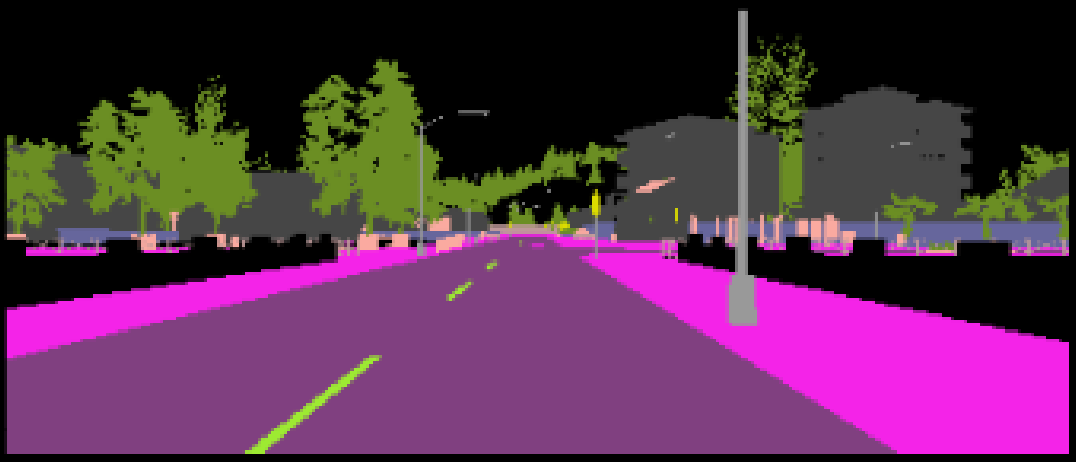
Cheating by Segmentation



Thomas A. van Orden

Layout: typeset by the author using LaTeX.
Cover illustration: CARLA

Cheating by Segmentation

Thomas A. van Orden
12192880

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

June 25, 2021

Abstract

Urban driving is a tough problem to solve, especially when training would be done in the real world. A solution would be to learn to drive in a realistic simulation using a student-teacher approach. If the learned student model is then transferred to the real world and performs weak, the simulation-real-world gap is too large and training should be done with simulation data that better represents the real-world data. In this thesis, we propose a variation on the work from "Learning by Cheating" to use semantic segmentation data instead of bird's-eye view data. The former closer resembles the perspective of drivers in real-world situations, narrowing the simulation-real-world gap. The semantic segmentation data should be sufficient enough to be able to learn a nearly perfect driving policy regarding lane following, obstacle avoidance, traffic lights and navigation. End-to-end conditional imitation learning is shown to be prone to distribution mismatches. Therefore, learning how to recover from a slight deviation is difficult from a perfect expert. We have shown that the "Learning by Cheating" approach with semantic segmentation data instead of bird's-eye views results in a robust agent that is able to solve most of the navigation challenges without dense traffic. Providing the supervised privileged agent with a semantic segmentation consisting of 13 classes is sufficient to learn the driving policy. The resulting policy is then robust enough to act as a teacher for a student agent. Our agent is validated on the original CARLA, *CoRL2017*, and *NoCrash* benchmark. It successfully solves the *Empty* task of the *CoRL2017* benchmark. On the other tasks and the more difficult *NoCrash* benchmark, our agent yields competitive results to Chen, Zhou, et al. In addition, we improve the generalization performance to new environments in terms of the number of collisions by a factor 2.

Contents

Chapter 1

Introduction

In the Netherlands, 21,400 people have been severely injured due to a traffic accident in 2019 (SWOV, 2020). Moreover, in the same year, 661 people did not survive such accidents (CBS, n.d.). Improving the driving competencies of road users could reduce the number of these injuries and fatal accidents. However, road users might not always be willing to take additional driving lessons. When vehicles would exhibit driving behaviour such that they drive better than humans can, safety will improve too. Autonomous vehicles (AV) are vehicles that are able to drive autonomously to some extent. The capabilities can be measured in terms of the driving level at the scale of *Levels of driving automation* as defined by the Society of Automotive Engineers (SAE) (SAE International, 2018). However, building AV systems is a difficult problem to solve since it entails a multidisciplinary problem consisting of tasks such as perception, planning, and vehicle control.

Interest in autonomous vehicles is almost 100 years old and impressive progress has been made (Kröger, 2016). Already in 1925, Houdina Radio Control demonstrated a remote-controlled car cruising down Broadway: the "American Wonder" (Time Magazine, 1925). A second truck followed the car to send out radio signals to the remote-controlled vehicle which caused small electric motors to control it. However, the experiment failed when the operators lost connection and the vehicle crashed. Norman Bel Geddes provided a glimpse into the future at the New York World's Fair with his Futurama exhibit, sponsored by General Motors, in 1939 (Ferlis, 2007). His exhibit showcased vehicles following the road based on electromagnetic fields that were generated by embedded circuits in the lanes. A "traffic control tower" would be at the heart of the autonomous vehicle network to direct all vehicles. This idea was later in the 1950s demonstrated by RCA Labs to work in a controlled laboratory environment with a miniature vehicle (Wetmore, 2003). Only 4 years later in 1957, they showed a full-size car guided by magnetic fields on

a 400-foot public highway (RCA, 1958). In 1960, Stanford University was the first to demonstrate a vehicle able to navigate without outside help, such as radio signals or magnetic fields, in a highly controlled environment (Earnest, 2021). They used cameras and computers to perceive the world around the vehicle and plan its path. This can be seen as a turning point in history from guidance-based systems to autonomous-based systems — laying the foundation of computer vision (CV) in autonomous driving. During the next two decades, resources improved and systems were developed that used CV, LiDAR, and robotic control (Dickmanns et al., 1994). Despite the impressive progress these decades brought, the new century, 2000, gave the research a real boost.

One key advancement on this timeline was when the Defense Advanced Research Projects Agency (DARPA) of the United States Defense Department set out its first Grand Driving Challenge in 2004 (Dudley, 2015). Teams could participate in the challenge by developing an autonomous car able to drive a 150-mile track in the Mojave Desert. Despite no team finishing the first challenge, the total three Grand Driving Challenges may have served as catalysts for the development of AVs. In 2009, Google started a new self-driving-car project, which caused major automotive manufacturers' interest in AVs too, in the years following (Harris, 2014). Six years later, Tesla Motors updated their Model S software over the air to enable hands-free highway and freeway control for their owners (Kessler, 2015). However, due to regulations, owners still had to keep their hands on the steering wheel and pay attention at all times. Besides Tesla, other automotive manufacturers started to introduce autonomous features, such as autonomous lane keeping, to their vehicles as well. Despite the continuous improvement of autonomous systems, the first fatal accident with an autonomous vehicle, a Tesla Model S, was reported in 2016 (Yadron and Tynan, 2016).

As the above overview of AV history shows, autonomous systems need to be trained and tested and will therefore not always drive safely from the start. Hence, training those in the real world could cause dangerous traffic situations — as with the "American Wonder". Current technologies such as simulations make it possible to use a safe environment to train and test. Furthermore, simulations enable fast and parallel experiments with possible benchmarks to objectively compare different systems.

1.1 Aim

Chen, Zhou, et al. (2019) have shown with "Learning by Cheating" that a student-teacher approach to autonomous-driving problems can be successful. By leveraging the benefits of first training a teacher with privileged information to then transfer this knowledge to a student, they were able to achieve high scores on relevant benchmarks, *CoRL2017* (Dosovitskiy et al., 2017) and *NoCrash* (Codevilla, Santana, et al., 2019). However, their approach is difficult to deploy in the real world, since it heavily relies on bird's-eye view data, subsequently referred to as birdview. Ground truth birdviews in real-world environments are not widely available and classical image-to-birdview transformations such as Inverse Perspective Mapping (IPM) (Mallot et al., 1991) are inaccurate since IMP assumes the world to be flat. Moreover, even more recent deep-learning-based approaches (Reiher, Lampe, and Eckstein, 2020) also lack accuracy due to occlusions in the original image leading to missing information in the birdview.

Therefore, this thesis tries to use ground truth segmentation data — from a driver's perspective — instead of birdviews, since current publicly available algorithms are able to accurately create segmentation images from real-world RGB input (Weber et al., 2021). The new approach will be validated on the two benchmarks. This leads to the following research question: *How is the performance of Chen, Zhou, et al.'s method affected when a ground-truth segmentation view is used instead of a birdview?* The aim of this thesis is to examine the effect of the segmentation data on the trained models, at every step, and provide insight into the problems of changing the data its perspective and type. Moreover, by validating every step of the method in the aforementioned benchmarks, the sub-questions: *What is the effect of the new data on the teacher's performance?*, and *To what extent can the new teacher transfer its knowledge to the student?*, will be answered as well.

Chapter 2

Theoretical Foundation

Throughout this thesis, multiple concepts will be exploited and some prior knowledge acts as the basis of more complex applications. This chapter introduces and explains these concepts, and provides the prior knowledge to better understand this thesis' approach. First, the two main neural network approaches that are commonly used to solve an autonomous-driving problem, end-to-end learning (Bojarski et al., 2016) and decomposed learning (Dosovitskiy et al., 2017), are introduced. Second, this thesis' input of the neural network, semantic segmentation data, is explained. Third, a more complex parallel network configuration, known as multi-head, which is used as the output of the network is described. Finally, the final step of the training method called data aggregation (DAgger) (Ross, Gordon, and Bagnell, 2011) is introduced.

2.1 Decomposed learning

Decomposed learning splits a complex task into different sub-problems and is also known as a modular pipeline (Dosovitskiy et al., 2017). In this way, every sub-problem can be solved and trained in a specific manner. Therefore, the approach can be optimized for every step. The output of the first sub-module can serve as input to the next module, which could then serve as input to the following, and so on. This approach has the advantage that the performance can be inspected at the module level. Especially if the performance is less than expected, the cause could be one sub-module to insufficiently solve its task instead of the whole network failing. However, a disadvantage is that for every module a training strategy should be implemented. Moreover, the training time could be higher if the sub-modules use extensive networks to solve their problem. Figure 2.1 shows an example from Serban, Poll, and J. Visser (2018) of multiple modules to solve an autonomous driving task. "A Standard Driven Software Architecture for Fully

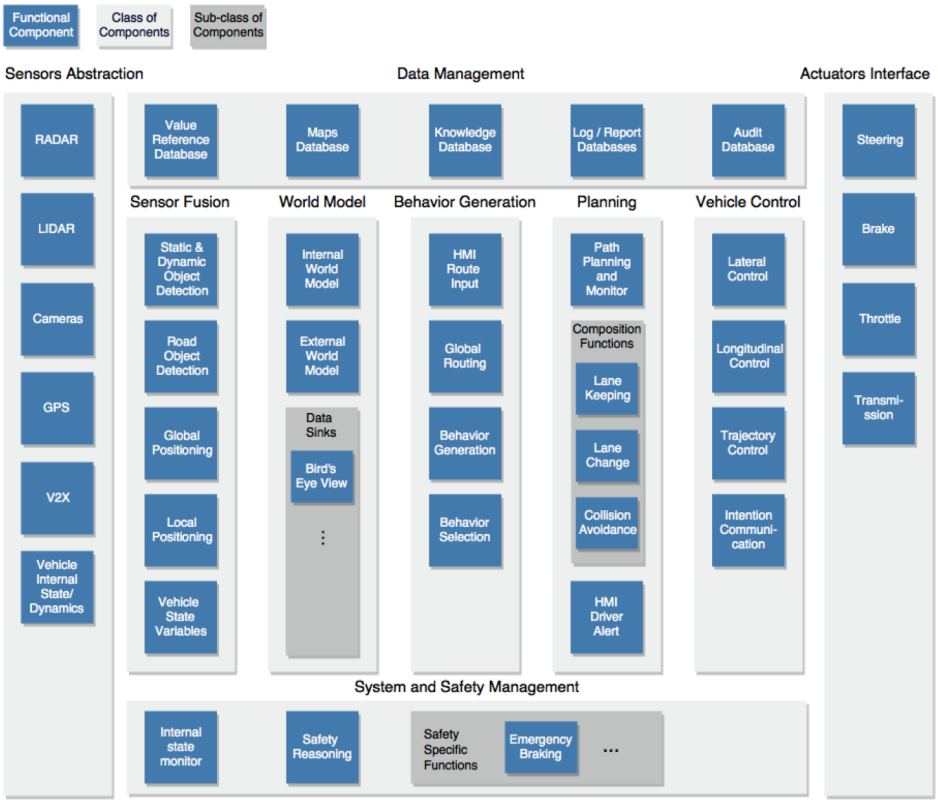


Figure 2.1: Decomposed learning to predict actuator values from sensor data. Figure and approach from "A Standard Driven Software Architecture for Fully Autonomous Vehicles" (Serban, Poll, and J. Visser, 2018).

Autonomous Vehicles" (Serban, Poll, and J. Visser, 2018) proposes an architecture that takes sensor data as input and uses different intermediate modules such as *Sensor Fusion* and *Planning* to generate actuator values for *Steering*, *Brake*, *Throttle*, and *Transmission*. This architecture shows the possible complexity of a decomposed-learning approach.

2.2 End-to-end learning

In contrast to decomposed learning, end-to-end learning is a technique to train neural networks in a direct supervised input-output fashion. The problem is defined as solving the function that best matches the input to the network to its desired output. It is used in particular in the context of (deep) neural networks since current networks contain millions of parameters that need to be trained. All available and relevant input is given to a network to be then supervised with

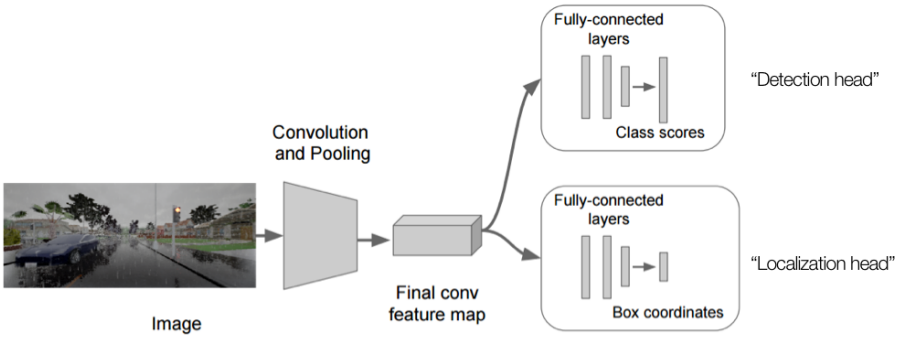


Figure 2.2: Multi-head approach example.

ground truth output. The strength of end-to-end learning is that the network learns all required skills in one go to solve the problem. However, this is also the downside of the technique, because if the network fails to solve the task sufficiently, it is hard to determine if it can solve some steps or if it fails in total. Knowing if and which steps the network can solve and which it cannot enables further research in the latter direction. In the context of autonomous driving, sensor information, for example, a camera or LiDAR stream, can be used as input to the network. For the output, a steering angle can be used as the ground truth label to let the network predict where the vehicle should steer towards.

2.2.1 Multi-head

Certain tasks require multiple predictions to get to the final prediction. Localization of an object in an image is such a task. This task consists of two steps: detecting and classifying the object, and localizing where the object is in the image. For example, consider an image with a car, the network needs to detect that there is a car (detection and classification) and tell where — which pixels — the car is (localization). These type of tasks can be approached with multi-head neural networks, first introduced by (Sermanet et al., 2014). The *backbone* can be seen as the heart of the network; it performs all convolutions and pooling and thereby learns features from the input. These features can then be fed to multiple blocks of layers, typically fully connected layers, which each draw their own conclusion based on the features. In the context of the car image, a backbone CNN learns image features and feeds these into a head for detection and a separate head for localization. The former head learns a mapping from features to classes such as buildings and cars. On the other hand, the latter head learns a mapping from image features to bounding box coordinates. These multi-head approaches are an efficient way to re-use the image features for multiple tasks. Figure 2.2 shows an example of a multi-head approach.

2.3 Student-teacher setup

A student-teacher setup combines end-to-end imitation learning with decomposed learning. This means that a problem is split into two sub-problems, which could be identical, and are both to be solved by the student and teacher, respectively. In general, this method is referred to as knowledge distillation (Hinton, Vinyals, and Dean, 2015) and is a form of model compression (Buciluundefined, Caruana, and Niculescu-Mizil, 2006). Knowledge distillation aims to achieve similar performance with a smaller student network in comparison to a larger teacher network. This is done by first training the teacher network to solve the task and then train the smaller student network to learn this exact behaviour as well. Not only the teacher's loss is used to supervise the student, but also the outputs at every level of the teacher's network. Data is passed through the teacher network to get all intermediate states. These outputs in combination with the final teacher output are used to back-propagate the student network which has seen the exact same input.

Knowledge distillation can also be taken a step further by giving the student a different input than the teacher, but remaining the same desired output. This lets the student solve a slightly different task than the teacher since the input changed. This approach has been proposed by Chen, Zhou, et al. (2019) in the context of autonomous driving. Section Section 4.1 describes their approach in more detail. Chen, Zhou, et al. mention that this approach has the benefit of splitting the driving task into learning to see and learning to act, which results in better performance.

2.4 Semantic segmentation

RGB cameras are commonly known and generate 3-channel — Red, Green, Blue — images. Each pixel is represented by 3 values that range from 0 to 255, for standard 24-bit color, one for each channel. An 364×180 image thus yields $364 \times 180 = 65,520$ pixels. This gives the image space of this image $(255^3)^{65,520} = (16,581,375)^{65,520}$ possibilities.

Semantic segmentation can be seen as a processing step that groups pixel values into several discrete classes. In other words, certain pixel values — colours — in an RGB image are assigned a single value — class. For example, an image of trees is not represented by a variety of green values, as RGB would do, but all pixels that belong to the tree have the same value. If this is done for multiple classes, each class can be assigned its own channel in an image. With

N segmentation classes, the image would have N channels — RGB has three — where each channel is a binary mask for that specific class. This brings the image space of a similar-sized image to: $2^{384\times160\times N} = 2^{65,520N}$ possibilities. Notice that with a binary mask there are 2 possible pixel values. So, as long as $N < 23$, see Equation 2.1, semantic segmentation yields a smaller image space. A smaller input image space means in general faster training and semantic image data have been proven to boost urban-driving performance for a convolutional neural network (CNN) (Zhou, Krähenbühl, and Koltun, 2019).

$$\begin{aligned} 2^{65,520N} &< (16,581,375)^{65,520} \\ \log_2(2^{65,520N}) &< \log_2((16,581,375)^{65,520}) \\ 65,520N &< \log_2((16,581,375)^{65,520}) \\ N &< \frac{\log_2((16,581,375)^{65,520})}{65,520} \\ N &< \log_2(16,581,375) \\ N &< 23.983... \end{aligned} \tag{2.1}$$

In the context of this thesis, 13 semantic segmentation classes are used. The classes are defined by CARLA, the simulator as described in Section 5.1: *None*, *Buildings*, *Fences*, *Other*, *Pedestrians*, *Poles*, *RoadLines*, *Roads*, *Sidewalks*, *Vegetation*, *Vehicles*, *Walls*, and *TrafficSigns*. An example of a CARLA segmentation image is shown in Figure 4.1b. From this thesis' point of view, the segmentation classes *Roads*, *Vehicles*, *Pedestrians*, and *TrafficSigns* are of most importance. The first class represents the valid prediction space and the latter represent objects to possibly stop for.

2.5 DAgger

As proposed by Ross, Gordon, and Bagnell (2011), DAgger is an iterative algorithm to improve the performance of, in particular, sequential prediction problems. These are problems where the solution to the current question depends on the solution to the previous question, so the predictions are not independent and identically distributed (i.i.d.). The intuition behind DAgger is that a learned policy, in the context of this thesis; an agent, is used to create a dataset where this dataset is annotated with expert labels. This agent is then trained on this dataset and rolled out again to create a new dataset. This new dataset is combined with the original dataset and is subsequently used to train the agent with the new accumulated dataset. This iterative process is dependent on the predictions of the

agent, e.q. the next dataset is dependent on the current agent. In this way, the sequential element of the problem is captured by the algorithm. Out of all learned policies, the policy that performs best under validation circumstances is used. The downside of DAgger is that it could be a slow process since the agent is used online — that is, rolled out in the environment — to create a new dataset. However, in the context of autonomous driving and this thesis, Chen, Zhou, et al. (2019) have shown that using DAgger significantly improves the performance of their agent.

The explained concepts such as end-to-end learning with a multi-head configuration and a student-teacher approach will be used extensively in the following chapters. The next chapter provides the scientific context of this thesis which is closely related to these subjects.

Chapter 3

Related work

To put this thesis in context, this chapter describes related studies regarding autonomous driving. In particular, related work on variations of end-to-end approaches will be discussed, since this thesis is based on end-to-end learning too. Moreover, to provide a wide context, a reinforcement learning (RL) approach will be described too.

3.1 Conditional imitation learning

Codevilla, Müller, et al. (2018) proposed the extension of plain end-to-end learning by an additional high-level command to predict actuator values (CIL). They compare two different network architectures: a classic single-head end-to-end network which has an image, vehicle measurements and the additional command as input; and a branched architecture with multiple heads. In the latter configuration, the input only consists of an image and vehicle measurements while the high-level command serves as a switch for the multiple heads. Each head is specialized in a certain task: turning left, turning right, following the road. To incorporate enough variation in the dataset, they used a three-camera setup with one forward-facing camera and two cameras facing a bit to the side. Figure 3.1 shows a three-camera setup example. However, they found that this does not result in sufficiently robust driving. Therefore, noise is injected during the dataset generation which results in the agent steering a bit off the road. The branched architecture proves to be best with 88% and 64% success rates in CARLA's Town 01 and Town 02, respectively. However, the authors do not mention if any traffic was added to the environment during training or testing.

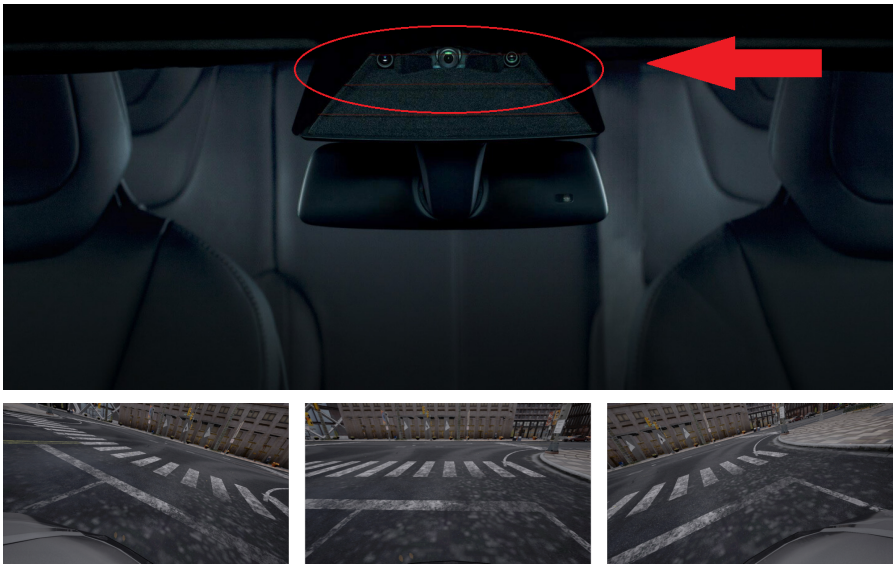


Figure 3.1: Example of three-camera setup behind a car's windshield (top) and an example-view of the three camera's from left, middle to right camera (bottom).

3.2 Conditional affordance learning

The conditional aspect of Codevilla, Müller, et al. (2018) is also incorporated by Sauer, Savinov, and Geiger (2018)'s approach (CAL). They use a low-dimensional intermediate representation, affordances, instead of directly learning actuator values. A video stream, which is a sequence of monocular RGB images taken from a forward-facing camera, serves as input to the network. The high-level directional command is used to switch between specialized branches, as proposed by Codevilla, Müller, et al. (2018). A set of multiple affordances is defined which should cover all necessary information to safely drive from a starting point to an end goal. For example, *hazard stop*, *distance to leading vehicle*, and *distance to centerline* are used. The network predicts values for each of the affordances which thereafter are used as input to a low-level PID controller to control the vehicle in CARLA. In other words, the network predicts values for a set of criteria which are then fed to a controller to transform them into actuator values. On the *CoRL2017* benchmark, this approach outperforms CIL on all tasks in the most difficult generalization conditions: a new town with new weather. However, their agent collides, on average, with a static object every 310 meters.

3.3 Improved conditional imitation learning

Codevilla, Santana, et al. (2019) aim to identify the limitations of behaviour cloning approaches. They propose an improvement upon CIL and propose a new driving benchmark, *NoCrash*. See Subsection 5.4.2 for more details on this benchmark. In contrast to CIL, the authors use a deeper ResNet-34 (He et al., 2015) network, add a separate speed prediction branch, use instead of Mean Square Error (MSE) the L_1 as loss, and pre-train on ImageNet (Deng et al., 2009). This method, subsequently referred to as CILRS, outperforms CAL on almost all tasks. In addition, CILRS almost solves the *Empty* task on the introduced *NoCrash* benchmark. Despite this, the approach drastically degrades in the presence of more dense traffic, indicating that using larger networks is not sufficient to solve the dense traffic scenarios.

3.4 Reinforcement learning

In contrast to the aforementioned end-to-end imitation approaches, Toromanoff, Wirbel, and Moutarde (2020) propose the use of RL for urban driving. Specific RL details are not discussed here, since this is out of the scope of this thesis. The authors propose a network consisting of a conditional architecture with a ResNet-18 (He et al., 2015) backbone, used as an encoder, in combination with Rainbow-IQN (Toromanoff, Wirbel, and Moutarde, 2019) RL training. The encoder is trained in a supervised fashion by predicting implicit affordances, a similar intuition as proposed by Sauer, Savinov, and Geiger (2018). A semantic segmentation representation and the distance to a traffic light are included as affordances. After training the encoder, the affordance decoders are removed — hence the naming implicit affordance, and the output — the implicit affordances — of the encoder is fed to the RL learning architecture. The reward function is based on the desired speed, desired position, and the desired rotation of the agent. CARLA's built-in waypoints are used to calculate the difference between the RL agent's position and rotation and the optimal position and rotation (the CARLA waypoint). Like Sauer, Savinov, and Geiger (2018)'s approach, the input to the network is a sequence of 4 RGB images and the past vehicle speed — that is the speed of the vehicle from the previous step. This RL approach achieves high scores on the *CoRL2017* benchmark, by solving almost all tasks in train and test weather. On the *NoCrash* benchmark, Toromanoff, Wirbel, and Moutarde (2020)'s approach yields similar results to the student-teacher approach of Chen, Zhou, et al. (2019) (LBC), as described in more detail in Section 4.1. However, in train weather, LBC significantly outperforms the RL-agent in the train and test town.

3.5 World on rails

In a more recent work, Chen, Koltun, and Krähenbühl (2021) propose a model-based approach which includes a world-model, RL action-value functions and knowledge distillation. The core of this approach is the world model which is able to simulate the agent's actions without actually executing the action. The important assumption is made that the agent's actions do not directly affect the world, hence the name "Learning to drive from a world on rails". The proposed method consists of multiple steps from which specific details are out of the scope of this thesis. First, the world model is trained on pre-recorded real-world trajectories which include sensor data, driving states and the agent's actions. Second, action values are computed for each possible action and for each frame in the dataset of trajectories. Finally, the action values are used to distillate the knowledge to a visual agent. This visual agent only has a monocular RGB image, its speed, and a high-level navigation command as input. By leveraging the world model, the visual agent can explore the consequences of its predicted actions, without actually executing them. The visual student has in addition to the task to predict the actuator values steer, brake, and throttle, the task to predict a segmentation image. This segmentation image is used as auxiliary loss and proves to significantly improve the performance of the agent. The authors mention that this improvement is, in particular, visible in generalization to new environments. The approach is validated in CARLA on the *NoCrash* benchmark and outperforms all prior works.

Despite all aforementioned approaches, including the current state-of-the-art by Chen, Koltun, and Krähenbühl (2021), the *NoCrash* benchmark is still not completely solved. The following chapter introduces our approach which is closely related to end-to-end conditional imitation learning and multi-head architectures.

Chapter 4

Approach

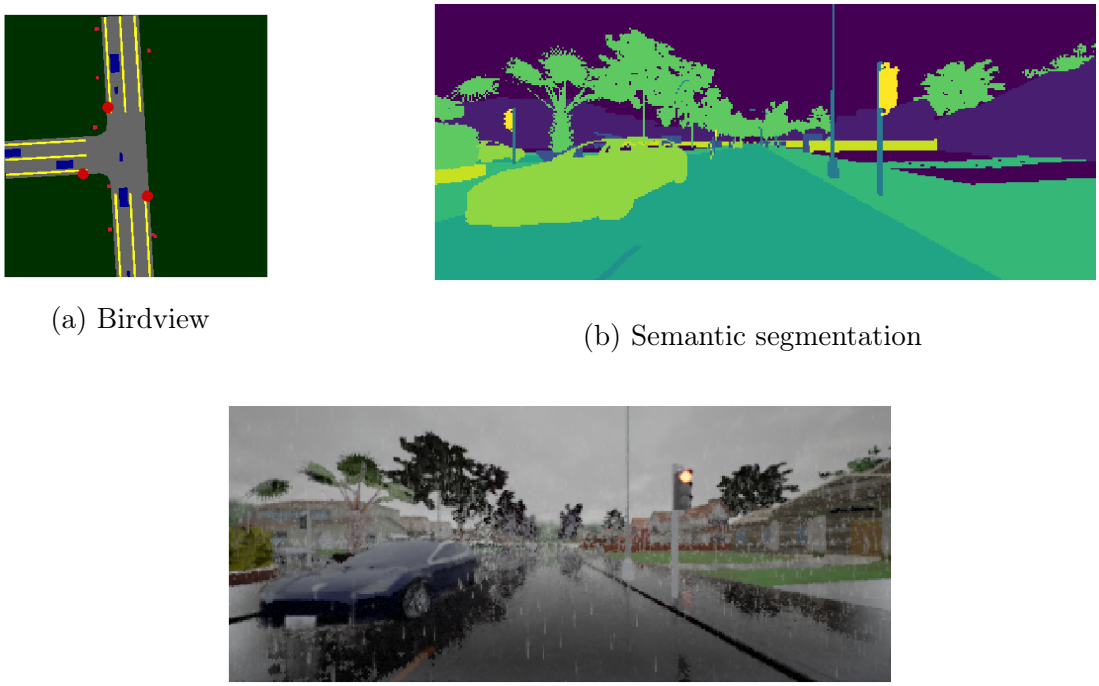
This section gives an overview of the method used in this research. The approach taken is closely inspired by the work from "Learning by Cheating" by Chen, Zhou, et al. (2019) to be able to best inspect the effect of replacing the source of the cheat data — from bird's-eye view to segmented data. Their method is therefore first described, followed by the significant changes we made.

4.1 Learning by Cheating

Chen, Zhou, et al. (2019) propose a method that tries to train a sensorimotor agent which predicts a steering, throttle and brake value given a monocular RGB image, the vehicle's speed, and a high-level navigation command, which guides the agent towards the goal, as input. They use the CARLA simulator (Dosovitskiy et al., 2017), as explained in more detail in Section 5.1, as the research environment. To solve the problem, their approach consists of two steps: first, training a privileged agent (teacher), and then training the sensorimotor agent (student) with supervision from the teacher. This approach is a form of knowledge distillation, as described in Section 2.3. Their approach is validated on two benchmarks: *CoRL2017* (Dosovitskiy et al., 2017) and *NoCrash* (Codevilla, Santana, et al., 2019). The next subsections describe each step of their approach in more detail.

4.1.1 Teacher training

The teacher has access to extensive simulation information. This information includes a ground-truth map from a bird's-eye view perspective, subsequently referred to as birdview. The birdview, as shown in Figure 4.1a, is a segmentation image that includes binary masks for lanes, traffic lights and their state, vehicles



(a) Birdview

(b) Semantic segmentation

(c) RGB

Figure 4.1: Overview of an example frame of the dataset.

and pedestrians in the vicinity of the agent. The teacher therefore only has to learn to act in the world, and not to see. Three channels of the birdview represent the traffic lights: one channel for green lights, one for yellow lights, and one for red lights. In addition to the birdview, the teachers' speed and a high-level navigation command such as *Follow*, *Left*, *Right*, and *Straight* are provided too.

A CNN is used to train the teacher by conditional imitation learning (CIL) (Codevilla, Müller, et al., 2018). The network consists of four heads, a multi-head configuration as explained in Subsection 2.2.1 and shown in Figure 4.2, which each correspond to one of the possible navigation commands — the four conditions. Each head outputs for each predicted waypoint a heatmap. The heatmaps are then converted to waypoints in the agent's reference frame. The ground-truth labels are based on the agent's future positions and are transformed back as waypoints into the agent's current reference frame. An Adam optimizer (Kingma and Ba, 2017) aims to minimize the L_1 distance between the network's predictions and these ground-truth waypoints. Data augmentation in the form of rotating and shifting are applied to the birdviews during training to let the agent learn to recover in case it for example faces a sidewalk.

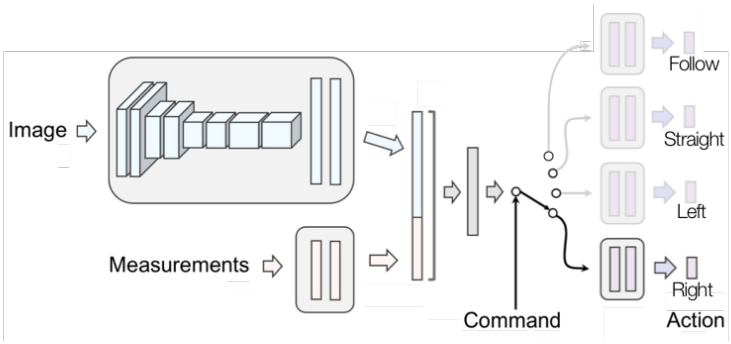


Figure 4.2: Multi-head configuration used in this study. Image altered from Codevilla, Müller, et al. (2018).

4.1.2 Student training

The student has to learn to mimic the behaviour of the teacher. However, the student does not have access to the privileged information the teacher has, but only to the vehicle's speed, a navigation command, and a monocular RGB image from a forward-facing 90° field of view (fov) RGB camera. Therefore, the student has to learn to see and act in the world. A similar, to the teacher, network with a CNN structure is used for the student which predicts waypoints in its image reference frame. To align the teacher's and student's reference frames, the teacher waypoints are transformed to the image reference frame too.

Training the student is done in two stages. First, the same trajectories used to train the teacher are used, but now with the supervision of the teacher on all branches in parallel. This means that the student learns what it should do if it is turning right and should suddenly turn left. Second, the student is rolled out in the environment and trained via DAgger, as explained in Section 2.5, which uses the teacher as an oracle. During both stages, the L_1 distance between the student's prediction and the teacher's predictions is minimized. Multiple data augmentations such as pixel dropout and colour perturbations are applied to the student's RGB input.

4.1.3 Validation

To validate the student, its waypoints are converted to a steer, throttle and brake value by two, lateral and longitudinal, low-level proportional integral derivative (PID) controllers. The longitudinal controller aims to calculate the throttle and brake value to match the target velocity of the waypoints. The target velocity is calculated based on the L_2 distance between all predicted waypoints. If the current

speed is larger than the target speed, the student brakes. The lateral controller predicts the steering angle to steer towards the student's prediction. This is done by fitting an arc through the predicted waypoints and steering towards a point on this arc. Both PID controllers contain parameters that determine the steering and acceleration behaviour. Chen, Zhou, et al. (2019) tune these parameters on a small subset of trajectories.

4.2 Cheating by Segmentation

As mentioned before, we propose changing the source of the cheat from birdview data to semantic segmentation data. This section, therefore, explains the consequences of this change to Chen, Zhou, et al. (2019)'s method.

4.2.1 Data

In contrast to "Learning by Cheating", we use a ground-truth semantic segmentation image from a forward-facing camera with 120° fov. The 13 segmentation classes, as explained in Section 2.4, are provided by CARLA. This data is preprocessed by converting all classes to a per-channel binary mask, resulting in a segmentation image with 13 channels. Figure 4.1b shows an example of a segmentation image. In contrast to Chen, Zhou, et al. (2019), we just use one channel for the traffic lights. This channel is only active if the traffic light in the vicinity of the agent is red. Otherwise, the traffic-light channel is empty — all zeros.

4.2.2 Training

Since we use different data, our training procedure has some significant changes. The changes to both steps of the training method will therefore be explained in more detail below.

Teacher

Our teacher has no access to a birdview, but to a segmentation image which is preprocessed as described in the previous subsection. The teacher predicts waypoints in the reference frame of the segmentation image. Therefore, this reference frame is already aligned with the student's RGB reference frame. Despite the perfectly aligned frames, the ground-truth labels are calculated differently. In principle, the intuition stays the same: future locations of the agent serve as the label for the current frame. However, some future locations may be out of sight for the current agent, due to the perspective change of the segmentation data. This is for

example the case in tight turns: the complete corner may not be visible from the driver's perspective if it is at the start of the corner. A top-down view, such as the birdview, does oversee the complete corner, so a point further down the corner is always visible in the current view. In the case of our segmentation data, some future locations may fall outside the image reference frame after re-projection. Therefore, we propose a new ground-truth labelling algorithm as shown in Algorithm 1.

With this new algorithm, the algorithm assures that there are always 5 ground-truth labels per frame. In addition, the new algorithm takes into consideration that red traffic lights — and other immediate breaking situations — should always yield a full stop label. In the case of "Learning by Cheating", this does not always hold. For example, if the agent is in front of a red traffic light, but the light will turn green in the next frame such that the agent will move in the next frame as well. Chen, Zhou, et al. (2019)'s algorithm would yield labels representing a moving agent in this situation, despite the red light. Our new algorithm does yield waypoints that represent a full stop due to the red light.

Moreover, the rotating and shifting augmentations as applied by Chen, Zhou, et al. (2019) do not yield the same desired result on our segmentation data. This is again due to the changed perspective, since a rotation (around the image centre) of the data does not let the agent face a sidewalk, but represents a skewed world. Therefore, noise is used for the agent that creates the dataset, as explained in Section 5.2. This has the desired effect of trajectory perturbations and lets the agent learn to recover from such situations.

Student

Besides changes to the teacher training method, the student training procedure has changed too. The student's RGB image has changed from a 90° fov to 120° fov image. This is an important change, since the fov of the teacher and student should align. If these are not aligned, point correspondence from the teacher image to the student image is difficult, since some points that are visible in the 120° fov frame are not visible in the narrower 90° fov frame.

$$\begin{bmatrix} x \\ y \end{bmatrix}_{model} = \frac{1}{2 \begin{bmatrix} x \\ y \end{bmatrix}_{image}} - 1 \tag{4.1}$$

With the perfectly aligned teacher and student frames, which both have the same perspective, the L_1 distance between the teacher's and student's predictions

Algorithm 1 Waypoint algorithm

```
function getWaypoints(frame, gap=5, n_step=5, buffer=40)
    tl ← getTrafficLightState(frame)

    if tl = red then
        goto projectVehicle

    waypoints ← {}
    for i in range(frame, n_step + 1 + buffer × gap, gap) do
        x, y, z ← getVehicleLocation(i)
        x_image, y_image, z_image ← worldToImage([x  y  z])
        if x_image, y_image, z_image in image frame then
            waypoints ← waypoints + {x_image, y_image, z_image}

    if len(waypoints < 2) then
        if gap = 5 then
            return getWaypoints(frame, gap=1)
        else
            goto projectVehicle

    if len(2 ≤ waypoints < 5) then
        goto interpolateWaypoints(waypoints)

    return waypoints

function projectVehicle(vehicle_location, vehicle_orientation)
    forward_vector ← getForwardUnitVec(vehicle_orientation)
    vehicle_location ← vehicle_location + 4 × forward_vector
    return worldToImage(vehicle_location)

function interpolateWaypoints(waypoints, degree=2)

    if len(waypoints == 2) then
        degree ← 1

    waypoints ← {}
    while len(waypoints < 5) do
        waypoints ← waypoints + interpolatePolyFit(degree, waypoints)

    return waypoints
```

are measured in model output space. This space is in the interval $[-1, 1]$ and is related to the image space. The image space can be converted to the model output space as shown in Equation 4.1.

This chapter has described our approach which changes birdviews from Chen, Zhou, et al. (2019)'s method to semantic segmentation data and proposes an improved waypoint algorithm. To be able to test the approach, we conduct a variety of experiments. Chapter 5 describes the experimental setup of these experiments, and Chapter 6 provides the results.

Chapter 5

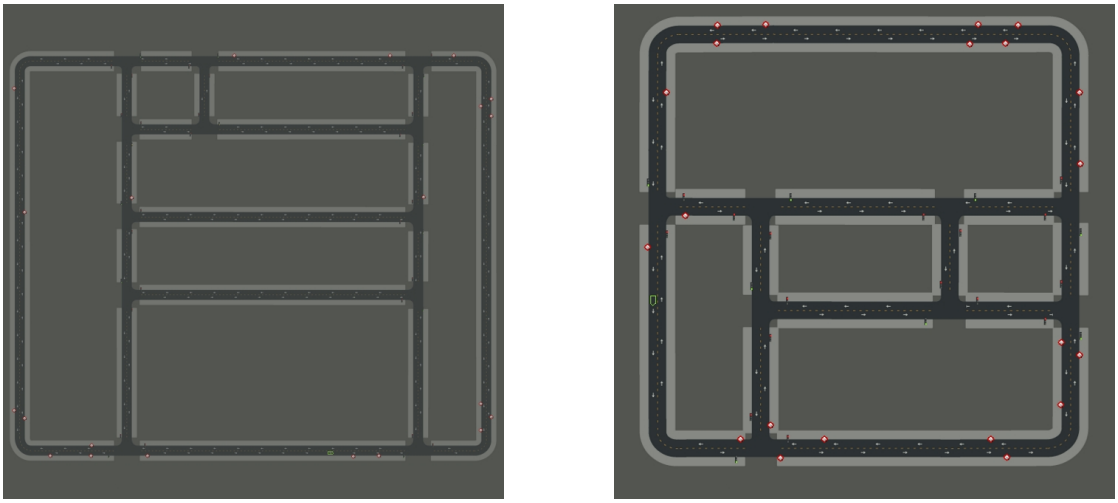
Experimental setup

The method described in the previous chapter has some specific implementation details. This chapter will describe these details, mention used software, and explains the validation benchmarks.

5.1 CARLA

In this research CARLA version 0.9.6 is used as simulator (Dosovitskiy et al., 2017). CARLA is an advanced and extremely realistic simulation software that includes other traffic, photo-realistic visualizations, different environments, and great flexibility. In addition, navigation planners are provided to direct an agent from one point to a certain goal. This planner provides the agent with high-level navigation commands which can guide the agent to a goal. CARLA's great flexibility is reflected by the number of parameters to tune the weather, the number and type of traffic agents, and the wide range of sensors, such as an RGB camera and a segmentation camera, available with each their own configurations.

Out of the current 8 available towns in CARLA, this research uses two towns, Town 01 and Town 02, to comply with the method and benchmarks of Chen, Zhou, et al. (2019). Our previous work has used other towns Town 3, Town 6 and Town 10HD (Van Orden and A. Visser, 2021). The layout of the towns used in this study is shown in Figure 5.1. Town 01 is a basic town which includes traffic lights, T-junctions and long straight roads. Town 02 is a smaller version of Town 01. This results in relatively more junctions or turns in Town 02 than in Town 01. In other words, the density of difficult traffic situations such as junctions is higher in Town 02 than in Town 01.



(a) Town 01 (b) Town 02

Figure 5.1: Layout of both CARLA towns used in this study.

As Chen, Zhou, et al. (2019) have mentioned, CARLA underwent significant changes to its rendering engine. Therefore, results on older versions of CARLA are not directly comparable. Moreover, Chen, Zhou, et al. (2019) state that the pedestrian simulation in CARLA 0.9.6 lacks real-world representation since they do not cross roads. Therefore, the proposed adjustments to the CARLA API Client[1] by Chen, Zhou, et al. (2019) — such that pedestrians do cross the road — are used in this study too. An example of a pedestrian crossing the road is shown in Figure 5.2.

5.2 Dataset

The dataset is created by using an agent, subsequently also referred to as ego-vehicle and implemented by Chen, Zhou, et al. (2019), that follows a set of pre-defined routes in Town 01. This agent autonomously drives from waypoint to waypoint by controlling the vehicle with two PID controllers. Instead of always following the waypoints, noise $\sim U(-0.20, 0.20)$ is used as the steering command of the agent for every 10 in 110 frames. This noise lets the agent steer a bit too far to the left or right. Consequently, the agent could then face a sidewalk on the side of the road and learns to correct. The dataset is collected in Town 01 under train weather conditions only, as defined by the benchmarks in Section 5.4. This set is split into a training and validation set, resulting in 80% train frames and

[1] https://github.com/dotchen/LearningByCheating

Figure 5.2: The pedestrian modification by Chen, Zhou, et al. (2019) to CARLA results in pedestrians crossing the road.

20% validation frames. The resulting training set consists of 167k frames out of 142 driven routes (episodes), and the validation set consists of 39k frames out of 33 episodes. The simulator is set to 10 fps with synchronous mode enabled, to ensure no frames are missing.

The ego-vehicle has two cameras mounted on top of its roof: an RGB and a semantic segmentation camera. Both cameras have a 120° fov and are placed in the exact same location and in the same orientation. The RGB camera has an 384×160 resolution, while the semantic segmentation camera's resolution is 192×80. The segmentation image can be smaller, since the image space is significantly smaller too, in comparison to the RGB image space, as explained in Section 2.4.

At every step, the following data are saved as a frame: an RGB image; a semantic segmentation image; ego-vehicle world position and rotation; camera world position and rotation; ego-vehicle speed; high-level navigation command; and traffic light state. Positions and rotations are given in simulation-world coordinates as x, y, z and $pitch, yaw, roll$, respectively. The navigation command is provided by CARLA and reflects the predefined routes. The commands are based on the map layout and follow traffic rules such as possible turns to take from a certain lane.

Figure 4.1 shows an example frame containing the original birdview (4.1a) — used as cheat by Chen, Zhou, et al. (2019), a semantic segmentation image (4.1b) — our proposed new cheat, and the corresponding RGB image (4.1c).

5.3 Training

The teacher uses a ResNet-18 backbone and is trained with a batch size of 64 and decreasing learning rate starting from $1e^{-3}$. To prevent overfitting, pixel dropout, motion blur and an affine transformation are used as augmentations. The augmentations are applied to the segmentation image as well as to the label waypoints.

The student has due to its larger input image space, the RGB image as explained in Section 2.4, a larger backbone. The ResNet-34 (He et al., 2015) backbone is pre-trained on ImageNet (Deng et al., 2009). The first training step of the student is done with a batch size of 96 and a fixed learning rate of $1e^{-4}$. The second step, DAgger, is configured to do six iterations where each iteration creates 4000 frames. Per iteration, the student model is trained for 20 epochs with batch size 120 and a fixed learning rate of $1e^{-4}$ too.

5.4 Benchmarks

The same benchmarks as Chen, Zhou, et al. (2019) are used to validate our agents: *CoRL2017* (Dosovitskiy et al., 2017) and *NoCrash* (Codevilla, Santana, et al., 2019). Both benchmarks consist of training and testing conditions with additional constraints. The training weathers apply to both benchmarks and are *ClearNoon*, *WetNoon*, *HardRainNoon*, and *ClearSunset*. Town 01 is used as training town and Town 02 for testing in both benchmarks. Benchmark-specific subjects are explained in the next subsections.

5.4.1 CoRL2017

CoRL2017 is the original CARLA benchmark and was introduced with "CARLA: An Open Urban Driving Simulator" (Dosovitskiy et al., 2017). The benchmark consists of four tasks with increasing difficulty: *Straight*, *One turn*, *Navigation*, and *Navigation dynamic*. Each task has 25 predefined routes with a start position and end goal. *Navigation*'s routes contain multiple turns and *Navigation dynamic* adds traffic — vehicles and pedestrians — to the environment. A trial is considered a success if the agent reaches the goal within a time limit, which is calculated based on the distance to the goal and a target speed of 5 km/h. Violating a red light or colliding does not terminate a trial, but could result in a time-out — failure. *WetCloudyNoon* and *SoftRainSunset* are used as test weather.

5.4.2 NoCrash

More challenging scenarios are provided by the *NoCrash* benchmark, introduced as a more complex benchmark than CoRL2017 by Codevilla, Santana, et al. (2019) in *Exploring the Limitations of Behavior Cloning for Autonomous Driving*. It tests the agent's ability in dynamic traffic situation with traffic lights and other traffic. Three tasks — *Empty*, *Regular*, and *Dense* — with each 25 predefined episodes make up the benchmark. The tasks increase in the number of dynamic agents such as pedestrians and vehicles, where *Empty* has no dynamic agents. In contrast to *CoRL2017*, an episode is considered a failure if the agent collides with a force higher than some threshold. Traffic lights may be violated, but the percentage of traffic lights ran is reported. To succeed in an episode, the agent should reach the end goal within a time limit and not collide. Test weathers are *WetSunset* and *SoftRainSunset*.

Due to CARLA's non-determinism, each task of the *NoCrash* benchmark is run multiple times, which is in accordance with Chen, Zhou, et al. (2019)'s validation. Moreover, in both benchmarks, all tasks under train weather conditions are run 100 times maximum, and 50 times maximum under test weather. We used seed 0 for the *CoRL2017* benchmark and seeds $0, 1, 2$ for the *NoCrash* benchmark, which is in accordance with Chen, Zhou, et al. (2019). The target speed of the agent is in both benchmarks clipped at 5 km/h. The next chapter reports the results of all experiments.

Chapter 6

Results

The results are split into two parts: the privileged teacher agent (CBS Teacher) and the RGB-student agent (CBS Student). In this way, the performance of purely the teacher, and the student-teacher approach to transfer the teacher's knowledge to the student can be inspected. Furthermore, the results of both agents are compared to Chen, Zhou, et al. (2019)'s, subsequently referred to as LBC. Appendix Table A.1 compares our results to competitive other approaches that used older versions of CARLA, to put our work in a wider context.

6.1 CBS Teacher

Table 6.1 shows the performance of the CBS Teacher on the *CoRl2017* benchmark, compared to the LBC Teacher. We achieve nearly as perfect results as the LBC Teacher, by solving 7 out of the 16 tasks and success rates above 95% for almost all remaining tasks. To test the agent in more difficult scenarios, the agent is also validated on the *NoCrash* benchmark and compared to LBC and CARLA's autopilot (AT), as shown in Table 6.2.

From the success rates in Table 6.2 it is clear that on all tasks the AT outperforms, or scores equal to, LBC Teacher and our CBS Teacher. The AT is rule-based and contains the expertise of experts on the most common traffic situations, so this out-performance is to be expected. However, even for the AT, increasing the number of traffic agents — vehicles and pedestrians — leads to a drastic decrease in performance. Moreover, AT's performance decreases by up to 30% when generalizing to the test town in Dense traffic. This shows that the test town is indeed a difficult town to solve, which is reflected by the LBC Teacher's decrease of 44% too.

Table 6.1: Success rates (higher is better, best scores are marked bold) of our CBS Teacher compared to the LBC Teacher on *CoRL2017* benchmark in train and test town. LBC numbers are taken from their GitHub[1], but not all numbers are reported.

Task	Train town Weather	LBC Teacher	CBS Teacher	Test town LBC Teacher	Test town CBS Teacher
Straight	train	–	**100**	**100**	97
One turn	train	–	**100**	**100**	97
Navigation	train	**100**	**100**	**99**	98
Navigation dynamic	train	**100**	99	**100**	**100**
Straight	test	–	**100**	**100**	98
One turn	test	–	**96**	**100**	96
Navigation	test	**100**	**100**	**100**	90
Navigation dynamic	test	**100**	98	**100**	**100**

A possible explanation of this decrease is the relatively higher number of complex situations, such as junctions, in the test town, Town 02, in comparison to the train town, Town 01, as described in Section 5.1. However, we are, like the other approaches, able to solve the empty town under the most difficult generalization conditions: test town with test weather. In comparison to the LBC Teacher, our CBS Teacher seems to suffer more from additional traffic.

Table 6.2: Success rates (mean and standard deviation over three runs) of our CBS Teacher on *NoCrash* benchmark in train town and test town compared to the LBC Teacher. Higher is better, best scores are marked bold. AT: CARLA's built-in autopilot, numbers taken from (Chen, Zhou, et al., 2019).

Task	Weather	Train town AT	Train town LBC Teacher	Train town CBS Teacher	Test town AT	Test town LBC Teacher	Test town CBS Teacher
Empty	train	**100** $\pm$ 0	**100** $\pm$ 1	90.7 $\pm$ 1.5	**100** $\pm$ 0	**100** $\pm$ 0	99.3 $\pm$ 0.6
Regular	train	**99** $\pm$ 1	96 $\pm$ 3	81.3 $\pm$ 6.5	**99** $\pm$ 1	95 $\pm$ 1	53.3 $\pm$ 3.1
Dense	train	**86** $\pm$ 3	80 $\pm$ 5	37.7 $\pm$ 2.1	**60** $\pm$ 3	46 $\pm$ 8	9.3 $\pm$ 4.0
Empty	test	**100** $\pm$ 0	**100** $\pm$ 0	96 $\pm$ 2.0	**100** $\pm$ 0	**100** $\pm$ 0	**100** $\pm$ 0.0
Regular	test	**99** $\pm$ 1	97 $\pm$ 3	77.3 $\pm$ 4.2	**99** $\pm$ 1	93 $\pm$ 2	58.7 $\pm$ 8.1
Dense	test	**83** $\pm$ 6	81 $\pm$ 6	36.7 $\pm$ 3.1	**59** $\pm$ 6	45 $\pm$ 10	6.0 $\pm$ 0.0

To substantiate this, a collision analysis is conducted. Figure 6.1 reports the number of collisions per 10 driven kilometres, grouped by traffic intensity, for the *CoRL2017* and *NoCrash* benchmark. *CoRL2017* only has one task with traffic, *Navigation dynamic*, which is comparable to the *Regular* task from *NoCrash*. Therefore, no number is reported for *CoRL2017* in Dense traffic intensity. The analysis shows a clear positive relationship between the traffic intensity and the relative number of collisions, with nearly zero collisions in empty towns. This holds for both benchmarks, which therefore indicates that our teacher drives similarly during both benchmarks, but since a collision in *NoCrash* fails the task, it scores lower on *NoCrash*. Hence, the *NoCrash* benchmark proves to be more difficult than the *CoRL2017* benchmark.

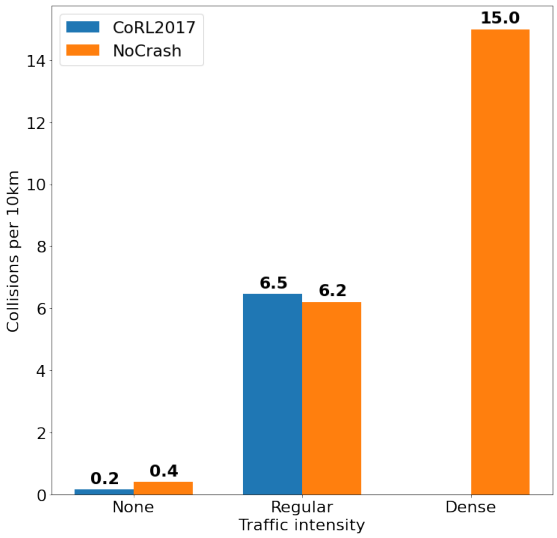


Figure 6.1: Number of collision of CBS Teacher per 10 driven kilometers over all *NoCrash* and *CoRL2017* trajectories, grouped by traffic intensity.

Yet, the generalization performance of our approach in terms of weather is excellent. In almost all tasks we achieve higher performance under test weather than under train weather, in both the train and test town. However, generalization in terms of different towns remains difficult, especially in scenarios with increased traffic such as *Dense*.

6.2 CBS Student

The CBS Student has been trained in two steps, as explained in Equation 4.2.2. Without DAgger training, the student agent will be referred to as CBS Student, and after 6 iterations of DAgger as CBS Student*. The following section, therefore, validates the agent after each step. Validation is done on the *CoRL2017* and *NoCrash* benchmark.

6.2.1 CoRL2017

Table 6.3 compares the CBS Student to Chen, Zhou, et al. (2019)'s student agent without DAgger (LBC Student) on the *CoRL2017* benchmark. In this context, our agent yields competitive results in comparison to the LBC Student. Generalization capabilities are similar to our CBS Teacher, with especially good performance in weather generalization. However, the gap between the CBS Teacher and the CBS Student is larger than the gap between LBC Teacher and LBC Student.

Table 6.3: Success rates (higher is better, best scores are marked bold) on *CoRL2017* benchmark in train and test town. LBC Student* numbers are taken from their GitHub[1].

		Train town				Test town			
Task	Weather	LBC Student	LBC Student*	CBS Student	CBS Student*	LBC Student	LBC Student*	CBS Student	CBS Student*
Straight	train	100	**100**	97	**100**	**100**	**100**	**100**	**100**
One turn		96	**100**	57	78	95	**100**	62	66
Navigation		94	**100**	48	80	94	**98**	52	57
Nav. dynamic		95	**100**	49	76	88	**99**	53	55
Straight	test	100	**100**	**100**	98	**100**	**100**	**100**	**100**
One turn		100	**96**	60	64	98	**100**	64	40
Navigation		98	**100**	48	68	98	**100**	34	28
Nav. dynamic		92	**96**	52	64	90	**100**	34	24

To close this gap, DAgger is applied and the resulting CBS Student* is validated on the same *CoRL2017* benchmark as well, as shown in Table 6.3 too. To provide a fair comparison, Chen, Zhou, et al. (2019)'s student agent with DAgger is reported as LBC Student*. CBS Student* significantly outperforms CBS Student on all tasks, except for the test town with test weather conditions. Even on the most difficult task in terms of traffic, *Navigation dynamic*, CBS Student* shows

an improvement of up to 27 percentage points over CBS Student. Notice that our CBS Student$^\bigstar$ does not outperform LBC Student$^\bigstar$. This reflects the behaviour of our CBS Teacher compared to the LBC Teacher, where the latter Teacher can make use of the bird's-eye view perspective which provides more depth information than the CBS Teacher's segmentation data perspective.

Ablation study: DAgger

Since DAgger did not increase performance in all scenarios, we conduct an ablation study, to inspect the effect of the number of DAgger iterations on the performance. The study is shown in Figure 6.2 and reports the success rate on *CoRL2017*'s *Navigation dynamic* task under test weather in Town 01 (Train) and 02 (Test) after multiple iterations. In this way, it isolates the generalization performance due to DAgger. The study shows that DAgger increases the success rate by up to 35 percentage points in train conditions. There seems to be a positive relationship between the number of iterations and the success rate in the train conditions. However, this does not hold for the test conditions. In the test scenario, DAgger seems to converge to a plateau that is lower (24%) than before DAgger (34%). Hence, DAgger might be sensitive to overfitting which could explain the diverging lines in Figure 6.2.

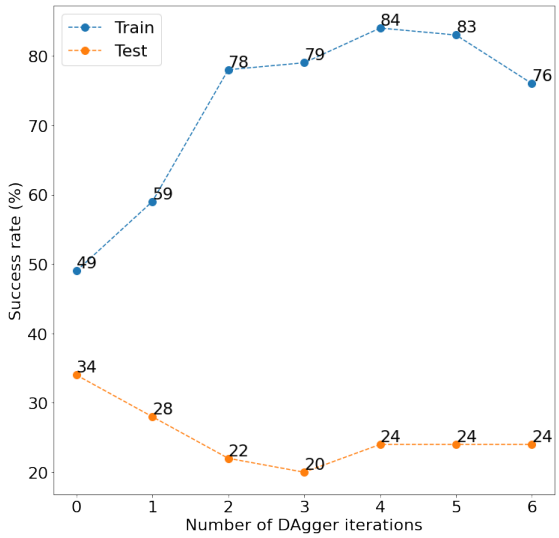


Figure 6.2: Ablation study of the effect of the number of DAgger iterations on the success rate. Validated on the *Navigation dynamic* task of *CoRL2017* benchmark in Train: Town 01 with test weather, and Test: Town 02 with test weather.

6.2.2 NoCrash

Despite the ablation study indicating DAgger to overfit, the CBS Student* is validated on the *NoCrash* benchmark, to provide a fair comparison to the LBC Student*, as shown in Table 6.4. The claim that the *NoCrash* benchmark is more difficult than the *CoRL2017* benchmark is additionally supported by the, on average, lower performance of both CBS Student* and LBC Student* on *NoCrash* in comparison to *CoRL2017*. The poor generalization performance of CBS Student* to new towns is substantiated by the scores in the test town tasks of *NoCrash* too. However, besides generalization performance, CBS Student* also struggles more in train conditions here, than it did on *CoRl2017*. This might be explained by the number of collisions that result in a failure on *NoCrash*.

Table 6.4: Success rates (mean and standard deviation over three runs) of student agents on *NoCrash* benchmark in train town and test town. Higher is better, best scores are marked bold.

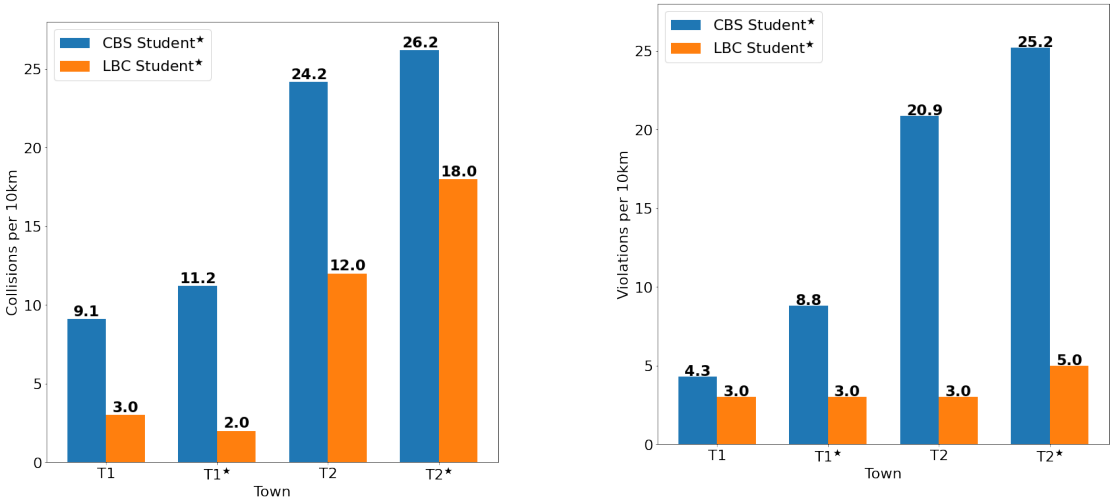
		Train town		Test town	
Task	Weather	LBC Student*	CBS Student*	LBC Student*	CBS Student*
Empty	train	**97** $\pm$ 1	58.7 $\pm$ 0.6	**100** $\pm$ 0	28.3 $\pm$ 0.6
Regular	train	**93** $\pm$ 1	40.7 $\pm$ 8.5	**94** $\pm$ 3	11.3 $\pm$ 2.3
Dense	train	**71** $\pm$ 5	10 $\pm$ 1.0	**51** $\pm$ 3	0.7 $\pm$ 0.6
Empty	test	**87** $\pm$ 4	28.0 $\pm$ 3.5	**70** $\pm$ 0	8.0 $\pm$ 0
Regular	test	**87** $\pm$ 3	12.7 $\pm$ 2.3	**62** $\pm$ 2	5.3 $\pm$ 1.2
Dense	test	**63** $\pm$ 1	2.0 $\pm$ 0.0	**39** $\pm$ 8	0.7 $\pm$ 1.2

To give more insight into this number, an infraction analysis is performed on the *NoCrash* trajectories. Figure 6.3 shows the relative number of traffic light violations and collision per 10 driven kilometres. Notice that Chen, Zhou, et al. (2019)'s reports infractions for the CARLA 0.9.5 environment, so the number of infractions might differ in version 0.9.6. Our reported infraction numbers are calculated by taking the average over all three tasks in the specific town and under the train and test weather. The analysis provides three main insights.

First, the number of collisions for CBS Student* is significantly higher than CBS Teacher's (Figure A.1a), which shows that the student does collide even more than our CBS Teacher. This reflects that the teacher's knowledge is not completely transferred to the student. Second, the LBC Student*'s collisions are increased by more than a factor 4 when generalizing to the new town. Our CBS Student*'s collisions are higher in general but are only increased by a factor of 2 in the new town. Third, CBS Student*'s traffic light violations are of the same magnitude

as its number of collisions, which substantiates the student's low capabilities for all immediate braking actions — stopping for a red traffic light, vehicle, pedestrian, or other close objects. In addition, our number of violations increases when generalizing to new weather or a new town. This indicates that generalization remains difficult and that DAgger might play an important role in generalizing performance, as also shown by the ablation study. In particular, the parameter configuration of DAgger might affect this performance.

This chapter has provided the results of our experiments where the CBS Teacher performs excellently on the *CoRL2017* benchmark and is competitive to LBC Teacher on the *NoCrash* benchmark. In particular, the CBS Teacher has difficulties with dense traffic scenarios. Moreover, the CBS Student achieves competitive scores on both benchmarks too, but a gap between CBS Teacher and CBS Student remains. An ablation study on DAgger suggests the possibility of overfitting the CBS Student*. The infraction analysis substantiates this, while we still improve relative generalization to new environments in terms of collisions by a factor of 2 in comparison to LBC Student*. The next chapter will interpret and discuss the results.



(a) Number of collisions per 10 km.

(b) Number of traffic light violations per 10 km.

Figure 6.3: Infraction analysis on *NoCrash* trajectories. T1: Town 01 with train weather. T1*: Town 01 with new (test) weather. T2: Town 02 with train weather. T2*: Town 02 with new (test) weather.

Chapter 7

Discussion

The results from the previous chapter show a couple of key findings. This section gives an interpretation of these findings, reports their limitations and provides recommendations for future work. The findings are reported in order, based on their importance in terms of impact on performance.

7.1 Empty versus dense scenarios

The student-teacher approach in combination with DAgger and the semantic-segmentation teacher has proven to yield successful scores in basic scenarios such as empty towns. Increasing the traffic or number of turns in the environment lets the agent's performance decrease. We show that this decrease can be explained by the larger number of collisions. Since Chen, Zhou, et al. (2019) use a birdview with 360° view, it provides more information of its surroundings to the vehicle. The semantic-segmentation data only provides a forward-looking 120° view. Hence, this could explain the difference in the number of collisions.

7.2 Knowledge transfer

The student's performance seems sensitive to the teacher's domain. This means that the teacher's training domain should cover as many situations as possible that the student might face too. For example, if no noise would be injected into the agent creating the dataset, the dataset would only consist of perfectly driven routes. This could cause the teacher to not learn how to recover from facing a sidewalk. Hence, the student's ability to do this could result to be low too. In particular, the generalization performance in terms of town of our CBS

Teacher substantiates this, in combination with the relatively low performance of CBS Teacher in dense traffic. This performance is indeed reflected by the CBS Student*' abilities which generalize poorly too and scores low in dense traffic as well. However, this only holds for the *NoCrash* benchmark, since our CBS Teacher almost solves all tasks in the *CoRl2017* benchmark, but CBS Student* does not. Therefore, there also seems to still be a large gap between student and teacher regarding knowledge transfer.

7.3 DAgger

DAgger has shown to narrow this gap and improve performance significantly. In particular, DAgger increases performance in the conditions it was applied to. This means that DAgger applied during training, so under training weather and in the training town, mainly boosts the performance in both benchmarks in this same training town with training weather. Therefore, DAgger seems to not be able to greatly widen the student's domain. In contrast to our results, Chen, Zhou, et al. (2019)'s results show that DAgger also improves performance in terms of town generalization. This might be due to the configuration of the DAgger parameters, with in particular the ratio of the number of training epochs per iteration and number of iterations. We used 20 epochs with in total 6 iterations. Chen, Zhou, et al. (2019) do not mention their DAgger configuration.

7.4 Future work

DAgger possibly overfitting the student on training conditions in combination with low teacher's capabilities in dense traffic could explain our hypothesis of an often colliding agent. Therefore, future work could research in the direction of the privileged teacher agent to improve this teacher's capabilities and prevent DAgger to overfit the student.

However, we expect the semantic-segmentation data not to cause the decrease in performance of the teacher, since the information available in Chen, Zhou, et al. (2019)'s birdview and our segmentation image is nearly identical. Despite this, the perspective change might result in more short-term behaviour, which could cause lower performance, instead of long-term behaviour. With short-term behaviour, reacting to the world in front of the vehicle is meant, such as steering a bit to the left because a vehicle is approaching. Long-term behaviour means anticipating on

the part of the road that is still quite far away from the vehicle. The birdview's perspective might favour long-term behaviour of the model since objects further away do not appear smaller on the birdview. This is in contrast to the more natural segmentation's perspective which visualizes object further away smaller, as human eyes do too.

Hence, future research could add depth maps to the teacher's input, since Zhou, Krähenbühl, and Koltun (2019) have shown depth maps to yield the second-best improvement to urban-driving, next to segmentation images. In addition, the gap between student and teacher might be further closed by applying a teacher-assistant as proposed by Mirzadeh et al. (2019). The authors state that if the gap between student and teacher is too large, the student's performance breaks down. Therefore, an intermediate network (teacher-assistant) is proposed to fill the gap between the student and teacher. Moreover, as our results have shown, DAgger tends to decrease generalization to new towns which is in contrast to the finding of Chen, Zhou, et al. (2019). Implementing improvements upon DAgger, as proposed by Prakash et al. (2020), might improve generalization. Furthermore, tuning the parameters of DAgger might also improve generalization performance.

Chapter 8

Conclusion

This thesis has proposed Cheating by Segmentation: a student-teacher approach with semantic segmentation data to learn autonomous driving behaviour. The essence of this approach is the use of more realistic data to train a teacher which transfers its knowledge to a student. With this approach, the thesis aimed to answer the following research question: *How is the performance of Chen, Zhou, et al.'s method affected when a ground-truth segmentation view is used instead of a birdview?* In addition, by validating the teacher and student separately, this study aimed to answer the sub-questions: *What is the effect of the new data on the teacher's performance?*, and *To what extent can the new teacher transfer its knowledge to the student?*, as well.

In conclusion, this thesis has shown that Chen, Zhou, et al. (2019) method's birdview can be replaced by a more realistic semantic segmentation image from a driver's perspective. With this more realistic data, our teacher is able to solve the *Empty* task on the *NoCrash* benchmark and scores 95% success rates or higher on almost all tasks of *CoRl2017*. Additionally, this thesis has demonstrated that the teacher's ability to transfer its knowledge to the student in combination with DAgger is proven to be moderate, but competitive to Chen, Zhou, et al. (2019)'s capabilities. Therefore, a student can learn its driving behaviour, for in particular train conditions, from a good teacher. The effect on the performance when changing Chen, Zhou, et al. (2019)'s method to use semantic segmentation data is a decrease in performance in dense traffic, but similar performance in empty scenarios. In dense traffic, the teacher insufficiently learns actions that require immediate braking, which subsequently has its effect on the learned behaviour of the student as well. One of the suggestions in future work is to provide the teacher with depth information, which could improve its immediate breaking behaviour. With this thesis, the first steps are set towards training autonomous driving student-teacher approaches with more realistic data.

Bibliography

[1] Mariusz Bojarski et al. *End to End Learning for Self-Driving Cars*. 2016. arXiv: 1604.07316 [cs.CV].

[2] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. "Model Compression". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. Philadelphia, PA, USA: Association for Computing Machinery, 2006, pp. 535–541. ISBN: 1595933395. DOI: 10.1145/1150402.1150464. URL: https://doi.org/10.1145/1150402.1150464.

[3] CBS. *Hoeveel mensen komen om in het verkeer?* https://www.cbs.nl/nl-nl/visualisaties/verkeer-en-vervoer/verkeer/hoeveel-mensen-komen-om-in-het-verkeer-. n.d.

[4] Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. "Learning to drive from a world on rails". In: *arXiv preprint*. 2021.

[5] Dian Chen, Brady Zhou, et al. "Learning by Cheating". In: *CoRR* abs/1912.12294 (2019). arXiv: 1912.12294. URL: http://arxiv.org/abs/1912.12294.

[6] Felipe Codevilla, Matthias Müller, et al. *End-to-end Driving via Conditional Imitation Learning*. 2018. arXiv: 1710.02410 [cs.RO].

[7] Felipe Codevilla, Eder Santana, et al. *Exploring the Limitations of Behavior Cloning for Autonomous Driving*. 2019. arXiv: 1904.08980 [cs.CV].

[8] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[9] Ernst Dieter Dickmanns et al. "The seeing passenger car'VaMoRs-P'". In: *Proceedings of the Intelligent Vehicles' 94 Symposium*. IEEE. 1994, pp. 68–73.

[10] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[11] David Dudley. *The Driverless Car is (Almost) Here*. 2015. URL: https://www.aarp.org/home-family/personal-technology/info-2014/google-self-driving-car.html.

[12] Les Earnest. *Stanford Cart*. 2021. URL: https://web.stanford.edu/~learnest/sail/oldcart.htmlf.

[13] Robert A. Ferlis. *The Dream of an Automated Highway*. 2007. URL: https://www.fhwa.dot.gov/publications/publicroads/07july/07.cfm.

[14] Mark Harris. *How Google's Autonomous Car Passed the First U.S. State Self-Driving Test*. 2014. URL: https://spectrum.ieee.org/transportation/advanced-cars/how-googles-autonomous-car-passed-the-first-us-state-selfdriving-test.

[15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[17] Aaron M. Kessler. *Elon Musk Says Self-Driving Tesla Cars Will Be in the U.S. by Summer*. 2015. URL: https://www.nytimes.com/2015/03/20/business/elon-musk-says-self-driving-tesla-cars-will-be-in-the-us-by-summer.html?hpw&rref=automobiles&action=click&pgtype=Homepage&module=well-region&region=bottom-well&WT.nav=bottom-well&_r=0.

[18] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[19] Fabian Kröger. "Automated Driving in Its Social, Historical and Cultural Contexts". In: *Autonomous Driving: Technical, Legal and Social Aspects*. Ed. by Markus Maurer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 41–68. ISBN: 978-3-662-48847-8. DOI: 10.1007/978-3-662-48847-8_3. URL: https://doi.org/10.1007/978-3-662-48847-8_3.

[20] Hanspeter Mallot et al. "Inverse Perspective Mapping Simplifies Optical Flow Computation and Obstacle Detection". In: *Biological cybernetics* 64 (Feb. 1991), pp. 177–85. DOI: 10.1007/BF00201978.

[21] Seyed-Iman Mirzadeh et al. *Improved Knowledge Distillation via Teacher Assistant*. 2019. arXiv: 1902.03393 [cs.LG].

[22] Thomas van Orden and Arnoud Visser. "End-to-end Imitation Learning for Autonomous Vehicle Steering on a Single Camera Stream". In: *Proceedings of 16th international conference on Intelligent Autonomous System*. IAS. 2021, pp. 224–235.

[23] Aditya Prakash et al. "Exploring Data Aggregation in Policy Learning for Vision-Based Urban Autonomous Driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[24] RCA. *Electronic Age*. 1958. URL: https://worldradiohistory.com/Archive-Radio-Age/Electronic-Age-1958-Winter.pdf.

[25] Lennart Reiher, Bastian Lampe, and Lutz Eckstein. "A Sim2Real Deep Learning Approach for the Transformation of Images from Multiple Vehicle-Mounted Cameras to a Semantically Segmented Image in Bird's Eye View". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7. DOI: 10.1109/ITSC45102.2020.9294462.

[26] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. *A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning*. 2011. arXiv: 1011.0686 [cs.LG].

[27] SAE International. *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*. SAE Standard J3016, Report No. J3016-201806. Published online. 2018. DOI: 10.4271/J3016_201806.

[28] Axel Sauer, Nikolay Savinov, and Andreas Geiger. *Conditional Affordance Learning for Driving in Urban Environments*. 2018. arXiv: 1806.06498 [cs.RO].

[29] A. Serban, E. Poll, and J. Visser. "A Standard Driven Software Architecture for Fully Autonomous Vehicles". In: *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2018, pp. 120–127. DOI: 10.1109/ICSA-C.2018.00040. URL: https://doi.ieeecomputersociety.org/10.1109/ICSA-C.2018.00040.

[30] Pierre Sermanet et al. *OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks*. 2014. arXiv: 1312.6229 [cs.CV].

[31] SWOV. *Ernstig verkeersgewonden in Nederland, SWOV-factsheet*. https://www.swov.nl/feiten-cijfers/factsheet/ernstig-verkeersgewonden-nederland. 2020.

[32] Time Magazine. *Science: Radio Auto*. 1925. URL: http://content.time.com/time/subscriber/article/0,33009,720720,00.html.

[33] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. *End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances*. 2020. arXiv: 1911.10868 [cs.LG].

[34] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. *Is Deep Reinforcement Learning Really Superhuman on Atari? Leveling the playing field*. 2019. arXiv: 1908.04683 [cs.AI].

[35] Mark Weber et al. *DeepLab2: A TensorFlow Library for Deep Labeling*. 2021. arXiv: 2106.09748 [cs.CV].

[36] Jameson Wetmore. "Driving the dream. The history and motivations behind 60 years of automated highway systems in America". In: *Automotive History Review* 7 (2003), pp. 4–19.

[37] Danny Yadron and Dan Tynan. *Tesla driver dies in first fatal crash while using autopilot mode*. 2016. URL: https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk.

[38] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. "Does computer vision matter for action?" In: *Science Robotics* 4.30 (May 2019), eaaw6661. ISSN: 2470-9476. DOI: 10.1126/scirobotics.aaw6661. URL: http://dx.doi.org/10.1126/scirobotics.aaw6661.

Appendix A

Benchmark results

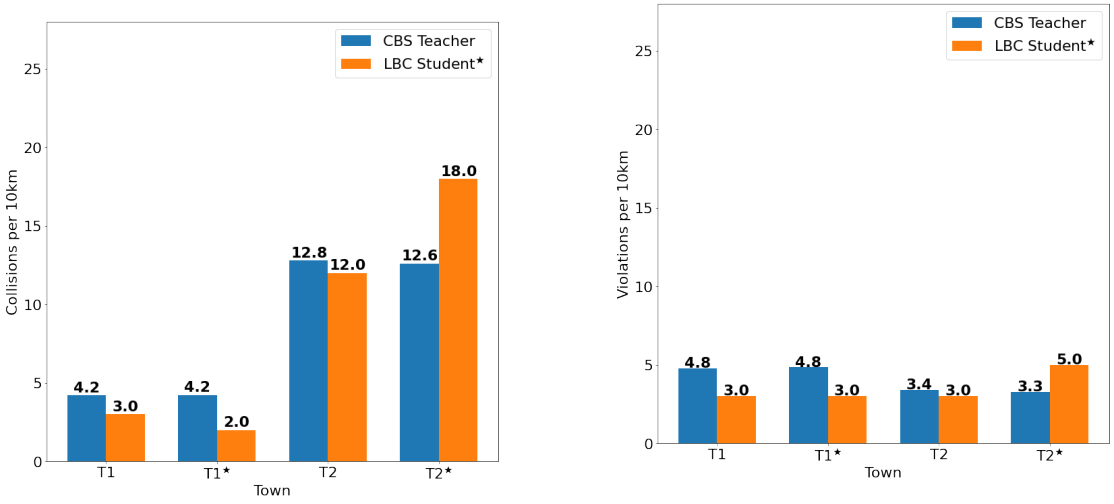
This Appendix provides additional results and analysis. Table A.1 shows our results in the context of other related studies. Notice that the other related studies have used older versions of CARLA. Therefore, their results are not directly comparable to ours, since we use CARLA version 0.9.6 and CARLA changed significantly with version 0.9.6 in comparison to older versions.

Table A.1: Success rates (higher is better, best scores are marked bold) on *CoRL2017* benchmark in train and test town. Our CBS Student★ is compared to Codevilla, Müller, et al. (2018) (CIL); Sauer, Savinov, and Geiger (2018) (CAL); and Codevilla, Santana, et al. (2019) (CILRS).

		Train town				Test town			
Task	Weather	CIL	CAL	CILRS	CBS Student★	CIL	CAL	CILRS	CBS Student★
Straight	train	98	**100**	96	**100**	97	93	96	**100**
One turn		89	**97**	92	78	59	82	**84**	66
Navigation		86	92	**95**	80	40	**70**	69	57
Nav. dynamic		83	83	**92**	76	38	64	**66**	55
Straight	test	98	**100**	96	98	80	94	96	**100**
One turn		90	**96**	**96**	64	48	72	**92**	40
Navigation		84	90	**96**	68	44	68	**92**	28
Nav. dynamic		82	82	**96**	64	42	64	**90**	24

The success rates in Table A.1show us that we outperform or score equal to all other approaches on 3 of the 4 *Straight* tasks. In more complex scenarios, our CBS Student★ yields competitive scores, but different CARLA versions may affect this performance.

To provide more insight into the CBS Teacher's driving behaviour, an infraction analysis is conducted on the *NoCrash* benchmark trajectories. This is shown in Figure A.1. Notice that the comparison is made with the LBC Student$^\bigstar$, since only this data was available. Nevertheless, our CBS Teacher shows similar results as LBC Student$^\bigstar$ in terms of the number of collisions and traffic light violations. Generalization in terms of towns proves to be difficult for both agents, with an increase by a factor 3 and 6 for the number of collisions for CBS Teacher and LBC Student$^\bigstar$, respectively. In contrast, CBS Teacher's traffic light violations decrease when generalizing to a new town. Generalization to new weather conditions is excellent for our CBS Teacher with similar collisions and violations in train weather and test weather.



(a) Number of collisions per 10 km.

(b) Number of traffic light violations per 10 km.

Figure A.1: Infraction analysis of our CBS Teacher on *NoCrash* trajectories compared to LBC Student$^\bigstar$. T1: Town 01 with train weather. T1$^\bigstar$: Town 01 with new (test) weather. T2: Town 02 with train weather. T2$^\bigstar$: Town 02 with new (test) weather.