Resource-Aware Active Learning for Object Detection

Master-Thesis
Ha Giang Hoang Tran
KOM-M-0671

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Resource-Aware Active Learning for Object Detection
Ressourcenbewusstes Aktives Lernen zur Objekterkennung

Master-Thesis
Studiengang: Informatik
KOM-M-0671

Eingereicht von Ha Giang Hoang Tran
Tag der Einreichung: 18. März 2024

Gutachter: Dr.-Ing. Tobias Meuser
Betreuer: Ahmad Khalil

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Ha Giang Hoang Tran, die vorliegende Master-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Darmstadt, den 18. März 2024

_____

Ha Giang Hoang Tran

Contents

Abstract

Autonomous driving is currently one of the topics where intensive research has been done over the last few years. One of the crucial aspects of it is safety through environmental perception. Thus, object detection is an important technology to get to know the surroundings, requiring object detection models based on deep neural networks (DNN). However, these models require a large amount of annotated data. Exchanging the data through the vehicular network for model training would result in a network overload because these DNN-based models require huge amounts of data.
Active Learning is a training method that selects informative samples from the dataset and therefore reduces the amount of required training data. Recent researches dealing with active learning for object detection (ALOD) however provide the dataset for the active learning cycle in a centralized manner. Therefore, this thesis proposes a distributed ALOD by dividing the active learning cycle into two parts, similar to a server-client architecture. This design imitates a deployment of the ALOD framework into real-world scenarios. Hence, the clients are responsible for collecting and selecting the most useful samples. These selected samples are transmitted by considering the network utility. Consequently, this aims to reduce the amount of data that is transferred in the vehicular network compared to the approach that would send every collected sample. On the other hand, the server receives these selected samples, provides them to an annotation process, and trains the model. Furthermore, it provides for the clients, class balance values as feedback, enabling them to refine their selection. The approach was evaluated by using the YOLOv8 model and the nuImages dataset. The results showed that the proposed approach is capable of reaching nearly similar performance scores as the typical centralized ALOD approach.

1 Introduction

With the increase of technologies and thus the growth of data, Machine Learning (ML) and, particularly, Deep Learning (DL) have gained significant interest over the last few years. One of the application areas that recent research focuses on is DL-based approaches in the domain of autonomous driving. Within autonomous driving the safety aspect is one of the crucial topics, because autonomous vehicles need to perceive and understand their environment [RF18]. Thus the cars are equipped with different sensors including radars, LiDars, and cameras. One of the common technologies to perceive the surroundings is object detection, utilizing DNN-based models.
However, deploying these DNN-based object detection models into real-world scenarios poses challenges. These models require a lot of labeled data for supervised training, and their effectiveness is constrained by what is learned in the training dataset. Furthermore, the traffic scenes in the real world are very dynamic, requiring these models to train on a lot of data to achieve robust model performance [RF18]. Moreover, datasets containing many similar samples might lead to overfitting of the model. This could result in an overly specified detection model and reduce the generalization ability [RF18, HBK+23]. Thus, to address these problems, the concept of active learning is slowly gaining more interest over the last years [RXC+21]. An active learning (AL) algorithm aims to optimize the labeling process by selecting the most informative samples for the model to train on, based on its current knowledge [Set09]. Consequently, the costs, time, and computing resources are reduced compared to a fully supervised approach. There are several approaches to query the most informative samples such as uncertainty-based approaches, diversity-based approaches, and the combination of uncertainty-based and diversity-based approaches. The combined approach, which is also called hybrid approach, combines the strengths of both strategies while reducing their weaknesses. The uncertainty-based approaches tend to select samples that are very similar thus leading to bias and overfitting. Conversely, the diversity-based approach aims to select samples that represent the whole dataset. This could cause, non-informative samples to be selected [RXC+21, HBK+23].

1.1 Motivation and Contribution

Currently, there are many approaches considering active learning for object detection (ALOD) employing different query strategies. In the domain of autonomous driving, there are approaches dealing with 2D object detection, and 3D object detection, which combine images with sensor data. Additionally, there are also object detection approaches that offload the detection tasks to the edge. Reviewing the current active learning approaches, it becomes evident that they mainly consider centralized approaches, where the dataset utilized for the framework is gathered beforehand and stored centrally. However, there are no ALOD approaches that consider the framework in a distributed way, similar to a server-client architecture, where clients, such as vehicles, collect and select the data for model training on the server. This approach should mimic the deployment of ALOD in a real-world scenario. Thus, this thesis introduces a distributed ALOD framework that divides the AL process into two components: Clients have the task of collecting data samples and then selecting the most informative ones. They are transmitted to the server while considering the network utility. On the other hand, the server is responsible for gathering the samples from all clients, then labeling these samples and training the model. With this methodology, we want to answer the following research questions:

- How does the distributed ALOD framework perform compared to a typical ALOD framework which has a centralized and global view of the whole dataset?
- What impact do adjustments to the class balancing method have on the performance of the distributed ALOD framework?
- How does the network utility affect the performance of the models?

1.2 Outline

This thesis is structured as follows: Chapter 2 presents relevant background information to understand the thesis. The contents of the chapter are about autonomous driving, its vehicular network, active learning, and object detection along with its metrics. In Chapter 3 a brief overview of the related work regarding edge-based object detection and ALOD approaches is given. The design of the distributed ALOD approach is described in Chapter 4. The subsequent Chapter 5 presents implementation details of the framework. In Chapter 6 the approach is evaluated and its results are analyzed. Finally, Chapter 7 concludes this thesis with a summary of the work and possible future works.

2 Background

This chapter presents the basic relevant information in order to understand the topic of this thesis. In Section 2.1 autonomous driving is introduced. Subsequently, Section 2.2 presents some basic information about the vehicular network. Then, Section 2.3 provides some general information on object detection, continuing with the idea behind active learning in Section 2.4. Furthermore, Section 2.5 explains the metrics that are used to measure the performance of object detection approaches.

2.1 Autonomous Driving

Over the last few years, autonomous driving has gained very much attention from the industry and academia. Due to that, the industry made significant enhancements to the capabilities and reliability of sensors, cameras, and vehicle-to-everything (V2X) communication. Hence, they also improved the autonomous driving capabilities which are defined at different levels by the Society of Automotive Engineers (SAE) [oAE21]. The SAE defined six levels of driving automation while levels 0 to 2 are categorized as features supporting the driver and levels 3 to 5 are the automated driving features. The different driving automation levels are displayed in Figure 2.1 with the tasks of the driver and features the levels are providing.

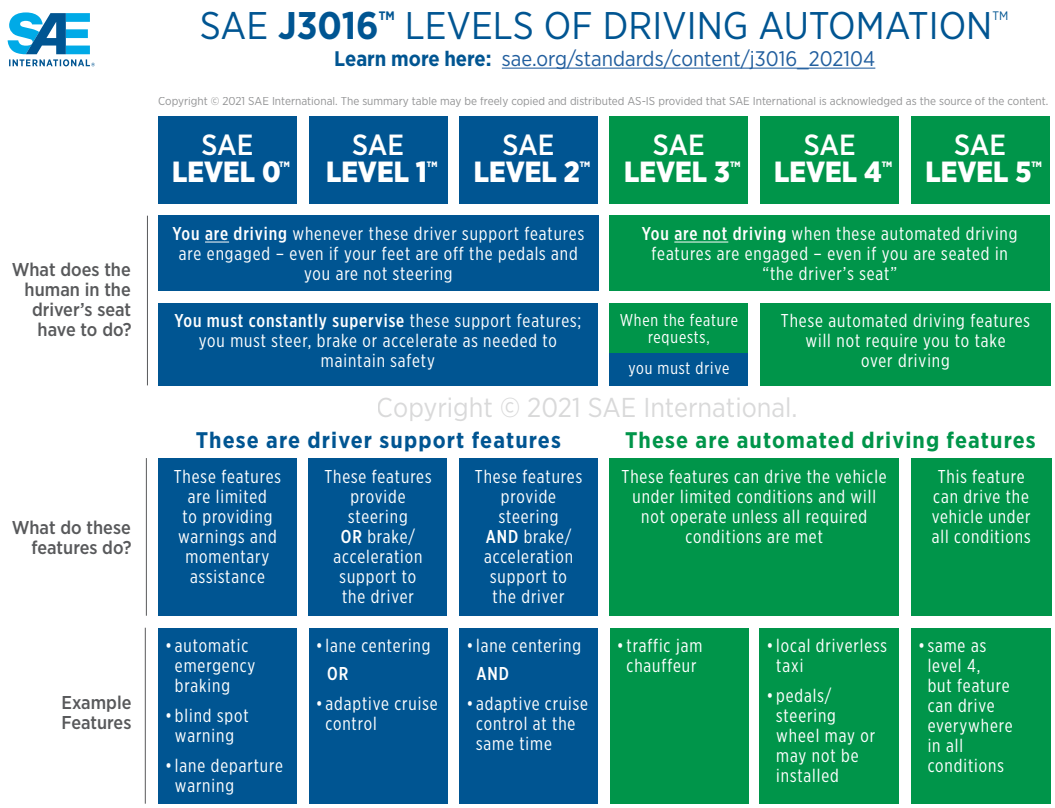

Figure 2.1: SAE levels of driving automation [oAE21].

The difference is that in the first group, the driver has to drive by himself, while in the second group, the car is driving by itself if the features are activated. Currently, there are some car manufacturers that

have the technology to fulfill the SAE-level 3 features. One German car manufacturer received the first official license for SAE-level 3 vehicles in Germany in 2021 [Bun21] and in the State of Nevada since January 2023 [MB23]. This means that they are allowed to deploy and sell vehicles that are equipped with the features of SAE-level 3.
This SAE-level 3 technology is known as Conditional Driving Automation, indicating that vehicles can autonomously operate under specific conditions, yet the driver must be ready to intervene if the features request it. Ensuring safety stands out as a primary objective in autonomous driving [RF18]. As a result, a range of sensors, cameras, and V2X communication play crucial roles in understanding and interacting with the environment, fulfilling the requirements for autonomous driving features.

2.2 Vehicular Network

This V2X communication includes vehicle-to-vehicle (V2V), vehicle-to-pedestrian (V2P), vehicle-to-infrastructure (V2I) and vehicle-to-network (V2N) communication as illustrated in Figure 2.2. Through this wireless interconnection collected information about the perceived environment and also information of the own vehicle can be exchanged [CHS+17]. Hence, V2X communication can handle three different messages: Cooperative Awareness Messages (CAM) [ETS14a], decentralized environmental notification messages (DENMs) [ETS14b], and Collective Perception Message (CPM) [ETS23]. All three messages are standardized by the European Telecommunications Standards Institute (ETSI). CAM is used to share information about the own status of the vehicle like current position, velocity, driving direction, acceleration, and vehicle type [ETS14a]. DENMs are event-based messages that share information about accidents or weather conditions that can affect the traffic flow and road safety [ETS14b]. CPM in contrast shares data gathered from the environment which includes surrounding objects, obstacles, and other road users that are detected through local perception sensors like radars, lidars, and cameras [ETS23]. Hence, to be able to detect objects in the environment object detection algorithms are required.

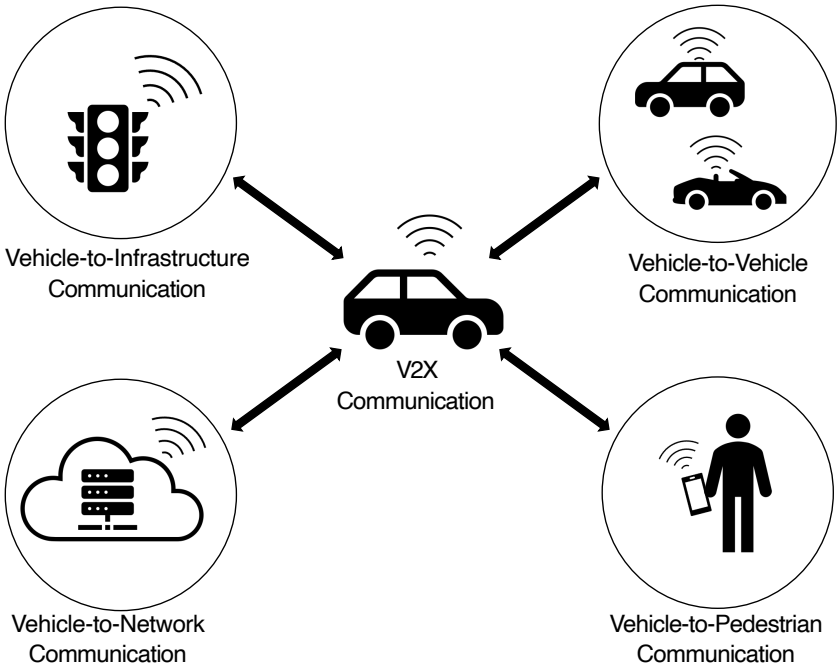

Figure 2.2: V2X communication and its four types of communication services.

2.3 Object Detection

Object detection is a computer vision technique that detects relevant objects in images or videos and is a crucial task in perceiving the local environment in the autonomous driving application. It combines classification and localization tasks with the use of deep learning. There are two main object detection approaches: Two-stage object detectors and one-stage object detectors. A two-stage object detector, like Faster-RCNN [RHGS15], uses a region proposal network that provides the region of interest (RoI) where an object might be present. The second stage predicts the bounding boxes using the RoIs. On the other hand, one-stage object detectors in contrast directly predict the bounding boxes and class probabilities [LOW+20]. A popular one-stage object detector is You Only Look Once (YOLO) published by Redmon et al. [RDGF16]. One-stage object detectors have the advantage of being faster because they process images in a single pass through the neural network. Nevertheless, the two-stage detectors have higher computational costs but have higher accuracy compared to one-stage detectors [LOW+20].
Given that YOLO [RDGF16] stands out as one of the most widely employed detection models, providing rapid predictions that align well with the requirements of autonomous driving applications, it has been selected for use in this thesis. Additionally, the latest YOLO model version, YOLOv8 published by ultralytics [JCQ23], represents a state-of-the-art model. YOLO applies a neural network to the whole image. This neural network divides the image into a $S \times S$ grid and predicts the bounding boxes and probabilities for each region with Convolutional Neural Networks (CNN) in real-time [RDGF16], as shown in Figure 2.3.

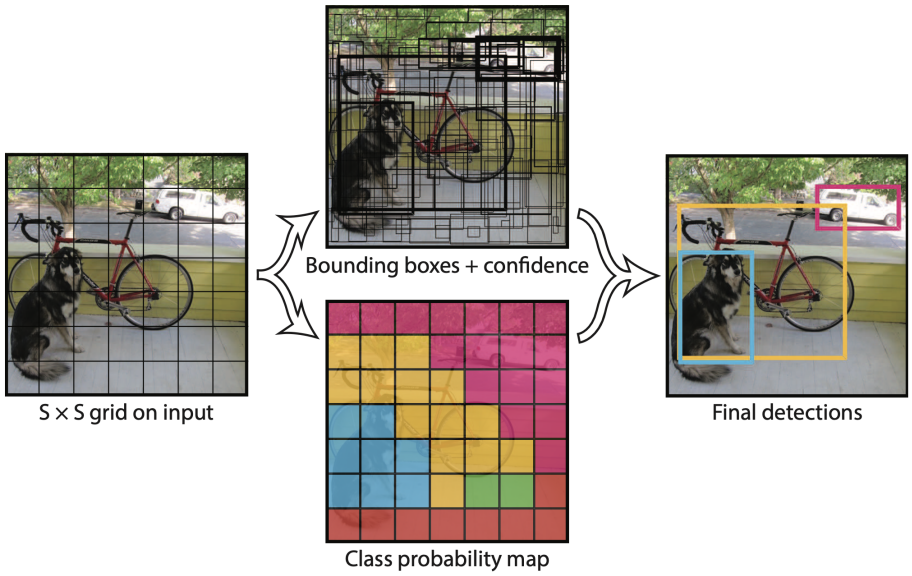

Figure 2.3: The YOLO model is considered as a regression problem. First the image is divided in a $S \times S$ grid. Then for each grid, B bounding boxes are predicted with the corresponding confidence scores and the C class probabilities. Finally, these predictions are represented as tensors with dimensions $S \times S \times (B * 5 + C)$ [RDGF16].

There are different ways to train those object detection models. First of all, the model can be trained using the supervised learning approach, where a fully labeled dataset is required. Secondly, a semi-supervised learning which uses a combination of labeled and unlabeled data. Then there is the approach of weak supervision which trains the model using partially labeled data and noisy data. Furthermore, transfer learning is an approach that uses a model pre-trained on different datasets and fine-tunes them on our dataset and task [RVHR19]. Despite the increasing amount of approaches utilizing semi-supervised or weakly supervised approaches, in practice, the training of such safety-critical features relies on fully supervised approaches. However, traditional supervised approaches suffer from

the drawback that the labeling tasks are highly costly and time-consuming [KBM23, RVHR19, GAA+24]. Hence, Active Learning (AL) is a concept to reduce the aforementioned problem.

2.4 Active Learning

Active Learning (AL) is slowly gaining more attention after Deep Learning got huge interest from researchers due to the amount of information that is available nowadays. Hence, there are many datasets with large amounts of annotated data but the annotation process requires many resources such as experts to annotate, time, and costs. This is why AL plays a significant role because the idea of it is to select samples with the highest value to provide them in a training dataset. This selected dataset is then given to an annotator to label and then used to refine the model [RXC+21].

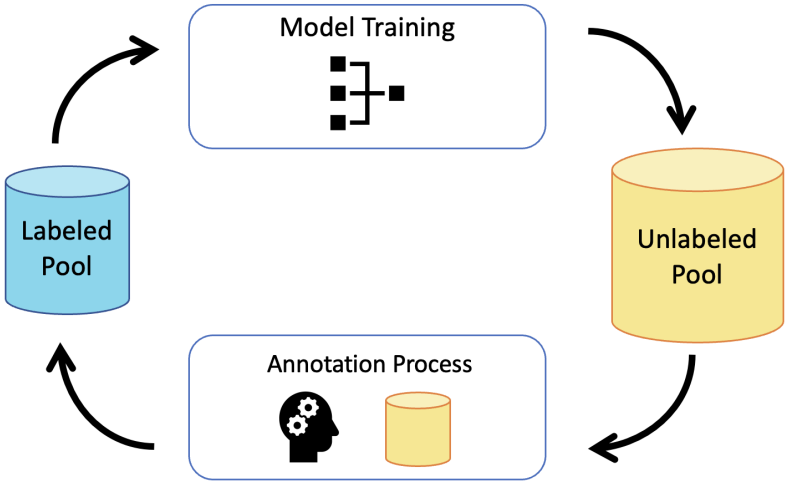

Figure 2.4: Active Learning Cycle.

Figure 2.4 illustrates the AL cycle. Here, the cycle can start with a small labeled subset and an unlabeled data pool. The small labeled dataset is used for training the machine learning model. The new model is used to predict the labels in the unlabeled pool and uncertainty estimation approaches are applied for each sample. Based on the uncertainty scores the most informative samples are selected and sent to an annotator for labeling. After labeling these samples they are added to the labeled dataset and the new iteration begins [Set09, RXC+21]. AL can be used in different ML and DL applications like classification, recognition, and object detection tasks.

2.5 Object Detection Metrics

This Section provides an overview of commonly used evaluation metrics for the evaluation of object detection models. The first Subsection 2.5.1 explains the metric Intersection over Union. Then, the next Subsection 2.5.2 defines the metrics Precision, Recall, F1-Score, and the concept of the confusion matrix. Subsection 2.5.3 introduces the metric mean Average Precision.

2.5.1 Intersection over Union

Intersection over Union (IoU) is a metric to measure the quality of the predicted bounding boxes. Therefore, it plays a crucial role in determining the quality and accuracy of a detection model. IoU is the overlap of the predicted bounding box and the ground truth bounding box. As a mathematical term it is calculated by taking the intersection area divided by the union area of the two bounding boxes, see Figure 2.5.

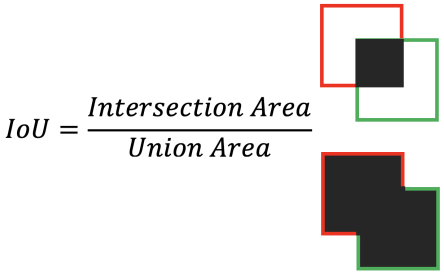

Figure 2.5: Calculation of the Intersection over Union.

The result of this formula indicates how well the predicted bounding box aligns with the ground truth bounding box. This means a higher IoU indicates a greater localization accuracy of the model [Ved23].

2.5.2 Precision, Recall, F1-Score, and Confusion Matrix

Precision, Recall, and F1-Score are important metrics to measure the performance of the object detection model. To understand those metrics the basic concepts have to be explained: True Positive (TP) is a metric that represents the predictions correctly made by the object detection model, where the IoU value is equal to or greater than a specified IoU threshold. False Positives (FP) are predictions incorrectly made by the model, which means that the detected object doesn't exist in the ground truth or the IoU value is below the threshold. In the case of False Negative (FN) the model does not detect the object which exists in the ground truth [Ved23].

In the model evaluation, Precision is a crucial metric to measure how many predictions of the model are correct. More precisely it distinguishes correctly detected objects from the false positives. Hence, Precision gives insights into the model how accurately it makes positive predictions. A high Precision score indicates how well the model can provide reliable positive predictions [Ved23].

$$\mathrm{P} = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

Recall measures the ratio of correct detection to all ground truth objects. A high Recall score denotes how well the model can detect relevant objects [Ved23].

$$\mathrm{R} = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truth}}$$

The F1-Score is the harmonic mean of Precision and Recall, balancing both metrics. Hence, a high F1-Score indicates good performance on Precision and Recall [Ved23].

$$F_1 = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}}$$

The confusion matrix gives a class-wise distribution of the predictive performance. Thus it gives more insights into which classes the model makes wrong predictions. It is a $N \times N$ matrix, where N represents the classes, and displays TP, TN, FP, and FN [Sur20]. The following Figure 2.6 illustrates the confusion matrix:

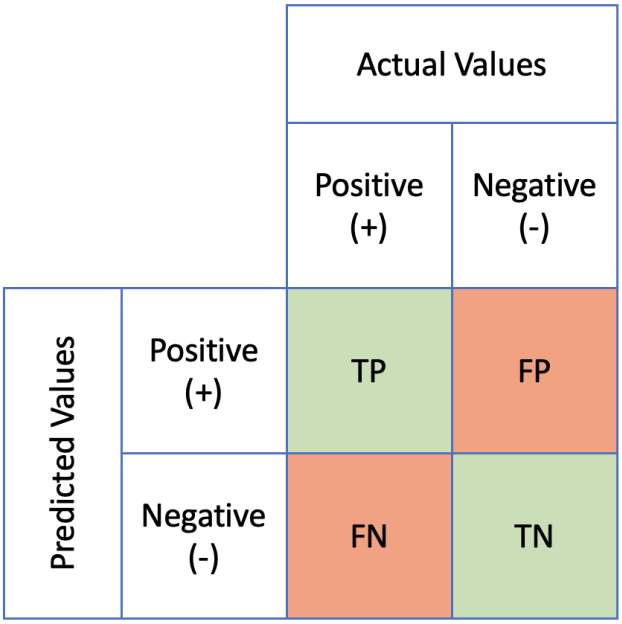

Figure 2.6: Confusion Matrix

2.5.3 Mean Average Precision

Average Precision (AP) and mean Average Precision (mAP) are commonly used metrics to evaluate object detection models and to identify the performance of detecting and localizing objects within images or video frames. AP is calculated by computing the precision-recall curve for a certain class. This precision-recall curve is generated by calculating precision and recall values at different confidence thresholds, where the precision values are plotted against the recall values. An Example of a precision-recall curve with different classes is displayed in Figure 2.7. AP is represented as the area under this curve. This means, that the closer the curve of the class or of the overall model is to the coordinate $(1, 1)$ the better the class or the model is performing. There are two methods to calculate the AP: the 11-point interpolation and all-point interpolation. In the 11-point interpolation, the precision values for eleven recall values in the range of [0.0, 1.0] with an increment of 0.1 is calculated. AP then can be calculated by taking the mean of these precision values [PNDS20]:

$$\text{AP} = \frac{1}{11} \sum_{R \in \{0, 0.1, 0.2, \ldots, 1\}} P(R).$$

In contrast to the 11-point interpolation, the all-point interpolation leverages all points using the following formula:

$$\text{AP}_{\text{all}} = \sum_{n} (R_{n+1} - R_n) P_{\text{interp}}(R_{n+1}),$$

where

$$P_{\text{interp}}(R_{n+1}) = \max_{\tilde{R}: \tilde{R} \geq R_{n+1}} P(\tilde{R}).$$

$P(\tilde{R})$ is the precision at the interpolation point R. The condition $\tilde{R} : \tilde{R} \geq R_{n+1}$ signifies that $P_{\text{interp}}(R_{n+1})$ represents the maximum precision among the interpolation point R and subsequent interpolation points [PNDS20]. Consequently, a higher AP value indicates higher detection performance for a specific class.

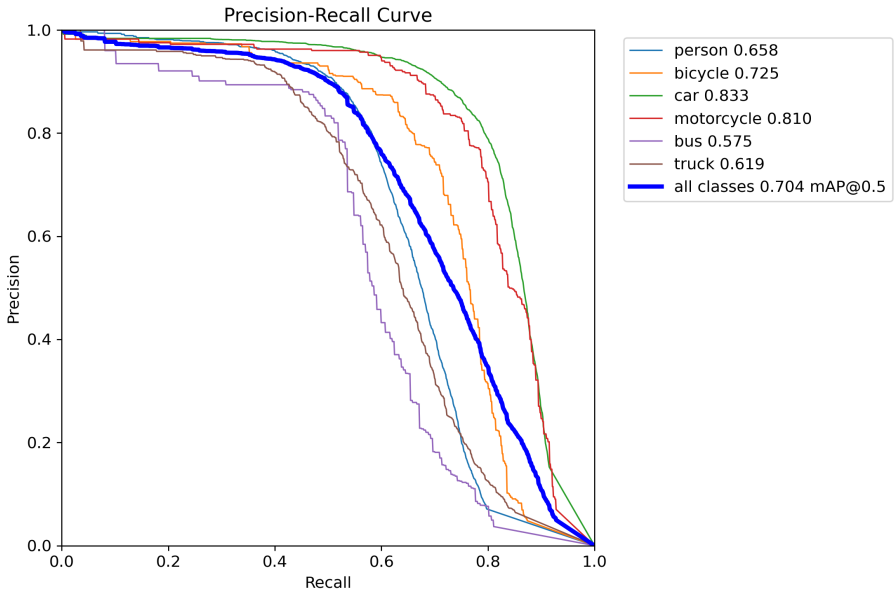

Figure 2.7: Example of a Precision-Recall Curve.

The mAP extends the concept of AP by evaluating the performance of the model over multiple classes. Therefore, the AP values of each class are calculated first and then the average of all AP scores across all classes is calculated using the formula:

$$mAP = \frac{1}{C}\sum_{i=1}^{C} AP_i,$$

where C is the number of classes. Because mAP considers precision and recall for different classes and IoU thresholds it is widely used to compare models. A higher mAP value indicates superior model performance [Ved23].

2.6 Summary

This chapter has introduced the most important components to understand this thesis. It started with the introduction of autonomous driving and its required technologies. Then the role of object detection and its importance in that domain is explained and how such object detection models are trained. After that, the functionality of active learning and its advantage of selecting informative samples to reduce the labeling task is described. And in the last section, the important object detection metrics are introduced. The next chapter presents the works related to the topic being covered by this thesis.

3 Related Work

This chapter gives an overview of works related to this thesis. First, Section 3.1 presents edge assisted object detection approaches. Then, active learning for object detection approaches are introduced in Section 3.2. Section 3.3 provides approaches that focus on active learning for object detection in the autonomous driving domain. Finally, Section 3.4 describes an analysis of the related work and a comparison with our approach introduced in this work.

3.1 Edge Assisted Object Detection

For autonomous driving, object detection is a crucial task to perceive the environment. Some researches thereby offload the computation tasks from the vehicle to the edge or cloud infrastructures. This is due to the rising computational capabilities these infrastructures have compared to the end-devices [KKKL21, LWZ+22, HRF24]. The approaches [KKKL21, LWZ+22, HRF24] share a similar idea, where the end-device, e.g., a vehicle, collects the data and sends it to the edge or cloud, which then performs object detection and sends back the results.
Liang et al. [LWZ+22] presents an edge-computing method called Edge YOLO. In the first step, their mobile platform collects real-time images and sends them to the edge server. The edge server then performs object detection using the modified YOLOv4 model and sends the results back to the vehicle. Furthermore, collected data are used to refine the model with the help of online data annotations and edge-cloud cooperation.
Kim et al. [KKKL21] proposes a similar approach with the difference that the vehicle checks the channel quality before sending the images. Moreover, their approach selectively transmits the original image only when the channel quality meets a certain level of quality. In case of poor channel quality, the vehicle compresses the image based on the RoI and sends this image version to the edge. However, this compression reduces the transmission latency according to the authors.
In the approach of Hawlader et al. [HRF24], they used image compression to minimize the data transmission latency between the vehicle and the edge or cloud. The authors employ JPEG and H.256 compression techniques for image compression. Similarly to the previous approaches, the object detection model, which in this case is YOLOv5, is trained on the edge or cloud.
As presented here, the approaches of edge assisted object detection rely on an object detection model trained in a supervised manner. Such supervised learning models have the disadvantage that the labels of the training dataset have to be annotated all in beforehand which is a time and cost intensive task for very huge datasets [RXC+21].

3.2 Active Learning for Object Detection

Active Learning for Object Detection (ALOD) gained interest over the last few years. As mentioned in Section 2.4 in active learning, the selection of informative samples is a crucial task to train the model. There are different methods to select those samples. Most of the ALOD approaches rely on the estimation of the uncertainty score of a sample and choose the most informative ones. Many approaches focus on classification and localization uncertainty only.
Yuan et al. [YWF+21] presented a multiple instance active object detection (MI-AOD) approach. MI-AOD utilizes discrepancy learning and multiple instances learning to learn and re-weight instance uncertainty. They furthermore introduced a model, which uses two adversarial instance classifiers to estimate the uncertainty of unlabeled instances by maximizing prediction discrepancy and minimizing classifier discrepancy. Another module within this approach establishes the relationship between instance and image

uncertainty. Therefore, each unlabeled image is an instance bag and evaluates instance appearance consistency across images to re-weight instance uncertainty. By iteratively bridging instance-level observation and image-level evaluation, this approach selects the most informative images.
Another approach is from Aghdam et al. [AGGWL19] who present an image-level scoring to select the best ranked images. They first compute the detection probability of each pixel in the image and aggregate them to image-level scores.
Kao et al. [KLSL19] introduced two uncertainty measurements called localization tightness and stability. Localization tightness calculates the difference of the Region of Interest (RoI), which is estimated first, with the final bounding box of the detector. Here, the samples with the lowest values are selected to refine the model. Localization stability measures the IoU of a sample which is affected by several Gaussian noise levels. The model is then called stable if the detection box of all noise levels does not significantly differ from the reference box, which has no noise. Hence, only the images with distinct localization predictions resulting from the addition of various types of noise are selected.
Another uncertainty-based method proposed by Choi et al. [CEL+21] uses a mixture density network. This network considers localization and classification-based aleatoric and epistemic uncertainties and aggregates them to an image informativeness score for the active learning selection.

Uncertainty-based methods often tend to select similar samples. Therefore, diversity-based methods, in contrast, try to select samples that represent the whole dataset.
Sener et al. [SS17] presented the core-set approach in which they attempt to solve the k-center problem. Therefore, they computed the Euclidean Distances between the extracted features and chose a small diverse set to train the model which should sufficiently perform well on the whole dataset.
Agarwal et al. [AAAA20] introduced a method for contextual diversity. It aims to select diverse samples in spatial and semantic contexts using the KL-divergence between the prediction probabilities of a sample and the previously selected subset.

Furthermore, there are approaches which combine uncertainty and diversity-based methods, aiming to select the most uncertain and diverse samples and hence, avoiding overfitting.
The authors of [YHC22] proposed a two stage framework called Plug and Play Active Learning (PPAL). In the first stage, it calculates and re-weights the sample uncertainty with the help of the introduced category-wise difficulty coefficient. The next stage measures the similarity of the objects in different images to select the most diverse samples from the first stage pool.
An entropy-based method to estimate the uncertainty is presented by Wu et al. [WCH22]. Besides that, they divide instance-level diversity into intra-image and inter-image diversity. The intra-image diversity uses the Entropy-base Non-Maximum-Suppresion to eliminate redundancy within an image. The inter-image diversity improves the intra-class diversity by a rejection redundancy process and the inter-class diversity by an adaptive bucket size for each class.

Despite the approaches focusing on uncertainty and or diversity estimation, there are also inconsistency-based methods. They apply multiple different augmentations on the same sample to identify the inconsistency of the model's predictions.
Elezi et al. [EYA+22] proposed a method based on sample augmentation. Here the original images and their augmentation are used for the acquisition score which represents the inconsistency and uncertainty of an image where images with the highest scores are selected. Furthermore, they introduced pseudo-labeling to prevent a distribution drift and to improve the results.
A two stage framework using data augmentation to calculate the consistency based on the predictions of the augmented and original data is presented by Yu et al. [YZYC22]. Using this consistency metric the most informative samples are selected. In the second step, this pool of informative samples is filtered by comparing them with the class distribution of the already labeled pool.

3.3 Active Learning for Object Detection in Autonomous Driving Scenario

In recent years, there has been a notable increase of interest in active learning within the literature concerning vehicular perception, particularly focusing on object detection.
Haussmann et al. [HFC+20] presented an ALOD system using an ensemble of object detectors that provide bounding boxes and probabilities for each class. After that, a scoring function is applied to calculate the informativeness score of each sample. Based on this informativeness score the samples are selected. Furthermore, they implemented a diversity-based method using extracted embeddings.
Schmidt et al. [SRTK20] proposed a method to estimate uncertainty using ensembles in combination with a continuous training strategy and a data balancing method which they tested on the Kitti dataset [GLSU13].
Another approach is introduced by Hekimoglu et al. [HBK+23] which considers non-redundancy. Their algorithm combines uncertainty and diversity in sample selection. It takes into account sample similarities and aggregates individual scores to generate a batch-wise acquisition score. By leveraging extracted object features instead of whole image features, the algorithm selects samples that are distant from other informative ones. This results in a diverse set of object types, shapes, and angels.
A similar approach addressing redundancy and class imbalance reduction is proposed by [KBM23] which estimates an optimal batch size while selecting the samples.
Another approach from Hekimoglu et al. [HFZ+23] presents a multi-task active learning approach considering the inconsistency in the object detection and semantic segmentation domain to select informative samples. For this multi-task approach, the predictions of the object detector and the semantic segmentation have to align. If they do not align, it indicates that one of the tasks fails and makes this sample an informative one to be selected.

3.4 Analysis of Related Work

Edge assisted object detection and active learning for object detection are handling the object detection task and especially the model training in a different way. While the edge assisted approaches gave an idea of how the object detection task could work in the real world, the AL approaches show how these models can be trained with lower effort for the labeling task. Table 3.1 gives an overview of the approaches with a short description of the key contribution compared to other approaches. Furthermore, it lists the detectors and datasets used in their approaches.

The edge assisted object detection method provides a connection between the edge and the end-devices in which they send images to the edge server, which performs the object detection task on the edge side. The problem with these approaches lies in the fact that they rely on traditional supervised training methods, which require a huge amount of annotated data. Moreover, while Liang et al. [LWZ+22] offer a refinement process for the detection model, they do not actively select informative samples during this stage. Instead, they label all collected data gathered by the end-device, leading to a huge volume of exchanged data in the vehicular network. Hence, the drawback of these approaches is that a huge amount of labeled data is required to train the model.

Active learning for object detection, on the contrary, aims to reduce that labeling task by selectively choosing only the most important samples, thereby enhancing the model. As presented in Section 3.2 certain approaches such as those proposed by [YWF+21, CEL+21], have mainly focused on sample selection based on the uncertainty estimation. However, in datasets containing images from continuous video streams, these approaches tend to select many similar samples, possibly leading to overfitting of the detection model. Conversely, other approaches only focus on diversity estimation, aiming to represent the entire dataset with a diverse selection of samples. However, this approach may overlook relevant samples the model is very uncertain of. Recognizing these limitations approaches such as those proposed

by [WCH22, HBK+23] combine uncertainty-based methods and diversity-based methods to overcome the disadvantages previously mentioned. This combination of methodologies allows for more robust and effective sample selections within the object detection task. However, none of the active learning for object detection approaches considered a distributed approach similar to the edge assisted object detection approaches. It means that the datasets that are used for the ALOD approaches are provided in a centralized manner. The drawback of this approach is that it assumes that the whole dataset is getting sent from the end-devices without considering to reduce the volume.

Therefore, this thesis aims to bridge this research gap by presenting a distributed ALOD framework. In this framework, clients are responsible for selecting informative samples from their locally collected datasets using a combination of uncertainty-based and diversity-based methods, while also incorporating feedback from the server for balancing the dataset. Subsequently, the selected data is transmitted to the server based on network utility considerations. The server handles the responsibilities of labeling the data, training, and validating the detection model.

Table 3.1: Overview of the related work.

Literature	Approach	Dataset	Detector	Description
[KKKL21]	Edge-network assisted real-time object detection network	KITTI	RFBNet	Vehicle sends compressed image based on RoI if channel quality is poor.
[LWZ+22]	Edge YOLO	COCO2017, KITTI	Edge YOLO	Model is refined with all images collected by the vehicle.
[HRF24]	Edge and cloud V2X-based real-time object detection	generated with CARLA	YOLOv5	Utilizing JPEG and H.265 compression to send images from vehicle to the edge and cloud.
[YWF+21]	Multiple Instance Active Learning for Object Detection	Pascal VOC 2007 & 2012, MS COCO	RetinaNet, SSD	Utilization of discrepancy learning and multiple instance learning to learn and re-weight instance uncertainty.
[AGGWL19]	AL for Deep Object Detection	CityPersons, Caltech Pedestrian, BDD100K	own detection network	Computes pixel-scores and aggregates them to image-level scores.
[KLSL19]	Location aware ALOD	Pascal VOC 2007 & 2012, MS COCO	Faster R-CNN, SSD	Introduced two localization uncertainty measurements called localization tightness and stability.
[CEL+21]	ALOD via Probabilistic Modeling	Pascal VOC 2007 & 2012, MS COCO	SSD	Estimation of aleatoric and epistemic uncertainty.
[SS17]	Core-set selection for AL	CIFAR, SVHN	VGG-16	Calculating Euclidean Distances between extracted features to obtain diverse samples.

Literature	Approach	Dataset	Detector	Description
[AAAA20]	Contextual Diversity for AL	Cityscapes, BDD100K, Pascal VOC 2007, CIFAR-10 & CIFAR-100	SSD	Diverse samples with spatial and semantic context are selected.
[YHC22]	Plug and Play ALOD	Pascal VOC 2007 & 2012, MS COCO	RetinaNet, Faster-RCNN	A two stage framework that calculates and re-weights sample uncertainty and selecting diverse samples based on similarity.
[WCH22]	Entropy-based ALOD	Pascal VOC 2007 & 2012, MS COCO	Faster R-CNN, RetinaNet	Entropy-based uncertainty estimation with a diversity method to reduce redundancy.
[EYA+22]	Rationalizing The Labeling Costs for Training Object Detection	Pascal VOC 2007 & 2012, MS COCO	SSD	Sample augmentation is used identify inconsistency and uncertainty.
[YZYC22]	Consistency-based Active Learning method for Object Detection	Pascal VOC 2007 & 2012, MS COCO	Faster R-CNN, RetinaNet	Utilization of augmentations to identify inconsistency and comparing selected images with distribution of labeled pool.
[HFC+20]	Scalable ALOD	Internal dataset	One-stage detector based on UNet-backbone	Employs an ensemble of detectors to calculate the informativeness score based on the probabilities and selects diverse samples using extracted embeddings.
[SRTK20]	ALOD with ensemble	KITTI	Faster R-CNN	Utilization of ensemble strategy and data balancing method.
[HBK+23]	Non-Redundant and Informative Sampling	Pascal VOC, KITTI, CIFAR-10 & CIFAR-100	CenterNet	Selection of sample by comparing the images based on their similarities and how distant the object features are.
[KBM23]	Cost-effective ALOD	Caltech Pedestrian	Tiny-YOLOv3	Addressing redundancy and class imbalance and estimation of optimal batch size.
[HFZ+23]	Multi task consistency for AL	nuImages, A9-Dataset	own ResNet structure	Identify inconsistency in the object detection and semantic segmentation domain.

4 Design

This chapter outlines the thesis approach, which involves dividing an active learning for object detection framework into two main components: a data collection and selection phase, and a data reception and model training phase. Section 4.1 provides an overview of this framework concept. Subsequently, Section 4.2 and Section 4.3 describe the functionalities of each component of the framework.

4.1 Overview

The objective of this thesis is to design an ALOD framework that adapts to the network environment while efficiently selecting informative samples in a distributed manner. Figure 4.1 provides a visual representation of the ALOD framework. This framework is specifically tailored for application in the field of autonomous driving. The approach consists of a central entity depicted on the left side of the graphic and a node entity on the right side. The main task of the central entity is to gather all data from one or more nodes and train the detection model. On the other hand, the nodes have the task of collecting samples and selecting the most informative ones to transmit to the central entity. Due to the similarity to a traditional server-client architecture, the central entity is referred to as the server and the nodes as clients or vehicles throughout the subsequent sections. The following section elaborates on the functionalities and tasks of the client within this framework.

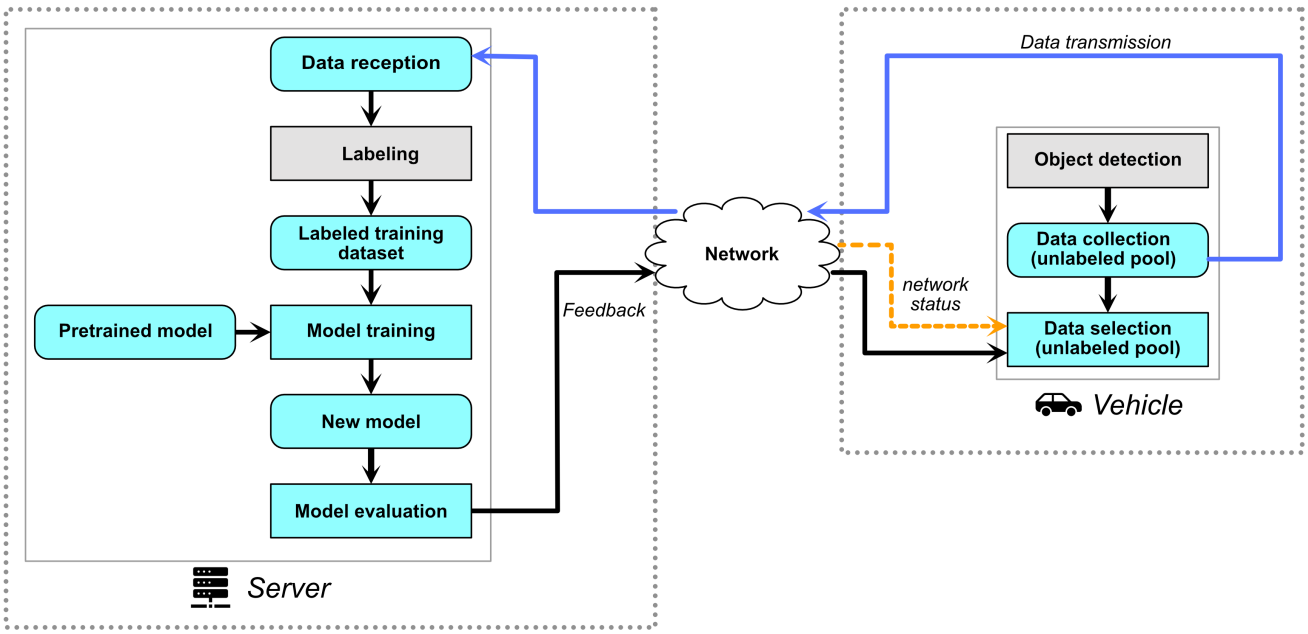

Figure 4.1: Overview of proposed ALOD design of this thesis.

4.2 Client Functionalities

In the context of autonomous driving, clients represent vehicles equipped with cameras and V2X communication capabilities. As stated in Section 2.1, one of the crucial features is to perceive the surroundings of the vehicle and detect relevant objects. The process steps undertaken by each client, are illustrated in Figure 4.2. These steps involve detecting objects directly on the camera footage using object detection

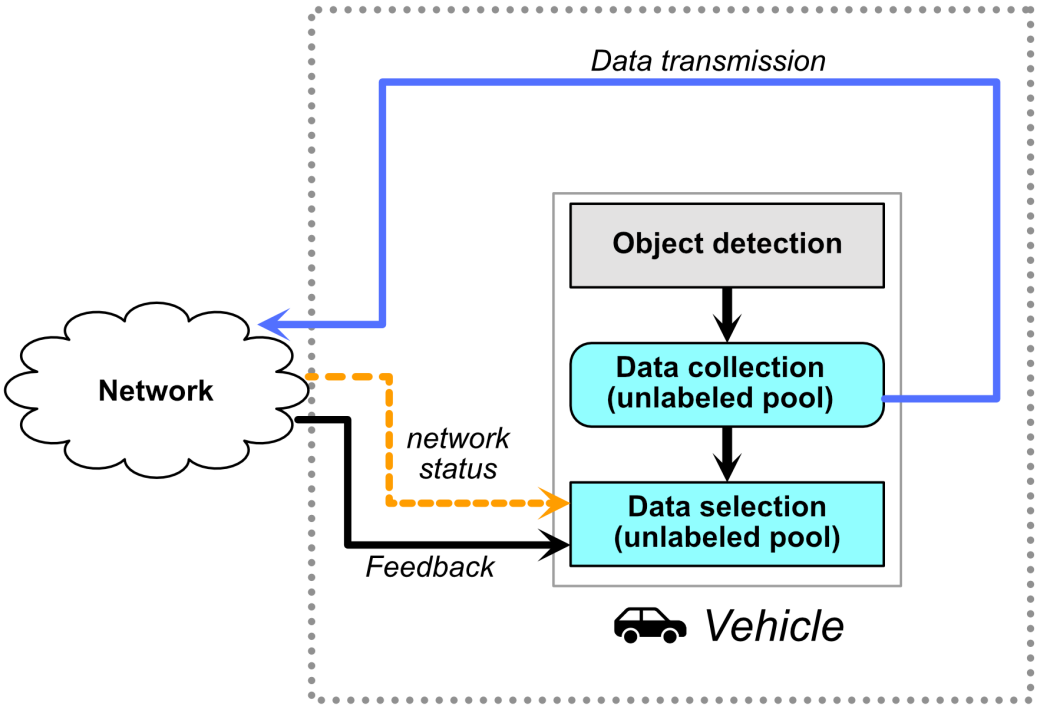

Figure 4.2: Client process steps: Utilizing object detection on images or videos captured by vehicle cameras. Collected images or video frames undergo data selection based on server feedback. Selected data is then sent to the server, considering network utility.

models, capturing them as images or video frames. These images are locally collected by each client and form what is referred to as the unlabeled data pool. In contrast, a centralized approach already has the data pool gathered beforehand. The next and crucial step in this active learning cycle involves selecting the most informative samples. Unlike a centralized approach, which has access to one unlabeled data pool and a global view of it, clients are only aware of their individually collected samples. Before selecting samples, an uncertainty estimation is performed on each unlabeled data sample. Entropy is utilized for the uncertainty estimation because it is a commonly used method in this research field, and due to its straightforward calculation, it is simple and computationally efficient. Entropy $H(I_{i,d})$ is calculated for each unlabeled image $\{I_i\}_{i\in[n]}$, where $[n] = \{1,\dots,n\}$ and $d \in D$ denotes the $d-th$ detected instance:

$$H(I_{i,d}) = -p_{i,d}\log(p_{i,d}) - (1-p_{i,d})\log(1-p_{i,d}),$$

where $p_{i,d}$ represents the confidence score predicted by the model for a certain class [WCH22, Set09]. After that, the instance scores are aggregated to obtain an image score.
For the aggregation Brust et al. [BKD18] proposed three aggregation strategies: sum, average, and maximum. The sum aggregation method favors images with multiple uncertainty detections. Images with no detections are valued with zero. The image score $s_{sum}(i)$ can be calculated with the entropy estimation as follows:

$$s_{sum}(i) = \sum_d H(I_{i,d}),$$

where s_i is the image score of image i. This sum method is sensitive to the number of detections in an image. To ensure comparability of the scores, the average aggregation method can be used to average over the number of detections:

$$s_{avg}(i) = \frac{1}{|D|}\sum_d H(I_{i,d}).$$

The third method is maximum aggregation, which considers only the highest detection score for the image score. While this method may result in information loss, it also increases the robustness when there are many noisy detections. The maximum is denoted as follows:

$$s_{max}(i) = \max_{d} H(I_{i,d}).$$

Several approaches like [HFC+20, CEL+21, HBK+23], made comparisons of the aggregation functions, where in most cases the maximum aggregation is slightly better. As a result, we are utilizing the maximum aggregation method in our framework. However, it is important to acknowledge that this approach may lead to information loss. To address this concern, we have incorporated a method that prioritizes images containing multiple objects. This approach involves counting the total number of objects detected by the model, regardless of their class. However, objects of the same class, except for the first instance, are counted with a reduced weight of 0.25 instead of 1.0. This strategy helps mitigate the potential for loss and ensures the capture of diverse and complex scenes by promoting the selection of images with a broader range of objects while avoiding bias towards specific object classes.

To mitigate redundancy in the sample selection, a diversity method is also employed. Specifically, we utilize the core-set approach described by Haussmann et al. [HFC+20], which was originally proposed by Sener et al. [SS17] within the context of active learning. The core-set approach aims to identify a representative subset of samples that effectively captures the characteristics of the entire dataset. Instead of labeling the entire dataset, this approach focuses on a smaller set of data points. These selected samples serve as a representation of the larger dataset and provide valuable information for the model training. First of all, images are represented using embeddings, which are vector representations. These embeddings capture essential information about the images in a lower-dimensional space. The core-set approach selects centroids iteratively. At each iteration, a new centroid c_i is chosen using the following formula:

$$c_i = arg \max_{x} \min_{c} s(x) d(x, c),$$

where x represents a data-point, $s(x)$ is the uncertainty score of data-point x, and $d(x, c)$ is the distance from the data-point to centroid c [HFC+20]. This means that for each centroid c_i, the data-point x is selected, which maximizes the product of its uncertainty score $s(x)$ and the nearest centroid c. Consequently, the core-set approach chooses data-points that are both uncertain and far away from existing centroids.

Additionally, a class balancing method is incorporated, aiming to select samples matching a desired distribution of different classes as closely as possible. The goal of this method is to give preference to underrepresented classes. This underrepresentation occurs in imbalanced datasets like autonomous driving datasets. For instance, classes like "bus" or "bicycle" may be underrepresented compared to classes like "car" or "person" due to their lower occurrence in typical traffic scenes. As a result, these underrepresented classes are assigned higher distribution values than those classes that occur more frequently. In this framework, the server provides the balancing values for the selection which is described in Section 4.3.

Before transmitting the selected images, the clients check the available network bandwidth and receive a network utility value accordingly. This value determines the number of samples the client sends to the server.

4.3 Server Functionalities

The processes of the server, including data reception from clients, aggregation into a data pool, annotating the samples, model training with its evaluation, and feedback generation for the client, are illustrated

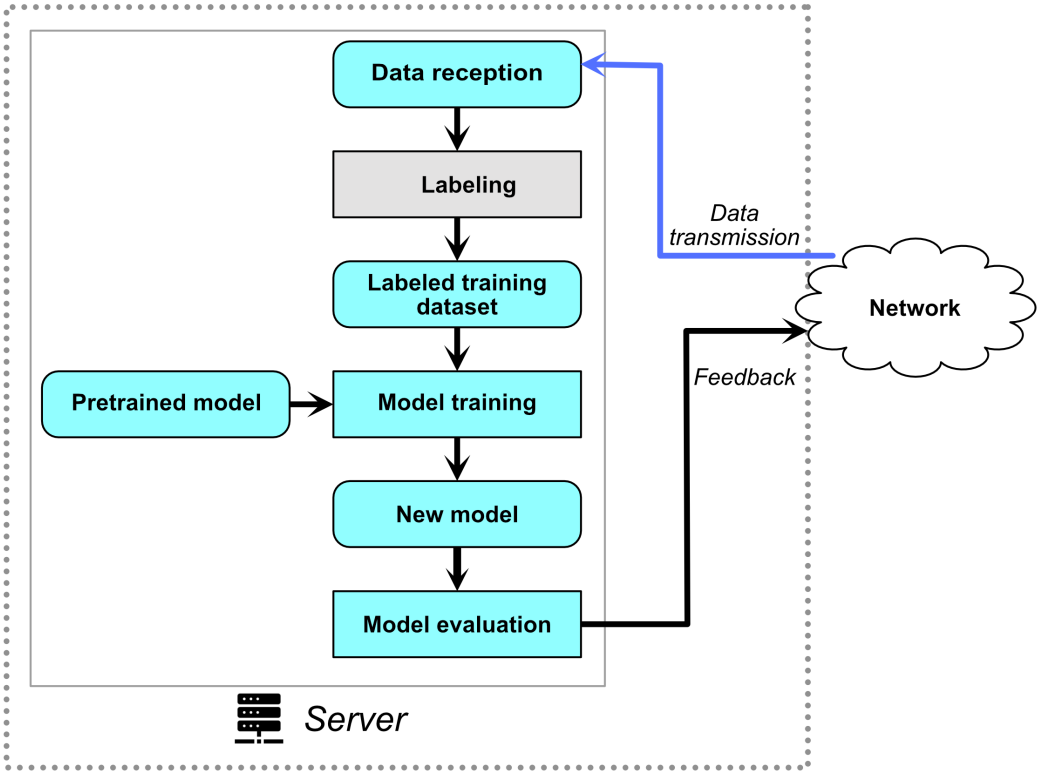

Figure 4.3: Server process steps: Reception of selected data from clients. Annotating the unlabeled data pool through a labeling process, for instance by a human annotator. Model training utilizing labeled data pool, using initially a pre-trained model. Subsequent training iterations utilize the best model from the previous iteration. Evaluation of trained model, followed by sending feedback messages to all clients for further optimization.

in Figure 4.3. Initially, the server receives the transmitted data samples from the clients, which are then gathered to a centralized data pool for subsequent processing. These data are subjected to a labeling process, done by for instance a human annotator, who annotates data provided by the clients. Subsequently, these annotated data are utilized for the model training within our framework, where we employ YOLOv8 [JCQ23] for this purpose. Following the labeling process, model training starts initially with a pre-trained model. Subsequent training iterations utilize the best model from the previous training cycle, ensuring progressive refinement. After each iteration, model validation is performed to evaluate performance metrics such as the mAP of the model and individual classes, as introduced in Section 2.5. Furthermore, based on the current, previous, and penultimate performance of each class, the server calculates the increment in mAP for each class. Using these increment values for each class, the server adjusts the balancing method values and provides this feedback to the client for further optimization of the selection.

4.4 Summary

This chapter has introduced the distributed ALOD framework, explaining the tasks of the client and server. Clients utilize entropy as an uncertainty estimation method and core-set as a diversity method to select the most informative samples. Additionally, a balancing method influenced by server feedback and a method for prioritizing multiple objects in an image is employed. After sample selection, clients transmit their data to the server, considering the network utility. The server receives and processes the data by annotating it and providing the labeled dataset for model training. Finally, the trained model is evaluated and feedback is sent to the vehicles for the next AL cycle.

5 Implementation

This chapter presents the implementation of our proposed distributed ALOD framework. First of all, Section 5.1 presents the technologies and hardware components used to build this framework. Subsequently, Section 5.2 describes the data preparations essential prior to the framework's execution. Section 5.3 and Section 5.4 illustrate the processes of the client and server respectively and how they interact with each other. Finally, Section 5.5 lists the limitations of the implementation, and Section 5.6 summarizes this chapter.

5.1 Environment

ALOD operates within the field of machine learning (ML), more precisely deep learning (DL), and computer vision. Consequently, it requires a programming language, which supports DL algorithms and models. Hence, we are using the programming language Python which is a widely adopted language for DL and data science tasks. Moreover, Python serves as the foundation for the majority of ALOD approaches documented in the literature. Furthermore, we employ PyTorch[1] for training our object detection model and accelerate the model training on GPUs by leveraging NVIDIA CUDA[2]. Additionally, we utilize Lightly[3] a platform to curate and manage our data effectively. We utilize as our object detection model YOLOv8, a state-of-the-art model published by ultralytics [JCQ23]. For experimentation and evaluation, we use the nuImages [CBL+19] dataset. This dataset comprises data collected from autonomous vehicles, providing a rich and diverse set of real-world scenes. Furthermore, the computing system consists of a NVIDIA RTX 4070 GPU and an Intel Core i5-9600K CPU.

5.2 Data Preparation

Before starting an AL cycle within the framework, several data preparation tasks need to be completed. Initially, the ground truth labels of the nuImage dataset are stored as JSON files and need to be extracted and formatted into YOLO format. For YOLO, each image requires a corresponding text file with the same name if the image contains objects. Consequently, if the image has no objects, there will be no label file available. Each line in the text file contains the following information for each object:

class_id, x_centre, x_centre, width, height

To correctly identify the labels for each image and convert them into YOLO, Algorithm 1 is employed. After all labels are extracted for the training and validation dataset into individual folders, the images also need to be organized into respective training and validation datasets. Moreover, we are utilizing multiple clients for the framework, thus we split the training dataset into equal amounts of images for each client's data pool. However, the validation images and labels as well as the training labels, are stored on the server. Hence, the training labels will be used as the labeling process, as illustrated by the gray rectangle labeled "Labeling" in Figure 4.3. Furthermore, in Figure 4.2, the object detection task is depicted within a gray box. This is because, in this framework setup, an initially pre-trained YOLOv8 model is used to predict all objects in the data pool of each client. Notably, this prediction task, aimed at obtaining all the prediction probabilities, is done in this process only at the beginning of the framework. The reason behind this approach arises from the fact that the trained model on the server side isn't updated directly after each iteration. This design mirrors real-world scenarios, where it wouldn't be feasible to update the model after every iteration due to resource constraints.

[1] https://pytorch.org/
[2] https://developer.nvidia.com/cuda-downloads
[3] https://www.lightly.ai

Algorithm 1 NuImages Label Extraction into YOLO Format

Define classes *person*, *bicycle*, *car*, *motorcycle*, *bus*, *truck* with NuImages category names
Define CLASS_NUMBER for each class

Procedure extract_label(key_camera_token):
Define $x1$, $y1$, $x2$, $y2$, c
Get token of image
Get filename of image
Create text file with *filename* as $file\_object$
Get a list of objects with associated token
for each object **do**
    Get category name
    Set c to the corresponding CLASS_NUMBER
    if annotation mask is not None **then**
        Get img_height
        Get img_width
        Get $x1$, $y1$, $x2$, and $y2$ from *bbox*
        Calculate x_centre as $((x2-x1)/2)+x1$
        Calculate y_centre as $((y2-y1)/2)+y1$
        Calculate $width = x2-x1$ and $height = y2-y1$ {height and width of bounding box}
        Normalize x_centre, y_centre, $width$, and $height$ by dividing them by img_w and img_h respectively
        if c is within our defined classes **then**
            Write "{c}{x_centre}{y_centre}{$width$}{$height$}" to $file\_object$
        end if
    end if
end for

while there are remaining samples **do**
    Get key_camera_token from each sample
    Call extract_label function with key_camera_token as argument
end while

5.3 Client Process

The process of a client starts with the connection to the server via sockets. To enable multiple clients, threads are employed. Once a client has successfully connected to the server, it receives a balancing distribution for the upcoming selection process from the server. For the first five iterations, the client receives the initial balancing distribution outlined in Table 5.1. This table presents six classes considered for the selection with their corresponding target values. Additionally, the table displays the adjusted target values according to the number of objects in the nuImages dataset, with the middle column reflecting this adjustment, and the right column illustrating randomly assigned target values. After the client receives the balancing distribution, it initiates the selection process utilizing the proposed methods outlined in Section 4.2: entropy, core-set, frequency estimation, and balancing. After the selection process selected n samples, the client checks the network utility value u ranging between $[0,1]$. Subsequently, it transmits $(1-u) \times n$ samples to the server. This client process is shown in Algorithm 2 and iterates for $k \in \mathbb{N}$ iterations.

Table 5.1: Initial balancing distribution for the selection process.

Object Class	**Target Value** (adjusted)	**Target Value** (random)
Person	0.1	0.175
Bicycle	0.225	0.1
Car	0.1	0.175
Motorcycle	0.225	0.15
Bus	0.225	0.2
Truck	0.125	0.2

Algorithm 2 Client process to select informative samples.

```
Run object detection & get prediction probabilities for each object
for j = 0 to k ∈ ℕ do
    repeat
        Connect to the server
        Receive Balancing distribution
        Start Selection
        repeat
            Calculate Entropy H(I_{i,d})
            Calculate the object frequency
        until checked every image I
        Select samples using Balancing & Core-Set approach
        Receive network utility value u
        Send data according to u
    until every client has sent data
end for
```

5.4 Server Process

The server process starts with the evaluation of the latest training cycle. Moreover, mAP values of the latest model and of every class are calculated and saved for further processing. Starting from the 5th iteration onwards, the mAP increase of each class is computed based on values of the last three iterations: the current (j), last ($j-1$), and penultimate ($j-2$) iteration. Using these increased values, the latest balancing distribution is adjusted. Specifically, the distribution value of the class with the highest increase (h_1) is decreased by value v_1, while the distribution value of the class with the lowest increase (l_1) is increased by the same amount v_1. This adjustment is also applied to the class with the second highest (h_2) and second lowest increase (l_2) by value v_2. However, all four adjustments are done if the increase of the two lowest classes does not exceed a certain threshold t_1, as outlined in Algorithm 3. Furthermore, the distribution values of the classes with the highest and second highest increase should not fall below a certain threshold t_2. These updated balancing distribution values are then transmitted to the clients as feedback to refine their selection process. However, if the current iteration is below the 5th iteration, the initial balancing distribution is sent to the clients.

After sending the feedback messages, the server awaits the reception of the unlabeled dataset from the connected clients. Subsequently, it initiates the labeling process by utilizing the ground truth labels provided by the nuImages dataset. Before the model training starts, the server randomly selects additional images from the validation data pool of 20% × the amount of samples initially chosen by each client,

Algorithm 3 Server feedback calculation.

```
if j ≥ 5 then
  if class_with_lowest_increase < t_1 then
    if h_1 ≥ t_2 then
      class_with_lowest_increase = l_1.value + v_1
      class_with_highest_increase = h_1.value - v_1
      update distribution values
    end if
  end if
  if class_with_second_lowest_increase < t_1 then
    if h_2 ≥ t_2 then
      class_with_second_lowest_increase = l_2.value + v_2
      class_with_second_highest_increase = h_2.value - v_2
      update distribution values
    end if
  end if
end if
```

along with the corresponding labels. Consequently, for each training cycle, the server's training pool consists of $\frac{1}{6}$ validation samples and $\frac{5}{6}$ training samples if the clients send all the samples they selected. The decision to store the validation dataset in the server rather than extracting it from the clients' data pool serves two primary purposes. Firstly, it prevents further reduction of the training data pool. Secondly, it allows to provide unbiased and random validation samples. This selected validation data pool and labeled training data pool are provided for the training of the YOLOv8 model. Moreover, it takes the best model of the previous training cycle for training, except for the first iteration where the pre-trained YOLOv8 model on the COCO dataset is utilized, which is provided by ultralytics [JCQ23]. The entire server process iterates for $k \in \mathbb{N}$ iterations, as illustrated in Algorithm 4.

Algorithm 4 Client process to select informative samples.

```
for j = 0 to k ∈ ℕ do
  Clients connect to the server
  Perform feedback calculation
  Send feedback to all connected clients
  Reception of unlabeled data pool from all clients
  Labeling
  Validation data pool selection
  Model training
end for
```

5.5 Limitations

The framework has two limitations. The first limitation is that it does not utilize a network simulation tool such as OMNeT++[4] to emulate a realistic communication environment because it would go beyond the scope of this work. Instead, we approximate the network conditions through the network utility value, representing the available network bandwidth, which allows a certain number of images to be transmitted by the clients. Furthermore, the values within the feedback algorithm are selected based on the trial and error methodology. These values are depicted in Table 5.2.

[4] https://omnetpp.org/

Table 5.2: Feedback algorithm values.

Parameter	Value
t_1	0.01
t_2	0.1
v_1	0.05
v_2	0.025

5.6 Summary

This chapter has introduced the implementation of our framework. It began by outlining the technologies and computation environment utilized to construct the framework. Additionally, it depicted the data preparation steps that needed to be done before running the ALOD framework. Subsequently, the process steps of the client and server are explained, and how they interact with each other. Finally, the two limitations of the framework are stated. Firstly, the absence of a network simulation tool like OmNeT++. Secondly, the use of values with the feedback algorithm is determined through trial and error.

6 Evaluation

In the two previous chapters, the design and implementation of the distributed ALOD framework were described. The focus of the evaluation will be on the proposed framework and how it performs in the different setups. Hence, the performances of the setups are investigated by using the metric mean Average Precision (mAP). Furthermore, this approach should show the feasibility of integrating clients into the process of active learning for object detection to mimic the usage in the real world. Section 6.1 describes the different evaluation setups. Subsequently, Section 6.2 shows the results of these evaluation setups. Finally, Section 6.3 analyzes the obtained results and sums them up.

6.1 Evaluation Setup

For the evaluation of the proposed distributed ALOD framework, we conduct three main setups:

- centralized approach as a reference setup,
- distributed ALOD without feedback approach,
- and distributed ALOD with feedback.

Thus, the goal of the comparison between the centralized setup and our proposed framework is a proof of concept. In the centralized approach, we select in each iteration 1000 images for training and 200 images for validation. Moreover, the setup of this approach is without a feedback configuration. For the distributed ALOD approach without feedback, we consider two different balancing distributions. Based on the results from this setup the balancing distribution with the better results is used for the subsequent evaluation setups incorporating the feedback algorithm. Additionally, we evaluate each of the distributed ALOD approaches with five different network utility setups: 0%, 30%, 60%, 90%, and randomized network utility. An overview of all the setups and their names in the evaluation are depicted in Table 6.1. Our evaluation framework employs ten clients, each equipped with an equal share of data randomly distributed from the nuImages dataset [CBL+19]. Each of the clients, selects 100 samples, while the server adds for each client 20 further randomly selected samples into the validation pool. This dataset comprises 96,000 images capturing diverse driving scenarios relevant to autonomous driving. To perform object detection, we utilize the YOLOv8 detection model, specifically the YOLOv8m variant. Each evaluation setup spans ten iterations, with the model trained for 20 epochs in each iteration. A batch size of 16 and a learning rate of 0.01 throughout the training process is maintained. To measure the performance of setups we utilize the mAP50, mAP75, and mAP50-95. mAP50 and mAP75 calculate the average precision at the IoU threshold of 0.50 and 0.75, respectively. This means that a predicted bounding box is considered correct if the overlaps of the prediction and the ground truth bounding boxes are at least 50% or 75%. mAP50-95 in contrast calculates the average precision at varying IoU thresholds, ranging from 0.5 to 0.95. These values provide are more comprehensive assessment of the model's performance because it captures the performance across different IoU thresholds. Additionally, the class imbalance is investigated using precision, recall, and F1-score to investigate which classes are underrepresented.

6.2 Evaluation Results

This Section presents evaluation results based on the evaluation setups presented in the previous section. Therefore, the model performances of each setup are investigated. The first Subsection 6.2.1 compares the distributed ALOD framework with the centralized approach, which is used as a reference. Here, the general performance of the setups applying only the network utility is compared. Subsequently,

Evaluation Setup	Feedback	Network Utility	Balancing distribution	Name
Centralized setup	no	no / 0%	adjusted	Central
Distributed ALOD	no	no / 0%	adjusted	CS_None_A
	no	random	adjusted	CS_Rand_A
	no	30%	adjusted	CS_30_A
	no	60%	adjusted	CS_60_A
	no	90%	adjusted	CS_90_A
Distributed ALOD	yes	no / 0%	adjusted	CS_None_AF
	yes	random	adjusted	CS_Rand_AF
	yes	30%	adjusted	CS_30_AF
	yes	60%	adjusted	CS_60_AF
	yes	90%	adjusted	CS_90_AF
Distributed ALOD	yes	no / 0%	random	CS_None_RF
	yes	random	random	CS_Rand_RF
	yes	30%	random	CS_30_RF
	yes	60%	random	CS_60_RF
	yes	90%	random	CS_90_RF

Table 6.1: Overview of all evaluation setups

Subsection 6.2.2 compares the setups employing the feedback algorithm, while getting more insights into class-wise performances. Finally, the last Subsection 6.2.3, presents the performances of all conducted evaluation setups.

6.2.1 Ressource-Aware Distributed ALOD without Feedback

This subsection presents the results of evaluating the distributed ALOD framework without the feedback algorithm enabled. First, we conducted the model training, configuring all clients to send every selected data to the server.

Figure 6.1 and Figure 6.2 depict the model training results of the central approach and the distributed ALOD approaches. In these figures, the distributed ALOD approach employing adjusted balancing values is designated as CS_None_A, while the approach utilizing the initially randomly selected values is denoted as CS_None_R. The corresponding balancing values are detailed in Table 5.1. Figure 6.1 illustrates the mAP50 values on the y-axis against the epochs on the x-axis. Conversely, Figure 6.2 shows the number of utilized data on the x-axis. Examination of the line plots reveals that the distributed approach with adjusted balancing values CS_None_A consistently outperforms the CS_None_R approach employing random balancing values. Additionally, it is noteworthy that the central approach outperforms both client-server approaches only towards the end of the active learning iterations. The reason why the distributed approaches achieve better values at the beginning could be that each of the clients selects from their dataset the most informative samples, which are compared to the samples of the central setup already specific samples. Thus the central setup has at the beginning a slightly more general model. Furthermore, due to the method that prioritizes images with many objects, the server selects those images first, which might not have objects that are underrepresented. Further investigation on this is conducted in the later stage of the evaluation when class-wise performances are analyzed.
Detailed mAP50 values for these three models at the end of the ten active learning iterations are pre-

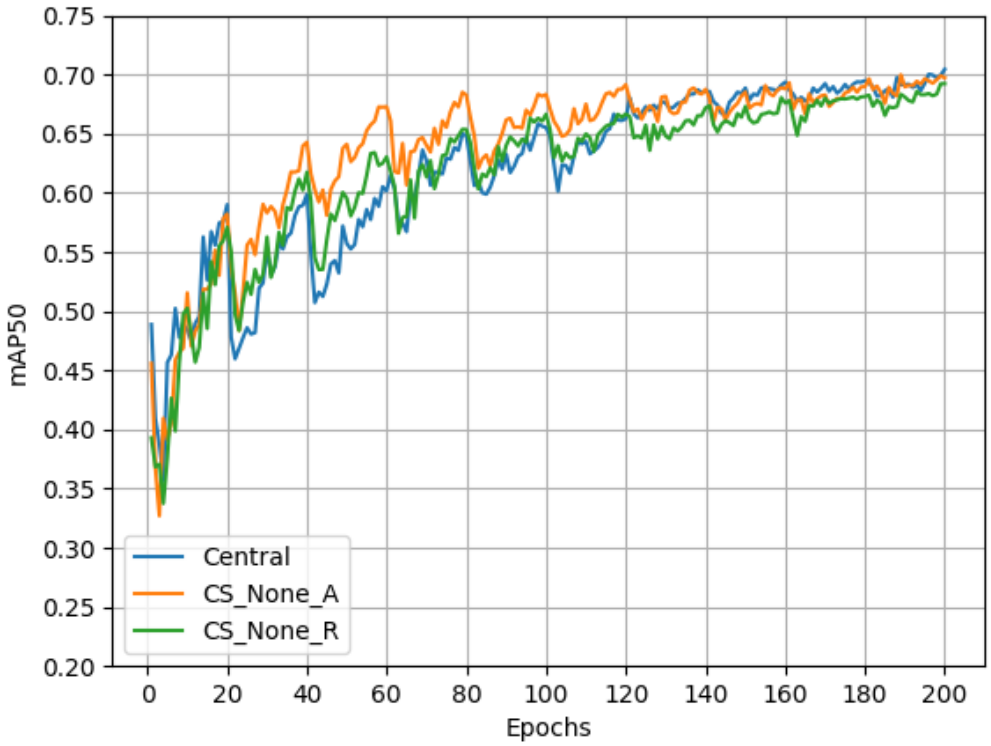

Figure 6.1: Comparison of mAP50 values between setups utilizing adjusted balancing (CS_None_A) and random balancing (CS_None_R) with respect to the central setup across epochs.

sented in Table 6.2. Notably, the central approach outperforms CS_None_A only by 0.0045, while CS_None_A outperforms CS_None_R by 0.0075, which are marginal differences. Based on these results, further ALOD runs considering only the network utility are done, using the adjusted balancing distribution values.

	mAP50
Central	0.70442
CS_None_A	0.69985
CS_None_R	0.69231

Table 6.2: mAP50 results of the central approach and the two distributed approaches at the 10th iteration.

Figure 6.3 presents the results of the distributed ALOD approach with different network utility configurations. Notably, in the setup denoted called CS_30_A, the number indicates the network utility value each of the clients receives. This means they are sending 30% fewer data samples to the server. On the other hand, in the setup of CS_Rand_A each client receives a randomized network utility value in the range from 0 to 1. Observing these results it is not surprising that the setups with higher network utility value achieve lower mAP50 scores due to the decreased amount of data that is sent to the server. Thus these performances were expected. A more comprehensive inside of the amount of data used and the corresponding mAP50 values are listed in Table 6.3 and visualized in Figure 6.4. These resources provide insights into the interplay between network utility, data usage, and model performance within the distributed ALOD framework.

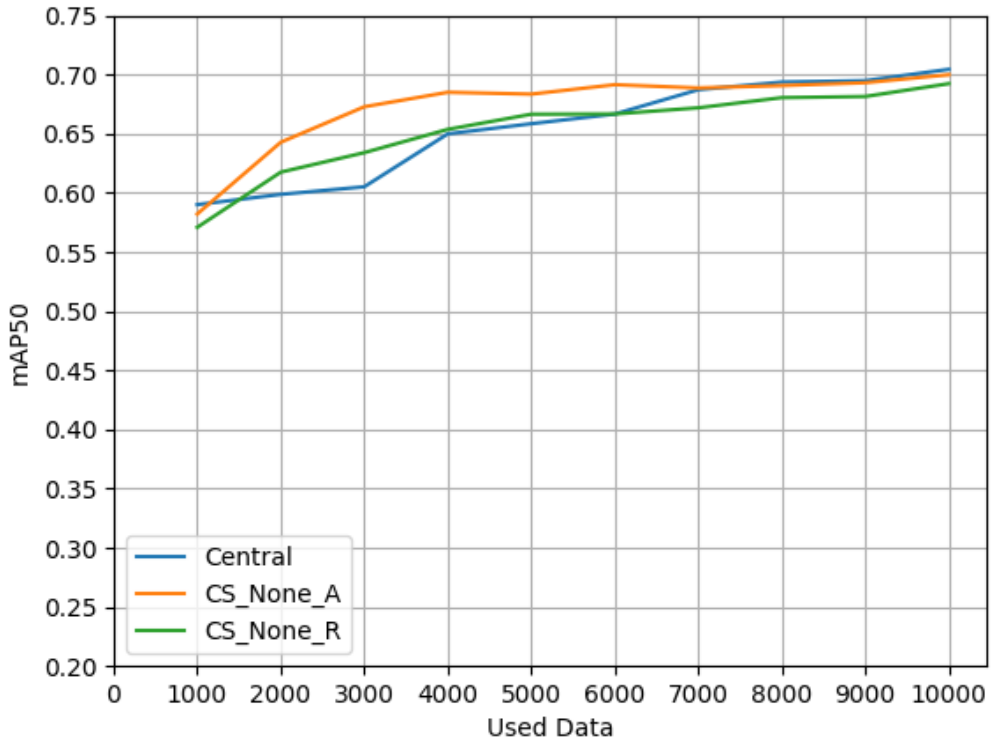

Figure 6.2: Best mAP50 values observed at each iteration within the central, as well as in setups employing CS_None_A, and CS_None_R.

It is interesting to observe from Figure 6.4 and Table 6.3 that despite the utilization of over 4000 images, the discrepancy between each setup is relatively minimal. Especially, this can be seen by the values of CS_Rand_N and CS_30_N, where there is not much improvement with additional 2300 images.

6.2.2 Ressource-Aware Distributed ALOD with Feedback

This subsection deals with the distributed ALOD approaches that integrate both the network utility and the feedback algorithm which is explained in Section 5.4. However, in contrast to the previous subsection the randomly selected balancing values will also be considered in the different evaluation setups. First, the performance of CS_None_AF is compared with the central setup, illustrated in Figure 6.5. It shows that both approaches have a very similar curve, while the central setup outperforms CS_None_AF at

	mAP50	Used Data
CS_30_A	0.68573	7000
CS_60_A	0.63537	4000
CS_90_A	0.5566	1000
CS_Rand_A	0.67903	4706
CS_None_A	0.69985	10000

Table 6.3: Amount of used data and mAP50 results of the distributed ALOD considering only network utility.

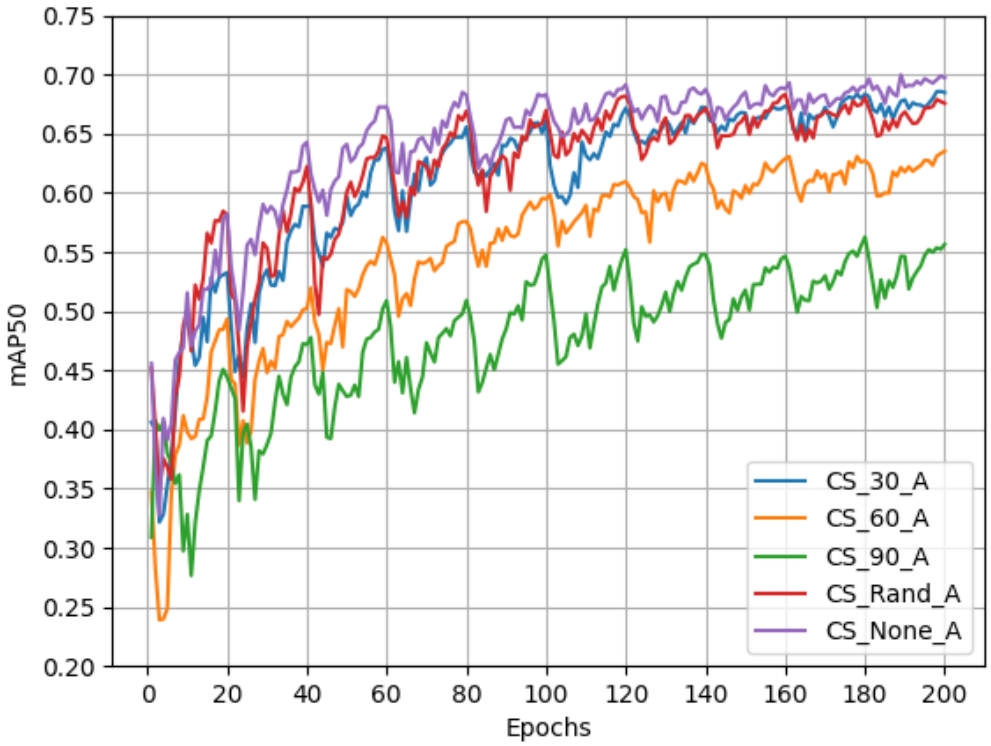

Figure 6.3: The distributed ALOD setups only considering the network utility.

120 Epochs or the 6th iteration. Nevertheless, this means that in the first few iterations, the approach CS_None_A and CS_None_R have slightly superior performances by comparing this figure with Figure 6.1. The reason why these four setups have different curves is that each client has different datasets and thus different distributions of the object classes. This becomes clearer in the evaluation of the individual class performances.

Table 6.4 presents the results of the approaches initially employing adjusted balancing, denoted by the suffix "AF", alongside randomly selected balancing denoted by "RF". The table shows that the setup CS_30_AF and CS_None_AF outperform the equivalent setup CS_30_RF and CS_None_R each by under 1%, respectively. However, in all other setups using 60%, 90%, and randomized network utility, the setups with the unadapted balancing value achieve better mAP50 results.

The feedback algorithm aims to identify classes with low performance improvements, indicated by their mAP50 values after each iteration, and allocate these classes a higher preference balancing percentage. Table 6.5 shows the mAP50 class metrics of the three distributed setups, which have received and utilized all selected samples compared with the centralized setup. First, observing the balancing values Bal_AF of the setup CS_None_AF in the last iteration, and comparing them with BAL_R of CS_None_R, a significant difference can be observed by three classes: bicycle, motorcycle, and truck. Notably. the bicycle and motorcycle classes exhibit a balancing improvement of 15% and 7.5%, respectively, along with a nearly 2% and 5% improvement in mAP50. However, despite the decrease in balancing for the truck class, as illustrated in Figure 6.6, it nevertheless maintains a similar performance as CS_None_R. Furthermore, a comparison between the CS_None_AF and CS_None_A results reveals that CS_None_A outperforms CS_None_AF, especially in the underrepresented classes like bicycle and truck. Similar results can be seen in the evaluation setups with 30% network utility, as depicted in

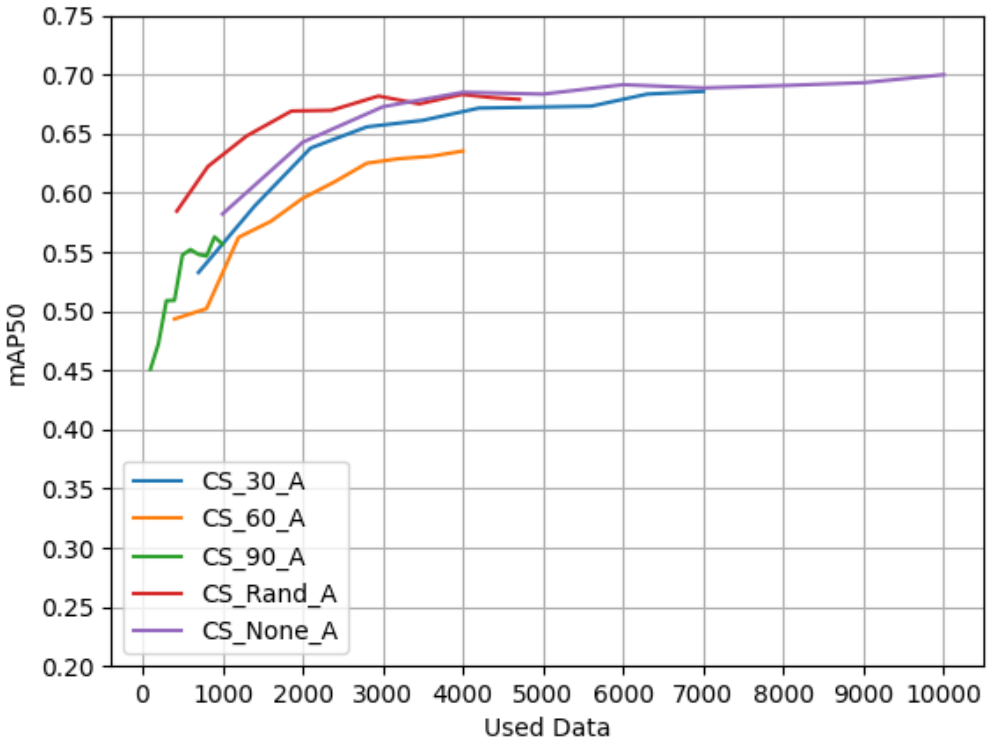

Figure 6.4: Distributed ALOD considering network utility: Visualization of the model performance based on their used data.

Table 6.6. CS_30_AF manages to outperform the other setups in the classes person and bicycle, while slightly lagging behind in the other classes where other setups achieve slightly better mAP50 values.

Figure 6.7 and Figure 6.8 show the curve of each class of the setups CS_None_AF and CS_None_A respectively. These curves first of all show, how different the selection of the samples is in these setups. The reason behind the different curves is that every client in every setup that is conducted in this work has a different subset of the nuImages dataset. Thus every client has different informative samples to provide from its dataset. This can be observed for instance by the classes truck and bicycle of the aforementioned figures. In CS_None_A the bicycle class has mAP50 values below 0.55, while the same class in CS_None_AF already started with mAP50 scores above 0.55. Similar happened to the class truck, while CS_None_A achieves scores above 0.5, the mAP50 values of CS_None_AF reached that score after the third iteration. Another reason why the curves of the classes bicycle, bus, motorcycle, and truck vary in each setup is that these classes are the underrepresented ones in the nuImages dataset. This means that in the distribution of the dataset to the clients, the number of samples of these classes differs a lot in each data pool of the clients. Hence, some classes might have many images of trucks while other clients have only a few. Because of this difference in the number of the individual object classes, the classes car and person, which are represented the most in the nuImages dataset, have comparably a stable curve. However, as depicted in Table 6.5 and Table 6.6 to the end of the conducted iterations the mAP50 values slightly reached similar scores, indicating also that the model slowly doesn't get many useful samples that the mAP values increase much more.

To gain deeper insights into the performance of the models, especially with respect to individual classes, the metrics precision, recall, and F1 score are investigated, which are described in Section 2.5.

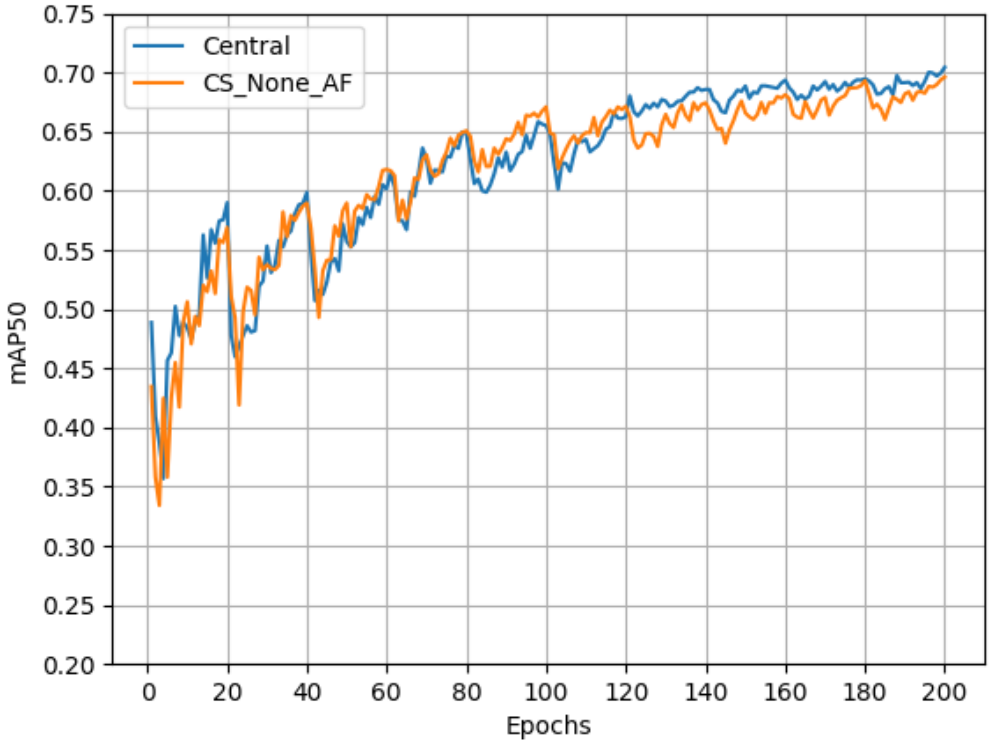

Figure 6.5: Comparison of the setups Central and CS_None_AF

	mAP50	Used Data
CS_30_AF	0.68458	7000
CS_30_RF	0.67818	7000
CS_60_AF	0.62996	4000
CS_60_RF	0.65771	4000
CS_90_AF	0.54739	1000
CS_90_RF	0.56863	1000
CS_Rand_AF	0.6513	4933
CS_Rand_RF	0.66762	5336
CS_None_AF	0.69623	10000
CS_None_R	0.69231	10000

Table 6.4: Distributed ALOD with feedback algorithm: comparing results with adjusted (AF) and random (RF) balancing.

	Central	CS_None_AF	CS_None_A	CS_None_R	Bal_AF	Bal_A	Bal_R
Person	0.68409	0.66043	0.65977	0.66358	0.125	0.1	0.175
Bicycle	0.699	0.70694	0.72521	0.68771	0.25	0.225	0.1
Car	0.83996	0.8318	0.83344	0.83612	0.175	0.1	0.175
Motorcycle	0.80584	0.82031	0.81061	0.77304	0.225	0.225	0.15
Bus	0.57265	0.57691	0.57528	0.60860	0.15	0.225	0.2
Truck	0.62499	0.58939	0.61993	0.58476	0.075	0.125	0.2

Table 6.5: Class-wise mAP50 and balancing values comparison. Bal_AF shows the balancing values of CS_None_AF at its last iteration. Bal_A presents the balancing values for the centralized approach and CS_None_A. Bal_R lists the values of CS_None_R.

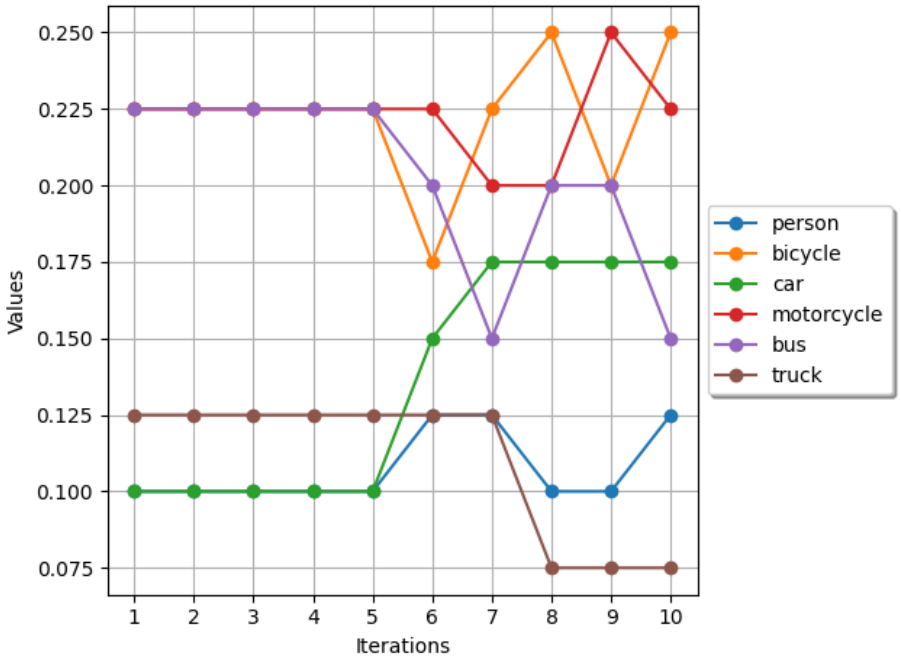

Figure 6.6: Balancing value change of CS_None_AF over iterations.

Therefore, we evaluated these metrics on the central, CS_None_AF, CS_None_A, and CS_None_R configurations. To ensure a fair comparison, we validate these four approaches on the same validation dataset. Table 6.7 presents the precision, recall, and F1-scores of the model of the setups. Examining the precision values, all these models achieve a high accuracy in making positive predictions. This means they can correctly identify approximately 80% of instances. However, the recall values hover around 63%, which indicates that it can identify only 63% of the positive instances. Nonetheless, the F1-Scores are all around 70.7% indicating that the models are still performing well.

Further investigation of the class-wise precision, recall, and F1-scores identifies where the class imbalance exists. For instance, the class-wise metrics of the CS_None_AF are depicted in Table 6.8. Observing the F1-scores of the classes person, bus, and truck, they are only achieving values ranging from 61% to 66%. Despite, the high precision of the person class at 85.8%, its recall is merely 54%. This means, that the model is only able to identify slightly over half of the positive instances in the dataset. Similar recall values are observed within the classes bus and truck. These values are also observed similarly in every other setup. Such an imbalance of the dataset can be analyzed and illustrated with the precision-recall

	CS_30_AF	CS_30_A	CS_30_RF
Person	0.66113	0.64811	0.64137
Bicycle	0.69680	0.68278	0.65683
Car	0.82108	0.83214	0.82228
Motorcycle	0.77208	0.78006	0.75630
Bus	0.59310	0.6103	0.60269
Truck	0.56062	0.58489	0.59541

Table 6.6: Class-wise mAP50 results in the 30% network utility setup.

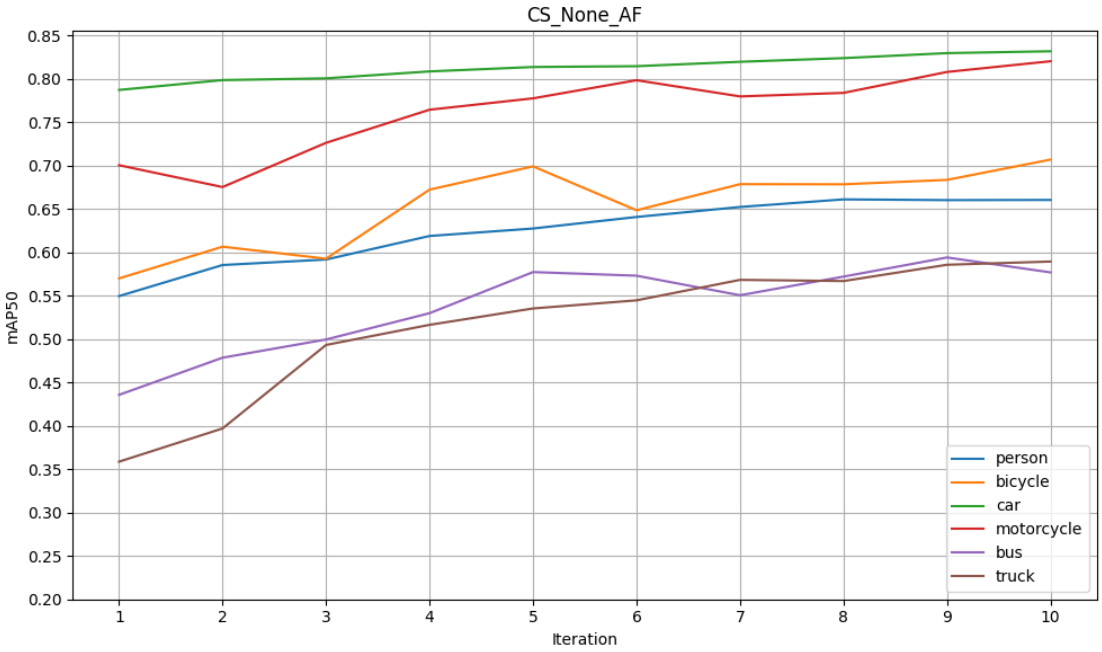

Figure 6.7: Class-wise graph curve of CS_None_AF.

curve, as illustrated in Figure 6.9. The curves show that the model has its difficulties, especially with the classes bus, truck, and person. However, to identify in what way the model struggles with these classes the normalized confusion matrix can be analyzed, illustrated in Figure 6.10. The matrix shows for example that background objects, which are objects of classes that are not considered in our model, are mistakenly predicted as persons. This is seen in the upper right corner and classified as false positive (FP) predictions. Additionally, persons are mistakenly predicted not as a person, which is a false negative (FN). Such mispredictions are also observed across all the other classes. Thus the model requires more training data for these classes.

6.2.3 Overall Setup Comparison

This subsection investigates the performance of the different setups. Moreover, the models are all validated on the same validation dataset to ensure comparison fairness, because the validation datasets while training the models are all randomly picked by the server and thus differ. Furthermore, not only the mAP50 metric is used but also mAP75 and mAP50-95. The first notable thing is that the central setup outperforms all other setups regardless of the different mAP thresholds depicted in Table 6.10. This Table lists all setups that evaluated and presents the mAP scores at IoU thresholds 50%, 75%, and 50% to 95%.

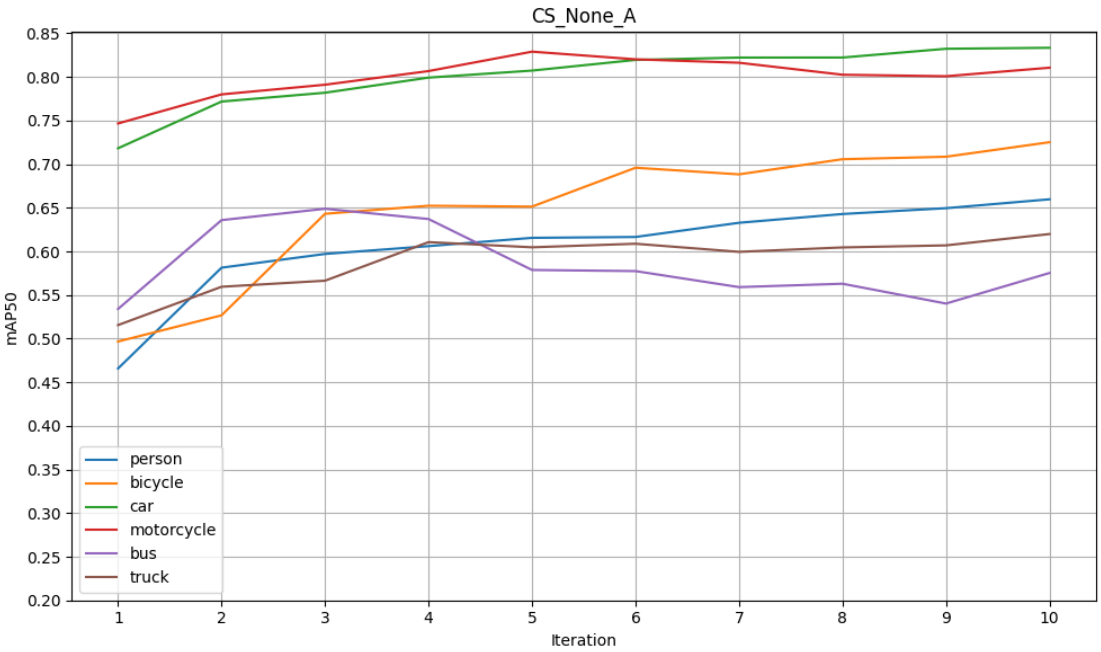

Figure 6.8: Class-wise graph curve of CS_None_A.

	Precision	Recall	F1-Score
Central	0.79556	0.63678	0.70737
CS_None_AF	0.79568	0.63731	0.70774
CS_None_A	0.81048	0.62696	0.70700
CS_None_R	0.81610	0.62307	0.70664

Table 6.7: Precision, Recall, and F1-Score of central and distributed ALOD.

Comparing the setups by their network utility, in most cases either the setup with the adjusted balancing and feedback or the setup with random balancing and feedback has slightly higher performance values. Notable, are the values of the CS_None setup where in the mAP50 CS_None_R marginally is better, while observering mAP75 and mAP50-95 CS_None_A has a slightly better performance. This performance difference within different thresholds is also observable within the 30% network utility setups. Nevertheless, the mAP scores within similar setups differ only marginally.
The same accounts for the performance values of Table 6.11, especially the scores at with approximately 4000 trained images. The performance scores with approximately 2000 images differ more because the models are mostly trained only two or three iterations at this point, depending on the amount of received data. Nevertheless, all the distributed approaches achieved higher mAP50 scores of more than 1% to nearly 5% compared to the centralized setup. This reason might be that the server selects more diverse samples in the first iterations compared to the distributed setups. While the clients have to select informative and diverse samples in a smaller dataset, the resulting dataset gathered by the server might be comparably more specific in certain areas of the feature map. Consequently, the models of the distributed setups are more specified in certain regions of the feature maps, while this happens at later stages in the central approach. Furthermore, due to the fact of prioritizing images with many objects, the central approach consequently does not pick images that have few and possibly rare objects in the first place.

	Precision	Recall	F1-Score
Person	0.85845	0.54377	0.66580
Bicycle	0.73816	0.69485	0.71585
Car	0.86786	0.74855	0.80380
Motorcycle	0.80215	0.76804	0.78472
Bus	0.74324	0.52743	0.61700
Truck	0.76423	0.54122	0.63368

Table 6.8: Class-wise Precision, Recall, and F1-Score of CS_None_AF.

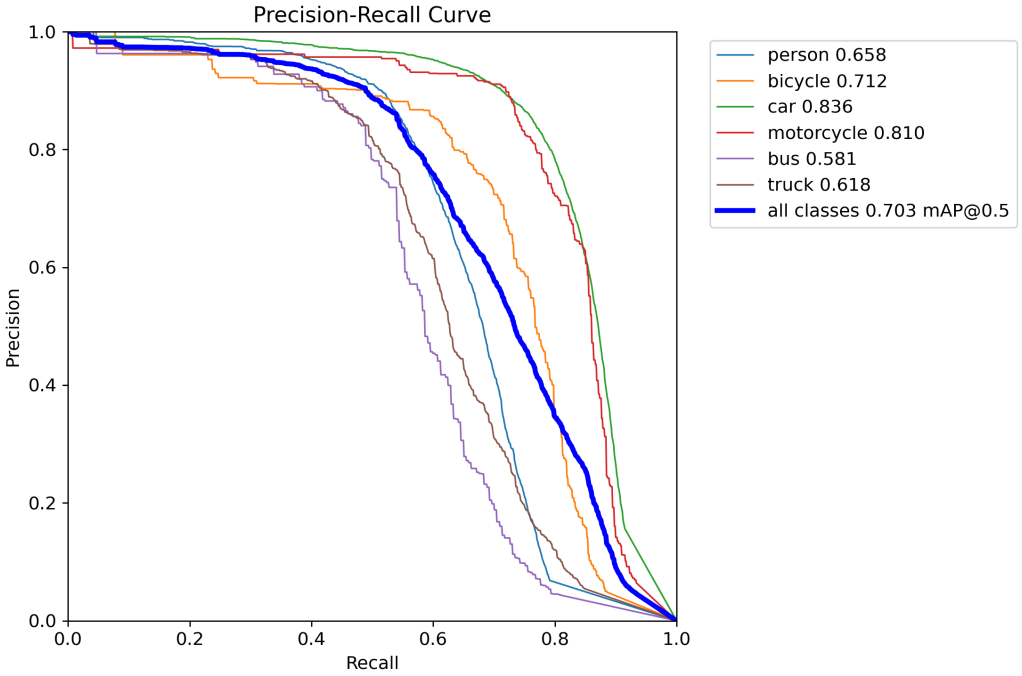

Figure 6.9: Precision-Recall Curve of CS_None_AF at iteration ten.

6.3 Analysis of Results

After presenting the results of the different distributed ALOD setups, this section analyses and discusses the obtained results. First, we investigated the performance of our distributed ALOD framework against the centralized approach. For our distributed ALOD, we conducted setups utilizing adjusted balancing values (CS_None_A) and randomly selected balancing values (CS_None_R) of the class distribution, which are depicted in Table 5.1. The results revealed that both distributed setups, each employing ten clients, managed to achieve comparable results, with mAP50 differences of around 1% below the score of the centralized setup. Similar results of slightly below 1% difference applied investigating the mAP75 and mAP50-95 scores.

Afterward, the results of setups only employing different network utilities are investigated. The mAP50 scores indicated that the performance of the models decreased as the server received less data. Interestingly, minimal differences were observed whether each of the clients sent the same reduced amount of samples or the clients were individually sending different amounts of samples, denoted with CS_Rand_(A/AF/RF), regardless of using the feedback algorithm or not, in the comparison of mAP50

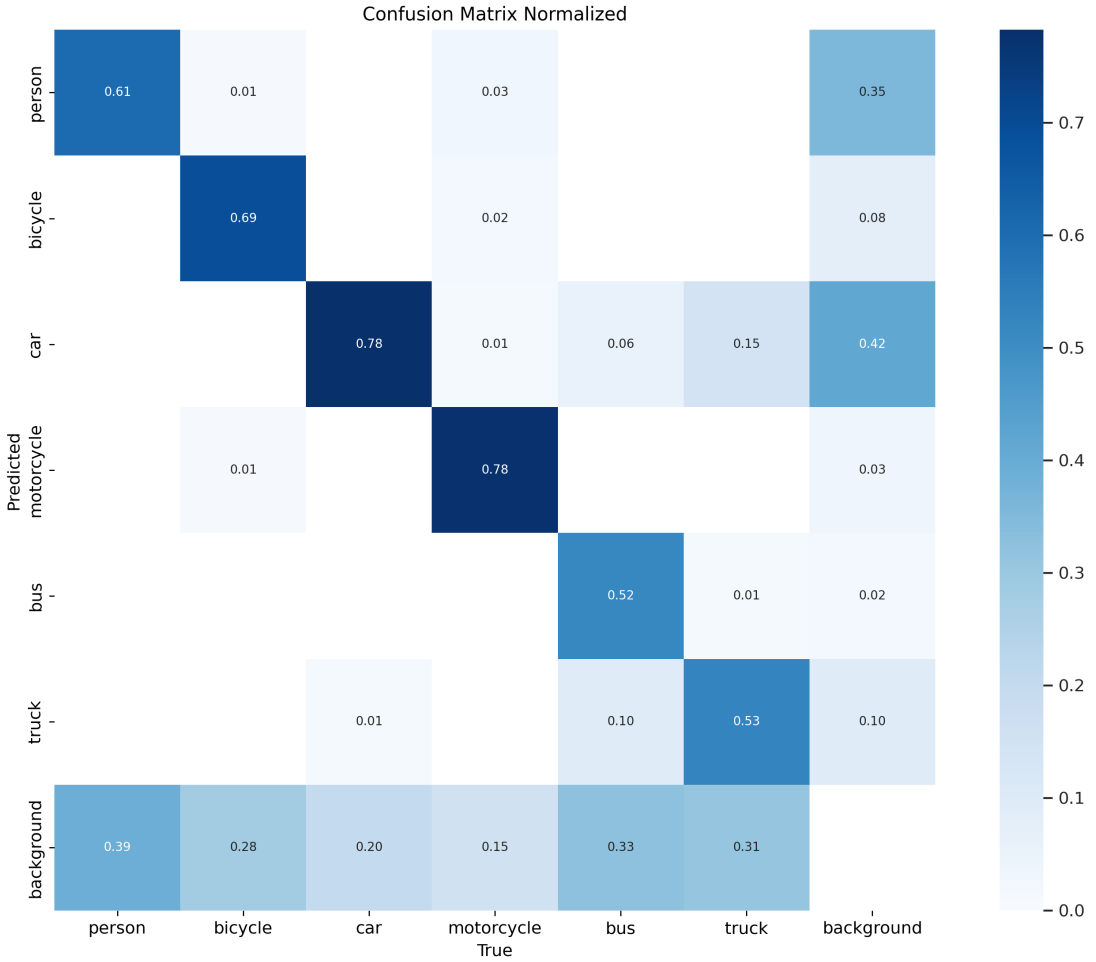

Figure 6.10: Normalized Confusion Matrix of setup CS_None_AF

values trained on a similar amount of data, illustrated in Table 6.11.
Moreover, we analyzed the impact of the server sending feedback to the clients in the form of adjusted balancing values. The results show that the feedback had only slight positive impacts on the overall model performance in certain setups. This can be attributed to two main factors: firstly, the adjustments did not start from the beginning of the AL iterations because the model has to be trained for some iterations to identify low performing classes. Thus, the feedback algorithm only has a few iterations because only ten training iterations are done for each setup. Secondly, the approach has to make a trade-off between high performing classes and low performing classes. This trade-off notably affected especially classes that are not much represented in the dataset. As depicted in Table 6.5, for instance, the class truck experienced a negative impact, while the class motorcycle experienced improved scores compared with other setups. This is reflected by the balancing scores illustrated in Figure 6.6. Similar results with the classes truck and bicycle occurred, illustrated in Table 6.6.
This imbalance in datasets generally affects the performance of underrepresented classes due to the lag of instances which is the case in the nuImages dataset. Due to this imbalance, the models are not able to fine tune object classes, thus models struggle to predict these objects correctly which is illustrated in the confusion matrix in Figure 6.10. Furthermore, the selection process does not necessarily have to adhere exactly to the balance values, as the unbalanced dataset may mean that there are not enough of these instances available. Consequently, the feedback algorithm could not reach its full potential to balance the class-wise performance in this evaluation because of the aforementioned reasons.
Furthermore, all the evaluation setups are validated on the same validation dataset to achieve a fair

Table 6.9: Comparison of the performance of the best model of all approaches based

	mAP50	mAP75	mAP50-95
Central	0.71097	0.51598	0.48605
CS_None_A	0.70404	0.51113	0.48241
CS_None_R	0.70485	0.50830	0.48039
CS_None_AF	0.70254	0.50414	0.47944
CS_30_A	0.68018	0.48870	0.46296
CS_30_AF	0.69036	0.48122	0.46441
CS_30_RF	0.68294	0.48941	0.46627
CS_60_A	0.65739	0.46784	0.44191
CS_60_AF	0.65077	0.45846	0.43641
CS_60_RF	0.65728	0.45956	0.43694
CS_90_A	0.57323	0.38156	0.36888
CS_90_AF	0.56846	0.37223	0.36473
CS_90_RF	0.57901	0.38522	0.37214
CS_Rand_A	0.65971	0.46677	0.44379
CS_Rand_AF	0.67019	0.47472	0.45114
CS_Rand_RF	0.68074	0.47759	0.45604

Table 6.10: Performance comparison of all setup regarding mAP50, mAP75, and mAP50-95

comparison, illustrated in Table 6.10. Some of these setups gained more mAP scores. Notably, is that based on these scores the randomized setups with enabled feedback CS_Rand_AF and CS_Rand_RF have higher performance compared to the setup without feedback, which was reversed with the individually selected validation dataset during the training process. Furthermore, here CS_Rand_RF nearly reached similar performance scores as the setups with 30% network utility, even though CS_Rand_RF used less data.
Additionally, the performance of several setups after approximately 2000 and 4000 data samples are analyzed, depicted in Table 6.11. The table reveals that there are only minimal differences in model performances starting at 4000 data samples. Furthermore, Table 6.11 and Figure 6.1 show that the centralized approach performs worse than the distributed ALOD setups. The reason might be that clients are selecting informative and diverse samples within their local data set. However, by gathering all these samples they are slightly more specific in some regions of the feature map, while the server has comparably a more distributed selected dataset. Consequently, the distributed ALOD models are performing better in these specific feature map areas. This can be seen in Table 6.12, where the distributed approach dominates the central setup in the classes bicycle, motorcycle, and bus. These differences can be seen by comparing the class-wise performance of Central in Figure 6.11 with the Figures 6.7, 6.8, 6.12 of CS_None_AF, CS_None_A, and CS_None_R respectively. Another reason might be the implementation of the method which prioritizes many instances in an image. Thus, the central setup ranks images that are useful for the model but contain fewer instances than other images slightly lower. Consequently, these useful images containing for instance the object bus are selected in a later stage.

In summary, investigating all the performance scores, the distributed ALOD framework is capable of providing equally informative samples for the server to train as in the central ALOD setup. Notably, unlike the central approach, the newly trained models after each active learning iteration are not updated to the clients for predicting the unlabeled data pool again to identify new potential useful samples. Con-

	mAP50 (≈2k)	mAP50 (≈4k)
Central	0.60266	0.66614
CS_None_A	0.61802	0.66706
CS_None_R	0.62120	0.66525
CS_None_AF	0.62554	0.67412
CS_30_A	0.64892	0.66872
CS_30_AF	0.63305	0.66530
CS_30_RF	0.62848	0.66712
CS_60_A	0.62283	0.65739
CS_60_AF	0.63109	0.65077
CS_60_RF	0.62842	0.65728
CS_Rand_A	0.64006	0.66334
CS_Rand_AF	0.62581	0.65877
CS_Rand_RF	0.63709	0.66905

Table 6.11: Comparison of the models after approximately 2000 and 4000 used data

	Person	Bicycle	Car	Motorcycle	Bus	Truck
Central	0.60658	0.47627	0.81154	0.75651	0.33834	0.61145
CS_None_A	0.58136	0.52672	0.77182	0.78002	0.63587	0.55948
CS_None_AF	0.58539	0.60652	0.79857	0.67528	0.47855	0.39697
CS_None_R	0.588594	0.58614	0.79069	0.67220	0.51919	0.49810

Table 6.12: Central, CS_None_A, CS_None_AF, and CS_None_R setup comparison of mAP50 scores at iteration two.

sequently, this approach of allowing the clients to select informative samples without constantly updating the model is applicable to real world scenarios. Additionally, the network utility approaches show that applying this approach to the real world scenario requires more iterations to reach performances comparable to the centralized setup. Overall, the performance of the distributed approach yielded compatible results which was not expected in the first place, due to the circumstances of multiple smaller datasets and the absence of model updates as in typical active learning cycles.

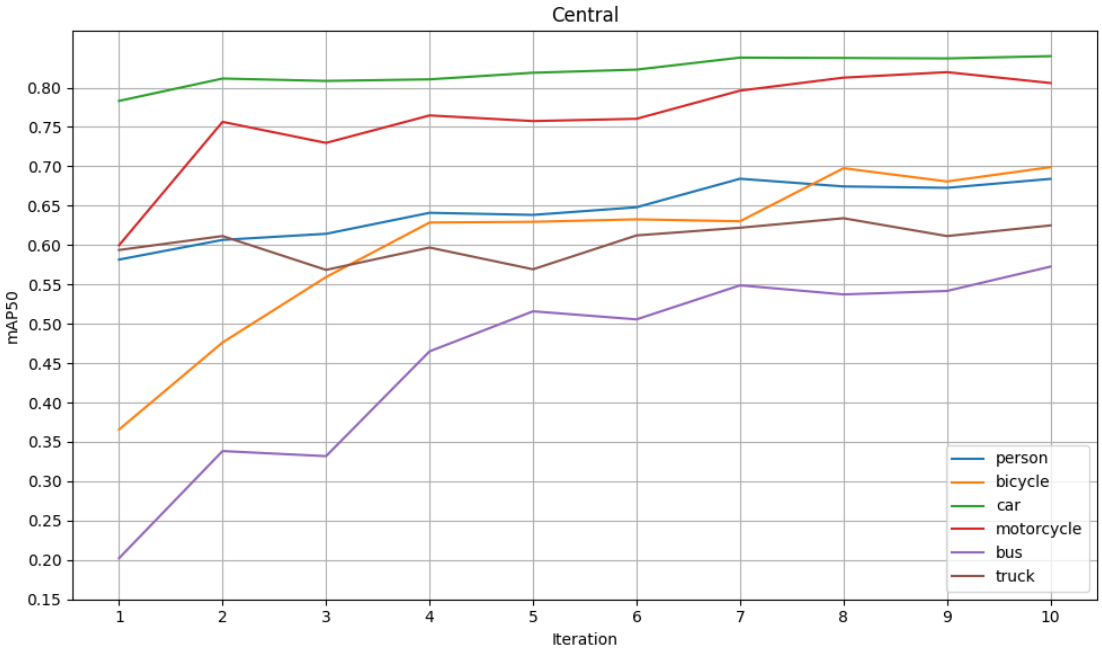

Figure 6.11: Class-wise graph curve of Central.

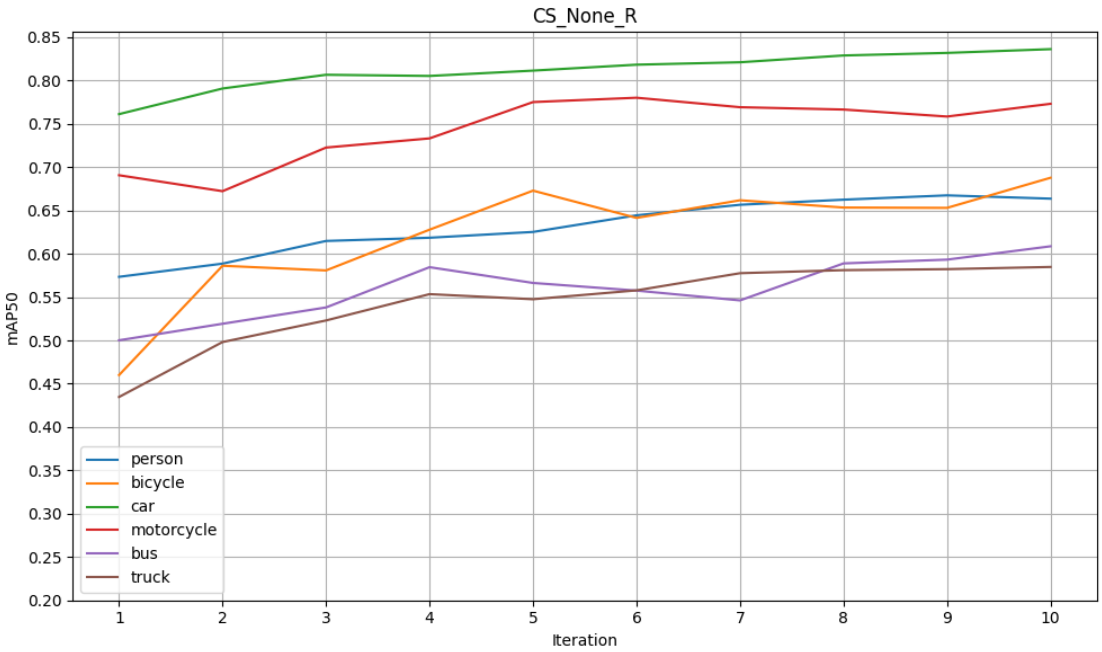

Figure 6.12: Class-wise graph curve of CS_None_R.

7 Conclusions

This chapter presents the final summary of this work and describes the main contributions in Section 7.1. Possible future work within the scope of this thesis is discussed in Section 7.2.

7.1 Summary & Contributions

Currently, vehicles are equipped with a lot of different technologies, such as radars, LiDars, or cameras, aiming toward fully automated driving. Thus, researchers are dealing with object detection to detect and thus perceive the environment to ensure safety, efficiency, and effectiveness [RF18]. Hence, this allows autonomous vehicles to get to know their surroundings better, which is important for path planning. Over the last few years, active learning for object detection has gained more attention because the crucial functionality of active learning is to identify informative samples and thus reduce the amount of data to be labeled [RXC+21]. These ALOD approaches are utilized in a centralized manner, meaning that the dataset is already provided beforehand. After reviewing the existing literature, it became evident that none of the previous studies have addressed the concept of active learning in a distributed manner. More specifically, there is an absence of approaches where vehicles autonomously collect and actively select informative samples, which are provided to a server for model training. This gap in research highlights the need to incorporate clients, such as vehicles, into the active learning cycle.
Thus this thesis proposed an ALOD framework incorporating a server-client similar architecture. Consequently, the steps of a typical active learning cycle are divided into two parts. The central entity or server is responsible for receiving data from the clients. This unlabeled data pool undergoes an annotation process, resulting in a labeled data pool. Then, the labeled dataset is used to train the YOLOv8 model. After each model training iteration, the results are investigated and feedback messages are sent in the form of a balancing distribution, enabling the clients to adjust their selection. The trained models are not updated for the clients to mimic real world scenarios, where it would not be possible to update the model frequently. Consequently, the clients operate with a pre-trained model to detect objects and predict the probabilities on the nuImages dataset, which is equally and randomly partitioned over all clients. Based on that the selection process starts utilizing entropy as the uncertainty estimation while prioritizing images with multiple objects. Additionally, to ensure diverse samples core-set and a class balancing method are implemented. Based on the network utility value, which indicates how much of the selected data the client can send, it transmits the samples to the server.
Evaluation of the proposed framework shows that its performance nearly reaches the scores of the centralized approach with differences of around 1%. Thus, this proves the feasibility of applying this concept to real world scenarios. The evaluation of the setups utilizing the network utility shows compatible results even though fewer data are utilized. This was especially highlighted by comparing the model at stages with a similar amount of used data for the model training. Nevertheless, these results also indicate that applying network consideration means that the model requires more iterations to reach the results of the centralized approach. Considering the performances of the setups utilizing the feedback process, the results gave insights that classes that are underrepresented can improve their performance. But this is connected with the trade-off that other underrepresented classes may experience a negative impact. Unfortunately, in imbalanced datasets, the feedback process based on class-wise performances can not reach its potential because not enough instances of the required classes are available. Furthermore, it was noticed that the performance curves of the central setup compared with distributed setups show differences in performance in the early stages of the active learning iterations, where mostly the central setup had slightly worse performance. The investigation of the class-wise performances in these iterations showed why the distributed setups were slightly superior. The reason is based on the fact that each client in all setups has a different client dataset. Thus the distribution of the images is different.

Consequently, clients can provide useful samples that are more specific compared to samples selected in the first iterations by the server which improves the performance of this certain class. This is because images with many images are prioritized and due to the smaller dataset, clients can select these specific images in an earlier phase if they do not contain many images with many objects. Thus images containing underrepresented classes with only a few overall objects are selected faster, while in the central setup, the images with many objects are prioritized first and those images with fewer objects but are very informative are selected in a later stage.

7.2 Future Work

There are several options for further work of active learning for object detection.
Firstly, integrating a network simulation framework such as OmNet++ could provide valuable insights into the behavior of the proposed framework under realistic network conditions, simulating for instance 5G network traffic. This would enhance the understanding of how network dynamics impact the performance and efficiency of the active learning process.
Another possible area for future work could be enhancing the feedback process by incorporating additional metrics like precision and recall.
Furthermore, a more extended evaluation can be conducted using a higher number of iterations. Additionally, the models could be evaluated by using two phases: in the first phase the models are trained on an imbalanced dataset. Subsequently, in the second phase, clients can collect more instances of the underrepresented classes by providing another dataset. Hence, the behavior of the framework could provide insights into its adaptability and effectiveness in addressing class imbalances.
Furthermore, this approach could be applied to other datasets and detection models, such as two-stage detectors like Faster-RCNN, as well as other YOLOv8 model variants.

Bibliography

[AAAA20] Sharat Agarwal, Himanshu Arora, Saket Anand, and Chetan Arora. Contextual diversity for active learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*, pages 137–153. Springer, 2020.

[AGGWL19] Hamed H Aghdam, Abel Gonzalez-Garcia, Joost van de Weijer, and Antonio M López. Active learning for deep detection neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3672–3680, 2019.

[BKD18] Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. Active learning for deep object detection. *arXiv preprint arXiv:1809.09875*, 2018.

[Bun21] Kraftfahrt Bundesamt. Kba erteilt erste genehmigung zum automatisierten fahren. https://www.kba.de/DE/Presse/Pressemitteilungen/Allgemein/2021/pm49_2021_erste_Genehmigung_automatisiertes_Fahren.html, December 2021.

[CBL+19] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[CEL+21] Jiwoong Choi, Ismail Elezi, Hyuk-Jae Lee, Clement Farabet, and Jose M Alvarez. Active learning for deep object detection via probabilistic modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10264–10273, 2021.

[CHS+17] Shanzhi Chen, Jinling Hu, Yan Shi, Ying Peng, Jiayi Fang, Rui Zhao, and Li Zhao. Vehicle-to-everything (v2x) services supported by lte-based systems and 5g. *IEEE Communications Standards Magazine*, 1(2):70–76, 2017.

[ETS14a] ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.02_60/en_30263702v010302p.pdf, 2014. [accessed: 12.02.2024].

[ETS14b] ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service. https://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.02.02_60/en_30263703v010202p.pdf, 2014. [accessed: 12.02.2024].

[ETS23] ETSI. Intelligent transport system (its); vehicular communications; basic set of applications; collective perception service; release 2. https://www.etsi.org/deliver/etsi_ts/103300_103399/103324/02.01.01_60/ts_103324v020101p.pdf, 2023. [accessed: 12.02.2024].

[EYA+22] Ismail Elezi, Zhiding Yu, Anima Anandkumar, Laura Leal-Taixe, and Jose M Alvarez. Not all labels are equal: Rationalizing the labeling costs for training object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14492–14501, 2022.

[GAA+24] Ross Greer, Bjørk Antoniussen, Mathias V Andersen, Andreas Møgelmose, and Mohan M Trivedi. The why, when, and how to use active learning in large-data-driven 3d object detection for safe autonomous driving: An empirical exploration. *arXiv preprint arXiv:2401.16634*, 2024.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[HBK+23] Aral Hekimoglu, Adrian Brucker, Alper Kagan Kayali, Michael Schmidt, and Alvaro Marcos-Ramiro. Active learning for object detection with non-redundant informative sampling. *arXiv preprint arXiv:2307.08414*, 2023.

[HFC+20] Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecky, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. Scalable active learning for object detection. In *2020 IEEE intelligent vehicles symposium (iv)*, pages 1430–1435. IEEE, 2020.

[HFZ+23] Aral Hekimoglu, Philipp Friedrich, Walter Zimmer, Michael Schmidt, Alvaro Marcos-Ramiro, and Alois Knoll. Multi-task consistency for active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3415–3424, 2023.

[HRF24] Faisal Hawlader, François Robinet, and Raphaël Frank. Leveraging the edge and cloud for v2x-based real-time object detection in autonomous driving. *Computer Communications*, 213:372–381, 2024.

[JCQ23] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolo. `https://github.com/ultralytics/ultralytics`, January 2023.

[KBM23] Mohib Eddine Khebbache, Salim Bitam, and Abdelhamid Mellouk. A cost-effective deep active learning for object detection in automated driving systems. *International Journal of Computing and Digital Systems*, 14(1):1–xx, 2023.

[KKKL21] Seung-Wook Kim, Keunsoo Ko, Haneul Ko, and Victor CM Leung. Edge-network-assisted real-time object detection framework for autonomous driving. *IEEE Network*, 35(1):177–183, 2021.

[KLSL19] Chieh-Chi Kao, Teng-Yok Lee, Pradeep Sen, and Ming-Yu Liu. Localization-aware active learning for object detection. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part VI 14*, pages 506–522. Springer, 2019.

[LOW+20] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128:261–318, 2020.

[LWZ+22] Siyuan Liang, Hao Wu, Li Zhen, Qiaozhi Hua, Sahil Garg, Georges Kaddoum, Mohammad Mehedi Hassan, and Keping Yu. Edge yolo: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):25345–25360, 2022.

[MB23] Mercedes-Benz. Als weltweit erstes automobilunternehmen zertifizierung für sae level 3-system für us-markt. `https://group.mercedes-benz.com/innovation/produktinnovation/autonomes-fahren/drive-pilot-nevada.html`, January 2023.

[oAE21] Society of Automotive Engineers. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. `https://www.sae.org/standards/content/j3016_202104`, April 2021.

[PNDS20] Rafael Padilla, Sergio L Netto, and Eduardo AB Da Silva. A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE, 2020.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[RF18] Qing Rao and Jelena Frtunikj. Deep learning for self-driving cars: Chances and challenges. In *Proceedings of the 1st international workshop on software engineering for AI in autonomous systems*, pages 35–38, 2018.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[RVHR19] Alex Ratner, Paroma Varma, Braden Hancock, and Chris Ré. Weak supervision: A new programming paradigm for machine learning. `https://ai.stanford.edu/blog/weak-supervision/`, March 2019. [accessed: 15.02.2024].

[RXC+21] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

[Set09] Burr Settles. Active learning literature survey. 2009.

[SRTK20] Sebastian Schmidt, Qing Rao, Julian Tatsch, and Alois Knoll. Advanced active learning strategies for object detection. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 871–876. IEEE, 2020.

[SS17] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[Sur20] Anuganti Suresh. What is a confusion matrix? `https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5`, November 2020. [accessed: 16.02.2024].

[Ved23] Henrique Vedoveli. Metrics matter: A deep dive into object detection evaluation. `https://medium.com/@henriquevedoveli/metrics-matter-a-deep-dive-into-object-detection-evaluation-ef01385ec62`, September 2023. [accessed: 16.02.2024].

[WCH22] Jiaxi Wu, Jiaxin Chen, and Di Huang. Entropy-based active learning for object detection with progressive diversity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2022.

[YHC22] Chenhongyi Yang, Lichao Huang, and Elliot J Crowley. Plug and play active learning for object detection. *arXiv preprint arXiv:2211.11612*, 2022.

[YWF+21] Tianning Yuan, Fang Wan, Mengying Fu, Jianzhuang Liu, Songcen Xu, Xiangyang Ji, and Qixiang Ye. Multiple instance active learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5330–5339, 2021.

[YZYC22] Weiping Yu, Sijie Zhu, Taojiannan Yang, and Chen Chen. Consistency-based active learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3951–3960, 2022.