

Marius Maaland
Anders Klever Kirkeby

Decentralized Autonomous Driving

Master's thesis in Computer Science
Supervisor: Pinar Öztürk, Hai Thanh Nguyen
January 2019

Marius Maaland
Anders Klever Kirkeby

Decentralized Autonomous Driving

Master's thesis in Computer Science
Supervisor: Pinar Öztürk, Hai Thanh Nguyen
January 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



NTNU

Abstract

Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Master of Science

by Marius Maaland &
Anders Klever Kirkeby

Autonomous vehicles are improving at a rapid pace, caused by numerous technology companies joining the race. However, current approaches rely on centrally trained models, which have some limitations. Gathering diverse data from areas like urban cities or rural towns, ideally from different countries, is expensive and difficult, even for large tech companies like Google and Tesla. This results in large amounts of training data for self-driving cars being recorded primarily in sunny climates on the United States' west coast, on wide, multi-lane roads, with a specific sign scheme.

Following advances in the field of Internet of Things, an increasing rate of diverse, distributed data is gathered. Paired with the implementation of the European Union's General Data Protection Regulation granting people ownership of their own data, and the right to share it as they like, a decentralized approach opens up for more diverse training data available to improve autonomous vehicles. This thesis proposes a new paradigm for how data is gathered, and how it can be distributed and applied in self-driving cars. It describes a decentralized network where the edges of the network are used to gather data from individual participants, and train the models locally. The models will be biased based on the context they were trained with, like urban or rural areas. Autonomous vehicles on the network can use one of two different techniques proposed in this thesis to combine the models in the network: a context-agnostic approach and a context-sensitive approach. The former, the *Ensemble Detector*, is an ensemble learning method performing weighted majority voting using an ensemble of models. The latter, the *Context-Sensitive Detector*, is a reinforcement learning method that allows for rapid switching between models in a constantly changing environment. The methods have been tested on images from the Berkeley DeepDrive dataset. Experiments show promising results for the Context-Sensitive approach. However, to make the ensemble detector viable for object detection, additional tuning is needed.

NTNU

Abstrakt

Fakultet for informasjonsteknologi og elektroteknikk

Institutt for datateknologi og informatikk

Mastergrad i Teknologi

av Marius Maaland &

Anders Klever Kirkeby

Teknologien rundt selvkjørende biler har sett en oppblomstring de siste årene, som følge av at flere og flere teknologi-giganter tar del i kappløpet. Dagens teknikker benytter seg av sentralt opptrente maskinlæringsmodeller. Innsamling av data fra forskjellige områder som byer og landeveier, ideelt sett også fra forskjellige land, er både vanskelig og dyrt, selv for store selskaper som Google og Tesla. Dette resulterer i at store mengder data tiltenkt selvkjørende biler i hovedsak kommer fra solfylte stater langs USAs vestkyst, med sine brede veier og særegne skilt.

De siste års forbedringer av tingenes internett (IoT), samt EUs innføring av personvernforordningen (GDPR) gjør at privatpersoner nå har eierskap over egen data. Dette muliggjør desentraliserte tilnærminger og danner grunnlaget for mer divers datainnhenting til bruk i forbedringen av selvkjørende biler. Denne avhandlingen foreslår et nytt paradigme for hvordan data samles, og hvordan de kan distribueres og brukes i selvkjørende kjøretøy. Oppgaven beskriver et desentralisert nettverk hvor nodene i nettverket brukes for å samle data fra deltakerene, og modellene trenes lokalt. Modellene vil ha en naturlig bias, avhengig av konteksten de er trent med, for eksempel landsbygda eller byen. De selvkjørende bilene i nettverket kan benytte seg av én av to foreslåtte metoder for å kombinere de forskjellige modellene i nettverket: en kontekst-uvitende, og en kontekst-bevisst metode. Den første metoden, *ensemble* detektoren, er en metode som bruker vektet majoritetsavstemning på en samling modeller for å kombinere deres ekspertise. Den andre metoden er en kontekst-bevisst detektor som bruker *forsterkende læring* for å kjapt kunne bytte modeller, basert på hvilke modeller som gjør det bra i nåværende kontekst. Metodene har blitt testet på bilder fra Berkley DeepDrive datasettet, og viser lovende resultater for bruken av kontekst-bevisste detektorer, men ikke fullt så gode resultater for *ensemble*-metoden.

Acknowledgements

We would like to thank and acknowledge our Master's project advisers Hai Thanh Nguyen and Pinar Öztürk for their help, guidance and wisdom. Their insights and feedback have been invaluable to this project.

We would also like to thank our friends and family for showing incredible support and understanding throughout the research and writing of the thesis.

Contents

Abstract	i
Abstract - Norwegian	ii
Acknowledgements	iii
List of Figures	viii
List of Tables	x
Acronyms	xi
1 Introduction	1
1.1 Motivation	1
1.1.1 Decentralization	2
1.2 Research questions	4
1.2.1 RQ1 - Decentralized Artificial Intelligence	4
1.2.2 RQ2 - Ensemble Detector	5
1.2.3 RQ3 - Context-Sensitive Detector	5
1.2.4 Thesis outline	5
2 Background and related work	7
2.1 Deep Learning	7
2.1.1 Convolutional Neural Networks	7
2.1.2 You Only Look Once	8
2.1.3 Transfer Learning	9
2.2 Decentralized Artificial Intelligence	10
2.2.1 Definition and variations	10
2.2.2 Advantages and disadvantages	11
2.3 Related work	12
2.4 Market research	13
2.4.1 Questions	13
2.4.2 Results	14
3 Data and evaluation	17
3.1 Dataset	17
3.1.1 Berkeley DeepDrive	17
3.1.2 Annotation tooling	18

3.1.3	Dataset statistics and distributions	19
3.1.4	Data allocation with enabled classes	19
3.1.5	Baseline for comparison	21
3.2	Prediction evaluation	21
3.2.1	Prediction definition	21
3.2.2	Intersection over Union	22
3.2.3	Duplicate detection avoidance	24
3.3	Performance evaluation	25
3.3.1	Mean Average Precision	25
3.3.2	F_1 -score	27
4	Decentralized Artificial Intelligence for autonomous driving	28
4.1	Ensemble Learning	29
4.1.1	Weighted Majority Voting and the Weighted Majority Algorithm	29
4.1.2	Ensemble Detector definition	30
4.1.3	Ensemble Detector implementation	31
4.1.4	Ensemble Detector predictions	32
4.2	Online Learning	33
4.2.1	Context-Sensitive Detector	34
4.2.2	Multi-Armed Bandit	34
4.2.3	Upper Confidence Bound	35
4.2.4	Sliding Window Upper Confidence Bound	36
5	Ensemble Detector experiments and results	38
5.1	Main purpose and metrics	38
5.2	Experiment test set	38
5.3	Experiment setup	39
5.4	Experiments	40
5.4.1	Experiment 1: 5-10 models, all classes	40
Setup	40
Hypothesis	40
Results	40
5.4.2	Experiment 2: 5-10 models, low quality training	44
Setup	44
Hypothesis	44
Results	44
5.4.3	Experiment 3: 5-20 models, high quality training	48
Setup	48
Hypothesis	48
Results	48
5.4.4	Experiment 4: 5-50 models, high quality training	52
Setup	52
Hypothesis	52
Results	52
5.5	Wrap-up	55
6	Context-Sensitive Detector experiments and results	56

6.1	Main purpose and metrics	56
6.2	Experiment setup	57
6.3	Experiment plots	57
6.4	Experiments	58
6.4.1	Experiment 1: 2 models, easily discoverable context	58
	Setup	58
	Hypothesis	58
	Results	58
6.4.2	Experiment 2: 8 models, challenging class selection	59
	Setup	59
	Hypothesis	60
	Results	61
6.4.3	Experiment 3: 4 models, additional models available	63
	Setup	63
	Hypothesis	64
	Results	64
6.4.4	Experiment 4: 2 models, changing the window size	65
	Setup	65
	Hypothesis	66
	Results	66
6.4.5	Experiment 5: 2 models, changing the discounting factor	68
	Setup	68
	Hypothesis	68
	Results	68
6.5	Wrap-up	69
7	Discussion and future work	72
7.1	Answers to the research questions	72
7.1.1	RQ1 - Decentralized Artificial Intelligence	72
7.1.2	RQ2 - Ensemble Detector	73
7.1.3	RQ3 - Context-Sensitive Detector	74
7.2	Challenges	75
7.3	Future work	76
7.3.1	Complete network training	77
7.3.2	Reinforcement Learning	77
7.3.3	Distribution selection	77
7.3.4	A combined approach	78
7.3.5	Alternative ensemble learning methods	78
7.4	Article	79
A	Survey	80
A.1	Survey questions	80
A.2	Survey results	81
B	Berkeley DeepDrive	85
B.1	Label format	85

C Article

87

Bibliography

95

List of Figures

2.1	CNN Architecture and Structure	8
2.2	YOLO Architecture	9
2.3	YOLOv2 Improvements	9
2.4	Centralized AI	10
2.5	Decentralized AI	10
2.6	Survey question 2	15
2.7	Survey question 3	15
2.8	Survey question 4	16
3.1	Example labeled image	19
3.2	Preparing a training data distribution containing images with more than 5 cars, less than 3 persons and at least 1 traffic light.	20
3.3	Distribution of classes over the dataset	21
3.4	Predicted bounding box versus ground truth bounding box	22
3.5	IoU mathematical formula	23
3.6	Comparison of poor, good and excellent IoU scores	23
3.7	IoU threshold of 0.5	25
3.8	IoU threshold of 0.3	25
3.9	Precision-Recall example plot	26
4.1	Markov Decision Process for object detection during driving	34
4.2	Upper confidence for 4 machines. Machine 3 will not be selected as one of the next machines	36
5.1	Experiment 1 mAP plot	42
5.2	Experiment 1 ground truth.	42
5.3	Experiment 1 drawn predictions.	43
5.4	Experiment 2 mAP plot	45
5.5	Experiment 2 ground truth.	46
5.6	Experiment 2 drawn predictions.	47
5.7	Experiment 3 mAP plot	50
5.8	Experiment 3 ground truth.	50
5.9	Experiment 3 drawn predictions.	51
5.10	Experiment 4 mAP plot	53
5.11	Experiment 4 ground truth.	53
5.12	Experiment 4 drawn predictions.	54
6.1	Experiment 1 Policy & Reward	60
6.2	Experiment 2 Policy & Reward	62

6.3	2bike1k observation, as selected by the CSD	63
6.4	1motorcycle1k observation, as selected by the CSD	63
6.5	Experiment 3 Policy & Reward	65
6.6	Experiment 4 Policy & Reward - tau 5	67
6.7	Experiment 4 Policy & Reward - tau 200	67
6.8	Experiment 5 Policy & Reward - epsilon 0.05	70
6.9	Experiment 5 Policy & Reward - epsilon 5	70
7.1	Ground truth for an image from the BDD dataset	76
A.1	Question 1.	81
A.2	Question 2.	81
A.3	Question 3.	82
A.4	Question 4.	82
A.5	Question 5.	83
A.6	Question 6.	83
A.7	Question 7.	84
A.8	Question 8.	84

List of Tables

6.1	mAP results for experiment 1	59
6.2	mAP results for experiment 2	61
6.3	mAP results for experiment 3	64
6.4	mAP results for experiment 4	66
6.5	mAP results for experiment 5	69

Acronyms

AI Artificial Intelligence. 1, 4, 10–12, 39, 73

BDD Berkeley DeepDrive. 3, 9, 17, 20, 24, 60, 68, 75, 77

CNN Convolutional Neural Network. 7, 12, 76, 77

CSD Context-Sensitive Detector. ix, 6, 12, 13, 27, 28, 34, 35, 56–66, 68, 69, 71, 75, 77–79

CV Computer Vision. 7, 74, 79

DAI Distributed Artificial Intelligence. 10

DL Deep Learning. 1–3, 5, 7, 9, 10, 19, 38, 71–74

DRL Deep Reinforcement Learning. 77

DzAD Decentralized Autonomous Driving. 2, 3, 14, 19, 21, 28, 48, 52, 72–74

DzAI Decentralized Artificial Intelligence. 1, 5, 6, 10–13, 20, 28, 72, 73

ED Ensemble Detector. 6, 13, 28, 29, 31, 38–41, 44, 45, 48, 49, 52, 69, 73, 74, 78

EU European Union. 1

FP False Positive. 26, 31, 49, 55

FPS Frames Per Second. 8, 75

GDPR General Data Protection Regulation. 1

GPS Global Positioning System. 2, 18

-
- GPU** Graphics Processing Unit. 8, 75
- IMU** Inertial Measurement Unit. 2, 18
- IoT** Internet of Things. 1
- IoU** Intersection over Union. 22–24, 26, 32, 49
- JSON** JavaScript Object Notation. 18
- MAB** Multi-Armed Bandit. 33, 35
- mAP** mean Average Precision. 26, 27, 39–41, 45, 48, 49, 52, 55, 59, 64, 66, 69, 73, 76, 77
- MDP** Markov Decision Process. 33, 34
- ML** Machine Learning. 1–3, 7, 9, 12, 13, 27, 29, 33, 75
- NLP** Natural Language Processing. 7
- NMS** Non-maxmimum suppression. 24
- NN** Neural Network. 7, 33
- NTNU** Norwegian University of Science and Technology. 13
- RL** Reinforcement Learning. 33, 34, 69
- RQ** Research Question. 4, 5, 38, 56, 72
- SW-UCB** Sliding-Window UCB. 37, 56, 57, 65
- TP** True Positive. 26, 31, 49, 55
- UCB** Upper Confidence Bound. 35, 36, 64
- US** United States. 17, 74
- USA** United States of America. 2
- WMA** Weighted Majority Algorithm. 29, 30, 78

WMV Weighted Majority Voting. [29–32](#), [49](#), [55](#), [73](#), [78](#)

YOLO You Only Look Once. [8](#), [9](#), [61](#), [75](#)

YOLOv2 You Only Look Once version 2. [8](#), [31](#), [41](#)

Chapter 1

Introduction

1.1 Motivation

Data availability is one of the driving factors of [Artificial Intelligence \(AI\)](#) research. Progress and breakthroughs in the field of [Machine Learning \(ML\)](#), especially within [Deep Learning \(DL\)](#), are largely dependent on the availability of *quality* data. The commercial field is dominated by a few large companies resourceful enough to gather the necessary data [1]. With the increasing popularity of [Internet of Things \(IoT\)](#)-devices [2], paired with the introduction of the [General Data Protection Regulation \(GDPR\)](#) [3] in the [European Union \(EU\)](#), individuals are gaining more access to, and power over, their own data. Emphasizing individual users' ownership and *right* to their own data can help form a new paradigm for the [AI](#) field. This proposed paradigm is described as [Decentralized Artificial Intelligence \(DzAI\)](#) in this thesis, and describes how data is collected, how models are trained on this data, and how they are used in a [DzAI](#) network.

One of the areas of [AI](#) that might benefit from such a decentralization, is the field of self-driving vehicles. Vehicles interacting autonomously are claimed to be more efficient and less prone to unnecessary stops and delays, which will ultimately result in less traffic congestion [4] - a growing problem in metropolitan cities worldwide [5]. Most importantly, the introduction of self-driving cars is anticipated to drastically reduce the number of accidents caused by motor vehicles [6][7]. Numbers from the Autonomous Vehicle Disengagement Reports submitted to the State of California's Department of

Motor Vehicles show a decreasing number of disengagements - times humans had to take control over the vehicle - year over year: Google's subsidiary company *Waymo* decreased their yearly number of disengagements from 272 in 2015, to 124 in 2016 and 63 in 2017 [8][9][10]. Even though this signals that self-driving cars are improving, most of the data they are trained on comes from the western states of the [United States of America \(USA\)](#) [11]. Furthermore, large portions of this data have been recorded on spacious highways with multiple lanes and uniform sign schemes, not the most diverse driving conditions.

Diversifying the data gathering for, training, and testing of, self-driving cars is difficult for several reasons. It is natural for the companies working on autonomous driving technology to conduct training and testing close to their headquarters. Deploying autonomous vehicles with safety drivers to more diverse locations is a resource intensive task, but it is also an important one. In any [Machine Learning \(ML\)](#) task, both the *quantity* and the *quality* of the data matters. Google's *Waymo* has been notable at gathering huge amounts of sensor data with its self-driving cars, having driven more than 9 million miles since the fleet started operating in 2009 [11]. Nevertheless, more data is still required for the continued improvement of autonomous vehicles. Diverse data from cities in different countries, of different sizes, with different sign schemes etc. is especially important. The hypothesis of this thesis is that this can be achieved with a [Decentralized Autonomous Driving \(DzAD\)](#) network where participants gather driving data using their personal vehicle during their daily commute. The data can be of many shapes and forms, depending on the equipment available, such as [Global Positioning System \(GPS\)](#) data, [Inertial Measurement Unit \(IMU\)](#) data or imagery (photos or videos). The latter will be used as an example in this thesis, focusing on the task of performing object detection on images containing cars, pedestrians, traffic lights and other objects relevant for the training of self-driving cars.

1.1.1 Decentralization

A [DzAD](#) network will rely on numerous participants to gather images during driving. This data can be used to train [Deep Learning \(DL\)](#) object detection models locally on the participants' devices. The models in the network will be biased based on the context they were trained with. A model trained on data primarily from urban areas could have

a stronger bias towards *pedestrians*, while another model trained in a rural context could have a stronger bias towards *trucks*. These object detectors can then be shared on the DzAD network with other parties in need of training data, making data available that would not otherwise be shared or even gathered in the first place. However, since the raw data is not shared directly, but *models* that are *trained* on the data are shared instead, techniques for combining these models and utilizing their biases need to be developed.

In general [Machine Learning \(ML\)](#) sees a high correlation between increasing the amount of data, and achieving higher accuracy, especially within the sub-field of [DL](#). The models making up the DzAD network are unfortunately trained on relatively small datasets. Even though the participants in the network can easily gather large amounts of data by simply driving around with a camera and sensors, gathering the data is only half of the equation. To be useful for supervised [DL](#) the data also needs to be labeled and annotated. Class labels and bounding boxes are needed for every detectable object, at a minimum. Metadata like weather, scene and time of day can also be added. This labeling and annotation task demands tedious and meticulous work, and participants cannot be expected to label and annotate thousands of images, even with the help of tools. A survey prepared by the authors, exploring people’s willingness to perform this task, is described in [section 2.4](#), and shows that most participants require a high compensation to label images. Thus, for a DzAD network to be useful, it needs to be able to combine and utilize multiple [DL](#) models trained on small datasets. Autonomous vehicles on the network can use one of two different techniques proposed in this thesis that combine the models in the network: a context-agnostic approach, and a context-aware approach. The former, the *Ensemble Detector* described in [section 4.1.2](#), is an ensemble learning approach performing weighted majority voting using a collection of models. The latter, the *Context-Sensitive Detector* described in [section 4.2.1](#), is a reinforcement learning approach that allows for rapid switching between models in a constantly changing environment. The methods have been tested on images from the [Berkeley DeepDrive \(BDD\)](#) dataset described in [section 3.1](#), with experiments and their results shown in [chapters 5 and 6](#).

The proposed paradigm brings many hypothetical advantages and disadvantages, a selection of which will be explored in this thesis. They are described in more detail in [section 2.2.2](#), and are summarized below.

Advantages that will be explored in this thesis:

- Improved preservation of privacy. Not transmitting raw data greatly reduces the risk of exposing sensitive information, which in turn will increase the likelihood of data owners being willing to share their models.
- Greater data diversity. With improved preservation of privacy leading to more sharing of data, this data will likely come from more diverse locations and environments as well, increasing both the quantity and the quality.

Disadvantages that will be explored in this thesis:

- Lower accuracy. Object detectors trained on small datasets generally perform worse than detectors trained on larger datasets.
- Labeling. If the data for the locally trained models is to be sourced from individual users, the task of annotating and labeling the images for training needs to be simple.

1.2 Research questions

1.2.1 RQ1 - Decentralized Artificial Intelligence

What advantages and disadvantages does Decentralized Artificial Intelligence (DzAI) bring to autonomous driving?

A centralized [AI](#) model trained on a large dataset will in most cases perform better than a decentralized network consisting of weaker models trained on smaller datasets, in terms of raw accuracy. However, the decentralized approach proposed in this thesis introduces potential advantages in other areas such as privacy, and data diversity, that may compensate for the loss of accuracy. For autonomous vehicles, a high accuracy in object detection is of utmost importance, as it is integral to the safety of the vehicles. It is therefore crucial to reduce the loss of accuracy using techniques like Ensemble Learning or Online Learning, discussed in [RQ2](#) and [RQ3](#).

1.2.2 RQ2 - Ensemble Detector

Can a collection of weak [Deep Learning \(DL\)](#) object detectors be combined to perform as well as, or better than, a strong [DL](#) object detector?

In the [DzAI](#) network proposed in [12], ensemble learning showed promising results in the task of classification. Multi-class object detection is a more challenging task, and combining the predictions of multiple models is less trivial than classification. Section [4.1](#) describes a way to use Ensemble Learning to combine multiple object detectors trained on small datasets to increase their collective accuracy. Experiments 1 through 4 in chapter [5](#) look at the predictive performance of a [DzAI](#) network using this technique, and the summarized, concluding answer to [RQ2](#) can be found in section [7.1.2](#).

1.2.3 RQ3 - Context-Sensitive Detector

Can detection and exploitation of bias in weak [Deep Learning \(DL\)](#) object detectors, in a context-dependent environment, outperform strong context-agnostic [DL](#) object detectors?

Rather than combining models *ahead of time*, section [4.2](#) describes a technique that looks at the current context, like driving in a city or out on the countryside, in order to retrieve predictions from models specialized in that context. Experiments 1 through 5 in chapter [6](#) look at the feasibility of a [DzAI](#) network utilizing such an approach. A summary of this, along with a concluding answer to [RQ3](#) can be found in section [7.1.3](#).

1.2.4 Thesis outline

Chapter [2](#) introduces background theory on the choice of [Deep Learning \(DL\)](#) architecture, and defines [Decentralized Artificial Intelligence \(DzAI\)](#). The chapter also presents market research, and discusses related work.

Chapter [3](#) presents the dataset, and how it is used in the thesis. It explains the format of predictions made on the dataset, and introduces metrics and concepts that are crucial to understanding the experiments.

Chapter 4 provides detailed explanations of the 2 proposed DzAI methods: the [Ensemble Detector \(ED\)](#) and the [Context-Sensitive Detector \(CSD\)](#). Next, chapters 5 and 6 cover the experiments performed to evaluate the [ED](#) and the [CSD](#), respectively.

Finally chapter 7 summarizes the results of the experiments, and answers the research questions posed in the introduction chapter. The chapter also discusses challenges encountered in the thesis, and presents future work.

Chapter 2

Background and related work

2.1 Deep Learning

Deep Learning (DL) and traditional Machine Learning (ML) both offer ways to train models to make predictions on data. Their difference lies in the number of *layers* comprising their Neural Network (NN) structure. The increased layer count allows for more complex abstractions and composite representations than possible in *shallower* networks.

DL is heavily used in areas such as Computer Vision (CV) and Natural Language Processing (NLP), and made feasible by recent increases in computing power, and availability of labeled data [13]. During the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2015, multiple DL methods surpassed the human recognition rate of 95% [14].

2.1.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a class of deep neural networks, commonly used for visual data analysis [15]. CNNs are trained end-to-end, meaning the task of learning features, detecting objects and predicting labels are considered *one big task*. As such, the CNN consists of one input layer and one output layer. In between is a number of *hidden* layers that typically consist of convolution layers, activation layers, and polling layers, as depicted in figure 2.1 [16].

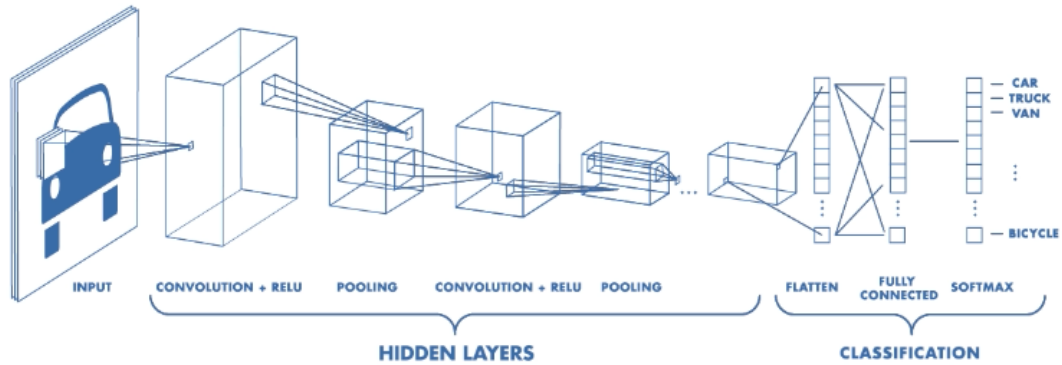


FIGURE 2.1: CNN Architecture and Structure

The classification portion of the network often consists of a couple of fully connected layers, where every neuron in one layer is connected to every neuron of the next layer. Finally a *softmax* layer is used to output the labels or a classification for a given input.

2.1.2 You Only Look Once

The object detection system used in the experiments, found in chapter 5 and 6, is an adaptation of the [You Only Look Once version 2 \(YOLOv2\)](#) architecture [17] written in TensorFlow [18] and Keras [19]. The main reason for selecting [YOLO](#) as the foundation architecture for this thesis is its speed. Its ability to perform real-time predictions at 30 [Frames Per Second \(FPS\)](#) (running on a Pascal Titan X [GPU](#)) [17] makes it a realistic architecture for time-sensitive tasks, such as object detection whilst driving.

The architecture for the original [YOLO](#) network can be seen in figure 2.2 [17]. A natural effect of reducing the spatial dimensions gradually is that it makes it harder to predict smaller objects. Because of this, [YOLOv2](#) makes some important changes in the two final layers: Using a pass-through method, depicted in figure 2.3 [17], it uses information from earlier convolutions in order to retain more information and increase its ability to perform predictions on smaller objects. Even with these improvements it is important to state that the [YOLO](#) network still suffers a significant performance loss on smaller objects, which will be evident in the experiment chapters.

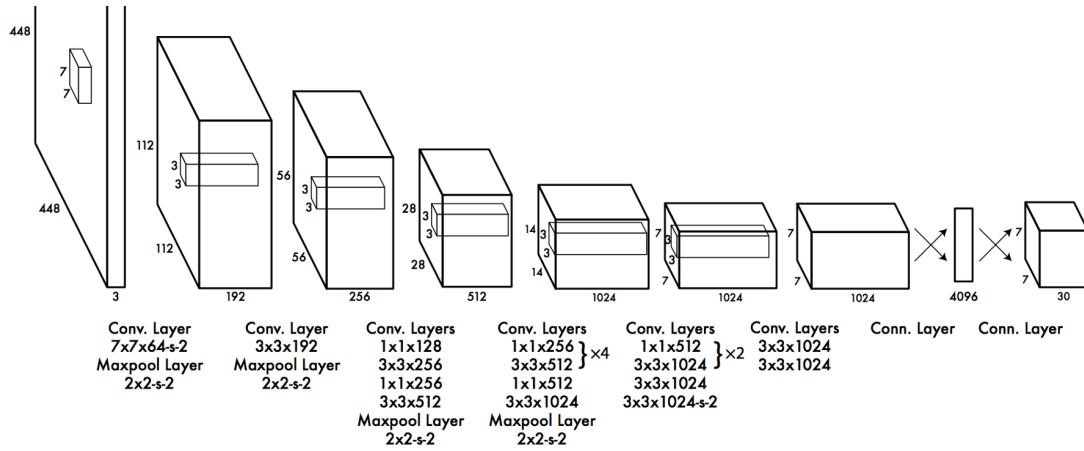


FIGURE 2.2: YOLO Architecture

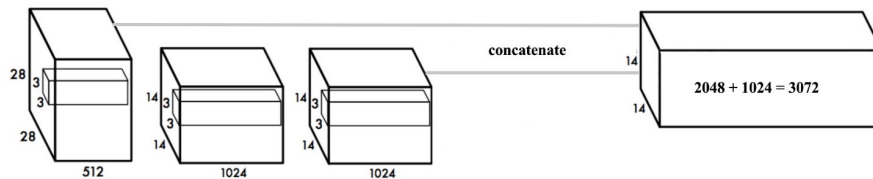


FIGURE 2.3: YOLOv2 Improvements

2.1.3 Transfer Learning

Transfer learning is an [ML](#) technique where a model trained for one task is adapted and used for a different task [20], allowing for rapid prototyping. It is not uncommon for [DL](#) models to take days or weeks to train from scratch, even on modern hardware. To save time, freezing all but the last few layers of a pre-trained model, putting the model in a *headless* state, makes it possible to only train the last few layers for a specific task. This method is commonly used in relation to image-based tasks due to their large feature spaces, where transfer learning is performed on pre-trained [Deep Learning \(DL\)](#) models that have already been trained on large datasets, like the ImageNet dataset [21]. Transfer learning is used in the thesis to drastically reduce the training time of the [YOLO](#) models used in the the experiments. The models use the weights provided by the creators of the [YOLO](#) network [22], pre-trained on the *COCO* [23] and *ImageNet* [21] datasets. By freezing all the network layers except for the last 2, models can quickly be trained on the [Berkeley DeepDrive \(BDD\)](#) dataset and learn the classes specific to this dataset.

2.2 Decentralized Artificial Intelligence

This thesis introduces a new paradigm for how data is gathered, how [Deep Learning \(DL\)](#) models are trained on this data, and how they can be used in a decentralized network. Before delving into implementations and experiments, it is important to explain how such a network would work, its advantages, disadvantages, and variations.

2.2.1 Definition and variations

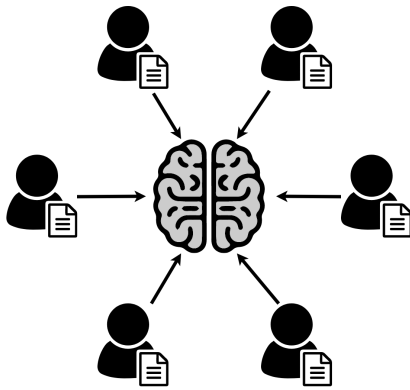


FIGURE 2.4: Centralized AI

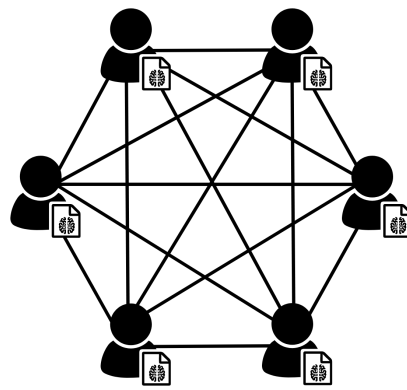


FIGURE 2.5: Decentralized AI

Centralized [Artificial Intelligence \(AI\)](#), illustrated by figure 2.4 [12], works by first aggregating large amounts of data at a single location, before training one or multiple models, with the goal of solving complex learning, planning and decision making problems. In contrast, [Decentralized Artificial Intelligence \(DzAI\)](#), illustrated by figure 2.5, does not require all of the training data to be aggregated at a single location. [DzAI](#) has several different definitions, names and variations, and there are slight but important differences between [Decentralized Artificial Intelligence \(DzAI\)](#) and [Distributed Artificial Intelligence \(DAI\)](#). They are both sub-fields of [AI](#) separate from the centralized approach, but differing in the way the participants work together in the network.

In a [DzAI](#) network every participant is autonomous and exists independently. The participants make up a network and cooperate with each other to perform a global or individual task [24]. This is the [AI](#) method that will be the focus of this thesis.

[DAI](#) however, involves a distributed group of participants working towards a collaborative solution to global problems. It is collaborative in the way that sharing of information between the participants is crucial for the group to produce a solution as a whole.

An example illustrates this slight but important difference more effectively. Consider a network whose goal is to be able to detect cars and pedestrians in an image. A *distributed* network would have two participants A and B, where A knows how to detect cars *only*, and B knows how to detect pedestrians *only*. Collaboration and sharing of information is critical for the network to accomplish its goal. If A is shown an image of a pedestrian, it would not be able to accomplish the goal individually. In a *decentralized* network on the other hand, both agents A and B would know how to detect both cars and pedestrians, and can thus operate independently, but by collaborating they can improve their collective accuracy.

2.2.2 Advantages and disadvantages

Like any other AI method, DzAI has both advantages and disadvantages. A large part of this thesis is concerned with examining and assessing whether the advantages of a decentralized network outweigh the disadvantages. All possible advantages and disadvantages are too numerous to feasibly explore in a single thesis. Based on importance and relevance, a few have been chosen, described below.

Possible advantages

- Improved preservation of privacy. Transmitting trained models encrypted and anonymously, rather than raw data, could reduce the ramifications of theft and unauthorized eavesdropping. Not transmitting raw data greatly reduces the risk of exposing sensitive information, which in turn could increase participants' willingness to share their models.
- Greater data diversity. Improved preservation of privacy may lead to increased sharing of data from diverse locations and environments, raising both the quantity and the quality.

Possible disadvantages

- Lower accuracy. Depending on the method of combination, the accuracy of the combined models might be worse than that of a single model trained on a larger dataset. A major part of this thesis is concerned with exploring different ways of

combining models and experimenting with how to choose the most optimal model in any given situation.

- Labeling. If data for the locally trained models is to be sourced from participants, someone needs to do the labeling. The process of labeling must be simple, efficient and user-friendly, and the individual models should require small amounts of data.

2.3 Related work

Existing approaches to [Decentralized Artificial Intelligence \(DzAI\)](#) tackle different aspects of the decentralization process. Projects like Ocean [25] and Datum [26][27] are trying to create a marketplace for sharing raw data in a decentralized fashion. SingularityNET [28], on the other hand, is a decentralized [AI](#) network, currently under development, that aims to let anyone create, share and monetize [AI](#) services.

A different approach, proposed by Google's AI division, is a learning method called *federated learning*, able to perform collaborative [Machine Learning \(ML\)](#) without centralized data [29]. Google uses federated learning to enable smart phones to keep all their training data on the device, and simultaneously share the prediction models to collaboratively learn and improve the model. After downloading the latest model, the devices on the network learn using their own local data and improve the model, before sharing the summarized changes with the rest of the network. This type of learning allows for smarter models, lower latency and less power consumption, while ensuring privacy, according to Google [30]. In comparison, the [CSD](#) approach proposed in this thesis gathers models trained on the edges, and uses context to switch between them in a rapidly changing environment.

The open-source OpenMined project [31] wants to decentralize [AI](#) by bringing models to each participant in a network, allowing them to train the models locally, using *federated learning*, while keeping the data on the device. Homomorphic encryption [32] of the models allows users of OpenMined's network to share their models with no risk of theft.

Recently, a centralized end-to-end approach for autonomous driving, proposed by Borsari *et al* of Nvidia [33], has received a lot of attention. They present a [CNN](#) consisting of 9 layers, supported by Nvidia's Drive Px 2 platform [34]. The system uses 3 front

facing cameras as input signals, and learns how to steer a vehicle by observing a human pilot. One of the benefits of this approach is that it optimizes all of processing steps simultaneously, where most other approaches aggregate methods, like lane marking detection, path planning, and control, causing slower computations. An approach like this requires a continuous stream of data, along with recorded steering wheel angles.

This thesis presents a new paradigm where privacy is an inherent attribute of the collaboration process. Similar to OpenMined, this thesis describes an approach where training is performed locally - on the edges of the network - without the data ever leaving the source. Rather than federating existing models, the proposed approach creates individual models at the data sources and combines them using an [Ensemble Detector \(ED\)](#) or a [Context-Sensitive Detector \(CSD\)](#), explained in detail in section 4.1.2 and section 4.2.1, respectively.

2.4 Market research

To get an idea to which extent people are interested in participating in a [Decentralized Artificial Intelligence \(DzAI\)](#) network to improve self-driving cars, a short survey was created. The purpose was to gauge people's interest in participating in such a network, and to determine the compensation they would expect for the service. Both a Norwegian and an English version of the survey were created. The Norwegian version was advertised using posters on the [Norwegian University of Science and Technology \(NTNU\)](#)'s Gløshaugen campus in Trondheim, Norway. It was also shared with the friends and family of the authors. The English version was posted on the social news aggregation and discussion website *Reddit* [35], on sub-forums related to [Machine Learning \(ML\)](#) and data gathering. In other words, the majority of the participants were either active in technical web forums, or attend a technical university, indicating a certain degree of technical interest. The two surveys gathered a total of 791 responses, 371 of which coming from the Norwegian version, and 420 from the English version.

2.4.1 Questions

The questions asked in the English version of the survey are listed in appendix A.1. The responses to the Norwegian version were translated and counted together with their

English counter-parts. The questions were designed to gain an understanding of people’s willingness to participate in a [Decentralized Autonomous Driving \(DzAD\)](#) network. In summary, the survey inquired about what kind of compensation people would require to capture images and then annotate and label them, for the purpose of improving self-driving cars. People were also asked whether they would be worried about their privacy, even if promised that the data shared would be anonymous.

2.4.2 Results

The results from the survey can be seen in appendix [A.2](#). The ages of the participants skewed towards the younger end, with 75% being between 18 and 30 years old. Close to all of the participants (98%) were already familiar with the concept of self-driving cars, adding more legitimacy to the subsequent answers. 88% of participants were willing to collect and share images from their personal driving, as seen in figure [2.6](#), but the compensation they required to do so varied significantly: 123 people said they would collect raw images (without labeling them) for less than \$1 per hour of driving. Approximately the same amount of people would not do the same task for less than \$7 per hour (fig. [2.7](#)). The majority of the respondents were in the lower half of the compensation range, i.e. \$5 per hour or less, a reasonable compensation for such a service.

When it came to labeling and annotating the images however, the respondents wanted much higher compensation, seen in figure [2.8](#). The most expensive option of receiving \$0.5 or more per picture annotated, corresponding to approximately \$20 per hour of work, was preferred by the most participants (268). Compensation of this size is not likely to be feasible for gathering thousands of annotated images.

Lastly, 60% of the participants were worried that the data could be linked to them personally, even if the data was claimed to be anonymous. However, this question was only presented to people who first indicated that they were willing to share their data. In other words, people being worried about their privacy does not mean that they are not willing to participate.

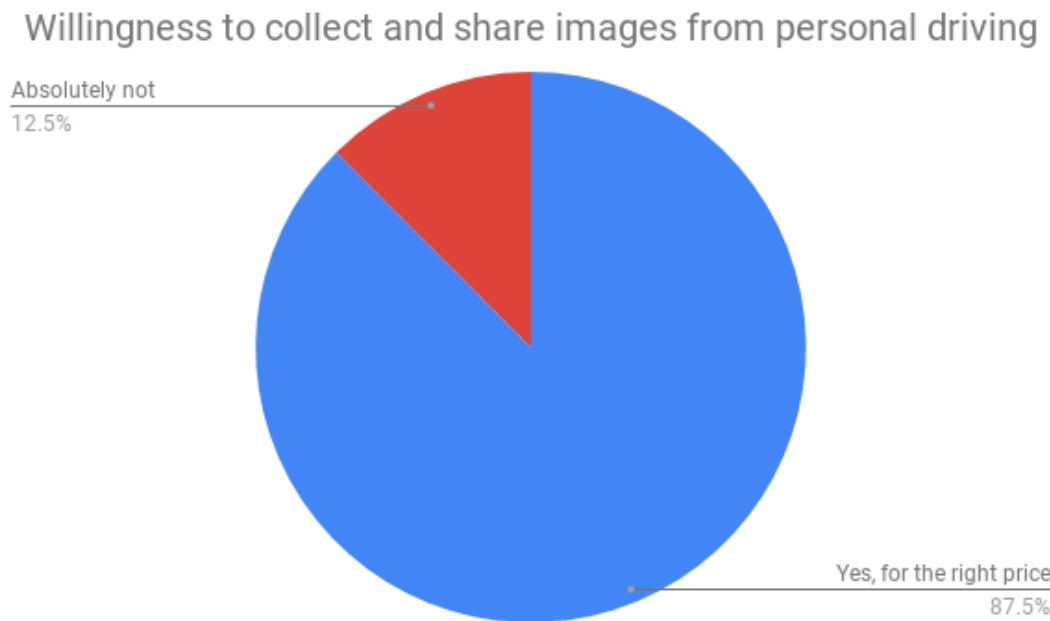


FIGURE 2.6: Survey question 2

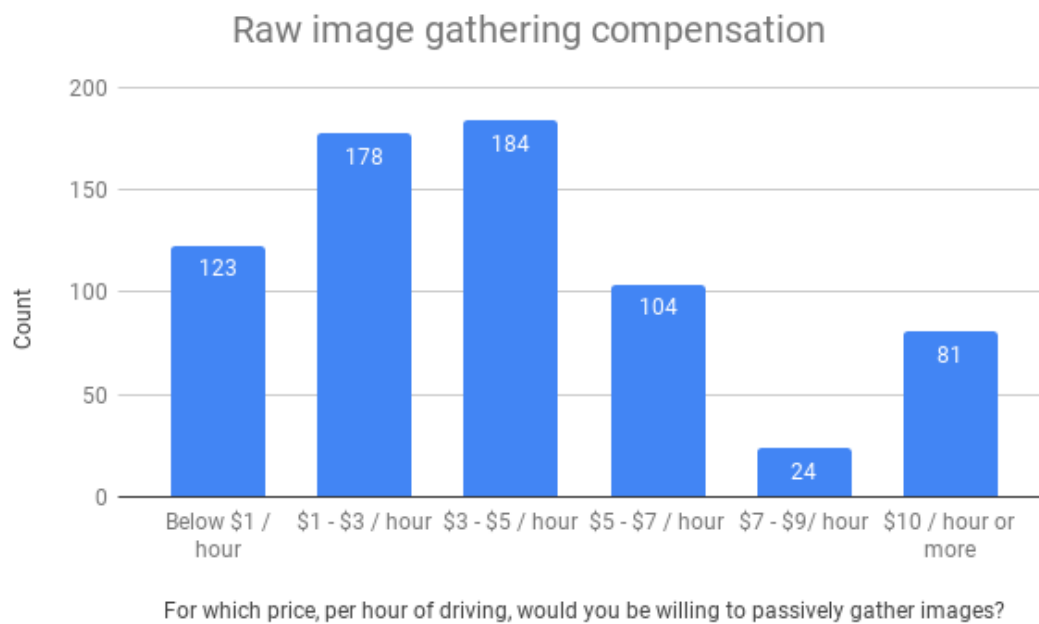


FIGURE 2.7: Survey question 3

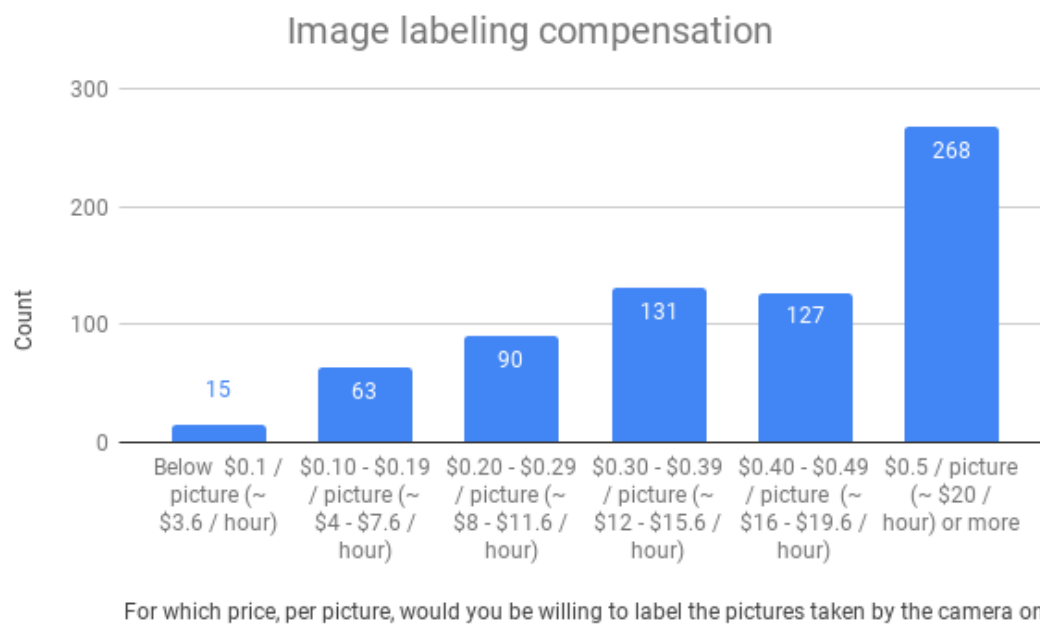


FIGURE 2.8: Survey question 4

Chapter 3

Data and evaluation

3.1 Dataset

The dataset used in the experiments in this paper is the *BDD100K dataset* provided by the [Berkeley DeepDrive \(BDD\)](#) project [36]. Facilitated by the researchers' own annotation tooling [37], the dataset consists of 100 000 videos with diverse annotations including image level tagging, object bounding boxes, driveable areas, lane markings, and weather data. The creators claim the dataset to be "the largest available dataset of annotated driving scenes" [36]. Compared to existing datasets used for driving image recognition benchmarks, it allegedly covers more realistic driving scenarios and captures more variation in appearance and pose configuration of categories of interest. The dataset is therefore considered more challenging than other sets.

3.1.1 Berkeley DeepDrive

The foundation of the dataset is a collection of 100 000 video clips captured by the front facing camera of cars driving around the [US](#) in diverse conditions. Extracted from the video clips are:

- 100 000 labeled key frame images, extracted from the videos at the 10th second.
- 10 000 key frames extracted for full-frame semantic segmentation. Semantic segmentation is the process of making dense predictions inferring labels for every

pixel, so that each pixel is labeled with the class of its enclosing region or object, e.g. every pixel that make up the object *car*, or the *sky* region.

- Full-frame semantic segmentation maps corresponding to the 10 000 images above.
- Segmentation maps of driveable areas in the images.
- Annotations of road objects, lanes and driveable areas in a [JavaScript Object Notation \(JSON\)](#) format.
- [Global Positioning System \(GPS\)](#) and [Inertial Measurement Unit \(IMU\)](#) information recorded along with the videos.

This thesis will use the 100 000 labeled key frame images extracted from the videos and their corresponding labeled annotations, with the fields shown in appendix B.1. Not all fields are available for every image, and not all fields are used in the experiments performed in this thesis. The most important pieces of information are the *category* and *box2d* fields, describing the objects in the images, and their bounding boxes, respectively. The *category* field describes the different *classes* of the objects in the images, and can take any of the following values: *Bus*, *traffic light*, *traffic sign*, *person*, *bike*, *truck*, *motor*, *car*, *train*, *rider*.

Figure 3.1 shows an example of a labeled image, with ground truth bounding boxes drawn around two cars, a traffic light and two signs. It also shows lane markings, drivable area and metadata like the weather, but only the object bounding boxes and classes are of interest in the thesis.

3.1.2 Annotation tooling

The team behind the *BDD100K* dataset has also created a versatile and scalable annotation tooling system [37], for creating annotations needed in a driving database, like bounding boxes and lane detection. The tool is designed to be user-friendly and efficient, and can be used by researchers and other people in need of labeled video frames or images of driving data.



FIGURE 3.1: Example labeled image

3.1.3 Dataset statistics and distributions

A dataset of this size allows for interesting partitioning based on class distributions, enabling experiments with pre-determined biases towards certain classes. The authors of this thesis have developed tools to specify and extract such distributions.

A *distribution* describes the count of each class in a training set, and can be used to create training sets with a specific bias. For example, a distribution can be specified to contain only images with at least 1 person in them. They can also be as specific as: "All images with more than 7 cars, less than 3 persons and no traffic lights". The purpose of this is to create biased training sets making models trained on these images better at detecting certain classes than others. The goal is to simulate a real-world scenario of combining models generated by participants driving in certain locations, like rural or urban areas.

3.1.4 Data allocation with enabled classes

Each of the [Deep Learning \(DL\)](#) models in the two different [Decentralized Autonomous Driving \(DzAD\)](#) methods is allocated a training set of a certain size, with a specific *distribution*, as described in section [3.1.3](#). Then, two unique random sample sets are

drawn from the subset, and used as training (90%) and validation (10%) data, illustrated in figure 3.2.

To better emphasize the effects of **Decentralized Artificial Intelligence (DzAI)**, *noise* is removed from the experiments by using models with different enabled classes. An *enabled class* is a class that a model has been trained to predict, by including the class' label in the training set. Different experiments may have different enabled classes, which will be indicated in the experiment setup description. For instance, one experiment is performed on models with all 10 classes enabled, while others are performed on models with only specific classes enabled, like *car*, *truck* and *person*.

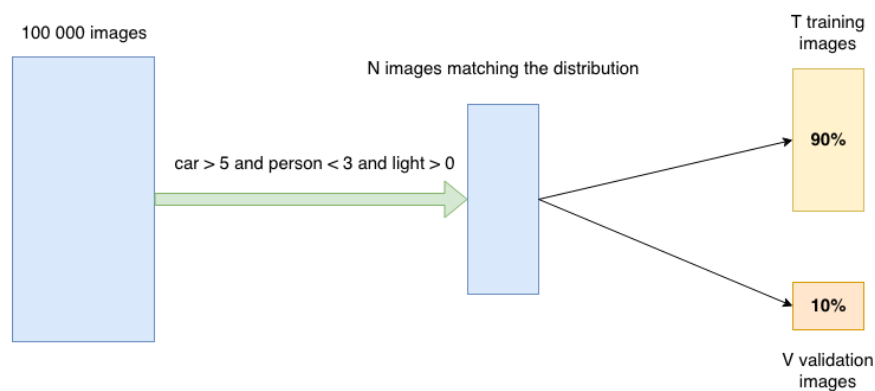


FIGURE 3.2: Preparing a training data distribution containing images with more than 5 cars, less than 3 persons and at least 1 traffic light.

The reason for removing noise from the experiments is that the images in the dataset contain high counts (18.4 on average) of different objects at varying distances and sizes. Compared to the *COCO* dataset, often used as a object recognition benchmark, images in the *BDD* dataset contain, on average, 3 times the number of objects per image [23]. Shown in figure 3.3 are the number of objects of each type. In addition, roughly 50% of the objects in the dataset are occluded (overlapping) and 45% are truncated (cut off by the image frame). There is also a large disparity between the counts of different classes: there are over 1 million instances of *car* objects in the entire dataset, but only 179 instances of trains [36]. Combined these factors make the task at hand, to detect and locate *all* objects in a given image, a challenging one.

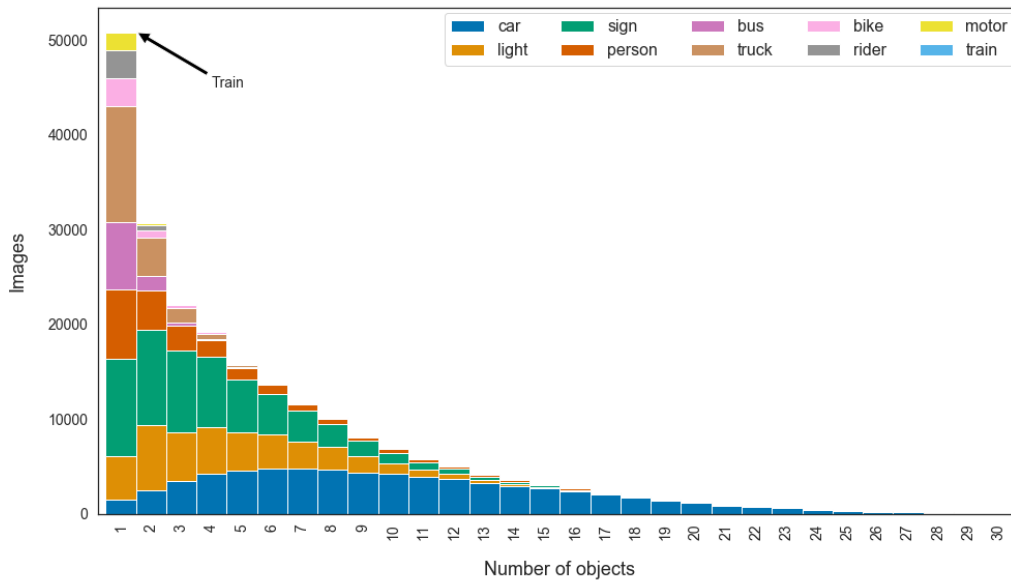


FIGURE 3.3: Distribution of classes over the dataset

3.1.5 Baseline for comparison

Serving as a baseline model for comparison in the experiments in chapters 5 and 6, is a *strong* model trained on 20 000 images, for 15 epochs. The 20 000 images are drawn randomly from the same distribution as the weak models, with the same enabled classes. The purpose of the baseline model is to simulate the centralized approach, trained on 20 000 images at a central location.

3.2 Prediction evaluation

3.2.1 Prediction definition

The object detectors trained in the [Decentralized Autonomous Driving \(DzAD\)](#) network will make *object detections* or *predictions* given an input image. These terms are used interchangeably throughout the thesis, and refer to the output from the models. Predictions have 3 components:

- The classes of the objects detected.
- Bounding boxes bounding the objects detected.

- A confidence score for each object, indicating how certain the model is of the detected object's class and bounding box.

3.2.2 Intersection over Union

To evaluate the accuracy of the predicted objects, doing an exact match comparison between the ground truth bounding boxes and the predicted bounding boxes is not a good approach. Such a comparison would indicate a failed match if a single pixel has been predicted incorrectly, which does not mean that the predicted bounding box does not accurately bound the detected object. An exact match is unlikely. Figure 3.4 [38] illustrates the comparison of a ground truth bounding box on a *stop* sign, with a hypothetical predicted bounding box. Even though the predicted bounding box does not exactly match the ground truth bounding box, it does arguably correctly communicate the location and dimension of the object, and should not be considered an incorrect prediction.

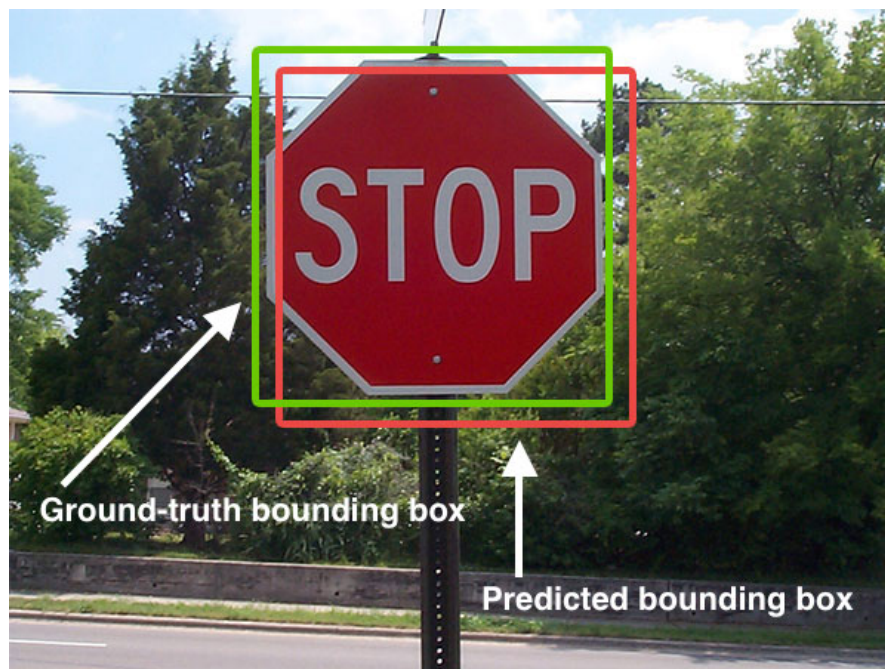


FIGURE 3.4: Predicted bounding box versus ground truth bounding box

A popular (and simple) method used to evaluate predicted bounding boxes is to use a technique called **Intersection over Union (IoU)**, formally known as the *Jaccard index*. As shown in figure 3.5 [39], the technique involves computing the ratio between the *area of overlap* between the *predicted* bounding box and the *ground truth* bounding box, and

the *area of union* i.e. the area encompassed by the predicted bounding box and the ground truth bounding box combined.

The **IoU** score will be used as a metric to reward predicted bounding boxes heavily overlapping with the ground truth. Figure 3.6 [40] shows good and bad examples of bounding boxes with different **IoU** scores. It also demonstrates that a closer match between prediction and ground truth yields a higher score, while an exact match is not *required* for the prediction to be considered accurate.


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


FIGURE 3.5: IoU mathematical formula



FIGURE 3.6: Comparison of poor, good and excellent IoU scores

3.2.3 Duplicate detection avoidance

In addition to using **IoU** to evaluate and score predicted bounding boxes, an **IoU** threshold can be used to avoid multiple predictions of the same object. Figure 3.7 shows the result of object detection run on an image from the **BDD** dataset. The model does a decent job at detecting the traffic lights and the 3 closest cars in the image. An important observation is the fact that the closest car straight ahead is predicted twice. One bounding box encloses the car pretty well. Unfortunately the model predicts a second car in the same area, shown by the second bounding box that is smaller and shifted slightly to the left. This is undoubtedly the same car, and there should only be one bounding box. This problem can in many cases be solved by performing **Non-maximum suppression (NMS)** [41]. A greedy version of **NMS** solves the problem of multiple predictions of the same object by:

1. Finding the confidence score, i.e. the probability of the box containing the object, for each detection.
2. Identifying the bounding box with the highest confidence score.
3. Suppressing (removing) all the bounding boxes which have **IoU** scores greater than a certain *threshold* with the bounding box with the highest confidence score.

Figure 3.7 shows a model run with an **IoU** threshold of 0.5, which is a high enough threshold to allow the duplicate bounding boxes to avoid being suppressed by **NMS**. Reducing the **IoU** threshold will increase the likelihood of a bounding box being removed for referring to the same object as another box, and might help in a situation such as this. Figure 3.8 shows the same image with the **IoU** threshold lowered to 0.3, where it is evident that the duplicate detection of the car has been removed. It is thus important to keep the **IoU** threshold in mind when working with object detectors, to avoid such issues. Based on preliminary tests, a threshold of 0.5 is used throughout this thesis.



FIGURE 3.7: IoU threshold of 0.5

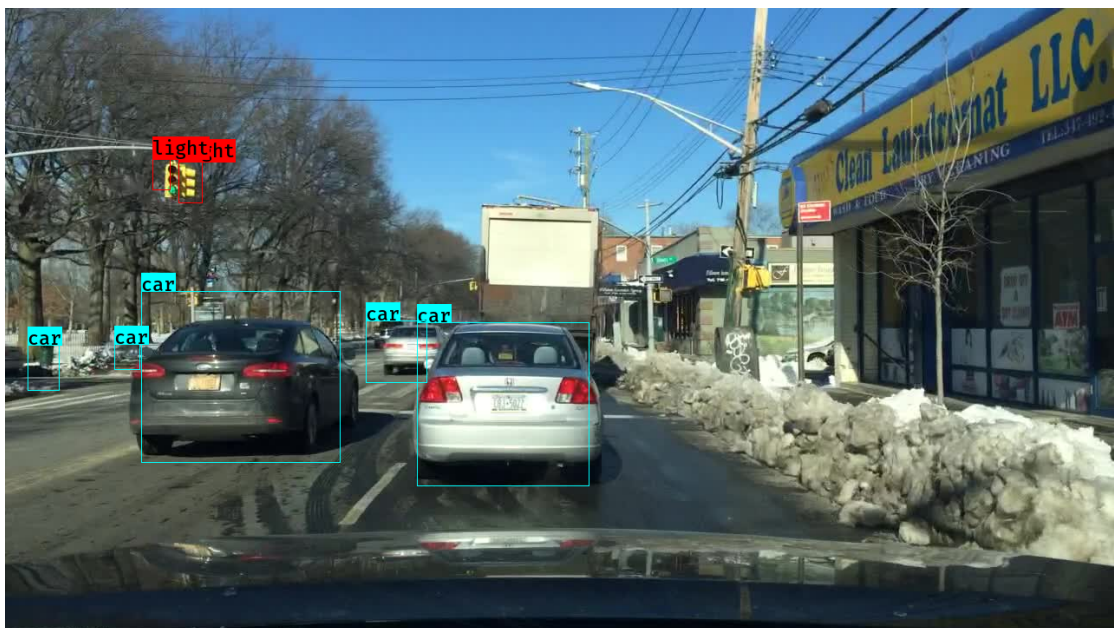


FIGURE 3.8: IoU threshold of 0.3

3.3 Performance evaluation

3.3.1 Mean Average Precision

Once a model's predictions have been determined on an image, they can be compared with the ground truth predictions to determine the accuracy of the model. Performance

is measured using [mean Average Precision \(mAP\)](#), a metric commonly used to measure the accuracy of object detectors [42]. [mAP](#) refers to the mean of the maximum *precision* values at different *recall* values. Precision measures how accurate predictions are, i.e. the percentage of positive predictions that are correct. Recall measures how good the model is at finding all the positive predictions. The mathematical definitions can be seen in equation 3.1 and 3.2.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (3.1)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3.2)$$

The definition of a [True Positive \(TP\)](#) is a prediction's bounding box with an [Intersection over Union \(IoU\)](#) score with the ground truth prediction above a certain threshold. An [IoU](#) score threshold of 0.5 is used in this thesis' experiments. Precision can be plotted against recall to get a plot similar to figure 3.9 [43]. The average precision (AP) can be found by calculating the area under this plot, and the [mean Average Precision \(mAP\)](#) is simply the average precision over all of the available classes.

The [mAP](#) metric is rewarded by [TPs](#) and punished by [False Positive \(FP\)](#)s, and will prefer few correct predictions, over many both correct and incorrect predictions.

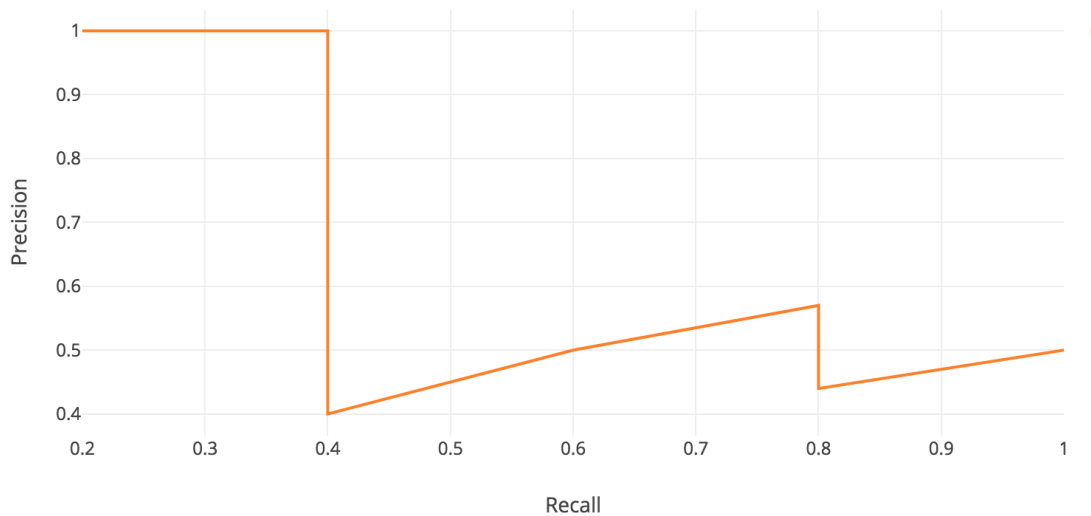


FIGURE 3.9: Precision-Recall example plot

3.3.2 F_1 -score

The mAP score is useful to assert the *overall* performance of a model over a set of images. However, to assess the performance of a model's predictions on a single image, the F_1 -score yields more useful information. Using the precision and recall metrics introduced in equation 3.1 and 3.2, the F_1 -score can be defined as [44]:

$$F_1 = \frac{2 \cdot \textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}} \quad (3.3)$$

The F_1 -score represents the harmonic mean between the precision and recall of the model's predictions. Commonly used as a performance metric in ML [45], this metric puts equal importance on the models' precision and recall values. Doing so ensures a balance between models making many, poor predictions, and models making few, good predictions. The F_1 -score is used by the CSD, introduced in section 4.2.1, in its underlying *policy* to select the best performing models in any given context.

Chapter 4

Decentralized Artificial Intelligence for autonomous driving

An autonomous vehicle in a [Decentralized Artificial Intelligence \(DzAI\)](#) network needs to be able to utilize and take advantage of the collective wisdom of the network's multiple participants. In the [Decentralized Autonomous Driving \(DzAD\)](#) network proposed in this paper, the participating models are trained on diverse small data distributions. Some have been trained in more urban contexts and are naturally better at predicting pedestrians and cars clustered together, while others might be trained in contexts where there are more trucks, like highways. Even though these *weak models* are unlikely to perform optimally on their own, their strengths can be combined. This thesis explores two ways of creating a [DzAD](#) system. The first of which is to utilize *ensemble learning* to combine the object detecting capability of each diverse model in the network into a more accurate [Ensemble Detector \(ED\)](#). The second approach involves using *online learning* to quickly switch between models with different biases using a [Context-Sensitive Detector \(CSD\)](#).

4.1 Ensemble Learning

The concept of combining the strengths of multiple weak learners into a *collective wisdom* is called *Ensemble learning* [46]. *Weak learners*, a term commonly used for these types of models, are common in areas where Big Data and [Machine Learning \(ML\)](#) are involved. They are also referred to as *experts*, *detectors* or simply *models*. These terms are treated as interchangeable, and models will be used consistently throughout the thesis.

There are several different methods within the space of Ensemble learning, two of which were used as inspiration for the [Ensemble Detector \(ED\)](#) implemented in this thesis:

- Bootstrap aggregating (also called bagging), where each model in the ensemble votes with an equal weight. In bagging, each model in the ensemble is trained on a randomly drawn subset of the training set [47].
- Stacking with [Weighted Majority Voting \(WMV\)](#) [48], where each model votes on the prediction, and the votes are weighted. The weights can all be equal, in which case they have no effect and the algorithm is equal to bagging. Alternatively unequal weights can be used, and tuned based on the models' performances on a training set. This is known as the [Weighted Majority Algorithm \(WMA\)](#) [49].

4.1.1 Weighted Majority Voting and the Weighted Majority Algorithm

The [Weighted Majority Algorithm \(WMA\)](#) is a meta-learning algorithm used to lift the performance of weak models trained on small datasets closer to that of a model trained on a larger dataset [49]. The algorithm performs [Weighted Majority Voting \(WMV\)](#) and is used to construct a compound model containing a pool of models, each with their own positive weight. The compound model collects votes from all of the models in the pool (line 3 in algorithm 1), and outputs a prediction by performing [WMV](#) (line 4 in algorithm 1). If the majority vote is incorrect compared to the ground truth, the models in the pool that voted for the wrong prediction are punished by having their weights reduced by a certain ratio ε , where $0 < \varepsilon < 1$ (line 6 in algorithm 1). Algorithm 1 is designed for binary classification, but the method can be extended to work with multi-class classification as well.

Algorithm 1: Weighted Majority Algorithm pseudocode

Input: N models predicting the outcomes, a parameter $\varepsilon > 0$

- 1 $w_i^1 \leftarrow 1$ for all $i = 1, \dots, N$ (weights initialized to 1 for all models) ;
 - 2 **for** rounds $t=1, 2, \dots$ **do**
 - 3 $f_i^t \in 0, 1$ (obtain prediction of model i , $i = 1, \dots, N$;
 - 4 $\hat{y}_t \leftarrow \text{round} \left(\frac{\sum_i w_i^t f_i^t}{\sum_i w_i^t} \right)$ (this is the WMV step) ;
 - 5 Outcome y_t is revealed ;
 - 6 $w_i^{t+1} \leftarrow w_i^t (1 - \varepsilon)^{\mathbb{1}[f_i^t \neq y_t]}$ (this is the weight tuning step) ;
 - 7 **end**
-

4.1.2 Ensemble Detector definition

The paper in [12] describes an implementation of an *Ensemble Classifier* using the [Weighted Majority Algorithm \(WMA\)](#) with a few alterations to work with multiple classes. The method used in the paper is reported to achieve good results when combining up to 100 classifiers trained on small partitions of the *MNIST* [50] and *20 newsgroups* [51] datasets. However, the accuracy of the [WMV](#) result was more or less the same before and after the weight tuning step (line 6 in algorithm 1), indicating little effect of the weight tuning process. Regardless of weight tuning, the results in [12] are promising. However the simple classification task in [12] is different from the multi-object *detection* task described in this thesis. First of all, this means that the Ensemble Classifier implementation must be altered to perform object *detection* instead. Secondly, when there are multiple objects to both classify and detect in each image, weight tuning in the [WMA](#) becomes more complicated. Therefore the models are not assigned tunable weights, instead a prediction's *confidence score* is used as the weight in the [WMV](#).

Summarized, the algorithm asks all the models in the ensemble for their predictions, and performs [WMV](#) on every object predicted, using the confidence scores as the weights. Thus a more certain model will vote with a larger weight than a less certain model. Primarily this is to reduce the complexity of the task. Consider an ensemble of models A and B. A makes a number of predictions, some are correct and some are not. Model B makes no predictions, neither correct nor incorrect. Even in this simple case it is non-trivial to determine how much these two models' weights should be punished relative to each other.

4.1.3 Ensemble Detector implementation

The **Ensemble Detector (ED)** contains a collection of trained **You Only Look Once version 2 (YOLOv2)** models, and produces predictions in the following way.

1. The **ED** is initialized with a certain number of models already trained on a small, biased, dataset.
2. The models in the ensemble are all weighted equally.
3. Given the data at hand, each model is asked for its prediction, which will be comprised of class labels, confidence scores, and boxes bounding the detected objects.
4. For each image, every model's prediction referring to the same object is determined. This is referred to as an *overlap* or a *conflicting prediction*.
5. **Weighted Majority Voting (WMV)** is performed on each of the overlapping predictions to determine the class. The confidence scores are used as the weights in the **WMV**. Choosing the object's resulting bounding box can be done in multiple ways, and choosing the box with the highest score is used in this implementation.
6. Any prediction that does not overlap with other predictions is kept unchanged and carried through to the ensemble prediction. These predictions are called *standalone predictions* and are important as they often represent objects that only 1 model managed to detect, and that was not found by the others. Sometimes these are **False Positive (FP)**s, but in most cases they are **True Positive (TP)**s and important to include.
7. Finally all the predictions are combined.

The biggest difference between the Ensemble *Classifier* described in [12], and the Ensemble *Detector* described in this thesis is the way voting is performed, and how the weights are used. In a classification task, a prediction can be deemed correct or false by simply comparing the predicted classification with the ground truth classification. In the case of the MNIST dataset, if the ground truth label of an image is '9', and the prediction is also '9', the prediction is correct, and the model(s) that voted for this prediction should not be punished. In the object detection task that this thesis is concerned with, the comparison between prediction and ground truth label is not so simple.

Figure 3.4 in section 3.2.2, showing two boxes bounding a detected sign, demonstrates this fact. A model not only has to classify the predicted objects, it also needs to locate and draw bounding boxes around them. The difference between the classified type of object (sign, car, pedestrian etc.) and the ground truth label can be done by a simple string comparison, but determining the *degree* of which a predicted bounding box is *correct* or not requires more sophisticated methods. This challenge is complicated even further when there are multiple objects of different classes in a single image. If a model detects all the cars and draws bounding boxes around them, but misses a number of pedestrians, it is difficult to say whether the model is good or bad.

4.1.4 Ensemble Detector predictions

The process of creating a *single* ensemble prediction based on multiple models' predictions is described at a high level in section 4.1.3. Point 4 needs a more thorough explanation, however. After all the models in the ensemble have been asked for their predictions, the ensemble algorithm needs to determine which model's prediction corresponds to the same object as another model's prediction, to be able to perform majority voting. In other words, if model A and B both predict 10 objects, the algorithm needs to know which, if any, are the same objects. It's not safe to simply iterate from left to right in the image, and assume that prediction X in model A corresponds to the same object as prediction X in model B . This approach will easily be offset by a mistake in one or more models.

Hence, corresponding predictions are determined by an [Intersection over Union \(IoU\)](#) threshold, a concept described in section 3.2.2. If two or more predicted bounding boxes from different models have an [IoU](#) value over a certain threshold, the predictions are assumed to refer to the same object, and an *overlap* is identified.

After all of the overlapping predictions have been determined, [Weighted Majority Voting \(WMV\)](#) can be performed to reduce the number of predictions to 1 per object. The predicted object's class is decided by a weighted majority vote, with the confidence score of each prediction used as the weights. The resulting bounding box is the one with the highest confidence score, with the same class as the voted class.

4.2 Online Learning

Online learning is a [Machine Learning \(ML\)](#) technique where models are adjusted as data becomes available, as opposed to offline learning where the best model is prepared ahead of time. Techniques from online learning are commonly used in [Reinforcement Learning \(RL\)](#) where actions are chosen based on prior rewards in order to maximize the overall reward.

One of the major challenges in [RL](#) is the balance between exploration (search for new knowledge) and exploitation (use of current knowledge). [RL](#) techniques utilized in approaches like DeepMind’s Alpha Go program [52] use two separate [Neural Network \(NN\)](#)s to balance this exploration-exploitation trade-off, and avoid excessive search depths slowing down the decision process. Object detection during driving is a challenging task that needs to be performed in *real-time*. In order to provide a proof of concept and assess the feasibility of this approach the experiments in chapter 6 explore a faster, stochastic scheduling approach, known as the [Multi-Armed Bandit \(MAB\)](#).

The process of selecting models to do object detection whilst driving is reminiscent of a single-state [Markov Decision Process \(MDP\)](#) [53], consisting of:

- S , a set of environments.
- A , a set of agent actions.
- $P_a(s, s')$, the probability of transitioning from state s to state s' from action a .
- $R_a(s, s')$, the reward after transitioning from state s to state s' from action a .

The goal is to find a policy π maximizing the cumulative reward over an infinite horizon. In the context of driving the goal is to select the models that best perform object detection in different contexts encountered while driving from A to B. The contexts are modelled as:

$$S_I = \{I_0, I_1, \dots, I_n\} \quad (4.1)$$

Where I is the set of images encountered in an area, like the city or the countryside. A drive from location A, through location B, ending in location C can be modelled as:

$$S_A \rightarrow S_B \rightarrow S_C \quad (4.2)$$

The decision process for object detection during driving is illustrated in fig. 4.1.

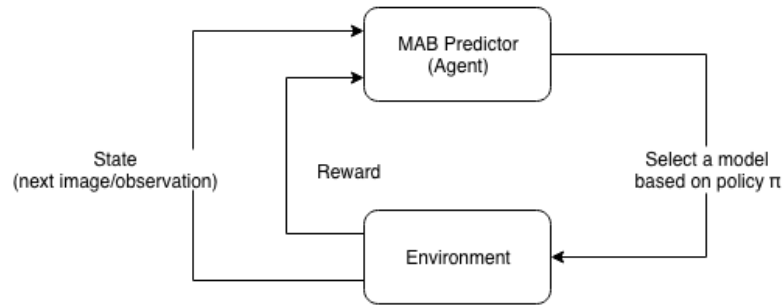


FIGURE 4.1: Markov Decision Process for object detection during driving

4.2.1 Context-Sensitive Detector

In order to utilize online learning to perform real-time predictions, the concept of using a [Context-Sensitive Detector \(CSD\)](#) is presented in this thesis. The detector consists of multiple weak models, trained on small datasets, similar to those introduced in section 4.1. Using a stochastic scheduling approach, explained in depth in section 4.2.2, the [CSD](#) attempts to pick the current best performing model based on historical knowledge. The idea is that the performance of each model is contextually dependent on the context they were trained in. The goal of this thesis is to assess whether finding and exploiting models trained on small, biased datasets can achieve similar performance to a model trained on a larger dataset.

4.2.2 Multi-Armed Bandit

The multi-armed bandit problem is a classic [Reinforcement Learning \(RL\)](#) problem and is often formalized as a single-state [Markov Decision Process \(MDP\)](#) [54]. Given n slot machines (sometimes referred to as one-armed bandits) the goal is to maximize the overall profits without knowing the different reward distributions for each slot machine. In order to maximize their profits, a gambler needs to find a balance between exploring (and

increasing their confidence in the distribution for a given slot machine), and exploiting the most rewarding machine.

The periodical change in object distributions observed while driving introduces another challenge to the [CSD](#). Where the classic [MAB](#) problem is an example of a stationary time-series, with the goal of finding the global maximum reward, the object detection example presents an ever-changing environment, and needs to be treated as a non-stationary time-series. Following is a brief introduction to the standard stationary [Upper Confidence Bound \(UCB\)](#) approach, and building on that, a non-stationary approach, using a sliding window.

4.2.3 Upper Confidence Bound

From the introductory example in section [4.2.2](#) each slot machine has their own distribution, and as such there should be a single distribution that maximizes the potential reward from a sequence of plays. With each play, the confidence in the distribution for the played machine increases and given an infinite number of plays it would be possible to identify the optimal machine. The [Upper Confidence Bound \(UCB\)](#) policy tries to balance the exploration-exploitation trade-off by only playing machines whose upper confidence is above the lower confidence of the currently best performing machine, illustrated in figure [4.2](#).

The [UCB](#) policy tries to exploit the machine that maximizes the policy reward found in equation [4.3](#) [55]. If no machines are outperforming the others, the machine is selected randomly. Mathematically the [UCB](#) policy can be expressed as:

$$\bar{x}_j + \sqrt{\frac{2 * \ln n}{n_j}} \quad (4.3)$$

where \bar{x}_j is the average reward obtained from machine j , n_j is the number of times machine j has been played, and n is the overall number of plays so far.

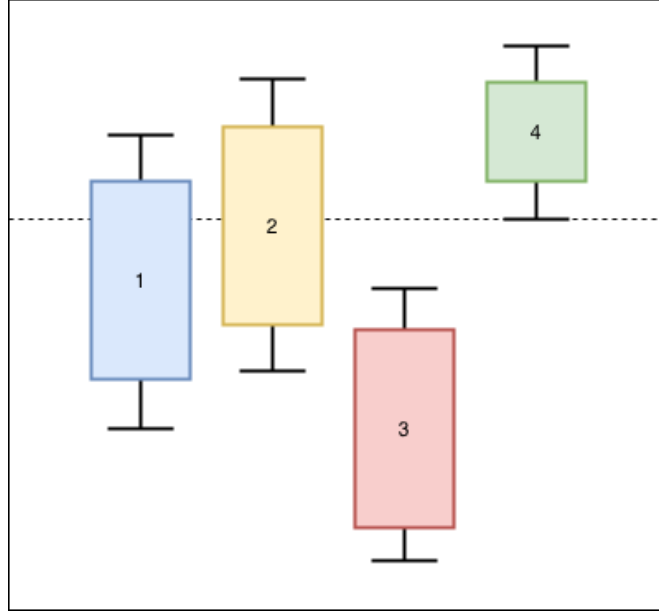


FIGURE 4.2: Upper confidence for 4 machines. Machine 3 will not be selected as one of the next machines

4.2.4 Sliding Window Upper Confidence Bound

While the standard **UCB** policy works great in a non-changing environment, it fails when the underlying distributions start to change. As such the confidence of the algorithm needs to be adjusted in real-time by looking at the distributions through a *sliding window*. By using a sliding window the goal is to find a balance between exploiting historical knowledge (*memory*), and punishing suddenly poorly performing models. This is especially important if the car is moving from location A, through location B and back to location A again:

$$S_A \rightarrow S_B \rightarrow S_A \quad (4.4)$$

In the above example the policy should observe a sudden performance drop when moving into S_B , and explore the other models until it finds one with better performance. When moving back into S_A again, the same thing happens, but it is important that it is able to pick up the same model that it used during the first pass through S_A . If that model is being punished too harshly transitioning into S_B , it could affect performance negatively in the long run. In order to retain recently well-performing models, and punish suddenly poorly performing ones, the policy is modeled as [56]:

$$\bar{x}_\tau + B\sqrt{\frac{\varepsilon \ln \min(t, \tau)}{n_\tau}} \quad (4.5)$$

The experiments in chapter 6 explore how the different hyper-parameters, the discounting factor (ε) and the window size (τ), affect the performance of the [SW-UCB](#) policy.

Chapter 5

Ensemble Detector experiments and results

5.1 Main purpose and metrics

The purpose of this experiment chapter is to answer the second [Research Question \(RQ\)](#), described in section [1.2](#). To answer the [RQ](#), multiple experiments with the [Ensemble Detector \(ED\)](#) described in section [4.1.2](#) are run, varying the hyper-parameters:

- Training quality, i.e. the size of the training set and the number of epochs. In [Deep Learning \(DL\)](#), increasing the size of the training set and the number of epochs generally results in better predictive accuracy (disregarding overfitting).
- Number of models in the ensemble. For the [ED](#) to be able to utilize the strengths of the weak models, it likely needs more than just a few models in its ensemble.
- Enabled classes. Excluding certain classes from the experiments helps to simplify the task, and make the effect of the ensemble learning more apparent.

5.2 Experiment test set

The models in the experiments were all tested on the same dataset. Preliminary experiments were conducted with a dataset created by drawing a set of images from all of the

validation sets of the models in the ensemble. One benefit of this approach is that the dataset is drawn from the same distribution the models are validated on. However, one or more models are guaranteed to be tested on images from their own validation set, violating the best practice requirement of independence between the test and validation sets.

Another problem discovered using this approach was the varying difficulty of images in the test set, as they contain objects of different sizes, at varying distances, and with challenging lighting. Additionally, many images are captured in motion and the objects are blurry, occluded, or truncated. This problem became evident after testing the same [Ensemble Detector \(ED\)](#) multiple times on different samples from the test set, resulting in wildly different performance scores.

To combat this, the experiments described below were performed with a predefined, hand-picked test set consisting of 100 images. The test set was created using a combination of distribution analysis and visual inspection, to determine a set of images without factors that make the objects unnecessarily difficult to detect. Half of the set consists of images taken in daylight, and half are taken in the dark.

5.3 Experiment setup

The experiments have 3 components:

- A *strong* model trained on 20 000 images with the intention of serving as a baseline for comparison, as described in section 3.1.5, representing the centralized [AI](#) approach.
- A group of weak models trained on either 500 or 1000 training images, over a certain number of epochs.
- An [Ensemble Detector \(ED\)](#) consisting of all of the weak models, representing the decentralized approach.

All of the components described above are tested on the same set of images. Their predictions are compared with the ground truth predictions and the [mean Average Precision \(mAP\)](#) (sec. 3.3.1) is computed and plotted. For each of the the groups the

average **mAP** and standard deviation is plotted. The purpose of this is to assess whether the **ED** performs better than the average weak model.

5.4 Experiments

5.4.1 Experiment 1: 5-10 models, all classes

Enabled classes: Bus, traffic light, traffic sign, person, bike, truck, motor, car, train, rider (all).

Setup Experiment 1 is set up as an initial experiment with all 10 classes enabled to assess how each of the different model components perform compared to each other, setting the course for the subsequent experiments. The ensemble consists of 5-10 models, each trained on 1000 images for 15 epochs.

Hypothesis The baseline model is expected to perform the best, and the weak models are expected to perform poorly based on the small training set. As the number of weak models increases, the performance of the **ED** is expected to follow, but stay below the strong baseline model trained on 20 000 images.

Results A plot of each component's performance can be seen in figure 5.1. The x-axis shows the number of models used, and the y-axis shows the **mean Average Precision (mAP)** in percentage points. The red dotted line indicates the **mAP** of the baseline model, which will be independent of the number of models used (x-axis). The orange solid line plot represents the **ED's** performance, ranging from an ensemble of 5 models up to 10 models, with a 5 model step size. The blue dotted line shows the average **mAP** of the models used in the ensemble, and the shaded area shows their standard deviation.

The observed standard deviation is quite large, ranging from $\sim 2\%$ **mAP** up to 10.5% with a mean of 7.2% for 5 models. This is likely caused by certain objects being more difficult to predict than others: The models have previously demonstrated difficulty detecting smaller objects, like traffic lights or traffic signs, or objects that are far away, like cars in the distance. Reducing the number of classes might help reduce the impact

of this issue. The 20k baseline model sits just below 11% **mAP** which is better than any of the individual models are able to perform on their own, but a lot worse than the base **YOLOv2** model on the *COCO* [23] dataset (44%) [17]. The **ED** is able to perform nearly as well as the baseline and indicates a slight increase in **mAP** as the number of models increases. An example of the **ED**'s predictions can be seen in figure 5.3a. The detector accurately predicts the cars, and two of the lights in the image, but most of the other predictions are not precise enough, negatively affecting its **mAP**.

In comparison, figure 5.3b shows the predictions of a model trained on images with at least 1 instance of traffic lights. This model also correctly predicts some lights, and a couple of cars, but not much else. No sign predictions are made. Lastly, figure 5.3c shows the predictions of the baseline model which also fails at predicting traffic lights and signs, but is more accurate than the aforementioned models on almost all other objects.

Figure 5.2 shows the same image with the ground truth predictions drawn onto it, showcasing how difficult the task at hand really is. Cars far away look a lot like traffic lights, and multiple objects are so close to each other that they get occluded. To try and reduce the complexity of the task, the subsequent experiments are performed with fewer classes enabled.

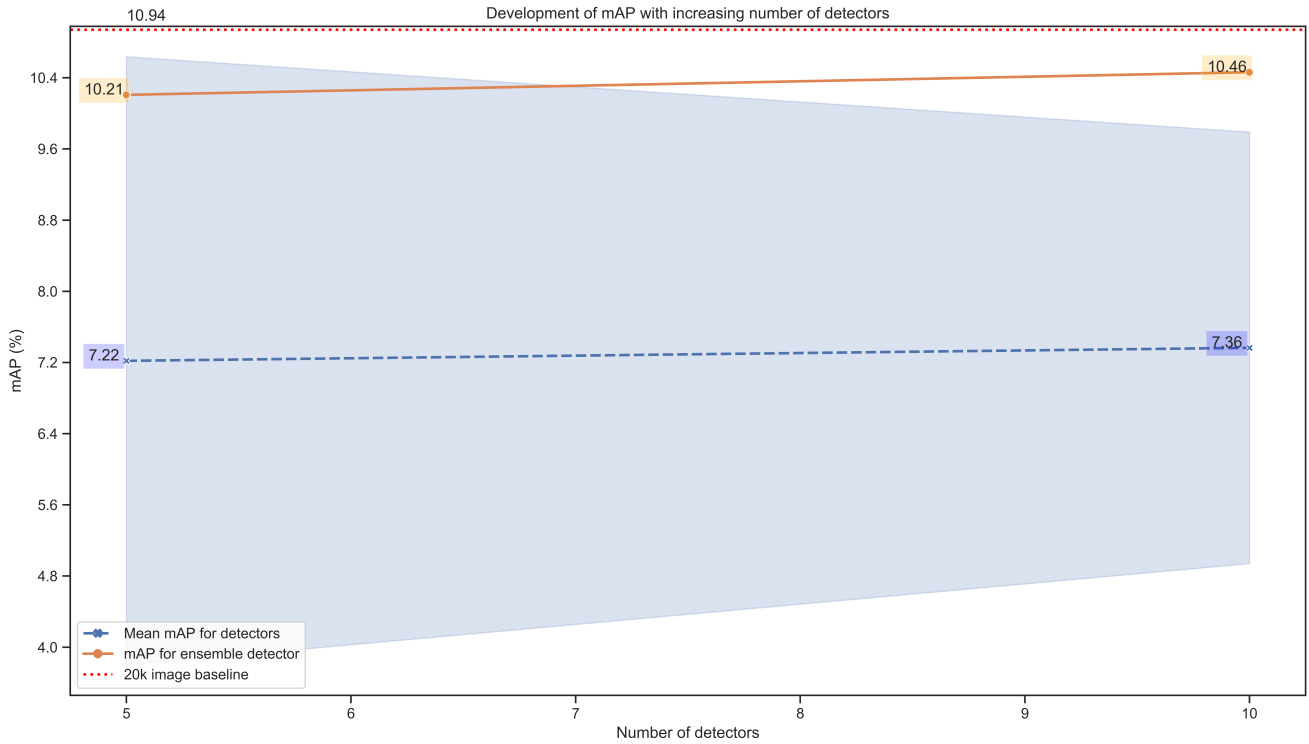
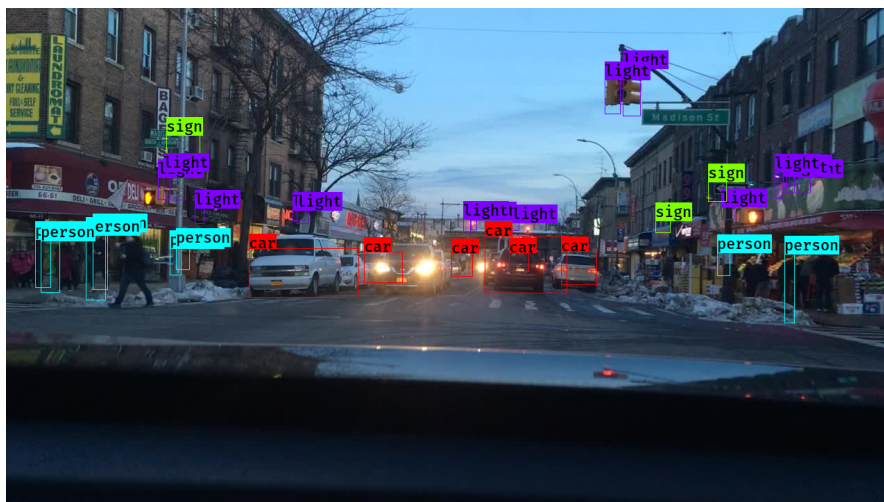


FIGURE 5.1: Experiment 1 mAP plot



FIGURE 5.2: Experiment 1 ground truth.



(a) Ensemble Detector.



(b) Model trained on images with at least 1 instance of traffic lights.



(c) 20k baseline.

FIGURE 5.3: Experiment 1 drawn predictions.

5.4.2 Experiment 2: 5-10 models, low quality training

Enabled classes: Car, truck, person.

Setup Experiment 2 is set up to evaluate models trained on a small training set of 500 images, for only 5 epochs. The number of enabled classes has been reduced from experiment 1, to reduce the complexity of the task and to emphasize the effects of the ED. The classes were selected based on preliminary experiments: *Traffic lights* and *traffic signs* were removed because the models had trouble predicting them regardless of the size of the training set. The *train* class was removed merely because of its rarity: there are only 179 instances in the entire dataset of 100 000 images [36]. The *bus* class was excluded for the same reason. *Motor*, *rider* and *bike* were also excluded for having a sparse presence, in addition to appearing as small objects in most occurrences, causing the same problem as *traffic lights* and *traffic signs*.

Of the remaining 3 classes, *car* was kept because of its prevalence in the training set: the class is present in $\sim 70\%$ of all the images [36]. Preliminary experiments showed that excluding it caused the models to generalize too much from similar classes like *truck*, and not properly distinguish the two. The *truck* class was kept for the same reason. Keeping one but excluding the other resulted in cars being predicted as trucks and the other way around, because of their similarities. Lastly, the *person* class was kept because it's the second most common class (excluding *lights* and *signs*) and its lack of similarity with the other classes.

Hypothesis With 3 classes instead of all 10, the performances of the baseline, the ensemble and the weak models are all expected to increase. Simultaneously the small training set and low number of epochs will likely decrease the performance of the weak models. The net result is expected to be a slightly improved performance from the weak models from experiment 1. The ED is hypothesized to be able to take advantage of the weak models' strengths and perform better than them, but not quite as well as the baseline.

Results Shown in figure 5.4, the performance of the baseline model increased by nearly 60%. The weak models, however, dropped in performance, but so did the standard

deviation, confirming the hypothesis that the high standard deviation in experiment 1 was caused by certain classes being more difficult to predict than others. Surprisingly the ED performed worst of all, at a poor 2.2% mAP for 5 models, below the average mAP of the weak models at 4.5%. When the number of models increased to 10, the performance of the ED increased approximately to the same value as the weak models.

To determine whether the decentralized approach (represented by the ED) is a useful alternative to the centralized approach (represented by the baseline), experiments with a higher number of models need to be performed, to confirm or deny that the trend demonstrated by the ED in figure 5.4 continues. The predictions made by the models in experiment 2 can be seen in figure 5.6.

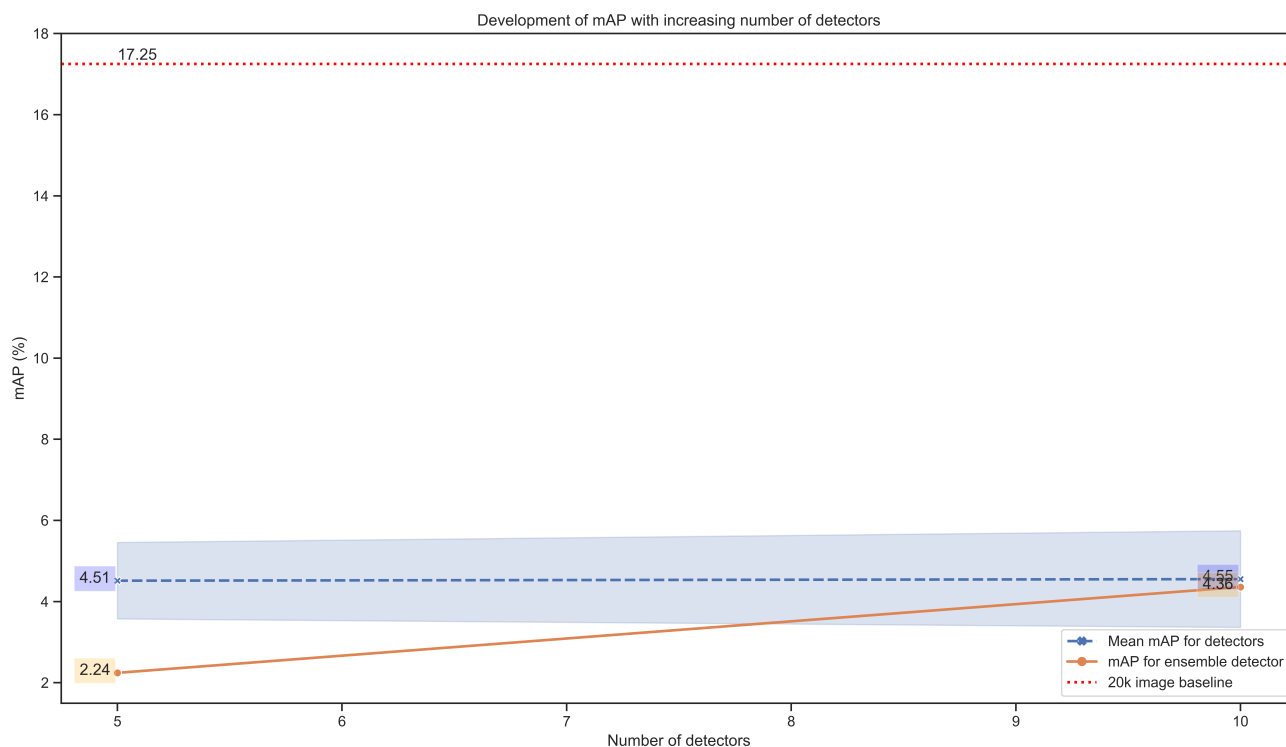


FIGURE 5.4: Experiment 2 mAP plot



FIGURE 5.5: Experiment 2 ground truth.



(a) Ensemble Detector (10 models).



(b) Model trained on images with more than 3 instances of trucks and more than 3 instances of cars.



(c) 20k baseline.

FIGURE 5.6: Experiment 2 drawn predictions.

5.4.3 Experiment 3: 5-20 models, high quality training

Enabled classes: Car, truck, person

Setup Experiment 3 is set up with the same classes as in experiment 2, with a range of 5 to 20 models, with a 5 model step size. EDs are constructed of the models at each step size. The quality of the training has also been improved: Each of the models is trained on 1000 images for 15 epochs in order to improve the performance shown in the previous experiment. The baseline remains unchanged.

Hypothesis This experiment's purpose is to answer the following two questions, raised by the previous experiment:

- How does the quality of the training affect the performance of the weak models after reducing the number of enabled classes?
- Does the mAP of the ED increase as the number of models increases?

The answer to the first question is presumed to be affirmative: Even though the number of classes were decreased from experiment 1 to experiment 2, the performance of the weak models dropped, presumably because of the reduced training quality. Improving the training conditions again is expected to raise the performance beyond the results seen in experiment 1.

More important is question 2, integral in determining whether [Decentralized Autonomous Driving \(DzAD\)](#) with ensemble learning can achieve results on par with the centralized approach. The ED's mAP is expected to increase together with the number of models, and approach the performance of the baseline model.

Results Figure 5.7 confirms the hypothesized answer to question 1: The mAP values of both the weak models and the ED are higher than in both experiment 1 and 2.

Shown in figure 5.7, the mAP of the ED is not strictly monotonically increasing, but decreases from 5 to 10 models before rising again. The explanation of this lies in how the [mean Average Precision \(mAP\)](#) metric works. As explained in section 3.3.1, the

mAP metric rewards **True Positive (TP)**s, i.e. predictions of the correct class, with bounding boxes that overlap with the ground truth predictions' bounding boxes with an **Intersection over Union (IoU)** over 0.5. Simultaneously it punishes **False Positive (FP)**s, i.e. predictions whose bounding boxes do not overlap with any of the ground truth bounding boxes with a sufficiently high **IoU** value. When combining an increasing number of weak models in an ensemble, the chance of at least one of those models contributing with **FP** predictions, increases. If the prediction in question is not *contested*, i.e. it has no overlapping predictions from other models, it is kept as is and not involved in the **Weighted Majority Voting (WMV)**. A trivial solution would be to simply remove these *standalone* predictions, as in many cases they represent obvious **FPS**. However, more often than not, they are correct predictions that none of the other models were able to predict, and are important to keep in the final prediction. After all, the whole point of the **Ensemble Detector (ED)** is to utilize the strengths of weak models.

This is evident when looking at the drawn predictions, for instance when comparing figure 5.9a, showing the predictions of an ensemble of 20 models, with figure 5.9b, showing the predictions of an ensemble of 5 models. The 5-model **ED** in figure 5.9b is unable to predict the leftmost *person* in the image, but the 20-model **ED** gets it right. At the same time figure 5.9a clearly includes more **FPS** for the reasons mentioned above, like the person erroneously being predicted on the street corner between the white car and the person in the middle.

Nonetheless, observing the performance trend of the **ED** with a further increase of the number of models is of interest and the topic of the next experiment.

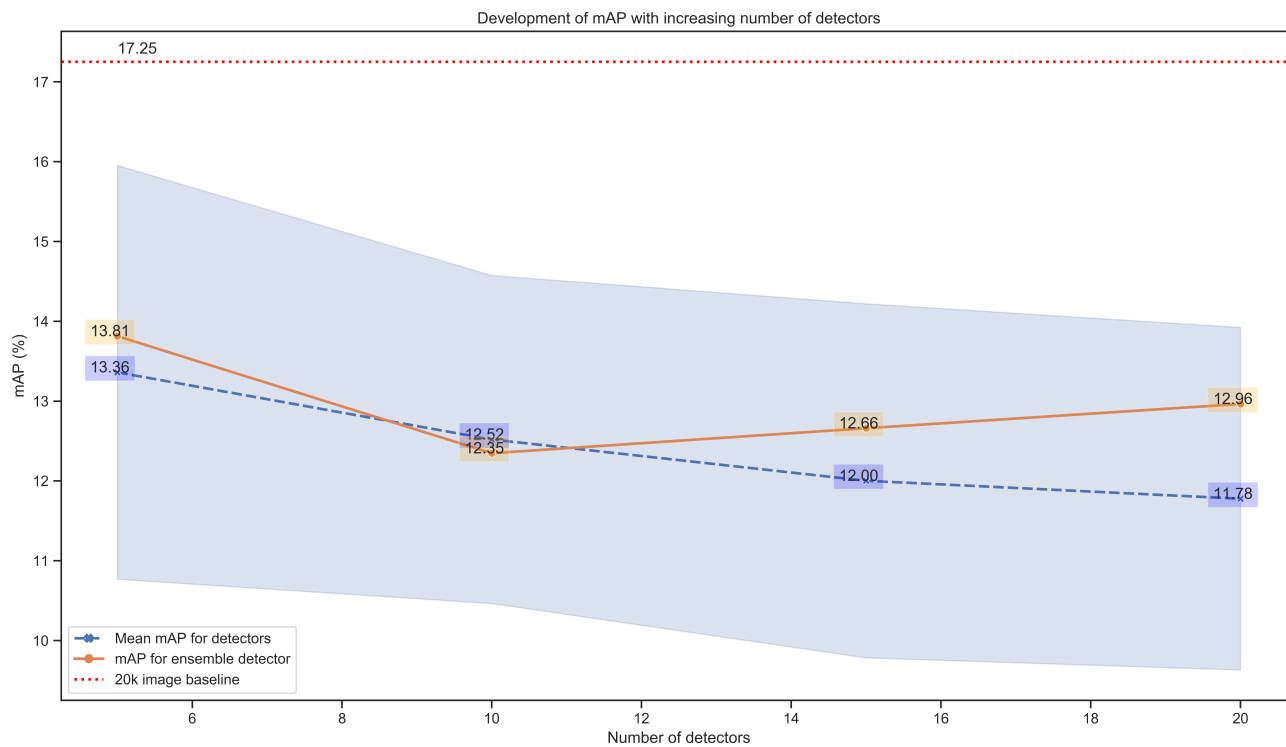


FIGURE 5.7: Experiment 3 mAP plot



FIGURE 5.8: Experiment 3 ground truth.



(a) Ensemble Detector (ensemble of 20 models).



(b) Ensemble Detector (ensemble of 5 models).



(c) 20k baseline.

FIGURE 5.9: Experiment 3 drawn predictions.

5.4.4 Experiment 4: 5-50 models, high quality training

Enabled classes: Car, truck, person

Setup Experiment 4 is identical to experiment 3, except that the model ensembles range from 5 all the way up to 50 models, again with a step size of 5 models.

Hypothesis The purpose of this experiment is to observe the effect of increasing the number of models even further, and assess whether a trend can be identified, warranting further experiments. Based on the results in, and for the reasons described in experiment 3, increasing the number of models further is not expected to provide more than a slight increase in [mAP](#).

Results Figure 5.10 shows the development of the [mAP](#) as the number of models increases from 5 to 50. Similar to experiment 3, the performance of the [ED](#) does not monotonically increase, but fluctuates between $\sim 11\%$ and $\sim 13.5\%$. It does not rise any closer to the baseline model's performance as the ensemble grows bigger, and shows no clear correlation with the ensemble size.

However, the [ED's mAP](#) values stay above the average [mAP](#) (the blue line in fig. 5.10) of the individual models for every datapoint except for one. In addition, the [ED's](#) performance sticks close to the top line of the shaded blue field indicating the standard deviation of the weak models. This is important when considering its use in [Decentralized Autonomous Driving \(DzAD\)](#): rather than using an [ED](#), one could simply pick a random model and trust its predictions. However, since the [ED](#) stays closer to the best case scenario (top line of the shaded blue area) than the worst case (bottom line), statistically it will perform better.

Nevertheless, the experiments do not show promising results for the performance of the [ED](#) compared to the baseline model. The final section of this chapter summarizes the results.

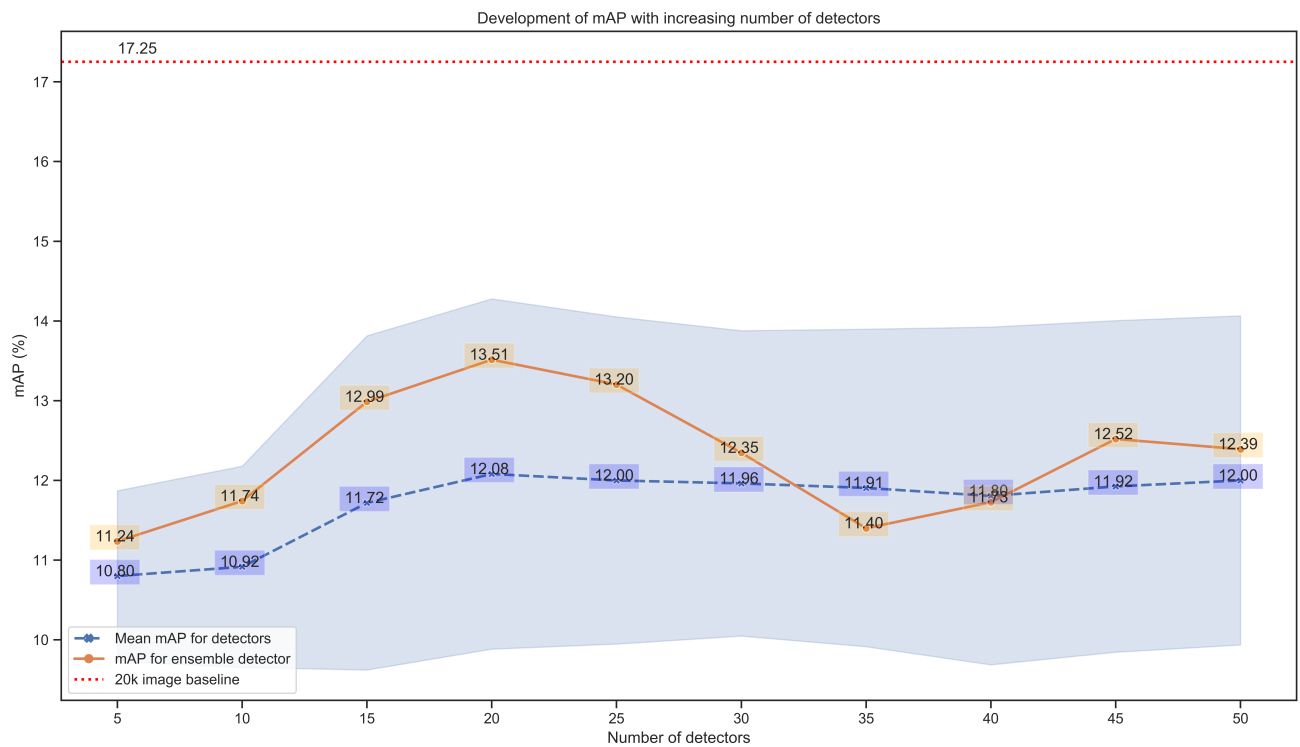


FIGURE 5.10: Experiment 4 mAP plot

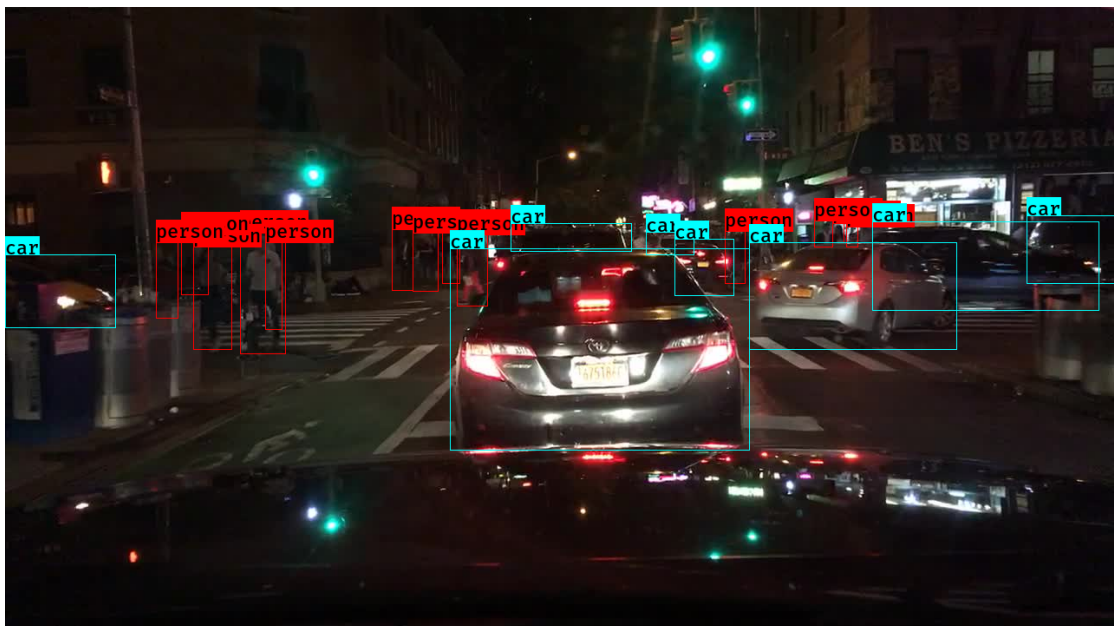
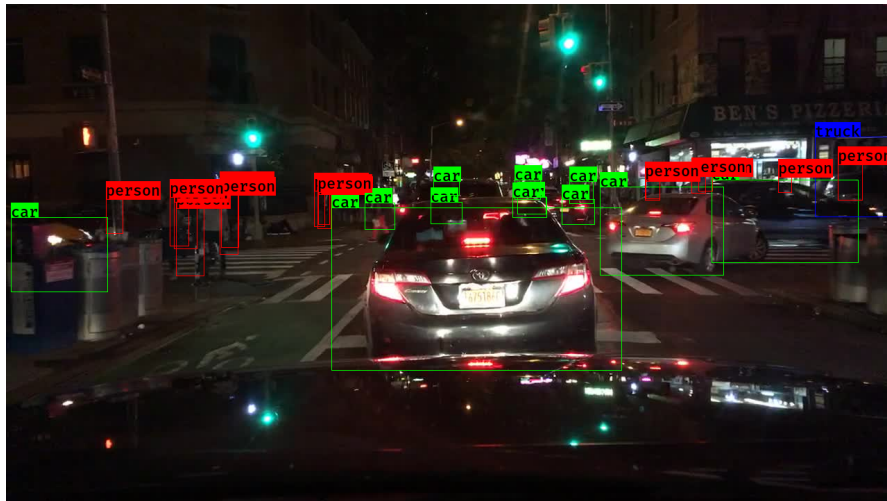
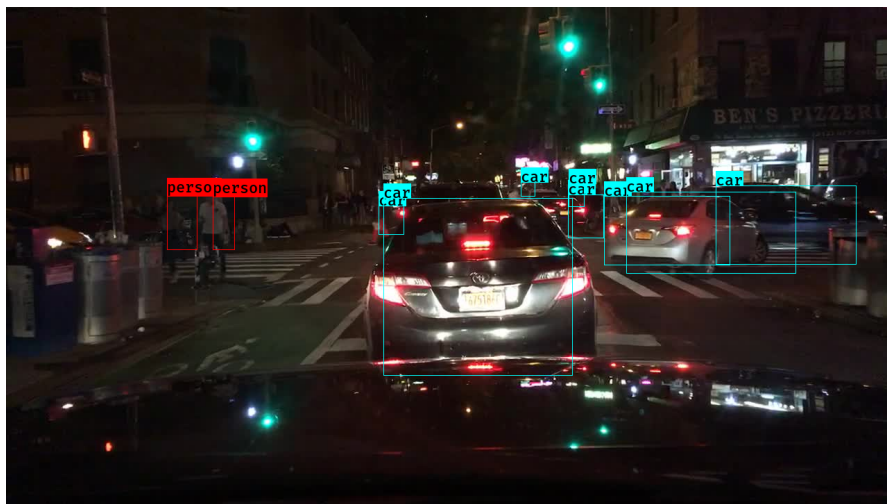


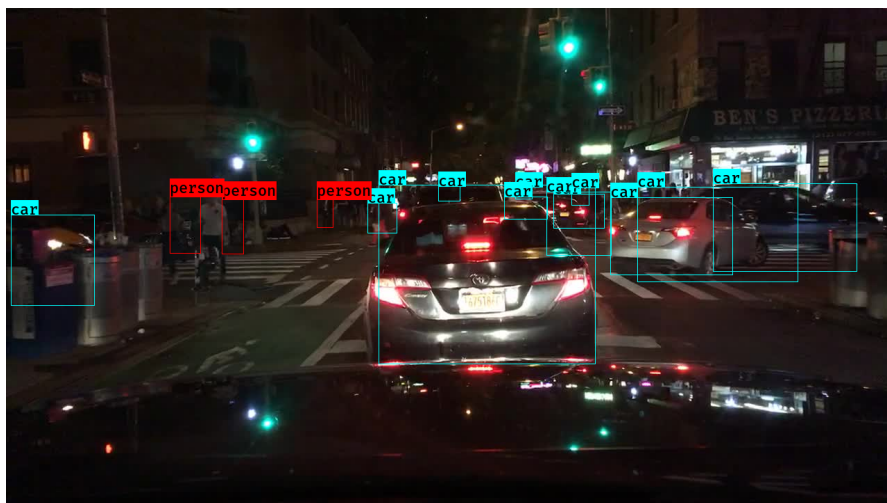
FIGURE 5.11: Experiment 4 ground truth.



(a) Ensemble Detector.



(b) Model trained on images with more than 3 instances of trucks and more than 3 instances of cars.



(c) 20k baseline.

FIGURE 5.12: Experiment 4 drawn predictions.

5.5 Wrap-up

Experiments 1 through 4 show that the number, and size, of enabled classes in the predictions matter. Certain classes are harder to get right than others, an observation supported by section 2.1.2. Using a lower number of enabled classes, and excluding certain objects known to be challenging, like traffic lights and signs, helps improve the performance. The experiments show that the quality of the training of the weak models greatly affects their performance, as well as the performance of the ensemble. Doubling the training set size from 500 to 1000 images, and increasing the number of epochs from 5 to 15, resulted in an average **mAP** value nearly 3 times as high, demonstrating the importance of data in these models. Alas, increasing the number of weak models did not improve the **mAP** considerably, nor was the trend monotonically increasing. Described in detail in experiment 3, this is primarily caused by *standalone* predictions becoming more abundant as the number of models in the ensemble increases. The standalone predictions do not overlap with any other predictions, and are therefore not involved in the **WMV**. They cannot be removed altogether, because many of them are **True Positive (TP)** predictions. Unfortunately some are **False Positive (FP)**s, with a negative effect on the **mAP** score of the ensemble. Sections 7.1.2 and 7.3 in chapter 7 suggest improvements that can be made to alleviate some of these problems.

Chapter 6

Context-Sensitive Detector experiments and results

6.1 Main purpose and metrics

Structured similarly to chapter 5, the purpose of this experiment chapter is to answer the third [Research Question \(RQ\)](#), described in section 1.2. To answer the [RQ](#) the [Context-Sensitive Detector \(CSD\)](#) is run, varying the hyper-parameters:

- Training quality, as explained in section 5.1.
- Class distribution. In order to simulate driving between different locations, multiple distributions of different classes are chosen, described within each experiment.
- Window Size, τ . Shown in figure 4.5, the windows size provides a *leading edge* in which the rewards are considered more important than in the trailing edge. Increasing the window size will increase the memory of the [Sliding-Window UCB \(SW-UCB\)](#) policy whilst decreasing the rate at which it switches to a new model.
- Discounting Factor, ϵ . Shown in figure 4.5, the discounting factor is used by the [SW-UCB](#) as a mechanism for punishing poorly performing models. Increasing the discounting factor should ensure faster switching between models, increasing the rate of exploration, while decreasing the rate of exploitation.

6.2 Experiment setup

The experiments consist of 4 components:

- A group of models trained on images from different distributions, for a set number of epochs. These *weak* models form the foundation for the CSD's decision process. For each experiment the values for each parameter is mentioned in the setup. In the resulting plots the different models are named based on their bias and the size of the training size. For example, the model *4truck1k* is trained on 1000 images with 4 (or more) trucks in them.
- A dataset S , containing N sequences of k images ($S_{k1}, S_{k2}, \dots, S_{kN}$).
- A CSD policy metric (SW-UCB) with hyper-parameters, explained in section 4.2.1. This is used to determine the currently best performing model, based on prior performance.
- A baseline model trained on 20 000 images from the same distribution as the models in the CSD.

6.3 Experiment plots

In order to make it easier to differentiate between the policy metrics and the performance of the CSD, the plots in the following experiments are split in two: one plot for the policy reward, and one for the F_1 -score, explained in section 4.2.1 and 3.3.2, respectively.

The x-axis shows the image number in the distribution sequence, and functions as a time indicator. If a person drives from the city (location A), through the countryside (location B), and back to the city again, the x-axis presents their progress throughout that drive.

For each model, the F_1 -score is plotted, and the overall performance of the CSD can be found by looking at the uppermost plot for any given x-value. Plotting the models in this way yields a better understanding of what is going on *behind the scenes*, and illustrates the selection process of the CSD.

Along with the models, the baseline plot for the 20k model is plotted. This illustrates how the [CSD](#), consisting of multiple *weak* models, performs, compared to a *strong* model.

6.4 Experiments

6.4.1 Experiment 1: 2 models, easily discoverable context

Enabled classes: Car, truck, person.

Setup Experiment 1 is set up as a baseline comparison experiment, using the best hyper-parameters found in preliminary experiments. This is done to provide an intuition for how the [CSD](#) approach works, and how the other experiments differ.

The distributed dataset is defined as:

$$S_{500truck} \rightarrow S_{1000person} \rightarrow S_{500truck} \quad (6.1)$$

Furthermore, the 2 models used in this experiment are trained on 1000 images over 15 epochs with a bias of 4 or more trucks, and 8 or more pedestrians. A window size (τ) of 20, and a discounting factor (ε) of 0.5 was used.

Hypothesis The baseline model was expected to be superior to the [CSD](#), simply due to the different training set sizes (20 000 images compared to 2×1000 images). However, the [CSD](#) was expected to detect the switching between the different contexts and select the best performing model accordingly. As such the hypothesis was that the *4truck1k* model would be used for predictions on the first 500 images, the *8person1k* model would be used for the following 1000 images, and the *4truck1k* model would again be used for the last 500 images.

Results A plot of the resulting performance can be seen in figure [6.1](#). It demonstrates an ideal scenario where the detector is able to easily differentiate between the performance of the truck biased model, and the person biased model. This is shown in

the rapid detection (and subsequent switching of models) of the underlying context seen around both the 500 and the 1500 mark.

A thing to note (which is also apparent in all the following experiments) is the *cold start* problem observed in the CSD. Seen in the first 20-30 points along the x-axis, the CSD struggles to determine the best performer in the beginning, due to the lack of historical data. This causes a small drop in performance, compared to the baseline model.

Around the 250 mark there is a sudden dip in the performance of the *4truck1k* model, and after using the *8person1k* model for a number of images it becomes evident that the first classifier is still the best performer, and the CSD switches back to this. This can be seen in both the policy reward and the following F_1 -score.

Also evident is the slight delay in the switching around both the 500 and the 1500 mark. The switching seems to occur about 20 units after the introduction of the new distribution, which could stem from using a window size (τ) of 20. The effect of changing the window size is explored more in experiment 4.

Perhaps the most surprising discovery is the difference between the CSD performance and the 20k baseline. Table 6.1 shows the overall mAP score for experiment 1. Comparatively the score for the 20k baseline on the same dataset is 20.54%. The majority of the performance difference stems from the switching between the models, shown in figure 6.1.

mAP [%]		τ
		20
ε	0.5	18.94

TABLE 6.1: mAP results for experiment 1

6.4.2 Experiment 2: 8 models, challenging class selection

Enabled classes: Bus, traffic light, traffic sign, person, bike, truck, motor, car, train, rider (all).

Setup After seeing the performance of the CSD on two clearly distinguishable models, this experiment attempts to introduce a more challenging scenario. Separate models are trained on datasets biased towards each class. No models trained solely on cars were

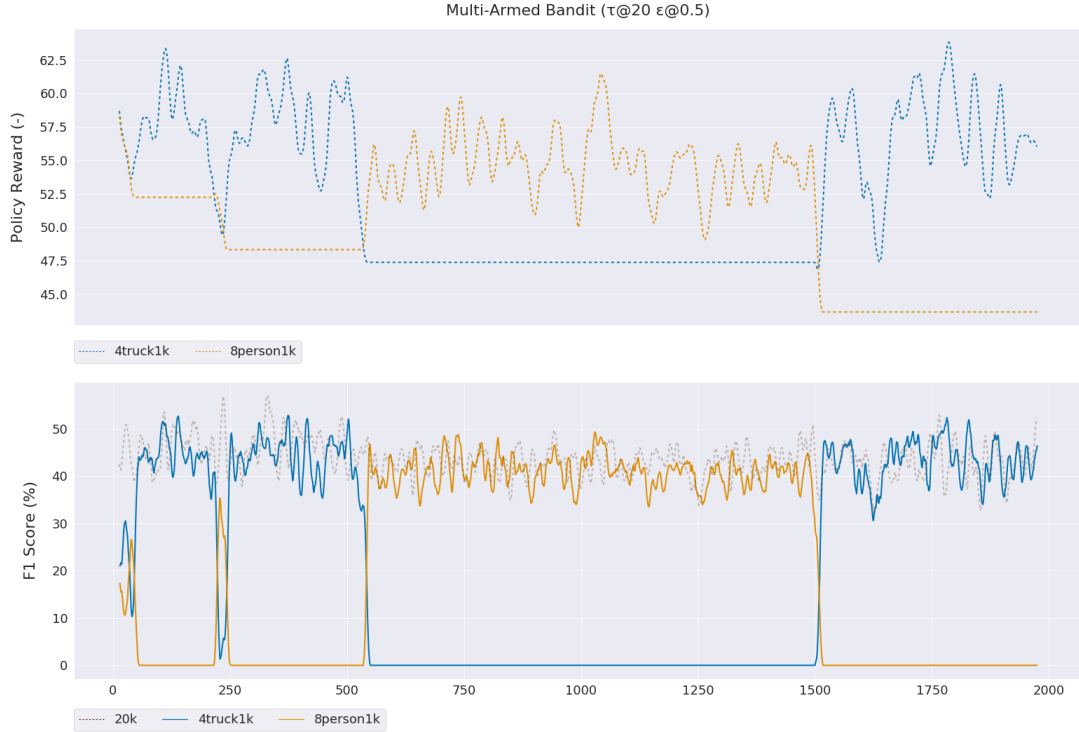


FIGURE 6.1: Experiment 1 Policy & Reward

included, to avoid any one model trying to *overpower* the others. This is due to the fact that most images in the [BDD](#) dataset contain one or more cars [36].

The dataset, now containing all classes, consists of:

$$\begin{aligned}
 S_{500bike} \rightarrow S_{500bus} \rightarrow S_{500motorcycle} \rightarrow S_{500sign} \rightarrow \\
 S_{500light} \rightarrow S_{100train} \rightarrow S_{500truck} \rightarrow S_{500person}
 \end{aligned}
 \tag{6.2}$$

Hypothesis With the introduction of more classes, some of them being inherently difficult to detect by the models, as explained in section 2.1, it is reasonable to assume that the [CSD](#) will have a harder time discovering the bias of each model. Using the F_1 score as the performance metric exposes the danger of models like the *12sign1k* or *12light1k* overpowering other models based on the number of objects they are able to predict.

Compared to the 20k baseline, the [CSD](#) in this experiment is expected to perform rather poorly due to the increased complexity in the distribution.

Results With the introduction of smaller objects like *traffic lights* and *traffic signs*, it is evident that both the 20k baseline model and the CSD show significant *drops* in performance. The F_1 -scores, on average, range from 25 to 35, compared to 40 to 50 in the previous experiment. As explained in section 2.1 this is partly caused by the YOLO network’s poor performance when it comes to detecting small objects. As such, part of the overall performance drop might stem from this. This becomes apparent when comparing the CSD performance, shown in table 6.2, to the baseline performance of $\sim 7.5\%$.

mAP [%]		τ
		20
ε	0.5	5.32

TABLE 6.2: mAP results for experiment 2

Looking at the first 500 images there are some interesting observations to be made. The distribution here contains images with more than 2 bicycles. These objects are often small and difficult to identify, and are usually accompanied by people, traffic lights, etc. It is therefore not surprising that the CSD is having a hard time determining the best performing model. However, the first two spikes seen in figure 6.2 stem from selecting the *2bike1k* model, and the third spike from the *1motorcycle1k* model. The last spike observed on the bicycle distribution comes from the *12person1k* model, which again stands to reason, as most bicycles are accompanied by a person, as seen in figure 6.3.

From the 500 to the 1500 mark, there are three spikes, stemming from the predictions from 3 different models: *12light1k*, *2bus1k*, and *1motorcycle1k*. The underlying distributions are shown in equation 6.3. Interestingly, the CSD discovers the correct models for the sequence of images, but in the wrong order. A plausible explanation for this is that the content of the distributions and the images the 3 models have been trained on, contain many similar objects. Figure 6.4 shows this to some extent, where the model is able to predict the motorcycle and rider (two objects), and 4 lights (where two of them are probable correct predictions).

$$S_{500bus} \rightarrow S_{500motorcycle} \rightarrow S_{500sign} \quad (6.3)$$

A notable concern with this experiment is that the *12sign1k* model seems to become dominant in the latter half of the distribution, even though the underlying distributions here, consist of:

$$S_{500sign} \rightarrow S_{500light} \rightarrow S_{100train} \rightarrow S_{500truck} \rightarrow S_{500person} \quad (6.4)$$

A likely cause of this is that most of the above distributions contain multiple objects, and that the F_1 -score does not separate between classes, but simply uses the true and false positives as its underlying metrics. From the distributions it is also true that images containing signs, lights and people (in particular) often contain multiple instances of that type (eg. 5 signs or 10 people).

However, the predictions throughout the experiment seem to follow the baseline dataset decently, only lagging 4 – 8% behind. This is surprising, based on the complexity of the distribution. One important takeaway from this experiment is that the different objects often appear in clusters (riders, motorcycles, lights, etc.), making it difficult for the CSD to accurately discover the underlying object distribution.



FIGURE 6.2: Experiment 2 Policy & Reward

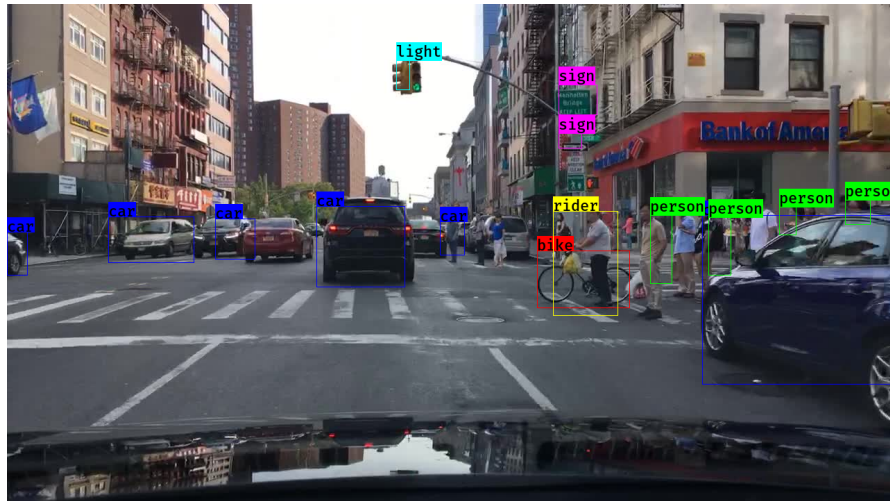


FIGURE 6.3: 2bike1k observation, as selected by the CSD

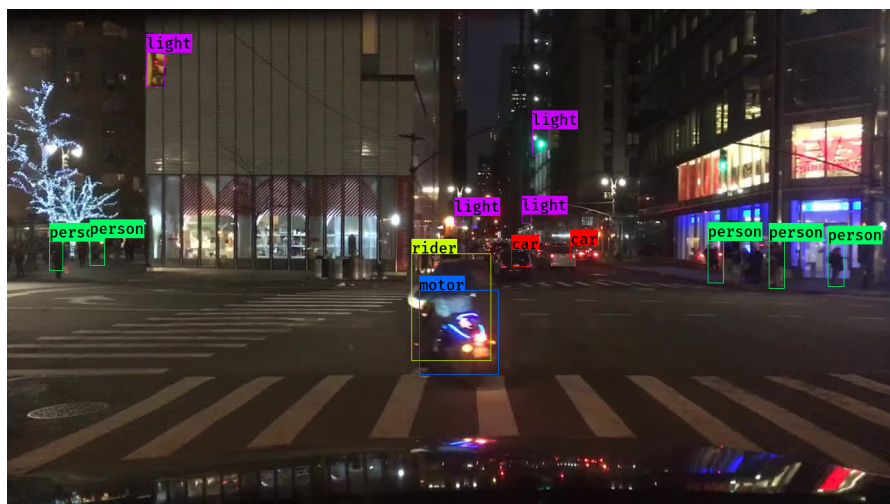


FIGURE 6.4: 1motorcycle1k observation, as selected by the CSD

6.4.3 Experiment 3: 4 models, additional models available

Enabled classes: Car, truck, person.

Setup Following the results of experiment 2 this experiment reduces the number of enabled classes, back to the same distribution found in experiment 1. This is done in order to assess the CSD's ability to achieve similar predictions to those found in experiment 1, using a wider selection of models.

The dataset distribution is equal to that of experiment 1, found in equation 6.1. In order to maintain the same number of available images for the underlying models, and

assert the [Upper Confidence Bound \(UCB\)](#)'s accuracy when the individual models' performance decreases, the models are trained on 500 images. The available models are: *4truck500*, *3truck500*, *10person500*, and *6person500*.

Hypothesis By increasing the number of available models, the [CSD](#) should be able to rapidly detect the best performing model, similar to what was seen in experiment 1. However, by introducing more models as a *fallback*, the [CSD](#) should be able to avoid the unfortunate switching that occurs if the model experiences a brief dip in performance. The goal is to avoid the effects of these events, like the one appearing around the 250 mark in experiment 1.

Results From the results seen in figure 6.5 it is evident that increasing the number of available models achieves a performance similar to that of experiment 1. When comparing the mAP in table 6.3 to the mAP from experiment 1 (table 6.1) and the mAP of the 20k model of 20.54%, it is apparent that the increase in the number of available models introduces a bit of noise. Most notable is the noise in the beginning of the predictions where the [CSD](#) has to establish a base performance for multiple models.

mAP [%]		τ
		20
ε	0.5	14.95

TABLE 6.3: mAP results for experiment 3

The switching phases visible around both the 500 and the 1500 mark in figure 6.5, do not seem to cause any notable performance hits. This showcases the [CSD](#)'s ability to accurately select well performing models. However, as the *10person500* and *6person500* models both perform quite well on pedestrian predictions, the [CSD](#) incidentally selects the *6person500* model and sticks with it. This is unfortunate as it is likely that the performance of the *10person500* model would be superior due to its heavier bias.

Having two *truck* classifiers in the model pool of the [CSD](#) causes switching in both of the truck distributions (the 500 images at the beginning and the 500 images at the end). As both models perform rather well on truck images, the sudden drop in performance from one causes the [CSD](#) to switch to the other truck classifier, and the noise at both the 250 and the 1700 mark causes some performance loss in the overall [mAP](#).

Lastly, part of the performance drop stems from the reduced performance of the individual models, simply due to the fact that they are trained on smaller datasets (500 images, compared to 1000 images in experiment 1). However, by halving the training sets, the CSD sees a drop in performance of only 21%, emphasizing the CSDs ability to utilize the strengths of individually weak models.



FIGURE 6.5: Experiment 3 Policy & Reward

6.4.4 Experiment 4: 2 models, changing the window size

Enabled classes: Car, truck, person.

Setup After testing different CSD configurations, the following 2 experiments look at the effect of changing the hyper-parameters (τ and ε). Using the same models and dataset distribution found in experiment 1, experiment 4 looks at both increasing and decreasing the window size of the Sliding-Window UCB (SW-UCB) policy. By doing so the goal is to establish how the change in window size affects the CSD's performance.

In order to assess how both widening and narrowing the window size affects performance, the experiment is run with a window size of 200 (10x baseline) and 5 (1/4th of the baseline).

Hypothesis By increasing the window size, the **CSD** is expected to retain performance scores longer, leading to slower switching between different models, and perhaps a more stable model selection.

Contrarily, the narrower window should cause faster switching and potentially more noisy performance. The narrower window should also increase the frequency at which the **CSD** explores new models and picks up previously under-performing models.

Results Table 6.4 shows the mAP for the two experiments. There is a rather steep drop in performance, compared to both experiment 1 and experiment 3, which were run on the same image distributions.

mAP [%]		τ	
		5	200
ε	0.5	4.84	8.44

TABLE 6.4: mAP results for experiment 4

Reducing the window size, the resulting plot seen in figure 6.6, causes the **CSD**'s *memory* to vanish quicker, causing abrupt switching when consecutive performance drops occur. This results in noisy performance on the first 500 images, but most importantly, it causes a drop in the policy score of the *4truck1k* model, preventing the model from being picked up in the last 500 images. Evident around the 1500 mark, the *8person1k* model sees a drop in performance, but due to the narrow window the drop is not enough to cause the **CSD** to explore other models.

On the other hand increasing the window size causes a significant performance boost, with a 75% higher **mAP** score compared to the window size reduction, shown in table 6.4. However, compared to the results in experiment 1 this is again rather low. Evident in the resulting plot, shown in figure 6.7, the wider window causes very noisy switching conditions, seen around the 500 mark. This stems from the **CSD** attempting to exploit the *4truck1k* model. The poor performance can be observed in the F_1 score of the *4truck1k* model around the 650, 750 and the 1000 marks.

Interestingly the switching around the 1500 mark causes little noise, largely due to the *8person1k* model's terrible performance detecting trucks.

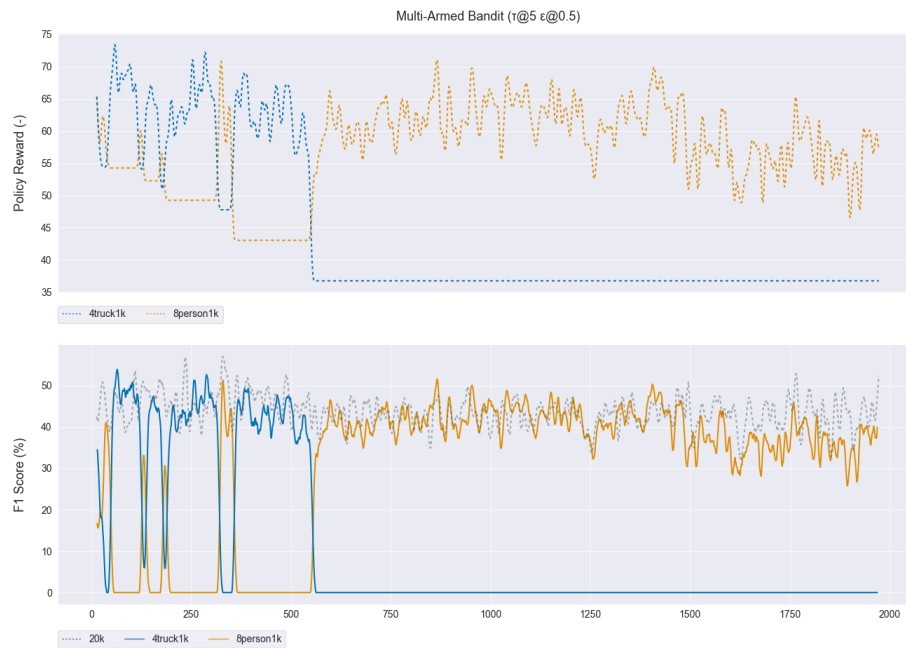


FIGURE 6.6: Experiment 4 Policy & Reward - tau 5



FIGURE 6.7: Experiment 4 Policy & Reward - tau 200

6.4.5 Experiment 5: 2 models, changing the discounting factor

Enabled classes: Car, truck, person.

Setup Similar to the test setup in experiment 4, this experiment explores how the change in the discounting factor (ϵ) affects the decisions made by the [CSD](#). The dataset distribution and models remain the same as in experiment 1, 3, and 4.

In order to test both positive and negative changes in discounting factors, the experiment is tested on ϵ values of 5 (10x baseline) and 0.05 (1/10th of the baseline).

Hypothesis By changing the discounting factor the rate at which the [CSD](#) *punishes* suddenly poor performing models, should change. Decreasing the discounting factor (to 0.05) should lead to the [CSD](#) picking up a new model after less of a decline in the policy reward, and increasing the discounting factor (to 5) should lead to a more stable model selection.

Extending on that, using a discounting factor that is too low encourages more exploration and could cause a performance loss as it discards models too quickly after observing only a few poor predictions. Subsequently, using a too high discounting factor encourages more exploitation and could similarly cause performance loss by allowing previously good performers to continue to make their *best effort* predictions.

Results Shown in figure [6.8](#), the increased discounting factor causes a drop in the policy reward from the 500 mark, to around 1100. The slightly lower policy reward makes it evident that the [CSD](#) allows the *4truck1k* to perform poorly due to its previously good performance. It is not until another performance dip that the *8person1k* model is picked up again. Apparent from the policy reward is the *8person1k* model's ability to perform truck predictions, as the switch around the 1500 mark happens almost immediately. This is due to the fact that the *4truck1k* model is decent at predicting pedestrians, but the *8person1k* is struggling when it comes to predicting trucks. This is not surprising as there are more images containing pedestrians than trucks in the [BDD](#) dataset, making it easier to train a model on images with pedestrians and without trucks, but difficult to train a model on trucks without any pedestrians.

The resulting **mAP** scores can be seen in table 6.5, reaching a similar performance to the one seen in experiment 3. Comparing this to the results from experiment 4, shown in table 6.4, emphasizes the importance of selecting the appropriate window size, in accordance with the underlying dataset.

mAP [%]		τ
		20
ε	0.05	14.25
	5	13.08

TABLE 6.5: mAP results for experiment 5

Figure 6.9 demonstrates a lower discounting factor, and the overall performance of the **CSD**. An interesting point to be made here is the drawn out switching happening around the 500 mark. This could be due to the combination of the low discounting factor, and the narrow window, making it difficult for the **CSD** to decisively pick up the *8person1k* model.

There is also a second performance anomaly around the 800 mark stemming from a couple of bad predictions made by the *8person1k* model. However, the performance of the *4truck1k* model is not as good as previously observed, and the **CSD** switches back after about 100 iterations.

6.5 Wrap-up

Experiments 1 through 5 show how the training quality, class distributions, window size, and discounting factor affect the **Context-Sensitive Detector (CSD)**. Similar to the results from the **Ensemble Detector (ED)** experiments, and supported by section 2.3, it is evident that excluding certain classes has a positive effect on the performance of the **CSD**.

The presented methodology displays robustness to the changing quality of the underlying models, with only a small performance drop observed between experiment 1 and 3. Experiments 4 and 5 showcase the importance of proper hyper-parameter tuning, and how the balancing of *exploration* and *exploitation* remains an integral problem in **Reinforcement Learning (RL)**.



FIGURE 6.8: Experiment 5 Policy & Reward - epsilon 0.05



FIGURE 6.9: Experiment 5 Policy & Reward - epsilon 5

However, observations from experiment 1 present promising results regarding the applicability of context-sensitive decision-making in [Deep Learning \(DL\)](#). The findings in this chapter also laid the foundation for an article written by the authors of the thesis, further discussed in [section 7.4](#), and presented in full in [appendix C](#).

Challenges and ideas for improvement related to the [CSD](#)-approach are further explored in [section 7.3](#).

Chapter 7

Discussion and future work

Chapter 7 provides answers to the [Research Question \(RQ\)](#)s posed in section 1.2, and presents afterthoughts on what could have been done differently, before suggesting ideas for future work.

7.1 Answers to the research questions

7.1.1 RQ1 - Decentralized Artificial Intelligence

Question: What advantages and disadvantages does [Decentralized Artificial Intelligence \(DzAI\)](#) bring to autonomous driving?

Described in chapter 4, there are multiple approaches to [Decentralized Artificial Intelligence \(DzAI\)](#). The applicability and viability of these approaches are discussed in depth in their respective sections (sec. 7.1.2 and 7.1.3).

The ongoing race towards fully autonomous vehicles is driven by large technology companies like *Tesla*, and *Waymo*. At the core of their dominance is their access to large amounts of image and sensor data, having driven millions of miles. In the [Deep Learning \(DL\)](#) field, the more data companies are able to gather, the more likely they are to prevail. The methods proposed in this thesis present alternative approaches to the acquisition and treatment of image data related to autonomous driving, through the use of a [Decentralized Autonomous Driving \(DzAD\)](#) network. The presented [DzAD](#) network makes more models, and by extension data, available. Not only does the amount of data

available increase, a decentralized network also makes it possible for drivers all over the world to participate, resulting in more *diverse* data. Finally, the increased availability of data can help smaller companies, researchers and other participants with limited resources join the race.

Privacy is one of the other major benefits achieved through the [DzAD](#) network. When training is performed locally (on the edge), without the data leaving the source, and only the models are transmitted over the network, little to no residual information can be traced back to the source. By not being asked to directly share private information related to their driving habits, the willingness of the public to participate increases, supported by the results of the survey presented in section [2.4](#).

However, a drawback of training models on small, distributed datasets is the loss of accuracy. Due to the potential implications of an [AI](#) making bad decisions while controlling a vehicle, this is of the utmost importance. The benefits of increased privacy are easily negated by the potential loss of lives, injuries and harm to infrastructure. Section [7.3](#) proposes solutions to some of these problems, and how to make [DzAI](#) viable in real world applications.

7.1.2 RQ2 - Ensemble Detector

Question: Can a collection of weak [Deep Learning](#) ([DL](#)) object detectors be combined to perform as well as, or better than, a strong [DL](#) object detector?

Weak models can be combined in many ways to increase their collective performance. Chapter [4.1.2](#) describes an [Ensemble Detector](#) ([ED](#)) as one possible technique for combining a collection of models. The [ED](#) fuses the strengths of multiple models by performing [Weighted Majority Voting](#) ([WMV](#)) on a per-object basis. Chapter [5](#) describes 4 experiments that explore the performance of such a technique in different scenarios, and whether parameters can be tuned to achieve performance similar to that of a centralized approach.

The results show that the centralized approach still outperforms the decentralized approach in terms of [mean Average Precision](#) ([mAP](#)). The [ED](#) fails to achieve the same level of accuracy as the baseline model. Additionally it was evident that the quality of the training of the weak models had a great effect on their, and by extension the [ED](#)'s,

performance. **Decentralized Autonomous Driving (DzAD)** revolves around using small amounts of data from many different sources, which limits the training of the models. Based on the results from the survey described in section 2.4, the average person is hesitant to spend a lot of time to annotate and label images captured while driving. Experiment 3 in section 5.4.3 indicates that a model needs to be trained on at least 1000 images to be able to perform decently. Based on the payment expected by the participants (sec. 2.4), it is unlikely that enough participants will be willing to annotate enough training images for a feasible price. The price is important because there are several existing, centralized solutions for collecting annotated data for **Computer Vision (CV)**, like **Mighty AI** [57] or **Amazon Mechanical Turk** [58]. Both of these services use real people to annotate images for use in **CV**.

The experiments show that the **ED** is unable to compete with the baseline model on performance, and the price people are willing to accept for annotating images is rather high, based on the survey. Thus the conclusion is that a **DzAD** solution using the **ED** is not suitable to compete with centralized solutions currently available, at least without additional experimentation and tuning.

However, a decentralized approach brings other advantages, and could be utilized in a hybrid approach with the current centralized solutions. While the datasets that participants in the **DzAD** network are expected to provide are too small to be able to train accurate models, they can be used in order to create more diverse models. The vast amount of training data gathered for self-driving cars today is gathered in the **United States (US)** [36]. Traffic in a country like Norway is very different, in terms of landscape, road design, lane structure, sign layouts etc. A hybrid approach could be developed where diverse data is gathered from cars all over the world, annotated and labeled by services like **Mighty AI** and **Amazon Mechanical Turk**, and used to train centralized models.

7.1.3 RQ3 - Context-Sensitive Detector

Question: Can detection and exploitation of bias in weak **Deep Learning (DL)** object detectors, in a context-dependent environment, outperform strong context-agnostic **DL** object detectors?

The results from the experiments in chapter 6 show the usefulness of being able to select the best performing [Machine Learning \(ML\)](#) models, based on context. However, as demonstrated by the experiments, the degree of usefulness in terms of performance, is highly dependent on the underlying context and hyper-parameter tuning.

The results from experiment 1, which shows the [CSD](#) functioning optimally, show how weaker models can be combined in order to reach performance on-par with, and sometimes even better than, the 20k baseline model. However, the overall performance loss experienced in the switching phases makes the overall performance slightly worse than that of the baseline model. This is even more evident when the underlying context is hard to distinguish, as seen in experiment 2.

Context is very important in human decision-making. Subconscious decisions are constantly made, based on the context the decision is made in. The [CSD](#)-approach mixes computers' and humans' decision making processes. The method displays potential in making context dependent decisions using local experts, rather than striving for a global expert.

The [CSD](#)-approach shows potential, and ideas for improvements are introduced and discussed in section 7.3. Even with little effort in optimizing for performance, the [CSD](#)-approach achieves speed comparable to that of the [You Only Look Once \(YOLO\)](#) network, running at about 15 [FPS](#) on a Pascal GTX 1080 [GPU](#). However, there are some caveats, such as the importance of selecting sufficiently distinguishable features in order to make the different distributions discoverable.

7.2 Challenges

A repeated topic throughout the thesis is the [YOLO](#) network's struggle to detect smaller objects. Many of the images in the [Berkeley DeepDrive \(BDD\)](#) dataset contain complex collections of objects, as depicted in figure 7.1. Some of the objects are inherently small and difficult to detect, like *signs* and *traffic lights*. Other objects are far away, appearing small, making abstractions very hard for the model. For instance, the car in the far distance in the middle of the image in figure 7.1 is rather similar to a traffic light.



FIGURE 7.1: Ground truth for an image from the BDD dataset

The true ramifications of this fact were first discovered at the experimentation stage, at which point adjustments were made, as explained in the experiment chapters. Ideally this would have been caught earlier, allowing for the selection of a different, simpler dataset, or a more appropriate CNN architecture. This would likely yield better overall mAP scores throughout the experiments.

A different approach to this could be to rate or filter the objects based on a distance metric. By removing, or reducing the importance of, objects more than a set distance away (perhaps based on the current speed of the car), the mAP of the models is hypothesized to improve significantly.

At the same time, the challenges accurately depict the complexity of the task. Real-time object detection remains the focal point for a lot of large tech companies, signifying its intricacies.

7.3 Future work

Following the answers to the research questions and the subsequent challenges, it is evident that the methodologies show promise, however there is room for improvement. The following sections examine different ideas for the continued exploration of the approaches proposed in this thesis.

7.3.1 Complete network training

The [Convolutional Neural Network \(CNN\)](#)s used in this thesis use pre-trained weights and transfer learning to only tune the weights of the last 2 layers of the network. This reduces the training time significantly, enabling training of a large number of models with different parameters and configurations, facilitating rapid prototyping. The models served their purpose perfectly by allowing many experiments to be performed, revealing trends and guiding future experiments. Training the entire network from scratch could have increased the performance of the models, but requires more time and resources allocated to the training process.

7.3.2 Reinforcement Learning

Building on the promising results from the [CSD](#) experiments, a more dynamic approach to hyper-parameter tuning could be a promising step forward. Building a [Deep Reinforcement Learning \(DRL\)](#) network [59] can provide additional flexibility to the [CSD](#). Using a modern approach to Q-learning, Deep Double Q-learning [60] can enable a more dynamic rate of adaptation, by allowing the network to learn a new model selection policy, rather than using a hyper-parameter tuned, pre-defined policy.

7.3.3 Distribution selection

The experiments run in chapter 6 make it clear that there is a significant performance difference between the easily distinguishable objects (experiment 1), and the not so easily distinguishable ones (experiment 2). Not only does the [mAP](#) score overall take a solid hit, but the model selection performed by the [CSD](#) becomes poorer. As such the use of sufficiently distinguishable contextual identifiers cannot be understated.

Because of this, the exploration of different labels as the distinguishable factor is proposed as future research. Rather than using the classes directly as the deciding contextual metric, a proposed solution would be the use of meta-information, also present in the [BDD](#) dataset: *weather*, *scene*, and *time of day*. This could prove to be a better factor for determining the contextual performance of the underlying models. They would then be trained with this in mind, with models specialized for snowy, windy, and foggy conditions, to name a few.

All the dataset’s available attributes can be seen in appendix B.

7.3.4 A combined approach

The experiments in chapter 5 show the potential in combining multiple models in order to improve their collective performance. Chapter 6 on the other hand, shows the potential in using multiple, specialized models in order to improve performance by allowing context-aware overfitting, and selecting models accordingly.

By combining the two approaches, creating *one* model that is able to generalize well over most driving scenarios, and using a CSD-approach to perform context-based model selection, the performance loss experienced while switching between models could be reduced. An approach like this could allow even more specialized models, potentially increasing performance.

7.3.5 Alternative ensemble learning methods

The **Ensemble Detector (ED)** used in the thesis uses **Weighted Majority Voting (WMV)** to combine the predictions of the models in the ensemble. Described in more detail in section 4.1.3, the detector performs **WMV** on each predicted object in the image, using the *confidence scores* as the weights. To alleviate some of the issues discovered in the experiments, alternative approaches can be explored. In their paper on *R-FCN Object Detection Ensemble based on Object Resolution and Image Quality*, Rasmussen, Nasrollahi and Moeslund describe a way to perform weighted averaging where the weights are distributed evenly across 5 different types of factors [61]. Examples of factors used in Rasmussen’s paper include object color, texture, shape and size. For the task at hand in this thesis, factors could include object shape and size to potentially mitigate some of the problems involved with detecting distant and/or small objects.

Another technique to try is the regular **Weighted Majority Algorithm (WMA)**, where each model is given an initial weight (or 1 per class), and the ensemble is trained on a training set where the models’ weights are decreased when they contribute to an incorrect ensemble prediction. A challenge with this approach is that it is not trivial to design a metric for deciding how much one model’s weight should be reduced compared

to another's, when the task is object detection rather than classification. Additionally, experiments in [12] show little benefit of this approach.

Finally, performing complete network training as described in section 7.3.1 is another option that is likely to boost the performance of any ensemble approach.

7.4 Article

Based on the promising results from the experiments in chapter 6, an article was written. It presents the usage of a [Context-Sensitive Detector \(CSD\)](#)-approach to perform contextual model selection in [Computer Vision \(CV\)](#), and can be found in its entirety in appendix C.

The article was submitted to the ICoIAS 2019 conference [62] on January 5th, and was accepted for publication.

Appendix A

Survey

A.1 Survey questions

1. Are you familiar with the concept of self-driving cars?
2. For the right compensation, would you be open to mounting a (free) camera on your car, that takes pictures at regular intervals and anonymously shares them with self-driving car companies?
3. For which price, per hour of driving, would you be willing to perform this service?
4. These pictures are more valuable if they are labeled. For which price, per picture, would you be willing to label the pictures taken by the camera on your car? Assume that labeling one picture takes between 1 and 2 minutes.
5. In light of your answers to the questions above, how much time a week would you be willing to spend labeling the pictures? Assume that you have access to an easy-to-use labeling tool.
6. Even though we claim that the data will be anonymous, would you be worried that it can be linked to you personally?
7. Would you be worried that by sharing your data, you could risk getting punished for driving too fast, driving illegal routes etc.?
8. How old are you?

A.2 Survey results

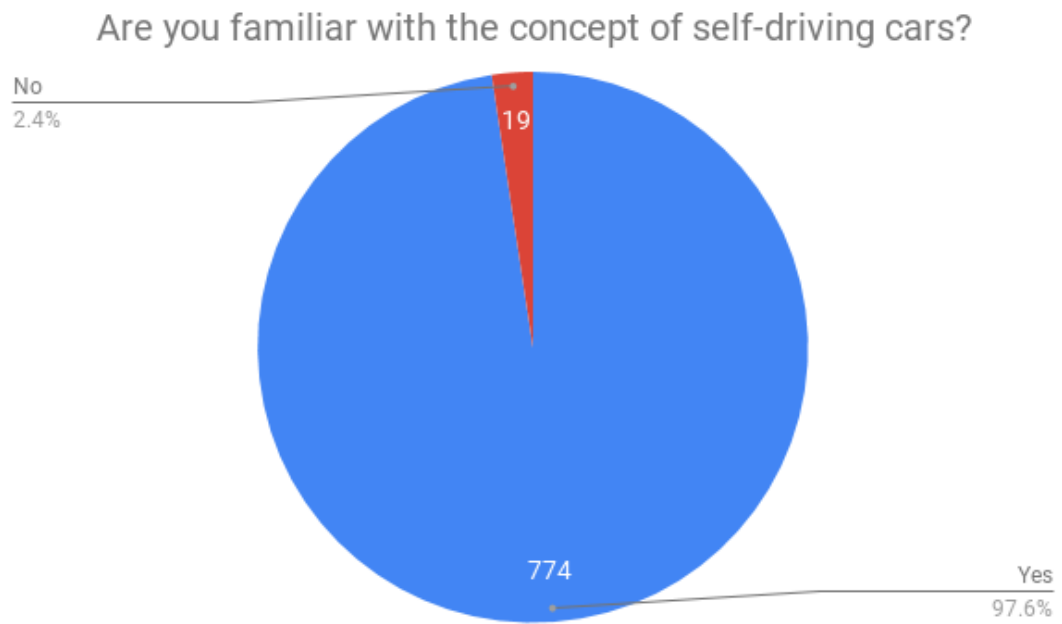


FIGURE A.1: Question 1.

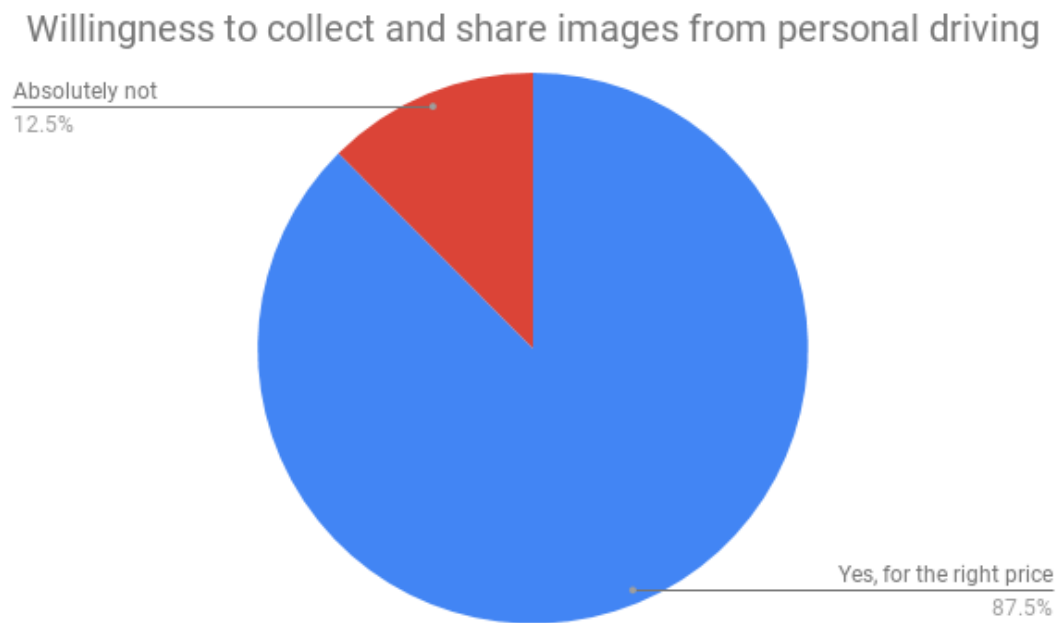


FIGURE A.2: Question 2.

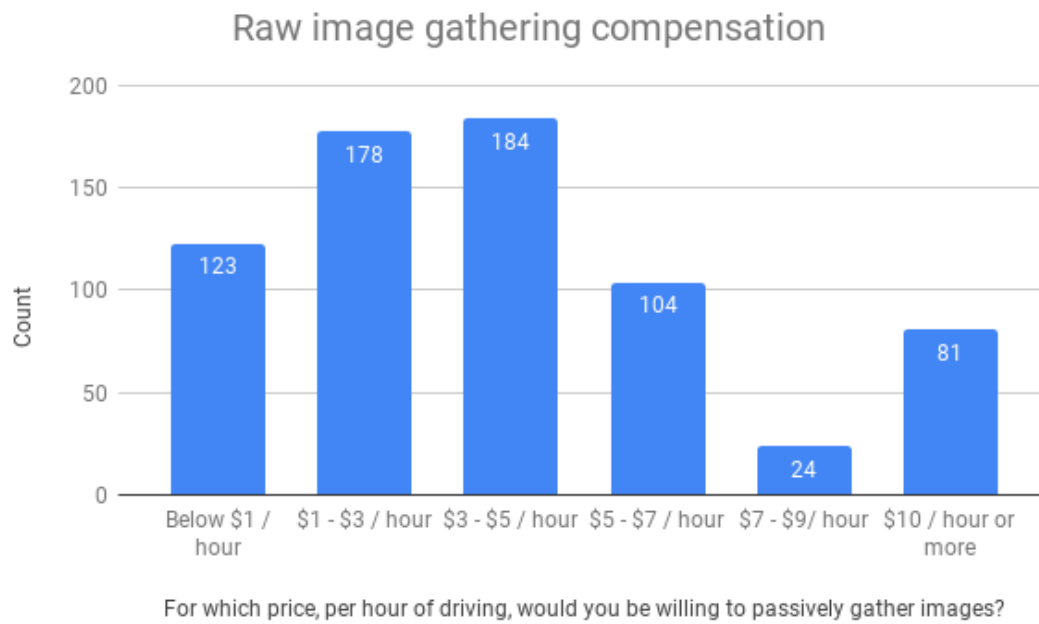


FIGURE A.3: Question 3.



FIGURE A.4: Question 4.

How much time a week would you be willing to spend labeling the pictures?

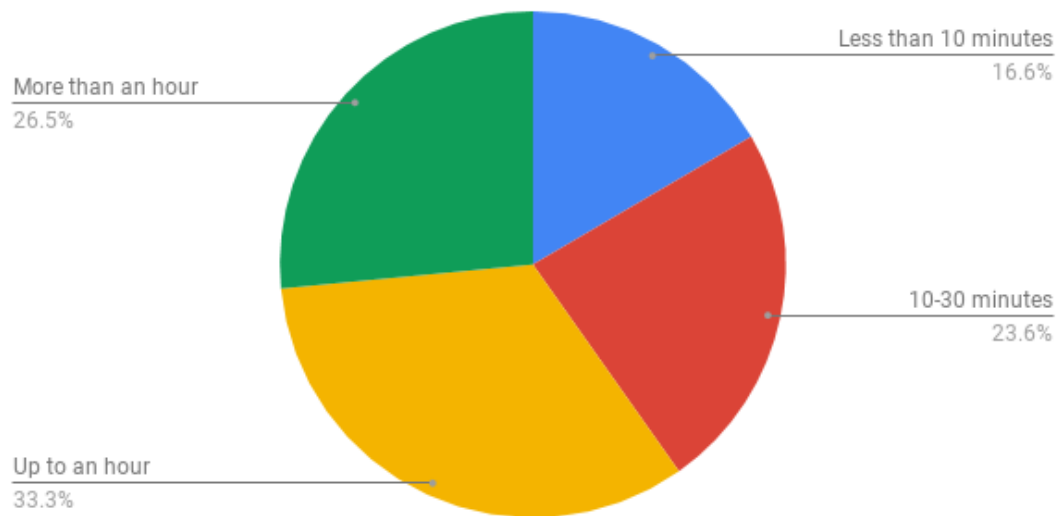


FIGURE A.5: Question 5.

Even though we claim that the data will be anonymous, would you be worried that it can be linked to you personally?

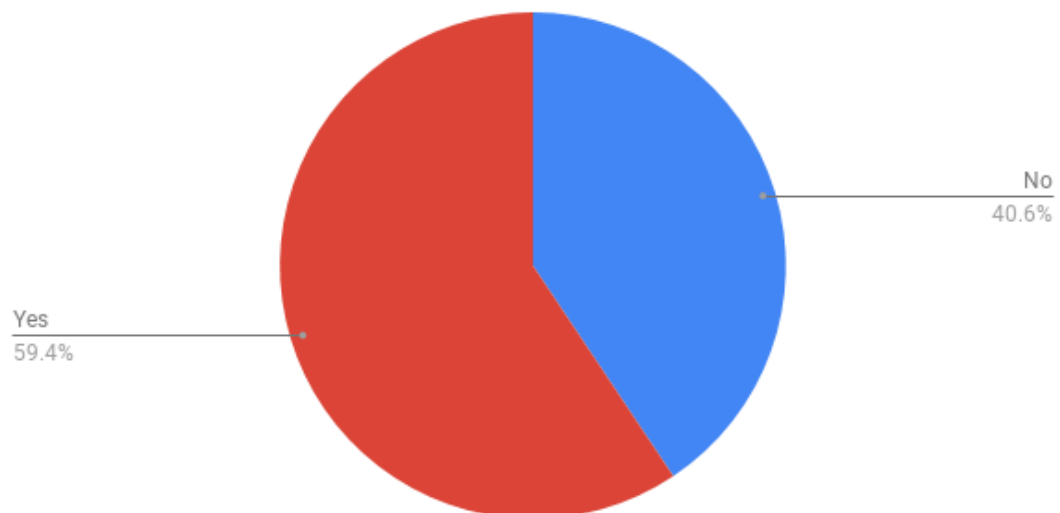


FIGURE A.6: Question 6.

Would you be worried that by sharing your data, you could risk getting punished for driving too fast, driving illegal routes etc.?

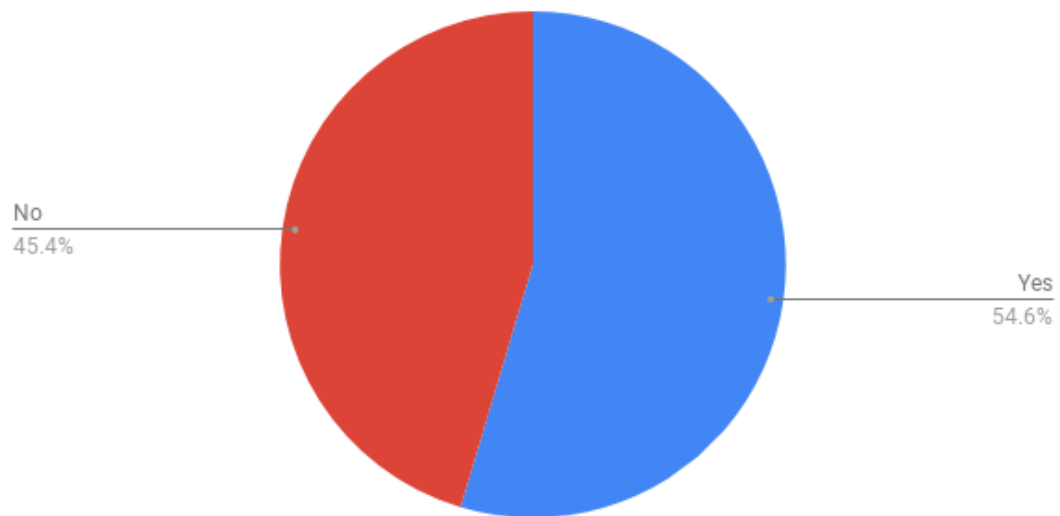


FIGURE A.7: Question 7.

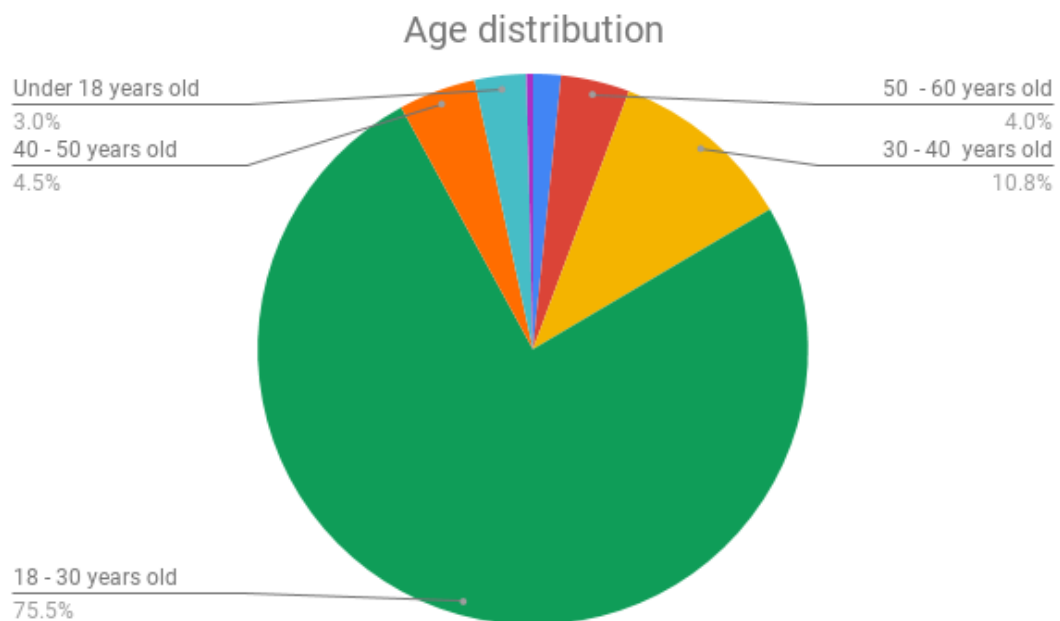


FIGURE A.8: Question 8.

Appendix B

Berkeley DeepDrive

B.1 Label format

- name: string
- url: string
- videoName: string (optional)
- attributes:
 - weather: "rainy|snowy|clear|overcast|undefined|partly cloudy|foggy"
 - scene: "tunnel|residential|parking lot|undefined|city street|gas stations|highway|"
 - timeofday: "daytime|night|dawn/dusk|undefined"
- intrinsics
 - focal: [x, y]
 - center: [x, y]
 - nearClip:
- extrinsics
 - location
 - rotation
- timestamp: int64 (epoch time ms)
- index: int (optional, frame index in this video)
- labels []:
 - id: int32
 - category: string (classification)

- manualShape: boolean (whether the shape of the label is created or modified manually)
- manualAttributes: boolean (whether the attribute of the label is created or modified manually)
- attributes:
 - occluded: boolean
 - truncated: boolean
 - trafficLightColor: "red|green|yellow|none"
 - areaType: "direct | alternative" (for driving area)
 - laneDirection: "parallel|vertical" (for lanes)
 - laneStyle: "solid | dashed" (for lanes)
 - laneTypes: (for lanes)
- box2d:
 - x1: float
 - y1: float
 - x2: float
 - y2: float
- box3d:
 - alpha: (observation angle if there is a 2D view)
 - orientation: (3D orientation of the bounding box, used for 3D point cloud annotation)
 - location: (3D point, x, y, z, center of the box)
 - dimension: (3D point, height, width, length)
- poly2d: an array of objects, with the structure
 - vertices: [][]float (list of 2-tuples [x, y])
 - types: string (each character corresponds to the type of the vertex with the same index in vertices. 'L' for vertex and 'C' for control point of a bezier curve.
 - closed: boolean (closed for polygon and otherwise for path)

Appendix C

Article

The following article was submitted to the ICoIAS 2019 conference [[62](#)] on January 5th, and was accepted for publication.

A Novel Decentralized Approach to Autonomous Driving

Marius Maaland
Department of Computer Science
Norwegian University of
Technology and Science (NTNU)
Trondheim, Norway
mariuama@stud.ntnu.no

Hai Thanh Nguyen
Department of Computer Science
Norwegian University of
Technology and Science (NTNU)
Trondheim, Norway
hai.nguyen@ntnu.no

Anders K. Kirkeby
Department of Computer Science
Norwegian University of
Technology and Science (NTNU)
Trondheim, Norway
andeki@stud.ntnu.no

Pinar Öztürk
Department of Computer Science
Norwegian University of
Technology and Science (NTNU)
Trondheim, Norway
pinar@ntnu.no

Abstract—Autonomous vehicles are improving at a rapid pace, caused by numerous technology companies joining the race. However, current approaches rely on centrally trained models, which has some limitations. Gathering diverse data from different areas like urban cities or rural towns, ideally from different countries, is expensive and difficult, even for large tech companies like Google and Tesla. This results in large amounts of training data for self-driving cars being recorded primarily in sunny climates on the United States’ west coast, on wide, multi-lane roads, with a specific sign scheme. However, with advances in the field of Internet of Things, an increasing rate of diverse, distributed data is available. Paired with the implementation of the European Union’s General Data Protection Regulation granting people ownership of their own data, and the right to share it as they like, a decentralized approach can make more diverse training data available to train models for autonomous vehicles. This paper proposes a new paradigm for how data is gathered for object detectors, and how they are distributed and applied in self-driving cars. We describe a decentralized network where the edges of the network are used to gather data from individual participants, and also to train the models locally. The models will be biased based on the context they were trained with, like urban areas with many pedestrians, or rural areas with many truck. An autonomous vehicle on the network can choose the best detector for the current context it find itself in, using a context-aware reinforcement learning approach named a *Context-Sensitive Detector*. The method allows for rapid switching between models, in a constantly changing environment, and has been tested on images from the Berkeley DeepDrive dataset, showing promising results.

Keywords; decentralized artificial intelligence, autonomous driving, convolutional neural networks.

I. INTRODUCTION

Conventional Artificial Intelligence (AI) follows a centralized distribution pattern where one organization is in charge of all the steps of the Machine Learning (ML) process. Contrasting that is Decentralized AI (DzAI), where learning is a collaborative solution solved by a distributed group of

participants. It is important to distinguish DzAI to distributed AI (DAI). In DAI agents work together to accomplish a global task, whereas in DzAI they collaborate in order to achieve their own task.

The notion of DzAI is nothing new [1], but has regained traction in the last couple of years. Centralized AI, especially in the context of autonomous driving, suffers from multiple problems in its current state:

- Most large companies working on autonomous driving technology gather their data close to their own headquarters, usually on the west coast of the U.S. This does not result in diverse data.
- Lack of trust as seen in the recent uproar concerning the political influence of tech giants like Facebook, Google, and Amazon. This is largely related to the lack of transparency regarding the nature, quantity and security of data that these companies gather.
- Also suffering is the price/availability ratio of AI. With great potential benefits for businesses, the demand for good datasets, and engineers to work with them, is in high demand. Unfortunately this demand outstrips the supply [2], and as such state-of-the-art AI is only available to larger, resourceful companies with rich datasets.

The implementation of the European Union (EU)’s General Data Protection Regulation (GDPR) [3] granted people access to their own data. However, there are currently no viable options for utilizing these *small individual datasets* without sacrificing privacy, and as such the power of information still lies with the big companies, rather than the consumers. Combined with advances in the field of Internet of Things (IoT) the amount of distributed data is growing at an increasing rate, but lacking of any mechanisms to protect the interests of the owners of this information. In order to assess

people’s willingness to participate in a decentralized network sharing such data we ran a survey. The survey was published on the campus of the Norwegian University of Science and Technology (NTNU) and online on forums related to ML and data gathering. It received a total of 791 responses, the majority of which were from people between the ages of 18 and 30 years old. Just under 90% of the participants said that they would be willing to gather and share images from their driving to improve self-driving cars, for the right price. Of these 90%, about 60% indicated that would be worried about their privacy when sharing such data, even though they were still willing to share it.

This paper proposes a new paradigm for the data gathering for AI models, and how they can be used by autonomous vehicles in a decentralized network. Not only is data gathered at the edge of the network, the models are trained there as well. The agents in the network proposed in this paper are Deep Learning (DL) models trained to perform object detection in images, for use in autonomous driving. The images might include multiple objects of different classes, like *car*, *person* or *traffic light*.

The object detectors in the network are trained by individual users, and will be biased based on the context they are trained in. In other words, an agent who gathers data in an urban environment will train a model that is better at detecting persons than trucks. Autonomous vehicles in the network can choose the best detector available for its current environment, e.g. a city centre, or a rural highway. To make this choice we propose an approach based on context-awareness, using reinforcement learning. The approach is called a *Context-Sensitive Detector* (CSD), and allows rapid switching between models, in a constantly changing environment. By understanding the performance of the models comprising the CSD, the model with the current best performance can be selected and exploited. Section V shows experiments run with the CSD on the Berkeley DeepDrive (BDD) dataset [4], with promising results.

II. DECENTRALIZED AUTONOMOUS DRIVING

A. Advantages and disadvantages

Like any other AI method, DzAI in autonomous vehicles has both advantages and disadvantages. It is important to assemble a list of these pros and cons, and map how they might influence the practical viability of such a system. The possible advantages and disadvantages are too numerous to feasibly explore in a single paper. Based on importance, a select few have thus been chosen, described below.

Possible advantages of DzAI in autonomous vehicles:

- **Improved preservation of privacy.** With the introduction of GDPR, people will have stronger rights to their data, and can be encouraged or incentivized to share it. Transmitting encrypted and anonymous trained models instead of raw data will reduce the risk of theft and unauthorized eavesdropping. Not transmitting raw data greatly reduces the risk of exposing sensitive information, which in turn will increase the likelihood of data owners being willing to share their data.

- **Greater data diversity.** With improved preservation of privacy leading to more sharing of data, this data will likely come from more diverse locations and environments as well, increasing both the quantity and the quality. This is important for any AI problem, but even more so for self-driving car models. As can be seen in section IV, the context that models are trained with has a great say in how they perform in different scenarios. Gathering data, and by extension models, from diverse areas will allow a DzAI network to perform well in a range of different scenarios.

Possible disadvantages of DzAI in autonomous vehicles:

- **Worse accuracy.** Depending on the model of assembly, the accuracy of the combined model might be worse than that of a single model trained on all of the training data at once. Section IV shows the results from experiments on how to choose the most optimal model in a given situation.
- **Labeling.** Object detectors are usually used in supervised learning. If the data for the locally trained models is to be sourced from individual users, someone needs to do the labeling. If this responsibility is to be put on the users, the process of labeling must be *really* simple, efficient and user friendly, and the individual models must require rather small amounts of data.

B. Architecture

The DzAI network proposed in this paper is envisioned to connect participants directly, with no need for a middleman, as indicated by figure 1. The network relies on individual users contributing by:

- Gathering driving data by mounting a camera or similar to their car.
- Training models locally using the gathered data. This step is explained more thoroughly in section II-C.
- Distributing these trained models to the rest of the network.

Once models are available on the network, autonomous vehicles also connected to the network can use our proposed approach for combining the models and choose the best performing one in each scenario. This process is described briefly in section II-D and more thoroughly in section IV-C.

C. Training

Every participant in the network trains their object detectors locally, with no data leaving their devices. Since object detection is a type of *supervised* learning, the data needs to be labelled after it has been gathered. This involves annotating the different objects in the images: drawing bounding boxes around the objects, and indicating their class. Since the raw data is only to be kept locally, the user needs to be able to perform this task on their own. A simple tool like www.scalabel.ai [5], developed by the creators of the Berkeley DeepDrive dataset, is suitable for such a task. The survey described in section I found that many people were

willing to gather and share raw images from their driving for a low fee (from \$3 / hour to below \$1 / hour of driving). However, the survey respondents demanded much higher compensation for the task of annotating the images gathered, but that participants were willing to spend up to an hour or more per week on the task. The conclusion is that the size of the model training sets needs to be kept as small as possible, to avoid being too time consuming (and therefore expensive) to annotate.

D. Utilization

Finally, when the DzAI network contains multiple models trained on diverse, biased data, they are available to be utilized by autonomous vehicles on the network. We present an approach for combining trained models in order to increase their collective accuracy, using *Context-Sensitive Detectors (CSD)*. The CSD-approach is a *reinforcement learning (RL)* technique, known as the *Multi-Armed Bandit (MAB)*[6]. Its goal is to *discover* the best performing model and exploit that model while it's performing well. This way a self-driving car is able to rapidly switch between models in the network, based on the environment it is driving in. When switching environments, e.g. driving out of the city into a rural area, the CSD will discover a change in the environment, and switch to a more suitable detector. A more detailed explanation of the CSD can be found in section IV-C.

III. RELATED WORK

Existing approaches to DzAI tackle different aspects of the decentralization process. Projects like Ocean [7] and Datum [8] are trying to create a marketplace for sharing data in a decentralized fashion. OpenMined [9] wants to decentralize AI by bringing the models to each participant in a network, allowing them to train the models locally while keeping the data on the device. Homomorphic encryption [10] of the models allows users of OpenMined's network to share their models with no risk of theft. SingularityNET [11] is a decentralized AI network currently under development that aims to let anyone create, share and monetize AI services.

This paper presents a different paradigm, where privacy is an inherent attribute of the collaboration process. Similar to OpenMined, this paper describes an approach where training

is performed locally at the edges of the network, without the training data ever leaving the source. But instead of further training a previously created model, our proposed solution creates individual models at the data sources, that are then combined into *Context-Sensitive Detectors*, explained in more detail in section IV-C.

Google's subsidiary company Waymo, representing the centralized approach, has been exceptional at gathering huge amounts of sensor data with its self-driving cars, having driven more than 9 million miles since the fleet started operating in 2009 [12]. However, large portions of the data gathered by Waymo and other large technology companies have been recorded in sunny climates, on spacious highways with multiple lanes, often unidirectional, i.e. not the most challenging driving conditions. More data is still required for further training and improvement of autonomous vehicles, especially diverse data from cities in different countries, of different sizes, with different sign schemes etc.

The most relevant related work is likely Google's AI division's learning method *federated learning*, which is able to perform collaborative ML without centralized training data [13]. Google uses federated learning to enable smart phones to keep all their training data on the device, and simultaneously share the prediction models to collaboratively learn and improve the model. After downloading the most current model, the devices on the network learn using their own local data and improve the model, before sharing the summarized changes with the rest of the network. [14]. The CSD proposed in this paper, however, gathers models trained on the edges, and uses context to switch between them in a rapidly changing environment. Google's approach is more suitable for a more constant environment.

IV. EXPERIMENT SETUP

In the following section we discuss the setup for the experiments as they relate to our proposed approach. First we'll provide more details on the dataset utilized. Second we present the underlying neural network architecture, and lastly we look at the details on the proposed methodology for combining networks.

A. Dataset

The dataset used in the experiments in this paper is the *BDD100k dataset* provided by the Berkeley DeepDrive project. Facilitated by the researchers' own annotation tooling, the dataset is comprised of over 100 000 videos with diverse kinds of annotations including image level tagging, object bounding boxes, driveable areas and lane markings. The creators claim the dataset to be "the largest available dataset of annotated driving scenes". Compared to existing datasets used for driving image recognition benchmarks, it allegedly covers more realistic driving scenarios and captures more variation in appearance and pose configuration of categories of interest. The dataset is thus considered more challenging than existing sets.

The foundation of the dataset is a collection of 100 000 video clips (1.8 Terabyte (TB)) captured by the front

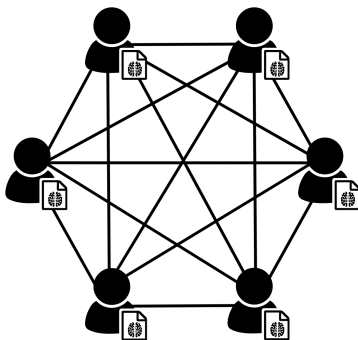


Fig. 1. Decentralized AI; models are traded on the network



Fig. 2. One of the 100k images in the BDD dataset

facing camera of a car driving around the U.S. in diverse conditions. From the video clips, 100 000 labeled key frame images have been extracted from the videos at the 10th second. From these key frame images, many attributes are available, like segmentation maps of driveable areas, and Global Positioning System (GPS) information. An example image can be seen in figure 2. The experiments in this article use the 100 000 labeled key frame images extracted from the videos and their annotation fields, which describe the classes of the objects detected in the image and their bounding boxes, respectively. There are 10 different classes: *Bus, traffic light, traffic sign, person, bike, truck, motor, car, train, rider*.

B. Computer Vision and Object Detection

Object detection is an important part of many autonomous systems, like facial recognition, camera surveillance, and self-driving cars. Recent years have seen an increase in the utilization of deep neural networks for computer vision tasks, mostly through the use of convolutional neural networks (CNNs).

All of the models used in the experiments presented in this paper were constructed based on the YOLOv9000 architecture [15]. With that in mind the methodology presented maintains the real-time object detection offered by said architecture, which is essential for time sensitive tasks like driving.

As this paper conveys a *proof-of-concept*, the YOLO models were initialized with a pre-trained set of weights, and only the last two layers were trained on the BDD dataset, using transfer learning. As such the accuracy presented in this paper is not comparable with state-of-the-art approaches, but only used for comparison against a baseline model trained on 20.000 images over 15 epochs.

C. Combining neural networks

Next we present an approach for combining trained models in order to increase accuracy.

Context-Sensitive Detector (CSD): The CSD-approach is a *reinforcement learning* (RL) technique, known as the *Multi-Armed Bandit* (MAB). Its goal is to *discover* the best performing model and exploit that model while it's performing well. To model the environment we use the notation S_A to indicate the types of objects observed in

location A. As such a drive from location A (eg. the city), through location B (eg. the countryside), and back to A again, can be modelled as:

$$S_{city} \rightarrow S_{countryside} \rightarrow S_{city} \quad (1)$$

The goal of the CSD is to balance exploration and exploitation. To do so the best performing model is chosen using a *policy*. For the changing environment in eq. 1 the Sliding-Window Upper Confidence Bound (SW-UCB) algorithm was used (eq. 2)[16]. This approach works for an infinite horizon time series as it only retains the reward metric for a set number of predictions using a *sliding window*.

$$\bar{x}_\tau + B \sqrt{\frac{\varepsilon \ln(t \wedge \tau)}{n_\tau}} \quad (2)$$

Here \bar{x}_τ is the average score over the last τ observations. B is the expected maximum score. The discounting factor ε is used for punishing suddenly poorly performing models, and τ is the window size. The term $t \wedge \tau$ is the minimum of the two. Lastly n_τ is the number of times the model was used during the last τ observations.

The CSD experiments in the experiments section (sec. V) explores how the two hyper parameters; the discounting factor (ε) and the window size (τ) affects the performance of the policy.

As the purpose of the experiments is to compare the SW-UCB performance to the baseline model, we use an F1-score (eq. 3) as the scoring measure. There exists scoring mechanisms that only rely on the models' own perceived score which would be more suitable for a production scenario. However, this falls outside the scope of the experiments and would require additional tuning.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3)$$

V. EXPERIMENT RESULTS

The CSD experiments present the mAP performance for different approaches, but more importantly present the underlying decision process, which is important for the conclusion and future work section.

To assess the viability of doing context-dependent selection of deep learning models in real time, the experiments in the following section were run with varying parameters, as described below:

- Training quality, number of available models and number of object types needing to be predicted (eg. *cars, trucks, and pedestrians*).
- Discounting factor (ε). The rate of which the CSD discards suddenly poorly performing models. Higher values should increase the rate of exploitation, and lower values should increase the rate of exploration.
- Window size(τ). The window of historical knowledge the CSD retains about the previous performance of the models. Increasing the window size should allow the

mAP [%]		τ		
		5	20	200
ε	0.05	0.98	14.25	5.08
	0.5	8.44	18.94	4.84
	5	8.38	13.08	4.62

Fig. 3. mAP results for varying hyper parameters

CSD to use a larger portion of previous performance in order to make decisions. However, using a too large window will prevent the CSD from discovering the changing context.

A. Distinguishable Context

In order to test the CSD’s ability to accurately select the appropriate models based on context, the majority of the experiments were tested on a distribution of images where the context is clearly distinguishable. Almost every image in the BDD-dataset contains cars and all the models trained on images from this dataset all perform adequately when it comes to car detections. By using images with *pedestrians* and *trucks* we discovered there was a smaller overlap in the selected images, and such these two labels were used to simulate an easily distinguishable context

The experiments were run on a distribution of images simulating a drive by varying the type of objects appearing at different time steps. In order to test the detector’s ability to both detect new contexts, and re-discover previously detected contexts, the experiments are run on the environment described in eq. 4.

$$S_{4truck500} \rightarrow S_{8pedestrian1k} \rightarrow S_{4truck500} \quad (4)$$

Here the numbers (500 and 1000 respectively) refer to the number of images, and the types (4truck and 8pedestrian) refers to the bias of the images. In other words the environment consist of 500 images with four or more trucks, 1000 images with 8 or more pedestrians, and lastly 500 images (different from the first 500) with four or more trucks.

The underlying models are trained on similarly biased images. The first model is trained on a dataset consisting on 1000 images containing 3 or more trucks, trained for 15 epochs. The second model is trained on a different dataset containing 1000 images containing 10 or more pedestrians, again trained for 15 epochs.

In table 3 the result of varying the hyper parameters ε and τ , can be seen. Comparatively the 20k baseline model (trained on 20k images containing cars, trucks and pedestrians for 15 epochs) achieves an mAP of about 21% on the same dataset.

Figure 6 shows the score and policy reward plotted against the distribution of images. Along with the F1-score the performance of the 20k baseline model is plotted as the dotted line. There are a couple of performance drops in the CSD around both the 500 and the 1500 marks. These drops stem from the switching between the two distributions and the CSD quickly picks up the other model. Also interesting is the drop around the 250-mark. This is likely due to a

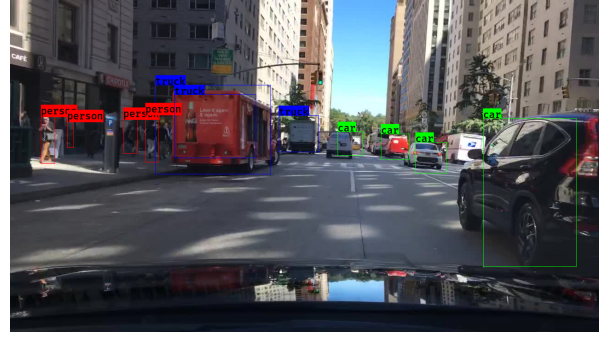


Fig. 4. Predictions for the 4truck1k model

couple consecutive images with challenging objects, making the CSD uncertain about the performance of the *truck*-model. After a couple of predictions using the *pedestrian*-model the CSD quickly corrects the reward (seen in the topmost plot) and switches back to the *truck*-model.

The predictions for an image done by the CSD can be seen in figure 4. It accurately uses the *4truck1k* model to perform the prediction, as there are 3 trucks in the image (two of which are correctly classified). The image also showcase the complexity of performing good predictions on a given image, given the number of trucks, cars, and pedestrians.

B. Indistinguishable Context

In order to test the CSD’s performance when the context is not easily distinguishable, the detector were also run on a distribution containing 500 images of each label in succession.

Table 5 showcases the mAP performance of the CSD where all classes in the BDD dataset are used. Comparatively the 20k baseline model on said dataset achieves a score of roughly 8%, also implying that the detection of some of the objects are harder than others. It is apparent from the results that the CSD struggles when the underlying context is insufficiently distinguishable. This is also one of the shortcomings of using the F1 score as the underlying performance metric, as the *number* of predicted objects plays an important role in the calculation. The score is oblivious to whether the model predicts 5/5 cars and 0/10 traffic lights even in an environment where traffic lights are the contextual differing factor.

C. Performance

Essential to the YOLO-architecture approach is its real-time performance. The methodology presented manages to maintain about 15 Frames-Per-Second on a Pascal GTX 1080 Graphics Processing Unit (GPU). With its pre-tuning the overhead of the CSD-policy itself is minuscule, and most

mAP [%]		τ
		20
ε	0.05	4.85

Fig. 5. mAP results for an insufficiently distinguishable scan area

of the added time comes from the F1-score calculations. It is therefore to be expected that a different scoring metric could see a significant drop in processing time.

VI. CONCLUSION AND FUTURE WORK

The CSD approach shows promise when the underlying contexts are easily distinguishable from one another. However, the experiment presented in table 5 showcases one of the shortcomings of the CSD-approach, as it takes a big performance hit when the contexts are harder to detect and exploit.

When using the approach proposed in this paper it is evident, even from the relatively few number of experiments, that the hyper parameters need to be tuned in accordance to the underlying context. With basis in this, we propose a different approach to the policy determination, proposed below. Nonetheless, with adequate tuning of the parameters, the CSD containing two models trained on 1.000 images each, almost reaches the same mAP as the model trained on 20.000 images.

In order to improve the adaptability of the CSD, a Deep Reinforcement Learning Network[17] could prove to be a viable approach. Rather than using a pre-defined policy, a policy could be learned using Q-learning. This could allow for more flexibility and rate of adaptation compared to the hyper parameter tuning explored in this paper.

However to fully take advantage of the presented methodology we propose an approach more similar to *federated learning*. By combining a *static* model with the dynamic models from the CSD, we expect to see less of a performance drop in the switching phases, and allow the dynamic models to do regional overfitting on the static model. By doing contextual overfitting we expect to see a performance boost even in challenging environments, where the most common approach in Deep Learning is to throw more and more data at the problem, in order to teach the model about special *edge* cases.

REFERENCES

- [1] Y. D. Miiller, “Decentralized artificial intelligence,” *Decentralised AI*, pp. 3–13, 1990.
- [2] C. Metz, “Tech Giants Are Paying Huge Salaries for Scarce A.I. Talent.” <https://www.nytimes.com/2017/10/22/technology/artificial-intelligence-experts-salaries.html>, 2019. [Online; accessed 01-January-2019].
- [3] O. J. of the European Union, “GDPR full text.” <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&qid=1501071426222&from=en>, 2019. [Online; accessed 01-January-2019].
- [4] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling,” *NASA Ads*, pp. 1–16, 2018.
- [5] “Scalabel.ai.” <https://www.scalabel.ai/>, 2017.
- [6] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.
- [7] “Ocean Protocol.” <https://oceanprotocol.com/#papers>, 2018.
- [8] “Datum - Blockchain Data Storage and Monetization.” <https://datum.org/>, 2018.
- [9] “OpenMined.” <https://www.openmined.org/>, 2018.
- [10] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.
- [11] “SingularityNET.” <https://blog.singularitynet.io/research/home>, 2018.
- [12] “On the Road – Waymo.” <https://waymo.com/ontheroad/>, 2018.
- [13] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated Learning: Strategies for Improving Communication Efficiency,” pp. 1–10, 2016.
- [14] B. McMahan and D. Ramage, “Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data.” <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>, 2017.
- [15] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” dec 2016.
- [16] A. Garivier and E. Moulines, “On upper-confidence bound policies for switching bandit problems,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6925 LNAI, pp. 174–188, Springer, Berlin, Heidelberg, 2011.
- [17] S. S. Mousavi, M. Schukat, and E. Howley, “Deep Reinforcement Learning: An Overview,” jun 2018.

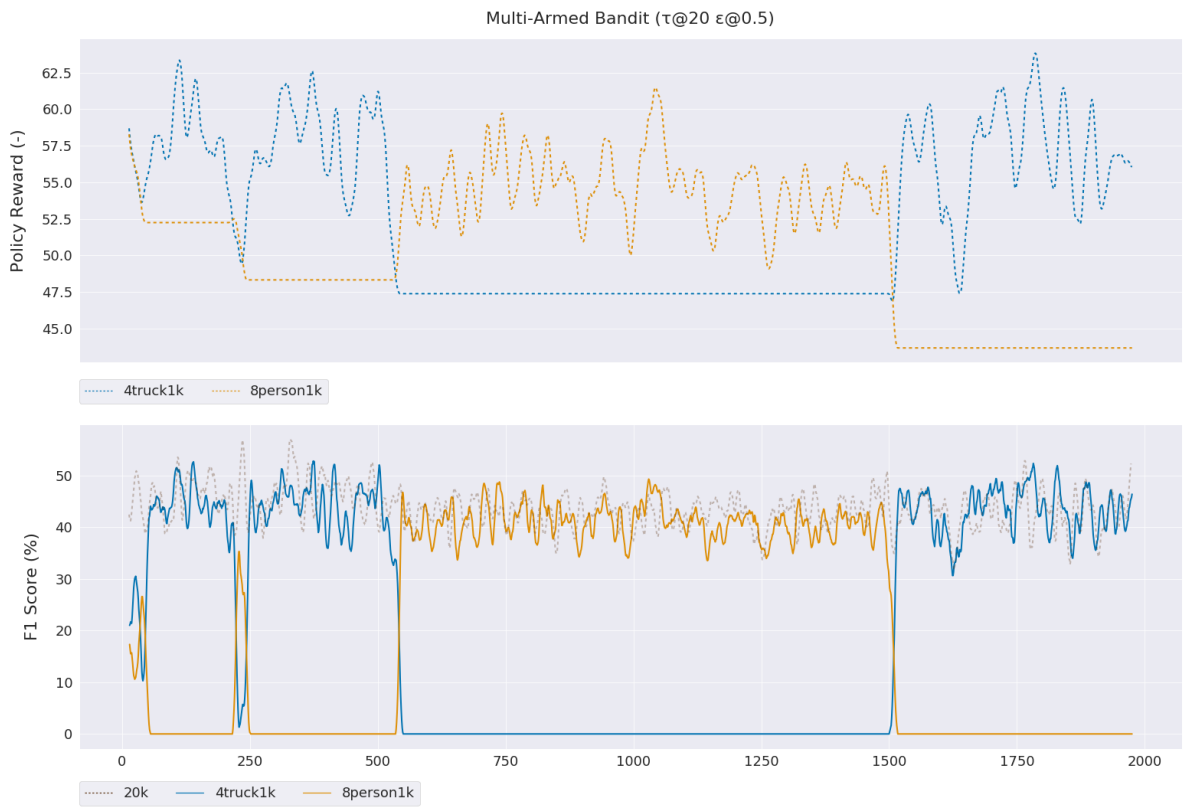


Fig. 6. Plotting the reward and policy metrics for the two models comprising the CSD

Bibliography

- [1] Google leads in the race to dominate artificial intelligence - Battle of the brains, 2017. URL <https://www.economist.com/business/2017/12/07/google-leads-in-the-race-to-dominate-artificial-intelligence>.
- [2] Andrew Meola. How IoT & smart home automation will change the way we live, 2016. URL <https://www.businessinsider.com/internet-of-things-smart-home-automation-2016-8>.
- [3] M Schulz and J.A. Hennis-Plasschaert. GDPR full text, 2016. URL <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&qid=1501071426222&from=en>.
- [4] Yuki Sugiyamal, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiko Nishinari, Shin Ichi Tadaki, and Satoshi Yukawa. Traffic jams without bottlenecks-experimental evidence for the physical mechanism of the formation of a jam. *New Journal of Physics*, 10(3):033001, 3 2008. ISSN 13672630. doi: 10.1088/1367-2630/10/3/033001. URL <http://stacks.iop.org/1367-2630/10/i=3/a=033001?key=crossref.d9c6328c540467e5d2beaf6961d03278>.
- [5] Benjamin Schneider. New Study of Global Traffic Reveals That Traffic Is Bad - CityLab, 2018. URL <https://www.citylab.com/transportation/2018/02/traffics-mind-boggling-economic-toll/552488/>.
- [6] Benjamin Zhang. Morgan Stanley: Autonomous Cars May Save US A Trillion Dollars - Business Insider, 2014. URL <https://www.businessinsider.com/morgan-stanley-autonomous-cars-trillion-dollars-2014-9?r=US&IR=T&IR=T>.

- [7] Umar Zakir, Abdul Hamid, Konstantin Pushkin, Hairi Zamzuri, Djahid Gueraiche, Mohd Azizi, and Abdul Rahman. Current Collision Mitigation Technologies for Advanced Driver Assistance Systems – A Survey. *PERINTIS eJournal*, 6(2):78–90, 2016. ISSN 2232-0725. doi: 10.1105/tpc.15.01050.
- [8] Ron Medford. Autonomous Vehicle Disengagement Reports 2015. Technical report, Google Auto LLC, 2015. URL https://www.dmv.ca.gov/portal/wcm/connect/df67186-70dd-4042-bc8c-d7b2a9904665/google_disengagement_report.pdf?MOD=AJPERES&CVID=.
- [9] Ron Medford. Autonomous Vehicle Disengagement Reports 2016. Technical report, Waymo, 2017. URL https://www.dmv.ca.gov/portal/wcm/connect/946b3502-c959-4e3b-b119-91319c27788f/GoogleAutoWaymo_disengage_report_2016.pdf?MOD=AJPERES&CVID=.
- [10] Ron Medford. Autonomous Vehicle Disengagement Reports 2017. Technical report, Waymo, Sacramento, 2017. URL <https://www.dmv.ca.gov/portal/wcm/connect/42aff875-7ab1-4115-a72a-97f6f24b23cc/Waymofull.pdf?MOD=AJPERES&CVID=>.
- [11] On the Road – Waymo. URL <https://waymo.com/ontheroad/>.
- [12] Marius Maaland and Anders Klever Kirkeby. Decentralized Artificial Intelligence utilizing Blockchain technology. 2018.
- [13] Gustav von Zitzewitz. Survey of Neural Networks in Autonomous Driving. (April): 0–8, 2017. URL https://www.researchgate.net/publication/324476862_Survey_of_neural_networks_in_autonomous_driving.
- [14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 12 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y. URL <http://link.springer.com/10.1007/s11263-015-0816-y>.
- [15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015. ISSN 0028-0836. doi: 10.1038/nature14539. URL <http://www.nature.com/articles/nature14539>.

-
- [16] Shyamal Patel and Johanna Pingel. CNN Architecture. URL <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--14895127657.html>.
- [17] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. 12 2016. URL <https://arxiv.org/abs/1612.08242>.
- [18] TensorFlow. URL <https://www.tensorflow.org/>.
- [19] Keras Documentation. URL <https://keras.io/>.
- [20] Lisa Torrey, Lisa Torrey, and Jude Shavlik. Transfer Learning. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.146.1515>.
- [21] ImageNet. URL <http://www.image-net.org/>.
- [22] Joseph Redmon. YOLO: Real-Time Object Detection, 2016. URL <https://pjreddie.com/darknet/yolov2/>.
- [23] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. 5 2014. URL <http://arxiv.org/abs/1405.0312>.
- [24] Yves Demazeau and Jean-Pierre Muller, editors. *Decentralized A.I.: Proceedings of the European Workshop on Modelling Autonomous Agents in a Multi-Agent World*. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 1990. ISBN 0444887059.
- [25] Ocean Protocol — A Decentralized Data Exchange Protocol to Unlock Data for AI — Ocean Protocol. URL <https://oceanprotocol.com/#papers>.
- [26] Datum - Blockchain Data Storage and Monetization. URL <https://datum.org/>.
- [27] Roger Haenni. Datum - White Paper. 2017. URL <https://datum.org/>.
- [28] SingularityNET. URL <https://singularitynet.io/>.
- [29] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated Learning: Strategies for

- Improving Communication Efficiency. pages 1–10, 2016. ISSN 2162-7843. URL <http://arxiv.org/abs/1610.05492>.
- [30] Brendan McMahan and Daniel Ramage. Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data. URL <https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>.
- [31] OpenMined. URL <https://www.openmined.org/>.
- [32] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, 2009. URL <http://portal.acm.org/citation.cfm?doid=1536414.1536440%5Cnhttp://crypto.stanford.edu/craig/craig-thesis.pdf>.
- [33] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Praseon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. 4 2016. URL <http://arxiv.org/abs/1604.07316>.
- [34] Autonomous Car Development Platform from NVIDIA DRIVE PX2. URL <https://www.nvidia.com/en-gb/self-driving-cars/drive-platform/>.
- [35] Reddit. URL <https://www.redditinc.com/>.
- [36] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. pages 1–16, 2018. URL <http://arxiv.org/abs/1805.04687>.
- [37] Scalabel.ai. URL <https://www.scalabel.ai/>.
- [38] Adrian Rosebrock. Intersection over union sign example, . URL https://upload.wikimedia.org/wikipedia/commons/2/2d/Intersection_over_Union_-_object_detection_bounding_boxes.jpg.
- [39] Adrian Rosebrock. Intersection over union formula, . URL https://upload.wikimedia.org/wikipedia/commons/c/c7/Intersection_over_Union_-_visual_equation.png.
- [40] Adrian Rosebrock. Intersection over union comparison, . URL https://upload.wikimedia.org/wikipedia/commons/e/e6/Intersection_over_Union_-_poor%2C_good_and_excellent_score.png.

- [41] Jan Hosang and C V May. Learning non-maximum suppression.
- [42] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Chapter 8: Evaluation in information retrieval. *Introduction to Information Retrieval*, (c):1–18, 2009. ISSN 13864564. doi: 10.1109/LPT.2009.2020494. URL <http://nlp.stanford.edu/IR-book/pdf/08eval.pdf>.
- [43] Jonathan Hui. Precision-recall plot. URL https://cdn-images-1.medium.com/max/1600/1*VenTq4IgxjmIpOXWdFb-jg.png.
- [44] Yutaka Sasaki. The truth of the F-measure. (March):1–6, 2007. URL <https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf>.
- [45] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003* -, volume 4, pages 142–147, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119195. URL <http://portal.acm.org/citation.cfm?doid=1119176.1119195>.
- [46] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC ©2012, 1st edition, 2012. ISBN 1439830037 9781439830031.
- [47] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. ISSN 0885-6125. doi: 10.1007/BF00058655. URL <http://link.springer.com/10.1007/BF00058655>.
- [48] Louisa Lam and Ching Y. Suen. Application of Majority Voting to Pattern Recognition: An Analysis of Its Behavior and Performance. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 27(5):553–568, 1997. URL <http://machine-learning.martinsewell.com/ensembles/LamSuen1997.pdf>.
- [49] N. Littlestone and M.K. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108(2):212–261, 2 1994. ISSN 0890-5401. doi: 10.1006/INCO.1994.1009. URL <https://www.sciencedirect.com/science/article/pii/S0890540184710091?via%3Dihub>.

- [50] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. URL <http://yann.lecun.com/exdb/mnist/>.
- [51] 20 Newsgroups Data Set. URL <http://qwone.com/~jason/20Newsgroups/>.
- [52] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science (New York, N.Y.)*, 362(6419):1140–1144, 12 2018. ISSN 1095-9203. doi: 10.1126/science.aar6404. URL <http://www.ncbi.nlm.nih.gov/pubmed/30523106>.
- [53] Ronald A. Howard and Ronald Arthur Howard. *Dynamic programming and Markov processes*. The M.I.T. Press, 1960. ISBN 0262080095. doi: 10.1107/S1600536806011159. URL <http://web.mit.edu/dimitrib/www/dpchapter.pdf>.
- [54] Michael N. Katehakis and Arthur F. Veinott. The Multi-Armed Bandit Problem: Decomposition and Computation. *Mathematics of Operations Research*, 12(2):262–268, 5 1987. ISSN 0364-765X. doi: 10.1287/moor.12.2.262. URL <http://pubsonline.informs.org/doi/abs/10.1287/moor.12.2.262>.
- [55] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002. ISSN 08856125. doi: 10.1023/A:1013689704352. URL <http://link.springer.com/10.1023/A:1013689704352>.
- [56] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6925 LNAI, pages 174–188. Springer, Berlin, Heidelberg, 2011. ISBN 9783642244117. doi: 10.1007/978-3-642-24412-4_{-}16. URL http://link.springer.com/10.1007/978-3-642-24412-4_16.
- [57] Mighty AI — Training Data for Computer Vision Models. URL <https://mighty.ai/>.
- [58] Amazon Mechanical Turk. URL <https://www.mturk.com/>.

- [59] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. Deep Reinforcement Learning: An Overview. 6 2018. doi: 10.1007/978-3-319-56991-8_{-}32. URL <http://arxiv.org/abs/1806.08894>http://dx.doi.org/10.1007/978-3-319-56991-8_32.
- [60] Zhi Xiong Xu, Lei Cao, Xi Liang Chen, Chen Xi Li, Yong Liang Zhang, and Jun Lai. Deep reinforcement learning with sarsa and Q-learning: A hybrid approach. *IEICE Transactions on Information and Systems*, E101D(9):2315–2322, 9 2018. ISSN 17451361. doi: 10.1587/transinf.2017EDP7278. URL <http://arxiv.org/abs/1509.06461>.
- [61] Moeslund Thomas B. Rasmussen Christoffer Bøgelund, Nasrollahi Kamal. Aalborg Universitet R-FCN Object Detection Ensemble based on Object Resolution and Image Quality R-FCN Object Detection Ensemble based on Object Resolution and Image Quality. 1(Ijcci):110–120, 2017. doi: 10.5220/0006511301100120. URL <http://vbn.aau.dk/ws/files/261392484/quality.pdf>.
- [62] 2nd International Conference on Intelligent Autonomous Systems. volume 7, pages 83–248, 1991. ISBN 09218890 (ISSN). URL <http://www.icias.org/submission.html><https://www.scopus.com/inward/record.uri?eid=2-s2.0-0026203464&partnerID=40&md5=1bf41fa612dd5194906eb7d50541c61e>.

