TECHNISCHE UNIVERSITÄT BERLIN
Electrical Engineering and Computer Science
Distributed Artificial Intelligence Laboratory

Applications of Robotics and Autonomous Systems

System Design Specifications

Group 3: Active Learning for Traffic Light Detection

Aaron Attila Yaşar Limberg – 486217
Marco Barry – 486217
Servet Öz – 397981

4 January 2024

Contents

1 Introduction

Self-driving, autonomous cars have been an exciting prospect for the modern city and world in recent years, conceivably decreasing traffic flow, increasing safety, and allowing a more laid-back commute. The requirements for such systems are very high: safety, efficiency and eco-friendliness must all be able to compete with traditional cars, if not surpass them. Modern automotive engineering already combines and is at the forefront of many fields: mechanical components, electronic sensors and computer systems all cooperate to produce a system that moves from point A to point B, while detail is put into safety features to ensure rider safety. In modern vehicles these systems already work together to provide a comfortable driving experience, autonomous cars' goal is to transform the experience from being a driver to being a passenger. Therefore, the requirements for sensors (quality and number), computing power and capability of control algorithms are higher. It is evident that the complexity of the problems posed by placing a computer-controlled agent on streets in the physical world, not in any kind of simulated environment, is very high.

The problems themselves are not always clear and easily solved through traditional methods, as it is not feasible to predict every possible situation in the planning and programming stage of the creation of a control algorithm: deployment of algorithm will almost certainly encounter some novel cases. From navigation to motor control, object identification to follow rules and street code; there is a whole host of obstacles a self-driving vehicle must face, it is almost certainly necessary to decouple them and solve them one by one.

In this project the team will tackle an important part of road furniture and traffic law: traffic light detection. The problem is of great importance to correct vehicle behavior and is well bounded. The vehicle must be able to detect and identify a traffic light and identify which light is signaling (RED – YELLOW - GREEN). BeIntelli, in partnership with DAI-Labor and TU Berlin, has tasked the team to create an algorithm that runs on the hardware in their vehicles that can complete the above mentioned task in real time on novel data.

1.1 Problem Definition

The problem faced is the following:

- Suppose the vehicle is traveling on roads, obeying traffic regulations, and behaving as any other law-abiding vehicle on German roads.
- The vehicle is equipped with an array of sensors, including front facing RCCB cameras.
- When the vehicle approaches traffic lights, for example at an intersection or zebra-crossing, it must be able to detect these and identify the signal conveyed by the color and shape of the lights. It must be capable of this under all circumstances, i.e., all possible weather and light settings.
- Here lies the problem: how would an autonomous vehicle detect, identify, and process those as traffic lights, in real-time, based on inputs from the environment? Does it classify the color of traffic light correctly?
- Labeling data is time expensive. Therefore information on which samples are informative is required.

The team must develop a suitable algorithm for traffic light detection and identification, that runs in real-time on the vehicles hardware, which achieves acceptable results of detecting traffic lights in a real-world traffic setting (meaning it must detect all lights that regulate the vehicle's movement), while keeping the labeling time in a reasonable margin.

1.2 Objectives

By means of machine learning and deep learning techniques and proven Computer Vision Image detection algorithms, the outcome of this project should be a system that detects, identifies, and classifies traffic lights from novel (video) data, that can then be deployed on BeIntelli's vehicles. If the resulting project is to be also deployed in cooperation with other algorithms on the ROS framework for fully autonomous control of the vehicle, then it can be considered a resounding success.

Specifically, the tasks at hand are broadly:

- Literature review on object detection, active learning
- Choice of publicly available datasets, object detection algorithm, active learning method
- Set up object detection pipeline for publicly available datasets - obtain pretrained model
- Apply model to DAI Lab data and evaluate output
- Apply active learning method to data collected by cameras of BeIntelli cars and pre-trained model - obtain trained model
- Compare performance of trained models
- Apply best trained model in BeIntelli cars

1.3 Use cases

We aim to apply our Traffic Light Object Detection model inside cars of the BeIntelli fleet.

1. **Scenario: Clear and Sunny Day**
    - The autonomous vehicle approaches an intersection on a clear and sunny day.
    - The front-facing cameras capture the images of the traffic lights.
    - The algorithm analyzes the images and detects the presence of traffic lights.
    - The algorithm correctly classifies the color of the traffic light.
2. **Scenario: Rainy Conditions**
    - The autonomous vehicle is driving in heavy rain, which affects visibility.
    - Despite the challenging conditions, the front-facing cameras capture images of the traffic lights.
    - The algorithm processes the images, applying image enhancement techniques to improve visibility.

- It detects the presence of traffic lights and accurately identifies their colors, considering the modified image inputs.
- The algorithm successfully classifies the color of the traffic light.

3. **Scenario: Low Light Conditions**
    - The vehicle is traveling during nighttime or in an area with poor lighting conditions.
    - The front-facing cameras capture images, but they are relatively dark and challenging to interpret.
    - The algorithm utilizes low-light image processing techniques, such as noise reduction and contrast enhancement.
    - It detects the presence of traffic lights and accurately identifies their colors by analyzing the processed images.
    - The algorithm correctly classifies the color of the traffic light.
4. **Scenario: Adverse Weather Conditions (e.g., heavy snowfall)**
    - The autonomous vehicle is navigating through a heavy snowfall, which severely reduces visibility.
    - The front-facing cameras capture snowy images with snowflakes obstructing the view of the traffic lights.
    - The algorithm employs advanced image segmentation techniques to distinguish the traffic lights from the snowy background.
    - It processes the segmented regions to detect the presence and positions of the traffic lights.
    - By analyzing the visible portions and inferring the likely state of the obscured parts, the algorithm correctly classifies the color of the traffic light.
5. **Scenario: Changing Traffic Light Sequence**
    - The vehicle approaches an intersection with multiple traffic lights regulating different lanes.
    - The front-facing cameras capture images of the traffic lights and their respective positions.
    - The algorithm processes the images and detects all traffic lights, considering their locations and configurations.
    - It accurately identifies the color of each traffic light and tracks any changes in the light sequence.
    - The algorithm correctly classifies the colors of all the traffic lights.
6. **Scenario: Nighttime with Glare**
    - The autonomous vehicle is traveling at night, and there is a strong glare from oncoming headlights.

- The front-facing cameras capture images of the traffic lights, but the glare affects their visibility.
- The algorithm employs glare detection and reduction techniques to mitigate the impact of the glare on the captured images.
- It processes the enhanced images, detecting and locating the traffic lights.
- By analyzing the remaining visible portions and applying color analysis, the algorithm accurately identifies the color of the traffic lights.
- The algorithm successfully classifies the color of the traffic light despite the presence of glare.

7. **Scenario: Sun Glare on Traffic Light Lens**

   - The autonomous vehicle is driving during daytime, and the sun is directly shining on the traffic light lens.
   - The front-facing cameras capture images of the traffic lights, but the intense sunlight causes overexposure on the lens.
   - The algorithm applies exposure compensation techniques to balance the brightness levels in the captured images.
   - It processes the corrected images, detecting the presence of traffic lights and their color cues.
   - By carefully analyzing the color information in the processed images, the algorithm accurately classifies the color of the traffic light, despite the sun glare on the lens.

Misuse Cases / Unexpected Situations Scenarios:

1. **Scenario: Traffic Light Malfunction**

   - The traffic light at an intersection experiences a technical malfunction, causing it to display incorrect or inconsistent colors or signals.
   - The autonomous vehicle's cameras capture images of the malfunctioning traffic light.
   - The algorithm struggles to correctly detect and identify the color or signal of the malfunctioning traffic light due to its erratic behavior.
   - The algorithm detects this condition and give the control to the driver.

2. **Scenario: Dirty or Obscured Traffic Light Lenses**

   - The lenses of the traffic lights are dirty, covered with grime, or otherwise obscured.
   - The autonomous vehicle's cameras capture images of the dirty or obscured traffic light lenses.
   - The algorithm struggles to properly detect and identify the color or signal of the traffic lights due to the obstructed lenses.
   - The vehicle may encounter challenges in accurately interpreting the obscured traffic lights, potentially leading to incorrect driving decisions.

3. **Scenario: Power Outage or Electrical Failure**

   - A power outage or electrical failure occurs, causing the traffic lights to go off or display no signal.
   - The autonomous vehicle's cameras capture images of the non-operational traffic lights.
   - The algorithm detects the absence of signals but may not be able to determine the color or state of the non-functioning traffic lights.
   - The algorithm detects this condition and gives the control to the driver.

4. **Scenario: Extreme Weather Conditions**

   - The vehicle encounters extreme weather conditions, such as heavy rain, fog, or a severe storm.
   - The autonomous vehicle's cameras capture images of the traffic lights in the challenging weather conditions.
   - The algorithm faces difficulties in accurately detecting and identifying the color or signal of the traffic lights due to reduced visibility or image distortions caused by the extreme weather.

5. **Scenario: Intentional Traffic Light Disregard**

   - The autonomous vehicle's system detects and identifies the traffic light correctly.
   - However, the driver intentionally overrides the system's instructions and disregards the traffic light signals.
   - The vehicle proceeds through the intersection against a red light, risking collisions or violating traffic regulations.
   - The system may generate alerts or warnings to the driver, but the driver intentionally ignores them, jeopardizing safety.

1.4 General constraints

The constraints of the project are that the system should be able to function on Berlin roads, under many weather and light conditions. If some conditions do not meet performance requirements (which are very high for safety reasons), for example it is not possible for night time functionality, this is acceptable and allows room for further development in future works.

It is important to note that there are constraints on limited resources, both in terms of data and time. There is limited training time and processing power for developing Networks, so a limited number of Networks and/or epochs are allowed. In terms of data, there is limited quantity of quality labelled data suited to the application (the dataset selected may also be unbalanced), so steps to counter this are time intensive and, therefor, limited.

Furthermore, deployment as ROS node is to be considered a stretch goal as the main goal is to build a functioning model for Traffic Light Detection.

2 System Requirements, Architecture and Specifications

2.1 Initial Requirements

The following are the requirements which were thought to be crucial for the project. Things evolved and changed, here is the summary of what we expected to what we got.

- The algorithm should be able to run in real-time on the vehicle's hardware, ensuring timely processing and response. → **Tested and functional**
- The algorithm must accurately detect and locate traffic lights in a real-world traffic setting, including intersections and zebra crossings. → **It detects and locates traffic lights well, although some flickering is still visible**
- The algorithm should be able to handle various weather and lighting conditions, such as rain and nighttime. → **Very limited weather conditions, no nighttime functionality (functions badly)**
- The algorithm must correctly identify the color of the traffic lights, distinguishing between red, green, and yellow signals. → **It definitely is able to identify the main colours, although directional colours and full colour are often confused.**
- The algorithm should be robust to occlusions caused by other objects, such as overhanging branches or large vehicles. → **Seemingly acceptable perfomance, training data includes many occluded traffic lights**
- The algorithm should adapt to unconventional traffic light placements or orientations, correctly detecting and identifying them. → **Not enough testing, might have difficulties with traffic lights not placed vertically.**
- The algorithm should minimize false detections or misclassifications, ensuring reliable and accurate results. → **Acceptable results**
- The algorithm must be optimized to improve detection accuracy and reduce processing time. → **Processing time is very good as the smallest possible YOLO model was utilised, detection accuracy is improvable but has had a great improvement compared to initial network**

2.2 System Architecture

Section taken from first document, as the system architecture has remained the same, with difference being that 3 different active learning approaches were utilised that all use different strategies and algorithms to try and improve the dataset (thus also performance).

The task at hand is, mainly to train and develop a system that can later be applied to an already working autonomous vehicle as a ROS node. Therefore the main system architecture is the development diagram of the system.

The process can be divided into 3 main stages:

- Train YOLO on the Bosch Small Traffic Light Dataset, get the best possible result on this trained YOLO network

- Then use this very same Network and apply an Active Learning paradigm on it. This is possible by leveraging the integrated confidence score YOLO applies to its predictions. So in essence, the team would be using an uncertainty query sampling strategy already available to all YOLO networks and then manually label the best (most uncertain) examples.
- The new dataset produced by the process will then be used to train the network again. Conceivably, the precision and recall of the new network on the test data should improve and so we have an improved system via Active Learning.

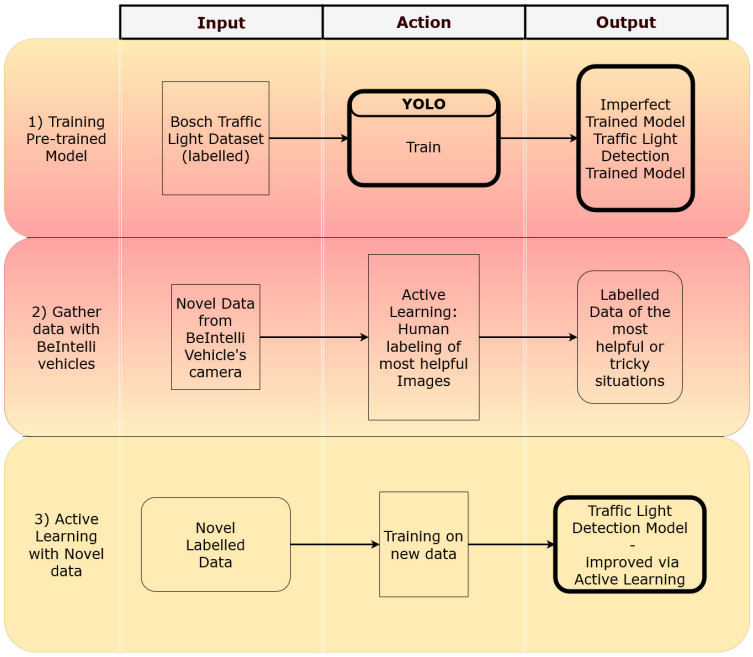

Figure 1: System architecture divided in three phases: 1. Initial training 2. Active Learning (labelling) 3. Active Learning (training on newly labeled dataset)

Once we have a working system, it will be used in the wider ROS achitecture of autonomous vehicles, which in the case of this work will be BeIntelli's vehicles. So in a generic ROS autonomous vehicle architecture, our node will replace the Traffic Light Detection algorithm.

2.3 System Specifications

When talking about system specifications, the scheme to consider is the system architecture represented in Figure 1.

Training Pre-trained Model

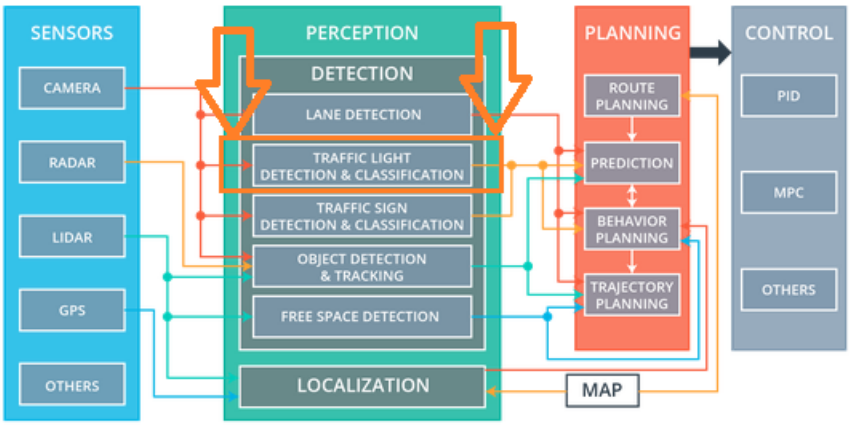

Figure 2: Generic ROS system architecture of autonomous vehicle. The highlighted node in the Perception box is the system that is of interest in the present work. (Image taken from: [https://taylor.raack.info/2018/08/autonomous-vehicle-technology-system-integration-with-ros/] and then edited)

- **Input**: Dataset to train and complete the model; this dictates the labels, largely the quality of the network and generally the diversity the network is able to comprehend. The selected Dataset is the Bosch Small Traffic Light Dataset [1], because it allows to recognise,the colours, on or off traffic light and different classes for direction signalled by traffic light.
- **Action**: Train YOLO [2] on the input dataset. This is done by feeding it the dataset and using "train.py" Python script in the YOLO repository to yield a trained Network.
- **Output**: At the of the first action, the training will yield a primitive and not very effective DNN that can identify Traffic Lights, but not to a high enough standard.

Gather data with BeIntelli's vehicles and label the data

- **Input**: The inputs for the second action will be the trained Network from the previous step and novel data from BeIntelli's vehicles (using video data from BeIntelli vehicles being that it is data from the sensors on which the application will actually be running on).
- **Action**: Uncertainty sampling. The idea is to use the uncertainty score of YOLO predictions (functionality already available in YOLO networks) to select the most uncertain Traffic Light predictions by the trained YOLO network. These will then be labelled by a human oracle. Something else to note, is that not only will it be necessary to manually label the most uncertain examples, but also a naive dataset will be required as to compare results with the Active Learning approach (i.e. manual labelling of dataset from scratch without any filtering or selecting of the most informative examples)

- **Output**: The output of this process will be two labeled datasets from the novel BeIntelli data, one naive dataset and another Uncertainty-based Active Learning Dataset.

Acive Learning with Novel data

- **Input**: Labelled Active Learning Dataset and the trained YOLO Network
- **Action**: The labelled active novel data is used to train the Neural Network again (thus completing the Active Learning cycle).
- **Output**: The output is be a YOLO network that has been further trained on more (and probably more informative data). Consequently, the final network can be compared to the initial trained YOLO network (the resulting one from process 1.) and even with a network further trained with the naive dataset.

2.4 Dataset

During the development process two datasets were utilised, the first of which freely available on the internet and the second one made available by *BeIntelli* that the team was given access to.

2.4.1 Bosch Small Small Traffic Lights Dataset

The *Bosch Small Traffic Light Dataset*, made available by Heidelberg University, is a collection of 13427 images, resolution of 1280x720, that contains 24000 annotated traffic lights. Both 12-bit HDR source and converted RGB data is present, but for the purposes of this project only the RGB data was used.
The data is divided into two sets: training set and testing set with different properties.

	# Images	# Traffic Lights	Annotation Freqency	# Classes
Training Set	5093	10756	0.5 /s	15 (red [+directionals left/straight/right/combinations], yellow, green [+directionals], off)
Testing Set	8334	13486	15 /s	4 (red, yellow, green, off)

2.4.2 BeIntelli Training and Validation Set

BeIntelli Dataset is taken from four different cameras; 2 forward facing (one of which with higher resolution), one left facing, and one right facing. The camera's resolution are 960x640 and 1920x1080. In total, 31512 unlabelled images make up the dataset. An important difference to the Bosch Dataset, is the the images taken by BeIntelli are from the same driving track (Brandenburg Gate area roads in Berlin) as the validation set, also provided by BeIntelli.
The validation set provided is a set of 200 handpicked and hand-labelled unique images, which containing all classes and varied situations and camera angles.

3 Methods

3.1 Pretraining

Before we did Active Learning, we trained YOLOv8n on the data available in Bosch Small Traffic Lights Dataset. The precision-recall curve of the pre-training can be seen in 3.

3.2 Active Learning

Each of the group members did five cycles of Active Learning with one query strategy. One cycle was defined as follows:

1. The query stategy decides for 200 images that will be labeled and added to the dataset.
2. The model from the previous cycle will be used to train on the new and old images of the dataset, but not on data from the Bosch Small Traffic Lights Dataset. YOLOv8n will be trained for a maximum of 200 epochs.

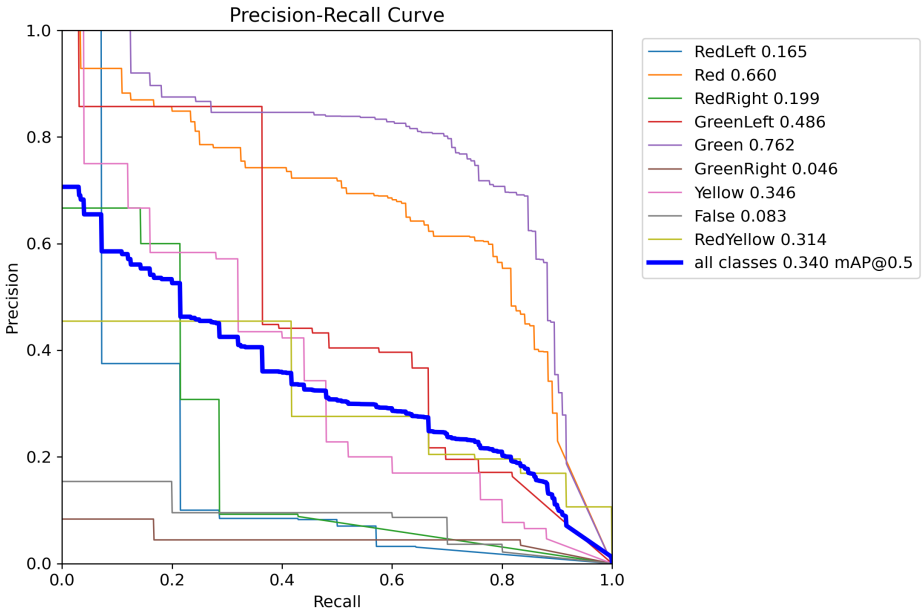

Figure 3: Precision-recall curve of YoloV8n trained on Bosch Small Traffic Lights Dataset.

3.2.1 Random Sampling

Random Sampling was our control method. In each cycle, 200 samples were picked randomly from the BeIntelli data.

3.2.2 Uncertainty Sampling

Samples were chosen according to the uncertainty of the current Yolo model. Uncertainty means that the model assigns one instance of a traffic light multiple classes with similar probability. Gaussian Uncertainty prediction attributes a value on the prediction uncertainty compared to other classes: for example for edge or novel scenarios a prediction may or may not be correct, but it will have a high uncertainty. As seen in this paper [3], it possible to treat the loss function not only as an error of the ground truth and the predicted truth, but more importantly from the ground truth and a Gaussian prediction, where the uncertainty measures where in the distribution the current prediction lies. This allows to apply an uncertainty score to predictions (with minimal performance loss).

3.2.3 Color Difference Sampling

The idea behind Color Difference Sampling is to find as diverse samples as possible. Therefore we search for the samples, that differ the most from the ones we have in the dataset already. One component of every object is it's color and for Traffic Light Detection the color is particularly important. Also samples with a different color than typical for a class can reveal instances that the current model labels wrongly. Therefore it was chosen as a query strategy.

For Color Difference Sampling first all images and labels from the Training Set were taken and an empty list was initialized for every class. For all instances of traffic lights in the already existing training data the pixel values were stored in the list that corresponds to the class of the traffic light. Then the average RGB values of the already existing training data were calculated separately for each class.

Then all the data that wasn't yet labeled was taken. The current Yolo model was applied to it. For every found instance of a traffic light in the new data, the average RGB value was calculcated. Then the squared distance from the average RGB value from the training set to the average RGB value of the instance in the unlabeled data was calculated and stored.

The 200 images containing the instances with the largest distance were chosen to be labeled.

3.3 Image Selection for Active Learning

Edge cases in Uncertainty Sampling

Figure 3 shows some edge cases in uncertainty sampling. These three traffic lights are noticed by the human labeler during labeling. These anomalies can be the result of inconsistency and difference between the shutter speed of the cameras and the flicker frequency of the LEDs in the traffic light. There are also the prediction scores by the model under each sample. These unexpected states of traffic lights are also difficult to label by human labeler and it might be beneficial to remove these images from the dataset to confuse the model less during training. However, all these images are kept in the training dataset and labeled according to the highest score provided by the model. In addition, these selected images also show that the main goal of uncertainty sampling is achieved by selecting the most confusing images not only for the model but also for humans.

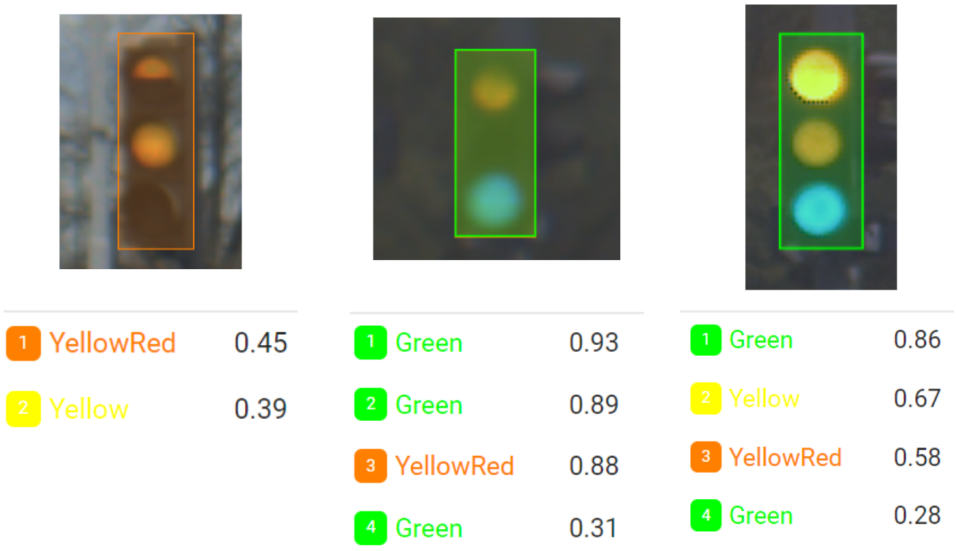

Figure 4: Edge cases in uncertainty sampling

3.4 Labelling Selected Images

Labelling selected images was done using Label Studio. The Label Studio Machine Learning Backend was set up to work with our YoloV8n model to ease the labelling. Images were loaded into the program, which made a prediction for every image based on the loaded Yolo model. Most of the predictions were correct and could be used as they were, sometimes a prediction had to be added, deleted or disambiguated (if there were multiple class predictions for the same traffic light instance).

4 Results

4.1 Random Sampling

Random Sampling as the control method was used to create 5 batches of 200 images, that were iteratively added to new "Active Learning" dataset to train the network. Thus on every iteration the best network of the previous iteration is trained on an expanded dataset. As expected, more data means improvements to the network. As seen below in 5. What

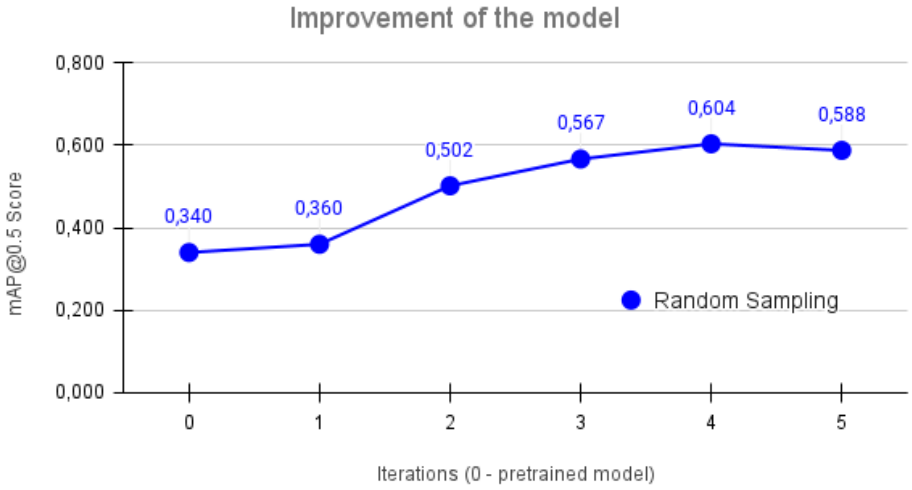

Figure 5: Improvement with control method (Random Sampling)

is interesting is that the mAP 50 score (mAP: (mean Average Precision) is an evaluation metric used in object detection models such as YOLO.[4]) improves steadily until the 5th iteration where there is no improvement, even though it was trained on 200 epochs. This may be due to an overrepresentations of a class leading to deterioration in the detection of the class itself. Following are Precision-Recall curves of the last two iterations, to show the difference in performance.

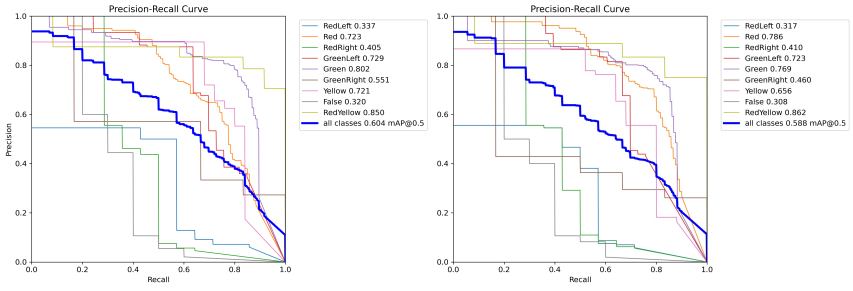

Figure 6: PR curves for iteration 4 [left] and 5 [right]

4.2 Uncertainty Sampling

As explained in the previous section, the training sessions are conducted on five occasions, each comprising 200 epochs, with an image size of 1280 pixels. The images are selected by comparing to class probabilities, with the utilization of an uncertainty sampling approach. Figure 13 shows the improvement of the model over all iterations. The zero iteration in the graph shows the confidence score of the pre-trained model for all classes. The model's performance is very stable, doubling until the fourth iteration, then dropping slightly from 0.656 to 0.633 in the fifth.

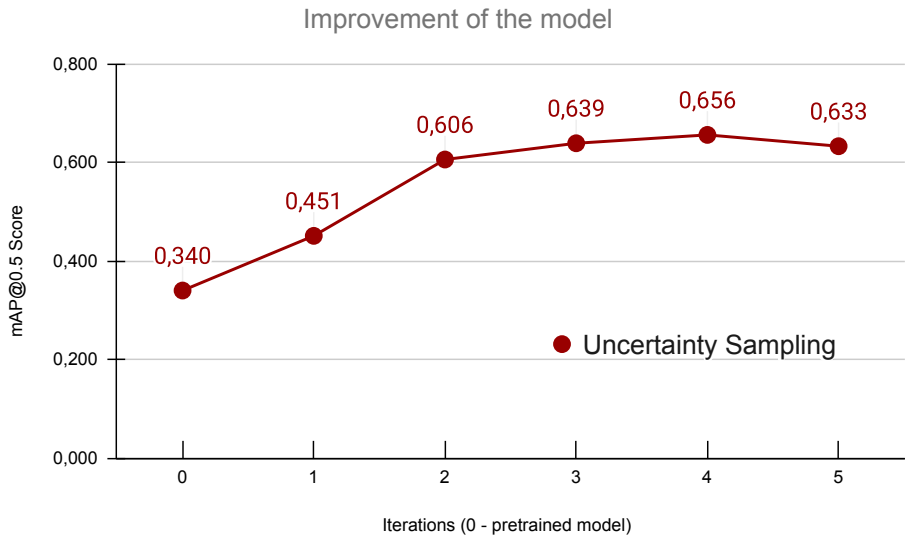

Figure 7: Improvement of the model using uncertainty sampling

The precision-recall curves of the fourth and fifth iterations are shown in Figure 8. The GreenRight predictions drop significantly from 0.80 to 0.59 in the fifth iteration. There is no significant difference in the other classes.

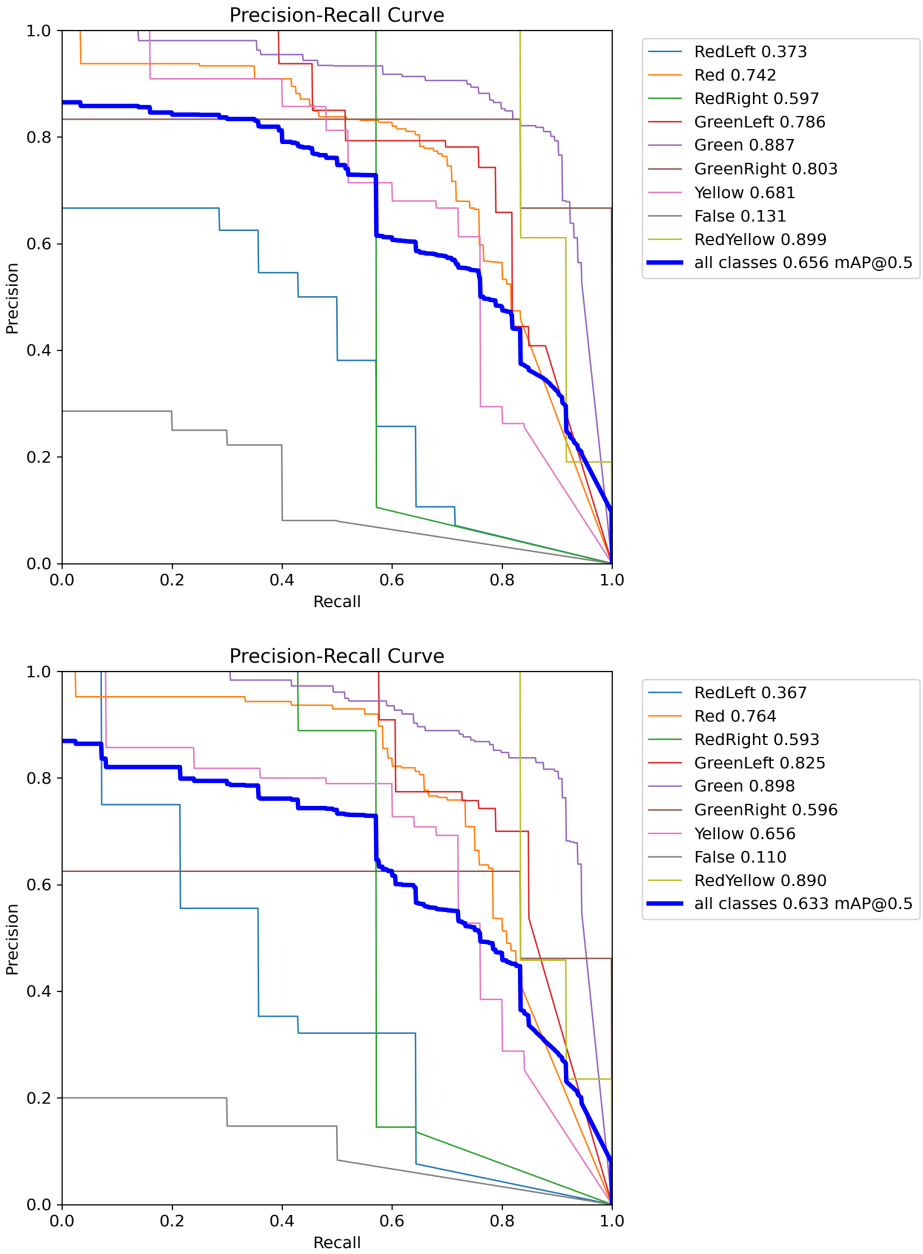

Figure 8: Precision-recall curves of the fourth and fifth iterations. (Top: forth iteration, Bottom: fifth iteration)

To address this degradation in performance, a few selected images are shown in Figures 9 and 10. There are many images containing this scene (see Figure 9) in the fifth iteration.

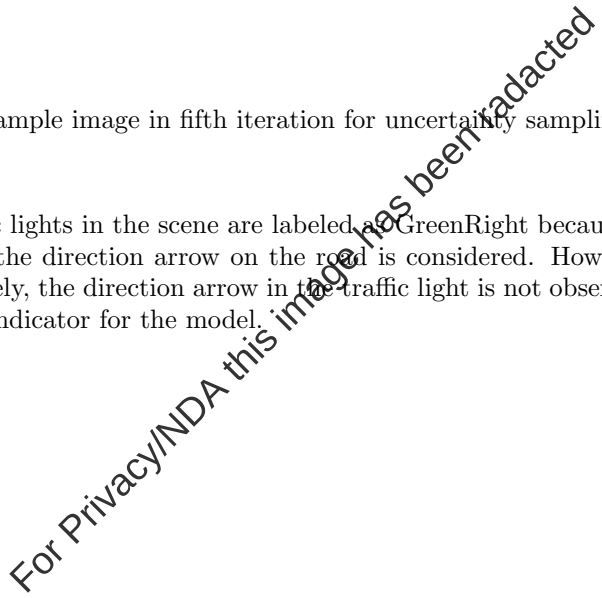

Figure 9: Sample image in fifth iteration for uncertainty sampling (far)

The marked traffic lights in the scene are labeled as GreenRight because it is obvious to the human labeler if the direction arrow on the road is considered. However, if the traffic light is examined closely, the direction arrow in the traffic light is not observable. Therefore, there is no direction indicator for the model.

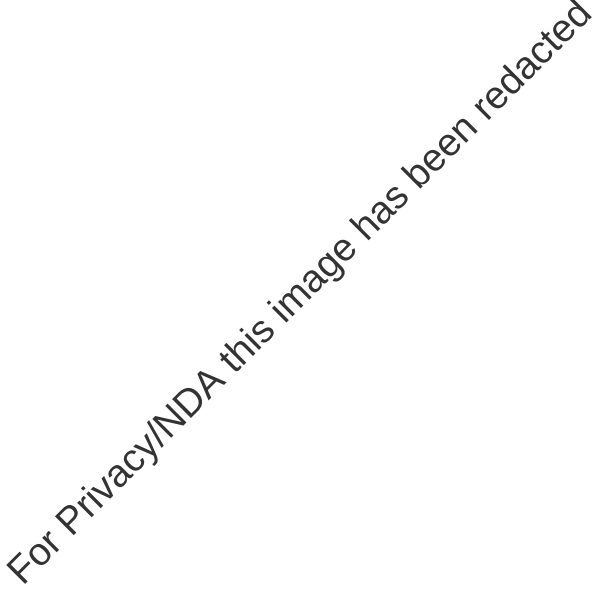

Figure 10: Sample image in fifth iteration for uncertainty sampling (closer)

When the vehicle approaches the traffic light in Figure 10, the direction arrow in the traffic light is visible. This shows us that it is important to label lights without direction from a distance, and to include directions only when the arrow in the traffic light is visible. For a future project, it may be beneficial to specify a labeling strategy in advance.

4.3 Color Difference Sampling

Color Difference Sampling did lead to very imbalanced classes: while there are 1297 Green traffic lights after the last Active Learning cycle, there are only 161 Yellow, 47 Red, 12 RedRight, 10 GreenLeft, 8 GreenRight, 7 False, 0 RedLeft and 0 RedYellow traffic lights in the data chosen by this query strategy. 0.84% of the labels are Green. Why so many Green samples were chosen is not clear.

Regarding the overall performance, with Color Difference Sampling the model improved slightly every cycle, as can be seen in 11. Still, Color Difference Sampling led to worse results than Random Sampling, which might be explained by the high class imbalance.

The precision-recall curve after the last Active Learning cycle can be seen in 12. When looking at it, one can see that the regular Traffic Lights without arrows were detected with an mAP@0.5 of: 0.677 (Red), 0.663 (Yellow) and 0.719 (Green). Classes with particularly bad low mAP@0.5 values are: RedLeft (0.197), GreenRight (0.043) and False (0.167). The other classes lie in between the two groups and have mAP@0.5 of RedRight (0.441), GreenLeft(0.565) and RedYellow(0.343)

When comparing the precision-recall curve of the fifth Active Learning cycle of Color Difference Sampling to the precision-recall curve of the pretraining one can see that for most classes there is only a slight increase in performance of maximally 0.1 (measured by mAP@0.5 value). This holds for: RedLeft (0.165 to 0.197), Red (0.66 to 0.667), GreenLeft (0.486 to 0.565), False (0.083 to 0.167), RedYellow(0.314 to 0.343). For two classes there was a slight decrease in performance when comparing mAP@0.5 value: Green (0.762 to 0.719),

GreenRight (0.046 to 0.043). Only for two classes there was a considerable improvement: RedRight (0.199 to 0.441), Yellow (0.346 to 0.663).

The improvement in detecting Yellow Traffic Lights might be explained by the rather large number of Yellow traffic light instances in the dataset, at least compared to the other classes except Green.

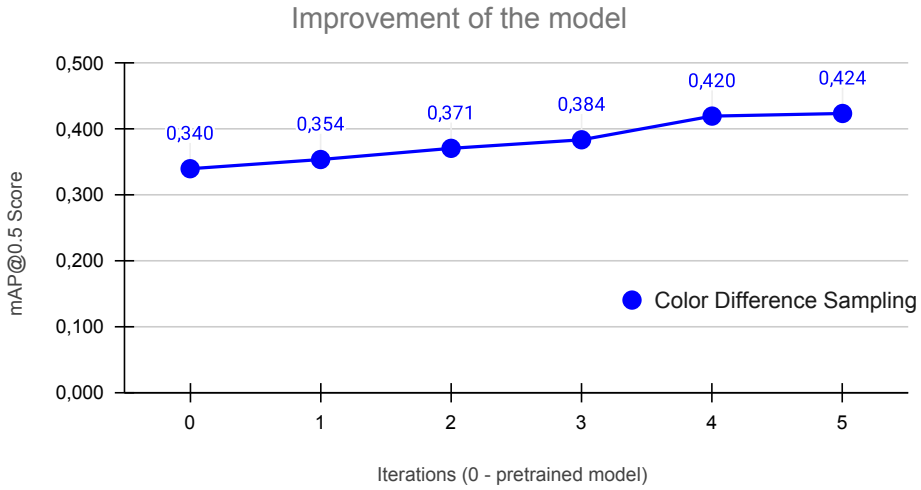

Figure 11: Improvement of the model using Color Difference Sampling

Color Difference Sampling failed at sampling data from different classes. Therefore an alteration would be suggested, which tries to sample equally many samples from each class and for each class chooses the ones with the largest color difference. Similar could be done for the Uncertainty Sampling. This could help whenever there is a class imbalance, such as it exists in the traffic light datasets.

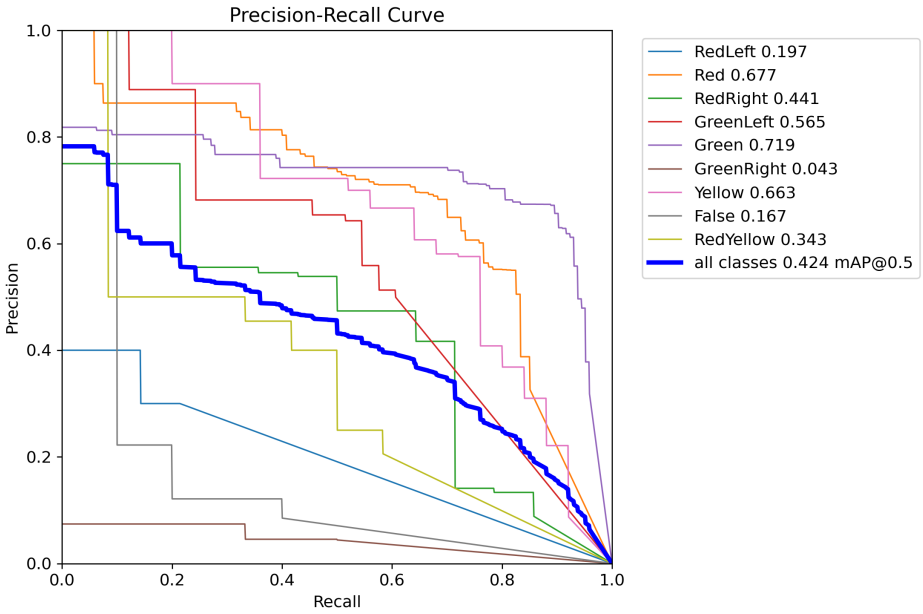

Figure 12: Precision-recall curve after the fifth (last) Active Learning cycle of Color Difference Sampling.

5 Conclusion

It is obvious that Uncertainty Sampling led to best performing Object Detection model. In fact, the best performing version of this Active Learning paradigm was tested and used on a BeIntelli vehicle and seemed to have a fairly good performance. It is important to note that to properly use the model, then the YOLO command *detect* must be accompanied with $imgsz = 1280$.

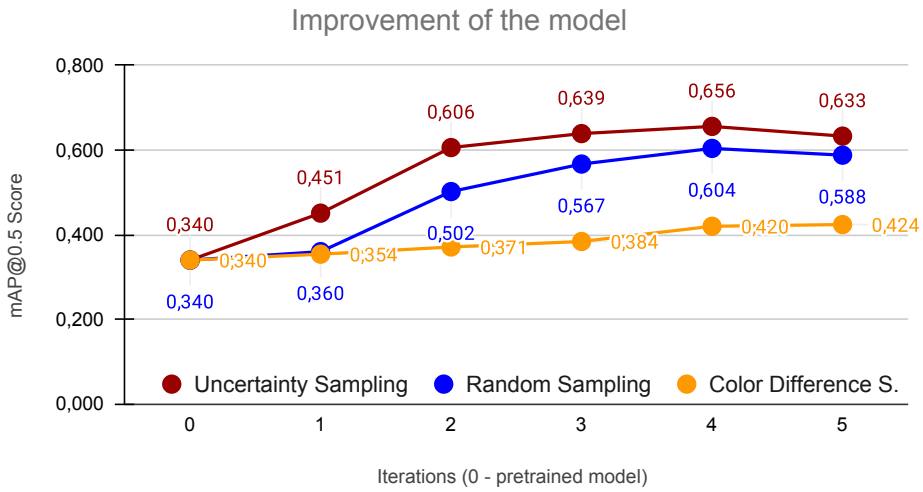

Figure 13: Improvement of the model all methods combined

There is definitely room for improvement and experimentation. One good example of this is to train on the the entirety of datasets available and not only the datasets made through Active Learning, maybe producing a more generalizing network at the cost of more computing time which was a constraint of this work. More possible improvements is to cascade a sharpening method (such as bilateral filter) to the training/detection, improving details in the image such as arrows of traffic lights. One further active learning method was proposed and this was k-clustering and using cluster-mean difference, but was not used during the project. This may be a future work and path to follow. Another definite help for future works would be to clearly define and write down a labelling process and standard in order for all Labelers to have as close as possible labelling results.

In summary, a system has been developed that can detect, locate and classify different traffic lights on the BeIntelli driving test-course. This has been achieved via multiple proposed and previously detailed Learning Algorithms, one of which seems to be more successful than the control method, although to further improve performance is still a question mark that hasn't quite been answered. The system runs on car hardware, is sufficiently effective in tests and can probably handle cloudy and sunny weather conditions.

Active learning seems to be anle to reliable and efficiently improve YOLO model performance as long as a well-thought out strategy is employed, which is not immediately obvious (as was the case for colour-mean labelling.

6 Frameworks and Tools

- Ubuntu 20.04 or 22.04
- Python
- Pytorch

- YOLOv5/v8
- ROS
- OpenCV
- Processing System: GPU Cluster of TU Berlin, Google Colab, Windows Desktop (CUDA enabled)

References

[1] Karsten Behrendt and Libor Novak. A deep learning approach to traffic lights: Detection, tracking, and classification. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE.

[2] Glenn Jocher et. al. ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support, October 2021.

[3] Florian Kraus and Klaus Dietmayer. Uncertainty estimation in one-stage object detection. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, oct 2019.

[4] Medium.com. map : Evaluation metric for object detection models. 2021.