Hybrid Human-Centric Haptic Shared Control using Artificial Neural Network and Model Predictive Control

Master Thesis

by

Hüseyin Harmankaya

Delft University of Technology

May 10th, 2023

TOYOTA

TU Delft Delft University of Technology

HYBRID HUMAN-CENTRIC HAPTIC SHARED CONTROL USING ARTIFICIAL NEURAL NETWORK AND MODEL PREDICTIVE CONTROL

MASTER THESIS

to obtain the degree of Master of Science in Robotics

at the Delft University of Technology,

to be publicly defended on Wednesday 17th of May, 2023,

by

Hüseyin HARMANKAYA

Supervisors:

Dr. ir. B. Shyrokau	CoR-IV, Delft University of Technology
Ir. A. M. R. Lazcano	UX-DS, Toyota Motor Europe

Thesis committee:

Prof. dr. ir. R. Happee,	Chair	CoR-IV, Delft University of Technology
Dr. ir. B. Shyrokau,	Supervisor	CoR-IV, Delft University of Technology
Dr. ir. D. M. Pool,	Committee member	AE, Delft University of Technology

Student number:	4674057
Project period:	June 6th, 2022 - May 10th, 2023

The work in this master thesis was performed at Toyota Motor Europe, Zaventem, Belgium, to whom the author is grateful for their support.

An electronic version of this thesis is available at http://repository.tudelft.nl/.

This thesis is confidential and cannot be made public until May 17th, 2025.

ACKNOWLEDGEMENTS

I have had the pleasure of experiencing what it is like to work in the industry I am so passionate about, the automotive industry, and doing so at one of the largest car manufacturers. It has made me grow both intellectually and personally. Therefore, I would like to extend my sincere gratitude to everyone involved in this challenging project during these challenging times dominated by Covid-19.

First of all, I would like to thank my supervisors. At Delft University of Technology, Barys Shyrokau has been a great source of inspiration and support with his extensive automotive knowledge. He always made time in his busy schedule to answer my questions and motivated me to get the most out of myself by believing in me. He is a true example of someone whose actions speak louder than words. At Toyota Motor Europe, Andrea Lazcano has always been ready to give me advice and extensive feedback at any given time. While being at the early stages of her career, she has already taken on a lot of responsibilities which is something I admire and look up to.

I would also like to thank my manager at TME, Xabier Akutain, for giving me this opportunity and making sure that I did not lose the bigger picture of my project. He also made me improve my race craft to stay competitive on the racing simulator. Moreover, my gratitude goes to the rest of the UX-DS team for their kindness and making me feel at home.

To my parents and sisters, who have always supported and trusted me with any decisions I have made and encouraged me to be the best version of myself. They have been there during the highs and lows, cheering me up in the most stressful moments.

To my lovely fiancé, who is always there for me no matter what. She helped me to stay motivated and always makes me laugh. Without her support, this project would not have been the way it is.

Lastly, to my friends and family, for all the fun activities to recharge from all the work.

Hüseyin Harmankaya
The Hague, May 2023

ABSTRACT

Advanced Driver Assistance Systems have become widely available in modern commercial vehicles. These systems bring safety to the roads and comfort to the driver by minimising human errors and lowering the mental and physical workload. However, most of these systems fail to consider the driver in the control loop. For instance, conventional Lane Keeping Assists are often set to strictly follow the centre of the lane while the driver shows non-linear behaviour during the steering task. This conflict in intentions between the assist and driver could result in it being turned off, eliminating the safety aspect.

Therefore, this study proposes a hybrid controller for a human-centric haptic shared Lane Keeping Assist, pairing a data-driven driver model with a model-based controller to foster the collaboration between the driver and assist. First, where parametric approaches struggle to model the driver's non-linear behaviour, this study proposes a Bidirectional Long Short-Term Memory network to learn the driver's behaviour from driving data where it predicts the steering wheel torque inputs. Even though models have been developed during this research with high accuracy surpassing the performance of previous work, it was found that the smoothness of the prediction is more substantial to create a pleasant human-centric assist. Thus, the proposed driver model has been tuned to have an accuracy ≥72.4% and a smoothness ≥0.85Nm/s over a 0.4s prediction horizon, where the smoothness metric is the Standard Deviation of the torque rate.

Second, a Model Predictive Controller is developed with a linear bicycle and steering model. The real-time predictions made by the driver model are used as a time-varying reference to the controller, along with the reference path. The Model Predictive Controller optimises the control input over the 0.4s prediction horizon, where it balances between accurately following the centre of the lane and adhering to the driver model's prediction based on the tuned weights.

To test the hybrid controller, three versions of the human-centric controller were developed for comparison and a state-of-the-art commercial solution was used as the baseline Lane Keeping Assist. The experiments were performed in Toyota Motor Europe's fixed-base driving simulator, where 15 participants tested and evaluated the four controllers. The objective results show a 113.1% increase in collaborative ratio while maintaining a similar path tracking performance compared to the baseline. Additionally, the subjective results show a significant preference for the proposed controllers by the participants over the baseline.

In conclusion, the proposed hybrid controller significantly improves the human-machine interaction by reducing conflicts and fostering collaboration.

Keywords: Haptic shared control, data-driven driver modelling, Artificial Neural Network, Model Predictive Control, human-machine interaction.

CONTENTS

ACRONYMS

ACC Adaptive Cruise Control. 28, 50

ADAS Advanced Driver Assistance Systems. 1, 16, 26, 28

ANN Artificial Neural Network. viii, ix, 2–4, 15–23, 26–28, 31–39, 45, 47–54, 56, 60, 62, 66–68

BiLSTM Bidirectional Long Short-Term Memory. viii, ix, 34, 35, 38, 39, 41, 43–46, 48

CAVIA Fast Context Adaptation Via Meta-Learning. ix, 67, 68

CNN Convolutional Neural Network. 19, 21, 27, 28, 32, 37, 38, 45

EMG Electromyography. 34, 35, 46

FANN Feedforward Artificial Neural Network. viii, 19–21, 27, 29–37, 45

FNN Fuzzy Neural Network. 31, 37

HG Haptic Guidance. viii, 33, 36, 38, 39, 46

HMM Hidden Markov Model. 2, 15, 16, 28, 31, 45, 48

KPI Key Performance Indicator. ix, 3, 4, 15, 16, 39, 41–43, 45, 46, 57, 59, 61, 62

LC Lane Change. 26–29, 31

LKA Lane Keeping Assist. 1–3, 36, 49, 50, 56, 57, 59–62

LSTM Long Short-Term Memory. viii, 19, 21, 23–26, 28–32, 34, 35, 37, 38, 45, 46, 49

MAML Model-Agnostic Meta-Learning. 67, 68

ML Machine Learning. viii, 3, 15–17, 45, 52, 66

MPC Model Predictive Control. viii, ix, 2, 3, 18, 19, 28, 36, 47–51, 53, 54, 62

MSE Mean Squared Error. 20, 29, 68

OA Optimal Action. 38, 39, 46, 49

PID Proportional-Integral-Derivative. 33, 37

RL Reinforcement Learning. 36, 38

RMSE Root Mean Square Error. viii, 16, 17, 19, 29–32, 35, 39, 58

RNN Recurrent Neural Network. viii, 21, 22, 24, 27, 32, 38

SD Standard Deviation. 39, 40, 56, 58, 59, 62

SRR Steering Reversal Rate. ix, 59, 61, 62

SVM Support Vector Machine. 15, 27, 28, 31

SWT Steering Wheel Torque. viii, ix, 39, 41–45, 52, 53, 57–59

TME Toyota Motor Europe. 2, 40, 52, 53

TTLC Time-To-Lane-Change. 27–31, 45

UA User Adaptation. 39, 66–68

UP User Prediction. 39, 46, 49

List of Figures

List of Tables

1

INTRODUCTION

Autonomous driving has become an extensive research field, with vehicle manufacturers investing heavily in developing this complex technology. However, during this time-consuming transition to fully autonomous driving, Advanced Driver Assistance Systems (ADAS) have provided means to use forms of partial automation in commercially available vehicles. These systems aim to increase road safety and driver comfort by minimising human errors and reducing the mental and physical workload, as they perform specific parts of the driving task or take over during emergencies [16] [17]. For example, an Autonomous Emergency Braking System will automatically make an emergency stop if it detects an obstacle in front of the vehicle or even steers away to avoid a collision [18]. Another form of ADAS is a Lane Keeping Assist (LKA), where this system performs the steering task [19].

Most of the current LKAs are in vehicles rated at SAE Level 2, which means that the human driver should always be ready to be the fallback entity [20]. Thus, if the LKA is not able to operate well in a specific scenario or makes a mistake, the driver should be ready to intervene and take over full control. However, due to the driver having an inactive role in the steering task where he/she mainly monitors the system, the risk of an unsafe transition of control increases with the reduced situational awareness [21]. Hence, the driver still has to stay involved to make sure that this transition of authority goes safely [22].

To increase the active involvement of the driver with the steering task when the LKA is active, the use of haptic shared control between the driver and the assist has been proposed [23] [24], where the control authority is shared between both. This enables the driver to maintain situational awareness and be ready to take over full control while the assist still performs a substantial part of the steering effort, enforcing the benefits of ADAS.

However, haptic shared control also results in the driver and automation having to cooperate more intensively. This imposes the risk of a mismatch in intentions between both, resulting in a conflict [25] [26]. For example, LKAs are conventionally developed to strictly follow the centre of the lane while a driver demonstrates 'satisficing' behaviour

[27], such as cutting the turns or driving more on the sides of the lane.

This mismatch in intentions could result in the driver not accepting the assist and turning it off completely. This eliminates the safety and efficiency that the assist brings to the roads. To prevent similar mismatches between a robot and its user, the Human-Robot Interaction field has proposed to incorporate human behaviour modelling in the planning of a robot's actions [11] [28] [29] [30]. This tackles the human-machine interaction problem where "the automation does not understand the human" [24].

Accordingly, driver behaviour modelling has been proposed for a symbiotic LKA by [31]. Their work focused on the parametric representation of the human body parts involved during the steering task in the Model Predictive Control (MPC) framework, specifically modelling the neuromuscular system, sensory organs and cognitive behaviour of the driver. This resulted in an assist which improved collaboration between the driver and assist. However, they discovered the driver's non-linear steering behaviour to be challenging to model with a parametric approach, as was also concluded by [32].

Therefore, a torque-based data-driven approach has been proposed to capture the non-linear behaviour from data [12]. Previous work already paired a Hidden Markov Model (HMM) with MPC on a steering wheel angle control basis and not the torque, resulting in an uncompliant assist [33]. Thus, the study by [12] focused on a HMM to predict the steering wheel torque inputs of the driver, which showed promising performance with accurate predictions. However, the main research gap in their work is the absence of a controller-in-the-loop implementation. Additionally, an Artificial Neural Network (ANN) has also proven to be a promising option for driver modelling over a HMM [1].

All in all, this thesis focuses on a human-centric haptic shared LKA, pairing an ANN driver model with a MPC controller to foster the collaboration between driver and assist, tackling the conflicts between both. Specifically, the proposed hybrid controller works with steering wheel torque control instead of the conventional steering wheel angle control, promoting a compliant assist. Finally, the hybrid controller's subjective and objective performance is evaluated on Toyota Motor Europe's (TME) advanced driving simulator against a commercially available LKA by performing driver-in-the-loop experiments.

1.1. RESEARCH OBJECTIVES

To address the research gap of a human-centric torque-based hybrid controller, this thesis presents an ANN driver model coupled to the MPC framework. The ANN captures the non-linear driver behaviour while the MPC controller gives an optimised control input, where it balances between accurately following the centre of the lane and adhering to the driver model's prediction. Thus, the main goal of this thesis is stated as:

The design and assessment of a novel torque-based human-centric haptic shared LKA using an ANN driver behaviour model and MPC to foster collaborative driving.

The main goal is broken down into the following research objectives.

1. Investigate ANN-based driver models to capture the driver's non-linear steering

behaviour, identifying trends of promising types of ANNs.

2. Design and validate the proposed ANN driver model on a highway-driving dataset collected from a fixed-base driving simulator. The performance is evaluated on the set Key Performance Indicators (KPIs).

3. Design a MPC framework which accommodates the predictions from the ANN driver model and models the steering-vehicle dynamics.

4. Tune the weights of the hybrid controller to obtain a collaborative haptic shared LKA without deteriorating path tracking performance, while ensuring driver-in-the-loop stability.

5. Design an experiment for a fixed-base driving simulator with the driver-in-the-loop to assess the subjective and objective performance of the proposed hybrid controller against a state-of-the-art commercial solution, additionally investigating different levels of control authority.

6. Compare the subjective and objective performance of the proposed hybrid controller to the state-of-the-art commercial solution with the set list of KPIs.

1.2. THESIS STRUCTURE

This thesis is structured as follows. Chapter 2 presents the IEEE journal paper containing the main approach and findings of this thesis. Subsequently, Appendix A gives an overview of Machine Learning (ML) studies for driver modelling. Additionally, it gives supplementary details about the proposed ANN model, such as the KPIs used to evaluate the ANN model and a brief feature analysis. Appendix B covers additional information regarding the hybrid controller, describing the solver settings and the software architecture used. Furthermore, supplementary results are given in Appendix C along with the list of KPIs. Lastly, the topic of user adaptation for future work is explained in more detail in Appendix D.

2

JOURNAL PAPER

This chapter presents the journal paper, written in the IEEE format, describing the main approach and findings. Appendix A.3.1 defines the KPIs used to assess the ANN driver model. Furthermore, Appendix C.2.1 defines the list of KPIs used to obtain the objective results.

Hybrid Human-Centric Haptic Shared Control using Artificial Neural Network and Model Predictive Control

***Abstract*—Commercially available Lane Keeping Assist systems fail to consider the driver's intentions since they mainly focus on minimising path tracking errors, resulting in conflicts between humans and automation. This often leads to users being unsatisfactory and turning off the assist, as a result diminishing the advantages such as reduced workload and increased road safety. Considering a driver model in the assist helps increase user acceptance. Therefore, we propose a torque-based hybrid controller for a human-centric haptic shared Lane Keeping Assist, pairing a data-driven driver model with a model-based controller to foster the collaboration between the driver and assist. First, the driver's non-linear steering wheel torque behaviour is modelled and predicted using a Bidirectional Long Short-Term Memory network with an accuracy $\geq$72.4% and a smoothness $\geq$0.85Nm/s over a 0.4s prediction horizon. Second, a Model Predictive Controller with a linear bicycle and steering model is developed, where it utilises the driver model's predictions as a time-varying reference. We developed three human-centric controllers for comparison and used a state-of-the-art commercial solution as the baseline controller. The experiments were performed in Toyota Motor Europe's fixed-base driving simulator, where 15 participants tested and evaluated the four controllers. The results show a 113.1% increase in collaborative ratio while maintaining a similar path tracking performance compared to the baseline.**

***Index Terms*—Haptic shared control, data-driven driver modelling, Artificial Neural Network, Model Predictive Control, human-machine interaction.**

I. Introduction

ADVANCED Driver Assistance Systems (ADAS) have become a widespread solution during the transition to fully automated driving. These systems aim to increase road safety and driver comfort by using partial automation to take over certain driving tasks, attempting to minimise human errors and reduce the workload of the task [1] [2]. For example, a Lane Keeping Assist (LKA) performs the steering task to follow the road centreline. Most commercially available LKA systems are found in vehicles which are rated at SAE level 2 or lower, making the driver the fallback entity when the assist fails to work.

However, due to the driver having an inactive role in the steering task where he/she mainly monitors the system, the risk of an unsafe transition of control increases. This is caused by the risk of the driver having reduced situational awareness and having to suddenly go from mental underload (e.g. boredom) to mental overload during the transition [3]. For example, it has been shown that the quality of a forced take-over decreases when the driver is distracted [4].

Thus, to increase driver involvement, but still maintain the safety and comfort aspect of ADAS, haptic shared control has been proposed [5] [6]. This ensures the active participation of the driver in the steering task, while the haptic shared control inputs of such LKA would decrease the mental and physical workload. However, this also means that the automation and the driver constantly interact. This could result in a conflict between them, as the intentions of both might not align [7] [8]. For example, the driver might prefer to cut sharp turns or drive on the sides of the lane, showing 'satisficing' behaviour [9], while a conventional LKA tends to follow the lane centre.

Consequently, the misalignment between driver and automation imposes the risk of ADAS being turned off, taking away the safety and comfort features. To prevent similar mismatches between a robot and its user, the Human-Robot Interaction field has proposed to incorporate human behaviour modelling in the planning of a robot's actions [10] [11] [12]. This tackles the human-machine interaction problem where "the automation does not understand the human" [6].

Therefore, driver behaviour modelling has been proposed for a symbiotic LKA by [13]. They focused on a parametric representation of the driver's neuromuscular system, sensory organs and cognitive behaviour in the Model Predictive Control (MPC) framework, along with a model of the vehicle dynamics. However, they discovered that a human portrays non-linear cognitive behaviour during the steering task, as was also concluded by [14], making it challenging to capture the driver's behaviour with a parametric approach.

Hence, this study proposes a data-driven approach for driver modelling, specifically using an Artificial Neural Network (ANN), based on the related work covered in Section II. This driver model is then used in the proposed MPC framework to obtain a hybrid controller for a human-centric LKA for haptic shared control.

II. Related work

Previous work related to driver modelling using a data-driven approach has focused on different aspects of the driving task. Lane change behaviour modelling has been an extensive research field for the implementation of different types of Machine Learning (ML) algorithms. For example, Long Short-Term Memory (LSTM) [15] networks have proven to be an accurate solution to predict the time left for a lane change to be performed by the driver [16] [17]. Predictions like these allow ADAS to anticipate future events and plan their steps accordingly.

The research on driver modelling using ML for the lane keeping task has been less extensive, especially using ANNs. However, to show the effect of a driver model for the lane keeping task, a comparison between a Feedforward ANN

(FANN) model, trained on driving data, and a MPC controller, containing a bicycle model and no driver modelling, is proposed in [18]. Their results showed that the steering wheel angle inputs of the FANN model gave a Root Mean Square Error (RMSE), with respect to the driver input, which was almost 10 times lower than the MPC controller's RMSE. This demonstrates the gap between the driver and the MPC controller when the driver is not accounted for.

Additionally, a comparison of different ML methods for driver steering modelling is proposed by [19]. It covered, among others, a Gaussian Process, Gaussian Mixture Regression (GMR), Hidden Markov Model (HMM) GMR and an ANN model, where the models predict the steering wheel torque (SWT) inputs of the driver. The ANN model performed the best, as it was the most accurate while also being the most robust to inter-driver variability. Notably, a relatively simple ANN model was used, namely a fully connected FANN with one hidden layer containing two nodes, where more performance could potentially be gained.

Hence, a FANN with four hidden layers is proposed by [20] to model the steering wheel angle inputs of expert drivers. The model was used to provide haptic guidance to novice drivers. Furthermore, a FANN model has been developed by [21] to predict the SWT inputs, but using two hidden layers with 32 nodes each. Sliding window features were used to feed the past input data to the model to make use of the temporal relations of human behaviour.

However, a Bidirectional LSTM (BiLSTM) [22] model is proposed by [23] due to its better performance on sequential data, where it predicts the SWT inputs of the driver using Electromyography signals. The model contained two BiLSTM layers, followed by a fully connected feedforward layer. The past 200ms of input features were fed to the model, which predicted the SWT for the upcoming 200ms. The BiLSTM model outperformed, among others, a LSTM model with the same architecture.

Another study has focused on capturing the motor intermittency of human drivers using a Deep Convolutional Fuzzy System model to predict the steering wheel angle inputs of the driver [24]. Furthermore, Deep Reinforcement Learning has also been used to predict the steering wheel angle inputs in an effort to create comfort-oriented haptic guidance [25].

However, the discussed ANN driver models lack the implementation of a controller-in-the-loop. Hence, a driver model using a HMM paired with MPC is proposed by [26], where the predictions from the driver model were used in the cost function of the MPC controller. However, the control input to the plant and the prediction of the HMM is the steering wheel angle instead of the SWT. This imposes the risk of the steering wheel becoming less compliant with human intervention.

Thus, a HMM has been implemented to make offline predictions of the driver's SWT inputs by collecting driving data in an advanced driving simulator [27]. Their model showed high accuracy and smoothness in its predictions, using a detailed feature analysis. This study uses the dataset collected by [27] to train the proposed ANN models, which are benchmarked against the HMM. However, an important research gap from their work is also the lack of a controller-in-the-loop implementation to develop and test a human-centric LKA.

Summarising, the main contribution of this study is the development of a driver behaviour model using a BiLSTM network combined with the MPC framework to create a human-centric torque-based hybrid controller for a haptic shared LKA to foster collaborative driving between the driver and assist. Moreover, the proposed controllers are tuned and evaluated extensively on an advanced driving simulator to obtain subjective and objective results representing the performance gain over a commercially available LKA.

Finally, the structure of this paper is as follows. Section III describes the BiLSTM network developed to model the driver. This is followed by the hybrid controller implementation in Section IV, where the MPC controller is outlined. The experimental setup is specified in Section V, followed by the results obtained from the experiments and the discussion in Section VI. Lastly, the main conclusions and future directions of this study are given in Section VII.

III. Artificial Neural Network driver model

The proposed hybrid controller uses an ANN model, specifically a BiLSTM network, to predict the SWT inputs of the driver. The purpose is to use the predictions to anticipate the driver's behaviour and make the controller act accordingly to foster the collaboration between driver and assist, creating a symbiosis. Thus, the predictions of the ANN model are used in the MPC controller, which will be discussed in more detail in Section IV. This section describes the data used to train the driver model, specifies the network's architecture and gives an assessment of its performance.

A. Dataset

The dataset to train, validate and test the BiLSTM model is obtained from [27]. In this dataset, seven participants drove a total of 14 hours in an advanced driving simulator, which has been used in this study as well. The task of the participants was to drive on the middle lane of a three-lane highway, where they only performed the steering task, as the vehicle was set to a constant speed of 100km/h. The 2 hours of driving for each participant consisted of 24 different road scenarios with different combinations of corners, where the data was collected at a rate of 100Hz. The data of each participant was split into 50% for training, 25% for validation, and 25% for testing, in accordance with [27].

Furthermore, the 18 candidate features described by [27] were also considered for the BiLSTM model. The noisy features were pre-processed using a zero-phase low-pass filter.

B. Architecture

The BiLSTM network has been developed using a thorough objective and subjective evaluation, where the hyperparameters are tuned through trial-and-error. First, the objective evaluation was done by training the BiLSTM model on the training dataset and assessing its performance on the validation dataset on two Key Performance Indicators (KPIs). These KPIs are the

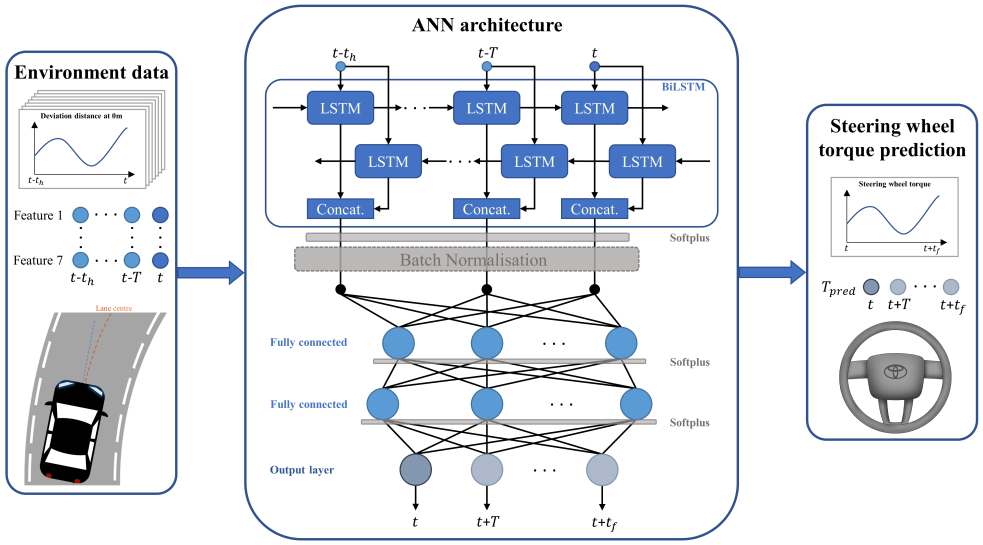

Fig. 1. Overview of the proposed BiLSTM model.

accuracy and smoothness of the SWT prediction, where the smoothness is defined as the Standard Deviation (SD) of the torque rate [27]. To accelerate the training time of each model iteration, four NVIDIA A10G Graphics Processing Units were used.

Second, the subjective evaluation was done by implementing the BiLSTM model in the driving simulator and letting expert drivers test the steering wheel feeling of different model iterations. To do this, the real-time SWT prediction was applied directly as an assist torque without using a controller, as it gives a better impression of the raw performance of the BiLSTM model. The model's hyperparameters were then further tuned based on the feedback of the expert drivers.

Figure 1 gives an overview of the final architecture obtained from the tuning process, where T_{pred} denotes the SWT prediction made by the BiLSTM model. The network starts with a BiLSTM layer with 20 features each in the hidden state for both the forward and backward pass LSTM layer. This is then followed by a normalisation layer, passing onto a fully connected linear layer with 20 nodes. Subsequently, a second fully connected linear layer with 25 nodes is used, followed by an output layer with 5 outputs. Table I lists the hyperparameters obtained during the tuning phase. The model is developed in Python [28] using the PyTorch library [29].

The input features found to perform the best out of the candidate features [27], after an extensive feature analysis, were the environment data captured from the virtual onboard sensors. The exact features are shown in Table II, where the distance mentioned is the look-ahead distance from the centre of the front axle to the centre of the lane. Similarly, the deviation of the vehicle is also measured with the same references. Furthermore, the input features' historical data of the past 0.5s, the historical window t_h, is also passed on to the network with steps of T=0.1s. The network gives the SWT predictions as an output, where it predicts the torque for the current timestep and the following 0.4s, the prediction horizon t_f, with a period of 0.1s, resulting in five output values.

TABLE I
HYPERPARAMETERS SET FOR THE BILSTM MODEL.

Hyperparameter	Setting
Loss function	MSE
Optimiser	Adam
Learning rate	$2.0 \cdot 10^{-5}$
Number of epochs	6
Batch size	500
Validation split	0.25
Historical window t_h	0.5s
Prediction horizon t_f	0.4s

TABLE II
INPUT FEATURES TO THE BILSTM MODEL.

Feature	Unit
Deviation Distance at 0m	m
Deviation Angle at 0m	rad
Road Curvature at 0m	1/m
Deviation Angle at 10m	rad
Road Curvature at 10m	1/m
Deviation Angle at 30m	rad
Road Curvature at 30m	1/m

C. Model assessment

The final BiLSTM model design is trained on both the training and validation dataset. The performance of the model, in terms of the KPIs, is assessed using the test dataset. Figure 2 shows the accuracy and smoothness scores of the SWT predictions over the prediction horizon. The average accuracy

and smoothness are respectively equal to 73.4% and 0.91Nm/s, where specifically the predictions for the current time step (t=0) result in 72.4% and 0.85Nm/s.

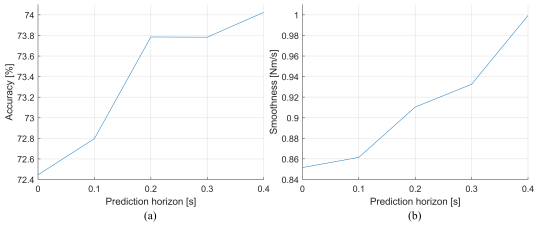

Fig. 2. Accuracy (a) and smoothness (b) of the BiLSTM model over the prediction horizon.

There was also a model trained during the development phase which obtained 96.3% accuracy and 2.55Nm/s smoothness for t=0. This model uses all 18 candidate features and 170 epochs, while the architecture and the other hyperparameters are the same as the final model. Thus, these scores show that the BiLSTM model is capable of outperforming the previous work done with a HMM [27].

However, a significant finding during the subjective evaluation of the BiLSTM model on the driving simulator is that the accuracy of the prediction is less relevant compared to the smoothness. Having a very accurate model of the driver also means taking over the imperfections the driver has. For example, humans make many small corrections during the steering task. If the BiLSTM model replicates this and applies it as haptic feedback, the driver will experience many small movements of the steering wheel, making the assist feel jerky. Thus, it is important to keep the smoothness KPI at a low level, inherently resulting in lower accuracy. Figure 3 illustrates this by giving a fragment of the predictions made by a very smooth and accurate model.

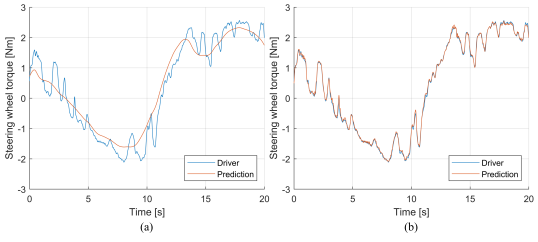

Fig. 3. Smooth (a) and accurate (b) prediction of the driver's SWT inputs.

IV. Hybrid Controller

The hybrid controller seeks to apply an optimised control input to the vehicle's steering wheel using MPC, where it attempts to balance out the prediction of the BiLSTM model to the accurate following of the lane centre based on the tuned weights. This improves the robustness and safety of the hybrid controller as the MPC controller is capable of compensating for false predictions made by the BiLSTM model and is able to function in a broader range of scenarios since it does not rely on data training.

This section covers the design of the hybrid controller. First, the pairing of the driver model with the MPC framework is described. Second, the vehicle dynamics are defined and, finally, the MPC design is specified.

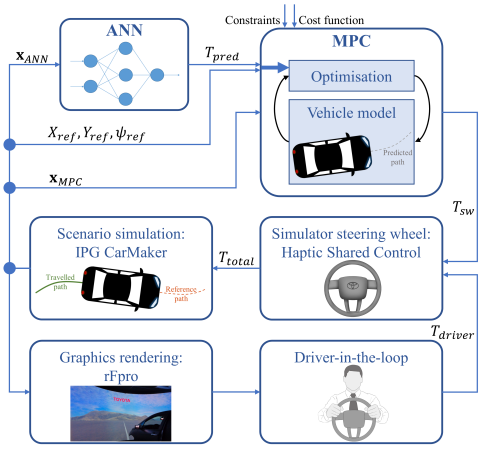

Fig. 4. Overview of the hybrid controller implementation in the advanced driving simulator.

A. Hybrid controller structure

Figure 4 shows how the BiLSTM model and MPC controller cooperate, where $\mathbf{x}_{ANN}$ denotes the input features to the ANN model as mentioned in Table II, T_{driver} stands for the SWT applied by the driver and T_{total} denotes the total SWT applied to the steering system. With steps of 0.1s, the BiLSTM model predicts 0.4s into the future, which is the same length as the prediction horizon of the MPC controller. The prediction is passed onto the MPC controller as a time-varying SWT reference signal, which can then be used during the optimisation steps. The MPC solver is developed in MATLAB using the FORCESPRO [30] software. The generated solver is then used in Simulink.

B. Vehicle model

The vehicle dynamics are represented using a linear bicycle model with linear steering dynamics, as adapted from [13] and [31]. The bicycle and steering models are shown in Figure 5. The vehicle model assumes a constant longitudinal velocity v_x, set at 100km/h, and linear tyre dynamics. The steering model works with the SWT instead of the steering wheel angle. Thus, the control input u of the MPC controller is the SWT rate $\dot{T}_{sw}$, which is integrated and applied as the assist torque of the LKA. The dynamics of the vehicle are described as follows.

$$\dot{X} = v_x \sin(\psi) + v_y \cos(\psi) \tag{1a}$$

$$\dot{Y} = v_x \cos(\psi) - v_y \sin(\psi) \tag{1b}$$

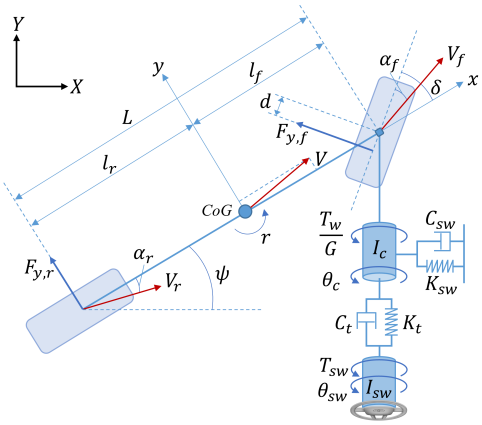

Fig. 5. Bicycle and steering model.

$$\dot{v}_y = \frac{F_{y,f}\cos(\delta) + F_{y,r}}{m - v_x r} \tag{1c}$$

$$\dot{r} = \frac{l_f F_{y,f}\cos(\delta) - l_r F_{y,r}}{I_{zz}} \tag{1d}$$

$$\ddot{\theta}_{sw} = \frac{T_{sw}}{I_{sw}} - \frac{K_t}{I_{sw}}\theta_{sw} - \frac{C_t}{I_{sw}}\dot{\theta}_{sw} + \frac{K_t}{I_{sw}}\theta_c + \frac{C_t}{I_{sw}}\dot{\theta}_c \tag{1e}$$

$$\begin{aligned}\ddot{\theta}_c = &-\frac{T_w}{GI_c} + \frac{K_t}{I_c}\theta_{sw} + \frac{C_t}{I_c}\dot{\theta}_{sw} - \frac{K_t + K_{sw}}{I_c}\theta_c \\ &- \frac{C_t + C_{sw}}{I_c}\dot{\theta}_c\end{aligned} \tag{1f}$$

where X and Y are the global coordinates of the vehicle, ψ is the yaw angle, v_y is the lateral velocity, m is the vehicle mass and r is the yaw rate. Furthermore, l_f and l_r denote the distance between the centre of gravity and the front and rear axle, respectively. I_{zz} is the yaw inertia of the vehicle at the centre of mass, I_{sw} is the steering wheel inertia and I_c is the inertia of the front wheels and the rack with respect to the pinion. θ_{sw} denotes the steering wheel angle and θ_c the steering column angle. K_t and K_{sw} are the steering column and self-centring stiffness, respectively, C_t is the torsion bar damping and C_{sw} is the steering mechanism damping with respect to the steering wheel axle. G denotes the steering gear ratio. Thus, the states vector is

$$\mathbf{x}_{MPC} = \begin{bmatrix} X & Y & v_y & \psi & r & \theta_{sw} & \dot{\theta}_{sw} & \theta_c & \dot{\theta}_c & T_{sw} \end{bmatrix}^T. \tag{2}$$

Furthermore, the road wheel angle δ is determined using the equation

$$\delta = \frac{\theta_c}{G}. \tag{3}$$

The lateral forces on the front axle $F_{y,f}$ and rear axle $F_{y,r}$ are described as

$$F_{y,f} = -C_{\alpha,f}\tan(\alpha_f) \tag{4}$$

$$F_{y,r} = -C_{\alpha,r}\tan(\alpha_r) \tag{5}$$

where $C_{\alpha,f}$ and $C_{\alpha,r}$ are the cornering stiffness of the front and rear tyres, respectively, and α_f and α_r are the slip angles at the front and rear axle, respectively, which are determined with

$$\alpha_f = -\delta + \arctan\left(\frac{v_y + l_f r}{v_x}\right) \tag{6}$$

$$\alpha_r = \arctan\left(\frac{v_y - l_r r}{v_x}\right). \tag{7}$$

Finally, the self-aligning moment T_w, due to the torque on the king-pin axis, is approximated as

$$T_w = F_{y,f} d \tag{8}$$

where d denotes the trail distance of the front tyre.

C. Cost function and MPC settings

The cost function brings the BiLSTM driver model and the MPC controller together. As mentioned before, the SWT predictions of the BiLSTM model T_{pred} are being used as one of the reference signals to the MPC. The cost function is described as

$$\begin{aligned} J = &\sum_{k=0}^{N_p-1} w_X(X_k - X_{ref,k})^2 + w_Y(Y_k - Y_{ref,k})^2 \\ &+ w_\psi(\psi_k - \psi_{ref,k})^2 + w_T(T_{sw,k} - T_{pred,k})^2 \\ &+ w_u u_k^2 \\ &+ w_{X_N}(X_{N_p} - X_{ref,N_p})^2 + w_{Y_N}(Y_{N_p} - Y_{ref,N_p})^2 \\ &+ w_{\psi_N}(\psi_{N_p} - \psi_{ref,N_p})^2 \\ &+ w_{T_N}(T_{sw,N_p} - T_{pred,N_p})^2 \end{aligned} \tag{9}$$

where w_X and w_{X_N} are the stage and terminal cost weights for the error in the global X-direction, w_Y and w_{Y_N} are the weights for the Y-direction, and w_ψ and w_{ψ_N} are the weights for the yaw angle error. w_T and w_{T_N} denote the stage and terminal cost weights for the difference between the applied SWT T_{sw} and the predicted SWT T_{pred} by the BiLSTM model. Thus, these two weights determine the level of the penalty when the prediction of the BiLSTM model does not match the actual SWT being applied. Finally, w_u denotes the weight for the control input $u = \dot{T}_{sw}$.

The reference signals Y_{ref}, X_{ref} and ψ_{ref} represent the desired path, which is set as the centre of the lane, while N_p denotes the prediction horizon. Table III shows the settings and weights used for the MPC controller, where $T_{s,c}$ is the sampling time of the MPC controller, set at 100Hz to match the standard CAN broadcast frequency on driving simulators, while $T_{s,sim}$ is the sampling time of the simulation, set at 1000Hz. The prediction horizon N_p and control horizon are both set at 40 timesteps (0.4s), as it was found to be the best trade-off between the computational load and controller performance.

The weights are determined through an extensive trial-and-error process. First, the weights were tuned to ensure closed-loop stability by running local simulations. Second, this set of weights formed a basis for the driving simulator implementation, where the weights were further tuned with Toyota Motor Europe's (TME) previous-generation driving simulator to ensure driver-in-the-loop stability. Lastly, the final set of weights was determined on the advanced driving simulator at TME, described in Section V-A, with the feedback of expert drivers to obtain collaborative haptic shared control, while considering the overshoot and settling time of the system.

TABLE III
MPC SETTINGS AND WEIGHTS.

Variable	Value	Variable	Value
$T_{s,c}$	$1 \cdot 10^{-2}$ s	$T_{s,sim}$	$1 \cdot 10^{-3}$ s
N_p	40	w_X	$1 \cdot 10^{-4}$
w_Y	$1 \cdot 10^{-4}$	w_ψ	$1 \cdot 10^{-1}$
w_T	$5 \cdot 10^{-4}$	w_u	$1.2 \cdot 10^{-6}$
w_{i_N}	$2 \cdot w_i$	i	$\{X, Y, \psi, T\}$
$\lvert\theta_{sw,max}\rvert$	360 deg	$\lvert\dot{\theta}_{sw,max}\rvert$	800 deg/s
$\lvert T_{sw,max}\rvert$	10 Nm	$\lvert\dot{T}_{sw,max}\rvert$	20 Nm/s
$\lvert r_{max}\rvert$	50 deg/s		

V. Driving Simulator Experiment

To evaluate the performance of the proposed LKA, three variations of the hybrid controller have been implemented in an advanced fixed-base driving simulator at TME. Additionally, a commercially available LKA is used as a benchmark. This section describes how the experiments are designed and performed, and specifies the four controllers.

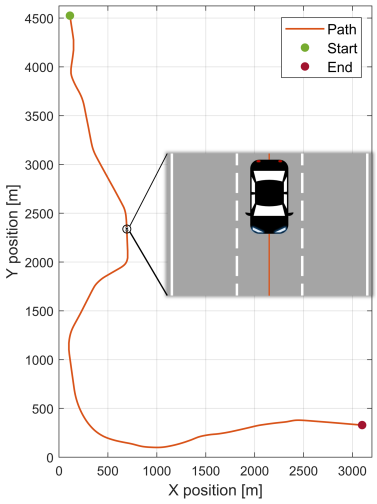

Fig. 6. Path used for the experiment on a three-lane highway.

A. Driving scenario

Figure 6 shows the path used during the experiments. This route contains a three-lane highway where a distance of 8.3km is covered at a constant speed of 100km/h, taking approximately 5 minutes. This path contains a variety of turns with different curvatures to expose the driver to various circumstances to experience the LKA's capabilities. The participants only controlled the steering wheel and were instructed to solely follow the second lane like they normally would. At the final straight section of the route, the participants were instructed to perform lane changes. First, a lane change to the right lane was performed, followed by a lane change back to the middle lane and eventually to the left lane. The lane changes were performed to give the participants a better impression of how much effort is needed to overrule the assist.

The driving simulator has a fixed base and uses the cockpit of a commercially available Toyota car and a 210° projection screen for better immersion, as shown in Figure 7. The dynamics of the vehicle are simulated in IPG CarMaker, except for the steering system which is simulated using Toyota's high-fidelity steering model [32]. IPG CarMaker is also used to define the scenario, where rFpro renders the graphics for the projection screen.

Fig. 7. Fixed base driving simulator at Toyota Motor Europe, Belgium.

B. Experimental procedure

The participants were first informed of their task during the experiment. Subsequently, they drove the scenario without the assists on, as to familiarise themselves with the driving simulator, the steering feel and the scenario. This was followed by running the same driving scenario but with one of the four LKAs. The scenario was then repeated for the other LKAs. The order of the assists was randomised (Randomised Latin Square Method) and the participants were not informed on which LKA they were testing, as to eliminate any bias [33]. The participants evaluated each LKA immediately after the scenario by filling in a questionnaire but were allowed to change their answers after testing the other LKAs.

A total of 15 participants, all TME employees, took part in the experiment with an average age of 30.5 years (SD=5.4), ranging from the age between 24 and 41 years old, and an average of 10.8 years (SD=6.7) of driving experience. Additionally, two of the participants are rated as expert drivers within TME.

C. LKA controllers

This section describes the four LKA controllers evaluated during the experiment.

1) Baseline LKA: A commercially available LKA is used as the baseline to compare with the proposed LKAs. This baseline controller aims to accurately follow the centre of the lane without considering the driver's intentions. It uses an adaptive level of control authority with the level being inversely proportional to the opposing SWT input from the driver. The LKA even gets momentarily deactivated if the driver opposes the assist with a SWT of 3Nm or higher.

2) ANN: The first proposed controller only uses the BiLSTM driver model. The prediction of the model at t=0 of the prediction horizon is passed on as the actual assist torque, instead of going to the MPC controller. The SWT inputs from this assist are set to have a control authority of 70%, leaving the remaining 30% to the driver to perform the lane keeping task. The ANN assist is implemented to test how well the BiLSTM model performs on its own without the use of a supplementary controller.

3) Hybrid-1: The other two proposed controllers are hybrid controllers, where both use the same set of weights, but their control inputs are adapted to two different levels of authority. The first mode, Hybrid-1, has been set to the same level of control authority as the ANN controller, namely 70%, allowing for a direct comparison between both controllers.

4) Hybrid-2: The second mode, Hybrid-2, is set to a control authority of 50%. This mode aims to give the driver more control to test its effect on collaborative driving. Thus, the two hybrid controller modes will give a better impression of the participants' preferred authority level for haptic shared control.

VI. Results and Discussion

This section discusses the obtained subjective and objective results. The KPIs for both results focus on five main elements, specifically the steering effort, tracking performance, collaborative behaviour, level of control authority and smoothness. The statistical significance of the four controllers is analysed for each KPI using a paired t-test.

A. Subjective results

As the participants had to fill in the questionnaire for each assist, they answered five questions per assist by awarding a point based on a seven-point scale. The ideal range for the first question is between 3 and 5 points, while the other four questions had an ideal range between 5 and 7 points. The questions focus on the following aspects:

1) Steering effort: How soft or heavy the steering system feels with the assist on. A grade of 1 means that it is too soft, while a grade of 7 signifies that it is too heavy.
2) Path tracking performance: The perceived path tracking performance and guidance level, where a grade of 1 means low and a 7 means high performance.
3) Collaborative behaviour: How well the driver's intentions are matched by the assist, thus whether there are no conflicts. A grade of 1 means bad and a 7 means good collaboration.
4) Authority level: How easily the assist can be overruled by the driver if desired. A grade of 1 means difficult and a 7 means easy to overrule.
5) Smooth control: How smooth the assist feels to the hand during haptic shared control, where a grade of 1 signifies abrupt and a 7 means smooth control.

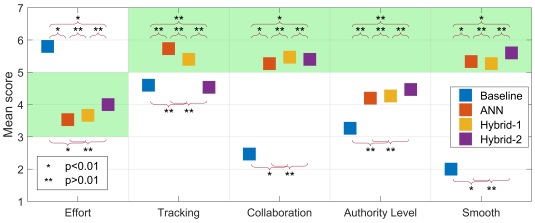

Fig. 8. Mean scores for the subjective results with the braces representing the p-values for the paired t-test.

Figure 8 shows the mean scores of the participants' answers for each question and assist, where the braces indicate the results of the paired t-test. The paired t-test demonstrates the statistical significance between the baseline LKA and the proposed controllers in terms of the steering effort, collaborative behaviour and smooth control. Table IV gives the exact values of the mean scores and shows the SD in parentheses. The proposed LKAs outperform the baseline in every metric, except for the Hybrid-2 controller in terms of tracking performance. This can be explained by the Hybrid-2 controller's lower level of control authority, making it feel like a worse path follower if the driver does not cooperate in the steering task. Besides, the baseline LKA tries to precisely follow the centre of the lane by also making constant minor corrections, giving the impression of accurate path following, even if it does not match the driver's desired path.

TABLE IV
Mean subjective results for the KPIs with the SD in parentheses.

Metric	Baseline	ANN	Hybrid-1	Hybrid-2
Steering effort	5.80 (1.26)	3.53 (1.19)	3.67 (1.29)	4.00 (1.36)
Tracking performance	4.60 (1.92)	5.73 (1.03)	5.40 (1.24)	4.53 (1.73)
Collaborative behaviour	2.47 (1.36)	5.27 (1.10)	5.47 (1.41)	5.40 (0.91)
Authority level	3.27 (1.83)	4.20 (1.57)	4.27 (1.33)	4.47 (1.77)
Smooth control	2.00 (1.13)	5.33 (1.23)	5.27 (1.44)	5.60 (1.30)

Furthermore, the proposed controllers score best for the KPIs interchangeably. Hybrid-2 was found to have the best steering feel, which can be explained by the lighter assist making the steering wheel feel slightly heavier to the driver. It also scored best regarding the authority level and smoothness of the assist. Again, the relative lightness of the assist contributes to giving more control to the driver and feeling less aggressive

to the hand. The ANN controller was found to have the best path tracking performance and the Hybrid-1 controller was evaluated to have the best collaborative behaviour by a small margin over the other proposed controllers.

Finally, the participants were asked to rank the assists at the end of the experiment based on which one they preferred the most. Seven out of the 15 participants ranked the Hybrid-2 controller in first place, while four participants ranked the Hybrid-1 assist and another four participants ranked the ANN assist as their favourite.

TABLE V
MEAN OBJECTIVE RESULTS FOR THE KPIS WITH THE SD IN PARENTHESES.

Metric	Baseline	ANN	Hybrid-1	Hybrid-2
Driver effort [Nm^2 s]	374.71 (95.88)	128.88 (17.22)	122.68 (13.95)	240.65 (20.42)
Controller effort [Nm^2 s]	-	363.15 (21.05)	368.71 (14.94)	193.28 (7.19)
Lateral RMSE [m]	0.33 (0.09)	0.38 (0.11)	0.34 (0.10)	0.33 (0.08)
Maximum e_y [m]	0.88 (0.27)	1.00 (0.24)	0.87 (0.21)	0.88 (0.19)
Mean e_y [m]	0.12 (0.10)	0.21 (0.13)	0.15 (0.14)	0.17 (0.12)
SD e_y [m]	0.29 (0.07)	0.30 (0.07)	0.28 (0.08)	0.27 (0.05)
Collaborative ratio [-]	0.38 (0.03)	0.76 (0.03)	0.77 (0.03)	0.81 (0.03)
Intrusiveness ratio [-]	0.62 (0.03)	0.24 (0.03)	0.23 (0.03)	0.19 (0.03)
Resistance ratio [-]	-	0.14 (0.02)	0.12 (0.02)	0.13 (0.02)
Contradiction ratio [-]	-	0.10 (0.02)	0.11 (0.02)	0.06 (0.01)
Coherence [-]	-	0.69 (0.04)	0.70 (0.05)	0.84 (0.02)
Level of control authority [-]	-	2.88 (0.52)	3.05 (0.44)	0.81 (0.10)
Steering reversal rate [reversal/min]	17.56 (3.66)	13.33 (2.76)	13.69 (2.47)	14.01 (3.02)
Driver smoothness [Nm/s]	3.16 (0.48)	1.98 (0.37)	2.00 (0.40)	2.17 (0.41)
Controller smoothness [Nm/s]	-	0.64 (0.04)	0.67 (0.05)	0.48 (0.04)

B. Objective results

Table V shows the mean objective results obtained from the driving data of the participants, with the SD given in parentheses. The first 13 KPIs are obtained from [13], with the metrics driver and controller smoothness added to the list, which is the same as the smoothness KPI used in Section III-C. Additionally, Table VII gives the results of the paired t-test for each pair of assists for every KPI. The pairs associated with the letters are shown in Table VI. For six of the metrics, a representative and comparable value cannot be obtained for the baseline LKA. This is due to a fundamentally different steering approach where the assist torque values are determined to actuate on the steering column instead of the steering wheel, resulting in assist torque values which are significantly higher. To avoid any misconceptions, those KPI values have been omitted, while still allowing for a comparison between the proposed controllers for the same KPIs.

TABLE VI
SYMBOLS FOR THE PAIRS USED FOR THE PAIRED T-TEST.

Symbol	P-value comparison
A	Baseline $\leftrightarrow$ ANN
B	Baseline $\leftrightarrow$ Hybrid-1
C	Baseline $\leftrightarrow$ Hybrid-2
D	ANN $\leftrightarrow$ Hybrid-1
E	ANN $\leftrightarrow$ Hybrid-2
F	Hybrid-1 $\leftrightarrow$ Hybrid-2

TABLE VII
PAIRED T-TEST P-VALUES FOR THE OBJECTIVE RESULTS.

Metric	A	B	C	D	E	F
Driver effort	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
Controller eff.	-	-	-	0.07	<0.01	<0.01
Lateral RMSE	<0.01	0.36	0.78	0.01	0.01	0.45
Maximum e_y	0.02	0.86	0.97	0.05	0.01	0.73
Mean e_y	<0.01	0.07	0.01	0.04	0.01	0.51
SD e_y	0.28	0.29	0.02	0.10	0.01	0.30
Collaborative	<0.01	<0.01	<0.01	0.36	<0.01	<0.01
Intrusiveness	<0.01	<0.01	<0.01	0.36	<0.01	<0.01
Resistance	-	-	-	0.02	0.69	0.05
Contradiction	-	-	-	0.09	<0.01	<0.01
Coherence	-	-	-	0.14	<0.01	<0.01
Authority	-	-	-	0.01	<0.01	<0.01
Steering RR	<0.01	<0.01	<0.01	0.57	0.33	0.57
Driver smooth.	<0.01	<0.01	<0.01	0.76	0.02	0.03
Control. smooth.	-	-	-	<0.01	<0.01	<0.01

The results show a significant drop in driver effort with the proposed controllers compared to the baseline, where the driver effort is defined as the integral of the driver's squared SWT input. The Hybrid-1 controller even shows a decrease of 67.3%. This is explained by the collaborative behaviour of the proposed controllers being significantly improved as it matches the driver's intentions, resulting in fewer conflicts and steering corrections as can be seen with the Steering Reversal Rate (SRR) with a decrease of up to 24.1%, but this is also related to the level of control authority. The control authority of the baseline LKA decreases with increasing conflicts, resulting in more effort needed from the driver. Furthermore, the driver effort of the Hybrid-2 LKA is almost twice as high as the other proposed LKAs, which is due to the lower level of control authority set. This can also be seen by the controller effort being up to 47.6% lower. Additionally, the paired t-test highlights the differences between the controllers as it shows statistical significance between each pair of LKA in terms of driver effort, with the difference being relatively small between the ANN and Hybrid-1 controllers.

Moreover, the lower level of control authority of the Hybrid-2 controller has also resulted in a 113.1% increase in the collaborative ratio over the baseline LKA and a 5.2% increase over the Hybrid-1 controller, with the collaborative ratio indicating the ratio between the time where the driver and controller applied a SWT with the same sign and the total scenario time. Thus, the driver and Hybrid-2 controller have a better agreement compared to the other LKAs, where the paired t-test shows statistical significance between the Hybrid-2 controller and the other controllers. Additionally, the Hybrid-2 LKA also outperforms the other proposed controllers significantly in terms of the contradiction ratio with a decrease

of up to 45.5% and coherence with an increase of up to 21.7%. However, leaving more control to the driver has introduced less smooth steering inputs, as a human makes more minor corrections during the steering task than the controller, with an increase in SRR of up to 5.1% and driver smoothness of up to 9.6% when comparing the Hybrid-2 controller to the other proposed controller. Nevertheless, the Hybrid-2 controller shows a decrease of 20.2% and 31.3% over the baseline LKA in terms of SRR and driver smoothness, respectively.

Furthermore, Figure 9 shows the boxplot for the lateral RMSE and the SD of the lateral error with reference to the centre of the lane. Notably, the ANN controller has relatively higher RMSE and SD values compared to the other controllers, with an increase of up to 15.2% and 11.1%, respectively. This shows the impact of the absence of a MPC controller, as the MPC element of the hybrid controllers also penalises for the lateral offset from the centre of the lane while the ANN controller does not directly account for it. The participants also noticed this behaviour from the ANN controller, where it would usually turn relatively early into a corner resulting in a higher lateral error or it would not perform minor corrections to follow the centre of the lane accurately on the straight sections. However, most of the participants did note that they found the early turn-in pleasant as it matched their driving style more, which explains the high subjective score for the path tracking performance of the ANN model.

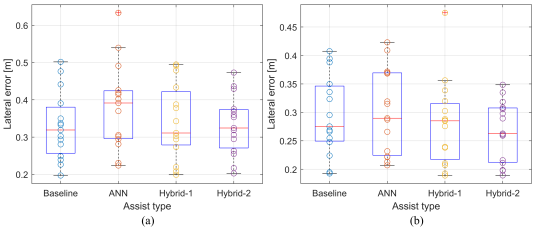

Fig. 9. Lateral RMSE (a) and SD of the lateral error (b) with respect to the centre of the lane.

Figure 10 shows the average steering inputs of the proposed controllers and the participants, with the coloured areas covering the respective SD. These graphs show the authority split between the controller and driver. While the ANN and Hybrid-1 controllers have more authority over the driver, the Hybrid-2 controller and the driver are more balanced. Additionally, it also demonstrates how the controller and driver complement each other without any significant conflicts.

Overall, the ANN and Hybrid-1 controllers have closely matched results on all the KPIs. However, the relatively poor performance of the ANN LKA on the lateral error metrics gives the Hybrid-1 LKA a slight preference between the two, as the path tracking performance is an important element of safety. Nonetheless, the lighter assist torque from the Hybrid-2 controller has resulted in better collaboration between the assist and the driver compared to the other controllers. Even though the driver has to put relatively more steering effort in, keeping the driver more engaged, the Hybrid-2 LKA still lowers the workload by splitting the effort in half, as

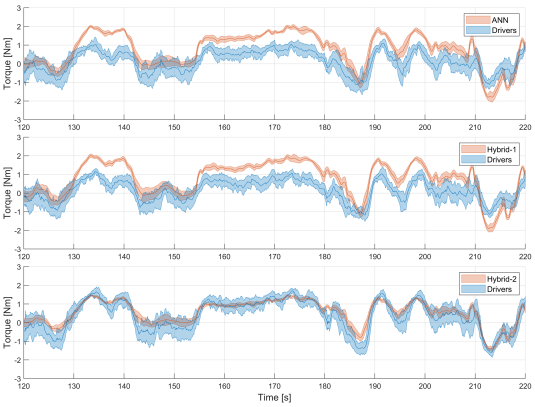

Fig. 10. Average and SD of the SWT inputs of the participants and proposed LKAs.

shown in Figure 10. Besides, the better collaboration would result in a long-term symbiosis between the LKA and the driver, preventing the assist from being turned off and, thus, maintaining the benefits of ADAS.

VII. Conclusion

In this study, we introduced a hybrid LKA using a BiLSTM driver model paired with a MPC controller to enable human-centric haptic shared control to foster collaborative driving between the driver and assist. The BiLSTM model was validated and tested to obtain the optimal driver model, as the driver portrays non-linear cognitive behaviour. Even though high accuracy ANN driver models surpassing previous work were obtained during development, it was found that the smoothness KPI was significantly more critical than the accuracy to obtain a pleasant assist.

Furthermore, the BiLSTM model makes a prediction over the prediction horizon during each iteration. This prediction is then used by the real-time MPC controller as a time-varying reference signal to create the hybrid controller. The MPC controller optimises the control input where it balances between following the centre of the lane and adhering to the predictions of the BiLSTM model, based on the tuned weights.

The subjective and objective results show the proposed controllers to significantly outperform the commercially available baseline LKA in almost every KPI. The proposed controllers decrease the driver's steering effort while substantially improving the collaboration with the assist. However, the second mode of the hybrid controller, Hybrid-2, has shown to be the most favourable assist as it fosters collaborative driving the most without lowering the path tracking performance.

Future work should focus on creating a hybrid controller capable of working at different speeds and road scenarios. This means that extensive data collection has to be done to train the ANN model. Furthermore, work should be done in creating a custom loss function when training the ANN model, where both the accuracy and smoothness KPI are accounted for. However, the challenge is to find a good balance of the

weights for such a loss function. Additionally, introducing user adaptation, such as online learning or meta-learning, to the ANN model would allow the model to converge to the preferences of a specific driver. This could result in an even better collaboration between the driver and assist.

To improve the MPC controller, the vehicle dynamics model should be more detailed to perform better at different speeds and scenarios. This can be done with a non-linear bicycle model or even using a planar model, while taking the non-linear steering friction into account in the steering model. Furthermore, other configurations to pair the ANN model with the MPC controller should be investigated and compared, especially implementing the ANN as a part of the plant model within the MPC controller. Moreover, an adaptive level of control authority on the hybrid controller should be studied to understand its effect on collaborative driving. Lastly, the hybrid controller should be tested on a physical vehicle to investigate its commercial feasibility. This would require the introduction of more robustness to the driver-in-the-loop system, such as lane departure warnings.

REFERENCES

[1] J. D. Lee, "Fifty years of driving safety research," *Human Factors*, vol. 50, no. 3, pp. 521–528, 2008.

[2] G. H. Walker, N. A. Stanton, and M. S. Young, "Where is computing driving cars?" *International Journal of Human-Computer Interaction*, vol. 13, no. 2, pp. 203–229, 2001.

[3] Z. Lu, R. Happee, C. D. Cabrall, M. Kyriakidis, and J. C. de Winter, "Human factors of transitions in automated driving: A general framework and literature survey," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 43, pp. 183–198, 2016.

[4] C. Gold, D. Damböck, L. Lorenz, and K. Bengler, ""take over!" how long does it take to get the driver back into the loop?" in *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, vol. 57, no. 1, 2013, pp. 1938–1942.

[5] F. O. Flemisch, K. Bengler, H. Bubb, H. Winner, and R. Bruder, "Towards cooperative guidance and control of highly automated vehicles: H-mode and conduct-by-wire," *Ergonomics*, vol. 57, no. 3, pp. 343–360, 2014.

[6] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: smoothly shifting control authority?" *Cognition, Technology & Work*, vol. 14, no. 1, pp. 19–28, 2012.

[7] M. Mulder, D. A. Abbink, and E. R. Boer, "Sharing control with haptics: Seamless driver support from manual to automatic control," *Human Factors*, vol. 54, no. 5, pp. 786–798, 2012.

[8] F. Mars, M. Deroo, and J.-M. Hoc, "Analysis of human-machine cooperation when driving with different degrees of haptic shared control," *IEEE Transactions on Haptics*, vol. 7, no. 3, pp. 324–333, 2014.

[9] M. A. Goodrich, W. C. Stirling, and E. R. Boer, "Satisficing revisited," *Minds and Machines*, vol. 10, no. 1, pp. 79–109, 2000.

[10] A. Kanazawa, J. Kinugawa, and K. Kosuge, "Adaptive motion planning for a collaborative robot based on prediction uncertainty to enhance human safety and work efficiency," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 817–832, 2019.

[11] Y. Cheng, L. Sun, C. Liu, and M. Tomizuka, "Towards efficient human-robot collaboration with robust plan recognition and trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2602–2609, 2020.

[12] H.-S. Moon and J. Seo, "Optimal action-based or user prediction-based haptic guidance: Can you do even better?" in *Proc. of the Conference on Human Factors in Computing Systems*, 2021, pp. 1–12.

[13] A. M. R. Lazcano, T. Niu, X. C. Akutain, D. Cole, and B. Shyrokau, "Mpc-based haptic shared steering system: a driver modeling approach for symbiotic driving," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 3, pp. 1201–1211, 2021.

[14] S. Kolekar, W. Mugge, and D. Abbink, "Modeling intradriver steering variability based on sensorimotor control theories," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 3, pp. 291–303, 2018.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] H. Q. Dang, J. Fürnkranz, A. Biedermann, and M. Hoepfl, "Time-to-lane-change prediction with deep learning," in *Int. Conference on Intelligent Transportation Systems*, 2017, pp. 1–7.

[17] F. Wirthmüller, M. Klimke, J. Schlechtriemen, J. Hipp, and M. Reichert, "Predicting the time until a vehicle changes the lane using lstm-based recurrent neural networks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2357–2364, 2021.

[18] H. Wei, W. Ross, S. Varisco, P. Krief, and S. Ferrari, "Modeling of human driver behavior via receding horizon and artificial neural network controllers," in *Conference on Decision and Control*, 2013, pp. 6778–6785.

[19] K. Okamoto and P. Tsiotras, "Data-driven human driver lateral control models for developing haptic-shared control advanced driver assist systems," *Robotics and Autonomous Systems*, vol. 114, pp. 155–171, 2019.

[20] H. Lee, H. Kim, and S. Choi, "Human driving skill modeling using neural networks for haptic assistance in realistic virtual environments," *arXiv preprint arXiv:1809.04549*, 2018.

[21] P. Krupka, P. Lukowicz, C. Kreis, and B. Boßdorf-Zimmer, "Learned steering feel by a neural network for a steer-by-wire system," in *Int. Munich Chassis Symposium*, 2020, pp. 449–464.

[22] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[23] Y. Xing, C. Lv, Y. Liu, Y. Zhao, D. Cao, and S. Kawahara, "Hybrid-learning-based driver steering intention prediction using neuromuscular dynamics," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 1750–1761, 2021.

[24] J. Chen, D. Sun, M. Zhao, Y. Li, and Z. Liu, "Dcfs-based deep learning supervisory control for modeling lane keeping of expert drivers," *Physica A: Statistical Mechanics and its Applications*, vol. 567, p. 125720, 2021.

[25] Z. Wang, Z. Yan, and K. Nakano, "Comfort-oriented haptic guidance steering via deep reinforcement learning for individualized lane keeping assist," in *Int. Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 4283–4289.

[26] S. Lefèvre, Y. Gao, D. Vasquez, H. E. Tseng, R. Bajcsy, and F. Borrelli, "Lane keeping assistance with learning-based driver model and model predictive control," in *Int. Symposium on Advanced Vehicle Control*, 2014.

[27] R. van Wijk, A. M. R. Lazcano, X. C. Akutain, and B. Shyrokau, "Data-driven steering torque behaviour modelling with hidden markov models," *IFAC-PapersOnLine*, vol. 55, no. 29, pp. 31–36, 2022.

[28] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[30] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017.

[31] T. Niu and D. Cole, "A model of driver steering control incorporating steering torque feedback and state estimation," 2020.

[32] M. Damian, B. Shyrokau, X. C. Akutain, and R. Happee, "Experimental validation of torque-based control for realistic handwheel haptics in driving simulators," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 196–209, 2021.

[33] B. Shyrokau, J. De Winter, O. Stroosma, C. Dijksterhuis, J. Loof, R. van Paassen, and R. Happee, "The effect of steering-system linearity, simulator motion, and truck driving experience on steering of an articulated tractor-semitrailer combination," *Applied ergonomics*, vol. 71, pp. 17–28, 2018.

A

MACHINE LEARNING FOR DRIVER MODELLING

This chapter will cover different ML techniques for driver modelling. First, an overview will be given of 'classical' ML methods used for driver modelling. This is followed by an extensive analysis of ANN implementations. Finally, the proposed ANN model for the hybrid controller is analysed in more detail, specifying the KPIs used to evaluate the model.

A.1. CLASSICAL MACHINE LEARNING STUDIES

This section briefly covers 'classical' ML algorithms used to model driver behaviour. 'Classical' ML mainly refers to ML techniques excluding deep learning. Additionally, a study comparing different ML, including an ANN, is discussed.

A.1.1. STATE-OF-THE-ART RESEARCH

Different types of 'classical' ML algorithms have been developed to predict the human driver's intentions for a variety of tasks. While some studies focus on a classification problem where the driver's behaviour can be approached categorically, such as a left or right turn, others work on a regression problem where the driver's behaviour is modelled in continuous-time, such as the exact steering wheel torque a driver is expected to give at a certain time.

For example, a classifier using a Support Vector Machine (SVM) and a k-nearest neighbours algorithm to determine the driving skill of the human driver while taking corners is proposed by [34]. Another algorithm is the dynamic Bayesian network which predicts the tiredness level of the driver [35]. Furthermore, a SVM with a Bayesian filter and a HMM is developed by [36] to predict the human driver's intentions at road intersections. Moreover, a random forest classifier which distinguishes different drivers based on their driving data is proposed by [37].

Additionally, the driver's longitudinal control actions are predicted by [38] using a

Gaussian Mixture Model. In an effort to improve fuel efficiency, a Markov chain has been proposed to predict the driver's throttle input level [39]. Nevertheless, HMM has been a popular data-driven method for driver modelling. Dating back to 1999, a HMM to predict the driver's subsequent actions is proposed by[40]. This is then followed up by more work using HMMs [41] [42] [43].

More recent work is done by [12], where a HMM is developed to predict the driver's steering wheel torque inputs. The data to train, validate and test the model was collected in a high-end driving simulator with human participants driving in highway scenarios at a constant speed of 100km/h. This dataset and driving simulator have also been used for the proposed hybrid controller of this study. Furthermore, an in-depth feature analysis has also been performed by [12] to find the best correlating input data. This helps to find the most optimal input features to maximise the scores on the two KPIs set by the author. These KPIs are the accuracy and the smoothness of the predictions, as described in Section A.3.1. However, as mentioned earlier, the main research gap in this HMM study is the absence of a controller-in-the-loop implementation.

A.1.2. COMPARISON OF MACHINE LEARNING ALGORITHMS

However, a comparison of different ML algorithms has shown ANN models to be an interesting approach for driver modelling [1]. Their study focuses on predicting the steering wheel torque inputs of the driver, with the aim of using it for haptic shared control in ADAS. Specifically, the ML methods considered are Gaussian Process (GP), Gaussian Mixture Regression (GMR), Hidden Markov Model Gaussian Mixture Regression (HMM-GMR) and an ANN. As a form of benchmark to these ML algorithms, a Piecewise Constant Model (PCM) and Piecewise Linear Model (PLM) have also been created. The PCM works with the assumption that the current driver steering torque will be the same as the torque given in the previous timestep, while the PLM takes the time derivative of the steering torque into consideration as well.

The input features used for the ML methods are the road curvature, side slip angle, yaw rate and the steering wheel angle, giving the steering wheel torque prediction as an output. The authors collected two datasets in a simulator, where one was a synthetic dataset created by letting a virtual driver drive the simulated road, while the other dataset was collected from real human driving on their driving simulator. The reason for these two datasets is that the synthetic dataset eliminates the inconsistency of human driving behaviour, enabling a better evaluation of the regression performance of each method. Thus, the real human dataset introduces the inconsistencies within and between each driver, which gives way to evaluating the methods' level of robustness to driver variability. The datasets are evaluated separately from each other. In all cases, the vehicle was kept at a constant speed of 50km per hour.

The results showed that the HMM-GMR and ANN performed better in terms of Root Mean Square Error (RMSE) than the other methods on the synthetic dataset, with the ANN slightly outperforming the HMM-GMR. For the real human dataset, the ANN obtained the best results on average, as shown in Figure A.1, proving that it is more robust to human variation. Finally, it was also found by [1] that the ANN is good at capturing the behaviour of novice drivers. It should be noted that this paper created a relatively simple ANN, with one hidden layer consisting of two nodes. This means that there is much

more room for improvement with optimised ANN methods, which is a topic covered in Section A.2.

Additionally, ANNs have also shown improvements in performance in other fields. For example, a comparison between ML techniques has been performed for language modelling [44] and for computer vision [45], among others.

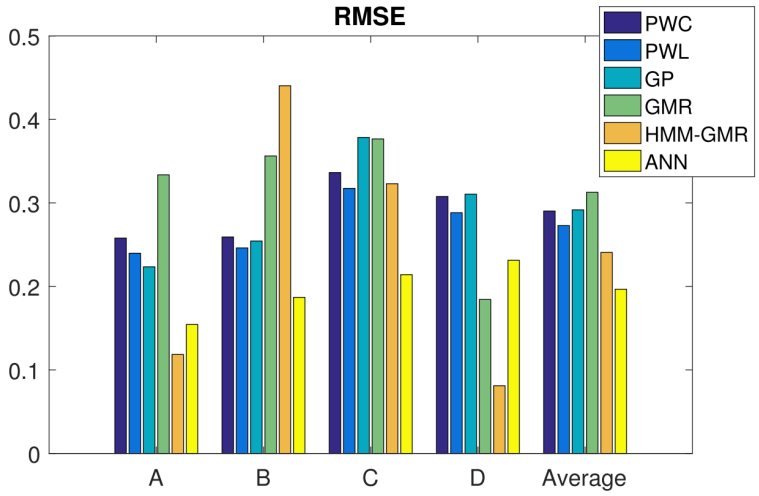

Figure A.1: The RMSE of the different ML methods on the real human dataset, which was split into four subsets A, B, C and D for cross-validation [1].

A.2. ARTIFICIAL NEURAL NETWORK STUDIES

This section gives an overview of different ANN implementations in literature for human behaviour modelling. First, a study will be highlighted which shows the effect of an ANN driver model. Second, the background information of three popular ANN methods will be given. This is then followed by an overview of work done with ANNs for human driver modelling, specifically for lane change and lane following tasks. ANN modelling for other driving related tasks will also be briefly touched upon. Finally, due to the limited work available for driver modelling and as a source of more inspiration, this section also covers work done on ANN implementations for human behaviour modelling for non-driving related tasks.

A.2.1. IMPACT OF A DRIVER MODEL

Before looking into the different ANN approaches for driver modelling, it is relevant to highlight the findings from the study by [2] where they show the effect of an ANN driver model. They propose a comparison between a model-based and data-driven approach to the lateral and longitudinal control of a vehicle. Specifically, they created a MPC algorithm, a topic which will be covered in more detail in Appendix B, and an ANN model. The lateral control aspect of each method is more relevant for this report, with the control input being the steering wheel angle. The driving scenario is related to racing, where the ideal trajectory is the racing line with a variable speed profile. The experiments were done in a driving simulator.

The MPC uses a bicycle model as the lateral dynamics model and does not include any other aspects to directly model the human driver. The ANN is a multi-layer feedforward sigmoidal neural network, where more details about the structure of the network have been left out. The data for the ANN was collected by letting professional human drivers drive at a track on the simulator.

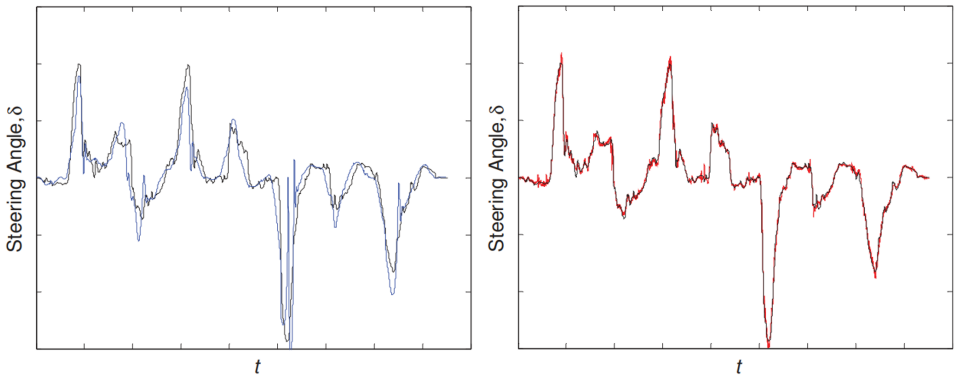

Figure A.2: Comparison between the control inputs given by the MPC controller (in blue) and ANN model (in red) with the correct driver inputs as the reference (in black) [2].

To test both algorithms, the control inputs for a specific section of the track were compared to how the professional drivers would have taken it. The results showed that

the RMSE between the MPC and human driver control input was $1.10 * 10^{-3}$ radians, while the ANN produced a RMSE of $1.97 * 10^{-4}$ radians. Figure A.2 shows the control inputs given by the MPC, in blue, and ANN, in red, compared to a professional driver, in black, for the same section of the track. Thus, this shows that the ANN "provides a better representation of human driver behaviour" [2] as expected, but also highlights the gap between a standard MPC controller and the driver as the driver's behaviour is not accounted for.

A.2.2. TECHNICAL BACKGROUND

Having highlighted the significant effect of an ANN driver model, this section covers the technical background information of three popular types of ANN, namely the Feedforward ANN (FANN), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) network.

FEEDFORWARD ARTIFICIAL NEURAL NETWORK

Starting with a more basic form of ANN, as was also used by [1] and [2], brings the FANN. The FANN is one of the first and simplest type of ANN, with its architecture shown in Figure A.3. It consists of an input layer followed by one or more hidden layers which then lead into the output layer. The number of nodes in the input layer depends on the number of features being passed on to the network. The nodes in the input layer only serve as a passage of the data to the hidden layer(s). The number of nodes in each hidden layer has to be determined during the tuning phase of the network and can significantly impact the performance of the model. The number of outputs required from the model determines the number of nodes in the output layer. As shown by the arrows in Figure A.3, the information flows from the input layer to the output layer without any form of cycles or loops happening, explaining the FANN name.

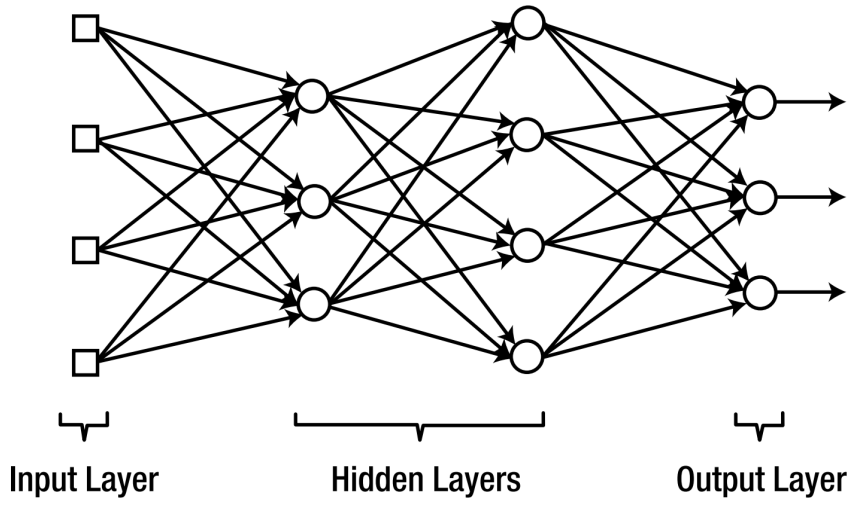

Figure A.3: Architecture of a FANN [3].

The nodes in the hidden and output layers perform a weighted sum, which then goes through an activation function. Figure A.4 shows this process, where three input signals are connected to the node, with each input signal having its own weight factor. Thus, these inputs are multiplied by their weights and then summed together, including a bias factor, as shown in equation A.1 [3].

$$v = \boldsymbol{w}\boldsymbol{x} + b \tag{A.1}$$

where v is the weighted sum, $\boldsymbol{w}$ is the set of weights, $\boldsymbol{x}$ is the set of input signals coming into the node, and b is the bias term.

This weighted sum is then passed on to an activation function, which gives the output of the node. Determining the best activation function is also part of the tuning phase when developing the network. The weights and biases of all the nodes are determined during the training process of the ANN.

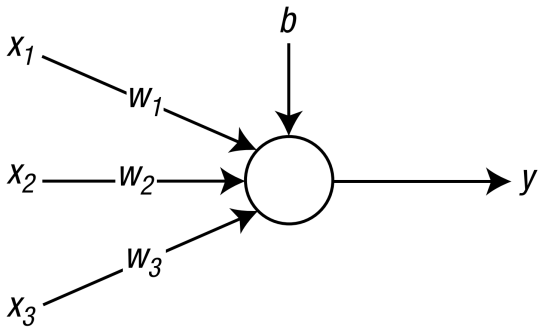

Figure A.4: A node, taken from a FANN, with three inputs [3].

A block diagram portraying the training process of an ANN in general for supervised learning is shown in Figure A.5. The training process is also similar for the other two ANN methods discussed in this section. The training data is collected beforehand, where both the input and correct output are known. For the first iteration of training, random weights and biases can be set for the ANN model. The input data is then passed through the ANN and its output is compared to the correct output using a loss function. A popular loss function is the Mean Squared Error (MSE), where the goal is to minimize it. The error is then used to proportionally adapt the weights and biases of the ANN in a process called backpropagation. These steps are then repeated until either a certain number of iterations have been reached or until a threshold for the MSE has been passed.

Once the network is trained, it can move on to the test phase and be used for its intended application. Only the input data is passed on to the model now, as the correct output is not known yet. The ANN model gives its own output as a prediction based on the input data it received.

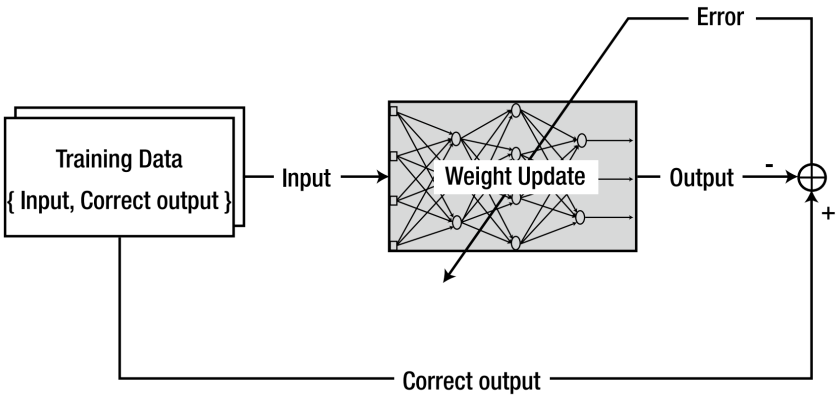

Figure A.5: Block diagram showing the training process of an ANN [3].

CONVOLUTIONAL NEURAL NETWORK

The CNN [46] [47] is a type of ANN which is popular for classification and computer vision tasks, such as object detection in images. A CNN uses convolutional layers as a feature extractor, as the extracted features can then be used for better predictions. Thus, a CNN is often constructed in a way that the input data passes through a convolutional layer, or multiple convolutional layers, followed by one or more fully connected feed-forward layers. However, the convolutional layer is still the main building block. The convolutional layer contains a filter, or also referred to as a kernel. This kernel is a two-dimensional array with weights and serves the purpose of feature detection in the data, for example, finding low-level objects in an image. Thus, the convolutional layer creates a feature map.

Figure A.6 shows an example of how this process works, with the greyscale pixelated boxes in the convolutional layer resembling the kernels. The input image is passed through four different kernels, extracting diverse features from the image. These feature maps can then be used by the following layers for better predictions, as there is more relevant information extracted from the original input.

Continuing with the example of an image, the kernel usually covers a relatively small area of the image in one iteration, as the kernel has a smaller size than the image. Thus, the kernel shifts by a predetermined stride, covering a new area of the image. This is done until the whole image is mapped from the original image to the feature map. This also means that the weights in a kernel, which remain constant during filtering, are shared over the complete input sequence. This parameter sharing makes a CNN more regularized compared to a FANN, as there are significantly less parameters involved during training and testing. The weights of the kernels are determined during the training phase with a process similar to the one already discussed for the FANN.

LONG SHORT-TERM MEMORY NETWORKS

LSTM [48] networks are a type of Recurrent Neural Network (RNN). A RNN, as the name suggest, has loops in them where a cycle is created between one or more nodes. This

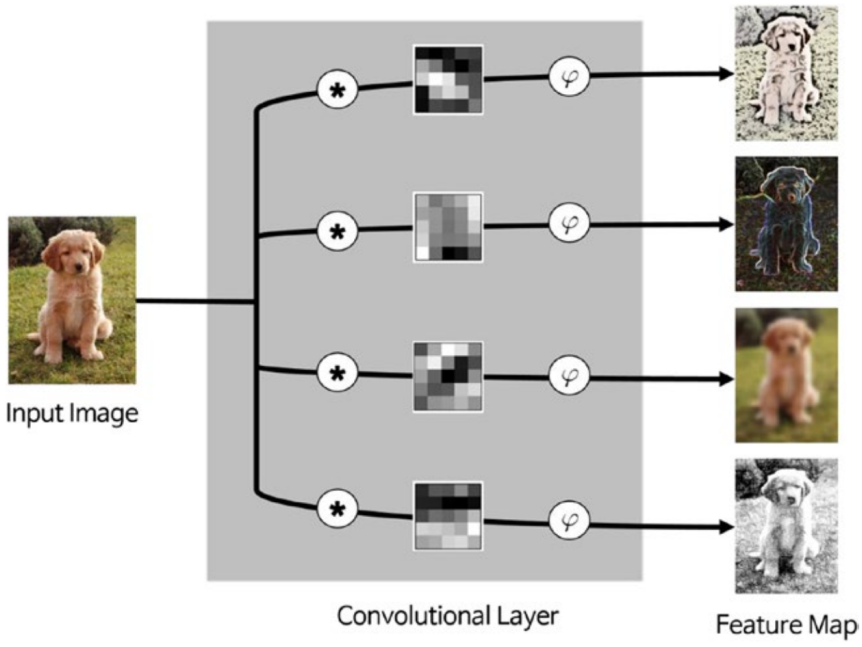

Figure A.6: An input image is passed on to a convolutional layer with four filters, resulting in a feature map with four images [3].

means that the output from a node could have an effect on the input from the next timestep. Figure A.7 shows an example of a RNN unit on the left, with a piece of ANN in the loop. On the right, Figure A.7 shows what the unfolded loop would look like. Thus, processed information from the past is passed on to the next timestep. Having this information passed on, along with the new inputs, enables RNNs to capture relations between past inputs and the new input. This is why a RNN is strong with sequential data, such as timeseries, making it popular for applications such as speech recognition and language modelling.

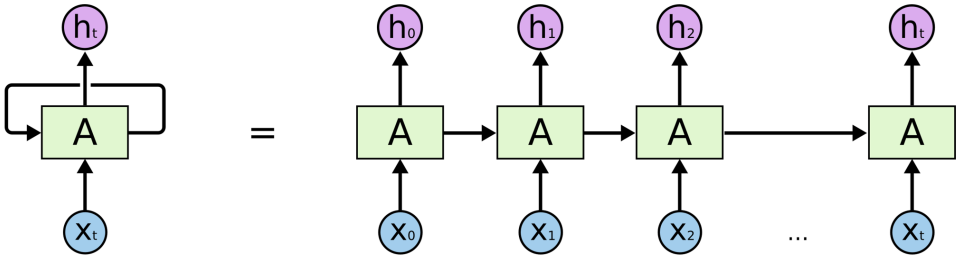

Figure A.7: RNN when unrolling the loop [4].

However, a standard RNN struggles to capture past information for the long-term.

That is why a LSTM has been developed. Figure A.8 shows the unfolded loop of three consecutive LSTM units, where X is the input and h the output from each unit. One unit contains four neural network layers, shown by the yellow boxes. The σ stands for an ANN layer with the sigmoid activation function, which saturates its output to be between 0 and 1, and the tanh stands for an ANN layer with the tanh activation function, which saturates the outputs between -1 and 1.

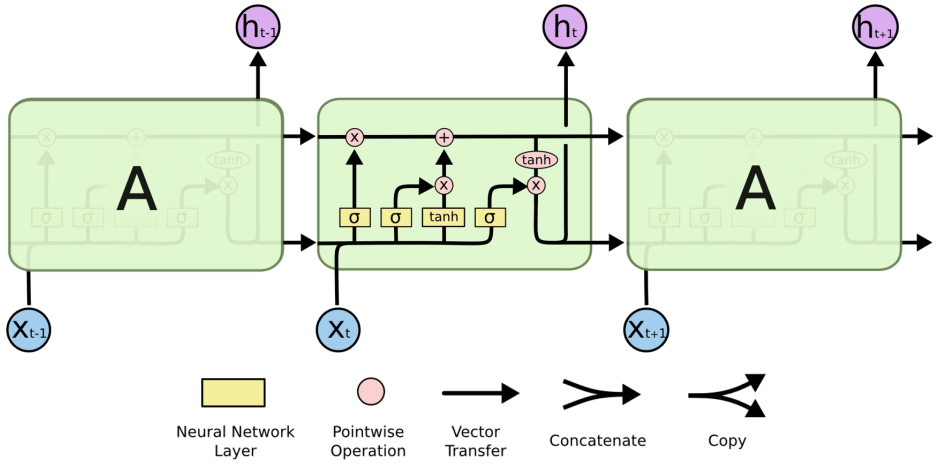

Figure A.8: Unfolded LSTM chain [4].

The part that enables the long-term memory of the LSTM, is the so called cell state C, as shown in Figure A.9. This part is often compared to a conveyer belt which contains information from the previous timesteps and carries it along into the next timestep. To manage the cell state, a LSTM unit uses three gates. Gates, as the name suggest, control what information to block or let through. The gates are the forget, input and output gate.

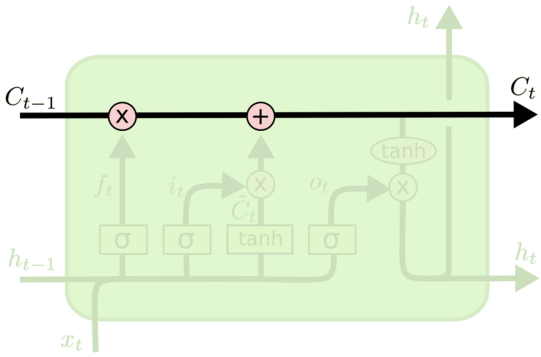

Figure A.9: Cell state of the LSTM [4].

The forget gate f_t, shown in Figure A.10, is a sigmoid layer that outputs which infor-

mation needs to be forgotten in the cell state. In other words, which information from the past timesteps should be taken off the conveyer belt. The equation for the forget gate is shown in equation A.2 [4], where W_f are the weights for the forget layer, b_f the biases, h_{t-1} is the output of the LSTM unit in the previous timestep and x_t is the input data from the current timestep. h_{t-1} passes the information for the short-term memory, as was also the case for a standard RNN. The output from the forget gate contains an array with values between 0 and 1, where a 0 would mean that a specific information needs to be completely forgotten, while a 1 would indicate that no information should be forgotten. The output is then pointwise multiplied with the cell state.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{A.2}$$

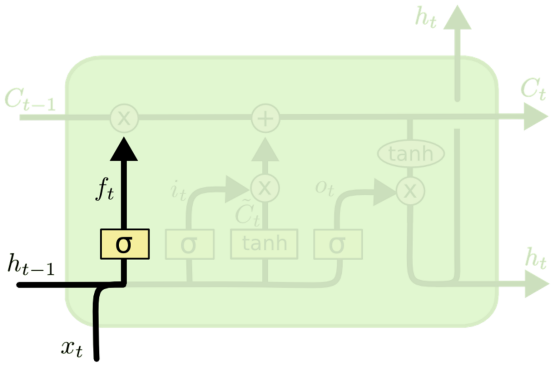

Figure A.10: Forget gate of the LSTM [4].

The input gate i_t, shown in Figure A.11, is also a sigmoid layer. It determines which new information should be added to the cell state, thus the new information that needs to be placed onto the conveyer belt. The potential new information, which has to pass through the input gate, is the so-called candidate state $\tilde{C}_t$. The equations for the input gate and the candidate states are shown in equations A.3 and A.4 [4], where W_i and b_i are the weights and biases for the input gate, and W_C and b_C are the weights and biases for the candidate state. The candidate state is obtained from the tanh layer and then pointwise multiplied with the input gate, as to determine which new information should be added to the cell state.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{A.3}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{A.4}$$

Figure A.12 shows the operations done to the cell state, where a piece of information from the previous timesteps is forgotten and new information from the current timestep is added. Equation A.5 [4] gives the operations done with the forget gate, input gate and candidate state. The cell state is then ready to move onto the next timestep. An

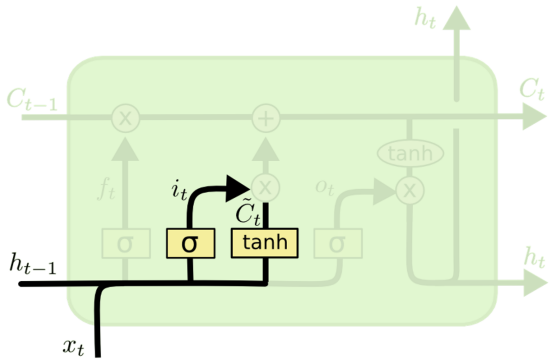

Figure A.11: Input gate and candidates state of the LSTM [4].

example to have a better image of what is happening is for the application of language modelling. If the subject in the first part of the text is a male, all the pronouns would correspond to that. Thus, the gender is stored in the cell state as to make sure that this information is still considered in future timesteps. However, if the subject changes and it is now describing a female, the forget gate would let the cell state forget the male gender and the input gate would add the female gender as new information to be passed onto future timesteps to have the correct pronouns.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{A.5}$$

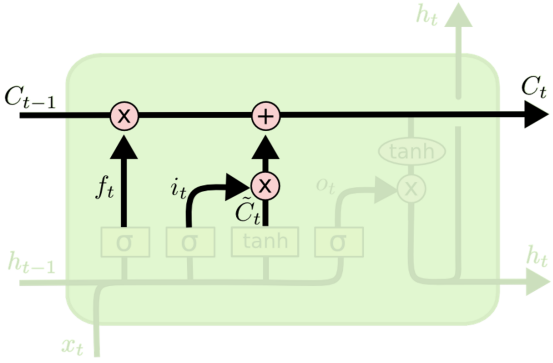

Figure A.12: Operations done to the cell state using the forget gate, input gate and candidate state [4].

Finally, there is the output gate o_t, shown in Figure A.13. This gate, which is again a sigmoid layer, determines which information from the new cell state to use as the final output h_t of the current timestep for this LSTM unit. The cell state itself is passed through a tanh function, saturating its values between -1 and 1. This is then pointwise multiplied by the output gate. Equations A.6 and A.7 show this process, where W_o and b_o are the weights and biases for the output gate.

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{A.6}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{A.7}$$

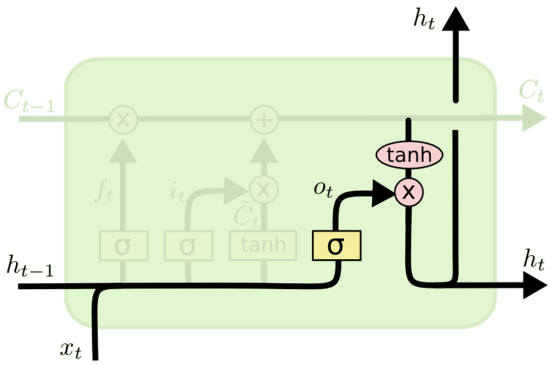

Figure A.13: Output gate and the final output of the LSTM unit [4].

All in all, this explanation has given a better understanding of how a LSTM network works. It shows how this network is able to capture both long and short-term information over sequential data, making it a very promising option for handling timeseries data.

A.2.3. DRIVER MODELLING

This section focuses on ANN models used for driver modelling. The driver model is usually developed for a specific application, such as lateral control or lane change predictions. Due to limited work available on ANN driver modelling for the lane keeping task, this section will first cover ANN driver modelling done for lane change predictions. This is still related to the steering task and the input features used are still relevant to the lane keeping task. Subsequently, an overview of the work done with ANN driver modelling for lane keeping will be given. Finally, human behaviour models for other driving related tasks will be briefly covered.

LANE CHANGE PREDICTIONS

In the domain of lane change predictions, ANN has been a powerful tool in predicting the next action of the driver. The driver could perform a lane change to the left or right, or continue driving on the same road. Predicting lane changes is useful to anticipate what the driver is going to do and for ADAS to act accordingly. For example, the ADAS do not have to give unnecessary warnings when the driver is not going to perform a lane change. This helps in avoiding mistrust from the driver.

Lane change prediction models are often developed as classifiers, where the labels are usually whether the driver is going to perform a Lane Change (LC) event to the left,

right or no LC. However, there are also studies focusing on a regression model. Specifically, they try to predict the Time-To-Lane-Change (TTLC). This is defined as the time between the current timestep and when the vehicle crosses the centre of the lane. Additionally, it was found that the architecture of a classification ANN model for lane change predictions, translates well to a regression model with only minor changes to the output layer of the network [7]. This makes the classification models also relevant for the regression task. This section first gives an overview of classification models, followed by regression models.

FANN classifier

A SVM classifier and a FANN classifier, and also a combination of both, has been proposed by [49]. In this study, they focused on the ego-vehicle merging from an auxiliary lane to the adjacent lane of the main road. This means that the classifiers only have one binary output, specifically whether the driver is or is not going to merge.

The classifiers have seven input features, such as the difference in velocity and acceleration between the ego-vehicle and the lead vehicle, where the lead vehicle is driving on the adjacent lane ahead of the ego-vehicle. Furthermore, the FANN consisted of one hidden layer with fourteen nodes. The combined classifier works on a voting method, where they tested a conservative and aggressive voting method. The conservative method would predict a merge event only if both the SVM and ANN predict a merge, otherwise it is classified as a non-merge. It is the other way round for the aggressive method, where both the SVM and ANN need to predict a non-merge event for the combined classifier to predict the same, otherwise it outputs a merge prediction.

The results show a mixed performance, as the SVM outperforms the ANN for non-merge predictions on one dataset, whereas the ANN is better at predicting merge events. For another dataset, it is the other way round where the ANN is better in predicting non-merge events but worse on merge predictions. For both datasets, the conservative combined classifier outperforms every other method strongly for non-merge classifications, but underperforms in merge predictions, while the aggressive combined classifier lightly underperforms for non-merge predictions and outperforms for merge classifications compared to the other methods. The combined classifier is a good inspiration for a regression problem, where two regression models could also be combined using, for example, a weighted average method.

CNN classifiers

Proceeding, a CNN classifier predicting between a LC or non-LC event has been proposed by [50], specifically a Multivariate Time Series Group-wise CNN (MTS-GCNN). The Multivariate Time Series part of the name indicates that the CNN will handle more than two timeseries data. Specifically, this paper used 22 physiological signals which are derived from the Electrocardiogram, Galvanic Skin Response and Respiration Rate raw signals, such as the heart rate and conductance of the skin.

Their results showed that the MTS-GCNN outperformed both a RNN and a more basic CNN with about 4% more accuracy on average, making the MTS-GCNN look promising. However, the physiological input data relies on physical sensors to be placed on the driver, making its real-world applicability on commercial vehicles more difficult or even impossible.

Another study also proposed a CNN, specifically a Multi-task Attention-based CNN (MACNN) [51]. Multi-task refers to the output of the MACNN where both a classification and regression output comes out. So, one output indicates whether the target vehicle, which is not the ego-vehicle, is expected to remain in the same lane or make a LC to the left or to the right (classification), while the other output predicts the Time-To-Lane-Change (TTLC) of the target vehicle. Having these predictions on vehicles around the ego-vehicle helps the ADAS to plan its next steps, as it can anticipate the movements of the other vehicles.

The attention-based part of the network is a layer which, as the name suggests, puts more attention into some parts of the input data. For example, the target vehicle's environment is full of different objects, but the most relevant information for a given situation could be that the car in front is going slow. This is an important factor for a lane change, thus also more relevant for the CNN to consider. This method could also be used for other driver models as the results show promising performance gains over the baseline models considered. Notably, the environment data around the vehicles is captured from a bird's-eye view in their simulation, instead of using the sensors on the ego vehicle.

This bird's-eye view approach was also used by [52]. However, they created this simplified view from data collected with standard sensors in vehicles during real-world driving. They used a CNN to predict a left, right or no cut-in from a target vehicle. Their network consists of 6 layers, but more details about the architecture and the input variables used are not clear. Interestingly, they implemented a MPC for Adaptive Cruise Control (ACC) which uses the predictions from the CNN. This enables the ego-vehicle to anticipate a cut-in and the MPC tries to act accordingly to improve comfort. Unfortunately, the exact implementation of the CNN output into the MPC has not been described. Their results do show that the ACC with a CNN outperforms an ACC with HMM and a commercially available ACC.

LSTM classifiers

Moving on to LSTM networks, a comparison between a Gaussian Classifier, SVM and LSTM is proposed by [53]. The classifiers give the output as a left, right or no LC. The input data to the LSTM is the lateral position change and the longitudinal position of the vehicle. The network consisted of one LSTM layer and the output layer used a softmax activation function. They found the LSTM to significantly outperform the other methods as it is much more capable in capturing temporal relationships in the data.

Another comparison with a LSTM network is proposed by [54], however, they compared it to an Echo State Network (ESN) [55], which is also a strong method for timeseries predictions. An ESN is part of the reservoir computing framework, which is an extension of ANN where the hidden layer is now a reservoir with randomly partially connected nodes with non-trainable weights, which are usually randomly initialised. Figure A.14 shows the architecture of an ESN. The reservoir is connected to an output layer, which is trainable. Thus, the reservoir serves the purpose of creating an embedding of the inputs.

The LSTM network and ESN were both used to classify for a left or right LC. After looking into different input feature selections, they found the combination of the steer-

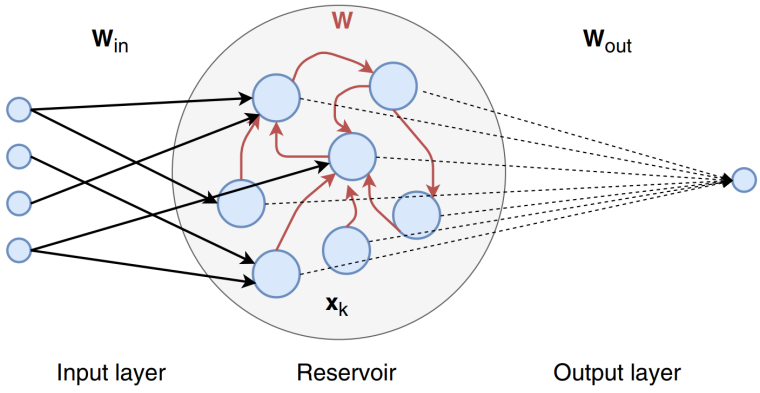

Figure A.14: Architecture of an ESN [5].

ing angle and the indicator of the vehicle to be the best performing. They also used the previous 99 data points per feature as an input to capture the events from the direct past influencing the immediate future. The results showed both models to perform well, but not significantly outperforming each other. Interestingly and inexplicably, the LSTM model is better at right LC predictions, while the ESN model predicts the left LC better. So, a hybrid model using both networks might be an interesting approach.

LSTM regressor

Proceeding to regression models, a LSTM network to predict the TTLC is proposed by [6]. Specifically, their model gives two outputs where one indicates the TTLC to the right and the other to the left. The LSTM model consisted of one LSTM layer followed by a FANN hidden layer and then by an output layer. The LSTM layer had an output dimensionality of 256, while the hidden layer consisted of 32 nodes. As mentioned, the output layer gives two outputs and the activation function used in both the hidden and output layer is the rectified linear unit (ReLU). The loss function used is the MSE.

Furthermore, they used a wide range of input features, such as the distance to the vehicle in front or in the rear, but also the relative velocity and acceleration to the surrounding vehicles. The data was obtained from the highD dataset [56], which contains real-world data. The model also receives the previous three seconds of the features as input. This is done to find relations over time affecting the predictions and because a LSTM network is strong in capturing these temporal relations. Some of the hyperparameters were tuned using a grid search, giving 54 possible combinations. Each combination was evaluated using a 5-fold cross validation.

The results show a RMSE of 0.674 seconds for TTLC to the left (TTLCL) and 0.743 seconds for TTLC to the right (TTLCR). Furthermore, Figure A.15 shows how the RMSE develops over the TTLC. TTLC predictions below 3 seconds appear to be accurate, while the RMSE also appears to be low around a TTLC of 7 seconds. This is because their study capped every TTLC in the data at 7 seconds, even if there is no LC for a while. Thus, values around 7 seconds will be predicted more often. However, this still shows that there is a limit to how far into the future predictions can be made, highlighting the

uncertainty.

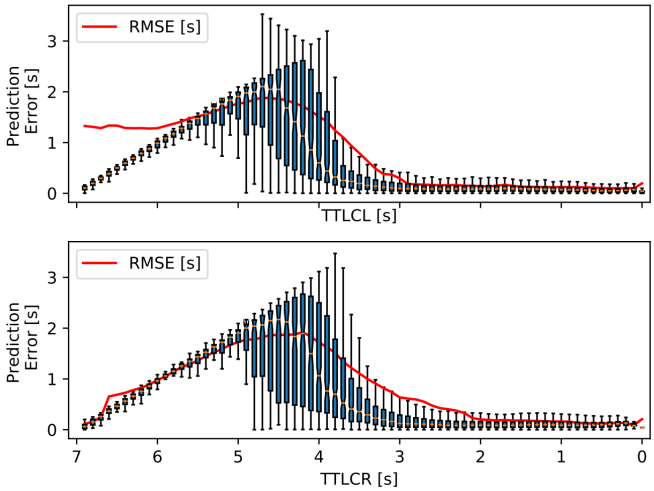

Figure A.15: RMSE and error distribution (boxplot) for the TTLCL and TTLCR predictions [6].

Similarly, a LSTM network has been proposed by [7] to predict the TTLC. However, they collected their data by letting 34 participants with different age and experience drive in a driving simulator. They split their datasets into 26 participants for training and 8 for testing, which means that the trained network will not have seen any of the test split participants during training. Their model uses 9 input features which can be characterized into three main groups, namely, the driver monitoring data, vehicle information data and environmental information data. These 9 input features are the head and gaze directions, obtained from a Head-Eye-Tracking system, the speed, acceleration, steering wheel angle and steering wheel torque of the ego-vehicle. Furthermore, they also use the lateral deviation of the ego-vehicle to the centre of the lane, the gap between the ego-vehicle and the next vehicle on the same lane, and the angle between the lane and the ego-vehicle's longitudinal axis.

Their model consists of one LSTM layer with 250 hidden features, followed by two FANN hidden layers with 256 nodes in each layer. This is then followed by an output layer which gives out two values, the TTLC to the left and to the right. The output layer uses the ReLU activation function. Furthermore, they also applied batch normalization next to implementing dropout layers. This helps in preventing overfitting and in accelerating the convergence.

Another interesting contribution is their use of personalisation in their model. Personalisation, in this case, means that the network is adapted for the driver it is going to make predictions for, also referred to as user adaptation. There are different methods to do this, which will be covered in Section D.1. This paper, however, uses a method where the model trained on the 26 participants is retrained on 20 minutes of driving data from a specific participant in the test set. The remaining data from that participant, which

is 10 minutes of driving, is then used as the final test data to evaluate the performance. This is then repeated for the other test participants. Thus, the initial trained model forms a basis for each personalised model.

Finally, the model is compared to a baseline model, which is a Random Forest (RF) regression model. The LSTM model outperforms the RF in terms of RMSE for each time horizon, as can be seen in Figure A.16. Furthermore, the personalised LSTM model showed better performance for each test participant compared to the base LSTM model. The improvement was especially apparent for larger labelling windows. This means that predictions within smaller time windows work fine with a general model, but predictions further into the future require more knowledge about the current driver.

Labeling Window	**2.5s**	**3s**	**4s**
RF Regression	0.3077	0.3415	0.3447
LSTM Regression	0.2723	0.3081	0.3292

Figure A.16: RMSE of the RF and LSTM regression models for three different labeling windows [7].

Fuzzy Neural Network regressor

Lastly, a different approach to the problem has been proposed by [57]. While the other studies created classifiers to predict a LC or regression models to predict the TTLC, their study predicts the future steering wheel angle applied by the driver. This is then used to determine the trajectory of the vehicle, which in turn indicates whether a LC is expected. The predictions were made using a Fuzzy Neural Network (FNN), which is a method where the ANN learns the parameters of the fuzzy system, such as the membership functions, from data. This makes the reasoning behind the ANN's output more interpretable.

The FNN has five inputs, namely, the longitudinal and lateral distance between the ego-vehicle and the vehicle in front. Also, the relative velocity between both vehicles, and the travel heading angle and the acceleration of the ego-vehicle are used. As mentioned, the only output is the steering wheel angle. Furthermore, this paper also implemented an adaptive learning method, where the RMSE of the prediction forms the basis for updating the membership functions and fuzzy inference. The results show that the FNN outperforms the four benchmark models they created for their comparison. These models are a conventional ANN (more details about the architecture were not mentioned), SVM, HMM and Multivariable Linear Regression. Most interestingly, they found out that the heading angle and acceleration of the ego-vehicle had the most influence on accurate steering wheel angle predictions out of the five input features.

LANE KEEPING TASKS

This section gives an overview of human driver modelling research related to the lane keeping task using ANN models. FANN models proposed by [1] and [2] were already covered in Sections A.1.2 and A.2.1, respectively. Some research focused on predicting the steering wheel angle, while more recent studies have focused on the steering wheel torque. Having the steering wheel torque as a control input to a haptic system allows the

system to feel more compliant to the driver, creating a better steer feel, more comfort and user acceptance.

FANN model for steering wheel torque predictions

A FANN model is proposed by [58] to model human driver steering behaviour. This model is intended to be used for a steer-by-wire vehicle where the torque feedback matches the driver and what he/she would normally feel, thus the model attempts to learn the steering feel. The input features to the ANN are the steering wheel angle, steering wheel angle speed, vehicle velocity, longitudinal acceleration, lateral acceleration and yaw rate. The model then gives the steering wheel torque as an output.

They investigated how different configurations of their FANN would influence the performance, by changing the number of hidden layers and the number of nodes per layer. Within the configurations they looked into, a FANN with two hidden layers and 32 nodes each performed the best with a RMSE of 0.426Nm. However, this model only uses the input from the current timestep to make a prediction into the immediate future, even though the data is in timeseries.

That is why they also investigated adding, so called, sliding window features. The sliding window is a set time window within which extra input features can be extracted from the immediate past, specifically the mean and standard deviation of the original input features. This time window then progresses as time passes on. This captures extra information about the inputs from the past.

Just as with the initial FANN, the model architecture with two hidden layers and 32 nodes per layer obtained the best performance for the FANN with sliding window features. This model gave a RMSE of 0.318Nm, outperforming the FANN without sliding window features. They found that their FANN is generally sufficiently accurate. However, it was lacking performance at lower speeds. Thus, they recommend investigating other types of ANN to improve the overall performance, where they mention CNN and RNN to be interesting options. Specifically, they mention LSTM models to be promising.

FANN model for steering wheel angle predictions

A FANN is also proposed by [8]. However, their goal was to model the steering wheel and accelerator pedal behaviour of only expert drivers. This model would then be used to provide haptic feedback and assistance to train novice drivers. To accommodate predictions for both the steering wheel and accelerator pedal, they used two FANN in parallel, where one only predicted the steering wheel angle and the other only the accelerator pedal angle. Both architectures were exactly the same, containing 4 hidden layers with the sigmoid activation function, where the first hidden layer contained 32 nodes, the second 16 nodes, the third 8 nodes and the fourth 4 nodes.

The input features consisted of vehicle and environmental state data. Specifically, the vehicle state contained the longitudinal and angular velocity, and the revolutions per minute of the engine. The environmental state was accounted for by using the distance between the driver and road boundaries from five different angles. These angles were 30°, 15°, 0°, -15°and -30°from the longitudinal axis. They chose this total of 60°as it covers the field of view of the driver in their driving simulator. The output from both networks gives a prediction of 0.2 seconds into the future, as to compensate for human body time

delays. The data to train the ANN was collected by letting five expert drivers drive specific paths in a driving simulator while trying to maintain a speed of 60 km/h.

Furthermore, they created a Proportional-Integral-Derivative (PID) controller to determine the Haptic Guidance (HG) torque for the steering wheel, where the error is the difference between the current steering wheel angle and the predicted expert driver steering wheel angle. The HG was applied as a guidance for the novice driver.

The results cover two parts of their work. First, they only looked at the prediction performance of the FANN model, where they analysed the steering wheel angle error between the model and what the driver actually gave as an input. When evaluating the model with new expert driver data, they obtained a mean prediction error of 1.55%. For data collected from novice drivers driving without any HG, they managed to get a mean prediction error of 2.55%. Figure A.17 shows the plot of the predicted steering wheel angle (in red) compared to the actual steering input from a novice driver.

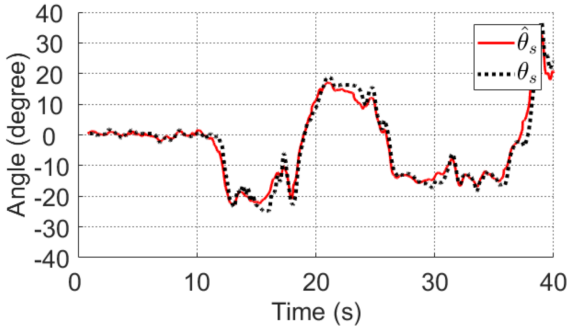

Figure A.17: Prediction performance of FANN model on a novice driver, with the predicted steering wheel angle (red line) and the actual driver input applied (dotted black line) [8].

Second, they compared the objective and subjective performance of their HG system to a conventional HG. In this case, the novice drivers drove different scenarios with the HG systems and were asked to answer questions regarding the effectiveness, comfort, fun and helpfulness of the assist. Looking at the objective performance, they obtained a mean prediction error of 1.10% for the FANN HG and 1.42% for the conventional HG. The prediction error in this case refers to the error between the novice driver's path with HG and the expert driver's path without HG. Furthermore, the FANN HG is better at following the centre of the lane compared to the conventional HG, while it performs worse in maintaining the correct heading angle of the car.

Finally, the subjective data shows that the FANN HG creates more comfort, while the conventional HG was found to be slightly more effective. However, in terms of fun and helpfulness, the participant preferred having no HG at all. This shows that there is still a gap present, which could be solved by having the ANN be trained on novice driver data as well, along with working with the steering wheel torque instead of the angle. This could improve the prediction results and lead to better acceptance of the assist, as it would feel more natural to the novice drivers.

Bidirectional LSTM model for steering wheel torque predictions

Since LSTM networks show good performance with timeseries data, a Bidirectional LSTM (BiLSTM) [59] network to predict driver steering wheel torque inputs using Electromyography (EMG) signals has been proposed by [10]. BiLSTM is a type of LSTM where there is both a, so called, forward and backward pass, as shown in Figure A.18. As was described in Section A.2.2, a LSTM unit passes its outputs to the next timestep, resulting in information from the past being carried forward. With a backward pass, the information from the input sequence flows to the previous timesteps, carrying information from later timesteps back. This helps in also finding relations from the present that have affected the past. The forward and backward layers get the same inputs, after which their outputs are concatenated as the output of the BiLSTM layer. This type of ANN has also been used for the proposed hybrid controller.

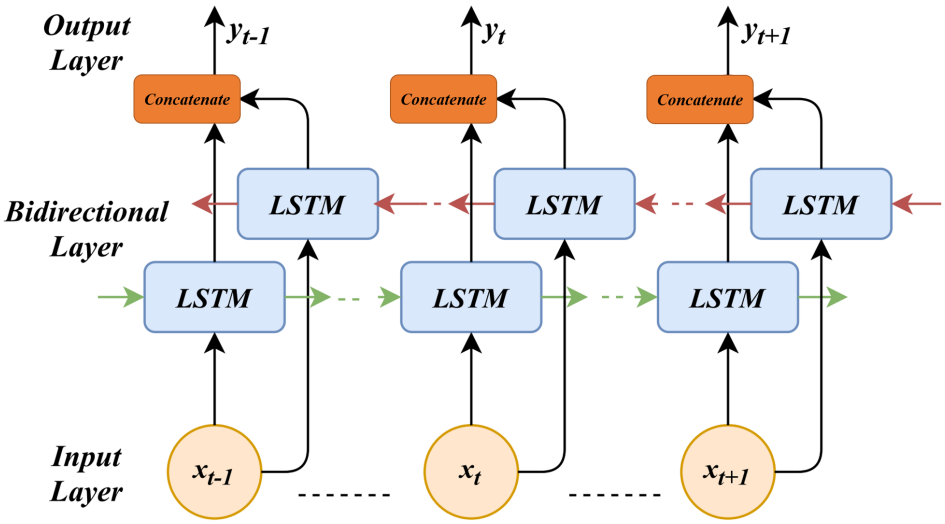

Figure A.18: Architecture of a BiLSTM layer with forward pass (green arrows) and backward pass (red arrows). Adapted from [9].

They used two BiLSTM layers for their steering wheel torque prediction. The first BiLSTM layer contains, in each direction, 60 LSTM hidden features, while the second has 40 LSTM hidden features. The second layer is then followed by a FANN hidden layer with 100 hidden nodes, followed by the output layer. The input data consists of 10 EMG signals coming from the upper-body and of the applied steering wheel torque measured by the torque-sensor. The data was collected by having 21 male participants drive in a driving simulator.

The historical horizon was chosen to be 200 timesteps, where one timestep is 1ms. This means that all the EMG and steering wheel torque data from the past 200ms was given as an input during each iteration. The prediction horizon was also 200 timesteps, resulting in steering wheel torque predictions of 200ms into the future as the output. This was done since they found out that approximately 200ms was the time delay be-

tween the EMG signals and the steering wheel torque being applied.

Another contribution by [10] is the use of transfer learning to create a second BiLSTM model for a discrete prediction of the steering input. Specifically, the prediction gives one of the five classes, namely, “right steering”, “right steering back”, “left steering”, “left steering back” and “hold”. The reason for this classifier is that it “can be an essential input to the take-over performance assessment system and a high-level collaborative decision-making system for the automated driving vehicle” [10]. The transfer learning is because this model uses the first BiLSTM layer from the continuous prediction model. This layer is seen as the temporal pattern extraction layer, which can be shared over both networks, as they share the same inputs. For the discrete model, this layer is then followed by a new BiLSTM layer with 40 LSTM hidden features, followed by a FANN hidden layer with 100 hidden nodes and, finally, followed by a softmax output layer.

For the results, they compared their continuous model to a FANN, Time Delay ANN (TDANN) and LSTM model, which has a similar architecture as the BiLSTM model. They also created a version of their BiLSTM model where only the EMG signals with a strong correlation to the output were used, resulting in only 3 out of 10 EMG signals being used. They found their normal BiLSTM model to perform the best with a RMSE of 0.582Nm, followed by their 3 EMG signal BiLSTM with 0.624Nm. The LSTM model fell behind with a RMSE of 1.548Nm. The FANN obtained a RMSE of 2.558Nm and the TDANN 8.902Nm. Overall, the BiLSTM outperforms the other models and Figure A.19 shows a plot of its predictions. It can be seen that this model works well, but it could still improve its predictions at the peak points, which are the steering reversals. Furthermore, they found that their history and prediction window worked well for their task. However, Figure A.19 does show a limitation of this research, where the plot suggests that a simplified steering model is used to simulate the vehicle where no friction forces are modelled.

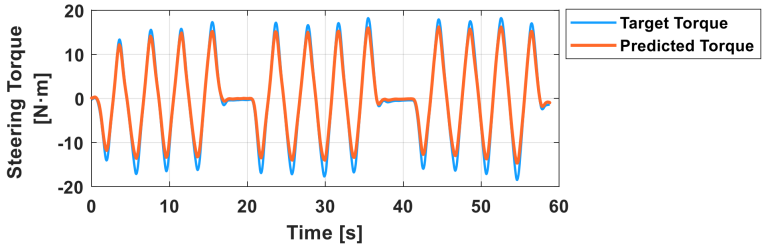

Figure A.19: Prediction performance of the BiLSTM model [10].

Deep Convolutional Fuzzy System model for steering wheel angle predictions

Another study used expert driver data to train its ANN model, specifically they developed a Deep Convolutional Fuzzy System (DCFS) model [60]. Their aim was to capture the motor intermittency of human drivers by modelling the driving behaviour of expert drivers. They used their DCFS model as the feedforward controller, while also implementing a supervisory feedback controller which only acts when the vehicle reaches the set boundaries, like a Lane Departure Assist.

The data was collected in a driving simulator where the input data to the model consisted of data obtained from the vehicle camera sensors and the vehicle state data. The output from the model was the steering wheel angle. Furthermore, they compared their controller to two MPC controllers, where MPC-1 penalized the control effort relatively more, while MPC-2 penalized the tracking error relatively more. They found their DCFS model to capture the expert driver behaviour the best without losing out too much in terms of tracking performance, while also capturing the motor intermittency. Furthermore, it seemed to lower the mental workload of novice drivers when they performed the driving task in the driving simulator.

However, during their data collection and experiments, the authors told their 15 participants to follow the centre of the lane. This reduces the human variability for the steering task. Thus, the DCFS model does not fully capture the natural driving behaviour of the driver and does not show full human-like behaviour as an assist. Furthermore, the speed of the vehicle was kept relatively low at a maximum of 35km/h.

Reinforcement Learning

All the ANN models discussed until now used supervised learning. This means that the ANN model has been trained with previously collected data which contains both the input and corresponding output features. With this data, the model learns a mapping between the input and output. Once the model is being used in real-time, the mapping gives a prediction based on the new input data passed on to the model. However, deep reinforcement learning to create a comfort oriented LKA for haptic guidance has been proposed by [61], where they try to increase driver acceptance of the assist.

Reinforcement Learning (RL) uses rewards and penalties to make the learning algorithm obtain the optimal policy for the task at hand. Essentially, the policy is a mapping between the input data and the action performed by the LKA, and it can often be a function approximator, such as an ANN. Given the overall goal, the learning algorithm tries to find the policy that gives the most rewards and continuously updates the policy based on the rewards. For example, sharper turns would require more steering input from the LKA to stay within the lane. Through trial-and-error, the RL algorithm learns the optimal action and remembers it for future sharp turns.

A Deep Q Network [62] has been researched by [61] as their RL method, where they used a fully connected FANN as the function approximator. The input data consists of the steering wheel angle and the lateral error between the vehicle and the centre of the lane. The evaluation of the model was done by looking into the lane keeping accuracy with respect to the centre of the lane, and the steering control activity, as this covers the comfort aspect of the assist.

They compared their RL method to a continuous and threshold-based HG. They found their method to obtain better performance on lowering steering control activity, while still having maintained an overall good lane keeping accuracy. However, their study presents a very novel approach to using RL for LKA. They simulated the driver for their experiments, in which they only used a straight road with the vehicle travelling at 60km/h. This oversimplifies the method significantly, making it difficult to assess the relevance of using RL for driver modelling.

OTHER DRIVING TASKS

This section briefly covers driver modelling studies which focus on other tasks than lane changes and lane keeping. Firstly, the work done by [63] models the driver for the longitudinal movement of the vehicle, specifically the throttle inputs. They did this by having a fixed ANN and an adaptive ANN in parallel. The fixed ANN is responsible for predicting the throttle behaviour of the human driver and acts as a controller to follow the target speed in a human-like fashion. The adaptive ANN uses an online training algorithm which has been implemented to account for any changes in the vehicle and road conditions, where it tries to reduce any small speed errors. Both networks use a single-layered FANN, where the fixed ANN uses five hidden nodes, and the adaptive ANN uses three hidden nodes.

Secondly, another study focused on the lateral and longitudinal trajectory predictions of surrounding vehicles [64]. These predictions can then be used by the ego-vehicle to plan its actions. For their model, they used one LSTM layer with 256 units, followed by two FANN layers with 256 and 128 hidden nodes. The input features are mainly related to information human drivers would base their actions upon, such as the relative velocity between the ego-vehicle and target vehicle, and the time-to-collision. The outputs are the predicted lateral position and longitudinal velocity of one target vehicle. Their data comes from the Next Generation Simulation US101 dataset [65], which contains more than 6000 trajectories of different vehicles during real driving on highways. Overall, their model shows good performance, with the LSTM model showing promising results in handling timeseries data.

Thirdly, a model containing two parts for autonomous driving is proposed by [66]. One part uses a CNN to detect objects on the road and determines the lane boundaries. The other part uses this information to plan and control the vehicle. This is done using a LSTM network, which contains three LSTM layers. The CNN model uses the images from three front-facing cameras mounted on a car, where the output of the CNN is used to map the environment features. The data used was collected using real-world driving. The extracted environment features are then passed onto the LSTM network alongside vehicle state data, such as the velocity. This network then gives the steering commands, which mimics human behaviour as it was trained on human driving data.

Finally, a FNN to predict the electric current going to the electric motor of an Electric Power Steering (EPS) system is proposed by [67]. The input features are the steering wheel torque inputs by the driver and the vehicle speed. Their aim is to achieve EPS which feels more natural to the driver, while avoiding the difficult parameter tuning of a conventional PID controller. They found promising results, where their FNN controller was more robust than a conventional PID controller.

A.2.4. NON-DRIVING RELATED TASKS

This section covers work done in ANN user modelling for tasks which are not related to driving. Even though it is not related to driving, it does give a good overview of which methods have shown to be promising in human modelling in other domains, broadening the scope of this report. This could be an inspiration for certain models or methods to be implemented in automotive applications.

For example, the model developed by [68] predicts human motion using their own

custom ANN called Motion Prediction Network where they use CNN layers to predict the pose of the human. They also use meta-learning, which will be covered in Section D.1, to learn new motion categories with limited data. A study by [69] also focuses on human activity, but specifically on activity recognition for Ambient Assisted Living systems. These are systems which are used for humans who need extra care, such as the elderly. They give a complete overview of the different modelling attempts in this domain, such as LSTM models.

Furthermore, a popular domain for human behaviour prediction is Human-Robot Interaction (HRI). The work by [70] tries to predict whether the human is going to take the task over from a collaborative robot. They used a Spiking Neural Network, which is inspired by the human brain. A RNN to predict the path of visually impaired humans when following a guiding robot has been proposed by [71]. This knowledge would help the robot to plan future actions while considering the human. Another aspect of HRI is that the human can teach the robot to perform certain tasks, as proposed by [72]. They created a modular architecture covering different aspects of performing a task. Thus, they were able to implement ANN models from other work for tasks such as language comprehension.

ANN is also popular for temporal modelling of other tasks besides human modelling, such as in the chemical industry and investing. A RNN to model a reactor in a chemical plant is proposed by [73], where they try to control the temperature of the reactor. The model developed by [74] makes predictions in the Energy Market, where it tries to determine the future electricity prices using a BiLSTM network. This model serves as an advisory tool for investors.

Finally, the research done by [11] will be covered in more detail, as it provides an interesting approach to human user modelling and applying HG to the user during a virtual air hockey game. Figure A.20 shows the experiment setup on the left, where a haptic device is used to control the player's paddle on the screen. On the right of Figure A.20, a schematic overview is given of the different elements of their architecture.

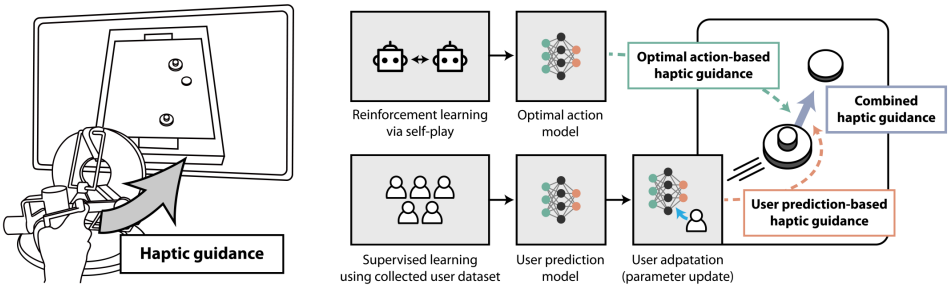

Figure A.20: The experiment setup (left) and HG controller architecture (right) [11].

Through RL, they have created an Optimal Action (OA) model, where two Artificial Intelligence agents play against each other to develop the optimal policy to play the game, which outperforms a human player. For a lane keeping task, this could be seen

as following the centre of the lane very precisely. On the other hand, they have also created a User Prediction (UP) model with a supervised ANN, which predicts the inputs the human player is going to give through the haptic device. An example of this for lane keeping would be the corner cutting human drivers sometimes do. The ANN is trained on previously collected data from users playing the game without any HG.

Since every human is different from the other, a specific user would play the game in a different manner. To close the gap between the base model and a specific user, they use meta-learning to account for User Adaptation (UA) with a relatively small amount of user-specific data. This means that the UP model gives better and more personal predictions for a specific user. Furthermore, their model gives the mean and standard deviation as the prediction. They used the standard deviation to quantify the uncertainty of the prediction, which could then be used to determine the level of HG. If the uncertainty is high, the HG would be very soft or not activate at all.

From the two models, they tested three types of HG: OA-based, UP-based and a combination of both. The combined method was implemented by taking the average of the predictions from the two models. They also added an extra weight which lowers the output of the combined method depending on the disagreement between the OA and UP model. The results showed that the combined method of HG "exhibited a further decrease in user disagreement ... without reducing any objective and subjective scores" [11].

A.3. PROPOSED BILSTM MODEL FOR THE HYBRID CONTROLLER

The architecture and hyperparameters of the BiLSTM model developed for the proposed hybrid controller is defined in Chapter 2. This section describes the KPIs used to evaluate the model and gives an additional analysis of the model's performance.

A.3.1. KEY PERFORMANCE INDICATORS

The two KPIs, as defined by [12], are used to evaluate the intermediary models during the tuning phase and the final model. The two metrics are the accuracy and smoothness of the prediction

1) The accuracy of the prediction is described as

$$A_{\mathbf{T}_{pred}} = \left(1 - \frac{1}{SD\left(\mathbf{T}_{driver}\right)} RMSE\left(\mathbf{T}_{pred}\right)\right) \cdot 100 \tag{A.8}$$

where $\mathbf{T}_{pred}$ is the predicted SWT the driver is expected to give and $\mathbf{T}_{driver}$ denotes the SWT the driver actually gave. Furthermore, the SD of the driver's SWT is defined as

$$SD\left(\mathbf{T}_{driver}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(T_{i,driver} - \mu_{\mathbf{T}_{driver}}\right)^2} \tag{A.9}$$

and the RMSE of the predicted SWT to the driver SWT is defined as

$$RMSE\left(\mathbf{T}_{pred}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(T_{i,pred} - T_{i,driver}\right)^2}. \tag{A.10}$$

2) The smoothness of the prediction is described as the SD of the time derivative of $\mathbf{T}_{pred}$. Thus, the smoothness metric is defined as

$$SM_{\mathbf{T}_{pred}} = SD\left(\dot{\mathbf{T}}_{pred}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\dot{T}_{i,pred} - \mu_{\dot{\mathbf{T}}_{pred}}\right)^2} \tag{A.11}$$

where a lower value represents a smoother prediction.

A.3.2. DATASET

The dataset was recorded by [12] on TME's advanced driving simulator, where 7 participants took part in a total of 14 hours of driving in an advanced driving simulator. The task of the participants was to drive on the middle lane of a three-lane highway, where they only performed the steering task, as the vehicle was set to a constant speed of 100km/h. The 2 hours of driving for each participant consisted of 24 different road scenarios with different combinations of corners, where the data was collected at a rate of 100Hz. The data was split into 50% for training, 25% for validation, and 25% for testing.

An extensive feature analysis has also been performed by [12] to find the most relevant input features for predicting the steering wheel torque actions of the driver. The relevant input features were shortlisted as the top 18 candidate features. The candidate features have also been considered for the proposed hybrid controller. These input features and their corresponding units are shown in Table A.1, where the first 11 features are vehicle data features and the last 7 features are environment data features.

Table A.1: Candidate features [12] with the first 11 features being related to the vehicle and the last 7 to the environment around the vehicle.

#	Feature	Unit
1	Steering Wheel Angle	rad
2	Steering Wheel Velocity	rad/s
3	Steering Wheel Acceleration	rad/s^2
4	Lateral Velocity	m/s
5	Lateral Acceleration	m/s^2
6	Slip Angle	rad
7	Yaw Rate	rad/s
8	Yaw Acceleration	rad/s^2
9	Roll Angle	rad
10	Roll Velocity	rad/s
11	Roll Acceleration	rad/s^2
12	Deviation Distance at 0m	m
13	Deviation Angle at 0m	rad
14	Road Curvature at 0m	1/m
15	Deviation Angle at 10m	rad
16	Road Curvature at 10m	1/m
17	Deviation Angle at 30m	rad
18	Road Curvature at 30m	1/m

Furthermore, since the raw data was used, the noisy input features were pre-processed with a zero-phase low-pass filter. The filter settings were set as shown in Table A.2.

Table A.2: Low-pass filter settings.

Parameter	Value
Sampling frequency	100 Hz
Pass band frequency	10 Hz
Stop band frequency	15 Hz
Pass band ripple	0.01 dB
Stop band attenuation	60 dB

A.3.3. FEATURE ANALYSIS

This section covers a small part of the feature analysis done to determine the ideal set of features for the proposed BiLSTM model. This will give an impression of the effect certain features have on the KPIs. To eliminate influence from any other factors, the hyperparameters have stayed constant while the weights of the model have been initialised with the same value for each set of features. The architecture of the model has also stayed the same as the model mentioned in Chapter 2. The hyperparameters used for this feature analysis are defined in Table A.3. Furthermore, only the training dataset is used to train the model and the validation set is used to evaluate it. The test set is left untouched as that has been used for the final evaluation described in Chapter 2.

Table A.3: Hyperparameters set for the feature analysis.

Hyperparameter	Setting
Loss function	MSE
Optimiser	Adam
Learning rate	$2.0*10^{-5}$
Number of epochs	10
Batch size	500
Validation split	0.25
Historical window	0.5s
Prediction horizon	0.4s

CANDIDATE FEATURES

First, all 18 candidate features mentioned in Table A.1 have been tested. Figure A.21 shows the accuracy and smoothness scores of this model over the prediction horizon. It also gives a fragment of the SWT predictions at t=0. The accuracy drops gradually over the prediction horizon while the smoothness metric increases. The smoothness of the drivers in the validation dataset is 2.29Nm/s, showing that this model's predictions are less smooth.

However, it should be noted that having more features also takes more epochs for the model to fit the data properly. For example, training this exact same model for 170

epochs gives the scores shown in Figure A.22, where it significantly improves in terms of both accuracy and smoothness. Besides, having a more complex architecture with more layers and nodes also results in better KPI performance for this feature set. Nevertheless, that is beyond the aim of this section, as it attempts to showcase the effects of certain types of features.

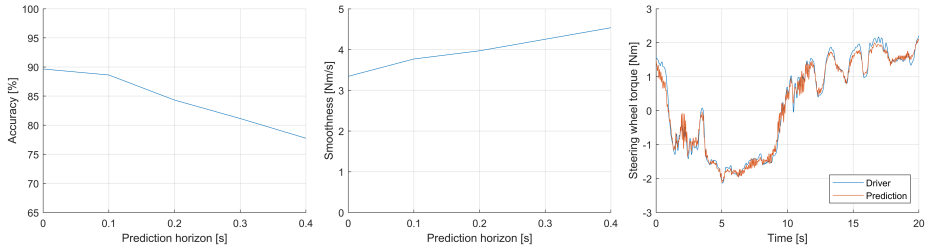

Figure A.21: Accuracy (a), smoothness (b) over the prediction horizon and SWT prediction at t=0 (c) using all 18 candidate features.

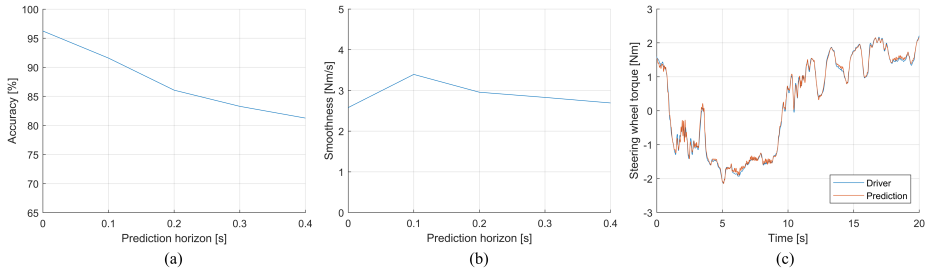

Figure A.22: Accuracy (a) and smoothness (b) over the prediction horizon and SWT prediction at t=0 (c) using all 18 candidate features, but with 170 epochs.

VEHICLE FEATURES

The second set of features only uses the data related to the vehicle, which are the first 11 features listed in Table A.1. Figure A.23 shows the performance of this model. It shows improved accuracy at t=0, but gradually gets worse over the prediction horizon. Additionally, it also performs worse in terms of smoothness over the entire prediction horizon. This shows that only having the vehicle features improves the prediction into the immediate future but does reduce the performance further into the horizon, while the predictions overall are not smooth.

ENVIRONMENT FEATURES

Thirdly, only the features related to the environment around the vehicle are used. These features are the last 7 features shown in Table A.1 and are data collected by the virtual sensors of the vehicle which capture information regarding the path of the vehicle and the relative deviation from it. Figure A.24 shows the performance of this model, where

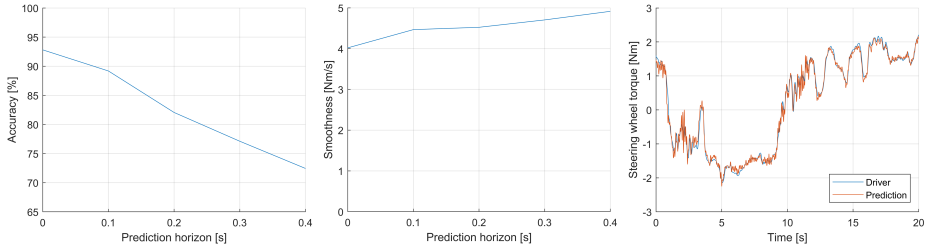

Figure A.23: Accuracy (a) and smoothness (b) over the prediction horizon and SWT prediction at t=0 (c) using the vehicle data features.

a completely different behaviour can be seen. The accuracy has gotten worse, while the smoothness has improved significantly. Furthermore, the KPIs stay approximately consistent over the prediction horizon. The SWT predictions from this model also show the effect of the improved smoothness, as the high to medium frequency oscillations have been eliminated.

During the subjective evaluations performed with expert drivers on the driving simulator to tune the BiLSTM model, as described in Chapter 2, it was found that this feature set created the most pleasant SWT feedback due to the smoothness. Additionally, this model is significantly smoother than the drivers in the validation set as it eliminates the minor corrections humans tend to make while steering.

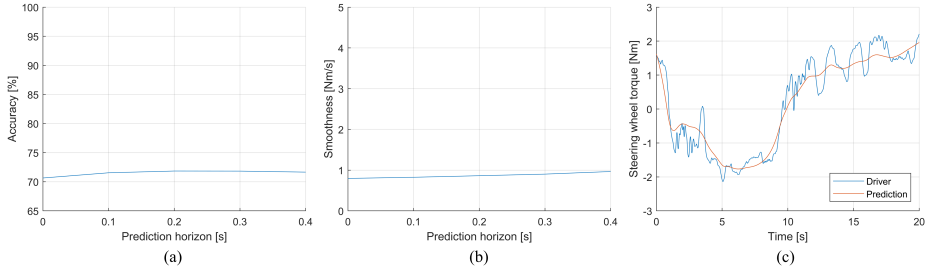

Figure A.24: Accuracy (a) and smoothness (b) over the prediction horizon and SWT prediction at t=0 (c) using the environment data features.

STRONGEST CORRELATED FEATURES

Thus, it has been shown that the vehicle data improves accuracy at t=0, while the environment data improves smoothness and creates more consistency over the prediction horizon. Using all the candidate features has shown how these two types of feature sets balance each other out. Lastly, it is interesting to look at the 10 strongest correlated candidate features, as determined by [12] and shown in Table A.4, to see how another combination of the vehicle and environment features performs.

Figure A.25 gives the plots of this model. It shows that the model performs slightly worse in terms of accuracy over the prediction horizon, except at t=0, compared to the model trained on all candidate features. However, it does show an improvement

in smoothness over the entire prediction horizon. This difference can be explained by fewer features being present in this feature set resulting in fewer epochs being needed to fit the data. Nevertheless, this still shows the balance both vehicle and environment features create between accuracy and smoothness, as the smoothness has improved due to more vehicle features being eliminated from the set.

Table A.4: Strongest correlated candidate features [12].

#	Feature	Unit
1	Steering Wheel Angle	rad
4	Lateral Velocity	m/s
5	Lateral Acceleration	m/s^2
6	Slip Angle	rad
7	Yaw Rate	rad/s
9	Roll Angle	rad
14	Road Curvature at 0m	1/m
16	Road Curvature at 10m	1/m
17	Deviation Angle at 30m	rad
18	Road Curvature at 30m	1/m

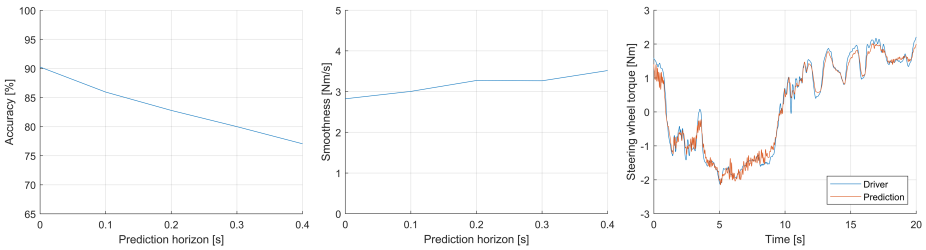

Figure A.25: Accuracy (a) and smoothness (b) over the prediction horizon and SWT prediction at t=0 (c) using the strongest correlated features [12].

A.3.4. PREDICTIONS OVER HORIZON

Thus, the environment features were found to be the most optimal for a pleasant assist. This section analyses the predictions made by the proposed BiLSTM model in more detail, specifically focusing on the performance over the prediction horizon.

As described in Chapter 2, the BiLSTM model has a prediction horizon of 0.4s with 0.1s intervals. This means that a prediction is made at t=0s,0.1s,0.2s,0.3s,0.4s, explaining the five outputs from the BiLSTM model. Figure A.26 shows those five predictions and the driver's SWT. The main differences between the predictions were found to be near the absolute peaks of the torque-time curves. The absolute peaks are usually the lowest for the predictions at t=0s and highest at t=0.4s. Furthermore, those absolute peaks also happen earlier for t=0s, showing stronger anticipatory behaviour.

This shows the predictions at t=0.4s to be slightly closer to the driver's actual steering inputs, as it is closer to the peaks in the driver's SWT. These characteristics are in accordance with the KPI performance of the model where the accuracy and smoothness metrics increase further into the prediction horizon, as was shown in Chapter 2.

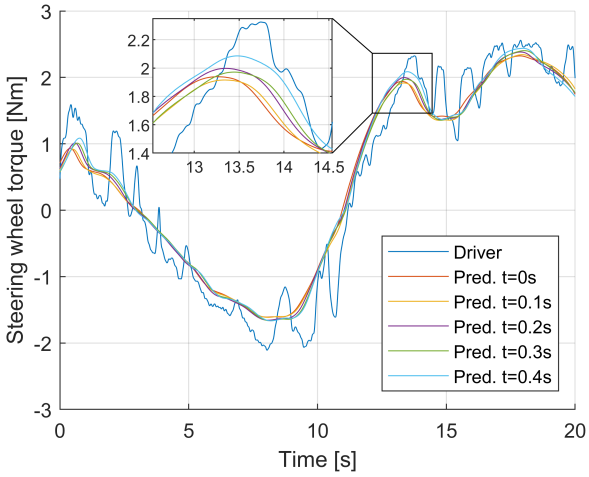

Figure A.26: SWT predictions made by the BiLSTM model over the prediction horizon at t=0s,0.1s,0.2s,0.3s,0.4s.

A.4. CONCLUSION

This appendix has focused on driver modelling using ML. First, an overview of 'classical' ML methods has been given, where HMM has shown to be a popular option. However, a comparison between different ML techniques has shown ANNs to be a promising option for driver modelling.

Subsequently, an overview of ANN implementations for driver modelling has been given, where first the background information was covered of three popular types of ANN. A FANN contains nodes which perform a weighted sum of the inputs and pass it through an activation function. A CNN uses a filter to create a feature map of the inputs, which highlights certain features of the data. A LSTM network uses gates to manage the flow of information, where it is able to store information for its long-term memory. This makes a LSTM network perform very well on sequential data.

These three ANN methods were also used by work related to lane change predictions. The LSTM network appeared to be the most popular for the regression models which predict the TTLC. This is due to LSTM layers extracting the temporal features of the data well. These models already gave an impression of potential architectures.

This appendix also covered ANN models for the lane keeping task directly. However, there were only limited studies available on this topic, with each using different types of ANN. Still, the implementation of a BiLSTM looks very promising as it appeared to outperform a regular LSTM, resulting in a BiLSTM network being developed for the proposed hybrid controller. These studies also gave a good impression of the wide range of

input features that can be used by the model. For example, EMG signals can be used, which would require physical sensors to be placed on the driver, while it would also be fine to only use the sensors already available on the vehicle to collect vehicle and environment data.

Furthermore, models for other tasks were also covered, both driving and non-driving related, where LSTM networks are still popular. These studies also form a great source of inspiration due to the different methodologies used. For example, combining an OA model with a UP model gave promising results for human-centric HG.

Lastly, the KPIs, accuracy and smoothness, used to evaluate the proposed BiLSTM model for the hybrid controller have been defined. Additionally, the proposed BiLSTM model's performance has been discussed in more detail.

B

HYBRID CONTROLLER

This appendix covers the second element of the hybrid controller, the MPC controller. First, a short introduction and background information will be given about MPC, as the MPC design of the hybrid controller has already been defined in Chapter 2. This will then be followed by an overview of methods considered to combine an ANN with a MPC, including the method used for the proposed hybrid controller. This will serve as an overview for potential future work on different configurations of hybrid controllers, which can then be compared to the proposed hybrid controller.

B.1. MODEL PREDICTIVE CONTROL BACKGROUND

MPC is a strong tool for optimising the present control input while considering calculated future events within a time window. As the name suggests, MPC uses a model of the plant for its predictions. This enables the algorithm to have an approximation of the plant's state. For vehicle control, for example, equations describing the dynamics of the vehicle enable the prediction of where the vehicle will be a certain amount of timesteps into the future. A more complex model would result in more accurate predictions, but, as a trade-off, increases computational load.

MPC uses a finite horizon for its predictions, where the time window shifts along with every timestep. This is also why MPC is referred to as Receding Horizon Control, where there are two horizons defined. One is the prediction horizon N_p which gives the number of timesteps to be predicted during each iteration. The other is the control horizon N_c which gives the number of control steps to be optimized in every iteration, where it has to be lower than or equal to N_p. They are often set to the same horizon length N.

The optimization is done using a cost function, where the aim is to minimize the cost over the horizon, while also considering the constraints set. During each control step, iterations are performed to obtain the minimum cost, after which the control input from the first timestep in the horizon is sent to the plant. One timestep later, the new state is measured and the MPC optimisation process is repeated. Figure B.1 shows a scheme portraying what the optimisation process does.

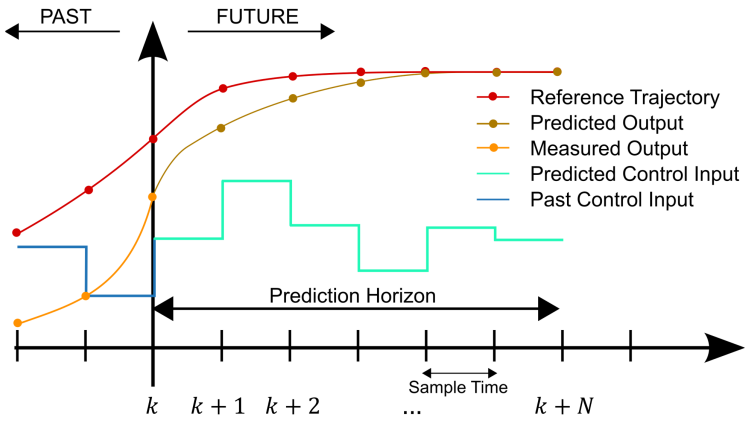

Figure B.1: Optimisation scheme of MPC. Adapted from [13].

MPC has become a popular control option for the automotive industry in recent years. For example, MPC has been used for stability and traction control implementations [75] [76]. Furthermore, it has also been used for lane keeping and obstacle avoidance tasks [77] [78], among many other tasks.

B.2. METHODS

This section covers four different methods to combine an ANN model with MPC. The ANN introduces human modelling into the overall controller. If the controller only used MPC, it would need a complex model to describe the non-linear behaviour a human portrays. The ANN results in simplifying the MPC, where the MPC serves the purpose of optimally controlling the vehicle while considering the constraints. The MPC controller also introduces robustness as it considers the accurate following of the centre of the lane along with the driver's behaviour.

Firstly, the method used for the proposed hybrid controller, as described in Chapter 2, is where the output of the ANN is used as a reference for the MPC, as shown in Figure B.2. Thus, along with the path reference, a reference steering torque is passed onto the MPC. Based on the weights set, the effect of the ANN can be manipulated to a pleasant controller which offers both a good path tracking performance and shows more human-like behaviour. This method has been inspired by [33], where they used a HMM to predict the driver's steering input and used it in the cost function of the MPC controller. However, their approach uses the steering wheel angle and the HMM only makes a prediction for one timestep. Thus, they use a bicycle model to predict the input features for the next timestep with which the HMM makes another prediction. This is repeated until all the steering wheel angle predictions are made over the prediction horizon. This limitation has been overcome by the proposed BiLSTM driver model, which makes a prediction over the whole prediction horizon with one pass.

An advantage of this method is that it uses the ANN prediction during optimisation. It also allows for the ANN to make predictions into the future, which can then be used

as the reference for the prediction horizon. However, the new weight needed to account for the extra reference could make the tuning process more complicated.

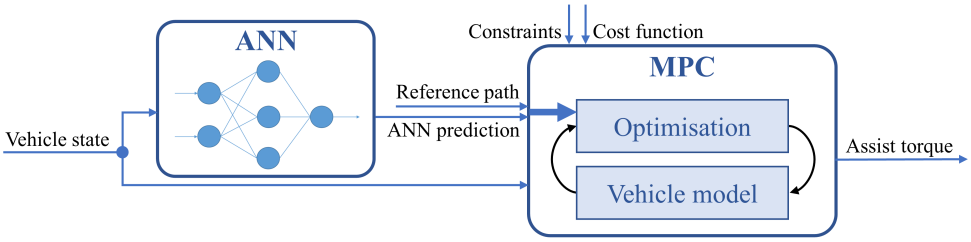

Figure B.2: (Method 1) ANN prediction used as a reference for the MPC controller. This method has been used for the proposed hybrid controller.

Secondly, a relatively simple method would be to use the ANN and MPC in parallel and take the average of both outputs. This is inspired by the work implemented by [11], covered in Section A.2.4, where they take the average of their OA and UP model. Figure B.3 gives a schematic overview of this method. The MPC remains a simple path follower where it tracks the centre of the lane, while the ANN predicts human steering behaviour. It is also possible to use a weighted average which could be manually tuned, or could be based on the uncertainty of the prediction of the ANN. An advantage of this method is that it is relatively easy to implement. However, the main disadvantage is that the ANN prediction is not taken into account during the optimisation steps of the MPC. Thus, the ANN predictions are not considered for control actions in the horizon.

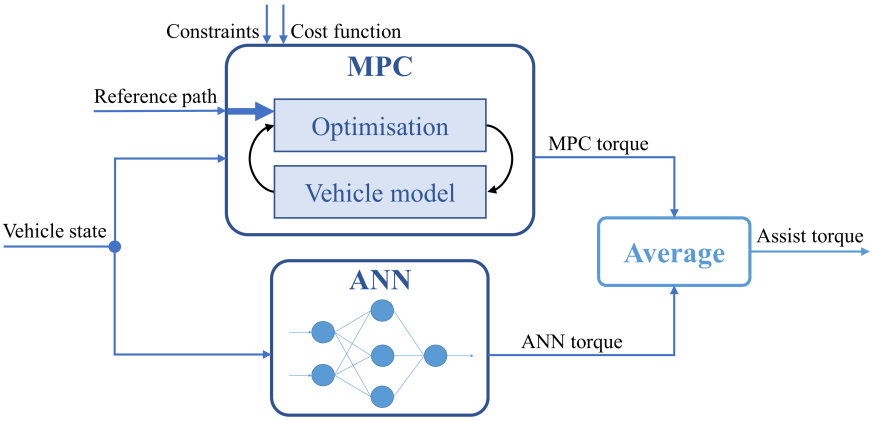

Figure B.3: (Method 2) Taking the average of the outputs from the MPC controller and ANN model as the control input.

Thirdly, a method which also passes its ANN prediction to the MPC, but now as constraints, is inspired by [79], where they adapt the constraints for the MPC based on the prediction of their LSTM model. Their LSTM model predicts the motions of surrounding vehicles, which then changes the boundaries of where the ego-vehicle could go. A similar method could be applied for the LKA controller, where constraints could be set on

the control input from the MPC based on a range predicted by the ANN model, as shown in Figure B.4. This would force the MPC to find a control input closer to the ANN prediction, making it feel more human-like. An advantage of this method is that the control input is optimized with the ANN prediction in mind. However, it could result in more iterations to meet the new constraints, or even lead to instability if the constraints cannot be met.

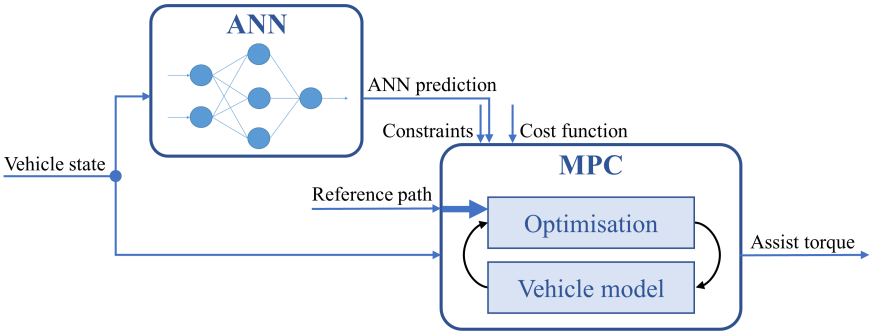

Figure B.4: (Method 3) ANN prediction used to adapt the constraints of the MPC controller.

Finally, it would also be an option to use the ANN as the plant model, or a part of it. For example, an ACC with MPC has been proposed by [80], where they use an ANN to predict the speed of the vehicle after a throttle or brake input. Thus, the ANN learns the vehicle characteristics as a way to model the plant. Their reasoning is that a mathematical formulation of the engine mapping is complicated to develop, where an ANN can learn it based on data. A similar method has also been implemented in the chemical industry [81], where they propose an ANN to model the reactor, which is being controlled by MPC.

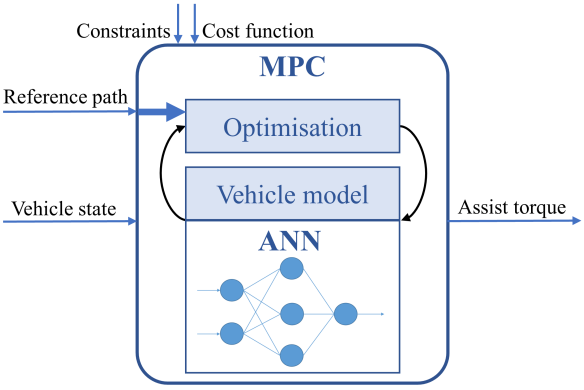

Figure B.5: (Method 4) ANN used to model the plant within the MPC controller.

This could also be done for the human-centric LKA, where a part of the plant modelling could be done by the ANN, as shown in Figure B.5. Specifically, the ANN could

only focus on driver modelling and the vehicle dynamics equations would stay. However, it is also possible to cover both the driver modelling and the vehicle dynamics with the ANN. An advantage of this method is that it is a direct part of the optimisation loop, where the ANN makes predictions for the horizon. A disadvantage is the data needed for the ANN, especially when it also covers the vehicle dynamics, becomes more complex. Besides, the computational load increases significantly due to the ANN model being run significantly more often, which could lead to a compromised ANN model.

B.3. FORCESPRO SETTINGS

This section gives the settings used for the MPC solver created with the FORCESPRO [82] software. FORCESPRO is a licensed software for generating numerical solvers for optimisation problems, mainly MPC. The solvers can be generated from MATLAB and Python [83], where the optimisation problem is described. The solver can then be run in MATLAB, Simulink, C/C++ and Python. FORCESPRO credits itself for being able to generate fast solvers, making it the software solution used for the proposed hybrid controller as it has to run real-time in the driving simulator.

Table B.1 gives the MPC settings, as was already discussed in Chapter 2. The sampling time of the MPC controller $T_{s,c}$ is set at 100Hz as to match the standard CAN broadcast frequency on driving simulators, while the sampling time of the simulation $T_{s,sim}$, the non-linear plant, is set at 1000Hz. The prediction and control horizons are set to the same length of 40 timesteps (0.4s).

Furthermore, Table B.2 gives the most relevant code options set for the FORCESPRO software, while the rest of the code options was set to the default option. All the MPC and FORCESPRO settings were tuned through trial-and-error to find a trade-off between computational load (real-time), and system performance and stability.

Table B.1: MPC settings.

Setting	Value	Description
$T_{s,c}$	$1 \cdot 10^{-2} s$	Sampling time of the controller
$T_{s,plant}$	$1 \cdot 10^{-3} s$	Sampling time of the plant (IPG CarMaker)
N_p	40	Prediction horizon
N_c	40	Control horizon
n_{var}	11	Number of variables
n_{eq}	10	Number of equality constraints
n_{par}	13	Number of runtime parameters

B.4. SOFTWARE ARCHITECTURE

This section covers the software architecture used to develop and operate the proposed hybrid controller. Different programming languages and software were used to work in cohesion, specifically, Python [83], MATLAB, Simulink, FORCESPRO [82] and IPG CarMaker. Figure B.6 shows which elements of the hybrid controller use which software, where there are also overlapping software.

Table B.2: FORCESPRO settings.

Code option	Setting	Description
maxit	200	Maximum number of iterations
nlp.integrator.type	ERK4	Type of integrator for the continuous dynamics
nlp.integrator.Ts	$1 \cdot 10^{-2} s$	Discretization interval of the integrator
nlp.integrator.nodes	5	Number of integrator nodes per interval
nlp.hessian_approximation	gauss-newton	Defining the method to compute the Hessian of the Lagrangian function
nlp.ad_tool	casadi-351	Defining the algorithmic differentiation tool
optlevel	3	Compiler optimization level
solvemethod	SQP_NLP	Defining the solve method used by the generated solver
sqp_nlp.use_line_search	0	Internal line search

First, the ANN was developed in Python using the library PyTorch [84]. The reason for using Python was the popularity and community support for ML related programming, also creating more room for future developments such as user adaptation. Initially, the TensorFlow [85] library was used to develop the ANN, but it was found that it introduced issues when cooperating with MATLAB and Simulink, which were eliminated with the use of PyTorch.

The reason that these Python libraries had to cooperate with MATLAB and Simulink is due to the proposed hybrid controller having to work in Simulink, as TME's high-fidelity steering model [86] for the simulation was implemented in Simulink and as the communication with IPG CarMaker also went through Simulink. To accommodate the ANN model in Simulink, a Python environment was created within MATLAB. The model would first be saved as a PT file after training. This model then gets loaded into MATLAB's Python environment. It was not possible to load the model directly into Simulink by the code generator, thus the 'coder.extrinsic' option was used to load the model through the MATLAB engine within Simulink.

Loading the model was done through object-oriented programming, where a class was created within Python. During the initialisation of the class, the model gets loaded resulting in the model only getting loaded once, reducing the computational load. Furthermore, the class contains a function which passes the input data to the model and gets the SWT predictions from the model. Summarising this process, Simulink passes the input features for the ANN model coming from IPG CarMaker to the MATLAB engine, which passes it onto the Python codes, where the input features are fed into the ANN model. The predictions are then sent back to the MATLAB engine and back into Simulink. It was tested that this process had an insignificant computational load. The

prediction is then used by the MPC solver.

The optimisation problem for the MPC solver is defined in MATLAB using FORCESPRO. FORCESPRO then generates the solver based on the problem formulation, described in Chapter 2, and the settings, mentioned in Section B.3. The solver is generated on the servers of FORCESPRO and sent back to the local computer, as it is a licensed product. This introduced issues as TME's firewall blocked any communications with the FORCESPRO servers. This was eventually solved by avoiding the certificate verification of the servers.

The generated MPC solver is then used in Simulink, where it receives the ANN model's SWT predictions along with the other input signals. The optimised assist torque is then passed into the high-fidelity steering model, which simulates the steering dynamics. Subsequently, steering signals are then sent to IPG CarMaker where the vehicle dynamics and experiment scenario are simulated. To have improved graphics, rFpro is used to project the driving scenario onto the screen of the driving simulator. Finally, the vehicle states are retrieved from IPG CarMaker to repeat the controller loop.

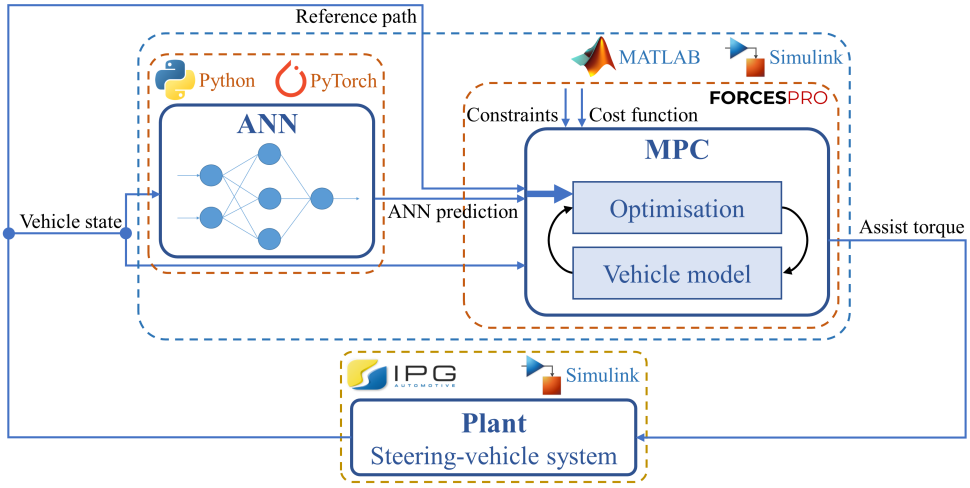

Figure B.6: Software architecture of the proposed hybrid controller.

B.5. CONCLUSION

In conclusion, this appendix gave a brief technical introduction to MPC, where its strength and popularity became clear. It also covered four potential methods for combining an ANN and MPC.

The first method, where the ANN prediction is used as a reference, has been used for the proposed hybrid controller as it was considered to be the most viable method. It allows for a modular implementation, as the ANN and MPC can be considered separate systems during development, but the ANN prediction is still taken into account during the optimisation steps of the MPC. It is also easier to ensure stability and it provides more room for tuning due to the weights when compared to the third method, where the ANN

predictions would be used as constraints.

Besides, the computational load for the first method is lower when compared to the fourth method, where the ANN models the plant. This is due to the ANN only making one prediction for every control step in the first method, while it would have to make a prediction for every optimisation step in the fourth method. However, this does mean that the ANN in the first method would have to make predictions into the future which fit in the horizon of the MPC, as has been done for the proposed hybrid controller.

Additionally, the settings for the FORCESPRO solver of the proposed hybrid controller have been given. Lastly, the software architecture of the proposed hybrid controller has been explained, where a combination of Python, MATLAB, Simulink, FORCESPRO and IPG CarMaker has been used.

C

RESULTS AND DISCUSSION

This Appendix covers the subjective and objective results from the driving simulator experiment in more detail, specifically covering more figures and tables to show the performances of the controllers. Manual driving performance will also be covered briefly.

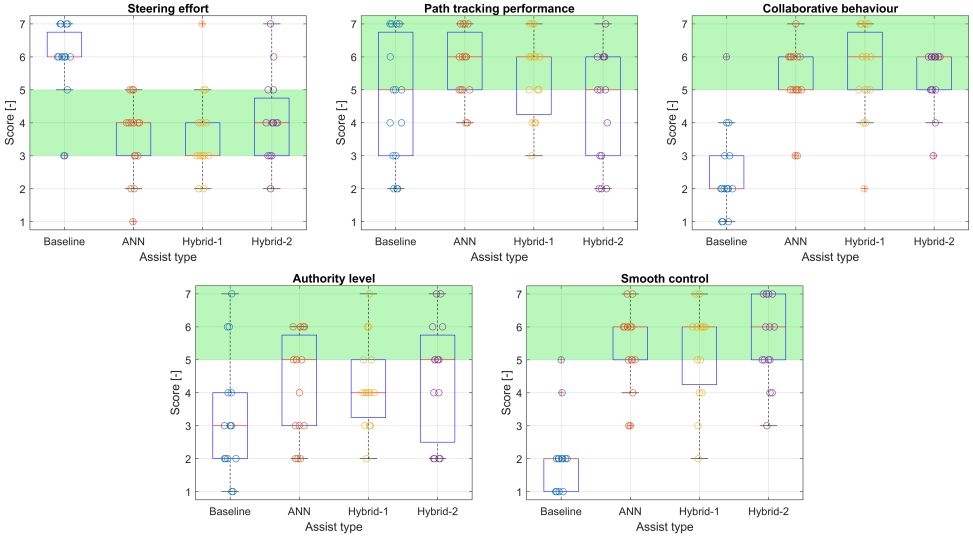

Figure C.1: Boxplot of the subjective results and the questionnaire scores given by the participants.

C.1. SUBJECTIVE EVALUATION

Figure C.1 shows the boxplots obtained from the questionnaire results, including the scores given by the participants. The plots show the variance in the participants' answers, where they had significantly different opinions in terms of the path tracking performance and authority level, showcasing the contrasting preferences of the drivers. For

the collaborative behaviour and smooth control questions, the participants appear to be more in line.

To get a better image of the preferences of the participants, they were asked to rank the four assists. Figure C.2 shows how many participants put each assist in the respective rank. For example, 7 participants put the Hybrid-2 controller in first place, while 12 participants put the baseline LKA in last place. The Hybrid-2 controller has also been placed in the third spot the most, while the Hybrid-1 and ANN assists stand out in second place. This partly shows the preference of the participants in terms of the level of control authority, where participants who prefer the controller to have more authority tend to put the Hybrid-2 controller in third place.

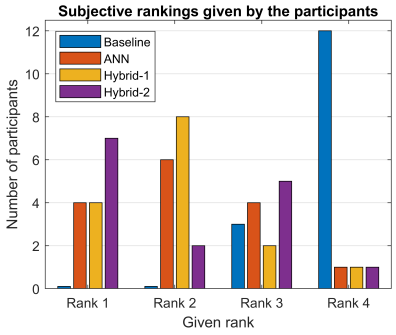

Figure C.2: Rankings given by the participants based on their preferences.

Table C.1 shows the mean subjective scores and the SD, as was already discussed in Chapter 2. Additionally, Table C.3 shows the results of the paired t-test for the subjective results, with the definition of the abbreviations shown in Table C.2. The paired t-test demonstrates the statistical significance of the baseline LKA paired with the other controllers in terms of the steering effort, collaborative behaviour and smooth control. This highlights the difference in scores between the baseline and the proposed controllers for these metrics.

Table C.1: Mean subjective results for the KPIs with the SD in parentheses.

Metric	Baseline	ANN	Hybrid-1	Hybrid-2
Steering effort	5.80 (1.26)	3.53 (1.19)	3.67 (1.29)	4.00 (1.36)
Tracking performance	4.60 (1.92)	5.73 (1.03)	5.40 (1.24)	4.53 (1.73)
Collaborative behaviour	2.47 (1.36)	5.27 (1.10)	5.47 (1.41)	5.40 (0.91)
Authority level	3.27 (1.83)	4.20 (1.57)	4.27 (1.33)	4.47 (1.77)
Smooth control	2.00 (1.13)	5.33 (1.23)	5.27 (1.44)	5.60 (1.30)

Table C.2: Symbols for the pairs used for the paired t-test.

Symbol	P-value comparison
A	Baseline ↔ ANN
B	Baseline ↔ Hybrid-1
C	Baseline ↔ Hybrid-2
D	ANN ↔ Hybrid-1
E	ANN ↔ Hybrid-2
F	Hybrid-1 ↔ Hybrid-2

Table C.3: Paired t-test p-values for the subjective results.

Metric	A	B	C	D	E	F
Steering effort	<0.01	<0.01	<0.01	0.72	0.29	0.49
Tracking performance	0.06	0.25	0.92	0.24	0.01	0.13
Collaborative behaviour	<0.01	<0.01	<0.01	0.60	0.74	0.87
Authority level	0.14	0.06	0.04	0.88	0.72	0.66
Smooth control	<0.01	<0.01	<0.01	0.84	0.50	0.33

C.2. OBJECTIVE EVALUATION

This section covers the objective evaluation of the assists. First, the KPIs will be defined. Second, the results will be discussed in more detail with additional figures, where a comparison will also be made with no LKA on (manual driving).

C.2.1. KEY PERFORMANCE INDICATORS

This section defines the 15 KPIs used to evaluate the objective results, where 13 of the KPIs are defined by [31]. The KPIs can be divided under the following five main categories: Steering effort, path tracking performance, collaborative behaviour, level of control authority and smooth driving.

STEERING EFFORT

1) Driver effort: Steering effort put in by the driver over the scenario time duration T, defined as

$$se_{T_{driver}} = \int_0^T \mathbf{T}_{driver}^2 dt \tag{C.1}$$

where $\mathbf{T}_{driver}$ denotes the SWT applied by the driver.

2) Controller effort: Steering effort put in by the assist torque of the controller, defined as

$$se_{T_{contr}} = \int_0^T \mathbf{T}_{contr}^2 dt \tag{C.2}$$

where $\mathbf{T}_{contr}$ denotes the SWT applied by the assist.

PATH TRACKING PERFORMANCE

1) Lateral RMSE: Root Mean Square Error between the lateral position of the front axle's centre and the lane centre, defined as

$$RMSE_y = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_{y,i}^2} \tag{C.3}$$

where N denotes the total number of datapoints and e_y the lateral error.

2) Maximum lateral error: Absolute maximum lateral position error, defined as

$$e_{y,max} = max(|\mathbf{e}_y|). \tag{C.4}$$

3) Mean lateral error: Defines as

$$\bar{e}_y = \mu_{\mathbf{e}_y} = \frac{1}{N}\sum_{i=1}^{N} e_{y,i}. \tag{C.5}$$

4) SD lateral error: Defined as

$$\sigma_{\mathbf{e}_y} = SD\left(\mathbf{e}_y\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(e_{y,i} - \mu_{\mathbf{e}_y}\right)^2}. \tag{C.6}$$

COLLABORATIVE BEHAVIOUR

1) Collaborative ratio: Ratio between the time when the driver and controller SWT have the same sign and the total scenario time, defined as

$$r_{co} = \frac{1}{T}\int_0^T sign\left(\mathbf{T}_{driver}\mathbf{T}_{contr}\right) dt \quad \text{if} \quad \mathbf{T}_{driver}\mathbf{T}_{contr} \geq 0. \tag{C.7}$$

2) Intrusiveness ratio: Ratio between the time when the driver and controller SWT have the opposite sign and the total scenario time, defined as

$$r_{int} = \frac{1}{T}\int_0^T sign\left(\mathbf{T}_{driver}\mathbf{T}_{contr}\right) dt \quad \text{if} \quad \mathbf{T}_{driver}\mathbf{T}_{contr} < 0. \tag{C.8}$$

3) Resistance ratio [87]: Ratio between the time when the driver and controller SWT have the opposite sign and the total scenario time, if the driver SWT is bigger than the controller SWT, defined as

$$r_{re} = \frac{1}{T}\int_0^T sign\left(\mathbf{T}_{driver}\mathbf{T}_{contr}\right) dt \quad \text{if} \quad \mathbf{T}_{driver}\mathbf{T}_{contr} < 0 \text{ and } \mathbf{T}_{driver} > \mathbf{T}_{contr}. \tag{C.9}$$

4) Contradiction ratio [87]: Ratio between the time when the driver and controller SWT have the opposite sign and the total scenario time, if the driver SWT is smaller than the controller SWT, defined as

$$r_{cont} = \frac{1}{T}\int_0^T sign\left(\mathbf{T}_{driver}\mathbf{T}_{contr}\right) dt \quad \text{if } \mathbf{T}_{driver}\mathbf{T}_{contr} < 0 \text{ and } \mathbf{T}_{driver} < \mathbf{T}_{contr}. \tag{C.10}$$

5) Coherence [88]: Cosine of the angles formed by the driver and controller SWT, where a positive value indicates collaborative behaviour, defined as

$$\gamma = \frac{\int_0^T \mathbf{T}_{driver}\mathbf{T}_{contr}\,dt}{\sqrt{\int_0^T \mathbf{T}_{driver}^2\,dt \int_0^T \mathbf{T}_{contr}^2\,dt}}. \tag{C.11}$$

LEVEL OF CONTROL AUTHORITY

1) Level of control authority [88]: Ratio between the controller and driver steering effort, defined as

$$T_{auth} = \frac{se_{T_{contr}}}{se_{T_{driver}}}. \tag{C.12}$$

SMOOTH DRIVING

1) Steering Reversal Rate (SRR) [89]: Number of steering reversals, larger than the gap value $\theta_{sw,min}$, per minute. The steering wheel angle and velocity data are filtered with a second-order Butterworth filter with the cut-off frequency set at 0.6Hz, as to lower the high-frequency noise. The number of steering reversals $n_{reversal}$ is determined by the number of times where

$$|\theta_{sw}(t_1) - \theta_{sw}(t_2)| \geq \theta_{sw,min} = 3 \text{ deg} \tag{C.13}$$

where t_1 and t_2 are the timesteps of two consecutive moments where the steering wheel velocity is equal to zero. The SRR is defined as

$$SRR = \frac{n_{reversal}}{T} \cdot 60. \tag{C.14}$$

2) Driver smoothness: The SD of the driver SWT's time derivative, as was described in Section A.3.1, defined as

$$SM_{\mathbf{T}_{driver}} = SD\left(\dot{\mathbf{T}}_{driver}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\dot{T}_{i,driver} - \mu_{\dot{\mathbf{T}}_{driver}}\right)^2}. \tag{C.15}$$

3) Controller smoothness: The SD of the controller SWT's time derivative, defined as

$$SM_{\mathbf{T}_{contr}} = SD\left(\dot{\mathbf{T}}_{contr}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\dot{T}_{i,contr} - \mu_{\dot{\mathbf{T}}_{contr}}\right)^2}. \tag{C.16}$$

C.2.2. RESULTS

This section covers the results obtained from the driving simulator experiment based on the KPIs. It should be noted that for certain KPIs a representative value cannot be obtained for the baseline LKA, as this controller is fundamentally different where the assist torque values are determined to actuate on the steering column instead of the steering wheel, resulting in significantly higher torque values. Thus, those KPI values are omitted from the results.

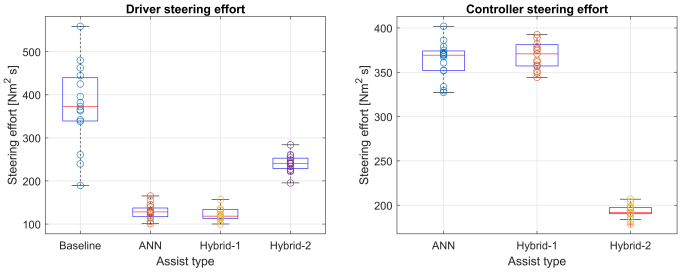

Figure C.3: Steering effort done by the driver and controller.

Figure C.3 shows the boxplots of the steering effort put in by the driver and the controller. The baseline controller shows a significantly greater variance, indicating that certain participants managed to work better with the assist than others as they needed to put in less effort to complete the lane following task.

Furthermore, the path tracking performance boxplots are shown in Figure C.4, where the lateral error is the distance between the centre of the front axle and the centre of the lane. The baseline LKA shows a higher variance in terms of maximum lateral error. Overall, the mean lateral error shows a tendency of the participants to drive more on the left side of the lane, which is potentially caused by the path having more left turns. One participant also had the tendency to drive mainly on the left side of the lane, especially with the ANN controller, resulting in higher lateral error values.

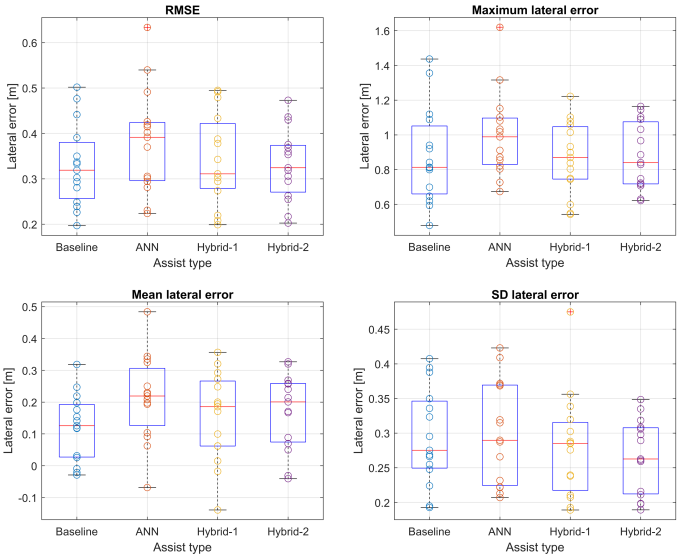

Figure C.4: Path tracking performance of the four assists.

Figure C.5 shows the collaborative behaviour boxplots. As discussed in Chapter 2, the proposed controllers significantly improve the collaborative behaviour of the driver with

the assist compared to the baseline LKA. Specifically, the Hybrid-2 controller scores the best, except for the resistance ratio where the Hybrid-1 controller shows the best results.

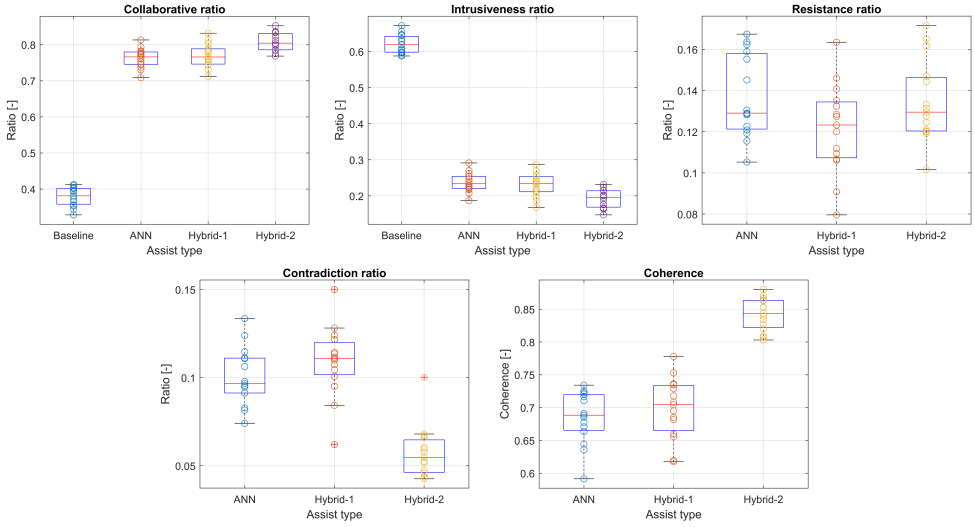

Figure C.5: Collaborative behaviour KPIs.

The level of control authority boxplot is shown in Figure C.6, where the differences in control authority are as expected due to the authority levels set in the controllers.

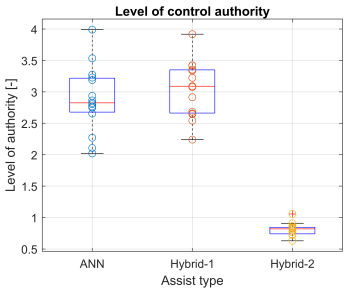

Figure C.6: Level of control authority of the assists.

Finally, the smooth driving boxplots are shown in figure C.7. These show that the baseline controller is less smooth as it results in a higher SRR and driver smoothness metric. Between the proposed controllers, the Hybrid-2 controller causes the driver to be less smooth due to the lower control authority, but does result in a smoother controller.

Moreover, it is also interesting to compare the participants' performance with the assists on and without any assist on (manual driving). In this case, only 7 out of the 15 KPIs can be used for comparison. Figure C.8 shows the boxplots of the driver steering effort, SRR and driver smoothness. As was expected, manual driving results in a higher driver steering effort. Thus, the assists do lower the physical workload. Furthermore, the

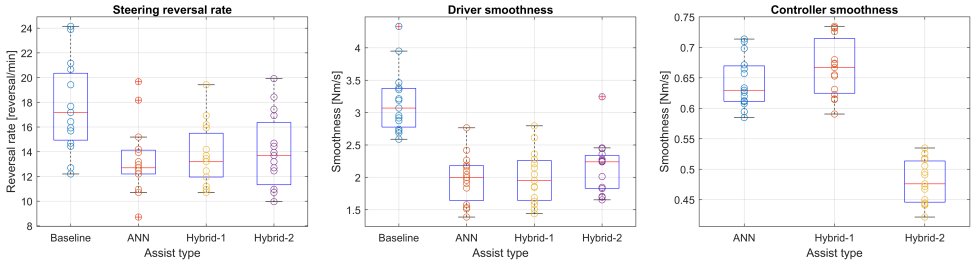

Figure C.7: Smooth driving KPIs.

proposed controllers do improve the SRR, while the baseline LKA causes the driver to be less smooth than when driving manually, indicating poor collaborative behaviour.

The path tracking performance boxplots are shown in Figure C.9. Overall, the baseline, Hybrid-1 and Hybrid-2 controllers result in a slight improvement in path tracking performance. The ANN controller performs worse, which is due to the lack of the MPC controller-in-the-loop, which would enforce better path tracking as is the case with the hybrid controllers.

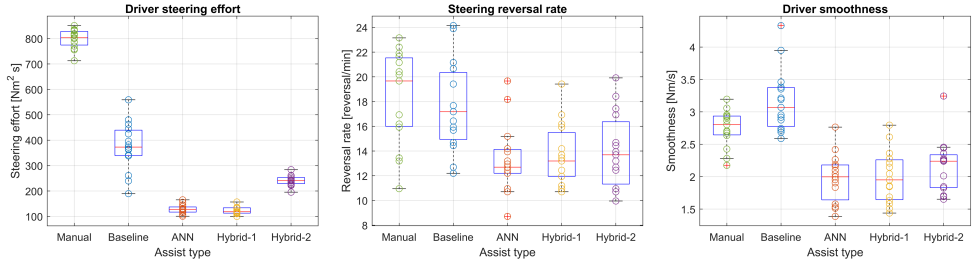

Figure C.8: Driver steering effort, SRR and driver smoothness for manual driving and the four assists.

Finally, Table C.4 gives the mean values and the SD of the objective results, also including the manual driving values. As mentioned earlier, KPI values which cannot be determined for the baseline controller and for manual driving have been omitted. The paired t-test results are shown in Table C.5, with the abbreviations defined earlier in Table C.2. The light-grey shaded fields highlight the pairs which demonstrate statistical significance for that specific KPI.

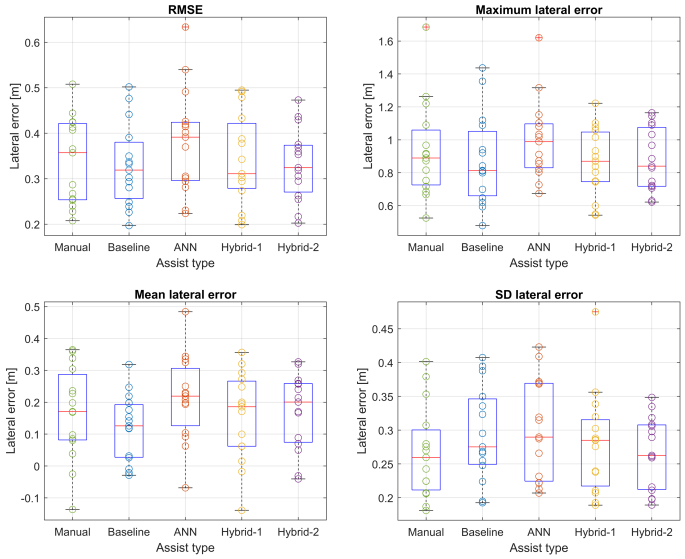

Figure C.9: Path tracking performance for manual driving and the four assists.

Table C.4: Mean objective results for the KPIs with the SD in parentheses.

Metric	Manual	Baseline	ANN	Hybrid-1	Hybrid-2
Driver effort [Nm^2 s]	799.30 (36.87)	374.71 (95.88)	128.88 (17.22)	122.68 (13.95)	240.65 (20.42)
Controller effort [Nm^2 s]	-	-	363.15 (21.05)	368.71 (14.94)	193.28 (7.19)
Lateral RMSE [m]	0.34 (0.10)	0.33 (0.09)	0.38 (0.11)	0.34 (0.10)	0.33 (0.08)
Maximum e_y [m]	0.93 (0.29)	0.88 (0.27)	1.00 (0.24)	0.87 (0.21)	0.88 (0.19)
Mean e_y [m]	0.17 (0.15)	0.12 (0.10)	0.21 (0.13)	0.15 (0.14)	0.17 (0.12)
SD e_y [m]	0.27 (0.07)	0.29 (0.07)	0.30 (0.07)	0.28 (0.08)	0.27 (0.05)
Collaborative ratio [-]	-	0.38 (0.03)	0.76 (0.03)	0.77 (0.03)	0.81 (0.03)
Intrusiveness ratio [-]	-	0.62 (0.03)	0.24 (0.03)	0.23 (0.03)	0.19 (0.03)
Resistance ratio [-]	-	-	0.14 (0.02)	0.12 (0.02)	0.13 (0.02)
Contradiction ratio [-]	-	-	0.10 (0.02)	0.11 (0.02)	0.06 (0.01)
Coherence [-]	-	-	0.69 (0.04)	0.70 (0.05)	0.84 (0.02)
Level of control authority [-]	-	-	2.88 (0.52)	3.05 (0.44)	0.81 (0.10)
Steering reversal rate [reversal/min]	18.22 (3.81)	17.56 (3.66)	13.33 (2.76)	13.69 (2.47)	14.01 (3.02)
Driver smoothness [Nm/s]	2.75 (0.28)	3.16 (0.48)	1.98 (0.37)	2.00 (0.40)	2.17 (0.41)
Controller smoothness [Nm/s]	-	-	0.64 (0.04)	0.67 (0.05)	0.48 (0.04)

Table C.5: Paired t-test p-values for the objective results.

Metric	A	B	C	D	E	F
Driver effort	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
Controller eff.	-	-	-	0.07	<0.01	<0.01
Lateral RMSE	<0.01	0.36	0.78	0.01	0.01	0.45
Maximum e_y	0.02	0.86	0.97	0.05	0.01	0.73
Mean e_y	<0.01	0.07	0.01	0.04	0.01	0.51
SD e_y	0.28	0.29	0.02	0.10	0.01	0.30
Collaborative	<0.01	<0.01	<0.01	0.36	<0.01	<0.01
Intrusiveness	<0.01	<0.01	<0.01	0.36	<0.01	<0.01
Resistance	-	-	-	0.02	0.69	0.05
Contradiction	-	-	-	0.09	<0.01	<0.01
Coherence	-	-	-	0.14	<0.01	<0.01
Authority	-	-	-	0.01	<0.01	<0.01
Steering RR	<0.01	<0.01	<0.01	0.57	0.33	0.57
Driver smooth.	<0.01	<0.01	<0.01	0.76	0.02	0.03
Control. smooth.	-	-	-	<0.01	<0.01	<0.01

D

FUTURE WORK

Chapter 2 already discussed the recommended future work on the topic of human-centric haptic control using a hybrid controller. This chapter gives more in-depth information regarding one of the recommended points, namely User Adaptation UA through online learning or meta-learning. This overview could be used as the basis of future work regarding UA, with meta-learning being an interesting option.

D.1. USER ADAPTATION

Every human is unique and portrays individual characteristic traits. For example, certain drivers might drive more on the right side of the lane or cut the corners more during a lane keeping task. Overall, every driver tries to find his/her ideal balance between lane keeping performance and maintaining a comfortable workload. However, having an ANN trained on a large dataset with different drivers, results in a generalized model which might not capture the small differences between each driver. To make the model more tailored to a specific driver, UA could be used to slightly adapt the generalized model to a specific driver. This section covers two UA methods, namely, online learning and meta-learning.

Online learning [90] [91] is a common method in ML, where the parameters of the model are still updated while being deployed. Normally, a ML model would be trained offline on all the available data at the time with backpropagation, where the optimizer tries to lower the loss function. For offline learning, the model would then be ready and be deployed without being changed, until enough new data has been collected to redo the whole process of offline training. However, for online training, the pre-trained model is still being optimized by the new data being collected during operation. This is again done using backpropagation and a loss function, but with only one datapoint or a small batch of datapoints. This enables the online learning model to converge more towards the new data. However, it is important that the correct output should be collected or measured during operation, as that will be used to quantify the error in the model.

There is a wide range of methods and implementations for online learning. An overview of a few methods and the main challenges with online learning is given by [92]. Unfortu-

nately, there are not many online learning implementations for driver modelling. However, it is still popular in other domains. For example, in user typing input predictions [93], where they try to predict the words the users want to type based on their gaze. Another example is the use of online learning for dialogue systems [94], where the system tries to be more personal.

In recent years, meta-learning [95] has started to become popular for model adaptation. Meta-learning is also referred to as "learning to learn", where it tries to adapt itself to a new context with limited new data. It essentially tries to capture the learning pattern of the actual model, which enables it to understand the model better. This results in only limited new data being sufficient to accurately adapt the model. For example, the general model will be trained on a large dataset with different drivers performing the lane keeping task. When a new driver is going to drive, only a few minutes of driving is already enough data to personalise the model to the new driver. This results in better predictions for that specific driver.

There is a wide range of meta-learning methods. A popular approach is the Model-Agnostic Meta-Learning (MAML) [96] method. The previously discussed study by [11] used the MAML method for their UA in their virtual air hockey game. They did this as their previous work demonstrated that meta-learning provided fast UA with very limited data of a new user, outperforming a typical ANN method [97].

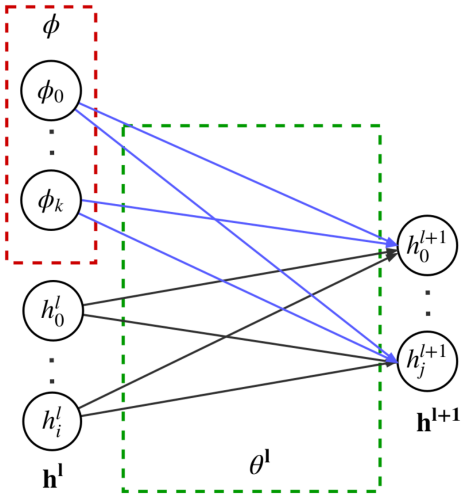

Figure D.1: CAVIA method with the context parameters shown in the red box [14].

However, the MAML method updates all the parameters of the network, making it more prone to overfitting the new data and being computationally heavy. That is why the Fast Context Adaptation Via Meta-Learning (CAVIA) method has been proposed by [14], where only a preselected set of parameters are updated. This set of parameters is referred to as the 'context parameters', as it quickly adapts the network to the new task or user, while the other parameters provide a generalized performance across many tasks

or users, thus staying constant after the initial training. Figure D.1 shows the basic idea behind the CAVIA method, where the red box contains the context parameters. The general parameters, shown in the green box, are trained like a normal ANN during the training phase with a large dataset containing different users. They remain constant during operation and only the context parameters are updated. The study by [14] showed that the CAVIA method outperformed the MAML approach.

Furthermore, a new meta-learning algorithm has been proposed by [15], which was inspired by the CAVIA method. They have implemented a second ANN which predicts the initialisation for user-specific context parameters. Figure D.2 shows their method, where the initialisation network determines the context parameters based on the new user. The context parameters are then used together with the constant general parameters to make the predictions.

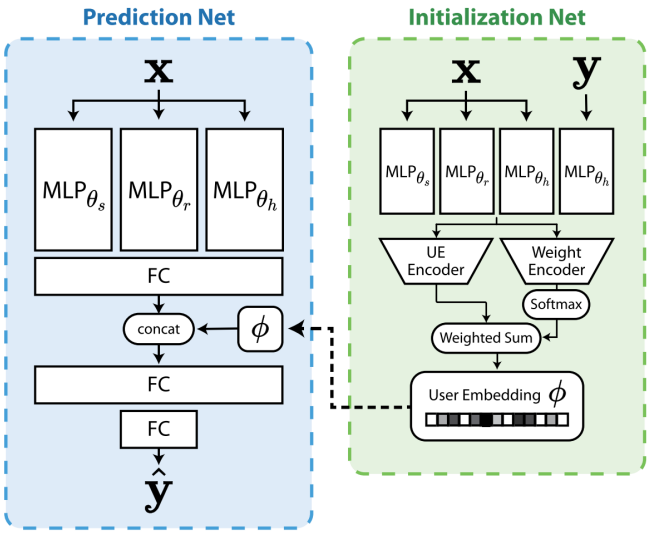

Figure D.2: Meta-learning method developed by [15].

In their experiments, they used the virtual air hockey game from their previous work in [11]. They compared two non-meta-learning baselines, where one was a fixed ANN and the other a fine-tuned ANN with a few gradient steps based on the new user. They also used five meta-learning algorithms as baselines, including the MAML and CAVIA methods. The other three methods are Reptile [98], MAML with Model Regression Network [99] and ANIL [100], which are all inspired by the MAML method. The results showed that all the meta-learning models outperformed the non-meta-learning models for user-specific predictions. The CAVIA method outperformed the other baseline meta-learning models. However, the proposed meta-learning method performed the best with the lowest MSE. This shows that their new method works best for fast UA using meta-learning.

Unfortunately, there were no papers found with a comparison between online learning and meta-learning. However, an online meta-learning method has been proposed

by [101], where the online learning part gradually adapts the network to the task at hand using new datapoints coming in, while the meta-learning part adapts the model to a new task using batches of new data. Nonetheless, its performance has not been tested against the previously mentioned meta-learning methods.

BIBLIOGRAPHY

[1] K. Okamoto and P. Tsiotras, "Data-driven human driver lateral control models for developing haptic-shared control advanced driver assist systems," *Robotics and Autonomous Systems*, vol. 114, pp. 155–171, 2019.

[2] H. Wei, W. Ross, S. Varisco, P. Krief, and S. Ferrari, "Modeling of human driver behavior via receding horizon and artificial neural network controllers," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 6778–6785.

[3] K. Phil, "Matlab deep learning with machine learning, neural networks and artificial intelligence," *Apress, New York*, 2017.

[4] C. Olah, "Understanding LSTM Networks," August 2015, [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: Sept. 5, 2022].

[5] P. Verzelli, C. Alippi, and L. Livi, "Echo state networks with self-normalizing activations on the hyper-sphere," *Scientific reports*, vol. 9, no. 1, pp. 1–14, 2019.

[6] F. Wirthmüller, M. Klimke, J. Schlechtriemen, J. Hipp, and M. Reichert, "Predicting the time until a vehicle changes the lane using lstm-based recurrent neural networks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2357–2364, 2021.

[7] H. Q. Dang, J. Fürnkranz, A. Biedermann, and M. Hoepfl, "Time-to-lane-change prediction with deep learning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–7.

[8] H. Lee, H. Kim, and S. Choi, "Human driving skill modeling using neural networks for haptic assistance in realistic virtual environments," *arXiv preprint arXiv:1809.04549*, 2018.

[9] I. K. Ihianle, A. O. Nwajana, S. H. Ebenuwa, R. I. Otuka, K. Owa, and M. O. Orisatoki, "A deep learning approach for human activities recognition from multimodal sensing devices," *IEEE Access*, vol. 8, pp. 179 028–179 038, 2020.

[10] Y. Xing, C. Lv, Y. Liu, Y. Zhao, D. Cao, and S. Kawahara, "Hybrid-learning-based driver steering intention prediction using neuromuscular dynamics," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 1750–1761, 2021.

[11] H.-S. Moon and J. Seo, "Optimal action-based or user prediction-based haptic guidance: Can you do even better?" in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–12.

[12] R. van Wijk, A. M. R. Lazcano, X. C. Akutain, and B. Shyrokau, "Data-driven steering torque behaviour modelling with hidden markov models," *IFAC-PapersOnLine*, vol. 55, no. 29, pp. 31–36, 2022.

[13] M. Behrendt, "A basic working principle of Model Predictive Control," October 2009, [Online]. Available: https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg. [Accessed: Nov. 24, 2022].

[14] L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson, "Fast context adaptation via meta-learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7693–7702.

[15] H.-S. Moon and J. Seo, "Fast user adaptation for human motion prediction in physical human–robot interaction," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 120–127, 2021.

[16] J. D. Lee, "Fifty years of driving safety research," *Human factors*, vol. 50, no. 3, pp. 521–528, 2008.

[17] G. H. Walker, N. A. Stanton, and M. S. Young, "Where is computing driving cars?" *International Journal of Human-Computer Interaction*, vol. 13, no. 2, pp. 203–229, 2001.

[18] L. Yang, Y. Yang, G. Wu, X. Zhao, S. Fang, X. Liao, R. Wang, and M. Zhang, "A systematic review of autonomous emergency braking system: impact factor, technology, and performance evaluation," *Journal of advanced transportation*, vol. 2022, 2022.

[19] R. Utriainen, M. Pöllänen, and H. Liimatainen, "The safety potential of lane keeping assistance and possible actions to improve the potential," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 556–564, 2020.

[20] SAE International, "SAE levels of driving automation refined for clarity and international audience," May 2021, [Online]. Available: https://www.sae.org/blog/sae-j3016-update. [Accessed: Aug. 15, 2022].

[21] Z. Lu, R. Happee, C. D. Cabrall, M. Kyriakidis, and J. C. de Winter, "Human factors of transitions in automated driving: A general framework and literature survey," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 43, pp. 183–198, 2016.

[22] P. Bazilinskyy, S. M. Petermeijer, V. Petrovych, D. Dodou, and J. C. de Winter, "Take-over requests in highly automated driving: A crowdsourcing survey on auditory, vibrotactile, and visual displays," *Transportation research part F: traffic psychology and behaviour*, vol. 56, pp. 82–98, 2018.

[23] F. O. Flemisch, K. Bengler, H. Bubb, H. Winner, and R. Bruder, "Towards cooperative guidance and control of highly automated vehicles: H-mode and conduct-by-wire," *Ergonomics*, vol. 57, no. 3, pp. 343–360, 2014.

[24] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: smoothly shifting control authority?" *Cognition, Technology & Work*, vol. 14, no. 1, pp. 19–28, 2012.

[25] M. Mulder, D. A. Abbink, and E. R. Boer, "Sharing control with haptics: Seamless driver support from manual to automatic control," *Human factors*, vol. 54, no. 5, pp. 786–798, 2012.

[26] F. Mars, M. Deroo, and J.-M. Hoc, "Analysis of human-machine cooperation when driving with different degrees of haptic shared control," *IEEE transactions on haptics*, vol. 7, no. 3, pp. 324–333, 2014.

[27] M. A. Goodrich, W. C. Stirling, and E. R. Boer, "Satisficing revisited," *Minds and Machines*, vol. 10, no. 1, pp. 79–109, 2000.

[28] J. R. Medina, T. Lorenz, and S. Hirche, "Synthesizing anticipatory haptic assistance considering human behavior uncertainty," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 180–190, 2015.

[29] A. Kanazawa, J. Kinugawa, and K. Kosuge, "Adaptive motion planning for a collaborative robot based on prediction uncertainty to enhance human safety and work efficiency," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 817–832, 2019.

[30] Y. Cheng, L. Sun, C. Liu, and M. Tomizuka, "Towards efficient human-robot collaboration with robust plan recognition and trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2602–2609, 2020.

[31] A. M. R. Lazcano, T. Niu, X. C. Akutain, D. Cole, and B. Shyrokau, "Mpc-based haptic shared steering system: a driver modeling approach for symbiotic driving," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 3, pp. 1201–1211, 2021.

[32] S. Kolekar, W. Mugge, and D. Abbink, "Modeling intradriver steering variability based on sensorimotor control theories," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 3, pp. 291–303, 2018.

[33] S. Lefèvre, Y. Gao, D. Vasquez, H. E. Tseng, R. Bajcsy, and F. Borrelli, "Lane keeping assistance with learning-based driver model and model predictive control," in *Int. Symposium on Advanced Vehicle Control*, 2014.

[34] N. P. Chandrasiri, K. Nawa, and A. Ishii, "Driving skill classification in curve driving scenes using machine learning," *Journal of Modern Transportation*, vol. 24, no. 3, pp. 196–206, 2016.

[35] Q. Ji, P. Lan, and C. Looney, "A probabilistic framework for modeling and real-time monitoring human fatigue," *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans*, vol. 36, no. 5, pp. 862–875, 2006.

[36] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, "Driver behavior classification at intersections and validation on large naturalistic data set," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 724–736, 2012.

[37] M. Enev, A. Takakuwa, K. Koscher, and T. Kohno, "Automobile driver fingerprinting." *Proc. Priv. Enhancing Technol.*, vol. 2016, no. 1, pp. 34–50, 2016.

[38] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, and F. Itakura, "Driver modeling based on driving behavior and its evaluation in driver identification," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 427–437, 2007.

[39] S. Di Cairano, D. Bernardini, A. Bemporad, and I. V. Kolmanovsky, "Stochastic mpc with learning for driver-predictive vehicle control and its application to hev energy management," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1018–1031, 2013.

[40] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural computation*, vol. 11, no. 1, pp. 229–242, 1999.

[41] N. Kuge, T. Yamamura, O. Shimoyama, and A. Liu, "A driver behavior recognition method based on a driver model framework," *SAE transactions*, pp. 469–476, 2000.

[42] T. Kumagai, Y. Sakaguchi, M. Okuwa, and M. Akamatsu, "Prediction of driving behavior through probabilistic inference," in *Proc. 8th Intl. Conf. Engineering Applications of Neural Networks*, 2003, pp. 117–123.

[43] N. Oliver and A. P. Pentland, "Driver behavior recognition and prediction in a smartcar," in *Enhanced and Synthetic Vision 2000*, vol. 4023. SPIE, 2000, pp. 280–290.

[44] T. Lynn, P. T. Endo, P. Rosati, I. Silva, G. L. Santos, and D. Ging, "A comparison of machine learning approaches for detecting misogynistic speech in urban dictionary," in *2019 International Conference on Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. IEEE, 2019, pp. 1–8.

[45] A. Caroppo, A. Leone, and P. Siciliano, "Comparison between deep learning models and traditional machine learning approaches for facial expression recognition in ageing adults," *Journal of Computer Science and Technology*, vol. 35, pp. 1127–1146, 2020.

[46] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[49] Y. Dou, F. Yan, and D. Feng, "Lane changing prediction at highway lane drops using support vector machine and artificial neural network classifiers," in *2016 IEEE international conference on advanced intelligent mechatronics (AIM)*. IEEE, 2016, pp. 901–906.

[50] J. Gao, Y. L. Murphey, and H. Zhu, "Multivariate time series prediction of lane changing behavior using deep neural network," *Applied Intelligence*, vol. 48, no. 10, pp. 3523–3537, 2018.

[51] S. Mozaffari, E. Arnold, M. Dianati, and S. Fallah, "Early lane change prediction for automated driving systems using multi-task attention-based convolutional neural networks," *IEEE Transactions on Intelligent Vehicles*, 2022.

[52] D. Lee, Y. P. Kwon, S. McMains, and J. K. Hedrick, "Convolution neural network-based lane change intention prediction of surrounding vehicles for acc," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[53] O. Rákos, S. Aradi, and T. Bécsi, "Lane change prediction using gaussian classification, support vector classification and neural network classifiers," *Periodica Polytechnica Transportation Engineering*, vol. 48, no. 4, pp. 327–333, 2020.

[54] K. Griesbach, M. Beggiato, and K. H. Hoffmann, "Lane change prediction with an echo state network and recurrent neural network in the urban area," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[55] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.

[56] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2118–2125.

[57] J. Tang, F. Liu, W. Zhang, R. Ke, and Y. Zou, "Lane-changes prediction based on adaptive fuzzy neural network," *Expert systems with applications*, vol. 91, pp. 452–463, 2018.

[58] P. Krupka, P. Lukowicz, C. Kreis, and B. Boßdorf-Zimmer, "Learned steering feel by a neural network for a steer-by-wire system," in *10th International Munich Chassis Symposium 2019*. Springer, 2020, pp. 449–464.

[59] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[60] J. Chen, D. Sun, M. Zhao, Y. Li, and Z. Liu, "Dcfs-based deep learning supervisory control for modeling lane keeping of expert drivers," *Physica A: Statistical Mechanics and its Applications*, vol. 567, p. 125720, 2021.

[61] Z. Wang, Z. Yan, and K. Nakano, "Comfort-oriented haptic guidance steering via deep reinforcement learning for individualized lane keeping assist," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 4283–4289.

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[63] D. G. James, F. Boehringer, K. J. Burnham, and D. Copp, "Adaptive driver model using a neural network," *Artificial life and Robotics*, vol. 7, no. 4, pp. 170–176, 2004.

[64] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*. IEEE, 2017, pp. 353–359.

[65] U.S. Federal Highway Administration, "US Highway 101 Dataset," June 2005, [Online]. Available: https://www.fhwa.dot.gov/publications/research/operations/07030/. [Accessed: Aug. 23, 2022].

[66] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain-inspired cognitive model with attention for self-driving cars," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 1, pp. 13–25, 2017.

[67] H. Zang and M. Liu, "Fuzzy neural network pid control for electric power steering system," in *2007 IEEE International Conference on Automation and Logistics*. IEEE, 2007, pp. 643–648.

[68] C. Zang, M. Pei, and Y. Kong, "Few-shot human motion prediction via learning novel motion dynamics," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 846–852.

[69] A. Patel and J. Shah, "Sensor-based activity recognition in the context of ambient assisted living systems: A review," *Journal of Ambient Intelligence and Smart Environments*, vol. 11, no. 4, pp. 301–322, 2019.

[70] T. Zhou and J. P. Wachs, "Early turn-taking prediction with spiking neural networks for human robot collaboration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3250–3256.

[71] H.-S. Moon and J. Seo, "Prediction of human trajectory following a haptic robotic guide using recurrent neural networks," in *2019 IEEE World Haptics Conference (WHC)*. IEEE, 2019, pp. 157–162.

[72] F. Helenon, S. Thiery, E. Nyiri, and O. Gibaru, "Cognitive architecture for intuitive and interactive task learning in industrial collaborative robotics," in *2021 the 5th International Conference on Robotics, Control and Automation*, 2021, pp. 119–124.

[73] V. A. Akpan and G. Hassapis, "Adaptive predictive control using recurrent neural network identification," in *2009 17th Mediterranean Conference on Control and Automation*. IEEE, 2009, pp. 61–66.

[74] J.-F. Toubeau, J. Bottieau, F. Vallée, and Z. De Grève, "Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1203–1215, 2018.

[75] D. Tavernini, M. Metzler, P. Gruber, and A. Sorniotti, "Explicit nonlinear model predictive control for electric vehicle traction control," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1438–1451, 2018.

[76] M. Jalali, A. Khajepour, S.-k. Chen, and B. Litkouhi, "Integrated stability and traction control for electric vehicles using model predictive control," *Control Engineering Practice*, vol. 54, pp. 256–266, 2016.

[77] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli, "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads," in *16th international IEEE conference on intelligent transportation systems (ITSC 2013)*. IEEE, 2013, pp. 378–383.

[78] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 174–179.

[79] S. Bae, D. Saxena, A. Nakhaei, C. Choi, K. Fujimura, and S. Moura, "Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent neural network," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1209–1216.

[80] P. Mahadika, A. Subiantoro, and B. Kusumoputro, "Neural network predictive control approach design for adaptive cruise control," *Int. J. Technol*, vol. 11, p. 1451, 2020.

[81] M. A. Hosen, M. A. Hussain, and F. S. Mjalli, "Control of polystyrene batch reactors using neural network based model predictive control (nnmpc): An experimental investigation," *Control Engineering Practice*, vol. 19, no. 5, pp. 454–467, 2011.

[82] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017.

[83] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[84] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[85] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: a system for Large-Scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[86] M. Damian, B. Shyrokau, X. C. Akutain, and R. Happee, "Experimental validation of torque-based control for realistic handwheel haptics in driving simulators," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 196–209, 2021.

[87] L. Saleh, P. Chevrel, F. Claveau, J.-F. Lafay, and F. Mars, "Shared steering control between a driver and an automation: Stability in the presence of driver behavior uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 974–983, 2013.

[88] B. Pano, Y. Zhao, P. Chevrel, F. Claveau, and F. Mars, "Shared control based on an ecological feedforward and a driver model based feedback," *IFAC-PapersOnLine*, vol. 52, no. 5, pp. 385–392, 2019.

[89] G. Markkula and J. Engström, "A steering wheel reversal rate metric for assessing effects of visual and cognitive secondary task load," in *Proceedings of the 13th ITS World Congress*. Leeds, 2006.

[90] A. Blum, "On-line algorithms in machine learning," *Online algorithms*, pp. 306–325, 1998.

[91] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: Learning deep neural networks on the fly," *arXiv preprint arXiv:1711.03705*, 2017.

[92] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas, "A review of adaptive online learning for artificial neural networks," *Artificial Intelligence Review*, vol. 49, no. 2, pp. 281–299, 2018.

[93] J. Gao, S. Reddy, G. Berseth, N. Hardy, N. Natraj, K. Ganguly, A. D. Dragan, and S. Levine, "X2t: Training an x-to-text typing interface with online learning from user feedback," *arXiv preprint arXiv:2203.02072*, 2022.

[94] N. Carrara, R. Laroche, and O. Pietquin, "Online learning and transfer for user adaptation in dialogue systems," 2017.

[95] J. Schmidhuber, J. Zhao, and M. Wiering, "Simple principles of metalearning," *Technical report IDSIA*, vol. 69, pp. 1–23, 1996.

[96] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[97] H.-S. Moon and J. Seo, "Dynamic difficulty adjustment via fast user adaptation," in *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 2020, pp. 13–15.

[98] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[99] Y.-X. Wang and M. Hebert, "Learning to learn: Model regression networks for easy small sample learning," in *European Conference on Computer Vision*. Springer, 2016, pp. 616–634.

[100] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of maml," *arXiv preprint arXiv:1909.09157*, 2019.

[101] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1920–1930.