Road Object Detection

ABU ZAHID BIN AZIZ* and MOKSHAGNA SAI TEJA KARANAM*, School of Computing, University of Utah, USA

Road object detection is an important component of autonomous driving systems, adding significantly to total navigation efficiency and road safety. Deep learning approaches have made significant advances in this subject in recent years, with two major architectures being single-stage (YOLO) and multi-stage (Faster-RCNN). We give a full comparison of these two cutting-edge techniques in the context of road object identification in this work. To ensure a fair and reliable comparison, we first present a comprehensive review of the fundamental principles and functioning mechanisms of both the YOLOv5 and Faster-RCNN models. We then assess their performance on a large-scale benchmark dataset with a variety of situations and levels of complexity. Our research shows that the single-stage YOLOv5 model has faster processing speeds and reduced computational requirements, making it appropriate for real-time deployment on embedded systems. The multi-stage Faster-RCNN model, on the other hand, has lower detection accuracy and higher false positive rates due to its more complicated architecture and region proposal methods. The codes for this project is publicly available here: https://github.com/mahimoksha/ECE6960-Deep_Learn_Img_Analysis_project

Additional Key Words and Phrases: YOLO V5, Faster-RCNN, style, Object Detection, Comparision, Mean Average Precision(MAP)

1 INTRODUCTION

Road object detection is a fundamental task in the development of advanced driver assistance systems (ADAS) and autonomous vehicles (AVs), playing a crucial role in ensuring safety, navigation efficiency, and real-time decision-making. With the advent of deep learning techniques, significant progress has been made in the field of computer vision, leading to the emergence of various object detection models that have been widely adopted for road object detection. Among these models, single-stage architectures like You Only Look Once (YOLO) and multi-stage architectures such as Faster Region-based Convolutional Neural Networks (Faster-RCNN) have gained considerable attention due to their impressive performance on benchmark datasets. The primary aim of this paper is to provide a comprehensive comparison between these two state-of-the-art approaches, assessing their strengths and weaknesses in the context of road object detection and offering guidelines for selecting the most suitable model for different applications.

If we briefly delve into the key differences between single-stage and multi-stage object detection models, we notice that single-stage models, such as YOLO, streamline the detection process by directly predicting object classes and bounding box coordinates in a single pass through the neural network. This results in reduced computational complexity and faster processing times, making them well-suited for real-time applications. On the other hand, multi-stage models, like Faster-RCNN, employ a hierarchical approach to object detection, generating region proposals and subsequently refining them through additional network layers. This multi-step process has a cost of increased computational requirements and longer processing times. Given these fundamental differences, it is essential to thoroughly evaluate and compare the performance of YOLO and Faster-RCNN in various scenarios and environments to gain a better understanding of their suitability for road object detection in autonomous driving systems.

*Both authors contributed equally to this project.

Authors' address: Abu Zahid Bin Aziz, u1410993@utah.edu; Mokshagna Sai Teja Karanam, u1418261@umail.utah.edu, School of Computing, University of Utah, Salt Lake City, Utah, USA, 84112.

Throughout the evolution of the YOLO architecture, several advancements and optimizations have been introduced, from the initial version (v1) to the version we used in this work, YOLOv5. In YOLOv1, the pioneering concept of single-stage object detection was introduced, which performed detection by dividing the input image into a grid and assigning each cell the task of predicting a fixed number of bounding boxes and class probabilities. However, YOLOv1 suffered from limited detection accuracy, particularly for smaller objects, and the inability to capture contextual information. To address these issues, YOLOv2 introduced anchor boxes and batch normalization, improving the model's accuracy and training stability. YOLOv3 further enhanced the architecture by incorporating a multi-scale feature pyramid and adopting the Darknet-53 backbone, which significantly increased the model's ability to detect objects across various scales and resolutions. YOLOv4, a subsequent iteration, introduced a series of optimizations that aimed at striking a balance between accuracy and speed. This version combined the best features from earlier versions and integrated additional techniques, such as the Bag of Freebies (BoF) and Bag of Specials (BoS), to improve the model's overall performance. Moreover, YOLOv4 employed the CSPDarknet53 as its backbone, which further enhanced the detection capabilities while maintaining real-time processing speed.

Finally, YOLOv5, the version used in this paper, builds upon the advancements of its predecessors by incorporating architectural improvements, such as the introduction of the Focus layer, which merges information from adjacent pixels to reduce the initial feature map size. Additionally, YOLOv5 employs an updated backbone, the CSPNet, and benefits from the integration of LeakyReLU and Mosaic data augmentation. These enhancements collectively contribute to the superior performance of YOLOv5 in terms of detection accuracy and processing speed, making it an ideal candidate for the comparison with Faster-RCNN in this study.

2 METHODS

2.1 YOLO

YOLOv5 [3], the fifth version of the YOLO (You Only Look Once) object detection model, builds upon the strengths of its predecessors while introducing several architectural improvements and optimizations to achieve better performance in terms of detection accuracy and processing speed.

- **Backbone**: YOLOv5 employs a CSPNet-based backbone, which stands for Cross-Stage Hierarchical Network. The CSPNet backbone enhances the gradient flow and network scalability, allowing for efficient feature extraction and improved learning capability.
- **Neck**: The neck of the YOLOv5 architecture consists of PANet (Path Aggregation Network) and BiFPN (Bidirectional Feature Pyramid Network). PANet enables efficient feature fusion across various scales, while BiFPN allows for bidirectional cross-scale connections, further improving the model's capability to detect objects of different sizes and at varying resolutions.
- **Head**: The head of the YOLOv5 model is responsible for predicting bounding box coordinates, objectness scores, and class probabilities. It utilizes a combination of convolutional layers and anchor boxes to generate predictions at three different scales. These predictions are then decoded to produce the final output in the form of bounding boxes and class labels for each detected object.
- **Focus Layer**: YOLOv5 introduces the Focus layer at the beginning of the network. This layer combines information from adjacent pixels in the input image, reducing the size of the initial feature map while retaining crucial spatial information. The Focus layer contributes to improved efficiency and model performance.

- **Activation Functions**: YOLOv5 uses the LeakyReLU activation function in its architecture, which helps in mitigating the vanishing gradient problem, leading to better training stability and faster convergence.
- **Data Augmentation**: YOLOv5 incorporates the Mosaic data augmentation technique, which combines four training images into a single mosaic image. This technique exposes the model to a more diverse range of object scales, orientations, and lighting conditions during training, ultimately improving its generalization capabilities.
- **Loss Function**: YOLOv5 utilizes the CIoU (Complete Intersection over Union) loss function, which takes into account the geometric and aspect ratio differences between the predicted and ground truth bounding boxes, leading to improved localization performance.

These architectural improvements and optimizations collectively contribute to the superior performance of YOLOv5 in terms of object detection accuracy and processing speed, making it a competitive choice for various object detection tasks, including road object detection in autonomous driving systems.

2.2 Faster RCNN

Faster R-CNN [1] is a state-of-the-art deep learning approach for object detection, which combines Region Proposal Networks and object detection into a single end-to-end trainable network. In this paper, we present a detailed analysis of the Faster R-CNN approach for the proposed task Road Object Detection.

The architecture of the Faster R-CNN model has its backbone network, region proposal network, and object detection network. We then discuss the training process for Faster R-CNN, i.e loss functions and learning rate scheduler. The Region Proposal Network (RPN) is a fully convolutional neural network that is capable of predicting regions of interest (RoIs) or object proposals within an image. Upon receiving an image as input, the RPN generates a set of object proposals, each represented by a bounding box and a corresponding score indicating the probability of the proposal containing an object of interest.

Subsequently, the object detection network utilizes these proposals as input to perform classification and refinement, ultimately resulting in the final detection results. The utilization of RPN allows the model to share computations between the region proposal and object detection stages, thereby enhancing the efficiency and speed of the model compared to previous approaches.

We have done some ablation studies by experimenting with several architectures for backbone networks the best of them are as follows:

2.2.1 ResNet-50. ResNet-50 [4] is a widely used deep convolutional neural network architecture that has shown exceptional performance in various computer vision tasks, including image classification and object detection. As the backbone network, ResNet-50 serves as the feature extractor for Faster R-CNN.

It takes an input image and processes it through a series of convolutional layers, enabling the extraction of high-level features. These features are essential for accurately detecting and localizing objects within the image. It allows the network to effectively learn complex representations.

To incorporate ResNet-50 into Faster R-CNN, the model is initialized with pre-trained weights which are obtained from training ResNet-50 on the ImageNet dataset. This initialization process allows the network to leverage the knowledge gained from a large-scale image classification task.

Furthermore, the model is fine-tuned using the existing dataset specific to object detection. The outcomes yielded by this approach notably outperform other backbone frameworks.

The results on Held out data which is trained on ResNet-50 are seen in Results section.

2.2.2 MobileNet-v2. I have done another ablation study where changing the backbone architecture to other pretrained model i.e Mobile Net-v2. MobileNetV2 [2] is a lightweight neural network architecture that has been specifically designed embedded vision applications.

As the backbone network similar to ResNet-50, MobileNetV2 serves as the feature extractor. As we are using pretrained model, the image will pass through the network with fixed weights which significantly reduce the computational complexity and model size compared to traditional convolutional layers and finetune the model in similar fashion to above study.

By using pretrained weights it enables to leverage knowledge learned to generalize well for object detection tasks.

As the dataset is very high , this lightweighted architecture MobileNetv2 will be useful for comparision as it can work well on limited computational resources.

2.2.3 VGG-19. The mode VGG-19 [6] is composed of 19 layers, which has a series of convolutional layers followed by subsequent max pooling operations. With small receptive fields compared to other pretrained ImageNet Models, these layers effectively capture intricate details from the input image.

VGG-19 is a relatively large and computationally expensive network compared to MobileNetv2. It can provide accurate feature representations, it is little bit slower in inference compared to other pretrained backbones.

On Overall comparision to all the different backbone architectures in Faster-RCNN,

As ResNet 50 introduces skip connections which will keep feeding input information and control the weights. The feature vector produces a fixed-size representation regardless of the input's spatial dimensions. This simplifies the model, reduces the number of parameters, and improves computational efficiency.

MobileNetV2 is a lightweight nature makes it easier to adapt and fine-tune for specific tasks.

VGG19 is a slighlt complex model is inference which takes some extra time to give results.

All models are failing in certain scenarios that is discussed in results sections.

3 EXPERIMENTS

3.1 Dataset

The images in this Dataset are the frames at the 10th second in the traffic videos. The split of train, validation, and test sets are the same with the whole video set. They are used for object detection, derivable area, lane marking. There are a total of 10 classes (objects) which are pedestrian, rider, car, truck, bus, train, motorcycle, bicycle, traffic light, traffic sign. There a total of 31105 train images with labels, 6496 validation images with labels. There is a held out test set which we have validated with the best model. The results are displayed below.

3.2 Learning Protocol

3.2.1 Faster-RCNN. For Faster-RCNN We use the GPU NVIDIA RTX A5000 and used Mean Average Precision metric from the torchmetrics library which predicted boxes and classes have to be in Pascal VOC format (xmin-top left, ymin-top left, xmax-bottom right, ymax-bottom right). used Adam optimizer and a learning rate of 0.001 for every study with a StepLR learning rate scheduler. every model is trained in 25 epochs to keep the fair comparision. The internal filters of for all the different studies architecture are 64, 128, 512 and the aspect ratio is 0.5, 1.0, 2.0. the region of interest pooler has the output size of 5 and sampling ratio as 2. From the graphs we understood that after certain epochs the MAP is becoming constant in most of the studies.

Table 1. Performance comparison between the models used in this project

Models	MAP	Inference Time (seconds)
Faster-RCNN-VGG	0.1005	0.075699
Faster-RCNN-ResNet	0.1756	0.07245388
Faster-RCNN-Mobilenetv2	0.1445	0.051901
YOLOv5	0.365	0.007

3.2.2 YOLOv5. In our experiments, we trained the YOLOv5 model using the Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of 0.01, momentum of 0.937, and weight decay of 5e-4. The model's loss function comprised a combination of Generalized Intersection over Union (GIoU) loss with a gain of 0.05, classification (cls) loss with a gain of 0.58, and objectness (obj) loss with a gain of 1.0. The cls and obj losses employed Binary Cross-Entropy (BCE) with positive weights of 1.0. During training, we utilized an Intersection over Union (IoU) threshold of 0.20 and an anchor-multiple threshold of 4.0. The focal loss gamma was set to 0.0.

For data augmentation, we applied a series of transformations, including HSV color space adjustments with hue, saturation, and value (brightness) factors of 0.014, 0.68, and 0.36, respectively. Additionally, we incorporated image rotation, translation, scaling, and shearing with degrees set to 0.0, translation factor at 0.0, scaling gain of 0.5, and shearing degrees at 0.0. These augmentations contributed to the model's robustness by exposing it to a diverse range of object appearances and variations during the training process.

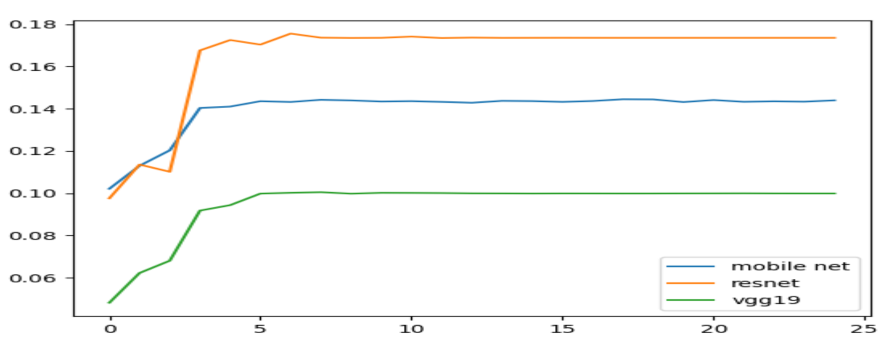
The YOLOv5 model was trained for 200 epochs with a batch size of 32 and an input image size of 640x640 pixels. This training protocol allowed the model to learn and adapt to various road object detection scenarios, ultimately yielding a high-performance detection system suitable for comparison with the Faster-RCNN model.

3.3 Results

3.3.1 Faster RCNN. The MobileNet model required a training time of 14.5 hours, and during inference, it took an average of 0.051901 seconds to process an image.

In contrast, the ResNet model took 33 hours and 55 minutes to train the complete dataset. For inference, it took an average of 0.07245388 seconds to obtain results for an image.

Similarly, the VGG architecture took 30 hours and 55 minutes for training the entire dataset. During inference, it took an average of 0.075699 seconds to obtain results for an image.



Mean Average Precision across Faster RCNN with different backbone architectures

From the above plot we can see that Resnet has significant difference compared to other backbones. we can observe that after certain number of epochs the metric becomes constant with very less variations maybe this is because the model is stuck in a local minima or a saddle point.

The following are the results with the best models on the test dataset:

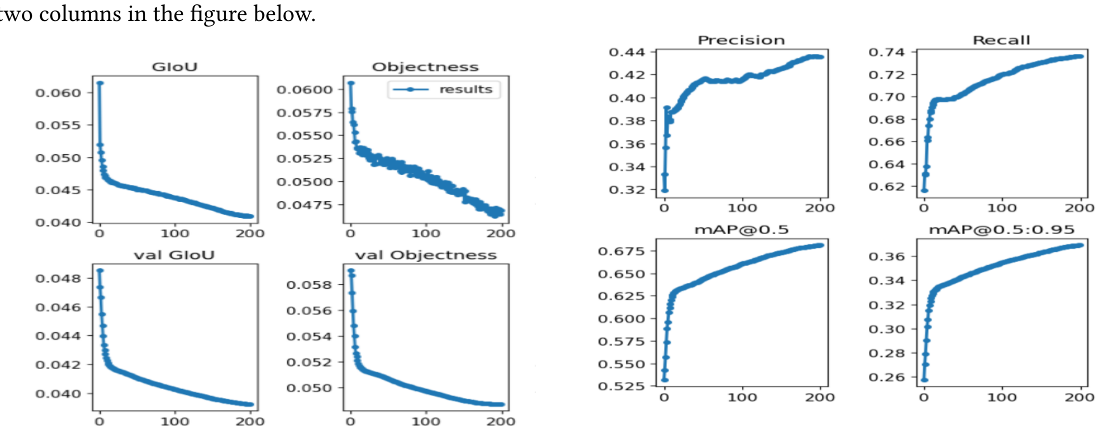
Resnet-50

VGG-19

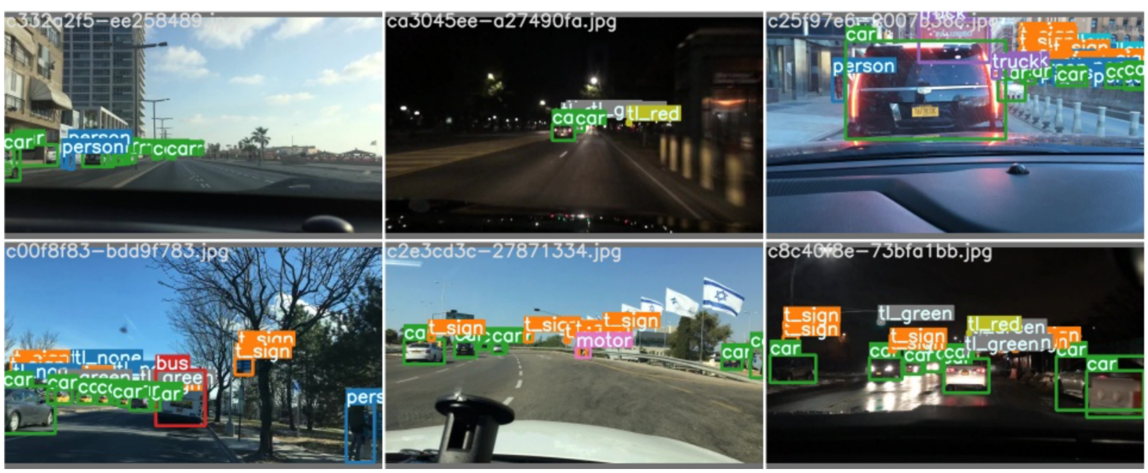
MobileNet-V2

From these results we can see that there are some cases where VGG is not able to find the car infront of the camera and resnet is not able to find the pedestrian. where as the mobile net is able to classify both as expected but not able to find smaller objects.

3.3.2 YOLO. The YOLOv5 model training took around 20 hours where 70000 images were used in training, 10000 in validation and 20000 in testing. The GIoU and objections loss is shown in left two columns in the figure below. We have plotted the precision, recall and mean average precision metric on the validation data which is shown in the rightmost two columns in the figure below.



Here are some sample predictions of the trained YOLOv5 model:



We can see that the performance of YOLOv5 models are much better in terms of accuracy and inference speed than the Faster-RCNN models.

4 CONCLUSION

From this project we learned how to tackle object detection problem , the preprocessing methods, using state of the art object detection problems like YOLO, Faster-RCNN with different bottlenecks. There is still room for improvement interms of accuracy for this task. In future this work can be extended and debugged as follows:

- We conducted experiments on pretrained backbone networks for Faster-RCNN. It is worth considering training the model from scratch, without relying on default weights and also can train on custom architectures which gives better bottle neck, as this approach may yield improved feature extractors for the object detection problem.
- Other metrics, such as mean average recall and others, can be utilized to assess the performance of the aforementioned models and determine areas where the model is underperforming at a granular level, specifically in terms of its weights.
- It might be good to identify the Average precision on smaller objects and larger objects separately as introduced in COCO dataset [5] Object detection model.

In this project, we conducted a comparative analysis of YOLO and Faster-RCNN models, employing various ablation studies. Overall, YOLO outperformed Faster-RCNN significantly in this scenario, demonstrating superior capability in identifying smaller objects. On the other hand, Faster-RCNN exhibited higher false positive rates. Given that YOLOv5 was developed after Faster-RCNN, it incorporates modular approaches and optimizations that could potentially enhanced object classification and detection.

REFERENCES

[1] Xinlei Chen and Abhinav Gupta. 2017. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138* (2017).

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[3] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, Yonghye Kwon, Kalen Michael, Jiacong Fang, Zeng Yifu, Colin Wong, Diego Montes, et al. 2022. ultralytics/yolov5: v7. 0-YOLOv5 SOTA Realtime Instance Segmentation. *Zenodo* (2022).

[4] Brett Koonce and Brett Koonce. 2021. ResNet 50. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization* (2021), 63–72.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 740–755.

[6] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).