Task 1 - Object detection

Team 4:

- María José Millán
- Agustina Ghelfi
- Laila Aborizka

DATASET DESCRIPTION - KITTI MOTS

KITTI-MOTS[1] is an extension of the KITTI dataset that provides multi-object tracking and segmentation (MOTS) annotations for cars and pedestrians. The dataset is organized into subfolders, each containing a sequence of images with two categories: 1 for cars and 2 for pedestrian

Train	Val
12 sequnces	9 sequnces
8,073 pedestrian masks	3,347 pedestrian masks
18,831 car masks	8,068 car masks

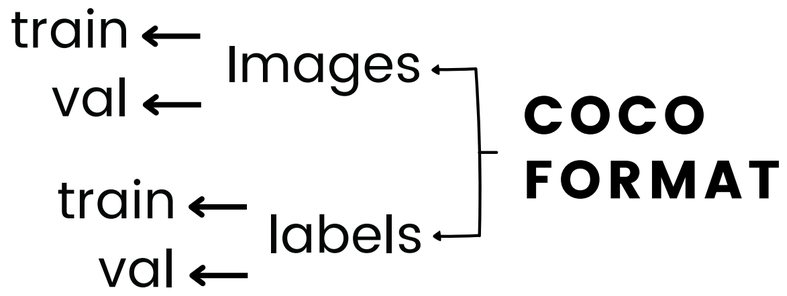
Test
29 sequnces
No masks

The annotations from train and validation is one .txt per sequence, structure:

`frame,id,class_id,width,height,rle`

id: have information of class_id performing floor division by 1000 and of intance_id by modulo 1000.

rle: compression method that encodes consecutive repeating values. Here is saved the bbox x,y,w,h.



YOLO

Annotations normalized

`class_id,x_center,y_center,w,h`

FASTER R-CNN (DETECTRON2) AND DETR

Annotations denormalized

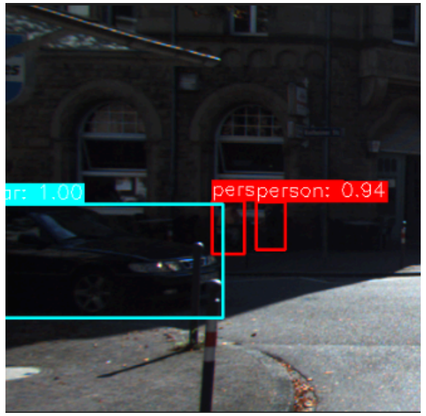
```
{"images":{..}
 "annotations":{..,"bbox":[xmin,ymin,w,h],..}
 "categories":{..}}
```

[1] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, A. Geiger, and B. Leibe, "MOTS: Multi-Object Tracking and Segmentation," arXiv preprint arXiv:1902.03604, 2019.

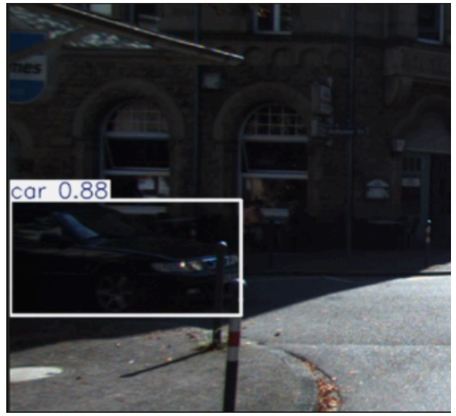
INFERENCES

FASTER R-CNN (DETECTRON2)

DETR



YOLOV11



The comparison of models was performed using the same **confidence threshold of 0.5**. Significant differences were observed in person detection within the same image. Detectron identified a single person, while YOLOv11 failed to detect any. On the other hand, DETR successfully detected both people present in the analyzed area.

Detectron and DETR show FN, while YOLO made no detections. At the first example, this might seem like a drawback, but in this case, it's beneficial. Here YOLO's lack of detection prevents adding more FN, leading to a better metric. This highlights how, depending on the context, no detection can be preferable to an incorrect one that negatively impacts performance.

EVALUATION

Metric Used: mAP50-95

To evaluate the different models we used the Mean Average Precision (mAP) metric, with varying thresholds from 0.50 to 0.95 (mAP50-95).

Label Conversion

Since KITTI-MOTS uses different labels than COCO, a class mapping was performed:

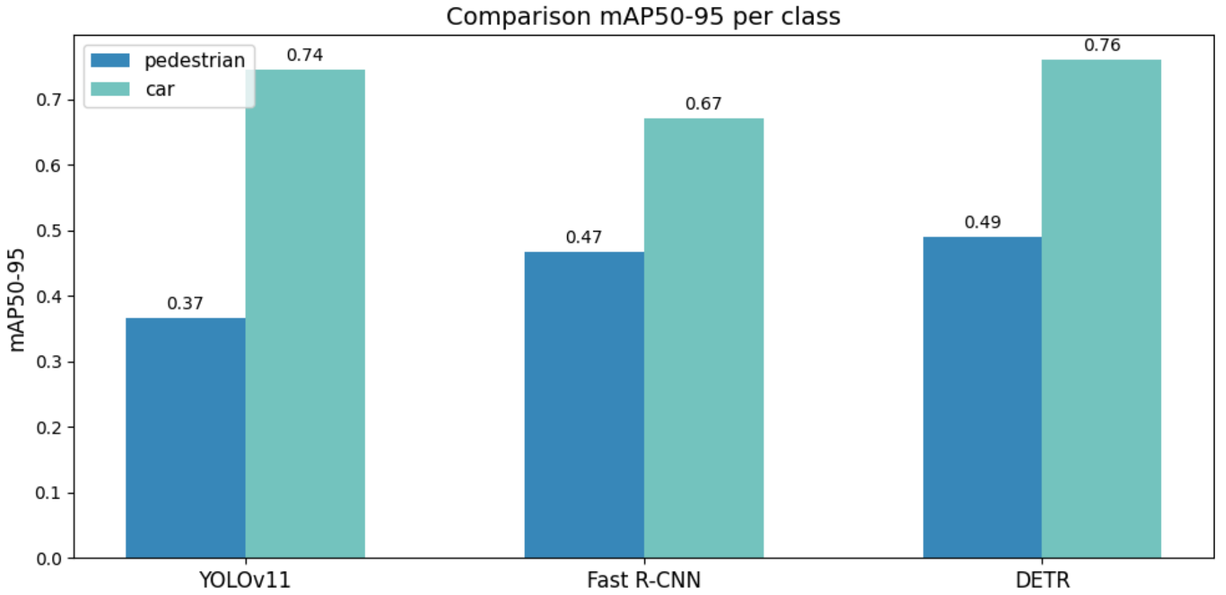
- "Pedestrian" from KITTI-MOTS is mapped to "person" in COCO.
- "Car" from KITTI-MOTS remains as "car" in COCO.

To implement this conversion:

- In Detectron2, MetadataCatalog was modified to reflect the mapped classes.
- In Ultralytics YOLO, the class to evaluate was defined in the configuration.
- In DETR generated predictions are transformed based on its value to match label scheme. DETR uses ‘1’ for person and ‘3’ for car.

Quantitative Results

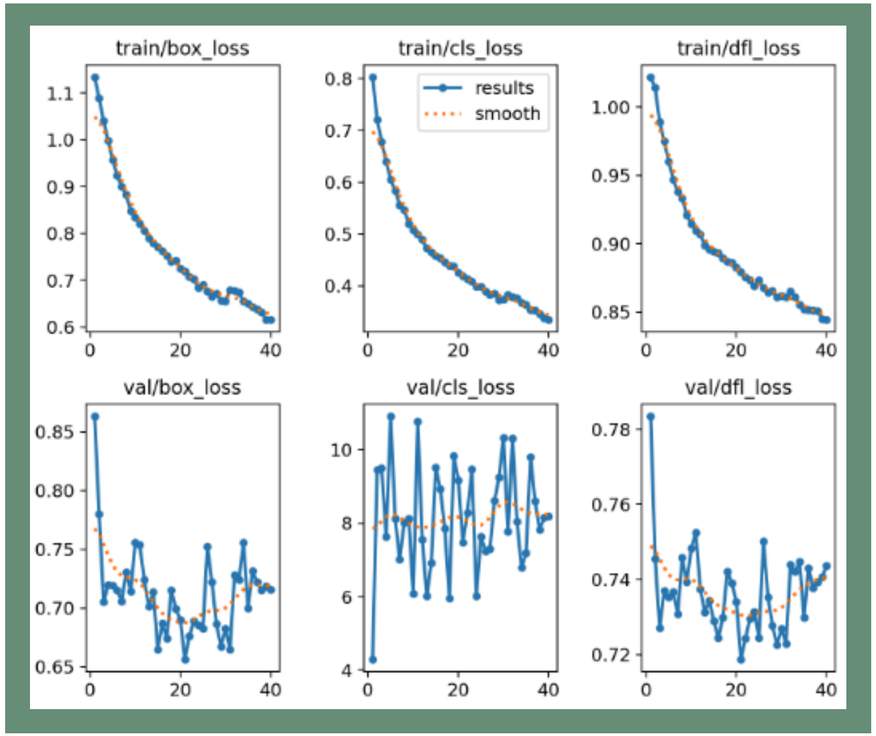
The obtained mAP50-95 values are shown in the barplot below, for each class.



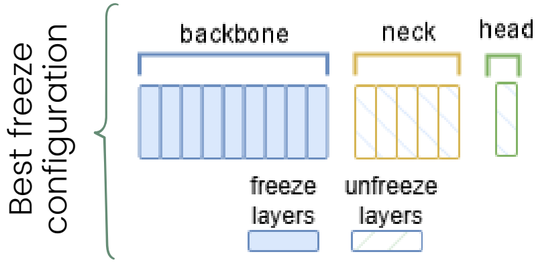
Observations

- YOLOv11 performs better on the "car" class (0.74) than on "pedestrian" (0.37).
- Faster R-CNN has a more balanced performance, detecting "pedestrian" better (0.47) but scoring lower on "car" (0.67).
- DETR performs best in detecting “pedestrian” with (0.49) and best on “car” with (0.76)

FINETUNE-YOLOV11



Graphs of the best model



Different hyperparameters such as learning rate, optimizer and momentum were analyze. Different parts of the model were frozen, and augmentations were applied in multiple combinations, we are presented some of them in the table.

Augmentations with best freeze configuration

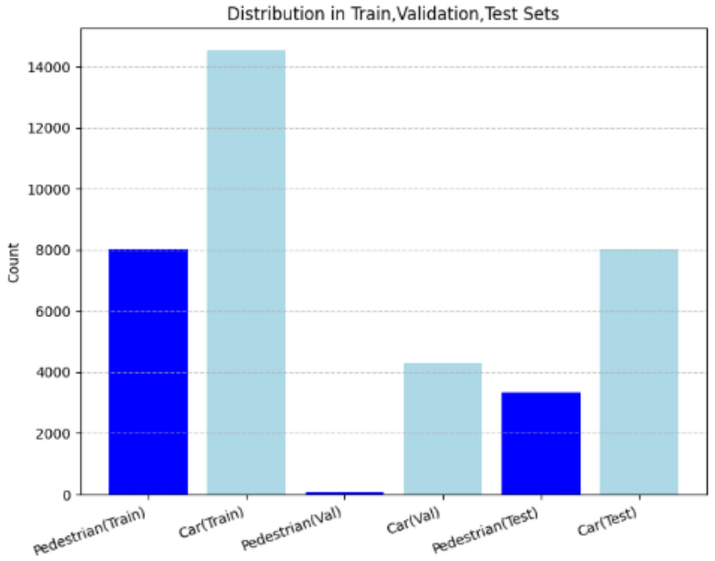
	Pedestrian	Car
backbone and neck freeze lr=0.001	mAP50=0.25 mAP50-95=0.19	mAP50=0.82 mAP50-95=0.63
backbone freeze lr=0.001 with clf=1 and dlf=2	mAP50=0.45 mAP50-95=0.33	mAP50=0.81 mAP50-95=0.64
backbone freeze lr=0.0005	mAP50=0.31 mAP50-95=0.23	mAP50=0.81 mAP50-95=0.65
backbone freeze lr=0.001	mAP50=0.62 mAP50-95=0.43	mAP50=0.84 mAP50-95=0.66
hsv_h=0.01,hsv_s=0., hsv_v=0.2, fliplr=0.5, translate=0.1, scale=0.3	mAP50=0.43 mAP50-95=0.33	mAP50=0.84 mAP50-95=0.65
hsv_h=0.015, hsv_s= 0.2,hsv_v=0.3 , translate=0.2, scale=0.4, crop_fraction=0.8	mAP50=0.45 mAP50-95=0.31	mAP50=0.84 mAP50-95=0.63
fliplr=0.5, translate=0.05, scale=0.2	mAP50=0.51 mAP50-95=0.36	mAP50=0.84 mAP50-95=0.63

All training losses decrease smoothly, indicating stable learning. However, the validation losses show more fluctuation, especially cls_loss, which suggests variability in classification performance. The box loss and dfl loss also fluctuate but show a downward trend, indicating that the model improves its localization over time. This suggests that the model is learning to detect objects effectively, but classification on the validation set remains more challenging.

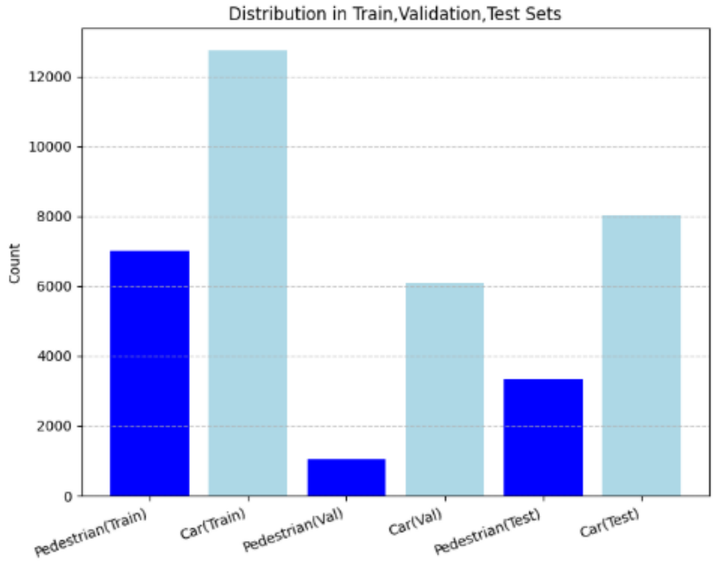
TWO DIFFERENTS DATASET SPLIT

In the initial split that is the one used in the previous experiments, where we split the reainging test in train and validation, we observed that the number of detected pedestrians in the training set was extremely low. To address this issue, we generated a new split with a higher number of pedestrian instances. Although pedestrians remain a minority class, the first split had almost none, making the evaluation less reliable.

SPLIT 1



SPLIT 2

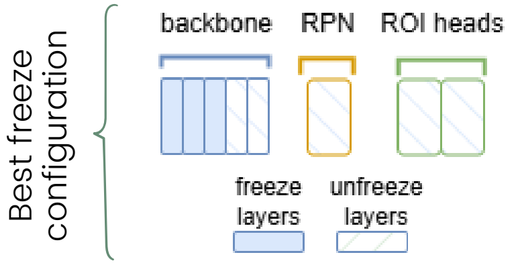


We decided to proceed with dataset split 2, as it had a better data distribution. The fine-tuning process was rerun on this new dataset, and the best results were achieved with the last augmentation presented in the previous slide. The final metrics obtained were mAP50 = 0.65 and mAP50-95 = 0.26 for the "Person" class, and mAP50 = 0.85 and mAP50-95 = 0.67 for the "Car" class.

FINETUNE-FASTER R-CNN

To improve the performance of Faster R-CNN on KITTI-MOTS, several experiments were conducted, including layer freezing, hyperparameter tuning, and data augmentation.

Layer Freezing Strategy



- The freezing process involved gradually unfreezing layers from the head to the entire network.
- The best configuration, involved freezing the first three layers of the backbone, while keeping the rest trainable.

Best experiments and results

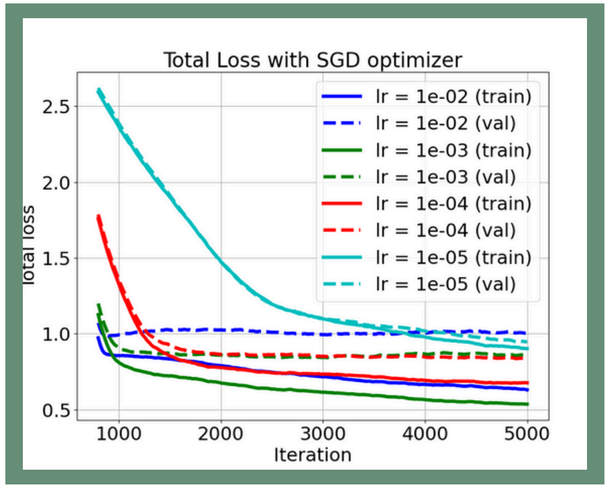
		mAP@50-95	
		Pedestrian	Car
FREEZE_AT=3, FPN=🔒, RPN=🔒, ROI=🔒	AdamW, lr = 1e-5	0.50	0.58
	SGD, lr = 1e-4	**0.59**	**0.61**
Compose([A.HorizontalFlip(p=0.5), A.ShiftScaleRotate(p=0.2), A.RandomBrightnessContrast(p=0.3)])	min_area = 200 min_visibility = 0.1	0.55	0.56
Compose([A.HorizontalFlip(p=0.5), A.ShiftScaleRotate(p=0.2), A.RandomBrightnessContrast(p=0.3)])		0.57	0.56

Hyperparameter Optimization

Experiments were conducted using two optimizers: AdamW and SGD. For each optimizer, different learning rates (1e-5, 1e-4, 1e-3, 1e-2) were tested to determine the best configuration.
The graph presents the variation of total loss in the case of SGD as the learning rate changes.

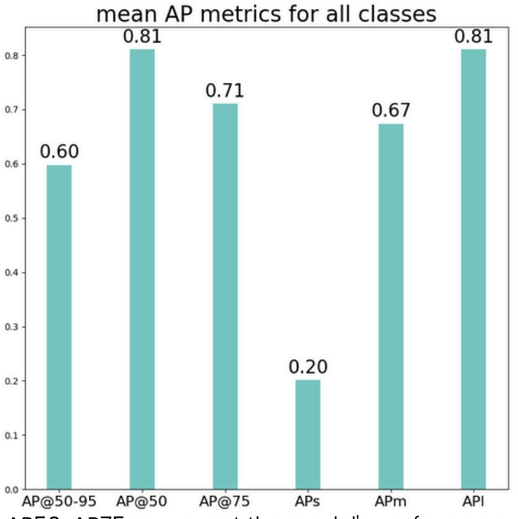
The total loss in Faster R-CNN comprises:

- Classification loss
  How well the model predicts object categories
- Bounding box regression loss
  Accuracy of bounding box coordinates.
- RPN loss
  Region proposal accuracy.



Various augmentation strategies were tested using Albumentations, but no significant performance improvements were observed.

The barplot reveals that the model performs well on medium and large objects, but struggles with detecting small objects, indicating potential areas for future improvement.



AP50, AP75: represent the model's performance for IoU 50% and 75%, respectively.
APs, APm, APl: represent the model's performance on small, medium, and large objects, respectively.

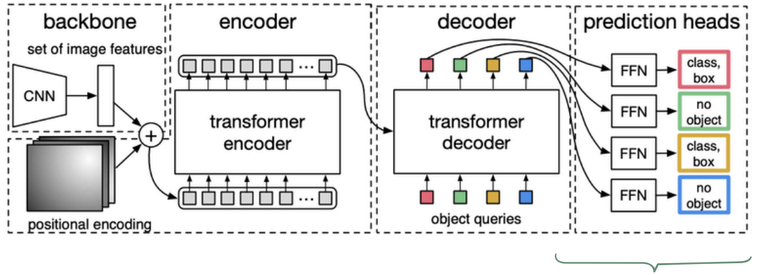
FINETUNE-DETR

Attempts to fine-tune DETR did not yield good results. During experimentation, we followed these steps:

1. Freezing all model parameters except the decoder and prediction heads.
2. Freezing all model parameters, including the decoder, but unfreezing the class prediction and bbox prediction heads.
3. Freezing all model parameters except the bbox prediction head.
4. Freezing all model parameters except the class prediction head.

Our experiments did not succeed in improving DETR's performance. We did see a slight improvement with unfreezing the class prediction heads, which we believe is because the classification task is generally easier to solve than the bbox prediction task.
We believe the challenges in fine-tuning DETR were due to the following reasons:

1. Transformers are data-hungry: DETR requires a larger dataset, which means more training time and higher memory consumption. Our dataset was not large enough to fully train the model.
2. Slow convergence of transformers: Transformers need longer training times and more epochs to show better results, but our training was not sufficient to reach optimal performance



Unfreezing the class prediction head
Best Freeze Configuration

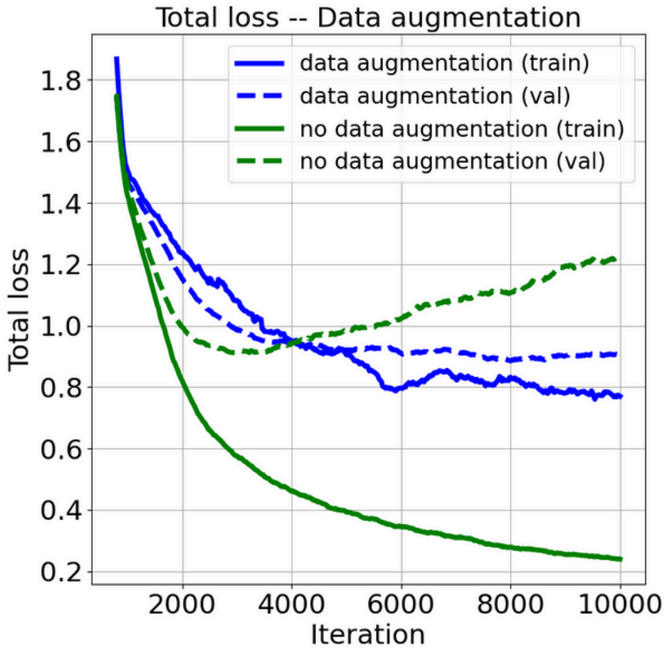
	Pedestrian	Car
Unfreezing the decoder and prediction heads	mAP50= 0 mAP50-95= 0	mAP50= 0 mAP50-95= 0
Unfreezing both bbox prediction head and class prediction head	mAP50= 0 mAP50-95= 0	mAP50= 0 mAP50-95= 0
Unfreezing the bbox prediction head lr= 1e-5 optimizer = AdamW	mAP50=0.001 mAP50-95= 0	mAP50= 0.004 mAP50-95= 0
Unfreezing the class prediction head lr = 1e-5 optimizer = AdamW	mAP50= 0.017 mAP50-95=0.006	mAP50=0.019 mAP50-95=0.003

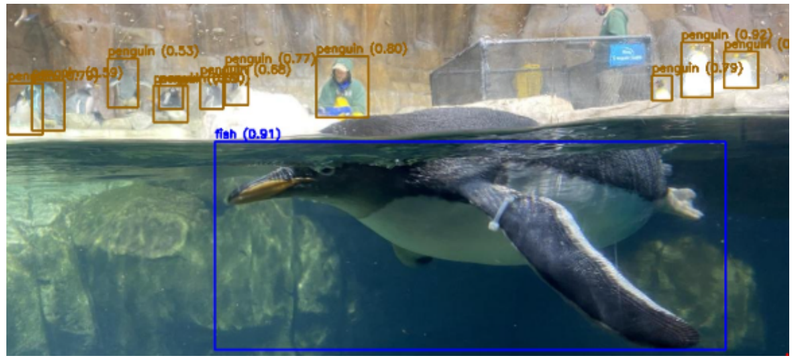
FINETUNE FOR A DOMAIN SHIFT

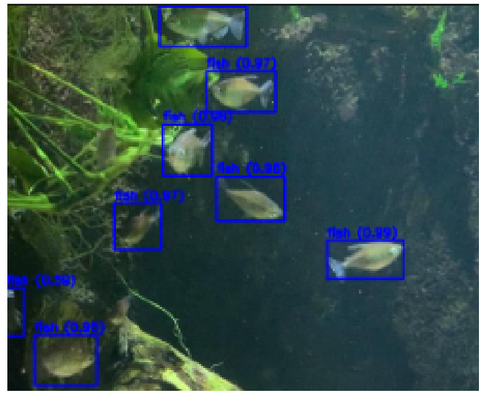
Aquarium Dataset from roboflow link in the image.It has 638 images annotated with 7 different classes of aquatic species.

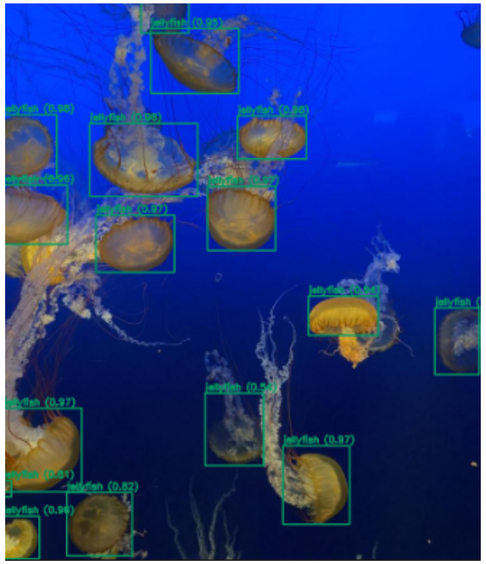
	mAP50-95		mAP50-95
Jellyfish	0.41	starfish	0.17
shark	0.31	puffin	0.15
fish	0.24	stingray	0.05
penguin	0.19		

Different learning rates and optimizers were tested, with AdamW and a learning rate of 0.0001 yielding the best results. Various combinations of augmentations were attempted; however, as seen in the graph comparing the loss, the model without augmentation exhibited overfitting, despite achieving a higher total mAP50 of 39. Therefore, the model with augmentation was chosen.









While the model performs well in detecting certain objects, as seen with the accurate detections of fish and jellyfish, it also produces many incorrect predictions. For example, in the image, it mistakenly classifies humans as penguins and misidentifies the closest penguin as a fish. This model could be improved by increasing the number of labeled training images or by generating synthetic data based on the existing images.

ANALYSIS BETWEEN MODELS

YOLOV11

PRE-TRAINED

person mAP50-95=0.36
car mAP50-95=0.74

FINETUNED

person mAP50-95=0.38
car mAP50-95=0.71

FASTER R-CNN

person mAP50-95=0.47
car mAP50-95=0.67

person mAP50-95=0.44
car mAP50-95=0.65

The comparison between YOLOv11 and Faster R-CNN highlights the trade-offs between the models before and after fine-tuning.

In the pre-trained stage, both YOLOv11 and Faster R-CNN, perform well in detecting cars, achieving high mAP scores, but struggle with pedestrian detection.

After fine-tuning, Faster R-CNN improves pedestrian detection, but at the cost of missing some cars:

In the pre-trained stage, Faster R-CNN detects all vehicles in the scene, while after fine-tuning, the model gains better awareness of pedestrians but struggles to recognize more distant or occluded cars.

This trade-off suggests that the optimal model selection depends on the application.

INFERENCE TIME

YOLO is significantly faster than Faster R-CNN for inference, making it well-suited for real-time applications where speed is critical. However, this often comes at the cost of slightly lower accuracy compared to two-stage detectors like Faster R-CNN.

FINETUNE

SUMMARY

- We performed fine-tuning for all three proposed models, with YOLO being the only one that showed improved results when comparing the evaluation of the fine-tuned model with the pre-trained one.
- We would like to highlight that in both YOLO and Faster R-CNN, the class that benefited the most from fine-tuning was "person." One reason for this improvement was the use of class-based weighting during training. Normally, models treat all classes equally when calculating the loss. However, if the dataset is imbalanced, the model may become biased toward the dominant class. By setting LOSS_WEIGHT, we adjusted the contribution of each class to the total loss, helping to balance the learning process.
- There were clear challenges when implementing fine-tuning for DETR, as the obtained mAP values were very low.

DOMAIN SHIFT

We finetuned detectron for an Aquarium Dataset from Roboflow :

638 images annotated - 7 classes of aquatic species

Various augmentation strategies were tested, and while the non-augmented model achieved a higher mAP, it exhibited clear overfitting, leading to the selection of the augmented model whith a total mAP50 of 39.

While the model demonstrates strong detections for certain objects, such as fish and jellyfish, it also produces significant misclassifications, like identifying humans as penguins or mistaking a penguin for a fish.

Further improvements could be achieved by increasing the number of labeled images or generating synthetic data.