C5: Visual Recognition
Report on Object Detection and Segmentation

Team 1: Carles Pregonas Graells, Pau Vallespí Monclús, Carlos Boned Riera, Marc Pérez Sabater

***Abstract*— Object detection and Segmentation are two of the most active topics in the field of computer vision. From 2012 on, this problem is addressed using Deep Learning architectures such as Faster R-CNN, MaskR-CNN, or YOLO, among others. In this paper, we study the behavior of different configurations of the models Faster R-CNN and Mask R-CNN using the Detectron2 framework. We show a qualitative and quantitative evaluation of the behavior of these models pre-training them in KITTI-MOTS and MOTSChallenge datasets. On the one hand, we observe a significant improvement in performance after fine-tuning these models on the datasets of interest and optimizing hyperparameters. On the other hand, we run inferences in unusual situations using out-of-context datasets and present interesting results that help us understand better the networks.**

I. Introduction

In the beginning of this course, we started with an exploration of PyTorch, a robust deep learning framework widely acclaimed for its versatility and performance, and will compare it with Keras.

Moving forward, we'll dive into the realm of object detection, recognition, and segmentation. Armed with tools like Detectron2 [1], equipped with cutting-edge models such as Mask R-CNN[2] and Faster R-CNN[3], we'll unravel the methodologies employed to pinpoint objects within images, discern their identities, and delineate their contours with precision. After an initial inference in this models, we will perform fine tuning on the models parameters in order to find the best configuration for our data. Alongside, we'll explore the utilization of YOLO[4], an alternative framework renowned for its efficiency and simplicity in object detection tasks.

Transitioning to the realm of image retrieval, we'll peer into the mechanisms underpinning content-based image search engines. Here, sophisticated algorithms analyze image content to retrieve relevant matches from extensive databases, facilitating seamless access to visual information.

Lastly, we'll turn our attention to cross-modal retrieval—a convergence of computer vision and natural language processing. By bridging different data modalities, such as images and text, we'll explore how meaningful cross-references and information retrieval are facilitated, paving the way for enhanced understanding and utilization of multimodal data sources.

Throughout this journey, we aim to provide a comprehensive understanding of visual recognition techniques, underpinned by the utilization of sophisticated tools and methodologies. By interpreting these diverse facets, we seek to underscore the significance of visual recognition across various domains and applications.

II. Related Work

In this section, we'll delve into the narrative of object detection, exploring various perspectives such as vanilla detectors and the evolution of more sophisticated methods.

A. Vanilla Object Detectors

The transition in object detection, led by deep learning, contrasts with the pioneering era of the 1990s, where computer vision relied heavily on handcrafted features due to limited image representation. Viola-Jones Detectors in 2001 [5], [6] achieved real-time human face detection, outperforming contemporaneous algorithms through techniques like "integral image," "feature selection," and "detection cascades." Subsequent innovations, like Histogram of Oriented Gradients (HOG) [7], significantly advanced pedestrian detection. Another milestone was the introduction of Deformable Part-based Models [8], [9] in 2008, which revolutionized object detection by decomposing it into distinct parts.

In the realm of deep learning, object detectors are categorized into two main groups: *two-stage detectors* and *one-stage detectors*. The former approaches detection as a "coarse-to-fine" process, while the latter aims to complete detection in a single step.

B. Two-stage detectors

R-CNN [10] uses selective search [11]. It begins by generating a collection of 2000 object proposals (candidate boxes). These proposals are then resized to a standardized image size and passed through a pre-trained CNN model like AlexNet, trained on ImageNet, to extract features and then an SVM is used to classify them. Finally, a linear regression model is trained to generate bounding boxes for each identified object. R-CNN takes around 47 seconds for each image (inference), and the training stage is expensive and slow, as it extracts features with a CNN from 2000 regions per image.

Earlier CNN models, such as AlexNet, necessitated a fixed-size input, typically requiring images to be resized to dimensions like 224x224 pixels. A first solution to solve this problem comes from **Spatial Pyramid Pooling** [12] (**SPP**), whose goal is to get fixed-length representations for variable-size feature maps. In this case, a CNN is run just once per image to obtain a feature map, and then a (variable size) window related to the region proposals to detect the objects in the image.

This concept is further developed in **Fast R-CNN**[13], which employs RoI pooling, akin to SPP. The RoI pooling layer adjusts the region proposals to match the CNN input

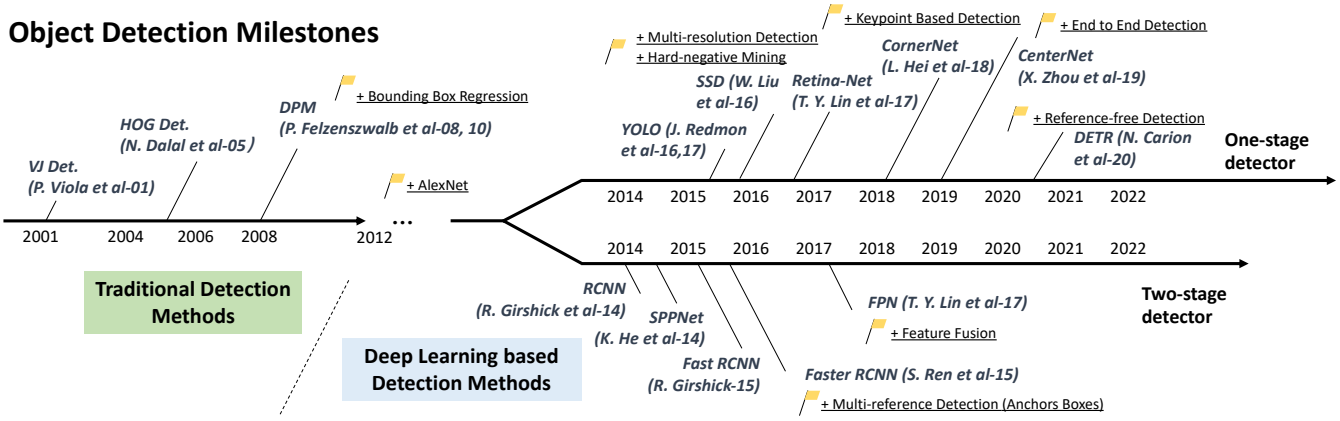

Fig. 1: Chronology of state-of-the-art object detection and instance segmentation architectures.

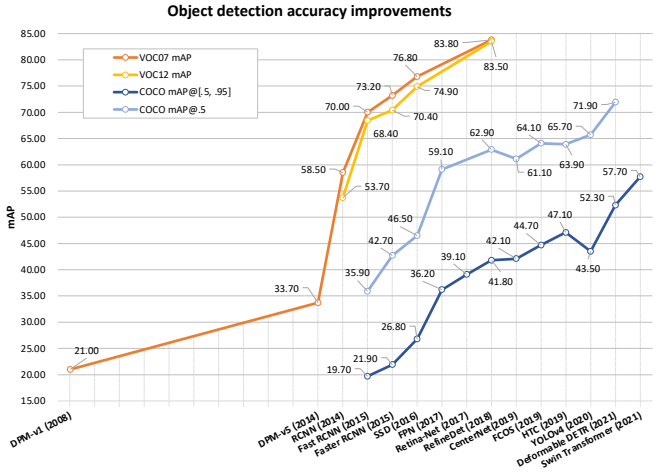

Fig. 2: Accuracy improvement in object detection across VOC07, VOC12, and MS-COCO datasets.

size. Each region is forwarded to a Fully Connected Network (FCN), where a softmax layer and a linear regression layer produce class predictions and bounding box coordinates. Fast R-CNN utilizes a combined loss function. Fast R-CNN addresses two primary challenges of R-CNN: it reduces the number of regions passed to a CNN from 2000 to one per image, and it merges feature extraction, classification, and bounding box generation into a single model. However, a new bottleneck emerges with the selective search algorithm used to locate Regions of Interest (RoIs), which is slow and time-consuming, resulting in an inference time of approximately 2 seconds per image.

To solve this problem, **Faster R-CNN** [14] introduces Region Proposal Networks (RPN): using a sliding window over the feature maps from a CNN, it generates 9 anchor boxes of different shapes and sizes at each sliding position. For each anchor, RPN predicts two things: the probability that an anchor is an object (class-agnostic), and the bounding box regressor for adjusting the anchors to better fit the object.

In Faster R-CNN, the inference time is around 0.2 seconds per image, which is much faster than R-CNN and Fast R-CNN, but is still far from real-time object detection.

C. Object instance segmentation

Moving towards segmentation, Mask R-CNN [15] builds upon Faster R-CNN by incorporating a parallel branch dedicated to predicting object masks alongside the existing branch for bounding box detection. This model applies segmentation to the Region Proposal Network (RPN) predictions, resulting in the generation of high-quality segmentation masks for each instance.

D. One-stage detectors

Instead of using region proposals, one-stage detectors make predictions by only looking into the input image once, which can result in a faster performance. One of the state-of-the-art single pass detectors is **YOLO** [16] (and its successors [17], [18], [19]) and so on upon YOLO9, which idea is to extract a feature map using a CNN called Darknet, divide it into SxS cells (YOLO uses $S = 7$), and predict one bounding box for each cell. YOLO predicts the class of that bounding box and if it is centered at that cell, and uses Non-Maximum Suppression (NMS) to reduce the number of output bounding boxes.

Another well-known one-stage detector is **SSD** [20], which makes predictions at multiple feature maps (YOLO only does it for one) using a VGG16 network [21] as a feature extractor. The idea is that each feature map, which has different local receptive fields, specializes in objects of different sizes.

The NMS method is also used at the end of the SSD model, and Hard Negative Mining (HNM) is then used to further reduce the number of predicted negative boxes.

Finally, **RetinaNet** [22] has been formed by making two improvements over existing single stage detectors: Feature Pyramid Networks (FPN) [23] and Focal Loss. FPN replaces the feature extractor of detectors like Faster R-CNN, and focal loss is introduced to handle the class imbalance problem with one-stage object detection models.

The most recent work is **DETR** [24]. Unlike traditional computer vision techniques, DETR approaches object detection as a direct set prediction problem. It consists of a set-based global loss, which forces unique predictions

via bipartite matching, and a Transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. Due to this parallel nature, DETR is very fast and efficient.

A comparison between some one-stage and two-stage detectors on different datasets such as the COCO dataset [25] is shown in Fig. 2.

III. Methodology

In this section, we describe the methodology employed for the experimentation, including model selection, dataset preparation, training procedure, and evaluation metrics.

Detectron2

We implemented the models using the Detectron2 framework. It includes a model zoo of state-of-the-art object detection and instance segmentation algorithms such as Faster R-CNN, RetinaNet and Mask R-CNN. Each model configuration has four parts:

- Backbone: ResNet (R) or ResNext (X)
- Number of layers: 50 or 101
- Backbone combination: ResNet + Feature Pyramid Network (FPN), ResNet conv4 backbone with conv5 head (C4) or ResNet conv5 with dilations (DC5)
- Learning Rate (LR) Scheduler: 1x or 3x

For example, Mask R-CNN R_50_FPN_3x uses ResNet with 50 layers as backbone, a FPN and a 3x LR scheduler.

In our case we choose between Faster R-CNN & Mask R-CNN.

Models' Methodology

We fine-tunned Faster R-CNN and Mask R-CNN models for object detection and segmentation tasks. We used pre-trained weights obtained from the MSCOCO2017 dataset to initialize the models. For each task we modyfied the labeling of the predefinided COCO classes. The models were then initialized with the pre-trained weights. In Figure 3 we can see a summary of the working flow of both models. The diagram shows schematically how an image is processed until we get the classification with bounding boxes, and the additional masks in the Mask-RCNN case.

Through iterative optimization, the entire model was fine-tuned, gradually unfreezing more layers and adjusting learning rates to learn task-specific features without forgetting pre-trained knowledge. Regularization techniques like dropout or weight decay were applied to prevent overfitting and improve generalization on unseen data.

Finally, we evaluated the trained models using the defined COCOEvaluator, measuring their performance on a validation set. Additionally, we conducted inference on the test set, generating predictions for object instances and their segmentation masks.

Out of Context Tasks

We studied the behaviour of the networks in unusual situations using **Out Of Context (OOC) datasets**. With this purpose, we use a small subset of images with OOC objects from SUN dataset[26], and we modify some specific images to perform experiments. We performed some basic rotations, texture changes and we paste some objects from other sources to the image. Some example can be seen in Fig. [12, 14, 13].

IV. Experimental design

In this section, we outline the experimental methodology employed in our study. We detail dataset selection, training strategies, hyperparameter tuning, and evaluation metrics.

All the experiments have been done with the default optimizer hyperparameters. The finetunne has been performed with the Adam optimizer with a batch size of 16 images per batch and 200 epochs. The experiments has been done over a GPU RTX 3090 with 24G.

A. Dataset

The dataset comprises a collection of images depicting various traffic scenarios from the vantage point of a vehicle driver. It spans a diverse range of environments, including urban streets, inter-urban roadways, and rural routes. Within this dataset, three categories of road users are annotated, along with designated regions that are to be ignored. For the purpose of our investigation, we have chosen to focus exclusively on the classes representing cars and pedestrians.

In our study, we assess the performance of pre-trained models by employing the COCO metrics provided by the Detectron2 framework [1]. To facilitate this evaluation, it was imperative to convert the KITTI-MOTS dataset into the COCO format. The KITTI-MOTS dataset encompasses two distinct annotation formats: PNG and TXT. These formats provide access to crucial data, including detection and segmentation masks, as well as other pertinent annotations necessary for both the evaluation and training phases. Detailed descriptions of these formats are available on the dataset's official documentation. An example of an image with the annotations in the KITTI-MOTS dataset can be seen in Image 4.

An analysis of the dataset reveals a distribution wherein 70% of the annotated instances are vehicles, and the remaining 30% are pedestrians. This disparity in representation suggests a potential bias where models might demonstrate enhanced detection capabilities for vehicles as compared to pedestrians, due to the more abundant exemplars present in the dataset.

B. Train-Val-Test Split

We utilize the official validation set of KITTI Multiple Object Tracking and Segmentation (KITTI MOTS) as our test set. For training and validation, we employ the training sets of KITTI MOTS consisting of 12 sequences and the MOTSChallenge consisting of 4 sequences. Detailed statistics are shown in Tab. I :

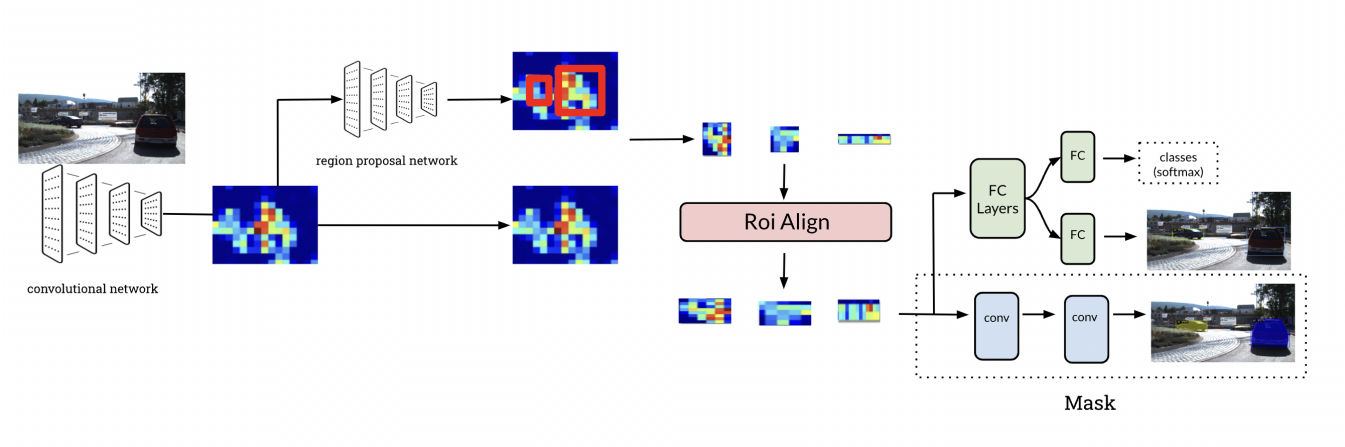

Fig. 3: Diagram for Faster R-CNN and Mask R-CNN extension

Fig. 4: Example image with annotations within the KITTI-MOTS dataset. There are some cars annotated with the confidence (in %) being each one painted with a different color.

	KITTI MOTS train	KITTI MOTS test	MOTSChallenge train
# Sequences	12	9	4
# Frames	5,027	2,981	2,862
# Masks Pedestrian			
Total	8,073	3,347	26,894
Manually annotated	1,312	647	3,930
# Masks Car			
Total	18,831	8,068	-
Manually annotated	1,509	593	-

TABLE I: Statistics of the modified split of KITTI MOTS and MOTSChallenge Datasets

To assess our model's performance, we employ a 4-fold cross-validation strategy. Each fold comprises 3 sequences from KITTI MOTS and 1 sequence from MOTSChallenge. In each experiment, we train our model using 3 folds and validate it on the remaining fold, iterating through all 4 possible combinations.

C. Metrics

Our evaluation of object detection models hinges on the Average Precision (AP) metric, which is computed subsequent to establishing a cut-off point for the Intersection over Union (IoU) ratio.

The IoU quantifies the extent of overlap between two bounding boxes. Specifically, within the context of object detection, it assesses the degree of correspondence between the predicted bounding box and the actual, or ground truth, bounding box. To this end, the IoU is determined for every prediction, and a predetermined threshold is applied to categorize the prediction as either a true positive or a false positive.

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} \tag{1}$$

For AP calculation, we construct the precision-recall graph based on a chosen IoU threshold. Although the formal definition of AP is the integral of this curve, it is often approximated for practicality. Our methodology adopts the COCO evaluator's approximation approach, which entails interpolating the precision across 101 evenly spaced recall levels after enforcing a monotonic decrease in precision to the right. AP can be computed for various IoU levels or across different scales of bounding boxes. The aggregate AP for the COCO dataset is determined by taking the mean of the APs calculated at IoU thresholds ranging from 0.5 to 0.95 in increments of 0.05.

V. Results and Discussion

In this section, we show the results of all the experiments that have been done. The results will be compared quantitatively and qualitatively.

A. Quantitative Results

In Tab. II the best AP is achieved with a threshold of 0.5. The detection is better in medium and large objects, while small ones (far away from the camera) are more difficult to detect correctly as can be seen in the table. The model can detect cars better than pedestrians, maybe because cars are more distinctive and easier to visualize than pedestrians. Additionally, inference computation time has been calculated for the inference of 11096 frames for the two models, resulting in an average time per image of 0.07s with Faster

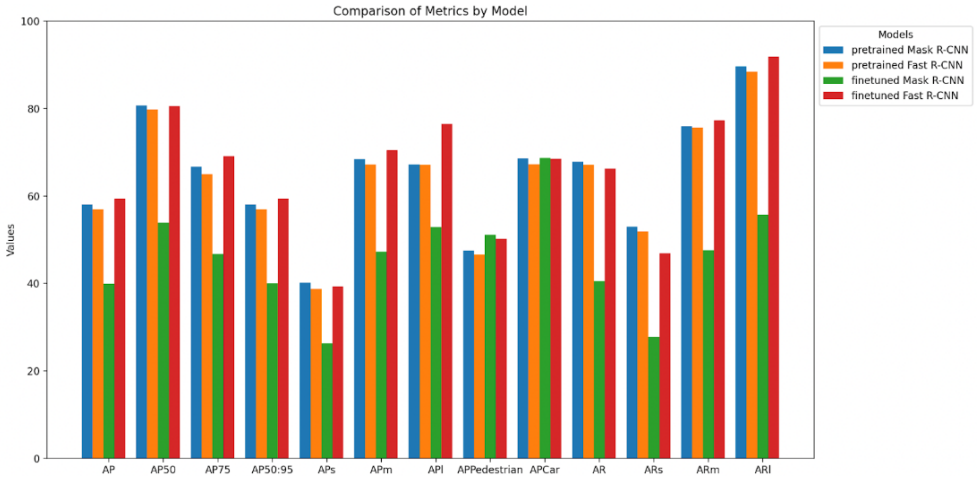

Fig. 5: Comparison among all the methods during Finetuning & Inference

R-CNN against 0.11s using Mask R-CNN. The difference of time for a single image is almost imperceptible, but at a large scale that means that Faster R-CNN (as the name indicates) is around 63.64% times faster than Mask R-CNN.

Model	AP	AP50	AP75	AP50:95	APs	APm	APl	AP Pedestrian	AP Car
Inference results with off-the-shelf methods									
Mask R-CNN	0.529	0.803	0.585	0.529	0.325	0.632	0.720	0.385	0.672
Faster R-CNN	0.569	0.798	0.650	0.570	0.387	0.672	0.671	0.466	0.673
Fine-tuned results with off-the-shelf methods									
Mask R-CNN	0.397	0.539	0.467	0.400	0.264	0.472	0.529	0.511	0.6873
Faster R-CNN	0.594	0.806	0.691	0.594	0.393	0.705	0.764	0.523	0.685

TABLE II: Comparison of inference and fine-tuned results using off-the-shelf methods

Also in the Tab. II it can be seen that after finetuning the models the AP decreases but in the classes that we are finetuning, which are pedestrians and cars. Faster R-CNN slightly performs better than Masker R-CNN. In the Fig. 5 there is a comparison bar plot with all the metrics we track. This figure also ratifies that Faster R-CNN performs better than Masker R-CNN.

B. Qualitative Results

For inference, in Faster R-CNN and Mask R-CNN, we have first applied a threshold of 0.5 to detect the predicted element as correct. We have seen many cases in which this low threshold misclassified some objects. For example, in the Fig. 6, the model misclassified the construction signal, predicting it as a person with a 0.5 of confidence. The effect of a low threshold can also be seen in Fig. 7 applying Mask R-CNN, where, even detecting and segmenting correctly the bicycle, the model also predicted the rear wheel of the bicycle as a bicycle.

In all of these cases, the problem is that it has been put a very permissive threshold, However, there exists also the opposite cases in which a correct prediction is found with a low threshold. In the Fig. 8 left image, both elements are bicycles and it detects a motorcycle with a 0.8 of confidence (incorrect) and a half a bicycle with a 0.5 of confidence (correct). In the right image, it detects correctly the bicycle

Fig. 6: Faster R-CNN inference example

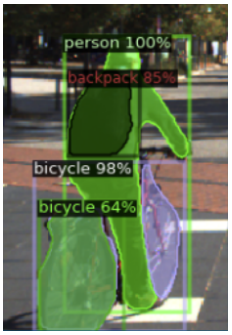

Fig. 7: Mask R-CNN inference example

with less than 0.7 confidence. With a higher threshold, the false positive rate is lower but the model left some correct detections that aren't visible but detectable. The fact of selecting the threshold is an important hyperparameter in which the goal of the task is important to make the model more restrictive or not.

Generally, both models, Faster R-CNN and Mask R-CNN detect with better precision cars and persons, while they fail more when detecting more unusual objects on the road. It can bee seen in the Fig. 9 that all the cars and traffic lights are correctly detected, but the operator's suit and cone on are classified as fire hydrant.

If we compare both models in the same images Fig. 10, Mask R-CNN shows slightly better qualitative results than

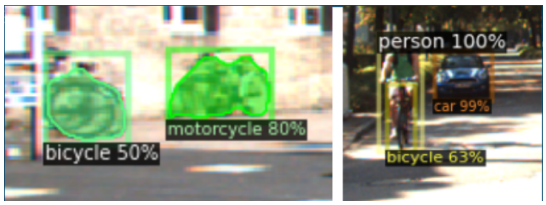

Fig. 8: Mask R-CNN and Faster R-CNN inference example

Fig. 9: Faster R-CNN inference example

Faster R-CNN. However, in most cases, they misclassify similar objects.

Fig. 10: Comparation between Faster R-CNN and Mask R-CNN

When finetunning the models Mask R-CNN worsened its performance when it was fine-tuned on the KITTI-MOTS dataset than the pre-trained model without fine-tuning. In Fig. 11 the segmentation task works a little bit worse than the only detection task. In the left image the pedestrian inside the red square is detected by Faster R-CNN but not by Mask R-CNN.

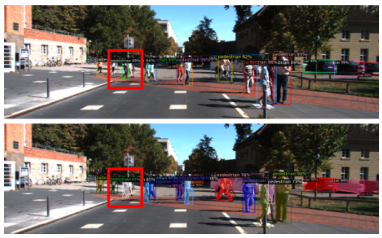

Fig. 11: Comparation between Faster R-CNN and Mask R-CNN when finetunning

C. Out of Context Tasks

Focusing on the qualitative results, our experiments aimed to prove the object detection model's adaptability to varied environmental conditions. In Fig. 14 the context of the image proves to be important. In the right of the image, the model struggles to predict the boat accurately, but altering the context enables it to make the correct prediction.

We also aimed to explore how the surroundings might influence the classification outcome. In this case, we see that features extracted from pixels unrelated to the actual object (a car) can impact the assigned class. In Fig. 14, where clouds are mistaken for snow, the model erroneously identifies the car as a snowboard.

Additionally, in Fig. 15 we have observed that the color shows to be closely associated with the classification, such as when a red car positioned vertically is misclassified as a fire hydrant. We haven't determined whether this is due to color or if it's influenced by the presence of clouds, perhaps resembling smog.

Fig. 12: Example of an augmented image in which a car is miss-classified as a snowboard.

Fig. 13: True Elefant in the Room

Fig. 14: Example of the classification of a boat which not only depends on the boat but more importantly in the context, in this case, the water.

VI. Conclusions

To conclude, the paper underscores the importance of considering both quantitative metrics and qualitative assessments when evaluating object detection and segmentation models. While Faster R-CNN excels in certain aspects such as speed and quantitative metrics, Mask R-CNN demonstrates superior qualitative performance. Additionally, the

Fig. 15: Car being misclassified as a fire hydrant

experiments highlight the critical role of contextual signals in model decision-making and the potential impact of biases introduced by out-of-context tasks. Moreover, through fine-tuning on relevant datasets, substantial improvements in model performance are achievable. Overall, these findings contribute to a deeper understanding of the capabilities and limitations of state-of-the-art object detection and segmentation models, facilitating further advancements in computer vision research and applications.

REFERENCES

[1] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," *arXiv preprint arXiv:1911.02549*, 2019.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv preprint arXiv:1703.06870*, 2017.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, pp. I–I, Ieee, 2001.

[6] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, pp. 137–154, 2004.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, Ieee, 2005.

[8] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8, Ieee, 2008.

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[11] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, pp. 154–171, 09 2013.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[13] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 06 2015.

[15] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[17] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 04 2018.

[19] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, "Yolov4: Optimal speed and accuracy of object detection," 04 2020.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, "Ssd: Single shot multibox detector," vol. 9905, pp. 21–37, 10 2016.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[22] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.

[23] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[24] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, "Microsoft coco: Common objects in context," 05 2014.

[26] M. J. Choi, A. Torralba, and A. S. Willsky, "A tree-based context model for object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 240–252, 2012.