



**TECHNISCHE UNIVERSITÄT BERLIN**
**Electrical Engineering and Computer Science**
**Institute of Commercial Information Technology and Quantitative Methods**

**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 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.
    - Despite the challenging conditions, the algorithm leverages deep learning models trained on diverse weather datasets to accurately identify the color 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.

8. **Scenario: Traffic Light Occluded by Other Objects**

- The autonomous vehicle approaches an intersection where the traffic lights are partially obscured by other objects, such as overhanging branches or large vehicles.
- The front-facing cameras capture images of the intersection and the partially occluded traffic lights.
- The algorithm employs object detection and segmentation techniques to identify the presence and location of the traffic lights behind the occluding objects.
- It processes the segmented regions and extracts color cues to accurately classify the color of the traffic lights.
- By considering the partially visible portions and inferring the obscured areas, the algorithm makes appropriate driving decisions based on the determined traffic light color.

**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.
   - If the confidence of the algorithm lies under specified threshold, it gives the control to the driver.
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.

6. **Scenario: False Triggering of Emergency Stop or Parking Brake**
   - The autonomous vehicle's emergency stop system is designed to engage in critical situations to ensure safety.
   - The driver trigger unnecessary emergency stops or pull parking break.
   - The system detects the false emergency stop signals but is unable to differentiate between genuine and intentionally created emergency situations.
   - The driver's misuse of the emergency stop system disrupts the flow of traffic, causes inconvenience to other drivers, and potentially leads to accidents.
7. **Scenario: Inattentive Driver during Handover Process**
   - The autonomous vehicle's system requires the driver to be attentive and ready to take over control when prompted.
   - The driver misuses the system by becoming distracted, engaged in non-driving-related activities, or falling asleep during the handover process.
   - The system alerts the driver to take control, but the driver fails to respond in a timely manner, resulting in delayed or improper driver intervention.
   - The vehicle may deviate from its intended path or fail to respond to changing road conditions, increasing the risk of accidents due to the driver's inattentiveness.
8. **Scenario: Tampering with Sensor Inputs**
   - The autonomous vehicle relies on various sensors, such as cameras, lidar, or radar, to perceive the environment accurately.
   - The driver intentionally covers or manipulates the sensors to provide false or distorted information to the vehicle's perception system.
   - The system receives misleading sensor inputs and may misinterpret the surroundings or make incorrect decisions based on the tampered data.
   - The driver's tampering with the sensor inputs compromises the vehicle's ability to navigate safely and increases the risk of accidents.

## 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 State-of-the-Art

## 2.1 Neural networks and Deep Learning

Traditional computing and programming where the problem is well defined and it is possible to divvy up the problem into smaller, more manageable problems is a tried and tested method that has developed alongside the growing processing power of digital computing systems. A slightly more recent paradigm of solving problems, born with the Percepetron in 1958 [1], is to feed the problem to the system and letting it self-organise and create a valid solution (quasi) autonomously. This approach is called machine Learning and the Perceptron is most elementary building block of Artificial Neural Networks.

In recent years, machine learning and, especially, deep learning has been gaining more and more traction, the second of which becoming popular in circa 2012, could only be developed by experts of the field often with backing of larger entities such as Academia or Corporations. The popularity if the technology has slowly brought the field to be grasped by more and more people as the requirements for development have been simplified, for example with Transformer Learning and pre-trained Models, and good data is more readily available, but also preparing data has become simpler. When speaking of deep learning it is important to note that it is a subset of Machine Learning, that is based on "Deep" Neural Networks, meaning an Artificial Neural Network with many layers.
"Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics." [2]

Deep Neural Networks (DNNs) offer several advantages over other types of AI models, making them a preferred choice in many applications. One key advantage of DNNs is their ability to learn and extract intricate patterns and features from large datasets. A strength of DNNs is their scalability. Additionally, DNNs can handle high-dimensional data effectively, making them suitable for tasks involving large-scale datasets or complex input spaces. With advancements in parallel computing and the availability of powerful GPUs, DNNs can be trained on vast amounts of data, leading to improved accuracy and generalization. Also, unlike traditional machine learning algorithms that rely on handcrafted features, DNNs can automatically learn hierarchical representations, enabling them to capture complex relationships in the data. This makes DNNs highly adaptable and capable of achieving superior performance in tasks such as image recognition, natural language processing, and speech synthesis. In addition to the intrinsic flexibility of Deep Neural Networks, transfer learning allows for further and more easily adaptation of already trained models to novel data and obstacles. In short, it is possible to exploit pre-trained models to create a custom made network without going throught the hassle of creating an entirely new pipeline from scratch thus resulting in lower amount of data requirements. Overall, the flexibility, scalability, and ability to automatically learn complex patterns make DNNs a potent and versatile tool, surpassing other types of AI models in many domains.

### 2.1.1 Convolutional Neural Networks

A particular subset of DNNs that are of greater interest to the proposed application of this paper are Convolutional Neural Networks (CNNs). These networks are useful to image analysis because they take into account distance and adjacency relation, as a generic DNN does not. This means grid-like data, such as images and raster graphics [3] that organise data in tables of pixels and channels are well suited for these networks.[4] One key component of CNNs is the convolutional layer, which applies filters to input images, allowing the network to automatically extract relevant features at different spatial scales. It is called "Convolutional Layer" because it executes a convolution operation of the image with the filter, or rather in more mathematical terms the kernel.

$$\textbf{Convolution:} \quad (f * g)(t) \coloneqq \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{1}$$

"A convolution is an integral that expresses the amount of overlap of one function g as it is shifted over another function f. It therefore "blends" one function with another." [5]
This enables CNNs to detect edges, textures, shapes, and other visual patterns, crucial for image understanding. CNNs are widely used for object detection, where they can accurately identify and localize objects within images or video streams. Applications range from surveillance systems to autonomous vehicles, enabling them to detect pedestrians, vehicles, or other relevant objects, they are proven and tested solution. CNNs are also employed in image classification tasks, where they can categorize images into predefined classes with remarkable accuracy.

Notable examples of Convolutional Neural Networks are the following:

- EfficientNet: EfficientNet is a family of CNN architectures that have achieved state-of-the-art performance with increased efficiency. EfficientNet models have been utilized in autonomous vehicles for various tasks, including object detection, semantic segmentation, and instance segmentation.
- AlexNet: Introduced in 2012, AlexNet was one of the pioneering CNN architectures that achieved a breakthrough in image classification. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by significantly outperforming traditional approaches, showcasing the power of CNNs.
- ResNet: Residual Networks [6] (ResNet), introduced in 2015, addressed the vanishing gradient problem associated with training deep neural networks. By using skip connections that bypass some layers, ResNet enabled the training of even deeper CNNs with improved accuracy. It has been widely adopted for various image analysis tasks. Furthermore, ResNet-18, a variant of ResNet, has been used in autonomous vehicles to segment and give context to the vehicles environment.
- YOLO (You Only Look Once): YOLO [7] is a real-time object detection framework that uses CNNs to simultaneously predict bounding boxes and class probabilities for multiple objects in an image. YOLO-based models have been utilized in autonomous vehicles to detect and track objects such as pedestrians, vehicles, and traffic signs.

### 2.1.2 YOLO

The selected approach for this work is to utilise YOLO, an innovative approach to object detection. Whereas traditional methods re-purpose classifiers for detection, YOLO proposes a different framework. It treats object detection problems as regression problems, predicting spatially separated bounding boxes and class probabilities. By employing a single neural network, it is possible to directly predict these values from full images in one evaluation. This unified architecture allows for end-to-end optimization, optimizing detection performance seamlessly. The base YOLOv3 model achieves real-time image processing at 45 frames per second. Even the smaller version, Fast YOLOv3, achieves an astounding 155 frames per second while doubling the mean average precision (mAP) compared to other real-time detectors. While YOLO may have more localization errors than state-of-the-art systems, it demonstrates a lower tendency to predict false positives on the background. Moreover, YOLO exhibits impressive generalization, outperforming other detection methods like DPM and R-CNN when transitioning from natural images to diverse domains such as artwork.[7]

Further models developed from YOLOv3, such as YOLOv5 and YOLOv8, developed by Ultralytics, offer even more robust performances in terms of accuracy and speed (especially YOLOv8). In terms of training time, which may be critical in this work, YOLOv5 would be the best choice. [8]

A great boon to this work is the possibility to output uncertainty scores from the model. All YOLO versions offer a confidence score on predictions and this can be leveraged to select what and what not will be segmented and classified. A further uncertainty score that is possible to apply, is a Gaussian Uncertainty prediction that attribute 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 [9], 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). This method is openly vailable on YOLOv3.

### 2.1.3 Active Learning

Active Learning is a machine learning approach that improves model performance by iteratively selecting and annotating the most informative data instances. Different query strategies to sample and label novel data exist, such as: Uncertainty sampling, where instances are chosen for which the model has low confidence in its predictions; Diversity sampling, a strategy that aims to cover a wide range of feature space by selecting instances that are dissimilar from each other; and finally information density sampling, which is a strategy that prioritizes instances that are likely to provide the most information about the data distribution.

Active Learning is valuable in improving datasets and DL by reducing annotation effort and accelerating the training process. It enables more efficient use of annotator resources, especially when labeled data is limited or costly. By selectively labeling the most informative

instances, active learning enhances the accuracy and generalization of deep learning models, making them more effective for real-world applications. Since the time and capacity for this work are limited, Active Learning is an excellent way to improve the effectiveness of DNN with a limited or unbalanced dataset.

When dealing with query strategies, the two main ones are uncertainty sampling and diversity sampling. In this section, it will be reaffirmed and further explained the characteristics of both and how they can improve a project. Often a blended strategy may be used thus obtaining further benefits from both.

Uncertainty Sampling involves selecting instances for labeling based on the uncertainty of the model's predictions. The goal is to prioritize instances for which the model is least confident in its classification. This strategy allows the model to actively seek clarification on challenging or ambiguous samples, thereby improving its performance. In image classification, uncertainty sampling can be applied by selecting images for labeling that have low prediction probabilities or high entropy. For example, if a deep learning model is trained to classify, uncertainty sampling may identify images where the model is uncertain about the classifications. By annotating and incorporating these challenging instances into the training set, the model can learn from its mistakes and improve accuracy in distinguishing similar instances of classes. Uncertainty sampling is particularly useful in scenarios where labeled data is limited or expensive to obtain. By strategically selecting instances that are most likely to improve the model's performance, uncertainty sampling enables efficient utilization of annotation resources and accelerates the learning process. It helps the model focus on challenging cases that require human expertise, ultimately enhancing the overall accuracy and robustness of the deep learning model.

Diversity Sampling, another query strategy in Active Learning, aims to select instances that cover a wide range of feature space or represent different regions of the data distribution. The goal is to ensure that the labeled data includes diverse perspectives and variations, providing a comprehensive representation of the underlying data. In object detection, diversity sampling can be used to ensure that the annotated instances cover a wide range of object categories, shapes, sizes, and orientations. By selecting diverse samples, the model can learn to generalize better and detect objects in various scenarios accurately. This approach is particularly valuable in complex and dynamic environments, such as autonomous driving, where the model needs to handle diverse objects and scenarios encountered on the road. Diversity sampling is also beneficial in text classification tasks. For instance, in document categorization, it can ensure that the labeled data includes documents from different topics or genres, capturing a wide range of linguistic patterns and semantic variations. This helps the model develop a more comprehensive understanding of the text data and improves its ability to generalize to unseen documents. By incorporating diverse samples during the active learning process, the model can learn from a representative set of instances, leading to better generalization and improved performance on a wider range of inputs. Diversity sampling helps overcome biases and limitations that may arise from imbalanced or homogeneous training data. It promotes a more comprehensive understanding of the data distribution and enhances the deep learning model's ability to handle diverse and complex real-world scenarios.

## 2.2 Dataset

Some datasets were taken into consideration for the current work, but the following factors decided the final choice: number of classes (based on direction indicated by traffic light, colour, on or off [8 total]), type of sensor used, type of traffic lights depicted. Therefore the Small Bosch Traffic Light Dataset was chosen, as it was filmed in Berlin using a RCCB sensor very similar to the ones that will be used and it has the appropriate number of classes for the task.

**Data description:** "This dataset contains 13427 camera images at a resolution of 1280x720 pixels and contains about 24000 annotated traffic lights. The annotations include bounding boxes of traffic lights as well as the current state (active light) of each traffic light. The camera images are provided as raw 12bit HDR images taken with a red-clear-clear-blue filter and as reconstructed 8-bit RGB color images. The RGB images are provided for debugging and can also be used for training. However, the RGB conversion process has some drawbacks. Some of the converted images may contain artifacts and the color distribution may seem unusual." [10]

# 3 System Requirements, Architecture and Specifications

## 3.1 Requirements

- The algorithm should be able to run in real-time on the vehicle's hardware, ensuring timely processing and response.
- The algorithm must accurately detect and locate traffic lights in a real-world traffic setting, including intersections and zebra crossings.
- The algorithm should be able to handle various weather and lighting conditions, such as rain and nighttime.
- The algorithm must correctly identify the color of the traffic lights, distinguishing between red, green, and yellow signals.
- The algorithm should be robust to occlusions caused by other objects, such as overhanging branches or large vehicles.
- The algorithm should adapt to unconventional traffic light placements or orientations, correctly detecting and identifying them.
- The algorithm should minimize false detections or misclassifications, ensuring reliable and accurate results.
- The algorithm must be optimized to improve detection accuracy and reduce processing time.

These system requirements form the technical storyline for the development and implementation of the traffic light detection and identification algorithm. Each requirement is crucial in achieving the specific objectives of accurately detecting and identifying traffic lights in real-time, under various environmental conditions and challenging scenarios.

## 3.2 System Architecture

The task at hand is, mainly to train and develop a system that can later be applied to an already working autonomous vehicle ROS as a 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 Small Bosch 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.

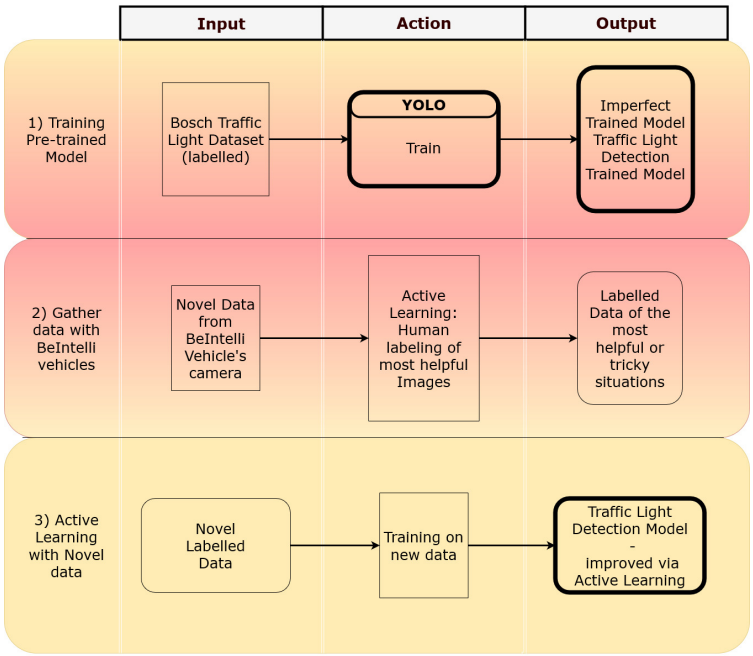


Figure 1: System architecture divided in three phases: 1. Initial training 2. Active Learning (labelling) 3. Active Learning (training on newly labeled dataset)

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.

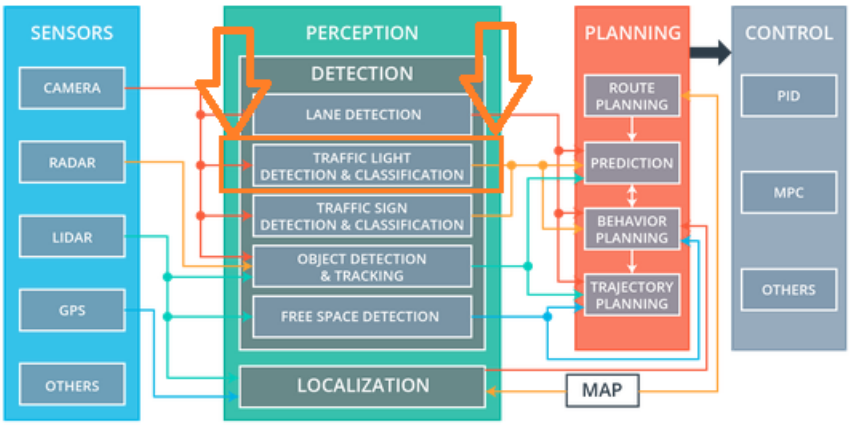


Figure 2: Generic ROS system architecture of autonomous vehicle. The highlighted node in the Perception box is the system that is of interest in the present work. (Image taken from: [https://taylor.raack.info/2018/08/autonomous-vehicle-technology-system-integration-with-ros/] and then edited)

## 3.3 System Specifications

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

### Training Pre-trained Model

- **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 [10], because it allows to recognise,the colours, on or off traffic light and different classes for direction signalled by traffic light.
- **Action**: Train YOLO [11] 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.

# 4 Timeline

- **W1 - W6:** Literature review on object detection, active learning, getting to grips with environment
- **W6 - W8:** Finalize choice of datasets, object detection algorithm, active learning methods
- **M1:** Presentation of SDS and SDS submission
- **Until W8:** Setting-up object detection and training pipeline
- **Until W10:** Applying pretrained model to DAI-Lab data and evaluation of the performance of the models
- **M2:** Midterm status update
- **Until W16:** Applying active learning method to data collected by cameras of BeIntelli cars and pretrained model - obtain trained model
- **M3:** Final presentation, show results and comparisons
- **Until W17:** Compare performance of trained models
- **Until W19:** Finalize documentation and source code for BeIntelli cars

Literature review, choice of dataset, active learning method have been carried out. Currently tests on best performing detection algorithm are active, as well as setting up pipeline for the model. As it stands, this timline is respected.

This timeline is based on the assumption that we receive access to the test dataset by Wednesday, the 31st of May.

# 5 Frameworks and Tools

- Ubuntu 20.04 or 22.04
- Python
- Pytorch
- YOLOv3/v5/v8
- ROS
- OpenCV

# References


[1] Frank Rosenblatt. *The Percetpron: A perceiving and recognizing automaton*. PhD thesis, Cornell Aeronautical Laborotary Inc., 1957.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. , 521(7553):436–444, May 2015.

[3] Ankit Jain. Vector vs Raster Graphics, 2022.

[4] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[5] Eric W. Weisstein. Convolution.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[8] Yolov8 vs. yolov5: Choosing the best object detection model.

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

[10] 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.

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