Visual Recognition

Object Detection, Recognition and Segmentation

Aleix Pujol, Diana Tat, Georg Herodes, Gunjan Paul
Master in Computer Vision

***Abstract*—This week came with the challenge of working with the Detectron2 framework and understanding how to deal with annotation formats. We have applied Faster R-CNN and Mask R-CNN models to track and segment the cars and pedestrians, using the KITTI-MOTS dataset and COCO weights. We have also tried the newest version of YOLO v9 and see its performance.**

I. Introduction

During this task we have used PyTorch to implement our neural network and Detectron2 for an efficient object detection PyTorch is a very adaptive and dynamic method of building models. Because of that, we can modify and debug the code easy, as well as transform it to PyTorch Lightning. On the other hand, Keras, has a more high-level abstraction and computational graphs that facilitate the efficient deployment and optimisation of the code.

This week we have been using Detectron2 framework which is made by Facebook Artificial Intelligence Research's. It is based on PyTorch and it features panoptic segmentation, bounding box and instance segmentation mask object recognition. It provides a wide range of models, such as Panoptic FPN, Dense Pose, TensorMask and the ones that we focused on, Mask R-CNN and Faster R-CNN.

Faster R-CNN provides accurate object localisation by using Region of Interest (RoI) pooling. It suggest regions for examination and then it uses Convolutional Neural Networks (CNNs) to extract the information, simplifying the detection process. Faster R-CNN produces generally accurate object localisation, but it can't work with masks, which are useful in situations when we need more precise information of the object boundaries. For this reason, Mask R-CNN was introduces as an extension of Faster R-CNN. It overcomes its drawback by having an extra branch that is responsible for segmenting the prediction mask. It uses pixel-wise segmentation and it more precisley in object identification.

Also, as an optional task we have tried the new version of YOLO, YOLO v9. YOLO is an algorithm that performs real-time object detection by dividing the image into a grid and predicting bounding boxes and class probabilities. We have evaluated and compared its performance with the Faster R-CNN and Mask R-CNN models.

II. Related Works

A. Neural Networks in Computer Vision

In the field of computer vision, as [2] says, deep learning has become indispensable for how machines perceive and process visual data. Convolutional neural networks (CNNs), in particular, are deep learning models that have demonstrated unmatched performance in tasks like object detection, instance segmentation, and image categorization. They are able to recognise complex patterns and features, going beyond conventional computer vision techniques, thanks to their autonomous learning of hierarchical representations from unprocessed visual data.

B. Object Detection

1) Objectives and Uses: Deep learning has brought to an important shift in object detection, a fundamental aspect of computer vision. In important studies like Zhao et al.'s [17] from 2018, the writers thoroughly examine this paradigm change by diving into the complexities of object detection enabled by deep learning techniques. A more recent study, published in 2020 by Xiao et al. [15], offers a modern viewpoint by showcasing the developments in deep learning-based object identification and its numerous uses in multimedia tools and applications. Moreover, Kaur and Singh's paper [5] from 2023 provides a thorough examination, highlighting how object detection is changing in the context of digital signal processing.

2) Two-Shot Detectors: Two-Shot Detectors are an important advance in the field of object detection since they enable more precise and nuanced identification of visual objects. 2015 saw the introduction of Faster R-CNN[3]. By integrating region proposal networks, this paper study transformed object identification and greatly enhanced real-time performance. The Faster R-CNN framework built on this basis to create the framework for later advancements in two-shot detection techniques.

One key illustration of this trajectory's evolution is the Faster Mask R-CNN refinement by Ren et al. [11] that same year. This iteration presents Region Proposal Networks (RPNs), which are meant to make region proposals more quickly and efficiently while also accelerating the object detection process. These Two-Shot Detectors are very useful in improving the accuracy of object detection and laying the groundwork for future improvements like the incorporation of instance segmentation as shown by Mask R-CNN [4]. This is especially true of Faster R-CNN and its variants.

With the introduction of Mask R-CNN in [4], instance segmentation has made substantial progress. This model adds a second branch to the Faster R-CNN architecture, which predicts segmentation masks in addition to bounding boxes and class labels. Mask R-CNN works by first producing region suggestions, which it then refines to provide precise object

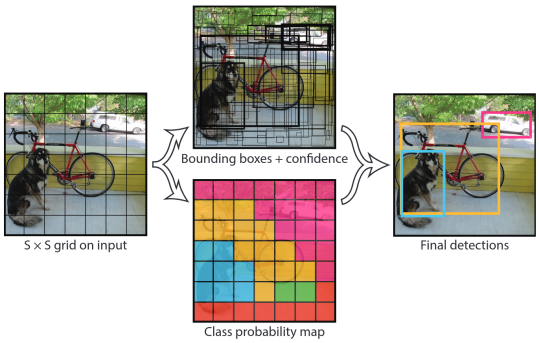

Fig. 1. YOLO models represent detection as a regression problem by dividing the image into a grid and predicting classes, bounding box coordinates, and confidences for each cell in the grid [10]

masks. A critical layer of information is added by integrating pixel-level segmentation, which allows the model to offer detailed spatial awareness of object instances inside an image.

3) Single-Shot Detectors: In the field of object detection, single-shot detectors (SSDs) have become an important category because they provide real-time processing capabilities without sacrificing accuracy.

The YOLO series of single-shot detectors, initially developed and iterated upon by Redmon and Farhadi in [10], [8], [9] employs methods such as a grid-based approach and the prediction of bounding boxes as deviations of anchor boxes to provide single-pass detection capabilities. This method has been extensively improved by various researchers in both indurtry and academia. The most recent updates such as YOLOv7 [13], and most recently, YOLOv9 [14] enhance the performance of single shot detectors even further by optimizing not only the network architecture but also introduce and apply methods such as ELAN, GELAN, and PGI to the training process to overcome issues such as the information bottleneck and inefficient parameter utilization[13][14]. Such improvements have made the more recent versions of YOLO not only faster, but also more accurate than their predecessors which removes the need to trade off speed or accuracy when choosing a model architecture for object detection.

4) Segmentation: In computer vision segmentation is important because it allows to understand the input data better. A in depth analysis of object detection using deep learning for segmentation is explained in [18], enhancing the precision of the segmentation results. In the paper [16] the focus is on exploring different models and approaches, highlighting the effectiveness of trying different methods and their applicability in different contexts. Furthermore, [6] provides an examination of object detection using deep learning in the context of digital signal processing, illuminating the significant developments and difficulties in the area.

5) RetinaNET: RetinaNet, as [7], is an object detection method that tackles the complexities involved in identifying objects of different sizes, particularly small and overlapping ones. The research paper focuses on addressing the challenges commonly found in object detection approaches. A notable aspect of this work is the introduction of the focal loss mechanism, which involves adjusting the weight of training samples. This strategy enables the model to prioritise instances during training leading to performance in detecting complex objects. Additionally RetinaNet incorporates a Feature Pyramid Network (FPN) that effectively captures features at scales proving essential for handling objects with dimensions. The combination of loss and FPN sets RetinaNet apart. Helps it achieve outstanding results, in practical object detection scenarios.

6) DETR: DETR (End, to End Object Detection with Transformers) represents a step in the realm of object detection. The study [1] introduces a method for detecting objects by using transformer architectures. In contrast to other techniques, DETR directly forecasts object queries and their positions removing the necessity for anchor boxes and non maximum suppression. This comprehensive framework simplifies the detection process. Tackles the challenges associated with object sizes and overlaps. The research highlights DETRs capacity to redefine how object detection is approached, delivering localisation and classification outcomes. It showcases the effectiveness of transformer models in tasks related to computer vision signalling a shift, from methodologies.

C. Object Detection

1) Objectives and Uses: Deep learning has brought an important shift in object detection, a fundamental aspect of computer vision. In important studies like Zhao et al.'s [17] from 2018, the writers thoroughly examine this paradigm change by diving into the complexities of object detection enabled by deep learning techniques. A more recent study, published in 2020 by Xiao et al. [15], offers a modern viewpoint by showcasing the developments in deep learning-based object identification and its numerous uses in multimedia tools and applications. Moreover, Kaur and Singh's paper [5] from 2023 provides a thorough examination, highlighting how object detection is changing in the context of digital signal processing.

III. Methodology

1) Faster-R-CNN: Faster R-CNN works by initial removing distinct attributes from the input photo making use of a distinct neural network (CNN) foundation such as ResNet or VGG. These functions are after that fed right into both the RPN coupled with a region-based discovery network.

The RPN runs by moving a tiny network normally a couple of distinct layers over the function map creating a collection of area propositions. These propositions include bounding boxes and also linked objectness ratings, showing the probability of having a things. The RPN is educated end-to-end to create premium area propositions effectively.

During training, the Faster-R-CNN model is trained end-to-end using a multi-task loss function that combines classification and regression losses. We employ a stochastic gradient descent (SGD) optimiser with momentum to minimise the loss function and update the network parameters.

In our experiments, we fine-tune a pre-trained Faster-R-CNN model on our dataset, which consists of images annotated with object bounding boxes. We evaluate the performance of the trained model using standard metrics such

as mean Average Precision (mAP) and compare it against baseline methods and other state-of-the-art object detection frameworks.

2) Mask-R-CNN: Mask R-CNN extends Faster R-CNN by integrating a totally convolutional network (FCN) branch together with the existing area proposition network (RPN) as well as region-based discovery network. This FCN branch produces division masks for recommended areas allowing pixel-level division within pictures. Unlike Faster R-CNN which concentrates exclusively on bounding box forecast, Mask R-CNN incorporates things discovery and also circumstances division jobs in an unified style making it ideal for jobs needing exact things limit circulation. Throughout training both designs maximise multi-task loss features consisting of category as well as bounding box regression losses, yet Mask R-CNN includes a mask forecast loss to educate the division mask generation branch.

During training, the Mask-R-CNN model is trained end-to-end using a multi-task loss function that combines classification, bounding box regression, and mask prediction losses. Similar to Faster-R-CNN employ a stochastic gradient descent (SGD) optimiser with momentum to minimise the loss function and update the network parameters.

In our experiments, we fine-tune a pre-trained Mask-R-CNN model on our dataset, which consists of images annotated with object bounding boxes and instance segmentation masks. We evaluate the performance of the trained model using standard metrics such as mean Average Precision (mAP) at various thresholds.

We can see that Mask R-CNN and Faster R-CNN for feature extraction have a same backbone, ResNet50, in Figure 2. Both of them have a Region of Interest (RoI) pooling layer for feature extraction from these proposals and a Region Proposal Network (RPN) for generating region proposals. Nonetheless, the "ROI Head" element of the networks accounts for the majority of the differences. The ROI Head in Faster R-CNN is made up of sub-networks for bounding box regression and object classification. On the other hand, Mask R-CNN builds upon Faster R-CNN by incorporating a second sub-network for mask prediction, which allows it to execute instance segmentation in addition to object detection.

3) YOLOv9: As an example of a newer SOTA object detection model we employ YOLOv9, an evolution of the YOLO (You Only Look Once) object detection framework, for our object detection task. YOLOv9 builds upon the principles of the YOLOv7 architecture while introducing several key improvements to enhance both accuracy and speed. Such as extending the ELAN method proposed in the YOLOv7 paper to create GELAN, as well as by introducing a new concept called Programmable Gradient Information.

One substantial addition in YOLOv9 is the expansion of the ELAN (Reliable and also Light-weight Anchor-free Network) approach recommended in the YOLOv7 paper, causing GELAN (International Extremum Learning-based Anchor-free Network) GELAN enhances the performance of item discovery by efficiently taking care of range variants and also things localisation precision. Additionally, YOLOv9 presents an unique principle called Programmable Gradient Information

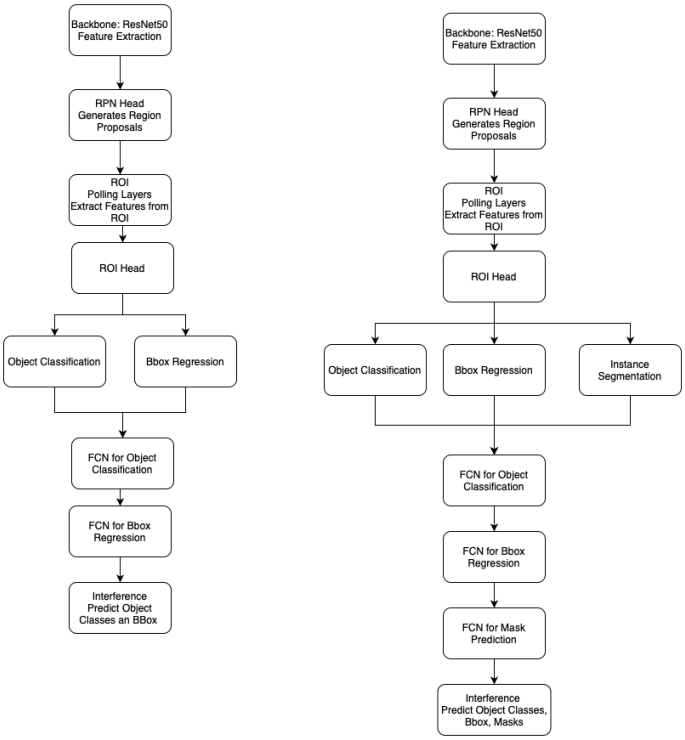

Fig. 2. Architecture of Faster R-CNN(a) and Mask R-CNN(b).

which better improves the version's capacity to discover things with high accuracy.

During training, YOLOv9 is trained end-to-end using a combination of localisation loss, confidence loss, and class probability loss. The model is optimised using the ADAM optimiser with techniques such as learning rate scheduling and momentum to improve convergence and stability. As is common with newer YOLO models, aggressive augmentation techniques such as mosaicing and color jitter are used in the training process to increase model robustness to noise, scaling and translation.

In our experiments, we fine-tune a pre-trained YOLOv9 model on our dataset, which consists of images annotated with object bounding boxes. We evaluate the performance of the trained model using standard metrics such as mean Average Precision (mAP) and Intersection over union(IoU).

We can observe the model architecture of YOLOV9 in the image 3. It predicts bounding boxes and class probabilities for each grid cell by first splitting the input image into a grid. A backbone network for feature extraction—typically Darknet or a comparable architecture—makes up the YOLOv9 architecture. Additionally, it has several detecting heads at varying scales, which enables the model to effectively recognise objects with a range of sizes and aspect ratios. Compared to two-stage detectors like Faster R-CNN, YOLOv9 is faster and more suited for real-time applications because it uses a single-stage architecture. Furthermore, YOLOv9 offers a number of enhancements over its predecessors, such as improved training procedures and feature pyramid networks (FPN), which lead to better detection performance. All things considered, YOLOv9 provides a speed and accuracy balance that makes it a popular

option for object detection jobs in a variety of areas.

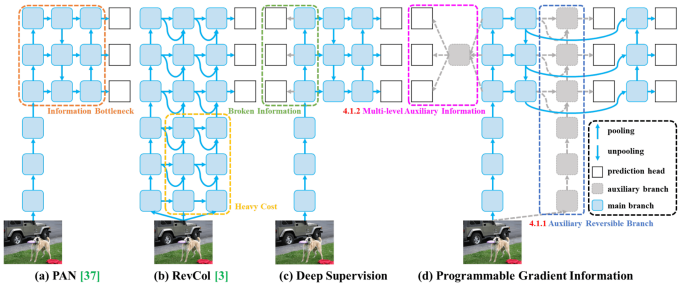

Figure 3. PGI and related network architectures and methods. (a) Path Aggregation Network (PAN)) [37], (b) Reversible Columns (RevCol) [3], (c) conventional deep supervision, and (d) our proposed Programmable Gradient Information (PGI). PGI is mainly composed of three components: (1) main branch: architecture used for inference, (2) auxiliary reversible branch: generate reliable gradients to supply main branch for backward transmission, and (3) multi-level auxiliary information: control main branch learning plannable multi-level of semantic information.

Fig. 3. Architecture of YOLOv9. Source: *YOLOv9: Ultralytics' YOLOv5 Implementation* [12]

In Figure 3, we can see the architecture of YOLOv9, which stands for You Only Look Once version 9. YOLOv9 is a state-of-the-art object detection algorithm that operates by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. The YOLOv9 architecture consists of a backbone network, typically Darknet or a similar architecture, for feature extraction. It further includes multiple detection heads at different scales, allowing the model to detect objects of various sizes and aspect ratios efficiently. YOLOv9 employs a single-stage architecture, making it faster and more suitable for real-time applications compared to two-stage detectors like Faster R-CNN. Additionally, YOLOv9 introduces several improvements over its predecessors, including feature pyramid networks (FPN) and enhanced training strategies, resulting in superior detection performance. Overall, YOLOv9 offers a balance between speed and accuracy, making it a popular choice for object detection tasks across various domains.

IV. Experimental Design

1) Dataset: KITTI-MOTS: The dataset includes a collection of training sequences totalling 12, which contain 8,073 pedestrian masks and 18,831 annotated car masks. The validation set is made up of 9 sequences with 3,347 pedestrian masks and 8,068 car masks. Additionally there are 29 sequences specifically reserved for testing. For labelling instances cars are labeled in order starting from 1000 onwards while pedestrians follow a similar pattern starting from 2000. We can observe how a frame from the dataset looks like in figures 4.

2) Metrics:

a) Overview: For the evaluation of our code, we have use Precision, Recall, F1, AP AP@50 and mAP@75.

Mean Average Precision, at k (mAP@k) assesses the balance between precision and recall focusing on the precision when considering the top k predictions for each image. It involves calculating the precision for each query based on the relevance of predictions within the top k.

$$\text{MAP@k} = \frac{1}{|Q|} \sum_{q \in Q} \left(\sum_{i=1}^{k} \frac{\text{Precision@}i \cdot \text{relevant}_{q,i}}{i} \right)$$

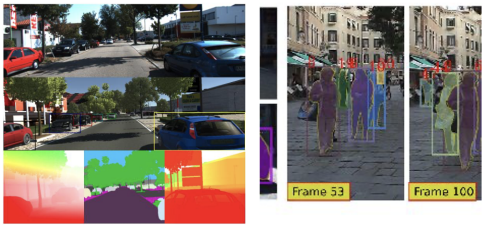

Fig. 4. KITTI MOTS Dataset(left) and Annotations(right).

Precision measures how accurately the model identifies objects by comparing true positive predictions to all predicted positives.

Recall indicates the models ability to detect all objects by comparing positives to all actual positives.

$$\text{Precision} = \frac{\text{TP(True Positive)}}{\text{TP(True Positive)} + \text{FP(False Positive)}}$$

Intersection over Union (IoU) is used to determine how much two bounding boxes predicted and ground truth behind them overlap, which enables one to get an idea of the object localisation accuracy by assessing intersection and union areas of these boxes. This helps in determining if a prediction was a true positive, (TP), a false positive (FP), or a false negative, (FN) depending on the overlap that occurs between the predicted box and the ground truth box.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} = \frac{\text{TP}}{\text{TP + FP + FN}}$$

The F1 Score combines Precision and Recall while AP is calculated by integrating over different IoU thresholds under the precision-recall curve. Its influence shows how much IoU contributes in evaluating object detection model effectiveness since it gives information about detection accuracy as well as localisation capability.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Together, these metrics provide a framework for evaluating the effectiveness of object detection models.

b) Code evaluation: In case of **Faster R-CNN** we have evaluated the architecture, including its convolutional layers, RPN, and R-CNN for object detection. The model was fine-tuned using our dataset, with careful adjustments made to hyper-parameters such as learning rates and batch sizes for optimal performance. We have constantly monitored the model's training progress to ensured convergence and prevent the overfitting. We have tested it on a separate validation dataset, assessing precision, recall, and mAP scores across various object classes. In addition, we have extracted some qualitative analysis to identify potential weaknesses and areas for improvement.

Similarly, we evaluated the **Mask R-CNN** for instance segmentation, including its FPN, RPN, and mask prediction head. By conducting hyper-parameter tuning experiments, we optimised the model for our specific task, exploring different learning rates, weight decay values, and augmentation

techniques. We have evaluated the model's performance on segmentation and object detection tasks, analyzing accuracy, IoU scores, and mAP across different object categories.

YOLOv9 was evaluated for real-time object detection. The emphasis was on its single CNN architecture, with a focus on balancing speed and accuracy. We have conducted extensive exploration of model configurations and fine-tuning parameters, considering dataset characteristics and augmentation during training. In addition to traditional metrics, real-time processing capabilities were assessed against speed benchmarks. The evaluation provided deep insights into YOLOv9's strengths and limitations for diverse object detection tasks.

3) Implementation details:

a) Faster R-CNN: We used PyTorch and Detectron2 for our Faster R-CNN implementation. The first step we took was to define the structure of the model and load pre-trained weights from the COCO dataset. We optimised performance by setting parameter like score threshold for predicting instances. For this purpose, we developed a function that loads images for testing on custom datasets and converts annotations into the COCO format using $convert_to_coco$ function. This involved extracting bounding box coordinates and class labels as well as converting coordinates to $XYXY_ABS$ format so as to ensure compatibility with Detectron2. After that, we saved the COCO dataset into a JSON file that could be employed in either training or evaluation purposes. Then, using an adjusted configuration, we created a Faster R-CNN predictor which could perform inference on test images. The Visualizer component visualised detection outcomes by superimposing bounding boxes and class labels onto input images for easier interpretation.

Fig. 5. Faster R-CNN Inferred Image

Afterwards we stored the visualized image in a file, which enabled us to analyze it and share the detection results. This method shows how efficiently we used trained object detection models from Detectron2 to make inferences on custom datasets proving how well Faster R CNN works, in practical situations.

Fine-Tuning Faster R-CNN After loading the model with the KITTI MOTS dataset, we moved to the next step of fine-tuning that model to both the domain and the task under test. To start with, we used the trained Faster R-CNN model with an R50 FPN backbone and 3x training duration as the starting point. During the retraining process we refined the hyperparameters in order to improve the models efficacy. This was done by training batch of 32 for 400 epochs and adjusting batch size per image to 128. Besides that, we looked through how different learning rates affected the model performance by checking outcomes on the test set.

b) Mask R-CNN: For Mask R-CNN the same approach was adopted to implement the model with all the required parameters and pre-trained weights from the COCO dataset. We used the R50-FPN architecture with a 3x training schedule. To ensure the reliability of our model's outputs we set the threshold limit for instance predictions. In contrast with Faster R-CNN, Mask R-CNN needs both images and masks as input for the segmentation of the pixel level. This leads to modifications in data format and mask annotation during pre-processing phase. Despite the modifications in the procedure, mask annotations came in addition to the bounding boxes. Moreover, when loading data, bounding box coordinates and mask attributes were also parsed and applied to the dataset, guaranteeing that the model received both image and mask data as input.

As before, we have created a visualizer instance that contains bounding boxes and maks, as we can see in the images 6 and 7 .

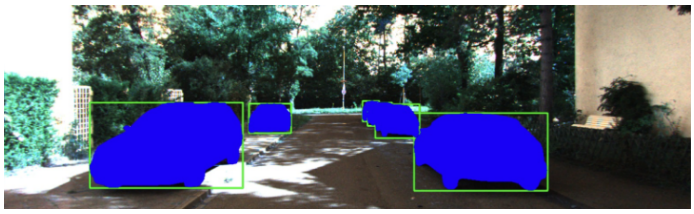

Fig. 6. Training KITTI-MOTS sample used for inference

Fig. 7. Inference output case

In general, Mask R-CNN was implemented using the same approach as for Faster R-CNN, the only differences being the input information.

c) YOLOv9: For the optional task we have used the new YOLOv9 algorithm. We start by placing the images in directories with their matching ground truth and labels. Then we are converting the bounding box coordinated from YOLO format to $XYXY$ which allows a consistent visualisation and easy of interpretation. Next we analyse each image and retrieve its dimensions in order to get the precise bounding box location. The text files annotations are parsed to get the bounding box coordinates and class IDs.

We can't say a lot about the implementation details, because it was already been implemented so we focused on fine-tuning it and see its performance on the person and car classes in the dataset. We have used fine-tuned the weights that were adjusted after being training on the COCO dataset.

This shows us that even though the model performs good when it comes to bigger objects, it fails to detect the small ones, as we can see in 8 and 9. We discovered that the class

10 objects in the KITTI-MOTS dataset might be be annotated to potentially resolve inconsistencies because their absence would affect the overall performance.

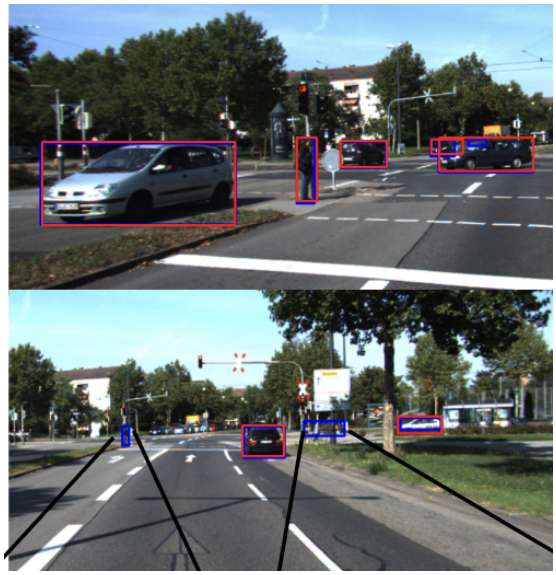

Fig. 8. YOLOv9 Predictions (red)

Fig. 9. YOLOv9 Ground Truth (blue)

V. Results

Upon evaluating the pre-trained Faster R-CNN version's efficiency on the KITTI MOTS examination information collection, it attained a Precision of 0.4191 Recall of 0.5702 and also F1 Score of 0.4629 as displayed in table I. We have observed some difficulties particularly with the "Pedestrian" classification, where we have obtained a precision of 0.1876 was observed, which can be seen in II. This concern originated from circumstances where the version inaccurately recognized obstacles as pedestrians a result of visual resemblances stressing the requirement for precision enhancement in real-world circumstances.

Metric	Precision	Recall	F1 Score
Overall	0.4191	0.5702	0.4629

TABLE I
Overall Metrics

The examination outcomes of the Mask R-CNN on the KITTI-MOTS information, summarised in Table III, highlighted its efficiency in pixel-level division, particularly with an mAP of 44.23% for cars together with 21.76% for pedestrians. The particular metrics such as AP@50 as well as mAP@75 offered understandings right into the design's precision at various IoU limits, showcasing its strength throughout numerous circumstances.

Class	Precision/Recall/F1	AP	AP@50	mAP@75
Pedestrian	0.1876 / 0.2769 / 0.2104	0.1876	9.9139	6.6093
Car	0.6505 / 0.8636 / 0.7155	0.6505	34.3741	22.9161

TABLE II
Class-wise Metrics

Class	mAP	AP@50	mAP@75
Pedestrian	21.76	14.756	8.633
Car	44.23	37.651	24.651

TABLE III
Mask R-CNN Evaluation Results on KITTI-MOTS

In the case of Faster R-CNN, we experimented with different learning rates, aiming to strike a balance between precision and computational efficiency. Our results, as shown in Table IV, underscore the importance of careful hyperparameter tuning in achieving optimal performance.

LR	**mAP**	**mAP@50**	**Car mAP**	**Pedestrian mAP**	**Time**
0.0025	13.171	32.268	13.046	6.721	39 ms
0.01	45.323	75.034	53.506	28.025	38 ms
0.025	46.636	78.035	52.961	32.304	41 ms

TABLE IV
Effect of learning rate on Faster R-CNN training

Similarly, for Mask R-CNN, fine-tuning with the ResNet-50-FPN architecture and a three-time training schedule yielded slight improvements in mAP for both bounding box detection and pixel-level segmentation tasks, as demonstrated in Table V.

LR	**mAP**	**mAP@50**	**Car mAP**	**Pedestrian mAP**	**Time**
0.025	12.781	12.791	5.479	1.670	55 ms
0.1	13.180	13.353	5.257	1.660	55 ms

TABLE V
Evaluation results for bounding box detection on the test set

For YOLOv9, our emphasis got on boosting discovery precision with fine-tuning. In spite of very little renovations in mAP50, we observed that comprehensive calibration as well as specification modification were important for taking on real-world difficulties efficiently. The efficiency contrast of YOLOv9 versions, detailed in Table ref tab: yolov9, highlights the requirement for nuanced optimization techniques to accomplish substantial gains in discovery precision.

Model	**mAP50**	**mAP50:95**	**Time**
YOLOv9 Pre-trained	0.838	0.593	4.8 ms
YOLOv9 Fine-tuned	0.834	0.654	4.9 ms

TABLE VI
Performance Comparison of YOLOv9 Models

LR	mAP	mAP@50	Car mAP	Pedestrian mAP	Time
0.025	1.421	6.094	6.506	0.000	55 ms
0.1	5.612	0.886	6.390	0.231	55 ms

TABLE VII
SEGMENTATION RESULTS ON THE TEST SET

On the whole our execution method focused on precise criterion adjusting and also version adjustment to attend to the particular subtleties of the KITTI MOTS information collection. By utilizing the toughness of each version design plus maximizing essential hyperparameters, we intended to optimize efficiency throughout numerous item discovery as well as division jobs eventually adding to innovations in real-world computer system vision applications.

VI. CONCLUSIONS

When comparing the different **object detection models** we have noticed that Faster R-CNN exhibited robustness, but it faced challenges in crowded or occluded scenarios. Mask R-CNN has improved segmentation, but it still struggles with different urban scenes. For YOLOv9 we noticed some speed capabilities, but again it fails to detect small object accurately.

In terms of the **dataset**, dealing with the KITTI-MOTS information, especially concerning annotation layouts as well as information prep work highlights the importance of precise information handling, which can considerably influence the efficiency of things discovery versions together with in circumstances requiring exact division.

We found it more difficult to implement and **fine-tune** Mask R-CNN than Faster R-CNN. While the algorithm and the approach was more or less the same, we think that the dataset preparation made the difference and it might be a reason why the model struggles to precisely segment pedestrian cases.

YOLOv9 clearly outperforms the R-CNN based methods, which is expected as R-CNN-s were SOTA for several years and YOLOv9 is very recent. YOLOv9 struggling with very small objects is expected as that is a common trait of object detection models, especially single shot detectors in general. We kept in mind substantial searching for that highlight the effectiveness of Mask R-CNN in pixel-level division jobs, attaining good results for mAP ratings for both car and pedestrian classes. Furthermore, YOLOv9 was more efficient contrasted to R-CNN-based approaches, especially in identifying smaller sized things, while its concurrent handling abilities and also precision make it a guaranteeing alternative for different things discovery jobs.

In conclusion, even though these models face obstacles including occlusion, crowded scenes, and small item recognition, our research shows how promising these can be in a range of practical applications. For the area of computer vision to grow further, ongoing research efforts focused on improving model architectures, investigating alternative datasets, and tackling new issues will be crucial. Moving forward, if we would explore alternative datasets, and addressing emerging challenges, would be interesting for further advancements in the field of computer vision.

REFERENCES

[1] Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020, pp. 213–229. DOI: 10.1007/978-3-030-58536-5_13. URL: https://scontent-mad2-1.xx.fbcdn.net/v/t39.2365-6/154305880_816694605586461_2873294970659239190_n.pdf?_nc_cat=108&ccb=1-7&_nc_sid=3c67a6&_nc_ohc=GV0hRgf0mloAX-6Ei5w&_nc_ht=scontent-mad2-1.xx&oh=00_AfAtMfolfuVZG6Wml8P15QF68mou9j_nTo-ALOOdzkLIOg&oe=65EF9903.

[2] Junyi Chai et al. “Deep learning in computer vision: A critical review of emerging techniques and application scenarios”. In: *Machine Learning with Applications* 6 (2021), p. 100134. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100134. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000670.

[3] Ross Girshick. “Fast R-CNN”. In: (2015), pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[4] Kaiming He et al. “Mask R-CNN”. In: (2017), pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[5] Ravpreet Kaur and Sarbjeet Singh. “A comprehensive review of object detection with deep learning”. In: *Digital Signal Processing* 132 (2023), p. 103812. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2022.103812. URL: https://www.sciencedirect.com/science/article/pii/S1051200422004298.

[6] Ravpreet Kaur and Sarbjeet Singh. “A comprehensive review of object detection with deep learning”. In: *Digital Signal Processing* 132 (2023), p. 103812. DOI: https://doi.org/10.1016/j.dsp.2022.103812. URL: https://www.sciencedirect.com/science/article/pii/S1051200422004298.

[7] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.324. URL: https://ecommons.cornell.edu/items/69defa7c-f492-4deb-bc1e-28c3a6607481.

[8] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: (July 2017).

[9] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *ArXiv* abs/1804.02767 (2018). URL: https://api.semanticscholar.org/CorpusID:4714433.

[10] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: (2016), pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[11] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: 28 (2015). Ed. by C. Cortes et al. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[12] Ultralytics. *YOLOv9: Ultralytics' YOLOv5 Implementation*. 2021. URL: https://docs.ultralytics.com/models/yolov9/ (visited on 03/31/2024).

[13] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, Page Range. DOI: 10.1109/CVPR.2023.XXXXXXX. URL: https://openaccess.thecvf.com/content/CVPR2023/papers/Wang_YOLOv7_Trainable_Bag-of-Freebies_Sets_New_State-of-the-Art_for_Real-Time_Object_Detectors_CVPR_2023_paper.pdf.

[14] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*. 2024. arXiv: 2402.13616 [cs.CV].

[15] Youzi Xiao et al. "A review of object detection based on deep learning". In: *Multimedia Tools and Applications* 79.33–34 (June 2020), pp. 23729–23791. ISSN: 1573-7721. DOI: 10.1007/s11042-020-08976-6. URL: http://dx.doi.org/10.1007/s11042-020-08976-6.

[16] Xiao Youzi et al. "A review of object detection based on deep learning". In: *CoRR* 79.33-34 (2020), pp. 23729–23791. URL: http://dx.doi.org/10.1007/s11042-020-08976-6.

[17] Zhong-Qiu Zhao et al. "Object Detection With Deep Learning: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* PP (Jan. 2019), pp. 1–21. DOI: 10.1109/TNNLS.2018.2876865.

[18] Zhong-Qiu Zhao et al. "Object Detection with Deep Learning: A Review". In: *CoRR* abs/1807.05511 (2018). URL: http://arxiv.org/abs/1807.05511.