnbot.2023.1058723/full

[3] https://ieeexplore.ieee.org/document/9445365

[4] https://github.com/JananiSBabu/Autonomous_Driving_Car_Detection_YOLO_Tensorflow

[5] https://www.youtube.com/watch?v=mRhQmRm_egc

[6] https://www.youtube.com/watch?v=hTCmL3S4Obw

[7] https://www.kaggle.com/datasets/alincijov/self-driving-cars/code

[8] https://medium.com/geekculture/self-driving-car-with-yolov5-and-roboflow-a5a2bf109c6

[9] https://iq.opengenus.org/yolov5/

[10] https://iq.opengenus.org/architecture-of-yolov3/