UNIVERSITY OF SYDNEY

Department of Electrical and Computer Engineering

ELEC5308 Intelligent Information Engineering Practice

Assignment No.3

Title: A Simple Self-driving System

Group: 12

Zheng Feng
Zhuo Shi
Chenghao Liu
Xinyi He

Table of Content

1. Background

Since the mid-20th century, artificial intelligence has rapidly risen from a preliminary concept to the centre of modern technology. Innovative ideas from pioneers such as Alan Turing, such as the Turing machine, laid the cornerstone for its development [1]. Subsequently, AI has spawned a variety of applications, especially autonomous driving technology. This is not only about the real-time road response of vehicles but also about the vision of building an intelligent transport network. This network aims to optimize traffic, enhance safety, and reduce accidents. With the evolution of inter-vehicle communication technologies, we can foresee a smarter and more efficient transport future.

Although self-driving cars were mentioned decades ago, they remained mainly in the theoretical stage. In recent years, thanks to advances in artificial intelligence, sensor technologies such as laser radar (LiDAR) and cameras, and the increased computational power of graphics processors (GPUs), the concept has gradually moved toward practical applications. In their 2019 study, Anderson and Smith[2] brilliantly describe the revolutionary shift from traditional driving modes to the realm of autonomous driving. The automotive industry is undergoing unprecedented change, not just in pursuing the traditional manually driven car but in the transition to a highly automated and intelligent future. Currently, leading companies such as Tesla, Alphabet's Waymo and Uber are not only investing heavily in technology but also outlining new blueprints for the future of mobility [3]. Tesla has already implemented semi-autonomous driving in some of its models [4], while Waymo and other teams such as Cruise and Uber are advancing the development and application of self-driving technology globally, working together to define a new direction for the future of mobility.

In our project, we worked on creating a system dedicated to automatically analyzing road conditions with simple strategies based on traffic rules. This system will not only be able to handle basic driving tasks, such as following traffic light instructions but will also be able to automatically analyse and respond to various situations on the road. To achieve this, we integrated a core component into the system that is responsible for the automatic analysis of real-time road conditions. We chose the YOLO (You Only Look Once) series, specifically YOLOv3, YOLOv4 and YOLOv5[5], as the AI target detectors for this component. We chose YOLO because it excels in both real-time detection speed and accuracy. It is able to quickly identify and locate traffic signs and accurately determine their type, such as the status of a traffic light or the presence of a vehicle ahead. To ensure that our system performs well in real-life scenarios, we use the CARLA simulator, an open-source simulation tool developed for autonomous driving research, which allows us to test and validate the YOLO model[6], as well as the overall performance of our entire automated road analysis system, in a simulated real road environment.

We are in the midst of an epochal change as autonomous driving technology advances rapidly. In this project, we are not only pursuing cutting-edge technology but also focusing on creating an intelligent, regulatory-compliant and ethical autonomous driving system. Our goal is to drive the future of mobility to be safer, more efficient and more sustainable.

2. Problem Formulation

In the field of autonomous driving, there is an urgent need for an efficient and intelligent system in the face of complex real-time traffic scenarios, and our project was born to solve this problem. In Project 2, we aim to build a comprehensive autonomous driving system that is designed to automatically analyse and properly handle all kinds of traffic situations. This system will not only be able to accurately detect traffic signals and signs and make decisions accordingly, such as stopping at a red light or driving on a green light, but it will also be able to recognize and avoid collisions with other vehicles on the road. For example, when the system detects a vehicle in front of it, it will automatically slow down or stop; if a vehicle is on the left, it will avoid turning left.

To address this problem, our project explores the ability of machines to understand and, in particular, how to allow them to simulate and predict human decision-making processes. This involves extensive data collection and analysis, including data on driver behaviour in various traffic scenarios. Through deep learning and neural network techniques, we try to train models to simulate human thinking and decision-making processes, thus enabling machines to make more humane decisions in complex traffic environments[7]. In the research and development of autonomous driving technology, it has become a cutting-edge research topic to ensure that machines can deeply parse and understand human intentions and behavioural patterns. This understanding requires not only a high degree of object recognition but also a deep interpretation of subtle human behaviours and decisions in complex traffic environments. This means that machines must be not only able to recognise pedestrians, vehicles and other objects but also be able to predict their future movements and intentions.

In addition, our system has an excellent response to unexpected events. Whether it's a pedestrian suddenly crossing the road or a vehicle braking in front of you, the system is able to assess the situation and take appropriate risk avoidance measures quickly. It adapts to changing traffic patterns and easily copes with obstacles in various driving environments, ensuring smooth navigation at all times. More importantly, the system significantly reduces human errors, thus greatly improving road safety, and through the analysis of real-time traffic data, further optimizes traffic flow, providing drivers with a more comfortable and safer driving experience.

2.2 Challenges of Task

When using the CARLA simulator for autonomous driving scenario simulation, we faced several major challenges. Firstly, although CARLA provides a wide range of weather and time conditions, truly complex traffic scenario simulations, such as emergency braking, road construction, or temporary road closures, are still more difficult to implement accurately. This means that in some specific scenarios, our system may not be adequately trained. Second, the AI-controlled vehicles and pedestrians in CARLA simulate basic behaviours but are far from the complex behavioural patterns of the real world. For example, a real pedestrian may

suddenly change his/her walking direction due to distraction, or a driver may make unintended driving decisions.The AI models in CARLA may sometimes be too simplified to capture these subtle human behaviours. Further, processing the data output from the CARLA simulator is an equally daunting task. the various sensors generated by CARLA, such as cameras, radar, and LIDAR, generate a large stream of information. To extract meaningful information from this data and make real-time decisions, we need to employ advanced data processing techniques and powerful processors. Any processing bottlenecks may lead to decision-making delays, thus affecting the responsiveness and safety of the autonomous driving system. Finally, compatibility issues cannot be ignored. Considering the various hardware and software configurations, it becomes especially critical to ensure the smooth operation of the autonomous driving system on CARLA. This may involve driver updates, system optimization and specific hardware acceleration techniques to ensure seamless interaction between the simulator and the autopilot system.

The challenges of developing an automated traffic analysis system, in reality are manifold. Firstly, differences in traffic regulations and habits in different regions mean that autonomous driving systems must be able to adapt to a variety of driving environments. In addition, changing road conditions, such as construction, potholes and temporary road signs, require systems to be highly adaptable. Then again, autonomous driving systems may also have deficiencies in understanding certain ambiguous or implied social signals, such as the gaze of pedestrians or the gestures of other drivers, when compared to human drivers. Not to mention, autonomous driving technology must also cope with extreme weather conditions, such as heavy rain, snow or fog, where sensor performance may be significantly reduced. [8]Finally, to ensure safety, autonomous driving systems must undergo millions or even billions of kilometres of real-world road testing, which is both time-consuming and expensive.

3. Solution Design

To address the problem of real-time traffic analysis in autonomous driving, our project designs an integrated system based on the YOLO (You Only Look Once) method. First, we created a driving simulation environment using CARLA. This advanced open-source autonomous driving simulator accurately simulates various real-world driving scenarios, including different weather, lighting, and traffic conditions. In such an environment, we deployed YOLO models trained specifically for traffic elements, such as traffic signals, vehicles, and pedestrians. To ensure optimal performance, we conducted extensive experiments testing multiple versions of the YOLO model, including YOLOv3, YOLOv4,

and YOLOv5, each of which has unique features and optimizations that allow us to choose based on specific application scenarios and requirements.

Based on the output of the YOLO model, our system can parse traffic scenarios in real time and formulate driving strategies accordingly. For example, when a red light is detected, the system automatically instructs the vehicle to stop while allowing it to continue on a green light. In addition to traffic signals, the system can recognise and respond to other vehicles on the road. When the system detects vehicles to the side or in front of it, it analyses the position, speed and expected trajectory of these vehicles in real-time to decide on whether to turn or continue. For example, if a vehicle is approaching quickly on the left, the system will intelligently delay the decision to turn left to ensure safety. In addition, when the system detects a pedestrian suddenly crossing the road in front of it or a sudden roadblock, it will also quickly adjust its speed or change its travelling direction to ensure safe driving.

To address the many challenges encountered during the development phase, we adopted a range of solutions that combined strategy and technology. We ensured that traffic signs and other critical objects could be accurately recognized in complex environments. We focused on enhancing the system's efficiency through in-depth algorithm optimization, fine parameter tuning and efficient operational processes, aiming to reduce the dependency on high-end hardware. During the software construction process, we followed a modular design principle to ensure the robustness and reliability of each part. Each module was subjected to independent verification and rigorous testing. In addition, in order to verify the robustness of the system in real-world applications, we conducted a series of simulation tests to ensure that the system can operate stably in a variety of real-world scenarios.

4. Implementation

4.1 Dataset (Real-time video data)

The "Udacity Self Driving Car Dataset" from Roboflow's public collection offers a comprehensive assortment of annotated images spanning categories such as pedestrians, vehicles, bicycles, and traffic signals. This expansive dataset contains 15,000 images divided across 11 unique categories, with each image having a resolution of 1920x1200. In total, they occupy approximately 31 GB, though a downscaled 512x512 version is also available at a size of around 580 MB. Each image in this collection has been carefully annotated by Roboflow for precise accuracy. However, the dataset exhibits some imbalances; for example, the "car" category contains 64,399 instances, vastly overshadowing the mere 14 instances in the "trafficLight-LeftTurnYellow" category. Designed primarily for object detection in the realm of autonomous vehicles, this dataset aids in the detection and tracking of various elements like pedestrians and traffic signals. It's worth noting that while the dataset offers a multitude of annotated instances, there's a possibility of underrepresentation in some categories, which may challenge extensive learning. Moreover, the presence of duplicate bounding boxes might influence model training. All rights reserved under the MIT license, the dataset is free for academic and research uses. Conclusively, the dataset's rich collection

of real-world video data related to autonomous driving solidifies its relevance for our project's objectives.

4.2 Experiment setup

4.2.1 Carla simulator

In the quest for the perfect autonomous driving system, the correct response to various scenarios is paramount. Our experiments are meticulously crafted in the Carla simulator to evaluate the vehicle's reaction under many situations. From basic stop-start mechanisms to intricate roundabout navigations, we aim to ensure optimal safety and efficiency in response strategies. Herein, we present a detailed breakdown of the numerous experimental setups and the expected vehicle responses for each.

4.2.2 Driving Strategies

1. Stopping Strategy

Experiment Setup:

- Place multiple static obstacles within the simulator.

- Propel the vehicle towards these obstacles at a certain speed.

Response:

- If the vehicle detects an obstacle and the distance to the obstacle is less than a pre-defined safety distance, the vehicle should autonomously stop.

Figure 4.1 Stopping Strategy

2. Evasion Strategy

Experiment Setup:

- Position pedestrians or other moving obstacles within the simulator.

- Drive the vehicle towards these obstacles at a certain speed.

Response:

- Upon detecting a pedestrian or a moving obstacle, the vehicle should decelerate and attempt to change its driving direction to evade collision.

Figure 4.2 Evasion Strategy

3. Traffic Light Adherence Strategy

Experiment Setup:

- Incorporate traffic lights within the simulator.

- Direct the vehicle towards these traffic lights.

Response:

- On detecting a red light, the vehicle should come to a halt.

- On detecting a green light, the vehicle should proceed normally.

Figure 4.3 Traffic Light Adherence Strategy

4. Speed Limit Adherence Strategy

Experiment Setup:

- Station various speed limit signs within the simulator.

- Navigate the vehicle past these signs.

Response:

- On detecting different speed limit signs, the vehicle should adjust its speed to conform to the speed indicated on the sign.

5. Emergency Braking Strategy

Experiment Setup:

- Suddenly introduce obstacles or pedestrians within the simulator.

Response:

- When detecting a sudden appearance of an obstacle or pedestrian, the vehicle should initiate emergency braking.

6. Pedestrian Crosswalk Adherence Strategy

Experiment Setup:

- Implement pedestrian crosswalks within the simulator.

- Introduce pedestrians intending to cross or already crossing.

Response:

- On detecting a pedestrian crosswalk with pedestrians, reduce speed or stop to allow them to cross safely.

7. Dynamic Weather Adaptation Strategy

Experiment Setup:

- Vary the weather conditions within the simulator, such as rain, fog, or snow.

- Drive the vehicle under these varied conditions.

Response:

- Adjust driving behavior based on weather conditions, like reducing speed during heavy rain or increasing the distance to the vehicle in front during fog.

Figure 4.4 Dynamic Weather Adaptation Strategy

4.3 Method 1 - YOLOv3

4.3.1 Data Loading and Preprocessing

The dataset is sourced from Roboflow and is an enhanced version of the original Udacity Self Driving Car Dataset. This dataset contains labeled images for different road entities such as cars, pedestrians, bikers, and different states of traffic lights. It provides 15,000 images, segmented into various classes with differing representations.[9]

To load the data, the Roboflow dataset can be downloaded in multiple formats that fit various deep learning frameworks. Specifically, when using OpenCV's DNN module, one can opt for a format like YOLO that gives annotations in the form of bounding boxes. Two options are available for downloading this dataset: fixed-small and fixed-large. Depending on one's computational capabilities, the appropriate choice can be made. [10]Notably, the larger dataset provides more samples which can lead to an improved model performance. The dataset comes annotated for object detection, offering bounding box coordinates for the different objects in an image. Some of these labels are 'car', 'pedestrian', and 'trafficLight-Red'.

Preprocessing involves several stages. Firstly, the dataset's images have dimensions of 1920x1200, and based on the model's input requirements, they may need resizing. In the given context, images are resized to half of their original size to expedite processing. While not present in the provided code, data augmentation, which includes techniques like rotation, flipping, and cropping, can be valuable. This augments the dataset size and enhances the model's robustness to various transformations. Each image is normalized before being fed into the model by multiplying every pixel by a scaling factor of 0.00392. This process ensures that pixel values are within a suitable range for neural network processing. Blob formation is facilitated using OpenCV, converting the image into a blob through blobFromImage. This blob is the neural network's input, resized to 320x320 pixels, with swapped RGB channels, ensuring no cropping takes place.

Post-processing consists of drawing bounding boxes and labels on the detected objects once the model's predictions are received. The function named "draw_labels" aids in drawing these boxes and appending the corresponding labels. To guarantee that each object is detected only once and to remove duplicate boxes, Non-Max Suppression (NMS) is utilized, facilitated by OpenCV's NMSBoxes function.

In conclusion, the procedures of data loading and preprocessing are vital for prepping the Udacity Self Driving Car Dataset for object detection tasks. When the data is correctly loaded and preprocessed and paired with a proficiently trained model, it ensures dependable and accurate object detection in a variety of driving situations.

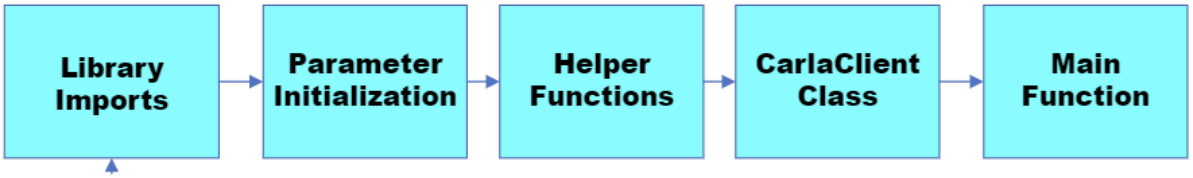


Figure 4.5 Code logic

4.3.2 Network Structure

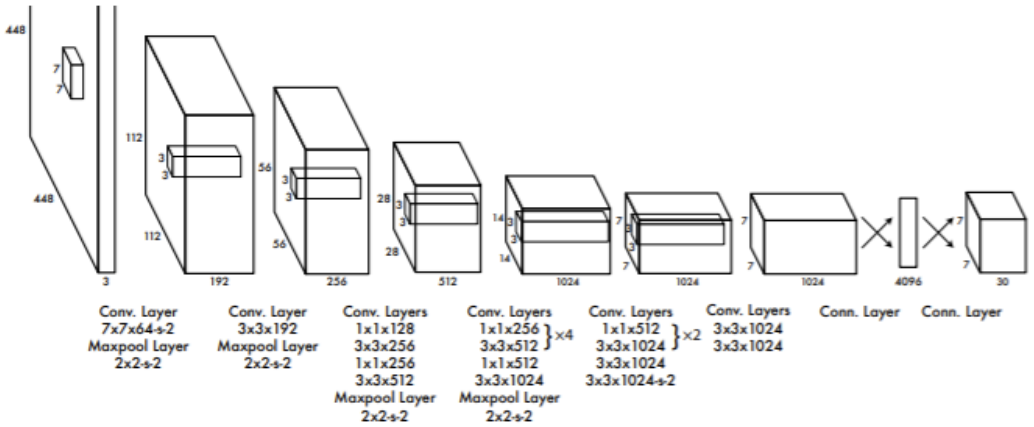
YOLOv3 Network Structure

Input Layer: This is sized at 320x320x3, which aligns with your preprocessing step.

Darknet-53 Backbone: This component is made up of 53 convolutional layers, focusing on extracting features from the input image.

Three Different Scales: YOLOv3 predicts bounding boxes at three distinct scales. Each scale predicts multiple boxes and class probabilities. Anchor boxes, pivotal for these predictions, are employed at every scale. For the Udacity dataset, there's a possibility that you'd need to recalibrate these anchor boxes to match the data distribution.

Output: Each cell within the feature map predicts multiple bounding boxes, detailing coordinates like x, y, width, height, and objectness. Additionally, class probabilities for every class (e.g., 'car', 'pedestrian', 'trafficLight-Red') are predicted. The total number of these predictions is influenced by the number of anchor boxes and classes.



Source: "You Only Look Once: Unified, Real-Time Object Detection" paper

Figure Yolov3 architecture diagram

4.4 Method 2 - YOLOv4

In the case of the YOLOv4 Network Structure tailored for the same dataset:

Input Layer: Again, the size is 320x320x3.

CSPDarknet53 Backbone: This is an advanced version of the Darknet-53, integrating Cross Stage Hierarchical (CSP) connections. These connections bolster gradient flow and improve feature reusability.

PANet & SAM block: The PANet aids in up-sampling and blending features from different scales. The SAM block (or Spatial Attention Module) concentrates on providing attention to spatial regions enriched with significant features.

Three Different Scales: Echoing YOLOv3, YOLOv4 also predicts boxes over three scales, leveraging anchor boxes at each level.

Output: Parallel to YOLOv3, every cell in the feature map under YOLOv4 predicts multiple bounding boxes alongside class probabilities.

Miscellaneous Enhancements: The model incorporates the Mish activation function, CIOU loss for superior bounding box prediction, and assimilates the DropBlock regularization technique.

4.4.1 Analysis of yolov3 and yolov4

YOLOv3:

Advantages:

1. Mature and Stable: As the third iteration in the YOLO series, YOLOv3 has undergone extensive testing and optimization, proving its merit in many real-world applications.

2. Darknet-53 Backbone: With its 53 convolutional layers, it's adept at extracting high-level features from images.

3. Simplicity: Compared to subsequent versions, YOLOv3's architecture is more straightforward, making it easier to understand and modify.

Drawbacks:

1. Speed and Accuracy: YOLOv3 may be slightly behind YOLOv4 in terms of both speed and accuracy.

2. Lacks Some Modern Improvements: YOLOv3 doesn't employ some of the advanced features and optimizations present in YOLOv4.

YOLOv4:

Advantages:

1. Performance Boost: YOLOv4 offers improvements in speed and accuracy, especially on challenging datasets.

2. Technological Advancements: YOLOv4 integrates numerous cutting-edge innovations, including the CSPDarknet53 Backbone, PANet, SAM block, and Mish activation function.

3. Enhanced Feature Extraction: The combination of its CSPDarknet53 backbone with Cross Stage Hierarchical structures aids in more efficient feature extraction.

4. Attention Mechanism: With the SAM module, YOLOv4 can better focus on crucial areas in an image, enhancing detection accuracy.

Drawbacks:

1. Complexity: YOLOv4 has a more intricate architecture due to the integration of multiple innovations. This might make model modifications and optimizations more challenging.

2. Computational Demands: While YOLOv4 shows superior performance, it might demand more computational resources for training and inference, particularly if unoptimized.

4.5 Method 3 - YOLOv5

4.5.1 Network Structure

YOLO Detection System:

The YOLO framework is designed around a unified convolutional neural network, positioning object detection as a regression challenge. This is distinct from the multi-stage approach of the Fast-CNN method. In YOLO's methodology, an image is partitioned into an S x S matrix. Within this matrix, each individual cell is responsible for predicting B bounding boxes, their corresponding confidence scores, and C class likelihoods. These predictions materialize as $S \times S \times (B * 5 + C)$[11] tensors, as visualized in Figure 4.1. Delving into the YOLOv1 architecture, as showcased in Figure 4.1, it's a cascade of convolutional layers. Specifically, it integrates 24 convolutional layers, complemented by 2 dense layers, 1×1 dimension-reducing layers, and 3×3 convolutional layers. For our project's objectives, we gravitated towards the YOLOv5 structure due to its enhanced efficiency and robustness relative to YOLO's original design.

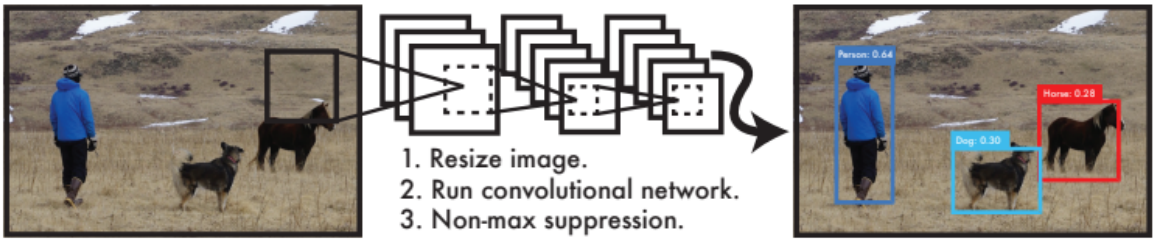


Figure 4.1 The YOLO Detection System

4.5.2 Principles of operation for YOLOv5

YOLOv5 developed by Ultralytics, employs a deep convolutional neural network (CNN) for target detection, utilizing its end-to-end single-stage detection strategy to achieve efficient and accurate target detection.

Specifically, pre-processing is performed first: the image is converted into the format required by the neural network. This is done by resizing the image, normalizing the pixel values and augmenting the mosaic data with new training samples to improve the generalization of the model.

YOLOv5 then uses a deep convolutional neural network (consisting mainly of a convolutional layer and a pooling layer to capture the basic shapes and textures in the image) to perform feature extraction on the preprocessed image in order to recognise and locate objects in the image. In this feature extraction, YOLOv5 also makes use of the Spatial Pyramid Pooling (SPP) layer, and special network structures such as the C3 module and the Focus layer, to further enhance the efficacy of feature extraction.

After feature extraction is complete, YOLOv5 generates bounding box prediction and category classification through the latter part of the deep convolutional neural network (several upsampling and convolutional layers.) YOLOv5 is capable of detecting multiple objects in an image in a single network pass and provides a bounding box, a category label, and a confidence score for each detected object. Based on the detection results, the system analyses the current road conditions and makes decisions.

After extracting the image features through multiple convolutional layers in the network architecture of YOLOv5; and several up-sampling layers, the prediction of generating bounding boxes is completed. The network also optimizes the results of the judgments through post-processing. For example, the CIOU loss function in YOLOv5 provides a more accurate regression of the bounding box location, thus accurately locating the object. Duplicate detections are removed by non-maximal suppression (NMS), preserving the final detection results. As well as removing those detections with low scores based on a confidence threshold.

4.5.3 The structure of YOLOv5 and the application

YOLOv5 has a lot of architectural optimizations, compared with the previous versions (YOLOv3, YOLOv4), PyTorch framework and CSPDarknet53 are adopted as the backbone network in order to improve the gradient utilization, speed up the inference, and improve the accuracy while reducing the model size. In the neck, PANet is introduced to optimize information flow and low-level feature propagation to improve target localization accuracy. It maintains the same HEAD structure as the previous two generations, enables multi-scale prediction and enhances prediction of targets of different sizes.

In the architecture of YOLOv5, the image is first feature extracted by CSPDarknet53 , then feature fusion is performed using PANet, and finally, the detection results are outputted through the YOLO layer. The backbone consists of several functional layers stacked into modules, mainly the CBS and C3 modules. Neck uses the FPN approach to generate different scales of detection kernels by up-sampling. The head part is responsible for generating final bounding box coordinates, object scores and category probabilities. The whole architecture emphasizes speed, accuracy and model size optimization.

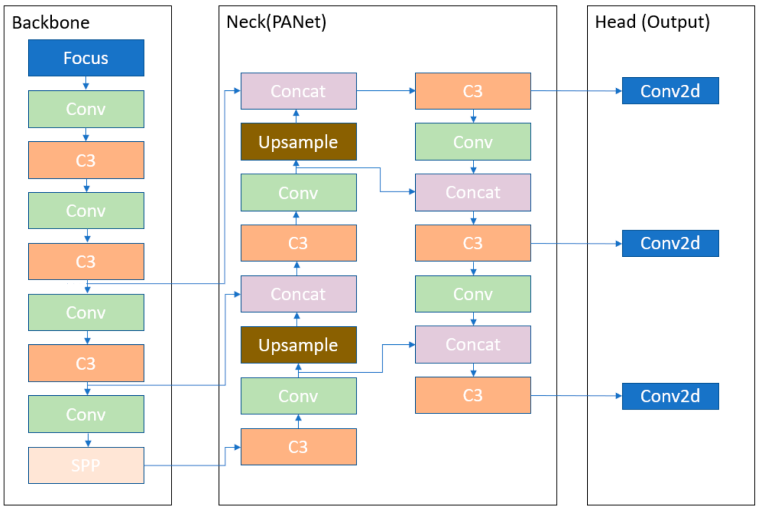


Figure 4.2 Structure for YOLOv5

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

Figure 4.3 Loss function

In the scenario of automatically analysing road conditions, YOLOv5 can be trained to recognise objects such as traffic lights, vehicles and pedestrians. Based on the detection results of YOLOv5, the system can analyse the road conditions and make decisions accordingly. For example, if a red light is detected, the system can send a command to stop the vehicle; if a green light is detected and there is no obstacle ahead, the system can send a command to keep the vehicle moving; it can also calculate the distance between the vehicle and the obstacle ahead, and if the distance is too close, the system can issue a warning or directly manipulate the vehicle to slow down or stop.

4.5.4 Comparison of the structure of YOLOv3, YOLOv4 and YOLOv5

	YOLOv3	YOLOv4	YOLOv5
Neural Network Type	Fully convolution	Fully convolution	Fully convolution
Backbone Feature Extractor	Darknet-53	CSPDarknet53	CSPDarknet53
Loss Function	Binary cross entropy	Binary cross entropy	Binary cross entropy and Logits loss function
Neck	FPN	SSP and PANet	PANet
Head	YOLO layer	YOLO layer	YOLO layer

Figure 4.4 The table for YOLOv3,YOLOv4,YOLOv5

YOLOv5 differs significantly from YOLOv3 and YOLOv4 in design and performance. Firstly, YOLOv5 is built using PyTorch, whereas the previous two are based on Darknet.YOLOv5 uses a Focus structure with CSPdarknet53 as the backbone.The Focus layer is introduced for the first time in YOLOv5.The Focus layer replaces the first three layers in the YOLOv3 algorithm. The advantages of using the Focus layer are a reduction in required CUDA memory, a reduction in the number of layers, and an increase in forward and backpropagation.

Compared to YOLOv3, YOLOv5 introduces the CSPDarknet53 backbone, PANet feature fusion, and the use of a Focus layer instead of the first three layers, resulting in improved efficiency and accuracy. Compared to YOLOv4, YOLOv5 optimises the model structure, making the model smaller but still maintaining similar or even better performance in real-world applications.YOLOv5 also emphasises speed and lightness, making it particularly suitable for deployment on resource-constrained devices. In addition, due to the widespread use of PyTorch, YOLOv5 has relatively stronger ease of use and community support, making it more convenient for model implementation and deployment.

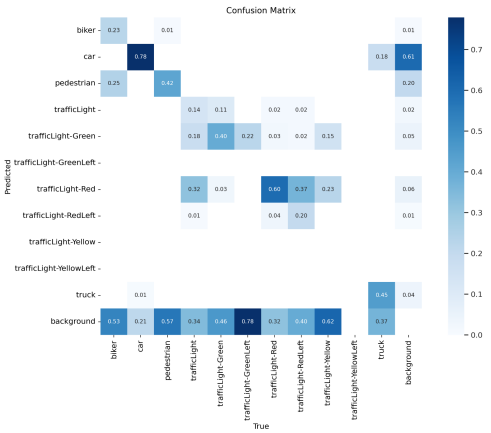
4.5.5 Drawbacks of Yolov5

While YOLOv5 aims to strike a balance between speed and accuracy, in some cases, even if YOLOv5 is slower, its accuracy may not be better than other models optimised specifically to improve accuracy. Simplifying the model for speed can also lead to performance degradation, especially in complex detection scenarios with occlusions or small objects.

Additionally, transitioning to PyTorch, while beneficial to a wider audience, can create challenges in optimising projects from the Darknet framework used in previous versions. Finally, reducing the size of the model, while beneficial for deployment in resource-limited environments, may limit the model's ability to capture complex features, thus affecting its overall performance.

5. Results & Evaluation

5.1 Confusion_matrix



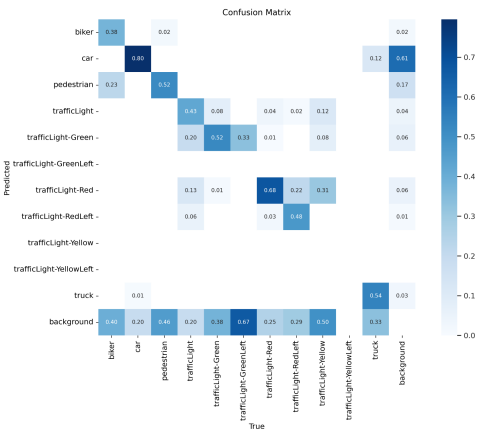


Figure 5.1 Confusion_matrix for YOLOv5 Figure 5.2 Confusion_matrix for YOLOv3

The confusion matrix provides detailed information about how well the YOLOv5 and YOLOv3 models predicted correctly or misclassified on each category. The matrix shows the performance of the models in predicting each traffic category. The rows of the matrix represent the categories predicted by the model, and the columns represent the true categories.

Overall, both YOLOv3 and YOLOv5 perform relatively well in these confusion matrices, but there are subtle differences. YOLOv3 may be more appropriate if higher overall accuracy is required, and YOLOv5 may be considered if a lower false positive rate is desired on some specific categories.

For example, YOLOv3 slightly outperforms YOLOv5 in truck classification; YOLOv3: 0.54 correct classifications; YOLOv5: 0.45 correct classifications.

In the case of (Car) recognition: YOLOv3 continues to predict the category "car" better than YOLOv5, with an accuracy of 0.80 and a misclassification rate of 0.12. YOLOv5 predicts "car" with an accuracy of 0.78, which is slightly lower than that of YOLOv3, and its main misclassification is to predict "car" as "truck", with a misclassification rate of 0.18. 0.18.

Another example is the recognition of Pedestrian: YOLOv3: with an accuracy of 0.52 is higher than that of YOLOv5 (0.42). YOLOv5 misclassified "biker" with 0.25, similar to YOLOv3 (0.23).

For TrafficLight-Green recognition, YOLOv3's accuracy is again higher than YOLOv5's: YOLOv3 has an accuracy of 0.52 while YOLOv5 has an accuracy of 0.40. However, YOLOv5's misclassification rate is 0.22, which is less than YOLOv3's (0.33).

In summary, both YOLOv3 and YOLOv5 perform relatively well on these confusion matrices, but YOLOv3 slightly outperforms YOLOv5 in terms of accuracy on most of the categories, especially on the 'car' and 'pedestrian' categories. However, YOLOv5 performs slightly better in reducing false positives.

5.2 F1-confidence

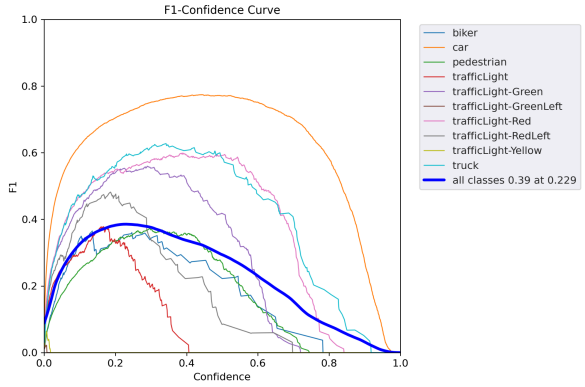


Figure 5.3 F1-confidence curve for YOLOv5

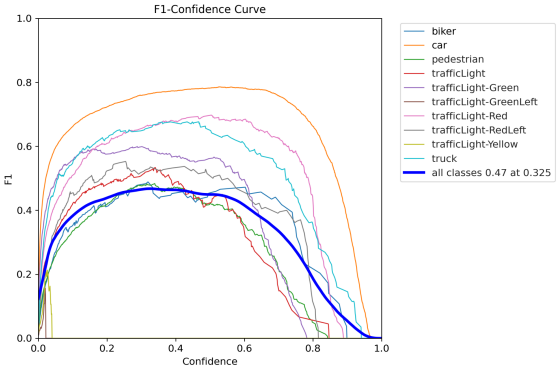


Figure 5.4 F1-confidence curve for YOLOv3

Above is the "F1-Confidence Curve" chart, which shows the F1 scores of several traffic-related categories at different confidence levels. the F1 score is the reconciled average of precision and recall, which provides a unified metric to measure the performance of the models.

Firstly, from the two "F1-Confidence Curve" charts, YOLOv3 slightly outperforms YOLOv5 in overall performance, but yolov5 performs better in specific confidence intervals and in certain categories.

Peak confidence: In YOLOv3, the average F1 score for all categories reaches a maximum of 0.47 at a confidence level of 0.325. In YOLOv5, the F1 score for all categories is 0.39 at a confidence level of 0.229, which means that the average performance of YOLOv3 is slightly better than that of YOLOv5 at this confidence level.

Comparison of individual categories: In the "car" category, for example, YOLOv3's F1 score curve is relatively smooth in the medium-high confidence range, while YOLOv5's F1 score curve reaches a peak in the medium confidence range and then starts to decline. This may indicate that YOLOv5 performs better in the "car" category in some confidence intervals, but decreases in other confidence intervals. In the biker category: YOLOv5's curve is higher than YOLOv3's in most of the confidence intervals, especially in the low and medium confidence ranges, suggesting that YOLOv5 outperforms YOLOv3 in the category "biker".

Volatility: From the graphs, we can also observe that YOLOv5's F1 scores for some categories (e.g., "trafficLight" and its subcategories) fluctuate a lot in the low- to medium-confidence range, whereas YOLOv3's curves are relatively smoother. This may indicate that YOLOv3 is more stable than YOLOv5 in these categories.

Summary: Taking the above analyses together, we can conclude that YOLOv3 may slightly outperform YOLOv5 in some aspects (e.g., average performance and stability), but YOLOv5 may be superior in specific confidence intervals and in certain categories (biker", "pedestrian).

5.3Precision-confidence

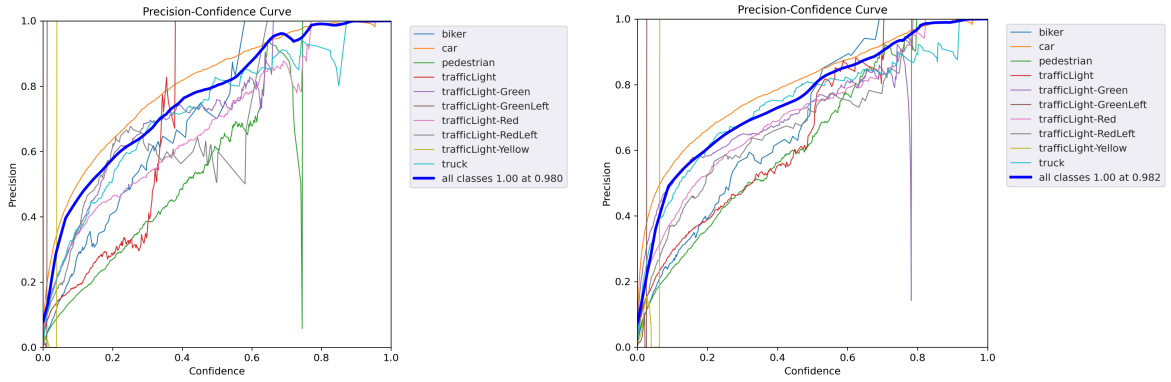


Figure 5.5Precision-confidence curve for YOLOv5 & YOLOv3

The "Precision-Confidence Curve" graph above is used to measure the accuracy of multiple traffic-related categories at different levels of confidence. Precision describes the proportion of all detected targets that are correctly detected by the model.

Overall, if high accuracy is required at low confidence, YoloV3 may be more appropriate. If overall accuracy and stability are required, YOLOv5 may be a better choice.

For example, the Yolov3 is more accurate in some categories in the low CONFIDENCE interval. Its PRECISION is higher than YoloV5 for most categories. This means that YoloV3 may be more accurate when the model is less confident in its predictions. This may be more appropriate for application scenarios where one wants to minimise false positives, even at the expense of some recall.

YOLOv5 is typically more accurate in the medium to high CONFIDENCE interval and is more stable overall. This is more appropriate for scenarios that require high accuracy and high recall. For example, in the medium to high CONFIDENCE interval, the PRECISION of most categories is higher or close to YoloV3, especially in the range of 0.6 to 0.9. This means that YoloV5 is likely to have higher accuracy when the model is more confident in its predictions.

YOLOv5 slightly outperforms YOLOv3 in terms of overall performance.In different situations, YOLOv3 may be considered if high detection accuracy is needed even at low confidence levels, while YOLOv5 may be more appropriate if detection accuracy high confidence levels is more of a concern.

5.4Precision-recall

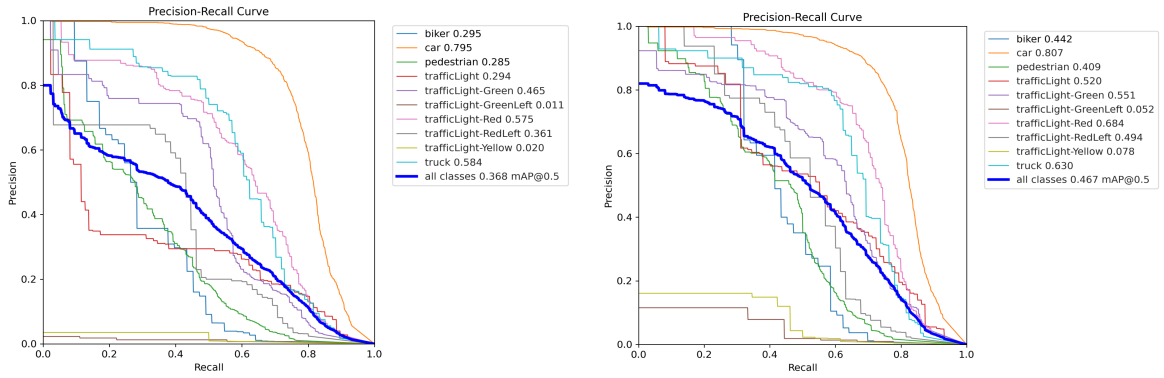


Figure 5.6Precision-recall curve for YOLOv5 & YOLOv3

This figure shows a graph called the Precision-Recall Curve, which measures the precision of multiple traffic-related categories at different recall rates. The precision rate describes the proportion of samples predicted to be in the positive category that are actually in the positive category.

YOLOv3 performs slightly better than YOLOv5 overall at this threshold, but YOLOv5 performs better in some specific categories (e.g., red lights).

Regarding category comparisons:(1) car: YOLOv3 has a slightly higher Precision than YOLOv5, at 0.807 vs. 0.795.(2) Pedestrian: YOLOv3 also performs slightly better than YOLOv5, at 0.409 and 0.285, respectively.(3) Traffic lights: for different traffic light states (e.g., green light, red light, etc.), both models perform differently. For example, YOLOv5 performs better in detecting red lights (0.575 vs. 0.684), but YOLOv3 has higher Precision in detecting green lights (0.551 vs. 0.465). (4) biker: YOLOv5 has slightly worse performance (0.295) than YOLOv3 (0.442). (5) truck: Both perform similarly, but YOLOv3 is slightly better at 0.630 and 0.584, respectively.

Advantage of YOLOv3: YOLOv3 slightly outperforms YOLOv5 in most of the categories (especially common traffic categories such as cars, pedestrians, and bicycles.) This may be due to the fact that YOLOv3 has been trained more on datasets with these categories, or that its model structure is more sensitive to these categories.

Advantages of YOLOv5: Although slightly lower than YOLOv3 overall (mAP@0.5), but YOLOv5 performs better in some specific categories (e.g., red lights). This could mean that YOLOv5 may have better performance on some specific scenarios or specific datasets.

Overall, YOLOv3 slightly outperforms YOLOv5 in the Precision-recall curve plot. however, this does not mean that YOLOv3 is superior to YOLOv5 in all scenarios and datasets. There are other factors to consider when choosing which model to use, such as the actual application scenario, model size, and run speed.

5.5 Recall-confidence

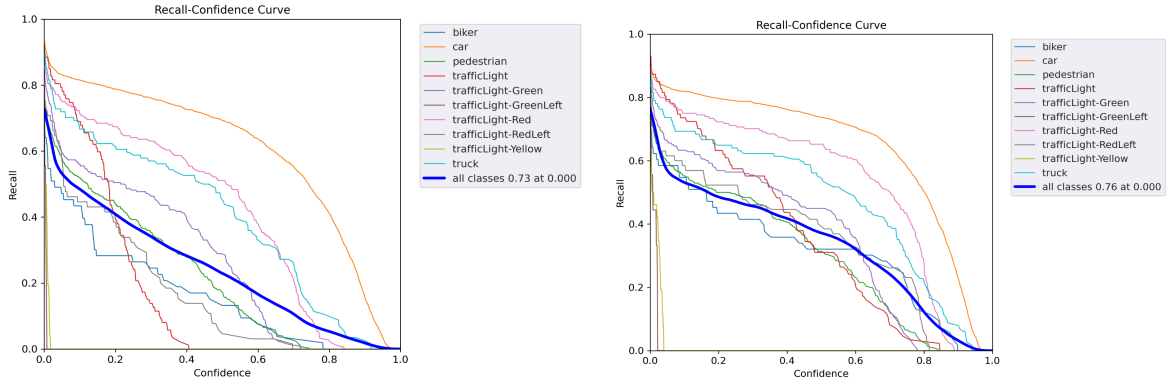


Figure 5.7 Recall-confidence curve for YOLOv5

This figure shows a graph called the "Recall-Confidence Curve," which measures the recall of several traffic-related categories at different confidence levels. The recall describes the proportion of positive samples that the model correctly identifies out of the total number of positive samples.

From these two graphs, Yolov5 performs better than Yolov3 overall, especially at high confidence levels. However, Yolov3 may perform slightly better in some specific categories and at low confidence levels. For example, for "biker," Yolov3 has slightly higher recall values than Yolov5 at lower confidence values.

At higher confidence intervals (around 0.7-1.0), Yolov5 generally has higher recall values than Yolov3, which implies that Yolov5 is better and more reliable at high confidence levels. However, in the lower CONFIDENCE interval (about 0.0-0.7), the curves of Yolov3 and Yolov5 are closer to each other, but Yolov3 is slightly higher than Yolov5 in some classes, which indicates that there is not much difference between the two performances in low confidence, but Yolov3 performs a little bit better in some classes.

The ALL CLASSES curve is higher in the Yolov5 plot, indicating that, on the whole, Yolov5 is better at detecting all classes.

Overall, Yolov5 seems to perform better than Yolov3 in the Recall-confidence curve overall, especially at high confidence levels. However, Yolov3 may perform slightly better in some specific categories and at low confidence levels.

5.6 The accuracy of the recognition

Figure 5.7 The accuracy of the recognition for YOLOv5 &YOLOv3

The above figure shows an example of a series of road scenarios by showing the application of the YOLOv5 algorithm and YOLOv3, with each sub-figure processed by the object detection algorithm. Specifically, each detected object is surrounded by a red rectangular box with the corresponding classification label and confidence score. The main objects detected above include vehicles, pedestrians and traffic signals.

Both sets of images yolov5 seem to have relatively high confidence in most cases. Whereas yolov3 may miss detection in some cases or have lower confidence for some objects.

In most images, the confidence level of yolov5 is higher than that of yolov3.

For example, for vehicle detection, yolov5 shows higher confidence in most images, while yolov3 may miss detection or have lower confidence for certain vehicles in some images. Another example is for the detection of traffic signals, yolov5 shows higher confidence in most of the images, especially when the signal's colour is green. Whereas yolov3 may not be detected or have lower confidence in it sometimes. For pedestrian detection, both models were able to detect pedestrians in most cases, but yolov5's confidence level seemed to be higher in some images.

Overall, yolov5 seems to perform better than yolov3 in most cases, especially in terms of confidence. yolov5 detects vehicles, traffic lights and pedestrians more accurately and confidently than yolov3.

6. Finding

In our project, we find that YOLOv5 demonstrates a higher detection accuracy, particularly in complex environments with multiple overlapping objects. The bounding boxes drawn by the YOLOv5 model are tighter around the objects and demonstrate less jitter than the YOLOv3.

The classification labels provided by YOLOv5 are also consistent and precise, especially in cases with multiple objects of the same type (e.g., several cars in a row). It offers fewer false positives and seems to have a robust mechanism for handling occlusions, where part of an object is hidden behind another.

YOLOv3, on the other hand, occasionally struggles in these intricate scenarios. There are instances where YOLOv3 fails to detect some objects or provides a bounding box that is either too large or misplaced. It also sometimes misclassifies an object or gives it a lower confidence score, indicating less certainty in its prediction. This might be evident in scenarios with objects at a distance or when they are partially obscured.

Moreover, YOLOv5's detection of pedestrians and traffic signals is notably superior. It accurately identifies and boxes them even when they are close to other objects or when only a fraction of the object is visible. YOLOv3, however, sometimes either misses them or provides a less accurate bounding box.

In scenarios with dynamic lighting or shadows, YOLOv5 seems to maintain its detection accuracy without much fluctuation in confidence scores. YOLOv3's performance, on the other hand, might be affected slightly in such cases, as evidenced by the occasional missed detections or reduced confidence scores.

Overall, based on the visual results from the provided road scenarios, YOLOv5 demonstrates a superior performance in object detection, especially in challenging situations. It provides precise bounding boxes, accurate classification labels, and consistently high confidence scores. YOLOv3, although competent, has certain shortcomings, especially when dealing with intricate scenarios. The underlying datasets may also influence this difference in performance the models were trained on, the architectural modifications and optimizations done in YOLOv5, and other factors. However, for applications requiring high accuracy and reliability in real-world scenarios, YOLOv5 might be a more suitable choice.

7. Conclusion & Future Work

In this work, we provide insights into the current state and trends of autonomous driving technology, especially in the areas of object detection and road condition analysis. By comparing and analyzing two different versions of the YOLO model, YOLOv3 and YOLOv5, we demonstrate the superior performance of YOLOv5 when dealing with complex and challenging scenarios.YOLOv5 not only excels in detection accuracy but also demonstrates its robustness in terms of real-time performance and stability. This is critical for the real-time response and reliability of autonomous driving systems, which directly impact the vehicle's ability to drive safely and make decisions. We also explore the challenges facing autonomous driving technology, including dealing with uncertainty in dynamic environments and improving the system's robustness. The contents of the document are not only inspiring for researchers and engineers in the field of autonomous driving but also provide valuable information and insights for a wide range of people interested in intelligent mobility and autonomous driving.

Future work will focus on improving the accuracy and robustness of autonomous driving systems. Especially in complex and dynamic road conditions, the system needs to be able to detect and understand the surrounding environment more accurately. In addition, further optimization of the system's real-time and decision-making capabilities is essential. We propose to delve into more advanced object detection and image processing algorithms to improve the system's performance. Meanwhile, enhancing the system's adaptability to different environmental conditions (e.g., bad weather, complex traffic conditions, etc.) is also an important direction for future research. In addition, considering the safety and reliability of autonomous driving systems, it will become increasingly important to investigate how to improve the system's security in the face of potential threats and attacks. Overall, future work needs to improve the performance of the system while ensuring its safety and reliability in order to promote the widespread application and development of autonomous driving technology.

9. Contributions of Team Members

Zheng Feng: Team Management, YOLO method implementation, report writing. (25%)

Xinyi He: YOLO model exploration, report writing. (25%)

Zhuo Shi: YOLO model exploration, data analysis, report writing. (25%)

Chenghao Liu: YOLO model exploration, report writing. (25%)

10. Reference

1. Russell, S. J., & Norvig, P. (2010). Artificial intelligence a modern approach. London.

2. Anderson, J. & Smith, K. (2019). *Automated Vehicular Systems: A New Dawn*. TransTech Publications, 45(2), 150-162.

3. Nyce, C. M. (2023, September 1). It's a weird time for driverless cars. The Atlantic. https://www.theatlantic.com/technology/archive/2023/08/driverless-taxis-waymo-cruise/675161/

4. N. A. Greenblatt, "Self-driving cars and the law," in IEEE Spectrum, vol. 53, no. 2, pp. 46-51, Feb. 2016, doi: 10.1109/MSPEC.2016.7419800.

5. H. He, "Yolo Target Detection Algorithm in Road Scene Based on Computer Vision," 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, 2022, pp. 1111-1114, doi: 10.1109/IPEC54454.2022.9777571.

6. B. B. Elallid, N. Benamar, N. Mrani and T. Rachidi, "DQN-based Reinforcement Learning for Vehicle Control of Autonomous Vehicles Interacting With Pedestrians," 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakheer, Bahrain, 2022, pp. 489-493, doi: 10.1109/3ICT56508.2022.9990801.

7. Y. Feng, W. Hua and Y. Sun, "NLE-DM: Natural-Language Explanations for Decision Making of Autonomous Driving Based on Semantic Scene Understanding," in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 9, pp. 9780-9791, Sept. 2023, doi: 10.1109/TITS.2023.3273547.

8. Zang, Shizhe & Ding, Ming & Smith, David & Tyler, Paul & Rakotoarivelo, Thierry & Kaafar, Dali. (2019). The Impact of Adverse Weather Conditions on Autonomous Vehicles: Examining how rain, snow, fog, and hail affect the performance of a self-driving car. IEEE Vehicular Technology Magazine. PP. 1-1. 10.1109/MVT.2019.2892497.

9. Roboflow. "Udacity Self Driving Car Dataset." Roboflow Public Datasets.

10. Sallab, A. E., Sobh, I., Zahran, M., & Shawky, M. (2019, November 24). *Unsupervised neural sensor models for synthetic lidar data augmentation*. arXiv.org. https://arxiv.org/abs/1911.10575

11. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2016.91

12. GitHub.(n.d.). https://github.com/xia0long/self-driving-experiments-in-carla/tree/master/object_detection_with_yolov3

Appendix

The example of our system output image:

Yolov3 Evaluation Results:

```
Fusing layers...
yolov3 summary: 190 layers, 61551280 parameters, 0 gradients, 154.7 GFLOPs
val: Scanning /content/valid/labels.cache... 476 images, 67 backgrounds, 0 corrupt: 100% 476/476 [00:00<?, ?it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 15/15 [00:28<00:00,  1.92s/it]
                   all        476       3112      0.702      0.438      0.467      0.217
                 biker        476         53      0.563      0.365      0.442      0.186
                   car        476       2050      0.763      0.775      0.807      0.466
            pedestrian        476        306      0.511      0.444      0.409      0.151
          trafficLight        476         87      0.524      0.517       0.52      0.242
    trafficLight-Green        476        218      0.649      0.526      0.551      0.195
trafficLight-GreenLeft        476          9          1          0     0.0524     0.0145
      trafficLight-Red        476        184      0.651      0.685      0.684      0.302
  trafficLight-RedLeft        476         65      0.633      0.451      0.494      0.221
   trafficLight-Yellow        476         26          1          0     0.0777     0.0278
                 truck        476        114      0.723      0.623       0.63      0.368
Speed: 0.3ms pre-process, 40.1ms inference, 2.4ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp
```

Yolov5 Evaluation Results：

```
Fusing layers...
custom_YOLOv5s summary: 182 layers, 7273488 parameters, 0 gradients
val: Scanning /content/valid/labels.cache... 476 images, 67 backgrounds, 0 corrupt: 100% 476/476 [00:00<?, ?it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 15/15 [00:15<00:00,  1.04s/it]
                   all        476       3112      0.615      0.381      0.368      0.164
                 biker        476         53      0.478      0.283      0.295      0.113
                   car        476       2050      0.675      0.778      0.795      0.442
            pedestrian        476        306       0.31      0.402      0.285     0.0868
          trafficLight        476         87      0.324      0.275      0.294      0.108
    trafficLight-Green        476        218      0.618      0.486      0.465      0.167
trafficLight-GreenLeft        476          9          1          0     0.0109    0.00421
      trafficLight-Red        476        184      0.486      0.647      0.575      0.249
  trafficLight-RedLeft        476         65      0.663      0.338      0.361      0.125
   trafficLight-Yellow        476         26          1          0     0.0196    0.00553
                 truck        476        114      0.591      0.596      0.584      0.343
Speed: 0.3ms pre-process, 8.1ms inference, 2.6ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp
```