Ruhr-Universität Bochum
Autonomous Vehicles and Artificial Intelligence
Summer Term 2022

Assignment 3

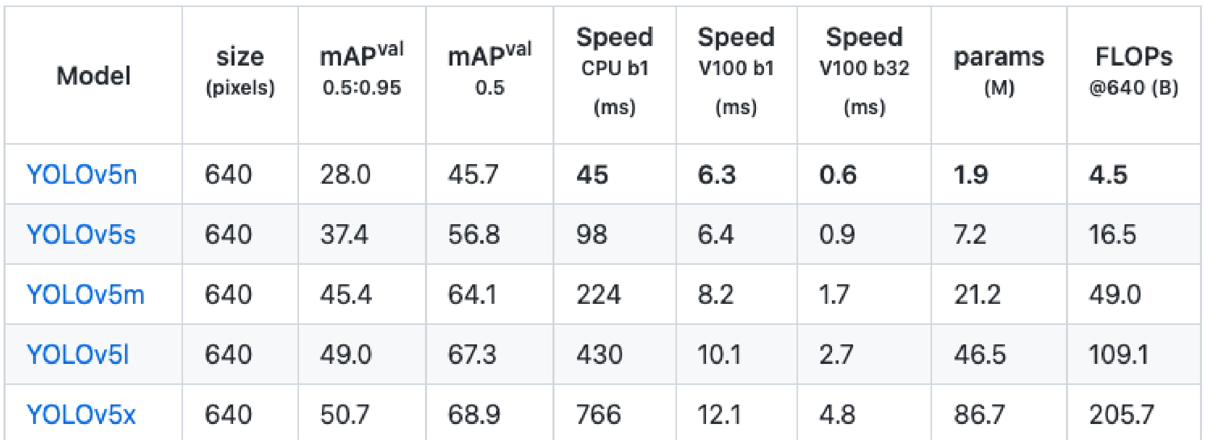
Perception: evaluation of the object detection model

Introduction

An object detection model was trained to detect bounding boxes for blue, yellow and orange traffic cones. The model will be used in the perception module of the turtlebot. In this report, the training process and performance of the trained model is documented and evaluated.

Training process

For good accuracy and high detection speed a pre-trained object detection model called YOLOv5s was chosen. The model is part of an open-source family of object detection architectures created by ultralytics. All models in that family are pre-trained on the COCO dataset and can easily be adapted to custom objects. The specific version of YOLOv5 was chosen based on the benchmark provided by ultralytics (Ultralytics, 2022). The S-version is the second fastest and second smallest model that processes 640 pixel wide images. It has a significantly better mAP score than the fastest model in the family which made it seem to be a good compromise of speed and performance.



Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

Benchmarks of the YOLOv5 model family (Ultralytics, 2022)

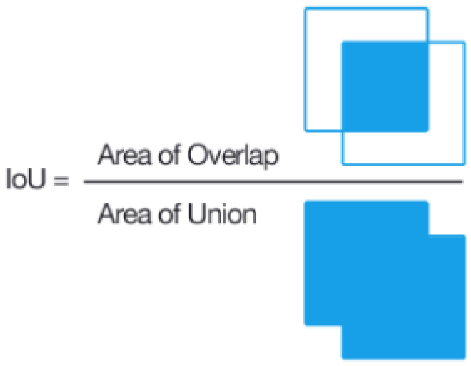
The training with custom images was achieved by using the YOLOv5 tutorial notebook hosted on Google Colab. A total of 250 images were split into training, testing and validation sets. The split ratio was 70% for training, 20% testing and 10% the validation set. These pictures were taken before with the turtlebots own camera and showed different settings of the three colored cones. The images have been labeled with the labeling tool Yolo_mark (AlexeyAB, 2019).

Mean Average Precision Theory

A common way to evaluate object detection models is by calculating a mean Average Precision (mAP) score. This is done by analyzing a new image set and its ground truth labels. The model performs inference on the images and the detected bounding boxes are stored. Then the Intersection over Union (IoU) is calculated for each detected bounding box. If the IoU is above a certain threshold, the bounding box is considered to be True Positive (TP) otherwise it's a False Positive (FP).

Calculating IoU:

- All detected bounding boxes are matched with one ground truth bounding box that shares the same label and has the highest overlapping area.
- The area of the intersection is divided by the area of union of the two bounding boxes.
- The ground truth bounding box which was matched with a detected bounding box is ignored in future matchings to prevent multiple detections of the same object.



IoU calculation visualized (Cartucho, 2019)

For the IoU calculation done in this report the detected bounding box is considered to be TP if the IoU ≥ 0.5.

Calculating Precision and Recall

By dividing the number of TPs with the total number of detected bounding boxes (TP + FP) the precision for a class can be calculated. By dividing the number of TPs with the total number ground truth bounding boxes (TP + FN) the recall value can be calculated.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

Formulas for calculating precision and recall (Khandelwal 2020)

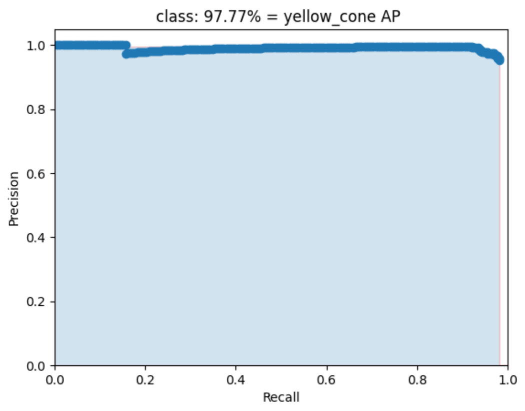
Calculation mAP

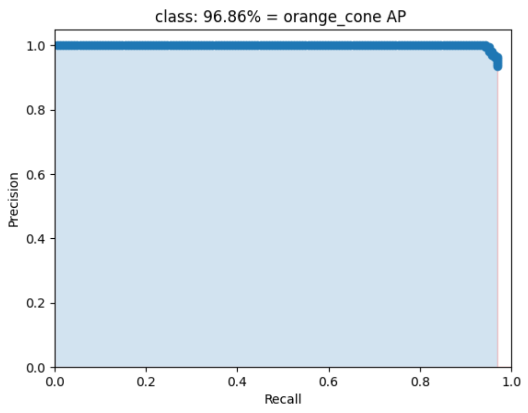
The calculation of precision and recall in this report was done on a per image and per class basis. All the Precision and Recall values are plotted on a precision-over-Recall curve where the values are sorted such that the precision values are monotonically decreasing (examples are in the following section). Then the area under the curve is calculated by numerical integration. The derived value is considered to be the mAP score, which can be calculated on a per class level or model level. The mAP score is between 0 and 1 and a higher score indicates a better performing model.

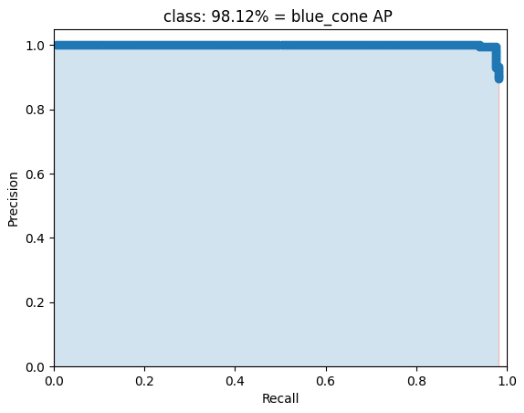
Results of the model evaluation

For the evaluation in the report, an open-source python implementation of the mAP scores calculation was used (Cartucho, 2019). The implementation uses the mAP criteria defined in the PASCAL VOC 2012 competition and outputs some visualizations. For the testset all the annotated images from group E were used, since they were not included in the training process. The output of the mAP calculation included the test images with the ground truth and detected bounding boxes on top. Some examples of these images can be found in the appendix of this document.

Precision over Recall curves per class:

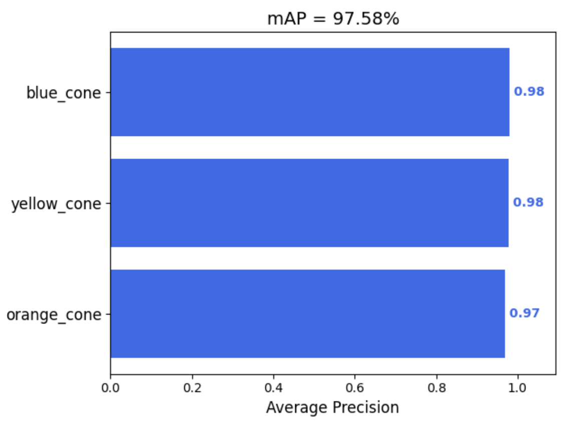


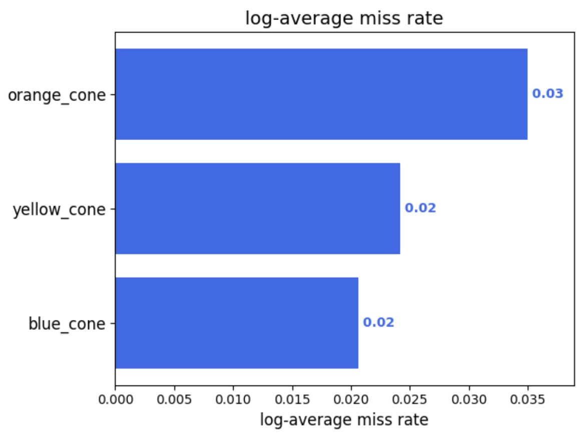


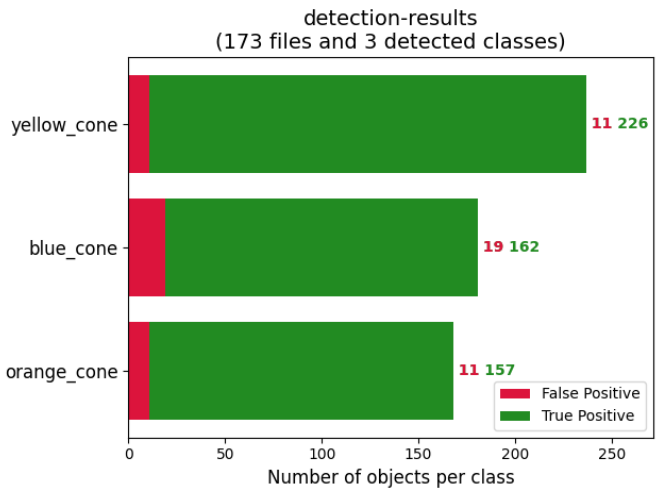


Detection Results:

As it can be seen in the chart called “Average Precision” the overall class mAP is 97,58% which can be considered to be good enough for the purposes of this project.

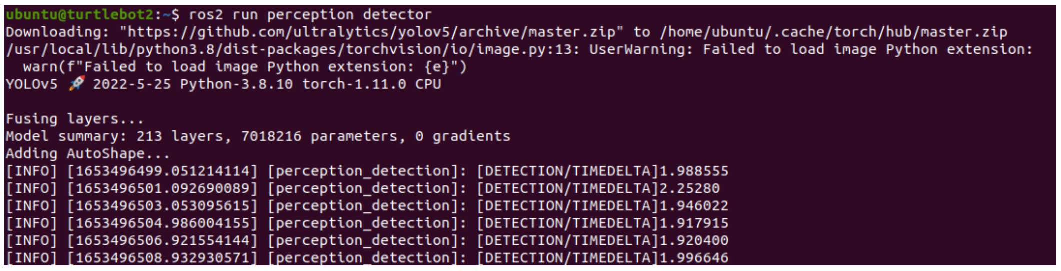






Running the model on the Turtlebot

The evaluation of the model was done on a laptop but it is assumed that the performance is the same on the turtlebot, except for the speed. In the test of running the model on the turtlebots hardware and analyzing a live video stream from its camera, an average inference time of about 2 seconds was achieved. The conclusion was derived by logging the inference time to the consol (see screenshot below)



```
ubuntu@turtlebot2:~$ ros2 run perception detector
Downloading: "https://github.com/ultralytics/yolov5/archive/master.zip" to /home/ubuntu/.cache/torch/hub/master.zip
/usr/local/lib/python3.8/dist-packages/torchvision/io/image.py:13: UserWarning: Failed to load image Python extension:
  warn(f"Failed to load image Python extension: {e}")
YOLOv5 🚀 2022-5-25 Python-3.8.10 torch-1.11.0 CPU

Fusing layers...
Model summary: 213 layers, 7018216 parameters, 0 gradients
Adding AutoShape...
[INFO] [1653496499.051214114] [perception_detection]: [DETECTION/TIMEDELTA]1.988555
[INFO] [1653496501.092690089] [perception_detection]: [DETECTION/TIMEDELTA]2.25280
[INFO] [1653496503.053095615] [perception_detection]: [DETECTION/TIMEDELTA]1.946022
[INFO] [1653496504.986004155] [perception_detection]: [DETECTION/TIMEDELTA]1.917915
[INFO] [1653496506.921554144] [perception_detection]: [DETECTION/TIMEDELTA]1.920400
[INFO] [1653496508.932930571] [perception_detection]: [DETECTION/TIMEDELTA]1.996646
```

Screenshot console output running the trained model on the turtlebot.

Challenges

The two main challenges were:

1. Training the model without Google Colab closing the session because of inactivity in the tab.
2. Getting the color coding straight. Converting the images from RGB to BGR fixed it.

Sources

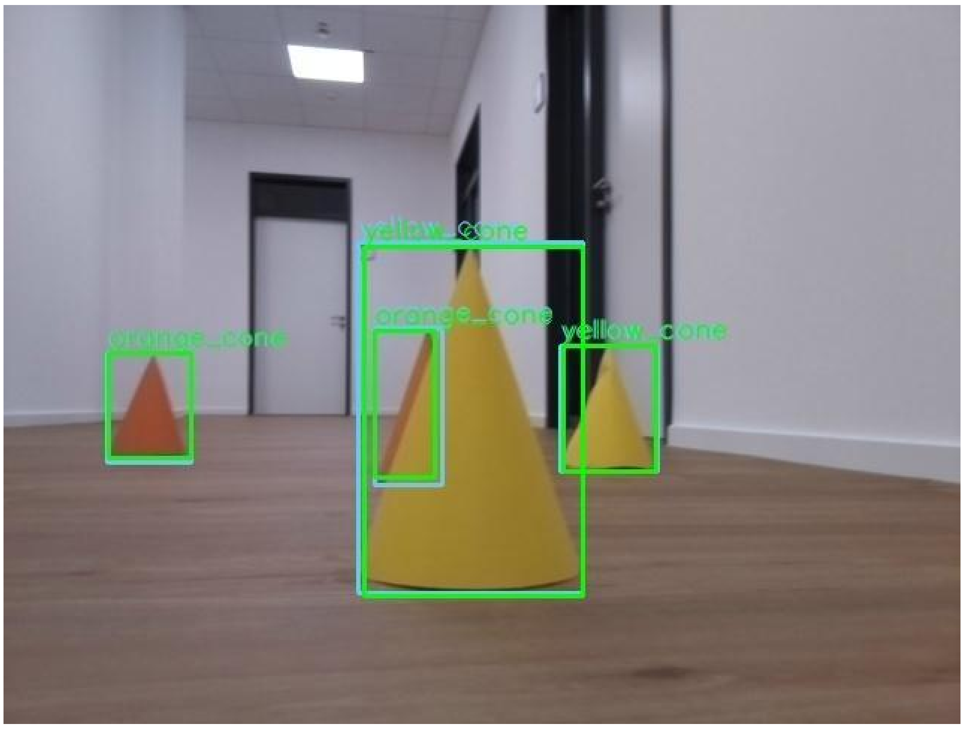
AlexeyAB. (2019). *AlexeyAB/Yolo_mark: GUI for marking bounded boxes of objects in images for training neural network Yolo v3 and v2*. GitHub. https://github.com/AlexeyAB/Yolo_mark

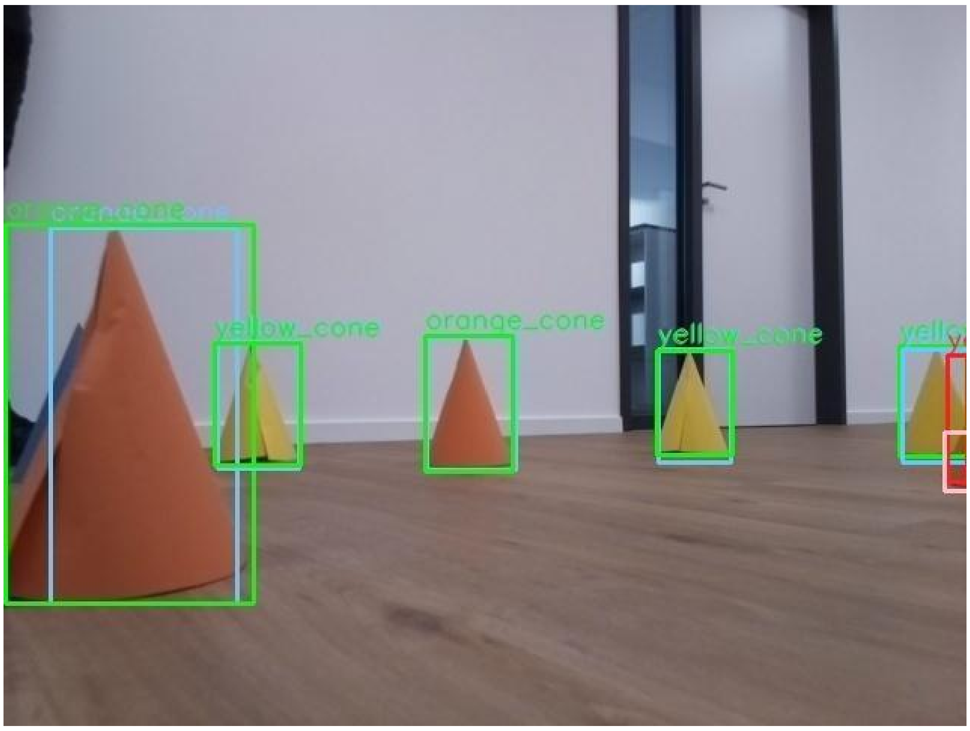
Cartucho, J. (2019). *mean Average Precision - This code evaluates the performance of your neural net for object recognition.* https://github.com/Cartucho/mAP

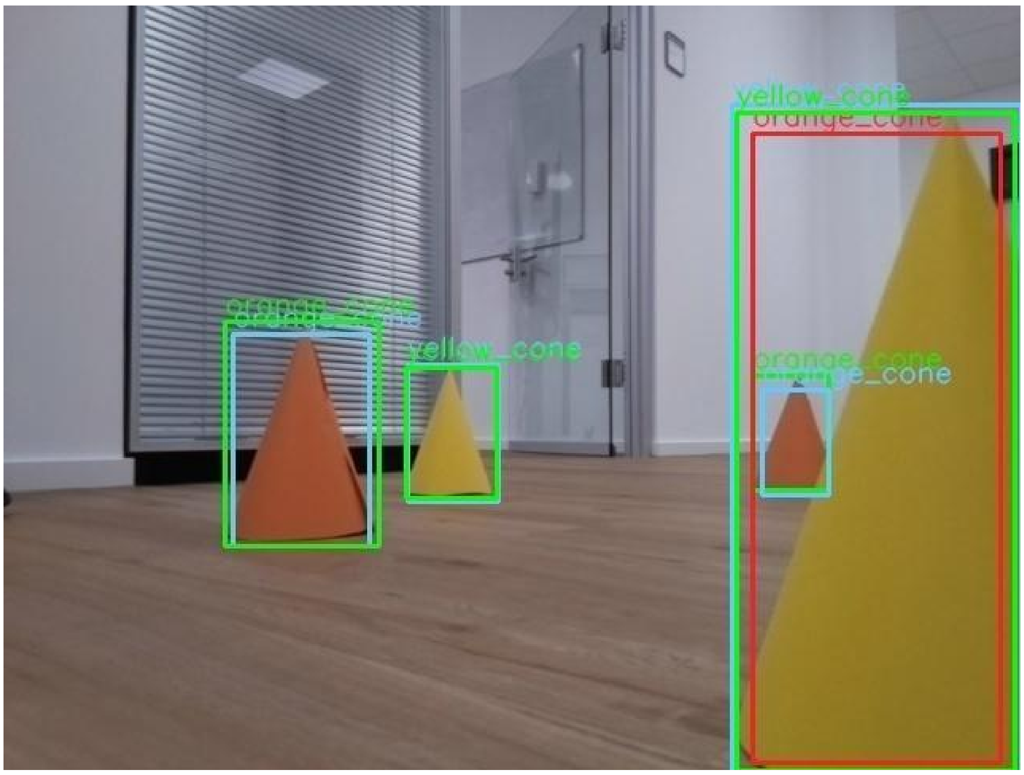
Khandelwal, R. (2020). *Evaluating performance of an object detection model | by Renu Khandelwal | Towards Data Science*. https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b

Ultralytics. (2022). *ultralytics/yolov5: YOLOv5 ? in PyTorch > ONNX > CoreML > TFLite*. https://github.com/ultralytics/yolov5

Appendix


yellow_cone
orange_cone
orange_cone
yellow_cone


orange_cone
yellow_cone
orange_cone
yellow_cone


orange_cone
orange_cone
yellow_cone
yellow_cone
orange_cone
orange_cone
orange_cone