Object detection in a cluttered kitchen setting

Julia Kuhn
e11710994@student.tuwien.ac.at

Bernhard Pandion
e11730467@student.tuwien.ac.at

Harald Straßgürtl
e11731708@student.tuwien.ac.at

***Abstract*—In order to further develop robotic systems that can support people in their daily lives, reliable object recognition is required. It is particularly difficult for robots to recognise objects in cluttered environments. For this reason, this paper compares three modern object recognition methods (YOLOv9, Detectron2 and Detic) and two multimodal large-scale language models (LLaVA and GPT-4o) in a cluttered kitchen setting. The experiments carried out show that the methods analysed provide useful results, especially for images with low clutter and good lighting. In case of the language models, the output is highly dependent on the prompts used as input. Therefore the standard object detection methods are preferred, when zero-shot detection is not needed. Out of the three standard object detection methods, YOLOv9 showed the best performance.**

I. Introduction

The fields of robotics and object recognition have witnessed significant growth in recent years. Advances in robotics have led to the development of robots that can assist humans with everyday tasks. A notable example is the Human Support Robot (HSR) from Toyota [1], which is equipped with cameras and a gripper to perceive and interact with its environment. A potential application for the HSR is tidying up a kitchen, which involves several key steps: locating, classifying and manipulating objects. This paper focuses specifically on the object recognition component of this process.

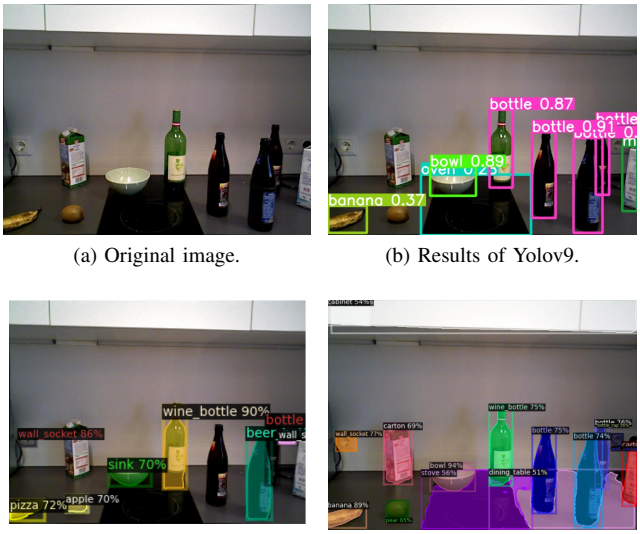
While humans can effortlessly locate and classify objects in cluttered environments, robots face considerable challenges when performing such a task. However, recent advancements in these areas hold promise for overcoming these challenges.

The aim of this research is to evaluate the performance of state-of-the-art object detection methods in kitchen environments. By constructing a representative dataset featuring various types of clutter and lighting conditions and conducting a comparative analysis, we seek to identify the most effective object recognition method for use in cluttered kitchen settings. Fig. 1 shows an example of such a kitchen environment and the detected objects using different methods.

The following work is structured as follows. Section II gives an overview of object detection models. Sections III and IV explain the datasets and methods used. The experiments and results are presented in section V and section VI summarises the work and provides an outlook for future research topics.

II. Related work

This section provides an overview of state-of-the-art object detection methods and multimodal large-scale language models (MLLMs) that are capable of processing images.



(a) Original image. (b) Results of Yolov9.

(c) Results of Detectron2. (d) Results of Detic.

Fig. 1: Example for a cluttered kitchen environment and of the detected objects using different methods.

A. YOLO

You only look once (YOLO) was first introduced in 2016 by J. Redmon et al. in their Paper "You Only Look Once: Unified, Real-Time Object Detection" [2]. Over the years many versions of YOLO were created to further improve the performance. It falls in the category of one stage detectors and uses convolutional neural networks (CNN) to generate its class predictions. As a one stage CNN detector YOLO divides an image into multiple grid cells and predicts for each of these cells the probability of any object class being there. Because it processes the image in one go, it is well known for its high computation and processing speed. Real time object detection is therefore the main application for YOLO [3], [4]. Although it is known for its speed, it has also high overall accuracy. The limitations of YOLO are mainly a failure to detect either objects which are close together, or objects of smaller scale. PASCAL VOC [5] was used for benchmarks in the early versions of YOLO, however since YOLOv3 all versions use the MS COCO dataset. The ninth version of YOLO, which was released in February 2024 and was used in this paper showed improvements in the network architecture and conquered information bottleneck problems by introducing the concept of programmable gradient information (PGI) [6]. It contains models for object-detection, -segmentation, panoptic

segmentation and image captioning. During the work on this paper, another version, YOLOv10 [7], was released, which shows improvements in performance-efficiency by reducing computational redundancies and introducing a model which is driven by both efficiency and accuracy. With these improvements the Authors lean further into the real-time object detection use-case of YOLO.

B. Detectron2

Detectron2 [8] is a library consisting of state-of-the-art algorithms for object detection, semantic segmentation, instance segmentation and panoptical segmentation. It was released by Facebook AI Research in 2019 and is a revised version of Detectron [9], which originates from the maskrcnn-benchmark [10]. Compared to its predecessors, Detectron2 has several advantages. It is faster than the original Detectron for a large number of standard models and has been implemented in PyTorch, which makes it easier to program than the original implementation in Caffe2. Furthermore, it has a modular design, which makes it more flexible and easy to extend and it supports new datasets such as LVIS [11] in addition to the COCO dataset [12]. Because of that it has been used in different applications such as [13]–[18].

In this paper we want to focus on object detection and instance segmentation for which we chose the models *Faster R-CNN* [19] and *Mask R-CNN* [20]. These models consist of a *backbone* which can be implemented as feature pyramid network (FPN) [21]. This network uses a single-scale image as input and generates proportionally sized RoI features from different levels of the feature pyramid as outputs. For *Faster R-CNN* the *head* generates the predictions of the class labels and bounding boxes using a two stage process. In case of *Mask R-CNN* these two outputs are extended by an additional output containing the object mask. The first stage of the *head* consists of a region proposal network (RPN) as described in [19] which proposes candidates for the object bounding boxes. The second stage contains the *Faster R-CNN* or *Mask R-CNN* algorithm itself, which predicts the classes, box outputs and (in case of *Mask R-CNN*) binary masks for each RoI.

C. Detic

By building on the foundation of Detectron2 [8], Facebook Research published in 2022 the object detector Detic [22], short for Detector with Image Classes. The two major differences are the usage of a method called WSOD (Weakly-Supervised Object Detection) [23] and CLIP (Contrastive Language-Image Pre-training) embedding vectors [24] as classifiers. With the integration of the semi-supervised WSOD method, Detic can use image-level annotations instead of bounding boxes, otherwise it uses the standard two-stage detector [25]. Therefore new datasets for training Detic can be created more efficiently, because image-level annotations are less time-consuming than making bounding boxes for every object in the images. Additionally datasets like ImageNet-21K [26], where the majority of images are only labeled at image-level, can be used for training. The CLIP embeddings are used to create a semantic connection between text and images, enhancing it's capabilities in recognising rare classes and zero-shot detection, which means detecting new classes absent in the training dataset. These adjustments result in the separation of finding the object in the image (localisation) and determining the object class (classification). Additionally Detic supports multiple different class vocabularies, for example LVIS [11] and COCO [12], but it is also possible to use custom vocabularies. There are multiple models of Detic which were trained on different datasets like ImageNet-21k, LVIS and COCO. Some of these models are also usable in real-time with varying degrees of accuracy and runtime.

D. Generative Pretrained Transformer (GPT-4)

Dedicated object detection frameworks are not the only way to detect objects within an image. MLLMs are language models which can process not only text, but also other forms of data, like images or audio in parallel to text-prompts.

OpenAI's LLMs found widespread use in the recent years even outside the machine learning community, because of their accessibility and media coverage. GPT-4V and GPT-4o, are both recent models from OpenAI, that, due to their vision capabilities, fall within the category of MLLMs [27]. Because of the good performance of their models, we wanted to include one of them in the comparison of this paper. GPT-4o was therefore selected, as it is the latest and most accurate model.

E. Large Language and Vision Assistant (LLaVA)

LLaVA is a open source MLLM, which connects traditional language based LLM capabilities with vision via an visual encoder. It uses a similar approach to Detic, utilising CLIP in order to generate language embeddings that can be processed by the Vicuna large language model. The most recent version of LLaVA is llava-v1.6-34b which was used in the experiments of this paper [28]–[30].

III. Dataset Description

To test these methods, the KitchenCOCOv2 [31] and KitchenLVIS [32] datasets were created. In this section, the structure of the datasets will be explained.

In order to create the datasets, 180 photos of a kitchen with different amounts of objects were taken from different views with the help of the human support robot *Sasha* from TU Vienna [1]. These photos were then combined with the images from the KitchenCOCO dataset [33], leading to a total of 285 images. The images were then labelled using the software *Roboflow* [34] according to the classes of the COCO [12] and LVIS [11] datasets, resulting in the corresponding datasets KitchenCOCOv2 and KitchenLVIS. KitchenCOCOv2 was split into two versions, where one contains all images and the other one contains 101 pictures with low clutter. The KitchenLVIS was categorised into low (100 images), medium (149 images) and high (36 images) clutter depending on the number of objects and into bright (159 images) and dark (126 images) depending on the exposure of light.

Fig. 2: Example images of the created datasets.

IV. Methods for object detection and training

With the developed test-datasets the object detection methods YOLOv9, Detectron2 and Detic and the language based models LLaVA and GPT-4o will be evaluated and compared. This section explains the setups used for the algorithms.

A. YOLOv9

1) Setup: With the release of YOLOv9, Wong-Kin Yiu, who implemented the paper [6], released 12 weights [35], which were pre-trained on the MS-COCO dataset. Out of these 12 weights, the weights for the two most accurate models (see TABLE I), YOLOv9-C and YOLOv9-E, were used. In addition to the standard C and E model weights, which use the novel Generalized Efficient Layer Aggregation Network (GELAN) of YOLOv9 in combination with PGI, we also evaluated the performance C and E weights without PGI (GELAN-C, GELAN-E) and the performance of the model without the auxiliary branch (C-converted, E-converted). For good comparability, all detection and validations used a confidence threshold of 0.001 and a non maximum suppression IoU threshold setting of 0.5. The inference-size was set to 640 as the KitchenCOCO and KitchenLVIS pictures were of the 640x480 px format. A minimum number of items was not set.

Model	AP^{val}	$AP_{50}{}^{val}$	$AP_{75}{}^{val}$	Param.	FLOPs
YOLOv9-T	38.3%	53.1%	41.3%	2.0M	7.7G
YOLOv9-S	46.8%	63.4%	50.7%	7.1M	26.4G
YOLOv9-M	51.4%	68.1%	56.1%	20.0M	76.3G
YOLOv9-C	53.0%	70.2%	57.8%	25.3M	102.1G
YOLOv9-E	55.6%	72.8%	60.6%	57.3M	189.0G

TABLE I: Performance of YOLOv9 models trained on the MS-COCO dataset with a test size of 640 [35].

2) Training on LVIS dataset: Because YOLOv9 has a pretrained model for the COCO dataset but not for the LVIS dataset, the training process on the LVIS dataset was carried out by ourselves. Two different approaches were realised. For the first approach, no prior weights were defined at the start of the training process. The second approach uses a method called transfer learning, where an existing trained model is reused. Because of the fact, that the training and validation set of COCO and LVIS are identical, the pretrained COCO model is well suited for this purpose. By freezing the backbone, which is responsible for the feature extraction, the training process for LVIS is realised on the basis of the pretrained COCO weights.

	name	short name	box AP	mask AP	model id
COCO Faster R-CNN	R50-FPN	CF50	40.2	-	137849458
	R101-FPN	CF101	42.0	-	137851257
COCO Mask R-CNN	R50-FPN	CM50	41.0	37.2	137849600
	R101-FPN	CM101	42.9	38.6	139653917
LVIS Mask R-CNN	R50-FPN	LM50	23.6	24.4	144219072
	R101-FPN	LM101	25.6	25.9	144219035

TABLE II: Weights used for evaluation of Detectron2 [36].

B. Detectron2

For Detectron2, different weights, which depend on the used baselines and training-datasets, can be chosen. In case of the viewing models *Faster R-CNN* and *Mask R-CNN* three different backbone combinations (FPN, C4 and DC5) are provided for the COCO dataset. For the LVIS dataset only the FPN backbone is available, which is the reason why we chose this backbone. It consists of a combination of ResNet-50 or ResNet-101 and FPN and obtains the best speed/accuracy tradeoff [36]. In total six different weights were used for the evaluation, depending on the task and dataset, which are summarised in TABLE II. For the following comparisons the *short names* are used to differentiate the weights. For the other parameters, the default configurations were used, which lead to efficient and sufficiently general object recognition [13].

C. Detic

Detic released multiple models, where the main differences between them are different detectors, backbones and training datasets. As the detector either CenterNet2 [37] or Deformable DETR [38] is utilised. The two different backbones are either the Swin-B [39] or a ResNet50 [40] backbone. The models with the combination of the CenterNet2 detector and the Swin-B backbone were used because they have good box mAP values on multiple benchmark datasets including LVIS and COCO [41]. The difference between the models from Table III with the given short names D1, D2 and D3 are the resulting weights due to training on different datasets and class vocabulary. The first two models D1 and D2 are trained on the full ImageNet-21K dataset. The D2 model is additionally trained on a combined LVIS-COCO dataset, where the COCO classes are mapped to the according LVIS classes via synsets. The D3 model is only trained on the overlapping classes of the

Name	Short Name	Trained Dataset
C2_SwinB_IN-21K	D1	ImageNet-21K
C2_SwinB_IN-21K+COCO	D2	ImageNet-21K, LVIS+COCO
C2_SwinB	D3	ImageNet-21K, LVIS Overlap

TABLE III: Weights used for evaluation of Detic [41].

LVIS and ImageNet-21K datasets. The score and Intersection over Union (IoU) threshold is set to 0.5 for the evaluations. In the further sections, the models will be referred to by their short names.

D. LLaVA and GPT4

To evaluate the vision capabilities of LLaVA and GPT-4o, the selection of a fitting prompt is essential for a proper result. Because of that, the following four prompts were tested for both, the COCO and the LVIS dataset classes:

1) *Which and how many objects of the COCO/LVIS dataset can you see in the image?*
2) *Object detection + COCO/LVIS dataset + count objects*
3) *You have the task to detect all objects in the image and label them based on the classes of the COCO/LVIS dataset.*
4) *You are an engineer labeling pictures for a kitchen database with the classes of the COCO/LVIS dataset. Give me a list of objects you can see in the image using only classes of the COCO/LVIS dataset. If a object class appears multiple times in the image, add the number of appearances in front of the object name. If you cannot recognise an object, give your best guess. The list should be in a single string format, like "1 X, 3 Y, 2 Z...". Only output the object list, nothing else.*

Because we evaluated each of the MLLMs with only 32 combinations of prompts and images for each dataset, we can not provide fully statistically relevant data, but we can provide estimates of the statistical behaviour of the MLLMs.

V. Experiments & Results

This section shows the results of the tests which were carried out. In the first part, the YOLOv9 training approaches and afterwards the different methods are compared.

A. Comparison of the training approaches for YOLOv9

The resulting mAP50 in dependence of the epoch for the YOLOv9 training with and without transfer learning is illustrated in Fig. 3. The mAP50 for the training without prior weights rises significantly, but shows linear behaviour after around 25 epochs. The curve for the transfer learning approach starts already at a mAP50 of 0.1 and rises steeply at the beginning, but flattens at around 25 epochs. It takes about 175 epochs until the mAP50 of the training without transfer learning overtakes the one with transfer learning.

B. Evaluation of the detection methods

After training, each method was tested. The resulting mAP50 values are shown in TABLE IV. To illustrate the differences between the methods, some examples of the detections

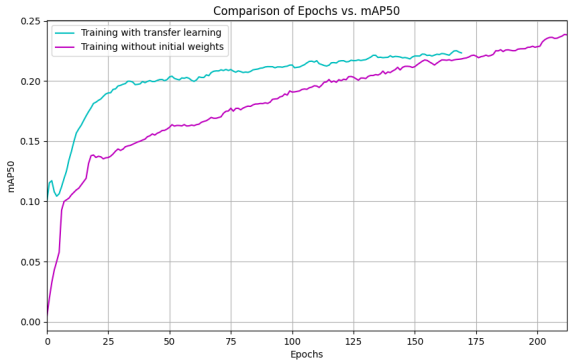


Fig. 3: MAP50 of the YOLOv9 training on the LVIS dataset with and without transfer learning from COCO dataset.

are shown in Fig. 4. Before we compare the methods, we would first like to take a closer look at the individual results.

1) Yolov9: TABLE IV shows the mAP50 performance of the self trained and pretrained weights [35]. Initially, the C-weight, E-weight and their GELAN counterparts were tested on the standard COCO dataset. Afterwards the weights were tested on the full and low clutter KitchenCOCOv2 dataset. Over the full KitchenCOCOv2 dataset, all pretrained COCO weights showed good performance. The C-weight reached 64% of its COCO mAP50, while the E-weight managed 77% of its COCO mAP50 in spite of the cluttered pictures. Interestingly, in both cases the GELAN counterpart performed slightly better than the standard weight. Reaching a 7% higher mAP50 value as it achieved on COCO, the E-weight performed rather well on the low clutter KitchenCOCOv2 split. The C-weight, on the other hand, showed only 83% of its COCO mAP50. Interestingly, inside of the KitchenCOCOv2 dataset, both GELAN weights performed minimally better than the normal weights. The performance of the model weights which do not include the auxiliary branch (C-converted, E-converted) are not displayed in TABLE IV, as their results were identical to the results of the standard C- and E-weights.

Regarding the self-trained YOLOv9 weights, no similar performance was achieved as before. The evaluation of both the fully trained weights and the training with the frozen backbone showed, on the LVIS dataset, 34% and 30% of the mAP50 that was previously achieved with the E-weight on the COCO dataset. The same evaluation showed on the KitchenLVIS dataset a slightly better performance of 46% of the COCO mAP50 for the fully trained and 43% of the COCO mAP50 for the frozen-backbone weights. Both the fully trained and the weights with frozen backbone showed a consistent performance over all KitchenLVIS splits, especially regarding different clutter sizes. A surprisingly good robustness against illumination changes was also shown for the self-trained LVIS weights, as the mAP50 decrease between KitchenLVIS-AllBright and KitchenLVIS-AllDark was only -13% for the frozen backbone weights and only -16% for the fully trained weights. The fully trained weights and the

	KitchenCOCOv2			KitchenLVIS						
method	weights	full	low clutter	weights	full	bright	dark	ClutterL	ClutterM	ClutterS
YOLOv9	C	49.9	58.2	full-train	**33.5**	**38.4**	**32.1**	**35.5**	**35.7**	**46.0**
	E	**67.9**	77.8	frozen	31.2	35.1	30.5	30.9	33.6	42.0
	GELAN-C	52.9	60.7	-	-	-	-	-	-	-
	GELAN-E	64.3	**78.2**	-	-	-	-	-	-	-
Detectron2	CF50	29.554	35.725	-	-	-	-	-	-	-
	CF101	28.079	36.770	-	-	-	-	-	-	-
	CM50	28.841	36.190	LM50	**8.885**	10.461	7.361	9.413	**8.761**	14.396
	CM101	**29.766**	**37.213**	LM101	8.631	**10.536**	**7.597**	**9.958**	8.392	**14.801**
Detic	D1	28.450	38.436	D1	20.033	22.675	18.802	21.598	20.402	28.973
	D2	**43.736**	**60.884**	D2	17.572	20.414	16.281	17.957	17.553	27.522
	D3	26.778	39.204	D3	**22.649**	**25.913**	**20.098**	**22.454**	**23.362**	**34.542**

TABLE IV: Comparison of the evaluated mAP50 values, shown in %, using the detection methods Yolov9, Detectron2 and Detic on the KitchenCOCOv2 and KitchenLVIS datasets.

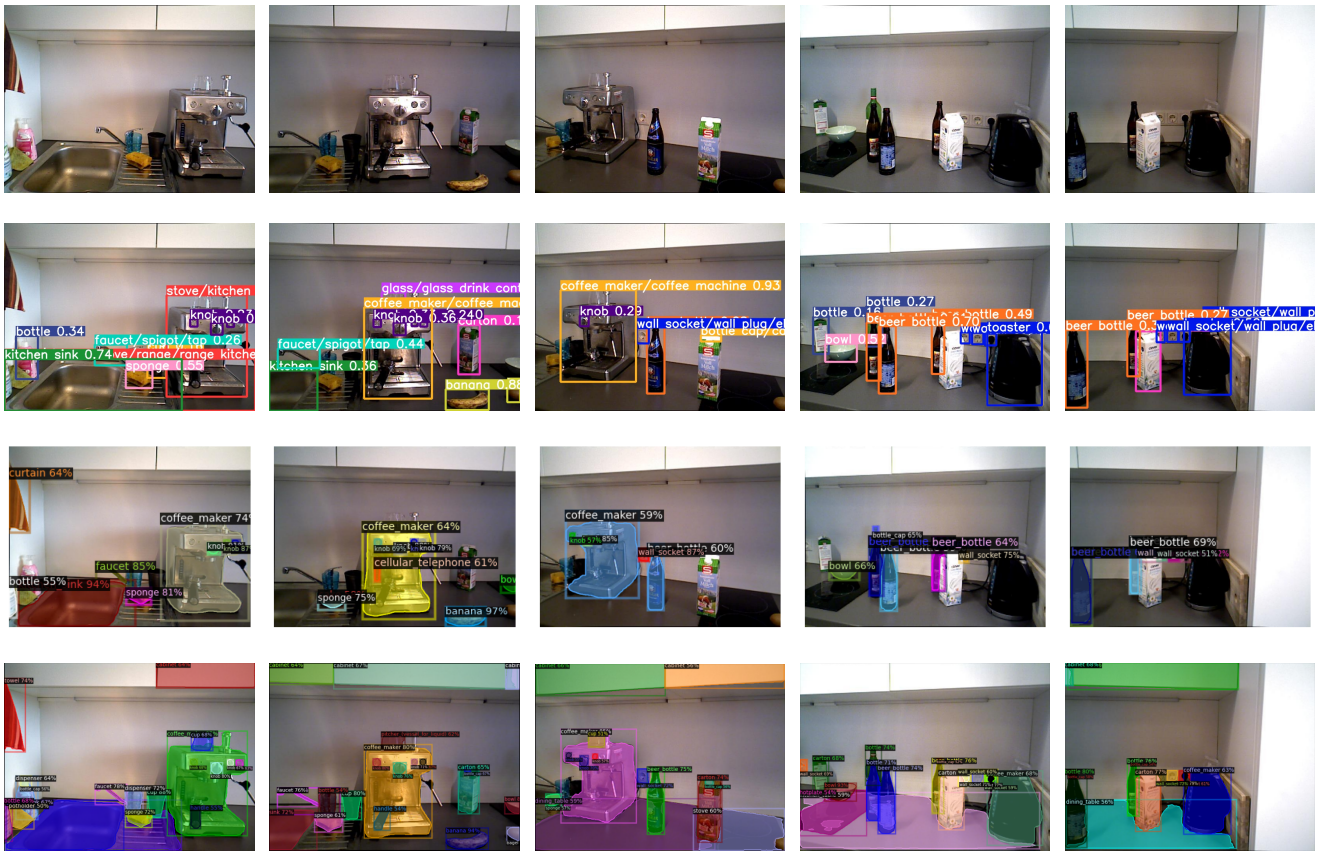


Fig. 4: Overview of some example detections: (a) original images, detected objects using (b) Yolov9 (c) Detectron2 (LM101), (d) Detic and detected classes of the language based models (e) LLaVA (prompt 2) and (f) GPT-4o (prompt 4).

weights with frozen backbone, evaluated on the KitchenLVIS-ClutterS split, showed 59% and 53% of the E-weight mAP50 evaluated on the KitchenCOCOv2-lowClutter split.

2) Detectron2: By comparing the values determined by Detectron2 from TABLE IV, it can be seen that the results for the different weights hardly differ from each other. However, there is a difference depending on whether the entire dataset or only parts of it are used. For example, an mAP50 of 37.213 % is achieved for the low clutter images from KitchenCOCOv2 using the weights *CM101*. For the entire dataset, on the other hand, only an mAP50 of 29.766 % is achieved. This is due to the fact that the images with more clutter contain many objects that cannot be clearly categorised into the limited classes of COCO. In the case of the low clutter images, there are fewer such objects. If you compare the results of KitchenLVIS, it is also noticeable that dark images are recognised more poorly than bright images. For KitchenLVIS, a maximum accuracy (for the ClutterS images) of 14.801 % is achieved, which is also significantly worse than the results of the KitchenCOCOv2 dataset with low clutter. The reason for that are the weights used for evaluation. As shown in TABLE II the weights which are trained on the LVIS dataset have a lower validation-AP value (23.6 %) than the weights which are trained on the COCO dataset (AP over 40 %).

3) Detic: As illustrated in Table IV, D2 performs best across all KitchenCOCOv2 versions, which is expected, since it is the only one trained additionally on the COCO dataset. On the full version, the mAP50 is around 15% better and for the low clutter around 20% better than D1 and D3, showing the effectiveness of the additional COCO training on D2. Even though D1 and D3 were not trained on COCO, their mAP50 values are reasonably good, showing a similar performance with only small differences.
In contrast, D3 has the highest mAP50 values of all the different versions from the KitchenLVIS dataset. Interestingly, even though the model D2 was trained on the combined LVIS-COCO dataset, it is outperformed by D1, which had no training on LVIS at all. This may occur due to the training of this specific model, because the combined LVIS-COCO dataset only contains the 80 classes of the COCO dataset, but mapped to LVIS classes. Therefore the model is not trained well enough to detect the rest of the LVIS classes, which leads to good performance on the KitchenCOCOv2 dataset but not as good on the KitchenLVIS. The mAP50 values for all models of the ClutterL and ClutterM versions are both approximately 10% lower than of the ones of ClutterS. In comparison, the mAP50 for the dark images is only around 5% lower than bright images, which implies a better robustness against different illuminations than against more clutter.
In comparison to the KitchenCOCOv2 results, the KitchenLVIS results are significantly lower regardless of the used model, which may be because COCO classes in general are more generic than LVIS classes and therefore the KitchenLVIS dataset has a lot more annotations.

C. MLLM Comparison

When testing the MLLMs, we noticed that the detection success is even more sensitive to bad illumination, than the dedicated object detection frameworks. For this reason, we only used low clutter and bright images for the evaluation. Some examples of these images and the detected classes are shown in Fig. 4. The experiments conducted with GPT-4o and LLaVA show, that both can correctly detect COCO and LVIS classes in a kitchen setting. However, looking at the data in TABLE V, it seems that GPT-4o gives more reliable results regarding hallucinations. Detection problems, like error messages or repetitions after some correct output, are displayed in TABLE V under ”error“. GPT-4o showed more error messages regarding the object detection process itself, while LLaVA showed repetitions of output. LLaVA shows not only a higher hallucination-percentage over tested prompts and datasets but also a worse F1 value than GPT-4o for prompt 1, 3 and 4. Only in the case of prompt 2 LLaVA performs better than GPT-4o. Regarding output failures LLaVA proves to be more reliable than GPT-4o.

D. Comparison

When comparing the results shown in TABLE IV, YOLOv9 proves to be the best detection method for all datasets. Only for the models with lower parameter count (C & GELAN-C), Detic performs slightly better than YOLOv9 if evaluated on the low clutter images of KitchenCOCOv2. Although, Detic only performs well in those scenarios, it has the advantage that it is agile regarding custom dataset vocabularies. Since, Detic is based on Detectron2, it is expected that Detic performs better than Detectron2. Especially for the KitchenLVIS dataset, Detectron2 has much lower mAP50 values than the other methods. This is because the weights provided for Detectron2 are only baselines weights, which are intended to be trained further.

Comparing the object detection algorithms with the examined MLLMs, shows that they perform decent on bright, low clutter pictures. However, if tested on pictures with bad illumination, we noticed that both GPT-4o and LLaVA decrease quickly in performance.

VI. Conclusion

In this work the object detection methods YOLOv9, Detectron2 and Detic as well as the MLLMs GPT-4o and LLaVA were compared in a cluttered kitchen. For this purpose, the datasets KitchenCOCOv2 and KitchenLVIS were created, which include images with different clutter and illuminations.

The results described in section V show that YOLOv9 performed best in this kitchen setting but needs to be trained for the detection of specific classes. If this training is not possible, Detic would be the better choice, especially for small datasets, because of it’s capability of zero-shot detection.

In this comparison, the standard weights provided by the methods were used for testing and were not specifically trained on kitchen datasets. Because of that, it would be interesting to investigate how the detections improve when the weights

	COCO								LVIS							
value	prompt 1		prompt 2		prompt 3		prompt 4		prompt 1		prompt 2		prompt 3		prompt 4	
in %	GPT4o	LLaVA	GPT4o	LLaVA	GPT4o	LLaVA	GPT4o	LLaVA	GPT4o	LLaVA	GPT4o	LLaVA	GPT4o	LLaVA	GPT4o	LLaVA
P	78.20	49.23	75.00	48.30	80.30	44.39	61.76	53.39	91.62	74.50	93.65	82.96	95.83	69.51	95.09	68.13
R	71.32	82.34	29.53	85.83	53.99	56.32	70.21	55.55	41.10	36.63	35.70	39.90	33.31	33.60	43.48	33.41
F1	74.60	61.62	42.38	61.81	64.57	49.65	65.71	54.45	56.75	49.11	51.69	53.89	49.44	45.30	59.68	44.85
halluc.	2	36	0	17	0	30	3	28	8	20	6	23	4	28	3	33
no output	0	0	50.0	0	25.0	0	0	0	0	0	12.5	0	50.0	0	0	0
error	0	12.5	37.5	0	50.0	25.0	0	25.0	0	25.0	75.0	0	12.5	37.5	0	50.0

TABLE V: Comparison of the evaluated precision P, recall R, F1-Score, hallucinated objects, prompts without output and errors during evaluation using LLaVA and GPT-4o on the low clutter sample images shown in the first row of Fig. 2.

are specifically trained on a kitchen dataset like KitchenCOCOv2 or KitchenLVIS. Additionally, further investigations are necessary to clarify, why the GELAN weights of YOLOv9 performed better than the standard weights. In the case of MLLMs, it would be interesting to investigate further prompts, like asking for specific objects and comparing these results.

Acknowledgment

This work was carried out as a group project by Julia Kuhn, Bernhard Pandion and Harald Straßgürtl. We divided the work and writing of the paper as follows:

- **Bernhard Pandion**: focus on Yolov9 and LLaVA (sections II-A, IV-A1, V-B1)
- **Harald Straßgürtl**: focus on Detectron2 and LLaVA (sections II-B, IV-B, V-B2)
- **Julia Kuhn**: focus on Detic and GPT-4o (sections II-C, IV-C, V-B3)

All other parts were written together.

References

[1] T. Yamamoto, T. Nishino, H. Kajima, M. Ohta, and K. Ikeda, "Human support robot (hsr)," in *ACM SIGGRAPH 2018 Emerging Technologies*, ser. SIGGRAPH '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3214907.3233972

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[3] J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 502–511, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:104292012

[4] J. Lee and K. il Hwang, "Yolo with adaptive frame control for real-time object detection applications," *Multimedia Tools and Applications*, vol. 81, pp. 36 375 – 36 396, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:240519114

[5] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[6] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," 2024.

[7] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," 2024.

[8] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[9] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[10] F. Massa and R. Girshick, "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch," https://github.com/facebookresearch/maskrcnn-benchmark, 2018, accessed: [Insert date here].

[11] A. Gupta, P. Dollár, and R. Girshick, "Lvis: A dataset for large vocabulary instance segmentation," 2019.

[12] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[13] V. Pham, C. Pham, and T. Dang, "Road damage detection and classification with detectron2 and faster r-cnn," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 5592–5601.

[14] G. Merz, Y. Liu, C. J. Burke, P. D. Aleo, X. Liu, M. Carrasco Kind, V. Kindratenko, and Y. Liu, "Detection, instance segmentation, and classification for astronomical surveys with deep learning (deepdisc): detectron2 implementation and demonstration with Hyper Suprime-Cam data," *Monthly Notices of the Royal Astronomical Society*, vol. 526, no. 1, pp. 1122–1137, 09 2023. [Online]. Available: https://doi.org/10.1093/mnras/stad2785

[15] F. Chincholi and H. Koestler, "Detectron2 for lesion detection in diabetic retinopathy," *Algorithms*, vol. 16, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1999-4893/16/3/147

[16] A. B. Abdusalomov, B. M. S. Islam, R. Nasimov, M. Mukhiddinov, and T. K. Whangbo, "An improved forest fire detection method based on the detectron2 model and a deep learning approach," *Sensors*, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1512

[17] R. Singh, S. Shetty, G. Patil, and P. J. Bide, "Helmet detection using detectron2 and efficientdet," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021, pp. 1–5.

[18] M. O. Silva, M. D. Valadão, V. L. Cavalcante, A. V. Santos, G. M. Torres, E. V. Mattos, A. M. Pereira, M. S. Uchôa, L. M. Torres, J. E. Linhares, N. E. Silva, A. P. Silva, C. F. Cruz, S. Rômulo, R. J. Belem, T. B. Bezerra, S. Waldir, and C. B. Carvalho, "Action recognition of industrial workers using detectron2 and automl algorithms," in *2022 IEEE International Conference on Consumer Electronics - Taiwan*, 2022, pp. 321–322.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," https://doi.org/10.48550/arXiv.1506.01497, 2016.

[20] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," https://doi.org/10.48550/arXiv.1703.06870, 2018.

[21] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," https://doi.org/10.48550/arXiv.1612.03144, 2017.

[22] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," 2022.

[23] CatOneTwo. (2020) Wsod-paper-list. [Online]. Available: https://github.com/CatOneTwo/WSOD-Paper-List/blob/master/README.md

[24] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[25] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[26] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," 2021.

[27] Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang, "The dawn of lmms: Preliminary explorations with gpt-4v(ision)," 2023.

[28] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[29] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," 2023.
[30] H. Liu, C. Li, Y. Li, B. Li, Y. Zhang, S. Shen, and Y. J. Lee, "Llava-next: Improved reasoning, ocr, and world knowledge," January 2024. [Online]. Available: https://llava-vl.github.io/blog/2024-01-30-llava-next/
[31] J. Kuhn, H. Strassguertl, and B. Pandion, "Kitchencocov2 dataset," https://universe.roboflow.com/lvarobotvision/kitchencocov2, June 2024, visited on 2024-06-18. [Online]. Available: https://universe.roboflow.com/lvarobotvision/kitchencocov2
[32] ——, "Kitchenlvis dataset," https://universe.roboflow.com/lvarobotvision/kitchenlvis, June 2024, visited on 2024-06-18. [Online]. Available: https://universe.roboflow.com/lvarobotvision/kitchenlvis
[33] D. Zimmer, "Kitchencoco dataset," https://universe.roboflow.com/dzws-pct7r/kitchencoco, jan 2024, visited on 2024-06-20. [Online]. Available: https://universe.roboflow.com/dzws-pct7r/kitchencoco
[34] B. Dwyer, J. Nelson, T. Hansen *et al.*, "Roboflow [software]," https://roboflow.com, 2024.
[35] W. K. Yiu, "Github implementation of paper - yolov9: Learning what you want to learn using programmable gradient information," 2024. [Online]. Available: https://github.com/WongKinYiu/yolov9
[36] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. (2021) Detectron2 model zoo and baselines. [Online]. Available: https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md
[37] X. Zhou, V. Koltun, and P. Krähenbühl, "Probabilistic two-stage detection," 2021.
[38] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 2020.
[39] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.
[41] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. (2022) Detecting twenty-thousand classes using image-level supervision. [Online]. Available: https://github.com/facebookresearch/Detic/blob/main/docs/MODEL_ZOO.md