Master in Computer Vision Barcelona

Image Classification

Module: C5

Group: 7

Students: Cristian Gutiérrez
Iñaki Lacunza
Marco Cordón
Merlès Subirà

Index

Task c: First tries.

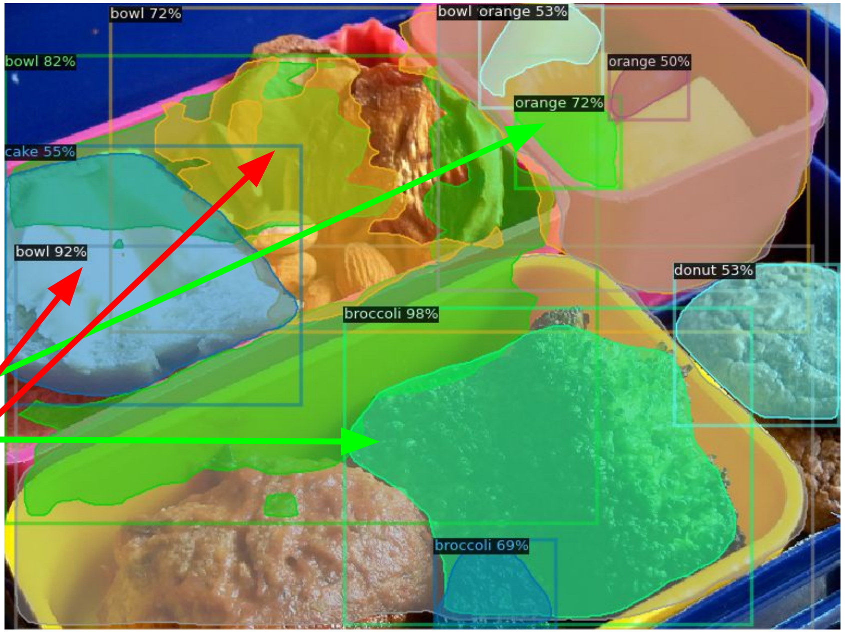
The first step in the week's project is install and understand Detectron2. For this purpose we are going to follow the demo tutorial that is in Google Colab. To be sure that all are perfect installed and ready to be used we are going to make some runs with the **COCO dataset**. This dataset is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- 330K images (>200K labeled)
- 1.5 million object instances
- **80 object categories**
- 91 stuff categories
- 5 captions per image

In order to run some predictions we are going to load a pre-trained model in COCO dataset: **Mask R-CNN**

✅ Some good detection/segmentations

❌ Some bad detection/segmentations

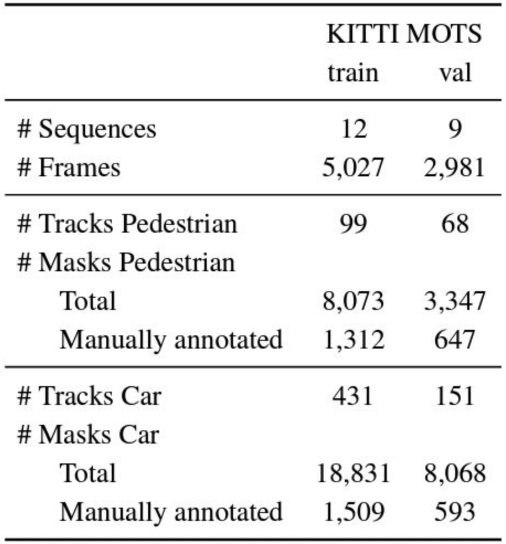


Task c: Inference on KITTI-MOTS dataset

Once we know that the Detectron2 framework works correctly, we can start to work with our project's dataset: KITTI-MOTS. This dataset contains 21 train sequences (12 train / 9 validation) and 29 test sequences. The sequences are road sequences where we can see specially cars and pedestrians.

To run an interference with this dataset we are going to use again the pre-trained models with COCO dataset. We have 2 differents:

- Faster R-CNN: extracts an infer class, confidence and bounding box for each proposal.
- Mask R-CNN: is an extension of Faster that predicts a binary mask for each ROI Head.



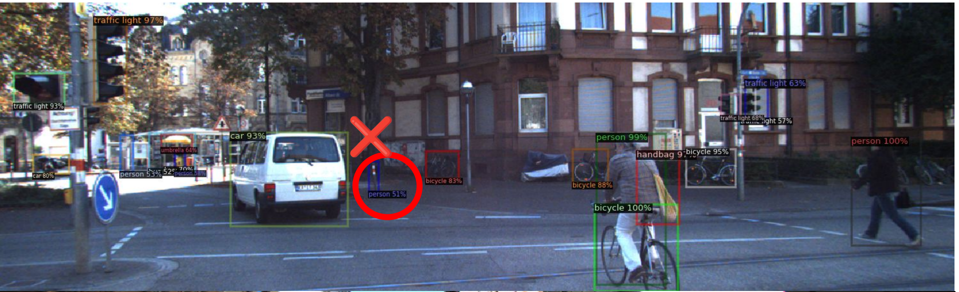
	KITTI MOTS train	KITTI MOTS val
# Sequences	12	9
# Frames	5,027	2,981
# Tracks Pedestrian	99	68
# Masks Pedestrian		
Total	8,073	3,347
Manually annotated	1,312	647
# Tracks Car	431	151
# Masks Car		
Total	18,831	8,068
Manually annotated	1,509	593

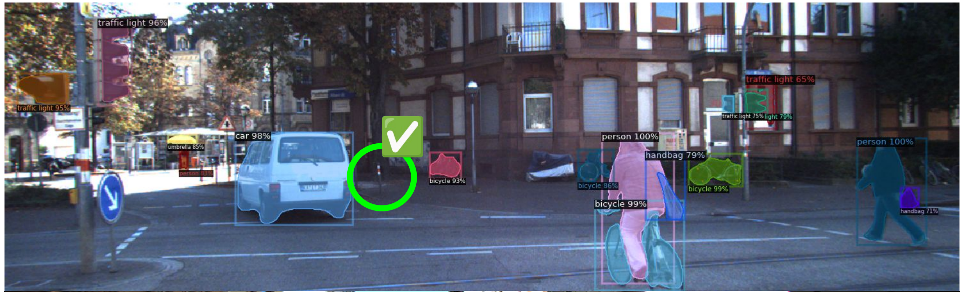
Example of KITTI-MOTS sequence.

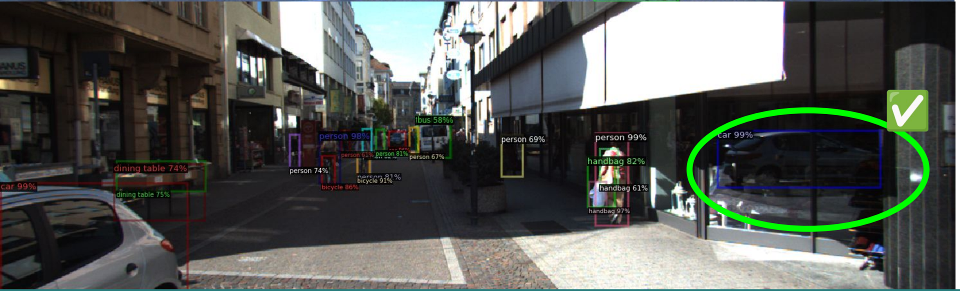
Task c: Inference on KITTI-MOTS dataset (quali results)

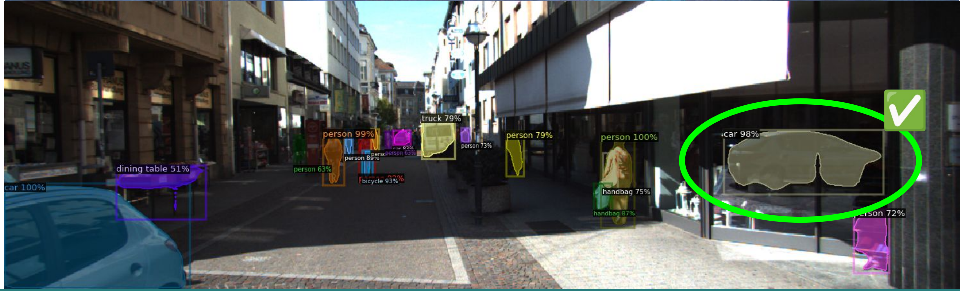
FASTER R-CNN

MASK R-CNN









As we can see in the images above the results are very similar. The main difference between the two models is the instance segmentation of the MASK model. Like the models are pre-trained with COCO dataset they recognize more objects than cars and pedestrians, for example traffic lights or bicycles. Only visualizing the images we can confirm that the pre-trained model works with a high precision in our images (the both has recognized the car in the mirror of the 2nd sequence) and there are few errors (mainly on small objects and in FASTER). Moreover, execution times were similar.

Task d: Evaluate models on KITTI-MOTS dataset

Now that we have ran the inference on our dataset we have to analyze quantitatively the obtained results.

To make it we are going to use the COCO metrics provided by Detectron2, but the main problem of this is that we have to map class labels of KITTI-MOTS to class labels of COCO in the ground truth data:

	KITTY		COCO
CAR:	1	$\longrightarrow$	2
PEDESTRIAN:	2	$\longrightarrow$	0
IGNORE:	10	$\longrightarrow$	X

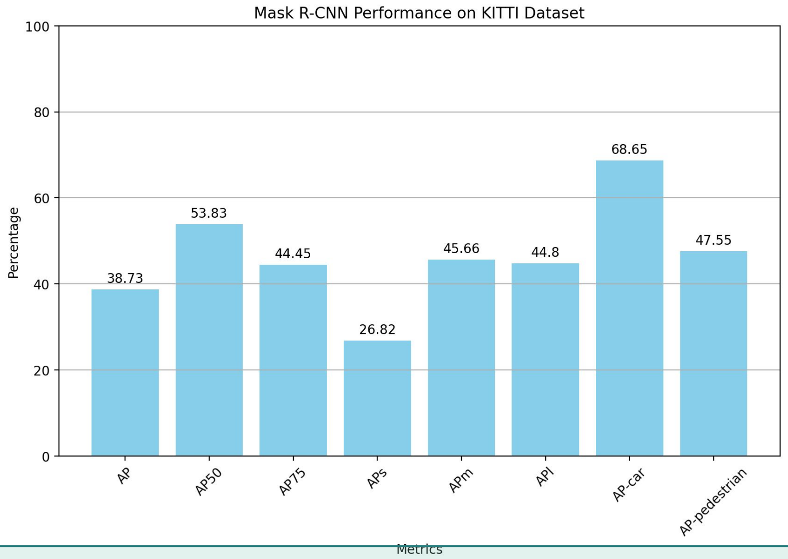
Problem found:
In the ground truth instances we found that there were some objects labelled with class 10, that does not correspond with any object of KITTY-MOTS so we will ignore it.

The metrics that COCO provides are mainly Averages Precisions (AP) and Average Recalls (AR), with different values of threshold in the Intersection over Union (IoU). AP $_{0.5}$ means that only the objects with a IoU higher than 0.5 will be considered as True detected, the same with AP $_{0.75}$. AP $_{0.5:0.95}$ refers to a mean average mAP over different IoU thresholds, from 0.5 to 0.95, step 0.05 (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95).

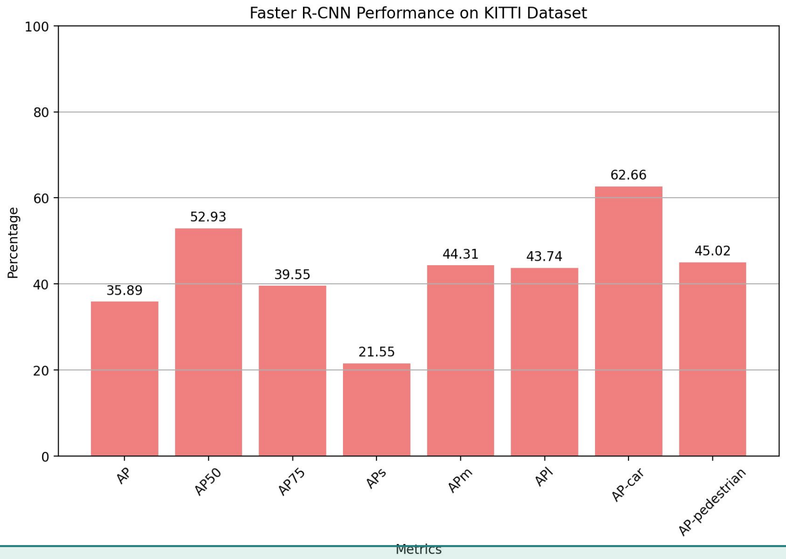
As we only want to run a inference on the pre-trained model, we need to split the train dataset into train and validation sequences, and only run the 9 validation sequences that we have splitted.

Task d: Evaluate models on KITTI-MOTS dataset

MASK R-CNN



FAST R-CNN



If we analyze the result plots we can see how similar are the performance between the two models, but it is true that the MASK model has slightly higher values. These results are expected due to the better performance that an instance segmentation should have over a bounding box. Moreover we can see how the AP-car is bigger than AP-pedestrian, that is another expected result, since the human shape is easier to confuse than the car one, how we have seen in task c (slide 5), where a pivot has confused with a pedestrian.

Task e: Fine-tune Faster R-CNN and Mask R-CNN

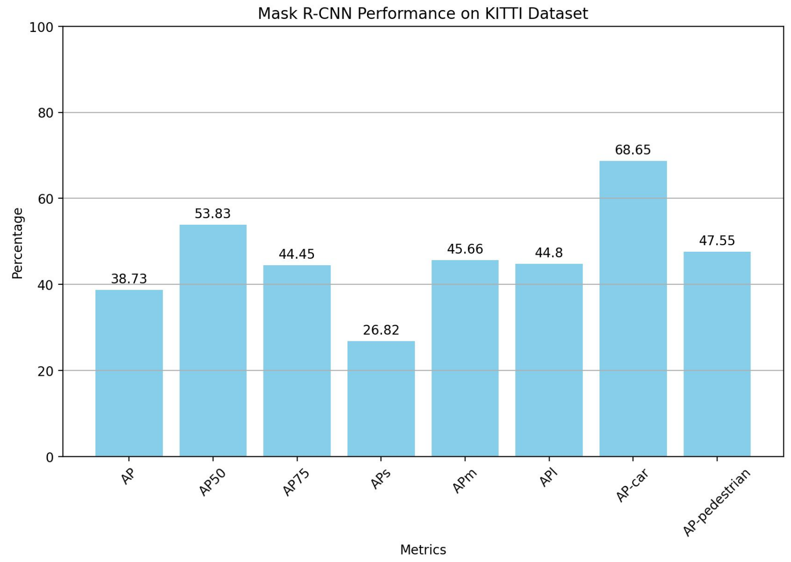
The goal of this task is to fine-tune both Faster R-CNN and Mask R-CNN using as guidance the COCO metrics we previously obtained. Moreover, we will focus first our efforts onto **Mask R-CNN** with its Detection metrics, that is, we will focus on the \`bbox\` metrics rather than the \`segm\` metrics as we think it has more weight during this lab scope.

Hyper-parameters

- Learning Rate: step size of each iteration
- Batch size
- LR Warm-Up strategies
- Non-Maximum Suppression threshold
- Segmentation threshold/score

Some revolve around LR because it affects the model ability to get to optimal solutions. Others such as NMS or the Segm Score revolve around improving the model's precision and results when predicting objects.

BASELINE MASK R-CNN METRICS



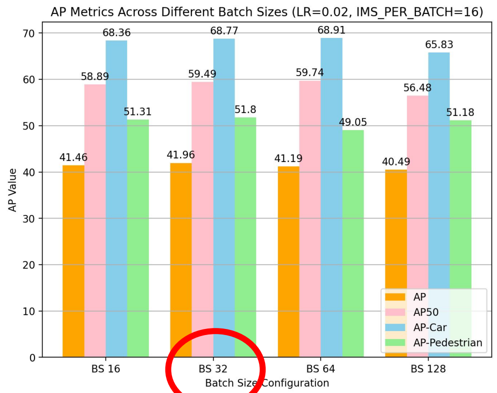
Task e: Fine-tune Faster R-CNN and Mask R-CNN

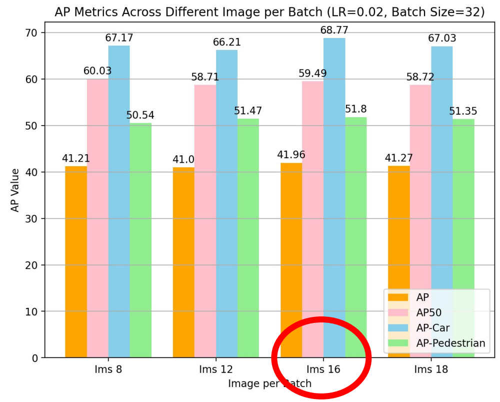
First, we identified the optimal combination of batch size and images per batch, conducting a benchmark with a fixed LR while varying both.

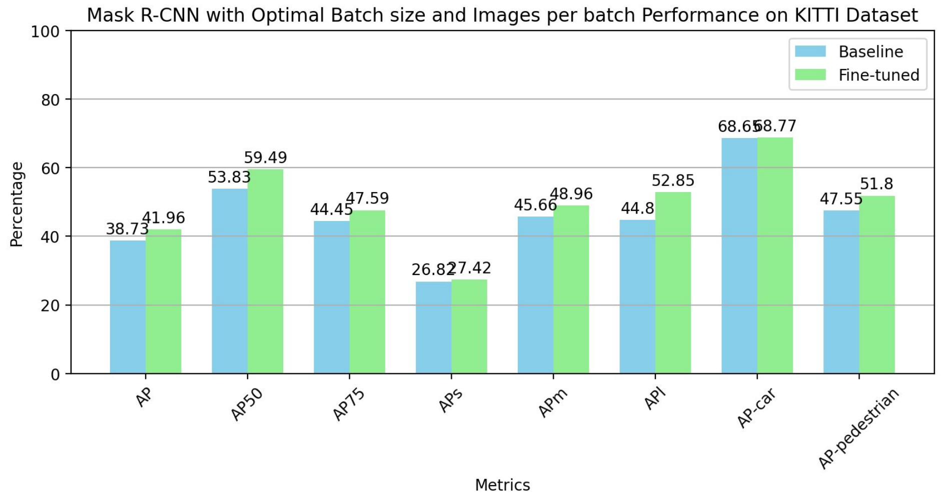
<u>Hyper-parameters</u>

- Max iters: **1500**
- Batch size: **32**
- Images per Batch: **16**

We then obtained the COCO metrics for all and the best results are obtained by a **Batch size of 32** and **Image per Batch of 16**.







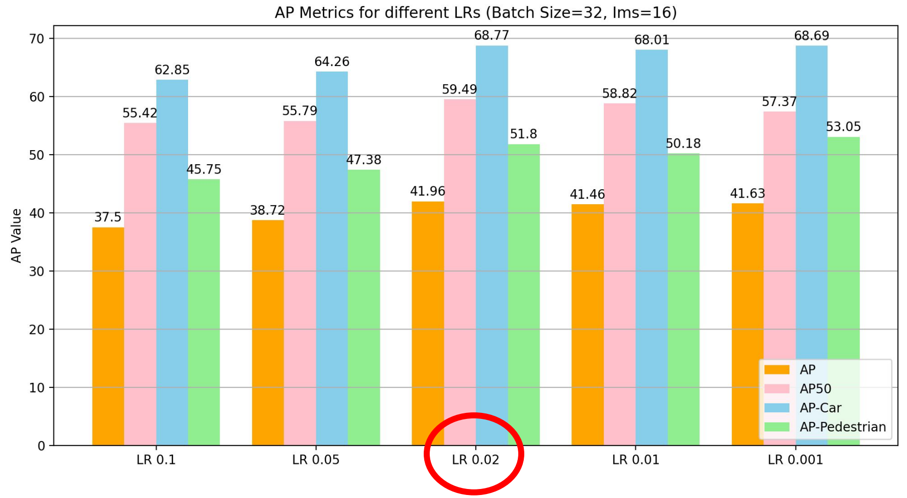
Task e: Fine-tune Faster R-CNN and Mask R-CNN

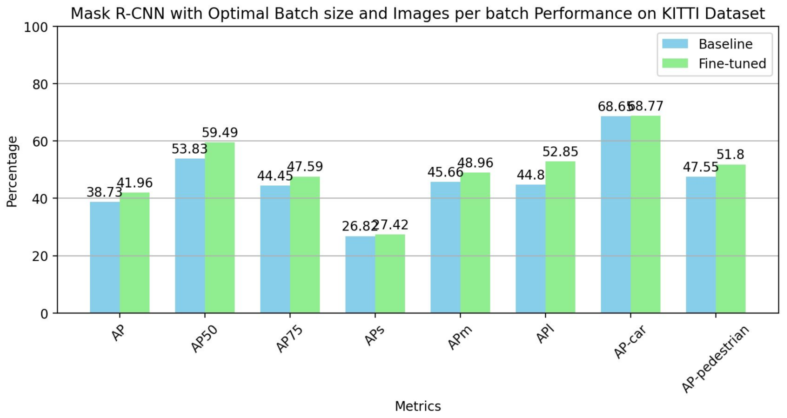
After finding the best Batch and Images per batch sizes, we now can find the optimal LR value, that is the step size we perform to converge to optimal solutions.

<u>Hyper-parameters</u>

- Max iters: **1500**
- Batch size: **32**
- Images per Batch: **16**
- Learning Rate: **0.02**

As we can see, the best LR value is what we were already using. In pedestrian detection smaller values of LR work well better but we will guide our fine-tune with the overall score.





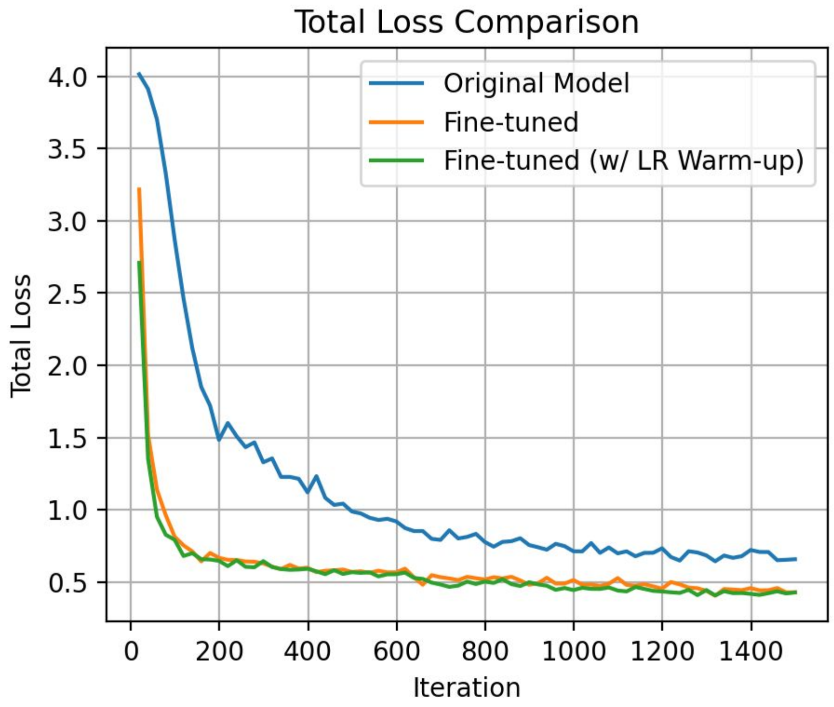
Task e: Fine-tune Faster R-CNN and Mask R-CNN

Next, we will focus on adding warm-up iterations at the beginning of the training to gradually increase the learning rate from a lower starting point to the predefined base learning rate. This should help with a vanishing / exploding gradients problem.

Hyper-parameters

- Max iters: **1500**
- Batch size: **32**
- Images per Batch: **16**
- Learning Rate: **0.02**
- **Warm up by 500 iters by a factor of 1/500 linear method**

The results are very similar with and without Warm-up, hence we will not consider it as it adds some time complexity during training.

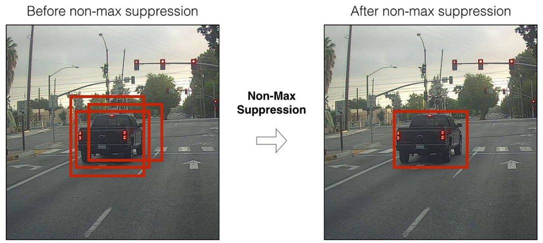


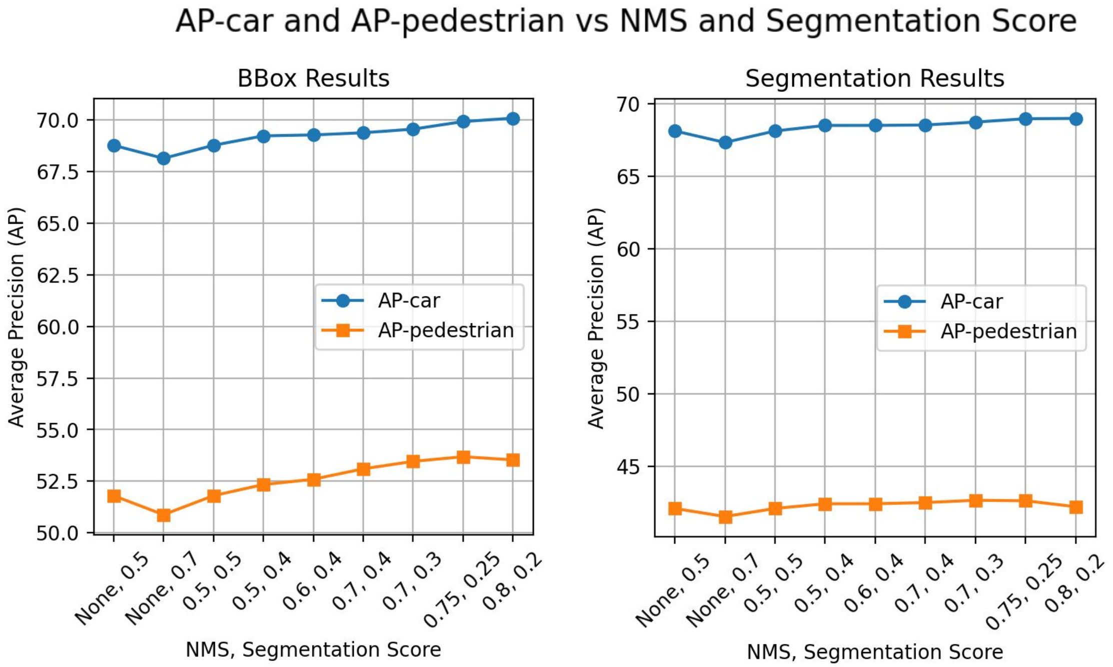
Task e: Fine-tune Faster R-CNN and Mask R-CNN

There is a **trade-off** between NMS and Segmentation Score. The lower it is our Score, the more aggressive will be our model in predicting objects and the more false positives NMS will filter out. Hence, we played with the threshold and obtained better results.

Hyper-parameters

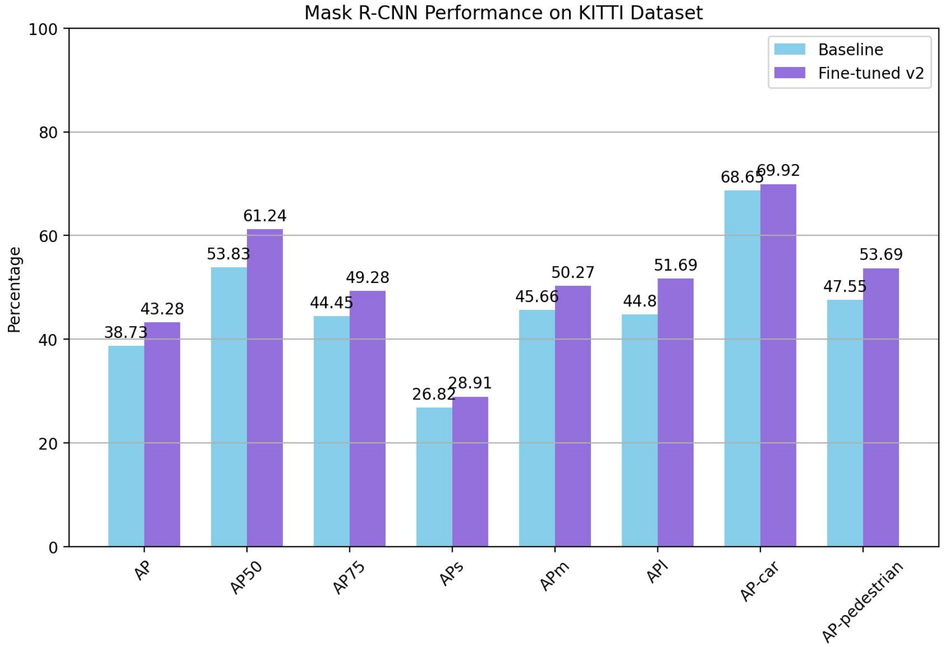
- Max iters: **1500**
- Batch size: **32**
- Images per Batch: **16**
- Learning Rate: **0.02**
- Non-Maximum Suppression: **0.75**
- Segmentation Score: **0.25**

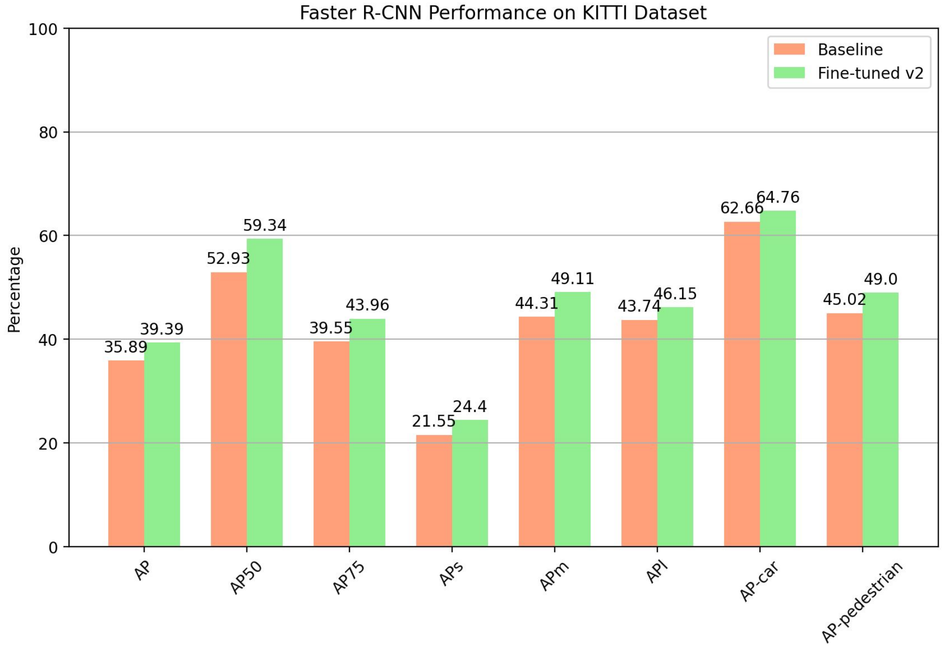




Task e: Fine-tune Faster R-CNN and Mask R-CNN

On Faster R-CNN we followed a similar procedure as we just did with Mask R-CNN, we had to lower the Images per batch so that it fit our resources, also note that it takes more time complexity the latter both on training and on inference.



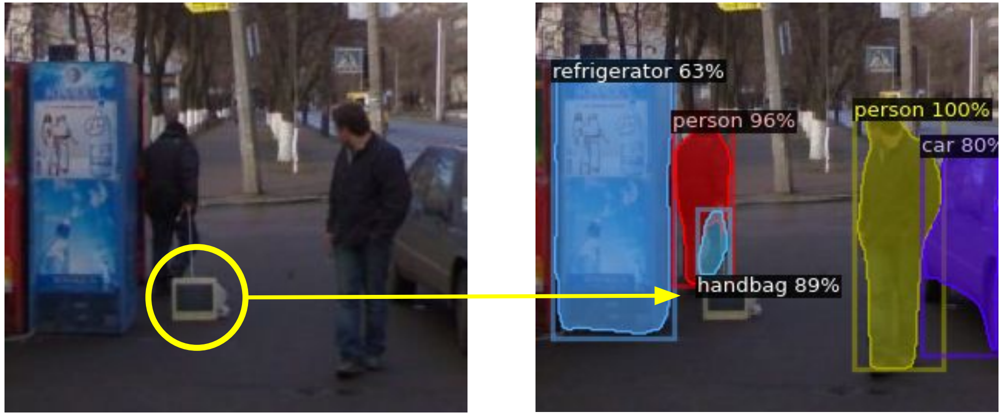


Max iters: **1500,** Batch size: **32,** Images per Batch: **16,** Learning Rate: **0.02**, Non-Maximum Suppression: **0.75,** Segmentation Score: **0.25**

Max iters: **1500,** Batch size: **32,** Images per Batch: **10,** Learning Rate: **0.002**, Non-Maximum Suppression: **0.7,** Segmentation Score: **0.3**

Task f.2 (Optional): Apply pre-trainedMaskRCNNorFasterRCNNon COCO in Out-ofContext Dataset.

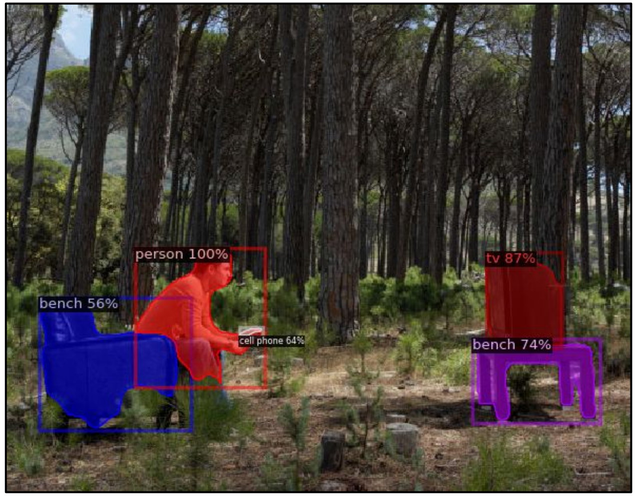
Now we are going to work with some special images. These images contains objects in in scenes that are not their “natural habitat”. The CNN has several techniques that employs the relation between labels to assign a class to an object, for instance if the model recognizes the class ‘clouds’ and there are another object that is similar to a plane, but the confidence is not too high the model will prioritize ‘plane’ class over others only because is rounded by clouds. Let’s see how it works using MASK RCNN:

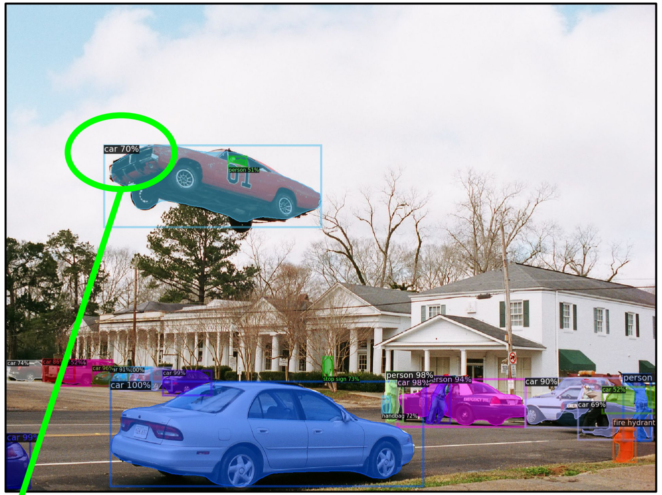


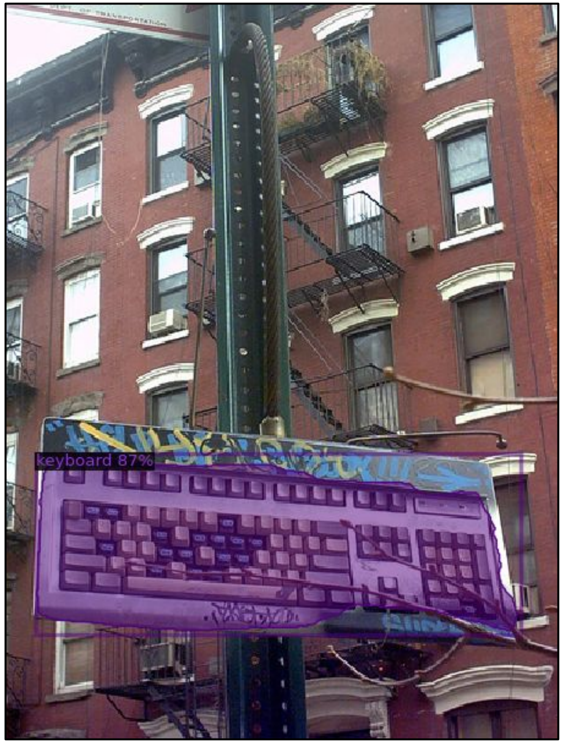
As we can see, all the ‘normal’ objects has been well classified but the computer not. Moreover, instead a computer has labeled a handbag with a 89% of confidence. It is a logical result because usually when there is an object in a person’s hand hanging it is usually a bag.

Task f.2 (Optional): Apply pre-trainedMaskRCNNorFasterRCNNon COCO in Out-ofContext Dataset.

Well classified images:



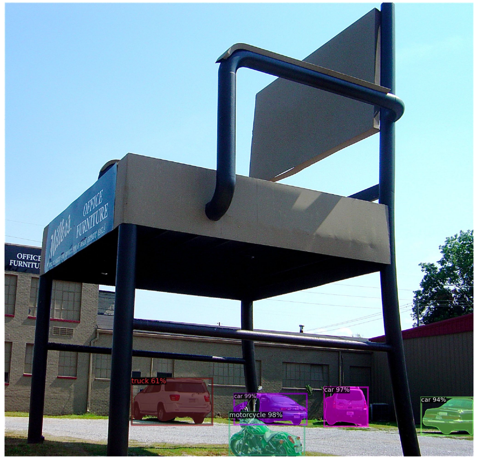




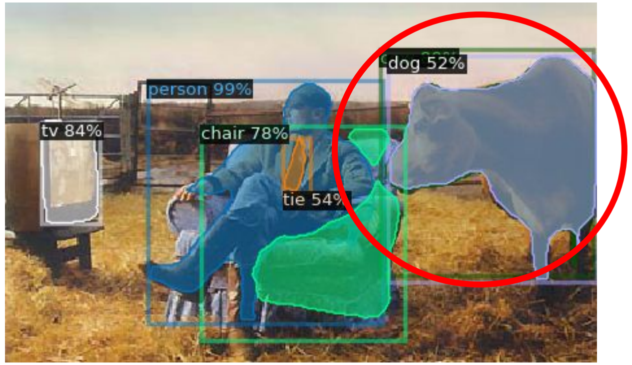
Despite the corrected label, the confidence percentage is lower than the rest of the image, (being all the car visible in a big size), only for being in the sky of the image

Task f.2 (Optional): Apply pre-trainedMaskRCNNorFasterRCNNon COCO in Out-ofContext Dataset.

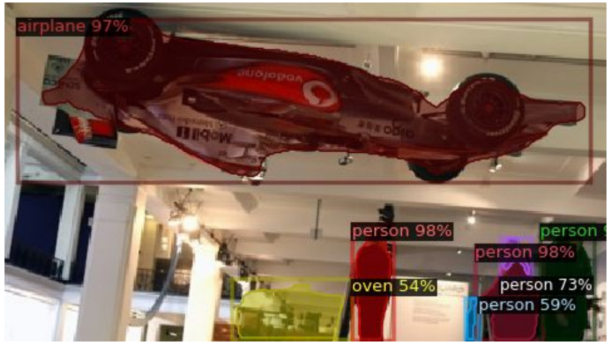
Wrong classified images:



Despite the full visibility of a standard chair the model does not recognize it only because of its size compared with a car or a motorbike



The model recognizes an animal shape, and how the most common animal that is usually near the human in the living room is the dog, it classifies the cow as one.

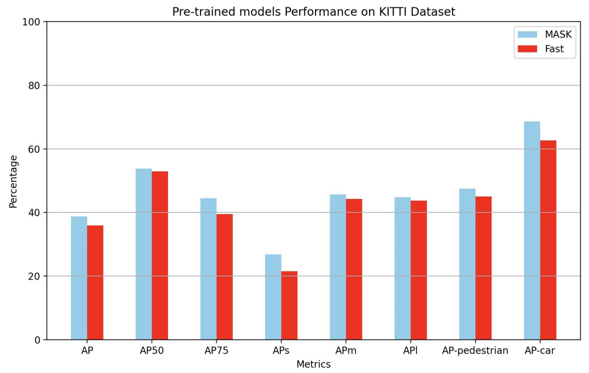


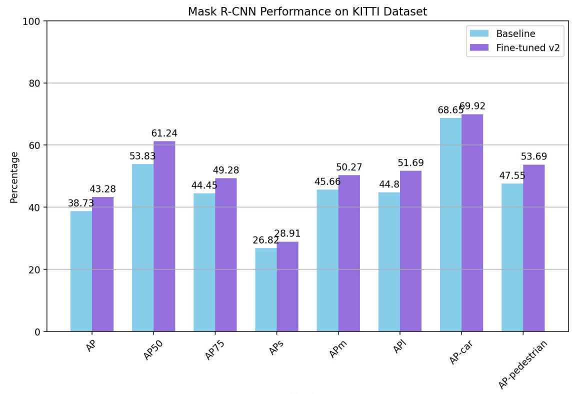
The model classifies a F1 car like an airplane with 97% of confidence, because it is upside down and over the people. This is a proof of the importance of the neighbour labels.

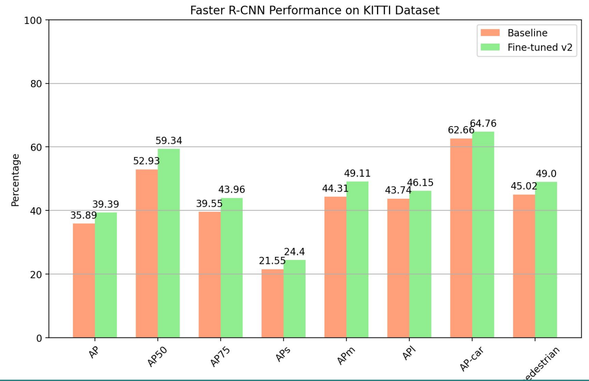
Summary slide

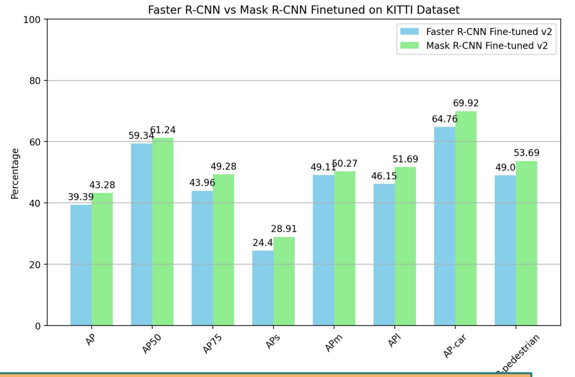
Results:

- We have obtained pretty acceptable AP values applying pre-trained models on COCO to our KITTY-MOTS dataset. (AP_{50}= 53%)
- Values obtained with MASK and FAST are very similar but MASK ones are a bit higher.
- With model fine-tuning we can improve a little our results. (AP_{50}= 61%)
- The hyperparameter selected are:
  - Max iters: 1500
  - Batch size: 32
  - Images per Batch: 16
  - Learning Rate: 0.02
  - Non-Maximum Suppression: 0.75
  - Segmentation Score: 0.25









Conclusions:

With the conclusion of our experiments we can confirm that the employment of a pre-trained model in new datasets can give us a pretty good performance results. Moreover we can improve these values with the application of a fine-tuning to the model, using part of our dataset as train data.
In terms of classification/segmentation results we have seen how the class pedestrian is harder to segmentate and classificate than car due to his more common shape.
Finally, with the out of context images we have shown the importance not only of the features of the object that we want to classify, but also the classes of the surrounding objects.