HAW HAMBURG

Bachelor Thesis

Fynn Luca Maaß

End-to-End Deep Learning for Lane Keeping of Self-Driving Cars with Focus on Robustness against System Discrepancies Regarding the Steering

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Fynn Luca Maaß

End-to-End Deep Learning für die Spurhaltung von selbstfahrenden Autos mit dem Fokus auf die Robustheit gegenüber Systemdiskrepanzen in der Lenkung

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stephan Pareigis
Zweitgutachter: Prof. Dr. Peer Stelldinger

Eingereicht am: 11. Oktober 2021

Fynn Luca Maaß

Title of Thesis

End-to-End Deep Learning for Lane Keeping of Self-Driving Cars with Focus on Robustness against System Discrepancies Regarding the Steering

Keywords

Autonomous Driving, End-to-End, PilotNet, Deep Learning, Steering Discrepancies

Abstract

End-to-end approaches for autonomous driving gain popularity since NVIDIA demonstrated its capabilities with their PilotNet architecture in 2016. Although many research projects are subject to the field of end-to-end architecures, less effort is taken into the issue that arises when a vehicle with different physical properties uses an architecture that it was not trained on specifically. This results in something that we refer to as 'system discrepancies' and can be understood as a sim-to-real problem. In this thesis we implement and evaluate NVIDIA's popular PilotNet against a system discrepancy which is an offset in the steering. This architecture maps an input image of the front facing camera to an absolute steering wheel angle. We use a custom data set including images with minimal complexity in regard to image features that we generated in the autonomous driving simulator CARLA. In the context of CARLA, we demonstrate that a steering offset negatively impact the driving performance and offsets the vehicle's position to an amount that is not acceptable in a real world scenario. We propose, implement and evaluate a prototype architecture called PilotNetΔ (PilotNet Delta) that has increased robustness against the steering offset and leads to improved results when comsidering the lateral offset on the road. PilotNetΔ uses a convolutional LSTM layer to map a sequence of images to a relative steering angle, which is the difference to the prevoius steering prediction.

Fynn Luca Maaß

Thema der Arbeit

End-to-End Deep Learning für die Spurhaltung von selbstfahrenden Autos mit dem Fokus auf die Robustheit gegenüber Systemdiskrepanzen in der Lenkung

Stichworte

Autonomes Fahren, End-to-End, PilotNet, Deep Learning, Systemdiskrepanzen

Kurzzusammenfassung

End-to-End-Ansätze für das autonome Fahren gewinnen an Popularität, seit NVIDIA 2016 mit der PilotNet-Architektur deren Fähigkeiten demonstrierte. Obwohl sich viele Forschungsprojekte mit End-to-End-Architekturen befassen, werden weniger Anstrengungen unternommen, um das Problem zu lösen, das entsteht, wenn ein Fahrzeug mit unterschiedlichen physikalischen Eigenschaften eine Architektur verwendet, für die es nicht speziell trainiert wurde. Dies führt zu etwas, das wir als "Systemdiskrepanzen" bezeichnen und als ein Sim-zu-Real-Problem verstanden werden kann. In dieser Arbeit implementieren und evaluieren wir NVIDIAs PilotNet hinsichtlich einer Systemdiskrepanz, die ein Versatz im Lenksignal ist. Diese Architektur bildet ein Eingangsbild der Frontkamera auf einen absoluten Lenkradwinkel ab. Wir verwenden einen benutzerdefinierten Datensatz mit Bildern mit minimaler Komplexität in Bezug auf Bildmerkmale, die wir im autonomen Fahrsimulator CARLA generiert haben. Im Rahmen von CARLA zeigen wir, dass sich ein Lenkungsversatz negativ auf die Fahrleistung auswirkt und die Position des Fahrzeugs in einem Ausmaß verschiebt, das in einem realen Szenario nicht akzeptabel ist. Wir schlagen eine Prototyp-Architektur mit dem Namen PilotNetΔ (PilotNet Delta) vor, implementieren und bewerten sie. Diese Architektur ist robuster gegenüber dem Lenkungsversatz und führt zu besseren Ergebnissen bei der Berücksichtigung des seitlichen Versatzes auf der Straße. PilotNetΔ verwendet ein Convolutional LSTM Layer, um eine Bildsequenz auf einen relativen Lenkwinkel abzubilden, der die Differenz zur vorherigen Lenkvorhersage darstellt.

Contents

List of Figures

List of Tables

Acronyms

CNN Convolutional Neural Network.

ELU Exponential Linear Unit.

FPM Frames per Meter.

FPS Frames per Second.

GPS Global Positioning System.

GPU Graphics Processing Unit.

IMU's Inertial Measurement Units.

LIDAR Light Detection and Ranging.

MSE Mean Squared Error.

PCA Principal component analysis.

PID Proportional Integral Derivative.

Radar Radio Detection and Ranging.

ReLU Rectified Linear Units.

RGB Red Green Blue.

ToF Time of Flight.

1 Introduction

Over the past few years, interest in autonomous driving has increased significantly in academia and industry. This technology will continue to gain in importance in the future and will prevail over conventional driving in the long term. There are various architectures using different sensors implementing different degrees of autonomy in vehicles. Sensors, such as cameras, Inertial Measurement Units (IMU's), Global Positioning System (GPS), Time of Flight (ToF) sensors like Light Detection and Ranging (LIDAR) or Radio Detection and Ranging (Radar). The processing of sensor signals typically takes place in a pipeline, where different sensor signals are accumulated, pre-processed and interpreted to gain an understanding of the current driving scene. This begins with typical data pre-processing, for instance low or high pass filtering or inter- or extrapolating of all kind of signals. Pre-processed data signals can be used independently or combined to extract features from them using algorithms with varying degree of complexity. A Principal component analysis (PCA) [1] can be used as an algorithmic approach to determine and extract important features from high dimensional sensor signals. An alternative to PCA is the more advanced autoencoder [1] for the same purpose. Clustering techniques [2] can be used to classify features or segment objects, for example in a LIDAR point cloud [3]. Highly complex neural networks can be used for bounding box estimation and classification of objects [4][5]. Those high-level features can be used further down the pipeline in other algorithms for route planning in order to make control decisions for the vehicle.

Pipelines for autonomous driving are very complex and consist of multiple components and procedures with some of them described above. An alternative solution to those complex pipelines are end-to-end architectures. They often require only minimal preprocessing of the input data and combine many steps of a typical pipeline. Such an end-to-end architecture, called PilotNet, was introduced by NVIDIA for autonomous steering of a vehicle in 2016 [6], using a Convolutional Neural Network (CNN) architecture. This architecture is able to drive successfully on a roadway with only 72 hours of data used

to train the model in a supervised manner from a single input image to a steering wheel angle. In May 2017 NVIDIA released another paper 'Explaining How End-to-End Deep Learning Steers a Self-Driving Car' [7]. This publication analyzes PilotNet, which image features are needed and their influence the steering output. Whenever PilotNet is mentioned in this work, it refers to the state of PilotNet from 2016 and 2017. The advantage of PilotNet is its relatively small amount of parameters in the neural network with just 250,000 and a good performance for lane keeping at the same time.
NVIDIA's publications about end-to-end autonomous driving aroused interest in the research community. Many other authors published different papers related to the work of NVIDIA and to end-to-end autonomous driving in general. [8] for instance, proposed a Convolutional Long Short-Term Memory Recurrent Neural Network (C-LSTM) to learn spacial as well as temporal dependencies of the input data. Furthermore, they also treat the regression problem of estimating the correct steering output as a classification problem. Others [9] use a 360 degree image input in combination with information of a route planner rather than just a single front facing camera to train the network and improve the driving performance. [10] presents an end-to-end architecture for driving via conditional imitation learning. This addresses the problem that the vehicle in most end-to-end approaches cannot be guided to take a specific turn.
In early 2021 NVIDIA published yet another paper 'The NVIDIA PilotNet Experiments' [11] concluding the work over the past couple of years and introducing a new, more complex version of PilotNet with fundamental differences to the previous version. This end-to-end architecture is not trained on steering angles as the older PilotNet, but on a trajectory consisting of multiple points. This trajectory is then used in combination with a feedback controller to steer the vehicle. The architecture is also capable of taking specific turns rather than just following a roadway. However, the training data required for this architecture is significantly more complex. Due to the novelty, it will only play a minor role in this thesis, but should nevertheless be mentioned for the sake of completeness.

1.1 Problem Definition

As described above, not only NVIDIA but other companies and research institutions have put a lot of effort into the field of supervised machine learning for end-to-end autonomous driving architectures. In a lot of cases, training data gathered is done on the same vehicle as testing the trained model. As a result, certain peculiarities present

in the system are learned as well. An easy example for this is the steering ratio, referring to the ratio between the turn of the steering wheel and the turn of the wheels. In other cases, training and evaluation of architectures are only done inside a simulation where mechanical properties of a vehicle are disregarded to a certain degree. If an architecture is now trained inside a simulation and tested in the real world, a problem referred to as the sim-real gap problem [12] often becomes present. Similarly, a problem can occur when training is done with a different vehicle than testing due to the system discrepancies.

This thesis focuses on a specific system discrepancy regarding the steering, which will

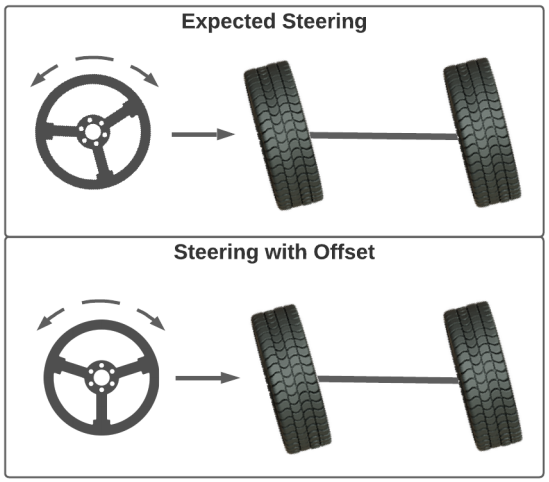


Figure 1.1: Illustration of a steering wheel offset in regard to the front axle of a vehicle. The top illustrates the relationship between steering angle and wheel position as expected. The bottom illustrates a steering offset. In this particular case of -15 degrees (to the left) for illustration purposes.

be referred to as steering offset. Figure 1.1 illustrates the meaning of a steering offset. The top part of figure 1.1 illustrates an expected steering behavior. This means the wheels of the vehicle steer right when the steering wheel is turned right, the wheels are straight when the steering wheel is straight and so on. The bottom part of figure 1.1 illustrates the steering with an offset. This means, in order to drive straight, the steering wheel needs to rotate 15 degrees to the right in order to even out the offset of -15 degrees for this particular case. Such a discrepancy can be caused due to a falsely calibrated or assembled steering wheel or due to failures in the mechanical hardware. In the case of a miniature vehicle, it could be a falsely aligned servo motor or an asymmetry in a 3D-printed chassis.

1.2 What to Expect from this Thesis

This thesis requires basic understanding of machine learning and addresses people already familiar with the basic concepts. In particular, the terms supervised learning, loss function - especially mean squared error, regression, training/validation data as well as principles of image pre-processing should be known. There are references to literature throughout this thesis for more advanced concepts. At this point we would like to make one more remark: Although all necessary information to recreate the experimental setup and reproduce the results are given, unnecessary implementation details, especially in regard to the integration of the simulation environment CARLA (chapter 2) and training pipeline (chapter 3) are omitted. The aim of this thesis is to introduce the reader to the issue and possible effects of a steering offset in regard to PilotNet. We will propose a general idea in the form of a new architecture including some pitfalls to reduce the initial problem.

1.3 Research Question

Starting from the problem of a steering offset, this thesis will deal with three main questions:

1. Chapter 4: How does an offset in the steering affect NVIDIA's end-to-end PilotNet architecture?
2. Chapter 5: How might an architecture be designed to gain robustness against this discrepancy, and how does it behave?
3. Chapter 6: How do both architectures differ in driving performance and training effort?

1.4 Structure of this Thesis

Chapter 2 will focus on the simulation environment and will provide information necessary to recreate the simulation setup which is the foundation of training and evaluation of the architectures. Chapter 3 will introduce the reader to the training pipeline, which includes general information about data gathering, data processing, model training and

evaluation, which is relevant to PilotNet and the upcoming architecture in this thesis. After these foundations have been clarified, chapter 4 gives a more detailed introduction to NVIDIA's PilotNet and presents results on its performance. Insights into the behavior of the trained model are used to develop another architecture. Chapter 5 proposes the new architecture with results to its performance. Chapter 6 discusses the results of both architectures and presents difficulties as well as advantages and disadvantages of the proposed architecture. Finally, the main aspects of this thesis are concluded in chapter 7.

2 The Simulation Environment

The development, training, and evaluation in this thesis is taking place inside a simulation. This chapter elaborates on details regarding the autonomous driving simulator CARLA [13], how system discrepancies in the steering are modeled and how the physical environment, meaning the maps used for training and evaluation purposes, are designed.

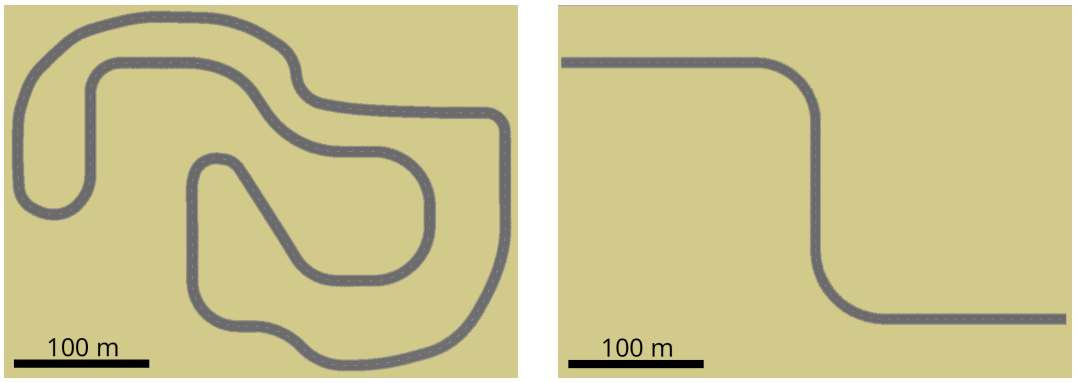
2.1 Autonomous Driving Simulator: CARLA

CARLA is an open-source driving simulator, specifically designed to tackle a wide variety of challenges in the autonomous driving space. Its capabilities consist in simulating and rendering a realistic environment, particularly in regard to a good, physical accurate world. CARLA features a large suit of sensors including depth- and Red Green Blue (RGB) cameras, LIDAR and IMU's amongst other sensors. These sensors can be attached to different actors like a vehicle to perceive the environment. CARLA provides a Python API to access and configure such actors, sensors and the virtual world. CARLA version 0.9.11 is used in this thesis in combination with Python 3.7.

2.2 Custom Maps for CARLA with RoadRunner

CARLA already features maps with high environmental detail and a variety of complex traffic scenarios including junctions. However, research on how certain image features influence steering decisions was undertaken by NVIDIA and is thus not included in this work. Therefore, we reduce the image complexity to simplify the experimental setup as well as the training procedures to gain a better focus on the research questions. Since NVIDIA's PilotNet is designed for lane keeping, no maps with junctions are beneficial for the experiment setup and the process of data generation.

RoadRunner [14] is a tool for designing maps for CARLA that is used to create two custom maps for the purpose of this thesis. Each map is a simple flat surface with a two lane road on top. Map 1, illustrated in 2.1a, is used for data gathering, which is described in 3.1. Map 2, illustrated in 2.1b, is a simple s-shaped road for the evaluation purposes of trained models described in 3.3.



(a) Map 1 for data gathering - length: 1554 m (b) Map 2 for evaluation - length: 557 m

Figure 2.1: Custom maps designed for evaluation and training purposes.

2.2.1 Client - Server Interaction

CARLA is a client-server architecture. The server is responsible for rendering sensor results, physics computations and updating the world-state. The client consists of custom modules that can be written to evaluate sensory data from the actors and apply commands back to them.
Figure 2.2 gives an overview of how the client-server interaction in CARLA works. The client is more of an interest, because inside of it sits our neural network architecture that is interchangable. Therefore, parts of the client's implementation will change throughout this thesis, depending on the neural network architecture that is embedded inside the client. The general order of events will always be the same and is as follows:

1. The simulation is ticked by the client.
2. The simulation simulates the next 0.1 seconds (moving the vehicle, capturing image etc.).
3. The image is received by the client, processed and fed into the neural network.

4. The output of the neural network is manipulated with the steering offset, depending on the current test setup, by subtracting 7.5 degrees.

5. The final steering command is sent back to the vehicle in the simulation. The simulation is ticked again by the client.

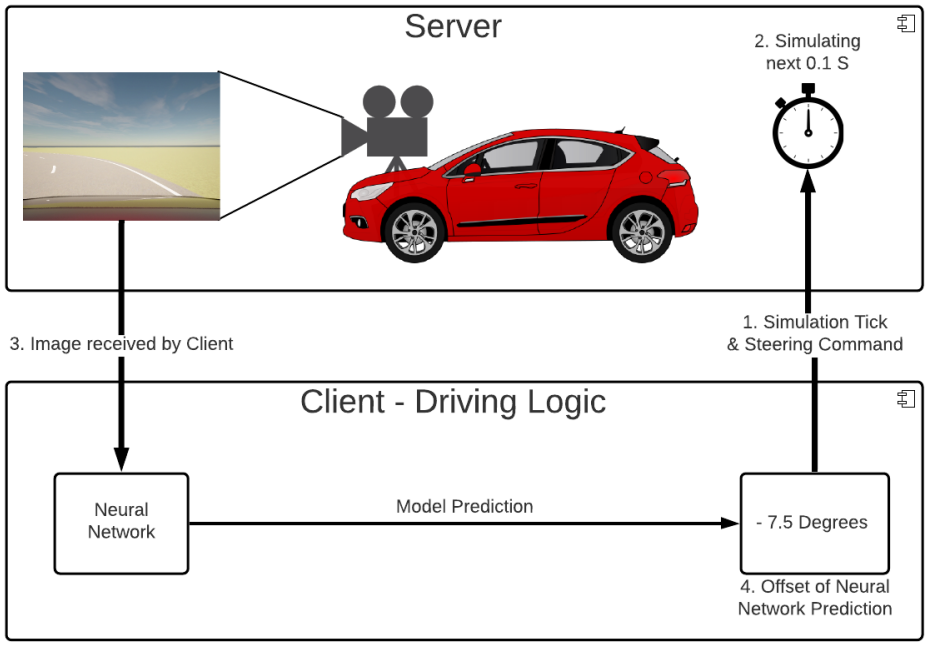


Figure 2.2: Client-Server Interaction

3 Training Pipeline

This section describes the implemented training pipeline as illustrated in figure 3.1. The training pipeline consists of three steps: The first step is the creation of the training data and described in section 3.1. The second step is the training process described in section 3.2, leading to a trained model that is evaluated in the last step in section 3.3. The training pipeline is implemented in Python 3.7 and uses TensorFlow 2.3.1 for the training procedure.

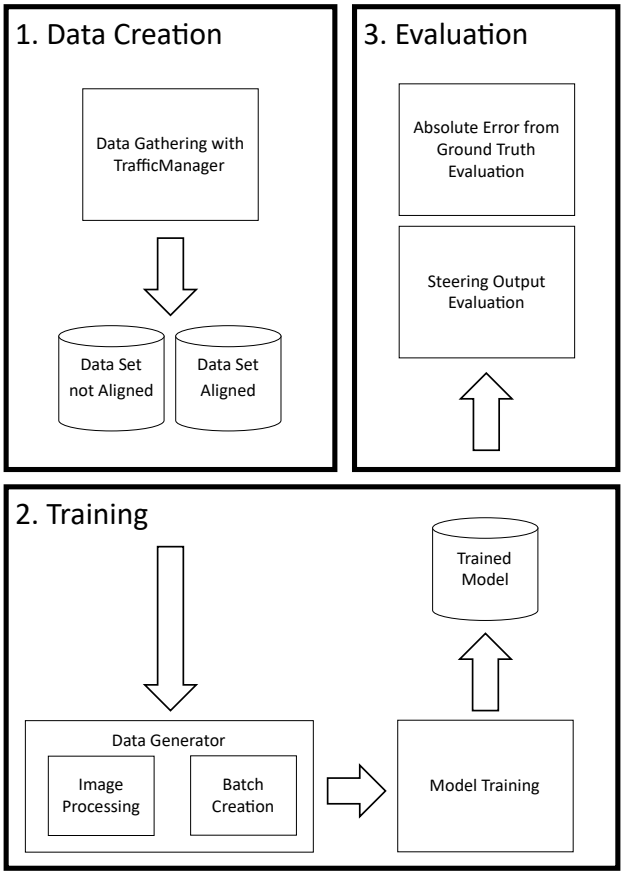


Figure 3.1: Training pipeline consisting of 3 steps.

3.1 Data Creation

As described in [6], data is generated in a real world with a human driver steering the car, while capturing the front facing camera and labeling the images with the steering angle. The process of data generation in this thesis is similar, yet automated and taking place inside of the CARLA simulator. This has the benefit of generating thousands of images with minimal resources of time and manpower. Therefore, the human driver is replaced with a software component from the simulation called TrafficManager [15]. TrafficManager is a software component capable of steering vehicles based on information that are accessed from the simulation and fed into the Proportional Integral Derivative (PID) [16] controller inside the TrafficManager.

Training data is generated by the TrafficManager steering the vehicle across map 1 as introduced in section 2.2. We specify and operate the sample rate of the camera not in Frames per Second (FPS), but instead in Frames per Meter (FPM), in order to achieve independence from the vehicle velocity. Unfortunately, the sampling rate of cameras in CARLA cannot be normalized to distance, consequently the TrafficManager drives at a constant speed of 5 meters per second, sampled at 10 FPS with the front facing camera, resulting in 2 FPM.
A distinction is made regarding the complexity of data gathered by the TrafficManager. This results in two different data sets, which are described below. Statistical measures on the data sets are presented in sections 4.2, 5.2 of the upcoming architectures.

3.1.1 Data aligned with Target Trajectory

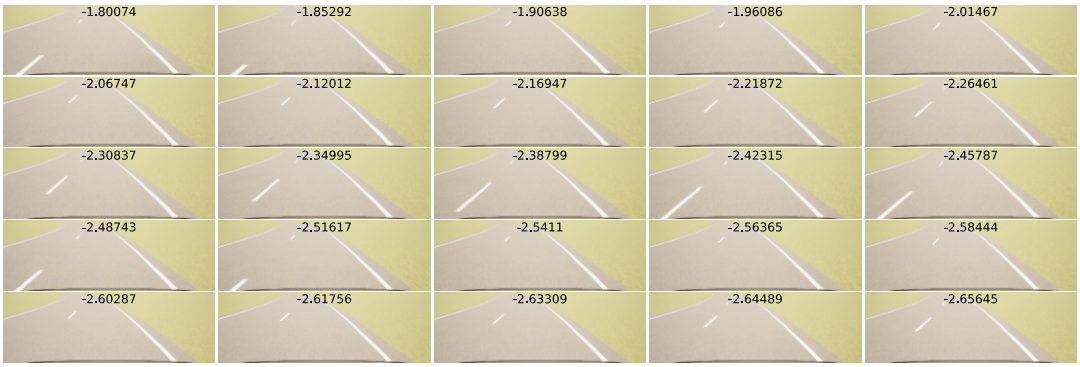


Figure 3.2: 25 frames long excerpt of generated data ordered line-by-line showing the vehicle's alignment to the target trajectory.

The first and simpler kind of data covers only the scenario of the vehicle being perfectly aligned with the target trajectory. Figure 3.3 illustrates an exemplary excerpt of this kind of data with 25 consecutive frames ordered line-by-line with the corresponding label in degrees to each frame overlaid in the aftermath. Negative numbers resemble steering to the left, zero is the neutral angle and positive numbers resemble a steering to the right. For orientation purposes, it is useful to look at the lane marking on the bottom right of each image to estimate the lateral position of the vehicle. As can be seen, the TrafficManager steers a slight left turn in this particular sequence while maintaining its lateral position on the road. The data set contains two large sequences of the vehicle driving, once clockwise, and once counterclockwise on map 1.

3.1.2 Data not aligned with Target Trajectory

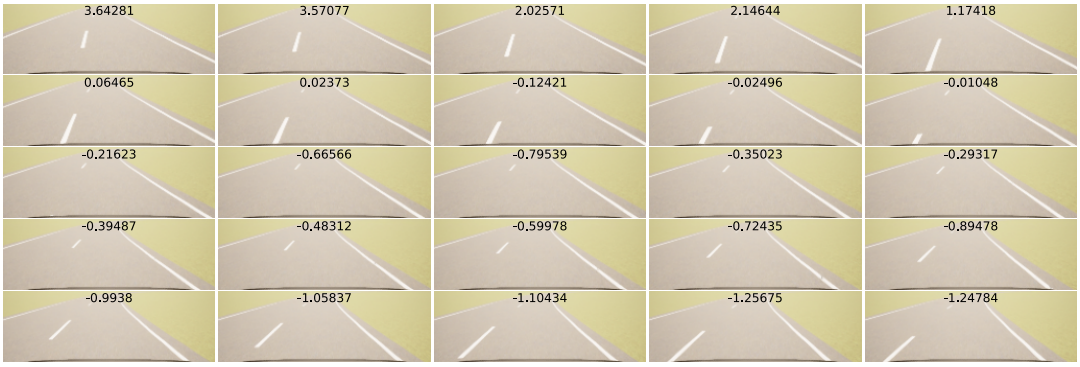


Figure 3.3: 25 frames long excerpt of generated data ordered line-by-line showing the vehicle not being aligned with the target trajectory, but returns to alignment.

The second and more complex kind of data includes sequences where the vehicle is not aligned with the target trajectory initially, but returns to alignment. Figure 3.3 illustrates a 25 frames long excerpt of this kind of data. In the first frame in the top left, the lateral offset of the vehicle can be observed by looking at the lane marking in the bottom right of the image. In order to recover from that position, the TrafficManager steers an s-shape back to the target trajectory. This begins with a right turn initially and merges into a left turn in the end. The data set contains 120 of those sequences with different initial positions and lateral offsets.

3.2 Training Procedure

The training procedure includes loading and pre-processing of the data, the actual training and the export of the trained model.

3.2.1 Data Generator including Image Processing

The illustrated data generator in figure 3.1 is responsible for creating training batches from the data sets and feeding them into a neural network. Depending on the neural network architecture, different generators are used. Every data generator uses the same steps for pre-processing, which includes cropping and scaling the images to the target resolution of 200x66x3 (width x height x channels) pixels, as well as converting the color space from RGB to YUV [17] to match the procedure proposed by NVIDIA. Image standardization is done inside the model, so it can be accelerated by the Graphics Processing Unit (GPU). Graphic 3.4 illustrates the original image captured and stored in the data set versus the processed (cropped and rescaled) image used during training.

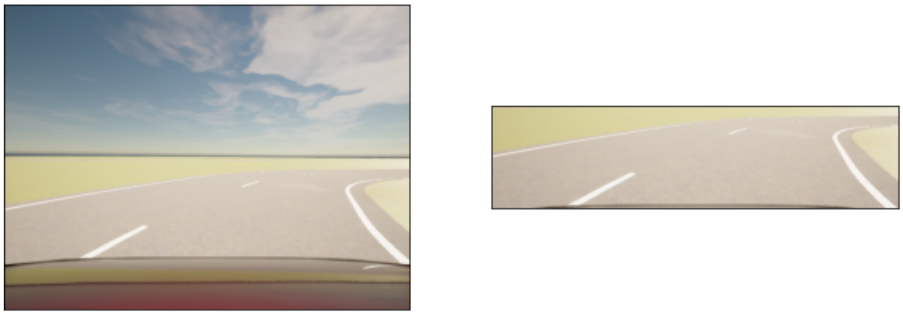


Figure 3.4: Example image with a resolution of 800x600x3 (width x height x channels) from Data Set (left) - cropped and scaled image only containing relevant parts of the image used for training with a resolution of 200x66x3 (right).

From a technical standpoint, the data sets are loaded into the main memory, where they are processed. The processed data sets are then accessed by the data generator to create batches in the training procedure. Accessing already processed data in the main memory accelerates the training process tremendously.

3.2.2 Model Training

Regardless of the different network architecture proposed in this work, training is always in a supervised manner treated as a regression problem. During the training, the training loss and validation loss is logged and displayed in Tensorboard. The model is saved in regular intervals. This allows to evaluate the model's performance at different points in time during the training process and encounters possible overfitting issues. Details on training hyperparameters such as batch size is provided in the corresponding sections of the model architectures. There is no test data set in the classical sense to rate each model's performance, which is described in the following section.

3.3 Performance and Behavior Evaluation Methods

There are no standard metrics for verifying model performance. Many publications use a test data set and take the Mean Squared Error (MSE) as a performance metric. Others use measures that describe the number of interventions in the steering in regard to the distance. Again others also consider ride comfort, for example, in terms of acceleration of the vehicle.
A frame by frame analysis expressed with the MSE of the model prediction is less useful. While the MSE is an indicator of training progress, the model has to actually drive to show how subsequent predictions of the model affect the performance. The evaluation of the model performance and behavior is directly compared to CARLA's TrafficManager in a driving test. We chose this method, because every model in this thesis is trained on the policy set by the TrafficManager and trying to imitate its behavior. The evaluation takes place on the s-shaped map 2 introduced in chapter 2.2. Although this map is not particularly challenging, it is well suited to demonstrate basic capabilities of the model and the influence of the steering offset on the model.

The TrafficManager navigates through the course on map 2 in order to create a baseline that every trained model can be compared to. While driving, the position of the vehicle and the steering angle are logged at every simulation tick. This creates a series of positions that are then calculated into a ground truth trajectory. Whenever a model is evaluated, it drives on the evaluation map and will log its position and steering output. Every model is evaluated twice: First, with no steering offset applied. Second, with the

steering offset applied. The following two sections introduce one metric each, which are used to quantify the models performance based on the logs.

3.3.1 Metric: Steering Output over Time

The first metric takes the steering output over time into regard. The model drives across the s-shaped evaluation map and the steering output is visually compared to the steering output of the TrafficManager. This comparison should give information about fluctuations in the steering signal and certain similarities in the steering behavior compared to the TrafficManager.

3.3.2 Metric: Deviation from Ground Truth Trajectory

The second metric will be the absolute distance from the model's position to the ground truth trajectory over time, when driving on the evaluation map. This gives information on how the current position deviates from the optimal position and under which circumstances this in- or decreases. "Wiggles" of the vehicle position can be observed using this method. Based on this metric, the Mean Absolute Error (MAE) in position can be calculated and will be used to quantify the model performance as well.

4 Set a Baseline: Recreating NVIDIA's PilotNet

This chapter introduces an implementation of NVIDIA's PilotNet architecture, which is integrated into the simulation environment from chapter 2 and uses the training pipeline described in chapter 3.

This chapter has the purpose of evaluating PilotNet in regard to a system discrepancy in the form of a steering offset in order to see, how it impacts the behavior of the vehicle. Section 4.1 introduces the neural network architecture. Section 4.2 gives details on the training process, including important hyperparameters and an overview of the training data with statistical measures. Section 4.3 presents the results based on the metrics from section 3.3 of the trained model with and without the steering offset.

4.1 PilotNet Architecture

Table 4.1 illustrates the CNN architecture of PilotNet with minimal changes to the original architecture. The input layer takes images with a resolution of 66x200x3 pixels (height x width x channels) encoded in the YUV color space. A standardization layer performs normal image standardization of the input and is not adjusted in the training process. Standardizing images in the network itself has the benefit of being GPU accelerated. Following that are five convolutional layers for feature extraction, the first three convolutional layers with a 5x5 kernel and a 2x2 stride, the last two convolutional layers with a 3x3 kernel and no stride. After the convolutional layers are 3 fully connected layers with decreasing size, leading to the final output layer, returning the steering angle. Compared to the original architecture from NVIDIA, this neural network uses Exponential Linear Unit (ELU) instead of Rectified Linear Units (ReLU) as activation functions for its layers and also contains two dropout layers. These minor changes are suggested by [18]. The architecture has roughly 250,000 parameters.

Layer Type	Stride	Activation	Output Shape	Params
Input	-	-	66x200x3	-
Standardization	-	-	66x200x3	-
Conv2D 5x5	2x2	ELU	24@31x98	1,824
Conv2D 5x5	2x2	ELU	36@14x47	21,636
Conv2D 5x5	2x2	ELU	48@5x22	43,248
Conv2D 3x3	1x1	ELU	64@3x20	27,712
Dropout 0.2	-	-	64@3x20	-
Conv2D 3x3	1x1	ELU	64@1x18	36,928
Flatten	-	-	1,152	-
Dropout 0.2	-	-	1,152	-
Dense	-	ELU	100	115,300
Dense	-	ELU	50	5,050
Dense	-	ELU	10	510
Output	-	-	1	11
				252,219

Table 4.1: NVIDIA's PilotNet Architecture - 252,219 Parameters

4.2 Training Details

The neural network is trained on single images labeled with the corresponding steering angle in degrees in a supervised fashion. Since the steering angles are continuous, the training is treated as a regression problem using the MSE as the loss function. The data used for training and validation is illustrated in a histogram in figure 4.1. This data is generated with the process described in section 3.1.1 and only contains data, that is aligned with the target trajectory. The data is split in 75% for training and 25% for validation.
Figure 4.1 illustrates that the data set is imbalanced. The most data is present around zero degree (neutral angle) with more than 500 collected images. The mean of the data is nearly zero degree and the standard deviation is 5.34 degrees. The data set includes 6,055 images in total.

The batch size is relatively large with 100 for a good representation of the distribution inside the batch.

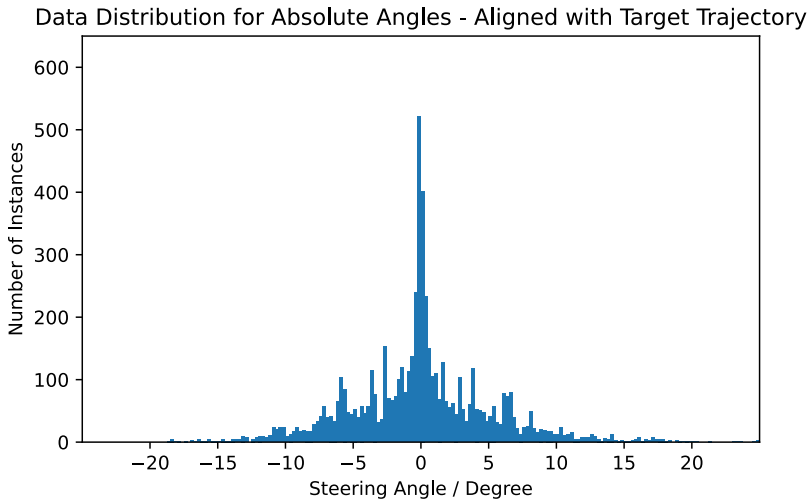


Figure 4.1: Histogram with bins of 0.25°width showing the distribution of data aligned with target trajectory. 6,055 Images - Mean: 0.008°- Standard Deviation: 5.33°

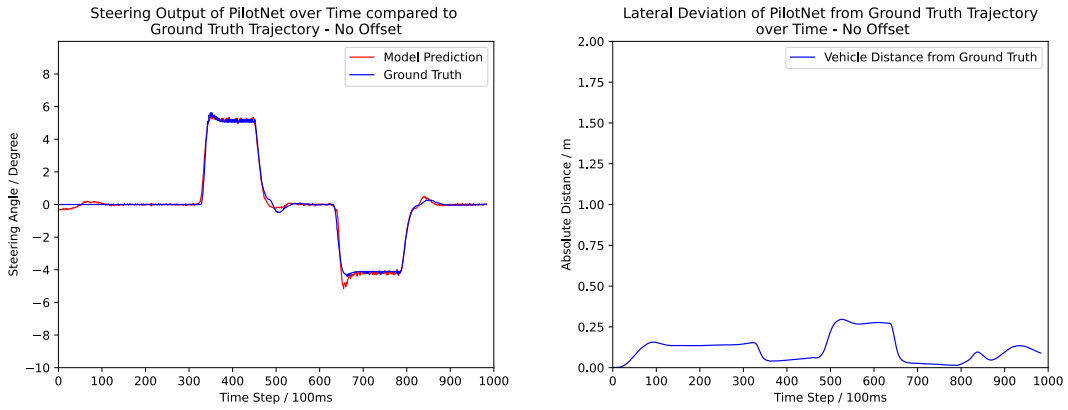
4.3 Behavior and Performance Results

In the course of training, the loss is converging to low level within a couple of minutes with no overfitting issues. The model from the end of the training process is now evaluated in this section based on the metrics described in 3.3 driving s-shaped map 2 introduced in section 2.2.

4.3.1 Performance without System Discrepancy

Figure 4.2a illustrates the ground truth steering signal over time (blue) compared to PilotNet's prediction (red). Figure 4.2b illustrates the deviation in position compared to the ground truth trajectory from PilotNet while driving.

In general, the driving performance of the model is very good. At the beginning between time step 0 and 100 some noticeable fluctuation of the model prediction can be observed. This temporary fluctuation results in a constant lateral displacement of the vehicle of about 0.15 m as illustrated in figure 4.2b. When reaching the right turn at time step 350 a slight overshoot in the ground truth as well as in the prediction can be seen. During the right turn, the absolute error in positioning is around 0.05 m. After the right turn



(a) Steering signal over time for PilotNet compared to the ground truth steering signal.

(b) Absolute distance from ground truth trajectory over time for PilotNet.

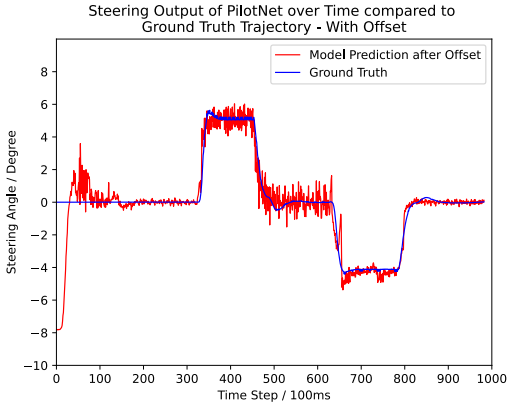
Figure 4.2: Performance results for NVIDIA's PilotNet driving on the s-shaped evaluation map with no steering offset.

at time step 500 some overshoot of the ground truth can be observed. The overshoot of the model is less aggressive at this point. On the straight segment between time step 500 and 650 the absolute error in positioning is between 0.25 m and 0.30 m and thus the highest error in positioning measured. Time step 650 marks the beginning of the left turn with some overshoot of the model and no overshoot of the ground truth. During the left turn the absolute error in positioning is again low with under 5 cm. After the left turn at time step 800 some overshoot of the model and the ground truth is once again noticeable.
The mean absolute error during this test is 0.11 m with a maximum deviation from the ground truth of 0.29 m.

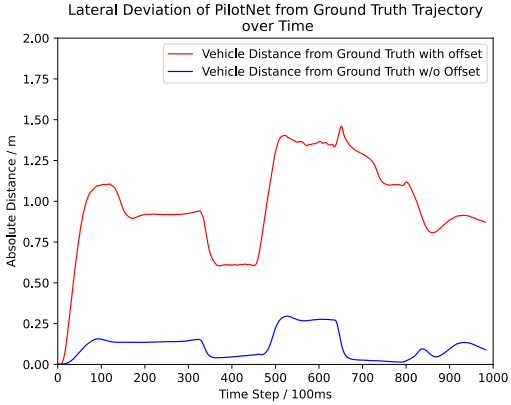
4.3.2 Performance with System Discrepancy

Before analyzing the graphs illustrating the performance of the model, an important distinction must be made in regard to the model output. Figure 2.2 from section 2.2.1 illustrated the client-server interaction. When looking at the driving logic inside the client, the steering signal can be either observed right after the neural network, or after the offset is applied. The output of the model after the offset is the signal that is sent as the steering command to the vehicle in the simulation and will be looked at in the following graphs.

The following figure 4.3 illustrates the results of the evaluation process described in 3.3. This steering offset is -7.5 degrees. Figure 4.3a illustrates the ground truth steering signal over time (blue) compared to the model's prediction after the offset (red). Figure 4.3b shows the deviation in position compared to the ground truth trajectory the model has while driving with the offset (red) compared to the result from the previous figure 4.2b, driving with no offset applied (blue).



(a) Steering signal over time for PilotNet compared to the ground truth steering signal.



(b) Absolute distance from ground truth trajectory over time for PilotNet driving with offset compared to the absolute distance driving without an offset.

Figure 4.3: Performance results for NVIDIA's PilotNet driving on the s-shaped evaluation map with steering offset.

Generally, the model is able to drive on track successfully from start to end. At time step 0 the model's prediction after the offset is -7.5 degrees, which corresponds to the offset. As the vehicle begins to move, it turns to the left, increasing the absolute error in positioning seen in the right graph. Due to the different lateral position of the vehicle, the model output increases to counteract the incorrect position. Between time step 50 and time step 200 notable fluctuations are observable, caused by the overshoot of the model counteracting the offset. After time step 200, the vehicle is in a stable position approximately shifted 1 m to the left. At this position the model's prediction after the offset is 0 degrees, which means the prediction of the model before the offset is 7.5 degrees and thus perfectly counteracting the offset. When entering the first right turn at time step 350, the model output changes accordingly, but contains some noise during the right turn. The lateral deviation in position decreases during the curve to 0.65 m and is constant throughout and not affected by the noise in the steering signal. When leaving the curve at time step 450, the error in position increases again to 1.4 m showing

the highest deviation in lateral position, the noise in the steering signal remains the same. Entering the left turn at time step 650, the error in lateral position decreases over time. At this point in time, whilst the noise in the steering signal becomes less frequent. Leaving the curve at time step 800 brings the error in lateral position back down to 1 m and the steering signal to a stable level.
The mean absolute error in positioning is 0.98 m with a maximum deviation from the ground truth of 1.46 m and is significantly higher than without an offset.

Figure 4.4 illustrates how an error in positioning of 1 m looks like from the vehicle's perspective. Image 4.4a shows the car driving on a straight segment with an offset of -7.5 degrees and being left-shifted of about 1 m. Figure 4.4b illustrates the vehicle driving with no offset on the straight segment and only being left-shifted of about 0.14 m.

(a) Vehicle driving with steering offset of -7.5 degrees being left-shifted of about 1 m.

(b) Vehicle driving with no steering offset being left-shifted only by 0.14 m.

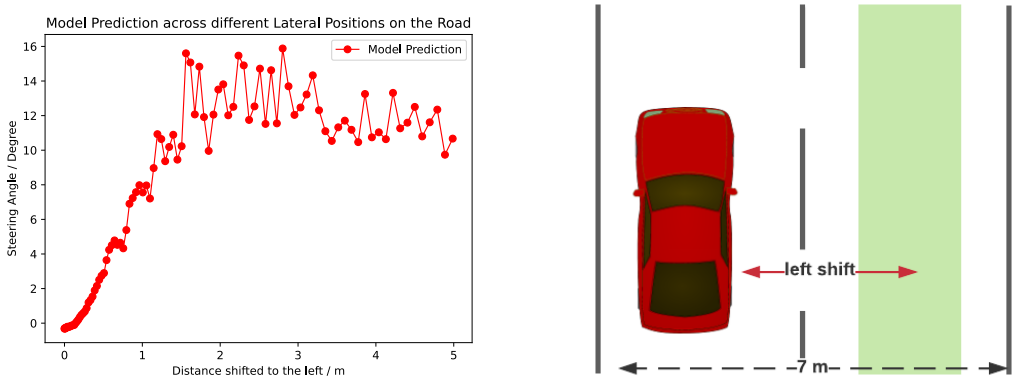
Figure 4.4: Comparison between lateral position of the vehicle with and without a steering offset applied from the vehicle's perspective.

4.3.3 Generalization Capability of the Model

Considering the conducted experiment in 4.3.2, an analysis of the model's prediction behavior dependend on the distance deviating from the ground truth could be interesting. Since the training data does not include scenarios where the vehicle deviates from the center of the right lane, this experiment demonstrates the generalization capabilities of the model.

Figure 4.5b illsutrates the experimental setup and figure figure 4.5a illustrates the result. The experiment is started with the vehicle being aligned with the target trajectory (green area). Now the vehicle is shifted to the left, increasing the lateral deviation from the ground truth trajectory (distance in red). The result, which is the model output

depending on the distance deviating from the ground truth trajectory, is illustrated in graph 4.5a. The graph shows that the model output is steadily rising until the first meter is reached. After that, the model prediction gets increasingly unstable with fluctuations of up to 3 degrees until it eventually begins to fall at around 3 meters. This graph confirms the measurements taken in section 4.3.2 in regard to the absolute steering error of approximately 1 m caused by a steering offset of -7.5 degrees. Moreover, the graph can be used to roughly determine the lateral deviation in position of the vehicle caused by a certain offset in the steering. For instance, an offset of -2 degrees would cause a deviation of 0.5 m in the lateral position.



(a) Model prediction across different lateral positions.

(b) Illustration of the vehicle being left-shifted on the road.

Figure 4.5: Results and experimental setup demonstrating the generalization capability.

4.4 Conclusion on NVIDIA's PilotNet Results

The training and evaluation of NVIDIA's PilotNet in the context of this simulation demonstrates the capabilities of the architecture. The results from section 4.3.1 show that the model has very good performance on the road with minimal deviation from the ground truth trajectory. This is accomplished with training on data, where the vehicle is exclusively aligned with the target trajectory. The results from section 4.3.2 demonstrate the influence of the steering offset applied to the model's output. It shows that such an offset causes the vehicle to depart from the center of the lane. An offset of -7.5 degrees (left) causes a left-shift of the vehicle of about 1 m on a straight road segment. The vehicle is able to drive with the lateral displacement on the road from start to finish. In this context, it is shown in section 4.3.3 that the model is able to generalize in scenarios

that are not covered in the training data.
Related to the experimental setup build for this thesis, the results demonstrate the basic functionality of the data gathering, model training and evaluation processes embedded in the CARLA simulator.

5 Proposal: PilotNetΔ

We propose a new architecture called PilotNetΔ (PilotNet Delta) with increased robustness against different steering offsets, characterized by a lower mean absolute error in positioning compared to the original PilotNet from chapter 4, when driving with an offset.

This chapter implements the new architecture which is integrated into the simulation environment from chapter 2 and uses the training pipeline described in chapter 3. This chapter presents changes in the architecture and evaluates PilotNetΔ in regard to the performance driving with and without a steering offset.

Section 5.1 introduces the neural network architecture. Section 5.2 gives details on the training process, including important hyperparameters and an overview of the training data with statistical measures. Section 5.3 presents the results based on the metrics from section 3.3 of the trained model with and without the steering offset.

5.1 PilotNetΔ Architecture

Table 5.1 illustrates the new CNN architecture of PilotNetΔ with some changes to the original PilotNet. The input layer takes a sequence of 3 images with a resolution of 66 by 200 by 3 pixels (sequence x height x width x channels) each, encoded in the YUV color space. Each image is spatially 0.5 m apart from the next one. A standardization layer performs normal image standardization and is not adjusted in the training process. Following that is a convolutional LSTM [19] layer with a 5x5 kernel and a 2x2 stride. This layer takes an input sequence and returns one set of feature maps per sequence. After that four more convolutional layers follow for feature extraction, the first two convolutional layers with a 5x5 kernel and a 2x2 stride as well, the last two convolutional layers with a 3x3 kernel and no stride. After the convolutional layers are 3 fully connected layers with decreasing size, leading to the final output layer. Compared to the original

architecture from NVIDIA, this neural network uses ELU as activation function as well for its layers and contains 4 dropout layers throughout the model. The architecture has roughly 315,000 parameters.

Layer Type	Stride	Activation	Output Shape	Params
Input Sequence	-	-	3x66x200x3	-
Standardization	-	-	3x66x200x3	-
ConvLSTM2D 5x5	2x2	ELU	24@31x98	64,896
Dropout 0.2	-	-	24@31x98	-
Conv2D 5x5	2x2	ELU	36@14x47	21,636
Conv2D 5x5	2x2	ELU	48@5x22	43,248
Conv2D 3x3	1x1	ELU	64@3x20	27,712
Dropout 0.2	-	-	64@3x20	-
Conv2D 3x3	1x1	ELU	64@1x18	36,928
Flatten	-	-	1,152	-
Dropout 0.2	-	-	1,152	-
Dense	-	ELU	100	115,300
Dense	-	ELU	50	5,050
Dropout 0.2	-	-	50	-
Dense	-	ELU	10	510
Output	-	-	1	11
				315,291

Table 5.1: PilotNetΔ Architecture - 315,291 Parameters

5.1.1 Model Output

The original PilotNet as described in chapter 4 trains on single images at a time ($image_{t0}$) labeled with the corresponding absolute steering angle (α_{t0}), creating the following mapping:

$$image_{t0} \mapsto \alpha_{t0} \tag{5.1}$$

The new PilotNetΔ has a different approach. The architecture takes the last three frames at a time $image_{t0}$, $image_{t-1}$, $image_{t-2}$ labeled with the amount the angle (α_{t0}) has to change compared to the previous angle (α_{t-1}), referred to as relative angle, creating the following mapping:

$$image_{t0} + image_{t-1} + image_{t-2} \mapsto \alpha_{t0} - \alpha_{t-1} \tag{5.2}$$

Since the architecture uses the difference (delta) of two values as an output, it is referred as PilotNetΔ. When using the neural network, the output over time is added to the absolute steering angle used as the steering command for the simulator.

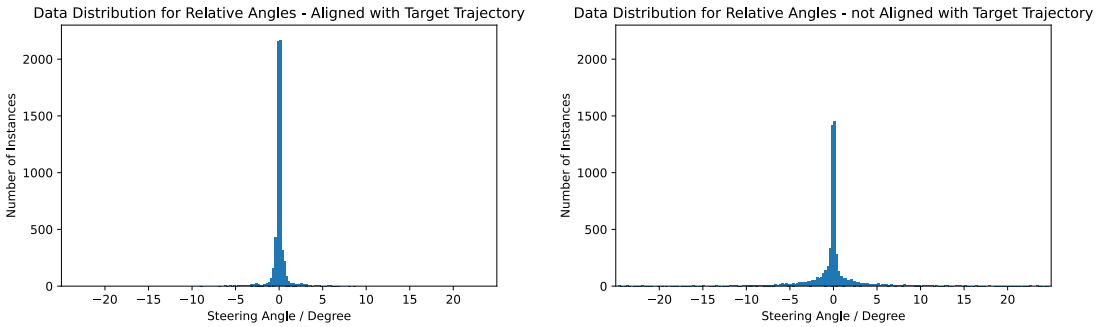
5.2 Training Details

Training the network continues to be a regression problem. We use a custom implemented loss function as variation of the MSE, which is illustrated below. y_{pred} is the predicted steering angle of the model, y_{true} is the ground truth steering angle:

$$\frac{1}{n}\sum_{t=1}^{n}(y_{pred} - y_{true})^2 * (|y_{true}| + 0.1) \tag{5.3}$$

The costs-sensitive loss function is a proposed solution for a problem resulting from the training data distribution for this architecture. This loss function is discussed in detail in section 6.3.1.

The following training data is used and distinguished in two groups:



(a) Histogram with bins of 0.25°width showing the distribution for relative angles of data aligned with target trajectory. 6,054 Images - Mean: -0.0001°- Standard Deviation: 0.84°

(b) Histogram with bins of 0.25 degree width showing the distribution for relative angles of not aligned with target trajectory. 5,920 Images - Mean: 0.0004°- Standard Deviation: 04.16°

Figure 5.1: Histogram comparison between relative steering angles for the data set containing data aligned with target trajectory to the data set containing data not aligned with target trajectory.

The first data set is illustrated in a histogram in figure 5.1a. This data set was generated with the process described in section 3.1.1 and only contains data, that is aligned with the target trajectory. In fact, the data illustrated in figure 5.1a is the same as the data from section 4.2 used for training the original PilotNet, but now in respect to the relative angles rather than the absolute angles. The data has a mean of nearly 0 and a standard deviation of 0.84 degrees. The vast majority of the data is around 0 degree contributing to a highly imbalanced distribution, with nearly no data deviation more than 5 degrees around the mean.
The second data set is illustrated in figure 5.1b. This data set was generated by the process described in section 3.1.2 and contains data that is not always aligned with the target trajectory. The data has a mean of nearly 0 and a standard deviation of 4.16, which is significantly higher than the data in 5.1a. The second data set also includes data deviating more than 5 degrees from the mean. The batch size is 200 for a good representation of the distribution inside the batch.

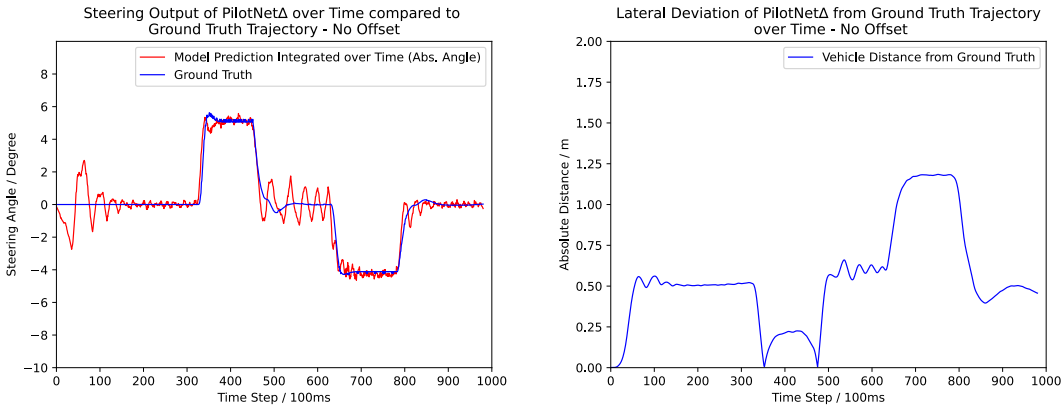
5.3 Behavior and Performance Results

Training PilotNetΔ took significantly longer and issues with overfitting were observed. Multiple models from the training process were examined. The results of the best performing model are presented in regard to the metrics described in 3.3, driving s-shaped map 2 introduced in section 2.2.

5.3.1 Performance without System Discrepancy

Figure 5.2 illustrates the results for PilotNetΔ driving with no offset. Figure 5.2a illustrates the ground truth steering signal over time (blue) compared to the integrated prediction of the model (red). Note that the model predicts relative angles, therefore this output is integrated over time resulting in the absolute steering angle that is used to steer the car. Figure 5.2b illustrates the deviation in position compared to the ground truth trajectory the model has while driving.

Overall, the performance of the model is good enough to drive the full course from start to finish. The right graph shows strong oscillations of the steering signal at the beginning of time step 0 with amplitudes of nearly 3 degrees. At time step 200 the oscillations decreased to a high frequent noise that is present throughout the experiment.



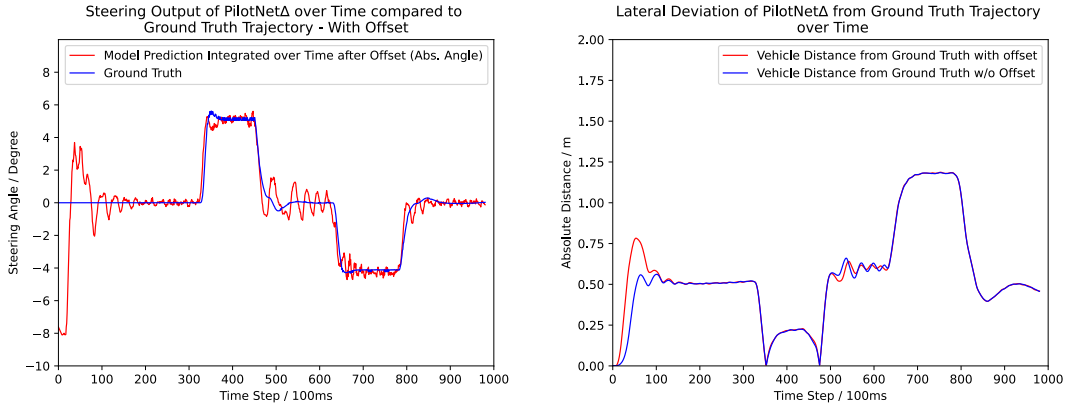
(a) Steering signal over time for PilotNetΔ compared to the ground truth steering signal.

(b) Absolute distance from ground truth trajectory over time for PilotNetΔ driving with offset compared to the absolute distance driving without an offset.

Figure 5.2: Performance results for PilotNetΔ driving on the s-shaped evaluation map without steering offset.

Interestingly, the lateral displacement seen in graph 5.2b is 0.55 m, even though no steering wheel offset is applied. When entering the first right turn at time step 300, the lateral displacement falls from 0.55 m to 0 m and then rises again to 0.2 m. At this point, the vehicle transitioned from the left side to the right side relative to the ground truth trajectory. The steering signal is around 5 degrees while driving through the curve. At time step 480 the vehicle transitions from the right side of the ground truth trajectory back to the left when leaving the curve, crossing it once again. At this point, the noise in the steering signal increases with amplitudes of ± 1 degrees, but with a relatively low frequency. Within 15 seconds between time step 450 and 600, the steering signal shows oscillation in five cycles, leading to a frequency of 0.33Hz. This oscillation of the steering signal has an effect on the lateral position. As observed in the graph 5.2b, the distance from the ground truth trajectory varies with distances in smaller than 0.1 m. Entering the left turn at time step 650, the steering signal changes to slightly less than -4 degrees. However, the lateral position increases in the course of the curve to up to 1.2 m, which is the highest value measured. Leaving the left turn at time step 800, the error in lateral position falls back to 0.4 m and seems to level off round 0.5 m. The steering signal remains noisy around zero degrees. The mean absolute error in positioning is 0.55 m with a maximum deviation from the ground truth of 1.18 m.

5.3.2 Performance with System Discrepancy



(a) Steering signal over time for PilotNetΔ compared to the ground truth steering signal.

(b) Absolute distance from ground truth trajectory over time for PilotNetΔ compared to the ground truth trajectory.

Figure 5.3: Performance results for PilotNetΔ driving on the s-shaped evaluation map with steering offset.

The following figure 5.3 illustrates the results of driving with the steering offset of -7.5 degrees. We differentiate again between the model output before the offset and the model output after the offset.

Figure 5.3a illustrates the ground truth steering signal over time (blue) compared to the integrated prediction of the model after the offset (red). Figure 5.3b shows the deviation in position from the ground truth trajectory of the model while driving with an offset (red) compared to the result from the previous figure 5.2b, driving with no offset applied (blue).

As seen in figure 5.3b, the lateral positions are nearly identical between driving with an offset and driving without an offset. The only difference is in the beginning phase between time step 0 and time step 100. The model output in figure 5.3a starts with a steering angle of nearly 8 degrees, mainly due to the offset. This causes the vehicle to drive left, which increases the lateral deviation from the ground truth trajectory. The model quickly counteracted this within the first 7 seconds. A bit of overshoot in the lateral deviation is measurable. The remaining behavior after time step 100 is identical to the observations in section 5.3.1. The mean absolute error in positioning is 0.57 m with a maximum deviation from the ground truth of 1.19 m.

5.4 Conclusion on PilotNetΔ Results

The proposed PilotNetΔ architecture is significantly harder to train. With some changes to the training process, including loss function, a larger data set and batch size, the trained model was able to achieve moderate to good results when simply driving on the road. When driving with the steering offset, the results are nearly the same as diving without an offset. The moderate performance is characterized by the lateral displacement of 0.57 m of the vehicle when simply driving on the road in combination with the fluctuations present in the steering. In general, the model is quite sensitive to the input data but successfully driving from start to end.

6 Discussion

This chapter compares the results of the two architectures examined (see section 6.1). In particular, it discusses the difficulties PilotNetΔ faces due to data quality and error-proneness (see section 6.2). In addition, necessary design decisions, including loss function and LSTM Layer, for PilotNetΔ are explained (see section 6.3), as well as design ideas that have proven to be non-functional (see section 6.4).

6.1 The impact of Steering Offset on PilotNet and PilotNetΔ Contradicted

The following table 6.1 concludes the driving performance described in chapter 4 for PilotNet and chapter 5 for PilotNetΔ in regard to the mean absolute error of the vehicle position with and without the steering offset.

Architecture	MAE	Max Error	MAE with Offset	Max Error with Offset
NVIDIA's PilotNet	0.11 m	0.29 m	0.98 m	1.46 m
PilotNetΔ	0.55 m	1.18 m	0.57 m	1.19 m

Table 6.1: MAE of PilotNet and PilotNetΔ during the evaluation process, driving with and without an offset.

The results on the PilotNet architecture demonstrate the architecture's capability, when driving under normal conditions, meaning without the steering offset. The deviation from the ground truth trajectory measured with the MAE of the lateral position on the road is only 0.11 meter, which is half the width of a typical car tire. When the steering offset is engaged, the results show that the architecture is not able to handle the problem well. As a result, the vehicles is shifted in average 1 meter to the left, when the steering offset is -7.5 degree. This is a misconduct, that is not acceptable in the real world, assuming a car width of 2 meters and the typical width of a German "Landesstraße" (equivalent to

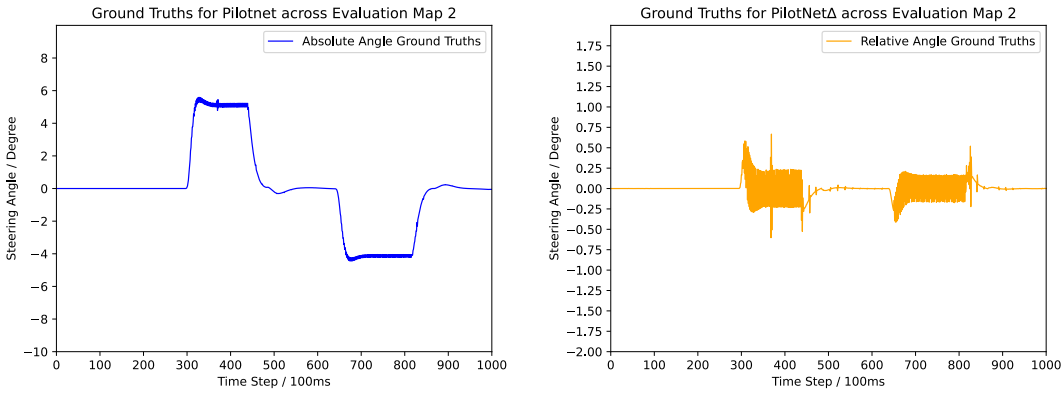
U.S. State Routes) of 3 meters. It is even worse when considering a maximum deviation of nearly 1.5 meters.

Among the findings of the PilotNetΔ architecture is that the MAE is 0.55 meters when driving under normal conditions. This is an interesting result because the vehicle tends to measurably be shifted to the left even though an offset is not applied. In the context of a real-world scenario, this must be evaluated critically. Moreover, it was measured in section 5.3.1 that the steering output contains noise throughout the experiment but was not affecting the vehicle's position due to the inertia of the vehicle steering. Besides the noise however, it was the case that the steering oscillated at some point with a frequency of 0.33Hz leading to an oscillation in the position of the vehicle. Using the measurements taken, it can be stated that the driving performance of the vehicle under normal conditions is better with the PilotNet architecture proposed by NVIDIA.

The most significant findings are in the measurements of PilotNetΔ driving with the steering offset. Regarding the MAE of the lateral position on the road, PilotNetΔ shows a lower MAE than PilotNet. Futhermore, the MAE is nearly unaffected between PilotNetΔ driving with an offset and driving without an offset. This demonstrates, that PilotNetΔ itself has a robustness against the steering offset, characterized by the fact that the general performance is nearly unaffected by the system discrepancy. PilotNetΔ has a clear advantage in this regard.

6.2 Difference between PilotNet and PilotNetΔ in regard to Data Distribution, Data Quality and Error Susceptibility

PilotNet and PilotNetΔ follow two different principles, steering the vehicle. PilotNet uses absolute angles and answers the question: 'What is the ideal position of the steering wheel?'. However, PilotNetΔ uses relative angles that are added up to an absolute steering angle and answers the question: 'How to change the steering wheel position?'. In mathematical terms, the relative angle is the difference quotient between two absolute steering angles.
Figure 6.1 exemplifies the relationship between relative- and absolute steering angles on the ground truth steering signal that was also used during the evaluation experiments described in section 3.3. These ground truths were not used for training. The left



(a) Ground truths (absolute steering angles) over time.

(b) Ground truths (relative steering angles) over time derived from the absolute steering angles.

Figure 6.1: Ground truth steering signal on map 2 over time, generated by CARLA's TrafficManager.

figure 6.1a illustrates the absolute steering angle signal generated by the TrafficManager driving on the s-shaped evaluation map. The figure 6.1b on the right illustrates the relative steering angle, which is the numerical differentiation of the steering signal from the left graph. Alongside these graphs, we can make a statement on the data quality, data distribution and the principles of steering a vehicle with relative angles.

6.2.1 Data Quality

The data quality for training PilotNet is none of a concern, as the results for driving under normal conditions show. However, data generated by the TrafficManager includes features that are suboptimal. One of those features is the small overshoot every time when leaving or entering a curve. This can be seen in the right graph in figure 6.1a at time steps 320, 500, 680 and 880. This is a standard phenomenon in control theory[20] and caused by the underlying PID controller in the TrafficManager. PilotNet learns this behavior, but is less of a concern due to the subtlety.

Another concern is noise (seen as a thicker blue line in the same graph) in the steering signal during the left and right turns between time step 300 to 450 and 680 to 820.

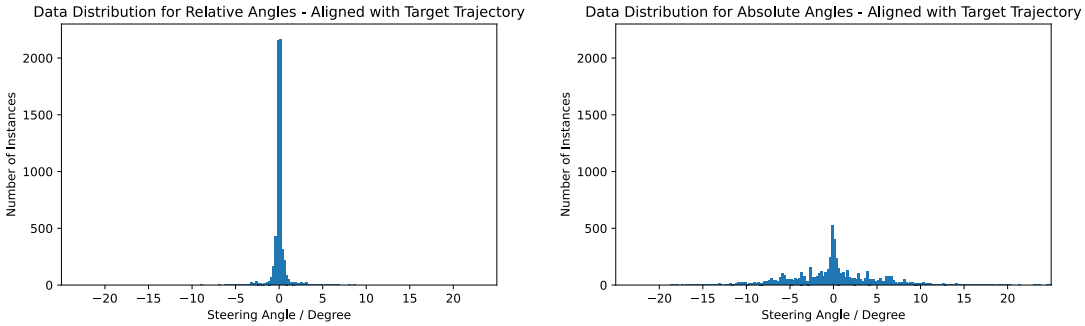
Figure 6.1b illustrates the derived relative angles. Where there was relatively low noise between time step 300 to 450 and 680 to 820 for the absolute steering angles, this noise is significantly increased for the relative steering angles. Even a small outlier in the absolute steering signal at time step 360 causes a huge spike in the steering signal for relative angles. The general circumstance that relative steering angles are very sensitive to noise is unfavorable because noise leads to contrary data.
Even though the sequence of the TrafficManager driving on the s-shaped evaluation track is not part of the training data, the observations made here do apply to the training data in general.

Further considerations of advanced filter- or differentiation methods to improve the data quality might be useful. Low pass filtering the data with a simple rectangular filter led to no training success. Apart from that, it must be considered if the benefit of automatized generation of data with the TrafficManager outweighs the time and effort it takes to manually steer the vehicle by hand, if this would lead to a better data quality.

6.2.2 Data Distribution

Figure 6.1b also gives an idea, how the output of PilotNetΔ must behave in order to drive throughout a curve. When entering the first right turn at time step 300, the model output spikes for a couple of time steps, meaning the steering wheel rotates to the right. When leaving the curve, the model prediction must spike again, in order to turn the steering wheel back to the initial position. The principle of steering the vehicle with relative angles leads to a highly imbalanced data distribution. Steering angles, who deviate from zero degrees, occur less frequently. If they occur, however, they tend to be smaller, which is unfavorable. To clarify this, the following figure 6.2 contradicts the data set with absolute angles for PilotNet to the dataset containing the same images but with relative angles for PilotNetΔ.

The side by side comparison illustrates the negative influence of relative angles on the data distribution. Figure 6.2a illustrates the data distribution for absolute steering angles, figure 6.2a illustrate the data distribution for relative steering angles. The differences already mentioned are clearly recognizable. As a consequence, the training process with relative angles is more difficult, because the respective data that leads to a significant change in the steering is very rare.



(a) Histogram with bins of 0.25°width showing the distribution for relative angles of data aligned with target trajectory. 6,054 Images - Mean: -0.0001°- Standard Deviation: 0.84°

(b) Histogram with bins of 0.25°width showing the distribution for absolute angles of data aligned with target trajectory. 6,055 Images - Mean: 0.008°- Standard Deviation:5.34°

Figure 6.2: Histogram comparison between absolute steering angles and relative steering angles for the set of images.

6.2.3 Error Susceptibility

From a theoretical standpoint, PilotNetΔ is more error-prone than PilotNet. The reason for this is that in case of PilotNetΔ, an incorrect prediction consequently changes the steering wheel position permanently. Thus, a wrong prediction is integrated over time, which leads to a deviation from the target trajectory. To achieve a similar effect in case of PilotNet, the neural network would not only have to make a false output once, but instead multiple times in a row.

6.3 Crucial Design decisions contributing to Training success of PilotNetΔ

In the introductory chapter 5 to PilotNetΔ, the architecture was presented without going into more detail about design decisions. In light of the information from section 6.2 regarding data quality, data distribution and error susceptibility, this section discusses key design decisions necessary to ensure a usable result.

6.3.1 Distribution re-weighing in Loss Function

There are several ways to compensate for an imbalanced data set. One way is to over sample or under sample data that is under- or overrepresented. This method is described in [21]. Although this method is mostly used for classification problems, it would also be suitable for our regression problem. On the other hand, there is the possibility of cost-sensitive learning, as described in [22]. Here, incorrect model prediction is associated with a cost factor that must be minimized. Our self-implemented loss function is a form of cost-sensitive learning, which has been implemented in a very simple form.
The following illustrates the loss function again, y_{pred} is the predicted steering angle and y_{true} is the ground truth steering angle:

$$\frac{1}{n}\sum_{t=1}^{n}(y_{pred} - y_{true})^2 * (|y_{true}| + 0.1) \tag{6.1}$$

This function first calculates the squared error of the model prediction compared to the ground truth. This error is then multiplied with a cost-sensitive factor that is mainly the amount of the ground truth angle. In our case, high amount of the ground truth steering angle correlates with a low probability in the distribution. Therefore, the loss of a data sample with a large steering angle, which is relatively rare in the distribution but very important for steering the vehicle, is associated with a higher cost. The loss of a data sample around zero degree and is associated with lower cost. The resulting effect is that a rare data sample weighs heavier in the mean squared error than a sample which is around zero degrees, thus containing less important information. For data samples with a ground truth of exactly zero, a problem would occur when using zero as the multiplying factor because it would completely disregard the loss of those samples. Therefore, our multiplying factor in the loss function includes a small constant to avoid this issue. We have made tests where we abandoned the small constant, leading to similar but a less good result. If we just use the standard MSE, no functional model could be trained that is able to drive from the start of the s-shaped evaluation map to the finish. Thus, it has been shown, that our loss function is crucial for the success of the training procedure. However, more research must be conducted in order to quantify the impact of the loss function, especially in regard to the small constant in the multiplying factor and the question of what actually leads to a better training result. Is it actually the re-weighing effect on the data, or is it some kind of low pass filter that suppress noisy data around zero

degrees, when calculating the loss? Furthermore, how sensitive is the new loss function to outliers in the ground truth data?

In a publication related to NVIDIA's research [23], another general problem with the mean squared error as an error function is described. This problem is also related to the PilotNetΔ architecture presented by us, to an extent that is possibly even more severe than in the original paper described.
The problem described deals with the time a prediction is made. Consider the PilotNetΔ model being in front of a left turn. The output of the model will spike at some point when entering the curve. In a real-world scenario it does not matter wether the vehicle steers a tenth of a second too late or too soon into the curve. However, when the model is in training, it could get punished by a very high loss when predicting a curve too early or too late, even though the behavior outside of the training is tolerable.
Therefore, additional considerations are needed. One solution might be to calculate the loss as it would be if the prediction was made at the correct time, but then punish it with a factor proportional to the time delay. This could lead to a model that is not punished in training for tolerable behavior, but would also be encouraged making predictions at the right time.

6.3.2 Using more Complex Training Data

Training PilotNetΔ on the same data as PilotNet, but with relative angles, was not sufficient enough to achieve a functional model as a result. Therefore, another data set was used in the training process described in section 5.2. The fact that this data set is necessary for the training success has mainly two reasons.

The first reason is, that this data set covers scenarios of the vehicle not being aligned with the target trajectory. From these scenarios, the model learns how to recover from a false position. This strengthens the robustness in case of a false steering decision, which is especially necessary for PilotNetΔ, as it is more error-prone than PilotNet. Including such data is a method that is also described in one of NVIDIA's publications [11] and addresses a known problem in the field of imitation learning that is described in various other publications such as [24].
The second reason is that the sequences in the additional data set contain many more changes in between the frames and the steering, leading to a higher standard deviation for relative angles, which is beneficial for the training process.

6.3.3 Hyperparameter Choice

The right choice of hyperparameters is a crucial one. Although the architecture of the neural network is very similar to PilotNet, the convolutional LSTM layer raises the question of the correct sequence length, i.e. how many images are fed into the network before an output is generated. If the sequence length is too long, there is the danger that wrong long-term dependencies are learned. If the sequence length is too short, necessary information which are needed for a correct steering decision are missed. Similar considerations can be made in regard to the FPM used in the process of data generation. If the FPM are too high, the images do not differ enough from each other and contain a lot of redundant information. If the FPM are too low, important information can be missed.
PilotNetΔ was trained with the parameters presented in chapter 5 where the sequence length was varied during the development process. Among others, sequence lengths of 1, 3, 5, and 10 were tried, with 3 proving to be functional. The other lengths did not result in successful training process. Not successful means the loss was either not converging or the model output biased towards the mean of the data to an extent, where the model prediction was constant. Similarly, frames per meter were varied and tested between both 2 frames per meter and 1 frame per meter, with 2 frames per meter proving functional.

Overall, the study of hyperparameter selection is not conclusive, because the difficulty consists in the correct combination of the right hyperparameters. It is the case that the sequence length, Frames Per Meter as well as the Loss Function and batch size all led to success only in the right combination. As soon as one thing was dispensed, the loss in training no longer converged or the trained model was simply not functional, in the sense that it could not keep itself on the road for a longer time.

6.4 General Ideas that did not Solve the Problem with the System Discrepancy

This section contains some initial ideas that came up during the development of the PilotNetΔ architecture, but did not contribute to the improvement of the steering offset problem.

6.4.1 Extend Training Data by Images with different Steering Offsets

One first idea might be to use NVIDIA's PilotNet and add additional data to the training data set, including data where the TrafficManager drives on the road with different steering offsets applied while gathering data, in order to gain robustness. However, this does not work with supervised learning, because it would generate contrary data.

6.4.2 Reinforcement Online Learning on the Steering Offset

Another idea is to use the trained PilotNet and use reinforcement online learning to train the model on a specific system discrepancy while it's driving on the road. For instance, with transfer learning only training the last couple of dense layers. In order to accomplish this, a cost function, or more general an optimization function, is needed to give feedback to the model in the training process. This function could be the distance between the vehicle and the center of the lane. This would be conceivable in a simulation where this information can be queried but is impossible in a real world, without further ado. Lastly, the step from a simulation to the real world is precisely the one that often causes the system discrepancies.

6.4.3 Low pass Filtering the Training Data

Two attempts were made by low pass filtering the relative angles with a rectangular low pass filter and kernel size of either 3 or 6. This reduces the noise dramatically, but does not eliminate noise in the data entirely. One problem with low pass filtering the data might be, that this induces some inertia in the steering signal and generates contradicting data. The steering angles at a certain time from the data set are then dependent on what the steering angle, and hence the road condition in the future is, which obviously varies.

6.4.4 Feed the Current Absolute Steering Angle into PilotNetΔ as additional Input

In the development process of the PilotNetΔ architecture, we tried whether is it helpful to feed the integrated output of the model, i.e. the absolute steering angle, back into the model. If this is done, the training process is accelerated enormously. However, the driving characteristics of the model are the same as with the original PilotNet architecture

from NVIDIA. This is because a fixed relationship between the absolute steering angle and the position on the road is then learned again. Exactly this relationship must be avoided. The aim is not to learn a relation between the position of the steering wheel (which could be changed variably by an offset) and the input images. The goal is to steer the vehicle, completely independent of the current steering wheel position. The relevant information is how the vehicle has already moved throughout the input sequence.

7 Conclusion

In this work, we implemented a pipeline around the CARLA simulation environment. The pipeline allows to generate data sets, use the data in the further training process to train different neural network architectures, and evaluate them afterwards. With this experimental setup, we implemented PilotNet, an architecture presented by NVIDIA. We used the results from PilotNet to validate the general functioning of the pipeline. More importantly, the general performance of the architecture is demonstrated. Although the general driving performance of PilotNet is quite good, it is shown that the architecture struggles with a steering offset. A steering offset causes a lateral displacement of the vehicle on the road. An offset of -7.5 degrees, for instance, causes a left shift of about 1 meter, which is not acceptable in a real-world scenario.

We proposed and evaluated a prototype architecture called PilotNetΔ, which has a robustness against the steering wheel offset. When comparing PilotNetΔ driving with an offset to PilotNetΔ driving without an offset, there is hardly any measurable difference in the lateral position, which previously was a problem caused by the steering offset. However, some issues still remain with PilotNetΔ.
This includes problems with the data quality in terms of data distribution and data noise. The effect of those issues on the driving performance is very difficult to examine without an alternative data set with better quality for comparison. To overcome possible problems caused by the data quality, a lot of effort has been put into architectural changes. This changes include additional dropout layers and increased batch size for a more stable training process and a customized loss function to handle the imbalanced training data distribution. Attempts including low pass filtering the data to reduce the noise led to no improvements in the training process. Hyperparameters for PilotNetΔ still need further investigation.
What is left is the empirical evidence that it is possible to steer a vehicle with relative angles in the context of this simulation setup, which is a success in itself. Even though the driving performance could be improved, the requirement to not deviate from the road

throughout the experiment is high and fulfilled by the model.
In addition, the idea of relative steering angles is characterized by the fact that it essentially solves the problem of steering wheel offset from a mathematical standpoint.

7.1 Outlook

In order for the PilotNetΔ architecture to be transferred to a real-world application, the training dataset must be extended to include significantly more complex data and ideally images from the real world. NVIDIA's has proven in the past, that learning complex image features with PilotNet is possible. In this context, the data set should be evaluated beforehand and a method for generating this data should be chosen, so that as little noise as possible is present in the collected data. The distribution of the collected data should be considered and, if necessary, a method presented in the paper should be used to reduce the imbalance such as re-weighting with a cost-sensitive loss function or over/under sampling of the data. With a data set of better quality, further investigations should also be made regarding the sequence length in order to further optimize this important parameter. If the architecture proves to be functional in the real world, which can be assumed from the results so far, the advantage of PilotNetΔ regarding the steering offset will have to be re-evaluated and the associated difficulties weighed up.
Of course, there is nothing to prevent PilotNetΔ from being combined with other end-to-end architectures. For example, [8] already has convolutional LSTM layers and could also be trained on relative steering angles instead of absolute steering angles.

Bibliography

[1] J. Almotiri, K. Elleithy, and A. Elleithy, "Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition," in *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2017, pp. 1–5.

[2] T. S. Madhulatha, "An overview on clustering methods," *ArXiv*, vol. abs/1205.1117, 2012.

[3] N. Chehata, N. David, and F. Bretar, "Lidar data classification using hierarchical k-means clustering," 2008.

[4] S. Ezz-ElDin, O. Nabil, H. Murad, F. Adel, A. AbdEl-Jalil, K. Salah, and A. Khan, "Mini-ssd: A fast object detection framework in autonomous driving," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2020, pp. 0377–0380.

[5] D. Tran, P. Fischer, A. Smajic, and Y. So, "Real-time object detection for autonomous driving using deep learning," 03 2021.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," *arXiv e-prints*, p. arXiv:1604.07316, Apr. 2016.

[7] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car," *arXiv e-prints*, p. arXiv:1704.07911, Apr. 2017.

[8] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies," *arXiv e-prints*, p. arXiv:1710.03804, Oct. 2017.

[9] S. Hecker, D. Dai, and L. Van Gool, "End-to-End Learning of Driving Models with Surround-View Cameras and Route Planners," *arXiv e-prints*, p. arXiv:1803.10158, Mar. 2018.

[10] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end Driving via Conditional Imitation Learning," *arXiv e-prints*, p. arXiv:1710.02410, Oct. 2017.

[11] M. Bojarski, C. Chen, J. Daw, A. Değirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, L. Jackel, Z. Jia, B. Lee, B. Liu, F. Liu, U. Muller, S. Payne, N. Kota Nagendra Prasad, A. Provodin, J. Roach, T. Rvachov, N. Tadimeti, J. van Engelen, H. Wen, E. Yang, and Z. Yang, "The NVIDIA PilotNet Experiments," *arXiv e-prints*, p. arXiv:2010.08776, Oct. 2020.

[12] D. Ho and K. Rao. Toward Generalized Sim-to-Real Transfer for Robot Learning. Accessed: 2021-10-07. [Online]. Available: https://ai.googleblog.com/2021/06/toward-generalized-sim-to-real-transfer.html

[13] (2021) CARLA 0.9.11 Documantation. Accessed: 2021-10-03. [Online]. Available: https://carla.readthedocs.io/en/0.9.11/

[14] (2021) RoadRunner. Accessed: 2021-10-03. [Online]. Available: https://de.mathworks.com/products/roadrunner.html

[15] (2021) CARLA TrafficManager. Accessed: 2021-10-03. [Online]. Available: https://carla.readthedocs.io/en/0.9.11/adv_traffic_manager/

[16] What is a PID Controller : Working & Its Applications. Accessed: 2021-10-03. [Online]. Available: https://www.elprocus.com/the-working-of-a-pid-controller/

[17] Wikipedia, "YUV — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=YUV&oldid=1046246569, 2021, accessed: 2021-10-07.

[18] D. Tian. (2019) DeepPiCar — Part 5: Autonomous Lane Navigation via Deep Learning. Accessed: 2021-10-03. [Online]. Available: https://towardsdatascience.com/deeppicar-part-5-lane-following-via-deep-learning-d93acdce6110

[19] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," *arXiv e-prints*, p. arXiv:1506.04214, Jun. 2015.

[20] Wikipedia, "Overshoot (signal) — Wikipedia, the free encyclopedia," http://en.wikipedia.org/w/index.php?title=Overshoot%20(signal)&oldid=1010083161, 2021, accessed: 2021-10-08.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *arXiv e-prints*, p. arXiv:1106.1813, Jun. 2011.

[22] C. Ling and V. Sheng, "Cost-sensitive learning and the class imbalance problem," *Encyclopedia of Machine Learning*, 01 2010.

[23] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1856–1860.

[24] P. de Haan, D. Jayaraman, and S. Levine, "Causal Confusion in Imitation Learning," *arXiv e-prints*, p. arXiv:1905.11979, May 2019.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

End-to-End Deep Learning für die Spurhaltung von selbstfahrenden Autos mit dem Fokus auf die Robustheit gegenüber Systemdiskrepanzen in der Lenkung

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original