DAI-Labor
Distributed Artificial Intelligence Laboratory

System Design Specifications of Experiments on different Sensor Setups for AD

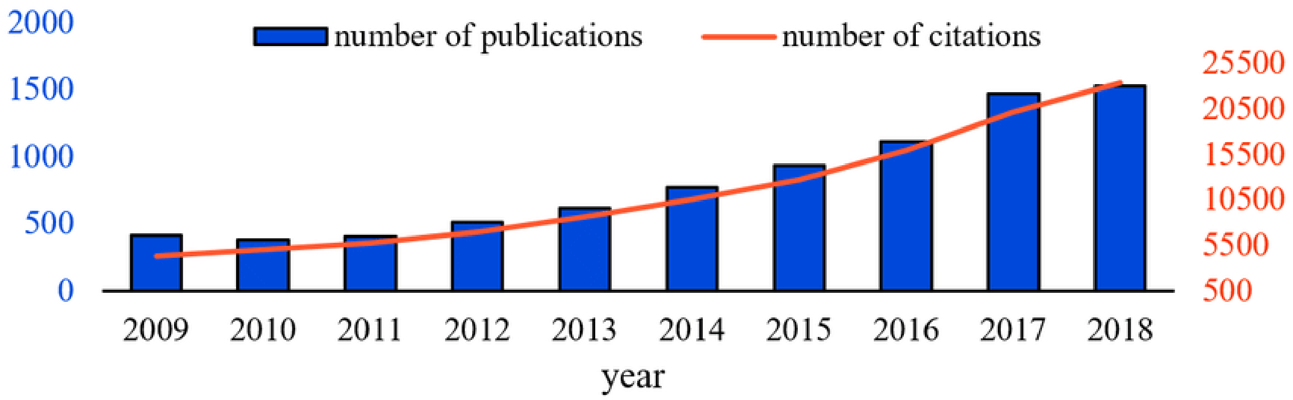
Applications of Robotics and Autonomous Systems

Team Members: Jakob Wirén, Julian Klitzing, Moritz Wassmer, Ege Atesalp, Can Kayser

Table of Contents

1 Introduction

Over the last decade, following the rise of popular electric car-brand Tesla, the field of autonomous driving has grown immensely, and for the last ten years an exponential increase has occurred in publications regarding relevant topics[1].



[Figure 1: Numbers of publications and citations in autonomous driving research from 2009 to 2018 [2]]

As the field of autonomous driving has grown, so has the amount of approaches tackling the problem. Two widely-accepted methods are the modular paradigm and the End-to-End paradigm. The modular paradigm makes use of the attractive engineering principle called divide-and-conquer - separating the problem of autonomous driving into clearly different subtasks. Such sub-modules classically include: Perception, Sensor Fusion, SLAM, Path-planning, Motion-planning. Within each sub-module a set of sub-sub-modules are quite often present, within Perception the tasks of Object detection, tracking, segmentation, depth estimation are generally found as an example. The end-to-end paradigm is in this aspect fundamentally different. Although the modularity and D&C approach presents many benefits such as easier debugging and interpretability, the complexity of the self-driving task requires a vast amount of submodules. The end-to-end approach makes the opposite trade-off, sacrificing interpretability and debugging for having a tiny amount of modules in the pipeline. By focusing on learning holistic models, end-to-end driving works by mapping raw sensor input directly to control signals for maneuvering. This “simple” approach of course puts huge requirements on the network architecture, data amount, training methods and sensor choice to work coherently.

As the market for autonomous vehicles explodes, the demand for hardware in terms of sensors is as much of a logistical problem as it is an autonomous system engineering problem. Which sensors are truly required for a robust and safe autonomous system? This question is as relevant in both self-driving paradigms, but which multimodal sensor set-up (multimodal sensor set-ups have been shown to increase performance in relevant key perception tasks compared to unimodal set-ups[3]) provides optimal performance remains less explored in the end-to-end field. As such, this project paper will aim to answer the question:

How does the choice of multimodal sensor set-up affect the performance of a given NN architecture, when evaluated in the autonomous driving simulator CARLA?

1.1 Problem Definition

Goal of the project: To compare and evaluate the effect of different sensor-setups in a “controlled” environment in CARLA.

Motivation: To aid informing the choice of sensor-setup in the autonomous vehicle sphere. Providing knowledge for start-ups/autonomous vehicle producers and researchers regarding the benefits and difficulties of different popular sensor set-ups in a certain testing environment. To inspire more research on the interplay between hardware choices and End-to-End architectures.

Clear problem definition: This project tries to add to the existing body of research about HW-related design choices in End-to-End autonomous driving by comparing multimodal sensor set-ups and their performance in a simulated environment. To do so we develop a pipeline to experiment with different sensor setups while keeping the dataset and model architecture relatively fixed.

We will aim to compare the performance of RGB-only, Lidar-only, and a multimodal RGB-Lidar set up using ResNet as the benchmark architecture.

1.2 Objectives

1. **Sensor choices**: Decide on a set of sensors and different combinations (setups).
2. **Expert Driver**: Choose an expert driver for CARLA.
3. **Dataset**: Generate a Dataset with the expert driver.
4. **E2E Model**: Implement an E2E model capable of driving autonomously.
5. **Data Preprocessing** (for all sensor setups): Build a pipeline to generate compatible data for the chosen architecture.
6. **Training**: Train the different models with varying sensor setups and varying data sizes.
7. **Evaluation**: Conduct the experiments and compare the sensor setups

1.3 Use cases / scenarios

The general use of this experiment is to find out how different sensor setups affect the performance of autonomous driving, to get an understanding of how sensors complement each other (and possibly how good they are on a standalone setup). Therefore we don't have explicit use cases in practice for the resulting agents. The use cases and scenarios where it could be used in practice therefore highly depends on if we can find a well defined data generation script which we can adapt, as well as finding a (pretrained) architecture. Also, training highly depends on computing power.

All in all, we can not make a clear statement in advance on which scenarios we expect the agents to handle. It could vary between SOTA performance on public benchmarks to simple arriving at the target in very easy conditions (same map, no pedestrians, no vehicles, good weather, easy route). SOTA use cases would involve very "extreme" scenarios, like defined here CARLA Autonomous Driving Leaderboard. Depending on how well the models work, we are going to use respective harder benchmarks.

Still we want to give a small scenario hierarchy. We plan on using a public benchmark/leaderboard if possible to evaluate. Still we can not guarantee that those scenarios are being tested explicitly in the benchmarks.

We expect **route completion and low infractions**

1. Easy environment: empty roads, good weather
2. Harder environment: other agents (cars, cyclists, pedestrians), varying weather
3. "Extreme" Situations: CARLA Autonomous Driving Leaderboard

1.4 General Constraints

- The architecture, dataset, evaluation are going to be fixed to one which can handle all sensors which we want to support
- We will decide on a set of sensors setups which we find feasible for the scope of our project and interesting for research (probably 3)

- No offroad scenarios
- We might limit the dataset size (decrease the number of maps) in case we find training takes too long.
- Policy learned is only applied to the car and it's setup as it was trained on

2 State-of-the-Art

As a starting point for grasping the complex field of end-to-end autonomous driving and the different methods found in that field, the survey paper *A survey of End-to-End Driving: Architectures and training methods* by Tampuu et. al [4] provides a fine overview of existing methods, input and output modalities, network architectures and evaluation schemes. It provides a well-balanced introduction and plenty of tips in regards to many important concepts such as data-set balancing, fusion methods and CARLA benchmarks. Our report takes inspiration from the lessons presented in Tampuu et. al's survey paper. As our research aims to look specifically into multimodal approaches to end-to-end autonomous driving, further inspiration was taken from papers stemming in this field. One such example of a paper would be *Multimodal End-to-End autonomous driving* by Xiao et. al. [5]. This paper analyzes whether combining RGB and depth modalities, such as via RGB-D or Lidar, produces better end-to-end AI drivers than relying on a single modality. They further examine the difference between multimodal sensor fusion occurring early, mid, or late in the pipeline. With CARLA as a testing platform, they conclude that early fusion multimodality outperforms single-modality. This conclusion serves as a key lesson for the work to be done in our paper. Thirdly, the effect of data-collection and selection is a highly relevant question for our research. For this, the paper *On the Choice of Data for Efficient Training and Validation of End-to-End Driving Models* by Klingner et. al [6] has been used to draw inspiration. The paper investigates the influence of several data design choices regarding training and validation of deep driving models trainable in an end-to-end fashion. They look at the amount of training data, validation design and random seeding as well as non-determinism. Their recommendations regarding data generation and route selection has affected the engineering choices leading to this report.

3 System Requirements, Architecture and Specifications

Since we are rather conducting an experiment than implementing new AD agents, we are focusing on our Pipeline in this part.

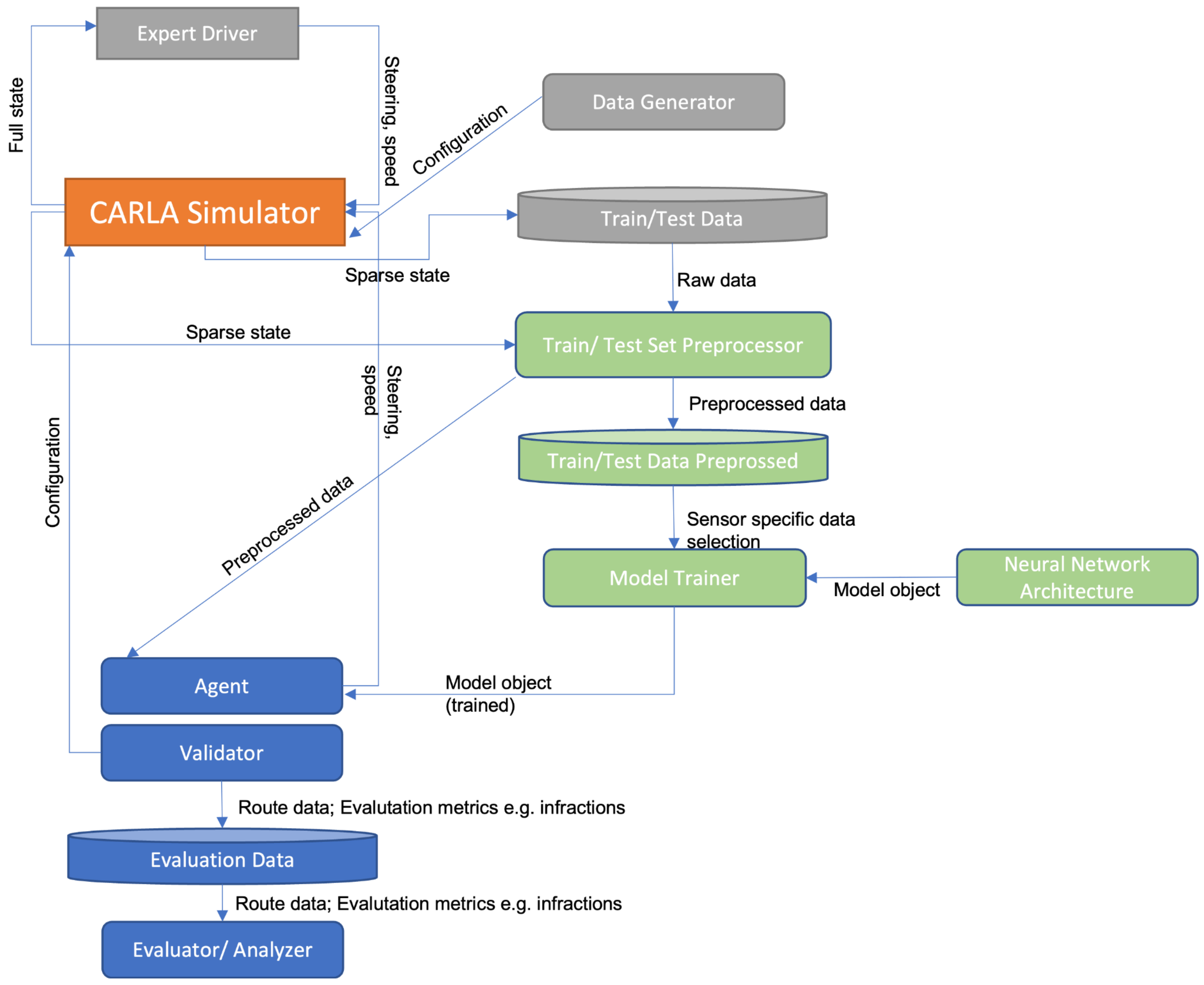
3.1 Requirements

1. **Sensor choices**: Decide on a set of sensors and different combinations (setups).
   - Chosen sensors and sensor setups must be relevant in current research.
   - Chosen sensors and sensor setups must be compatible with CARLA.
2. **Expert Driver**: choose or find an expert driver for CARLA.
   - The chosen expert driver must be configurable for the defined scenarios.
3. **Dataset**: Generate a Dataset with the expert driver.
   - Data must cover sufficient routes and scenarios and other agents to train a model which is at least able to arrive at targets.
   - Must capture all sensor measurements.
4. **E2E Model**: implement an E2E model capable of driving autonomously given navigational commands.
   - Must be able to handle varying sensor setups.
   - Must output a policy vector containing at least steering angle, but ideally acceleration as well (including breaking).
   - Must respect all traffic rules.
   - Optional: Should follow navigational commands.
5. **Data Preprocessing** (for all sensor setups): Build a pipeline to generate compatible data for the chosen architecture.
   - Must be able to handle all supported sensor data.
   - Transform it into a NN suitable representation.

6. **Training**: Train the different models with varying sensor setups.
    - The training method must train the models sufficiently but also not take too long.
    - The training method must optimize hyperparameters in a feasible way.
    - It has to be possible to save the models for reusing it for our autonomous agent later on.
7. **Evaluation**: Conduct the experiments and compare the sensor setups
    - Must measure interesting measures according to current research (eg. Route Completion (RC), Infractions Score (IS)).

3.2 System Architecture

In the following system architecture diagram the data engineering related components are highlighted in gray, the data science related components in green and the evaluation related components in blue.



3.3 System Specifications

The inputs and outputs of the individual nodes can be found in the systems architecture diagram in 3.2. In the following, only the functionality of each individual node is briefly defined.

CARLA Simulator

Used for data generation and interaction of our trained agent with the environment. For this experiment we have used CARLA 0.9.10.1.

Expert Driver

The expert driver is a hardcoded agent that can perform “perfect” driving in the CARLA simulator and is thus used for generation of the train and test data sets. It is a rule-based expert algorithm used and Its performance is an upper bound for the learning-based agent. The autopilot has access to the complete state of the environment including vehicle and pedestrian locations and actions.

Data Generator

Used to initialize the CARLA simulator environment (i.e. driving conditions such as other vehicles, pedestrians, …) and to instantiate the expert driver. It then runs the simulation and saves the data. We have used noise injected data in our training to improve our agents.

Train/Test Data

This data is the foundation for model training and evaluation. The database stores the *sparse state* emitted by the CARLA Simulator. The *sparse state* consists of all defined sensor data, vehicle control signals and vehicle state measurements.

Train/Test Set Preprocessor

Used to perform preprocessing tasks on the data stored in the Train/Test Data database to prepare them for neural network usage.

Train/Test Data Preprocessed

Is a database that stores the preprocessed data to make preprocessed data reusable.

Neural Network Architecture

Defines the neural network architecture that will be trained.

Model Trainer

Trains the given model architecture according to defined schedules, hyperparameters etc.

Agent

Is the end-to-end network that represents the learned driving policy and interacts with the CARLA simulator.

Validator

Similarly to the Data Generator, it initializes a simulator environment and initiates the Agent built upon the trained model. Then it runs the simulation and captures the driving performance of the autonomous agent for all tracks which are to be tested.

Evaluation Data

Contains the performance measures of the autonomous agents for every tested track.

Evaluator/ Analyzer

Visualizes and analyzes the evaluation data regarding overall performance of the autonomous agents. From that we are going to extract our results for the experiments i.e. comparing the driving performance between the different sensor setups.

4 Timeline

Time	Concept
W 1-2	Understand the topic, set up HW and SW. Tasks and roles delegated. All preparation for work done.
W3	First presentation - MS1. SDS Hand-in.
W 4-5	Finalize all design choices and constraints by the end of November. Collect a large and general database to begin working with.
W 6	Post-process data and begin training the initial network set-up (V1).
W 7-10	Training, improving, evaluation of V1 results.
W10	Internal Milestone/Deadline. Each working subteam holds a presentation for the rest of the team to ease documentation. V2 of architecture and code finalized. At this point the questions that have arisen during W7-10 must be considered and evaluated.
W12	M2 Presentation
W14	Internal Milestone/Deadline. Each working subteam holds a presentation for the rest of the team to ease documentation. Internal evaluation of V2 performance and critical examination of priorities in the last weeks. Final tweaking (hopefully) to reach V3/final product.
W 17	M3 Presentation. Final submission.

5 Frameworks and Tools

- For simulation, collection of custom data-sets and evaluation of performance, the widely-used simulator CARLA will be used.
- If we train our own models, the framework PyTorch will be used.
- For project planning and code version management, GitLab will be used.
- We are going to use open source code from GitHub/GitLab for Data Generation, Evaluation of our AD agents and maybe even for a model architecture. For example:
  - GitHub - autonomousvision/transfuser: [PAMI'22] TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving, [CVPR'21] Multi-Modal Fusion Transformer for End-to-End Autonomous Driving
  - GitHub - carla-simulator/leaderboard: CARLA Autonomous Driving leaderboard
  - GitHub - carla-simulator/scenario_runner: Traffic scenario definition and execution engine

6 Method

In order to achieve the aim of comparing the performance of different multimodal models in CARLA, we need a performance metric that quantifies performance. Quantitatively, this is what the CARLA leaderboard gives us.

The CARLA Leaderboard works to evaluate the driving proficiency of different agents in realistic traffic scenarios. Not only does this serve the purpose of giving a comparable metric between our different agents, but also provides the advantage of global comparability and reproducibility. The CARLA leaderboard is a widely used open platform that is the benchmark for how to perform fair and reproducible evaluations of autonomous vehicle agents within CARLA.

The CARLA Leaderboard uses a set of predefined routes. For each route, our agent is initialized at a starting point and directed to drive to a destination point. Routes are defined in a wide variety of both situations (such as freeways and residential districts) and weather conditions (such as daylight, sunset, rain, night etc).

Finally, the performance on this set of routes is measured by a standardized set of metrics such as route completion, running a red light, collision with a pedestrian, route deviation etc. Whenever such an event occurs during testing, it is recorded. The global infractions (returned by the Leaderboard) compress the individual route's data into a single value and is given as the number of events per kilometer.

Inspired by TransFuser, we made use of the so-called Longest6 benchmark. The Longest6 benchmark consists of 36 routes with an average route length of 1.5km (similar to the official leaderboard). Each route has a unique environmental condition obtained by combining one of 6 weather conditions (Cloudy, Wet, MidRain, WetCloudy, HardRain, SoftRain) with one of 6 daylight conditions (Night, Twilight, Dawn, Morning, Noon, Sunset).

Furthermore, we are able to give depth to the analysis by also *qualitatively* evaluating the performance of the agent in Carla. Analyzing the agent's behavior can sometimes hint towards *why* or *how* an agent differs from another. Note however that this type of qualitative analysis is very hard to use effectively in the case of End-to-End neural networks as the amount of parameters and possible reasons for certain behavior is huge.

7 Results

- **Longest6 results**
- **Analysis?**
- **Manual analysis of Carla performance / visualization analysis**

8 Conclusion/Discussion

- **What did we get done? What did we manage to achieve/what did we not manage to achieve (why?)?**
- **Future work**
- **What could have been done better**

9 References

[1] Retrieved from: "Key Ingredients of Self-Driving Cars" by Rui Fan, Jianhao Jiao, Haoyang Ye, Yang Yu, Ioannis Pitas, Ming Liu

[3] Retrieved from: C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum PointNets for 3D object detection from RGB-D data,” in Int. Conf. on Computer Vision and Pattern Recognition (CVPR), 2018

[4] Retrieved from: https://ieeexplore.ieee.org/document/9310544/

[5] Retrieved from: https://ieeexplore.ieee.org/document/9165167

[6] Retrieved from: https://arxiv.org/pdf/2003.06404.pdf