Physics-Informed Neural Networks for Vehicle Lateral Dynamics Modeling

Master's thesis in Programme MPDSC & MPSYS

Yuchuan Dong
Rishikesh Vishnu Sivakumar

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's Thesis in Programme MPDSC & MPSYS

Physics-Informed Neural Networks for Vehicle Lateral Dynamics Modeling

Yuchuan Dong
Rishikesh Vishnu Sivakumar

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2025

Physics-Informed Neural Networks for Vehicle Lateral Dynamics Modeling

Yuchuan Dong
Rishikesh Vishnu Sivakumar

Supervisor: Karthik Prasad and Utsav Khan, Zeekr
Examiner: Fredrik Bruzelius, Mechanics and Maritime Sciences

Master's Thesis 2025
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Visualizing lateral dynamics modeling using past trajectory data and embedded physical model..

Typeset in LaTeX
Gothenburg, Sweden 2025

Physics-Informed Neural Networks for Vehicle Lateral Dynamics Modeling

Yuchuan Dong, Rishikesh Vishnu Sivakumar

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

Accurate estimation of vehicle lateral dynamics is important for advanced driver assistance systems (ADAS) and autonomous driving applications. This thesis proposes a Physics-informed Neural Network (PINN) framework that combines fundamental vehicle dynamics, based on simplified single-track models and Pacejka tire formulations, with data-driven recurrent neural networks that incorporate attention mechanisms. By embedding physical constraints within the learning process, the hybrid model aims to improve interpretability, robustness, and real-time performance without relying on additional sensor inputs. Evaluations using high-fidelity simulations and real-world datasets suggest that the model can estimate latent states, such as lateral velocity, with promising accuracy. While the approach generalizes well across a variety of simulated driving scenarios, it faces challenges in maintaining comparable performance on real-world data mainly due to modeling uncertainties and measurement noise. These limitations highlight the need for further investigation into robustness and domain adaptation techniques. Future work will explore on-board deployment, adaptation to individual vehicle characteristics, and the integration of more detailed physical models. These developments could enhance the model's reliability and utility in practical applications, contributing to improved vehicle safety and operational efficiency.

Keywords: Physics-Informed Neural Networks (PINN), Vehicle Dynamics, Lateral Dynamics Modeling, Parameter Estimation, Recurrent Neural Networks, Attention Mechanism, System Identification, Hybrid Modeling, Autonomous Driving, State Reconstruction.

Preface

This report presents the outcome of our master's thesis project carried out at the Department of Mechanics and Maritime Sciences at Chalmers University of Technology during the winter of 2025 in cooperation with Zeekr Technology Europe in Gothenburg, Sweden.

Acknowledgements

We would like to thank everyone who has supported us throughout this project. We extend special thanks to our supervisors at Zeekr, Karthik Prasad and Utsav Khan, and the entire motion SW development team, for their invaluable guidance during our thesis. Through them, we have learned a great deal about vehicle dynamics, control, and the automotive industry as a whole. Their passion for the field has inspired us to keep moving forward. We are also grateful to Alexander Hägglund and Shreyas Kogalur at IPG Automotive Sweden for their technical support with CarMaker. Finally, we thank our academic supervisor, Fredrik Bruzelius, for his flexibility, willingness to assist, and valuable feedback throughout the project.

Yuchuan Dong, Rishikesh Vishnu Sivakumar, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

PINN	Physics Informed Neural Network
OEM	Original Equipment Manufacturer
MLP	Multilayer Perceptron
DOF	Degrees of Freedom
RNN	Recurrent Neural Network
GRU	Gated Recurent Unit
LSTM	Long Short Term Memory
PhyAtteN	Physics and Attention Attend in Network
BPTT	Back Propagation Through Time
RMSE	Root Mean Square Error
RWD	Rear Wheel Drive
AWD	All Wheel Drive
FLOP	Floating Point Operation
ADAS	Advanced Driver Assistance Systems
CG	Center of Gravity
EKF	Extended Kalman Filter
EV	Electric Vehicle
GNSS	Global Navigation Satellite System
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
PDE	Partial Differential Equation
SGD	Stochastic Gradient Descent
UKF	Unscented Kalman Filter
ADAC	Allgemeiner Deutscher Automobil-Club
AMS	Auto Motor und Sport
VDA	Verband der Automobilindustrie
GNSS	Global Navigation Satellite Systems
INS	Inertial Navigation Systems

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Parameters

m	Mass
g	Acceleration due to gravity
M_z	Yaw moment
l_f	Longitudinal distance from the vehicle's centre of mass to the front axle
l_r	Longitudinal distance from the vehicle's centre of mass to the rear axle
L_{wb}	Wheelbase
C_f	Front axle cornering stiffness
C_r	Rear axle cornering stiffness
B	Stiffness factor
C	Shape factor
D	Peak factor
E	Curvature factor
μ	Peak friction coefficient
h_{CoM}	Height of the vehicle's center of mass above the ground
$a_0 \dots a_8$	Pacejka '89 Magic-Formula load-dependency coefficients
L	Number of hidden layers in the network.
$n^{(\ell)}$	Number of neurons in layer ℓ.
T	Length of the input (or output) sequence.
η	Learning rate.
μ	Momentum coefficient .
λ	Weight-decay coefficient.

p	Dropout probability.
B	Mini-batch size.
E	Number of training epochs.
$W^{(\ell)}$, $b^{(\ell)}$	Weight matrix and bias vector at layer ℓ .
$W^{x,(\ell)}$, $W^{h,(\ell)}$, $b^{(\ell)}$	Input-to-hidden and hidden-to-hidden weight matrices, plus bias, at layer ℓ.
W^z, U^z, b^z, W^r, U^r, b^r, W^h, U^h, b^h	GRU parameter matrices and biases.
v^a, W^a, U^a, b^a	Attention mechanism parameters.
θ	Trainable parameters of the neural network (weights and biases).
N_{data}	Number of data (observation) points.
N_{phys}	Number of collocation (physics) points.
$\mathcal{N}[\cdot]$	Differential operator (or set of PDE operators) defining the physics.

Variables

v_y	Lateral velocity on body frame
$\dot{v}_y$	Lateral acceleration on body frame
v_x	Longitudinal velocity on body frame
$\dot{\psi}$	Yaw rate
$\ddot{\psi}$	Yaw acceleration
F_{yf}	Lateral front tire force
F_{yr}	Lateral rear tire force
δ_f	Front steering angle
$\dot{\omega}$	Yaw acceleration
α_f	Front tire slip angle
α_r	Rear tire slip angle
F_z	Vertical load
ΔF_z	Load transfer due to longitudinal acceleration
$F_{zf_{\text{static}}}$	Static normal load on the front axle
$F_{zr_{\text{static}}}$	Static normal load on the rear axle
F_{zf}	Dynamic normal load on the front axle (after transfer)
F_{zr}	Dynamic normal load on the rear axle (after transfer)
x	Input feature vector (or sequence $\{x_t\}$).

y	Network prediction (or sequence $\{y_t\}$).
$\hat{y}$	Ground-truth target (or sequence $\{\hat{y}_t\}$).
$a^{(\ell)}$	Post-activation at layer ℓ (MLP).
$z^{(\ell)}$	Pre-activation at layer ℓ (MLP).
$h_t^{(\ell)}$	Hidden state at time t in layer ℓ (RNN/GRU).
u_t	Pre-activation of the RNN output at time t.
s_t	Decoder hidden state at time t (in the attention model).
$e_{t,i}$	Unnormalized attention score for encoder position i at decoding step t.
$\alpha_{t,i}$	Normalized attention weight at step t for position i.
c_t	Context vector computed by attention at decoding step t.
ℓ_t	Per-step loss at time t (in BPTT).
$\mathcal{L}$	Total loss (summed or averaged over all steps or examples).
$\delta^{(\ell)}$, $\delta_t^{(\ell)}$	Error signal at layer ℓ (and at time t in BPTT).
$u(x)$	True physical field (solution of the PDE).
$u_\theta(x)$	Neural network approximation of $u(x)$.
$r(x;\theta)$	Physics residual $= \mathcal{N}[u_\theta](x)$.
x_j	Collocation points where physics constraints are enforced.

Contents

List of Figures

List of Tables

1

Introduction

This chapter introduces the motivation and foundation for modeling vehicle lateral dynamics using Physics-Informed Neural Networks (PINNs). Traditionally, physics-based models offer interpretability but may lack flexibility, while purely data-driven models struggle with generalizability and physical consistency. This chapter presents the rationale for exploring hybrid approaches that combine the strengths of both paradigms. It outlines the relevant background, clarifies the specific goals and scope of the project, and sets the stage for the theoretical and practical developments presented in the following chapters.

1.1 Vehicle Dynamics

Vehicle dynamics is the study of motion of vehicles. Unlike kinematics which deals only with the motion of the vehicle disregarding forces and moments, dynamics deals with the forces and moments that cause and affect this motion.

Vehicle dynamics is important for virtually all aspects of the automotive product cycle. Accurate modeling of vehicle dynamics is crucial for important automotive functions. These include simulation, performance and safety analysis, system integration, and model-based state estimation and control algorithms.

The precise modeling of a vehicle's response to inputs from the driver, road, and environment enables engineers to optimize vehicle behavior. This optimization improves safety, comfort, efficiency, and performance characteristics before physical prototypes are built.

1.2 Background

Traditionally, physics-based approaches have been applied to model various aspects of vehicle dynamics. These include the single-track model (commonly known as the bicycle model), the double-track model, and models incorporating roll and pitch dynamics. The bicycle model simplifies a four-wheeled vehicle by combining the left and right wheels into single front and rear wheels. This approach offers computational efficiency and can be suitable for moderate driving scenarios, though it may not capture certain behaviors under high lateral acceleration.

Double-track models represent all four wheels individually. This allows for more detailed modeling, particularly during cornering. However, this approach involves a larger set of parameters and higher computational cost. Roll-and-pitch models further extend the fidelity by including the vehicle's rotational dynamics. Incorporating suspension components and tire models can increase the complexity of such systems.

Physics-based models are valued for their interpretability and their basis in established physical principles. However, enhancing their fidelity can increase computational demands and introduce challenges in parameter identification. Accurately identifying parameters often requires dedicated testing procedures, which may be resource-intensive. Additionally, when operating near the limits of vehicle dynamics, such as during aggressive maneuvers, the assumptions of linearity in tire models or subsystems may become less valid, potentially affecting prediction accuracy.

In recent years, data-driven approaches have been explored for modeling vehicle dynamics. Examples include recurrent neural networks (RNNs), long short-term memory (LSTM) networks, and gated recurrent units (GRUs). These models are capable of learning complex relationships from data and, once trained, can offer fast inference. They have been used to capture nonlinear behaviors that may be difficult to model explicitly using traditional techniques.

Despite these advantages, data-driven models present challenges. They typically offer limited interpretability compared to physics-based methods, and it can be difficult to trace the rationale behind their predictions. Their performance on driving scenarios not well represented in the training data may vary, and they often require substantial datasets to reach satisfactory levels of accuracy. Scenarios that occur infrequently in the data, such as extreme maneuvers, can be particularly challenging. This may pose issues for applications where consistent performance across a wide range of conditions is important.

To address some of these limitations, recent research has investigated hybrid modeling approaches that combine data-driven methods with physical knowledge. These physics-informed or physics-aware neural networks aim to integrate physical laws and constraints within the learning process. This can be achieved through architectural choices, loss functions, or by learning residuals on top of a physics-based baseline.

While physics-informed neural networks offer several potential advantages, they also come with their own set of challenges. Designing appropriate physics-based constraints or loss terms requires careful consideration and domain expertise. If the embedded physics is oversimplified or not representative of the real-world system, it may introduce biases into the learning process. Additionally, balancing the influence of physical constraints and data-driven learning during training can be non-trivial, especially when data is noisy or limited. These factors may affect model performance or convergence, depending on the application.

Various strategies have been proposed within this hybrid framework. Some methods

use neural networks to learn discrepancies between simplified physical models and observed data. Others embed ordinary differential equations into the structure of the network or use custom loss functions that encourage adherence to conservation laws and other physical principles.These hybrid approaches have shown encouraging results in vehicle dynamics modeling. They offer a pathway to capture more complex behaviors than traditional models while maintaining consistency with known physics. In domains where safety and reliability are critical, such as autonomous driving, this balance can be particularly important.

In this thesis, we focus on the application of physics-informed neural networks to vehicle lateral dynamics. This narrower scope enables a detailed exploration of how physics-aware learning techniques can be applied to key aspects of vehicle behavior.

1.3 Goals of the project:

The primary goals of this thesis are:

- To design and implement a hybrid modeling approach for vehicle lateral dynamics that combines first-principles single-track equations and the Pacejka tire model with data-driven neural networks.
- To enforce physical consistency and interpretability by embedding physical constraints directly into the learning and inference process.
- To ensure the model architecture is lightweight and deployable in production environments without additional effort on hardware or infrastructure.
- To demonstrate that the proposed physics-informed neural network (PINN) generalizes across a range of driving scenarios, including extreme maneuvers.
- Conduct systematic evaluation and benchmarking against pure data driven method quantify accuracy, robustness, and computational performance.

1.4 Research Questions

The Box aphorism above splits naturally into two focused questions:

RQ1 – All models are wrong" *Quantify the mismatch.* How large, structured, and scenario-dependent is the error between the textbook single-track model and ground-truth lateral dynamics obtained from production-grade sensors?

RQ2 – . . . but some are useful" *Test a remedy.* Can a lightweight, physics-informed neural correction layer reduce the RQ1 error in real time *while* preserving interpretability?

1.5 Scope of the project

This thesis project focuses on developing a deployable estimator for lateral vehicle dynamics using physics-informed neural networks. The focus is on compatibility with OEM hardware and real-world deployment constraints rather than ideal lab setups.

- **Sensor budget:** Only measurements available on current EV platforms are used (steering input, wheel speeds, IMU data, axle torques). No specialized sensors for lateral velocity, tyre force, or normal load are required.
- **Model generality:** Tyre and vehicle parameters are learned from data while respecting physical laws.
- **Driving envelope:** Evaluation spans asphalt surfaces of varying grip and a wide range of maneuvers—from aggressive slaloms to routine urban driving.

Out of scope: Longitudinal and vertical dynamics, detailed suspension dynamics, aerodynamic loads, hardware-specific optimization and on-board deployment are left to future work.

2

Theory

In this chapter, we describe the key models that form the basis of our hybrid architecture for vehicle lateral dynamics. These include physics-based models that provide physically interpretable insights into vehicle and tire behavior. We also introduce data-driven models that learn patterns directly from sensor data. Finally, we present Physics-Informed Neural Networks (PINNs), which integrate both approaches by embedding physical laws into the learning process. Understanding these components is essential for grasping how the hybrid model leverages both domain knowledge and data to achieve accurate and generalizable performance.

2.1 Physics-based models

This section presents the physics-based models used in our hybrid model. These models are grounded in fundamental physical principles and equations of motion, offering a high degree of interpretability and often serving as the baseline for control and estimation systems. The sub-sections below introduce the single-track vehicle model and two widely used tire force models, the linear tire model and the Pacejka "Magic Formula" model. Together, these models form the physics backbone of our hybrid lateral dynamics model.

2.1.1 Single-track model

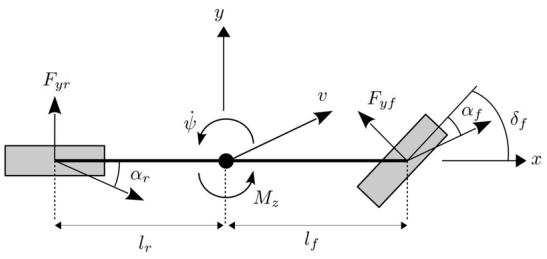

Figure 2.1: Schematic of the single-track vehicle model

Figure 2.1 shows the schematic of the single-track (bicycle) model, which represents a two-axle vehicle by combining each pair of wheels into one on the centreline through the vehicle's centre of mass. The other assumptions of this model include that

vehicle motion is restricted to the xy plane, the chassis behaves as a rigid body so that lateral and yaw motions follow fundamental laws of motion, the steering angle is small allowing linearization of the model and the longitudinal velocity is constant. Tire forces are typically represented by either a linear tire model or the Pacejka "magic formula" model. The model is fully specified by the vehicle mass m, the yaw moment of inertia M_z, and the distances l_f and l_r from the centre of mass to the front and rear axles.

Nonlinear representation of the single-track model:

$$m\big(\dot{v}_y + \dot{\psi}\, v_x\big) = F_{yf} \cos\delta_f + F_{yr} \tag{2.1}$$

$$M_z\, \ddot{\psi} = l_f\, F_{yf} \cos\delta_f - l_r\, F_{yr} \tag{2.2}$$

Taking small steering angles ($\sin\delta \approx \delta$, $\tan\delta \approx \delta$, $\cos\delta \approx 1$), the single-track model can be linearized to produce simple equations that relate the steering angle to lateral acceleration and yaw rate. This small-angle approximation retains the core coupling between steering input and lateral response, simplifying controller design and enabling efficient real-time implementation.

Linear representation of the single-track model:

$$m\big(\dot{v}_y + \dot{\psi}\, v_x\big) = F_{yr} + F_{yf} \tag{2.3}$$

$$M_z\, \ddot{\psi} = -\, l_r\, F_{yr} + l_f\, F_{yf} \tag{2.4}$$

Because it balances fidelity and simplicity, the single-track model is used in research and industry for control and simulation tasks. It serves as the foundation for model-based vehicle control strategies. Advanced driver-assistance functions such as lane keeping and yaw stability control continue to rely on it.

2.1.2 Linear Tire Model

In the single-track model, the lateral tire forces are often represented by a linear tire model, in which each lateral force is assumed proportional to its slip angle via the cornering stiffness of the corresponding axle:

$$F_{yf} = -\, C_f\, \alpha_f \tag{2.5}$$

$$F_{yr} = -\, C_r\, \alpha_r \tag{2.6}$$

Here C_f and C_r denote the front and rear cornering stiffnesses (in N/rad), α_f and α_r are the slip angles at the front and rear axles, and the negative sign indicates that the lateral force always opposes the slip direction.

The slip angles follow from the kinematic geometry of the velocity vector in the body-fixed frame:

$$\alpha_f = \delta_f \;-\; \arctan\!\left(\frac{v_y + l_f\, \omega}{v_x}\right) \tag{2.7}$$

$$\alpha_r = -\arctan\left(\frac{v_y - l_r\,\omega}{v_x}\right) \tag{2.8}$$

where v_x and v_y are the longitudinal and lateral velocities at the centre of mass, l_f and l_r are the distances from the centre of mass to the front and rear axles, ω is the yaw rate, and δ_f is the front-wheel steering angle. Under the small-angle assumption, these reduce via $\arctan x \approx x$ to

$$\alpha_f \approx \delta_f - \frac{v_y + l_f\,\omega}{v_x} \tag{2.9}$$

$$\alpha_r \approx -\frac{v_y - l_r\,\omega}{v_x} \tag{2.10}$$

The cornering stiffness coefficients C_f and C_r depend on tire construction, inflation pressure, and vertical load. While the linear tire model provides accurate predictions for slip angles up to approximately 5°, real tires exhibit saturation and nonlinear effects at larger slip angles, causing the linear approximation to degrade.

2.1.3 Pacejka "Magic Formula" Tire Model

While the linear tire model captures small-angle behavior, real tires exhibit strongly nonlinear force vs slip characteristics at moderate to large slip angles. To overcome this, the Pacejka 'Magic Formula' (MF) was developed. The Pacejka model provides an empirical representation that fits measured tire data across a large range of slip. In its most common form, the lateral force F_y is expressed as

$$F_y(\alpha) = D\,\sin\Big(C\,\arctan\Big\{\,B\,\alpha - E\,[\,B\,\alpha - \arctan(B\,\alpha)\,]\Big\}\Big). \tag{2.11}$$

In this expression, the parameter B is the stiffness factor, C is the shape factor, D is the peak factor, and E is the curvature factor.

The parameters B, C, D, and E are typically determined by fitting to experimental data for each tire under a given vertical load F_z. Often, D is expressed as

$$D = \mu\,F_z \tag{2.12}$$

where μ is the peak friction coefficient, and B, C, E are functions of F_z.

Equation (11) yields three characteristic regions: for very small α, $F_y \approx BCD\,\alpha$, recovering a linear slope; at moderate slip the force saturates near the peak D; and as α increases further the "wing" region appears, modeling the drop-off in lateral force beyond peak grip as seen in fig 2.2

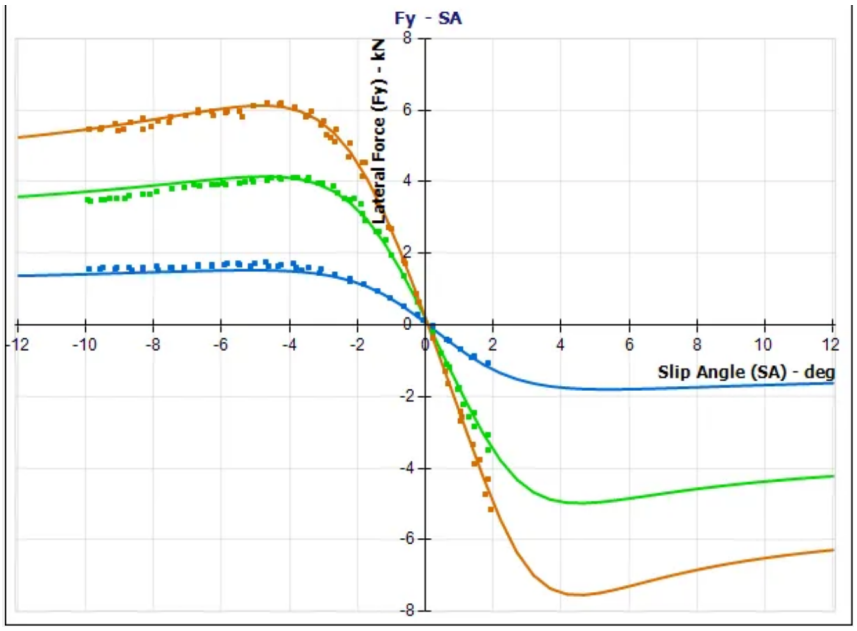

Figure 2.2: Lateral tire force F_y as a function of slip angle (SA) for three vertical loads: low (blue), medium (green), and high (orange). Taken from OptimumG [3].

Incorporating the magic formula into the single-track dynamics simply replaces the linear relations (5)–(6) with

$$F_{yf} = D_f \sin\Big[C_f \arctan\Big\{B_f\,\alpha_f - E_f\,[B_f\,\alpha_f - \arctan(B_f\,\alpha_f)]\Big\}\Big]. \tag{2.13}$$

$$F_{yr} = D_r \sin\Big[C_r \arctan\Big\{B_r\,\alpha_r - E_r\,[B_r\,\alpha_r - \arctan(B_r\,\alpha_r)]\Big\}\Big]. \tag{2.14}$$

using separate parameter sets (B_f, C_f, D_f, E_f) and (B_r, C_r, D_r, E_r) for the front and rear tires.

The Pacejka model thus captures both the linear cornering stiffness at small slip and the nonlinear saturation and roll-off behavior at high slip angles, making it suitable for high-fidelity vehicle simulations and advanced control design.

Pacejka89 model Implementation:

Since we use Pacejka89[2] in our implementation, we mention it as well. The fundamental equation is expanded to include horizontal and vertical shifts (S_h and S_v):

$$F_y = D\sin[C\arctan\{B(\alpha+S_h) - E(B(\alpha+S_h) - \arctan(B(\alpha+S_h)))\}] + S_v \tag{2.15}$$

Unlike simpler implementations that use fixed BCDE parameters, our model dynamically calculates these parameters based on the tire's normal load F_z. This is

accomplished through a set of intermediate coefficients (a_0 through a_8) that define how tire properties change with load.

The coefficient calculations are:

$$C = a_0 \tag{2.16}$$

$$D = a_1 F_z^2 + a_2 F_z \tag{2.17}$$

$$BCD = a_3 \sin\Big(a_4 \arctan(a_5 F_z)\Big) \tag{2.18}$$

$$B = \frac{BCD}{C \cdot D} \tag{2.19}$$

$$E = a_6 F_z^2 + a_7 F_z + a_8 \tag{2.20}$$

The same structure is used for both front and rear tire parameters with their corresponding coefficients.

The normal forces are calculated under the assumption of load transfer during longitudinal acceleration. Static loads are distributed according to the center of gravity position:

$$F_{zf\_static} = \frac{l_r}{L} m\, g \tag{2.21}$$

$$F_{zr\_static} = \frac{l_f}{L} m\, g \tag{2.22}$$

During acceleration, load transfers from front to rear (or rear to front during braking):

$$\Delta F_z = \frac{m \cdot a_x \cdot h_{CoM}}{L_{wb}} \tag{2.23}$$

$$F_{zf} = F_{zf\_static} - \Delta F_z \tag{2.24}$$

$$F_{zr} = F_{zr\_static} + \Delta F_z \tag{2.25}$$

Where l_f and l_r are the distances from center of gravity to front and rear axles, L is the wheelbase, m is the vehicle mass, g is the gravitational acceleration, a_x is the longitudinal acceleration, and h_{CoM} is the height of the center of mass.

2.2 Data-Driven Models

This section presents the data-driven components used in our hybrid modeling approach. These models learn patterns directly from temporal input–output data, allowing them to capture complex and nonlinear vehicle behaviors that may be difficult to model explicitly using physics alone. We describe neural network architectures that serve as the learning backbone of our system: Multilayer Perceptrons (MLPs) for static mappings, Recurrent Neural Networks (RNNs) and Gated Recurrent Units (GRUs) for sequential modeling, and attention mechanisms for context-aware estimation. The section concludes with Physics-Informed Neural Networks (PINNs), which blend these data-driven components with physical constraints to form the learning core of our hybrid lateral dynamics model.

2.2.1 Multilayer Perceptron (MLP)

A Multilayer Perceptron(MLP; see Figure 2.3) is a type of feedforward neural network that consists of an input layer, one or more hidden layers, and an output layer [4, 5]. Each layer is made of units called perceptrons or neurons that perform an affine transformation followed by a nonlinear activation [4].

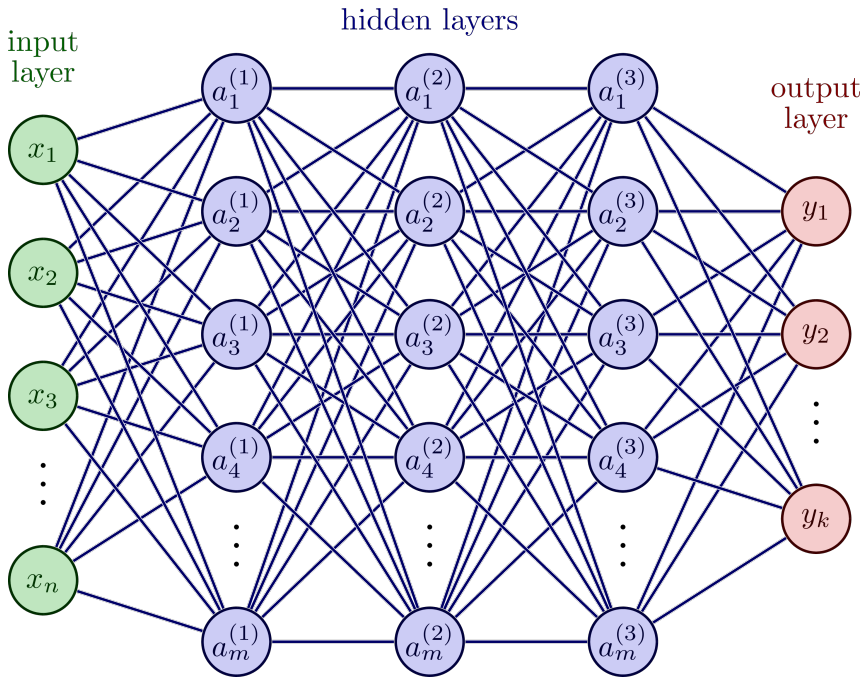

Figure 2.3: Architecture of a three hidden layer multilayer perceptron mapping input x to output y.

For an input vector x, we introduce a separate symbol for the pre-activation and post-activation at each layer. Let

$$a^{(0)} = x,$$

and for $\ell = 1, \ldots, L$ define

$$z^{(\ell)} = W^{(\ell)}\, a^{(\ell-1)} + b^{(\ell)}, \quad \ell = 1, \ldots, L \tag{2.26}$$

$$a^{(\ell)} = \sigma\big(z^{(\ell)}\big), \quad \ell = 1, \ldots, L \tag{2.27}$$

Here $W^{(\ell)}$ and $b^{(\ell)}$ are the learnable weight matrix and bias vector at layer ℓ, and σ is a pointwise nonlinearity (e.g. ReLU or sigmoid) [4].

After L hidden layers, the network computes its final linear output

$$z^{(L+1)} = W^{(L+1)}\, a^{(L)} + b^{(L+1)}, \tag{2.28}$$

and then applies an output activation:

$$y = \sigma_{\text{out}}\big(z^{(L+1)}\big), \tag{2.29}$$

where σ_{out} is, for example, the softmax in classification or the identity in regression. All parameters $\{W^{(\ell)}, b^{(\ell)}\}$ are learned by backpropagation to minimize a chosen loss function [7].

2.2.2 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN)[8] can be thought of as a feed-forward network that processes inputs one step at a time over time, where each input (commonly a time step t) has its own hidden layer. The weights are the same across all time steps.

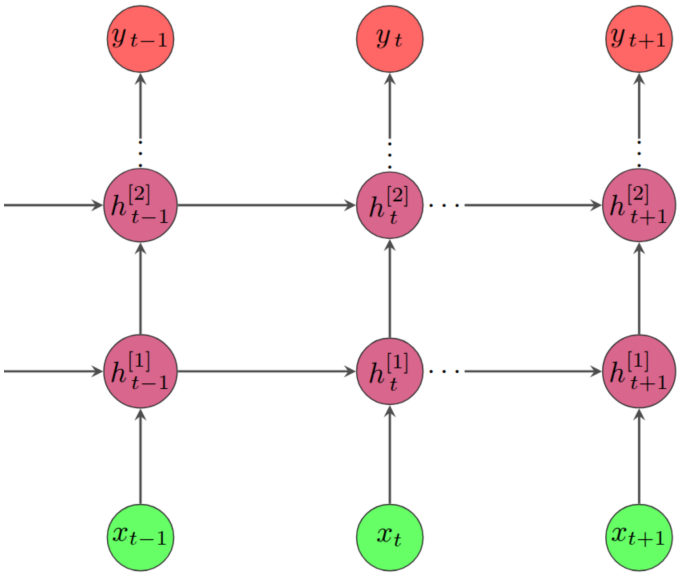

Figure 2.4: Architecture of a general Deep Recurrent Neural Network.

At each time step t, a stacked RNN (see Figure 2.4) processes the current input x_t and the previous hidden states $\{h^{(\ell)}_{t-1}\}_{\ell=1}^{L}$. For each layer $\ell = 1, \ldots, L$, we write

$$z_t^{(\ell)} = W^{x,(\ell)}\, h_t^{(\ell-1)} + W^{h,(\ell)}\, h_{t-1}^{(\ell)} + b^{(\ell)}, \quad \ell = 1, \ldots, L, \quad h_t^{(0)} = x_t \tag{2.30}$$

$$h_t^{(\ell)} = \sigma\big(z_t^{(\ell)}\big), \quad \ell = 1, \ldots, L \tag{2.31}$$

Here, $W^{x,(\ell)}$ and $W^{h,(\ell)}$ denote the input-to-hidden and hidden-to-hidden weight matrices at layer ℓ, $b^{(\ell)}$ is the bias vector, and σ is a point-wise activation function (tanh is mostly used). The output at time t is then

$$u_t = W^{(y)}\, h_t^{(L)} + b^{(y)}, \quad y_t = \sigma_{\text{out}}(u_t), \tag{2.32}$$

with σ_{out} (e.g. softmax or identity). All parameters are shared across time and learned by backpropagation through time [9].

2.2.3 Gated Recurrent Unit(GRU)

One major problem with recurrent neural networks is that they suffer from vanishing or exploding gradients [10, 12]. A vanishing gradient occurs when you backpropagate through time and the gradients approach zero, slowing learning and preventing the model from capturing long-term dependencies. In contrast, exploding gradients occur when the gradients grow exponentially, causing large updates and numerical overflow, which also hinders learning. One way to overcome this is to use a Gated Recurrent Unit [13].Long Short-Term Memory (LSTM) networks represent another approach to addressing these gradient issues and will be used as a benchmark in this thesis, though they are not reviewed in detail here[11].

The GRU reduces this by merging its memory and hidden vectors into a single state, then employing two gates to control information flow. At time step t, the update gate

$$z_t = \sigma\Big(W^z x_t + U^z h_{t-1} + b^z\Big) \tag{2.33}$$

determines how much of the previous activation h_{t-1} to preserve, whereas the reset gate

$$r_t = \sigma\Big(W^r x_t + U^r h_{t-1} + b^r\Big) \tag{2.34}$$

manipulates which components of h_{t-1} contribute to the candidate $\tilde{h}_t$. [13].

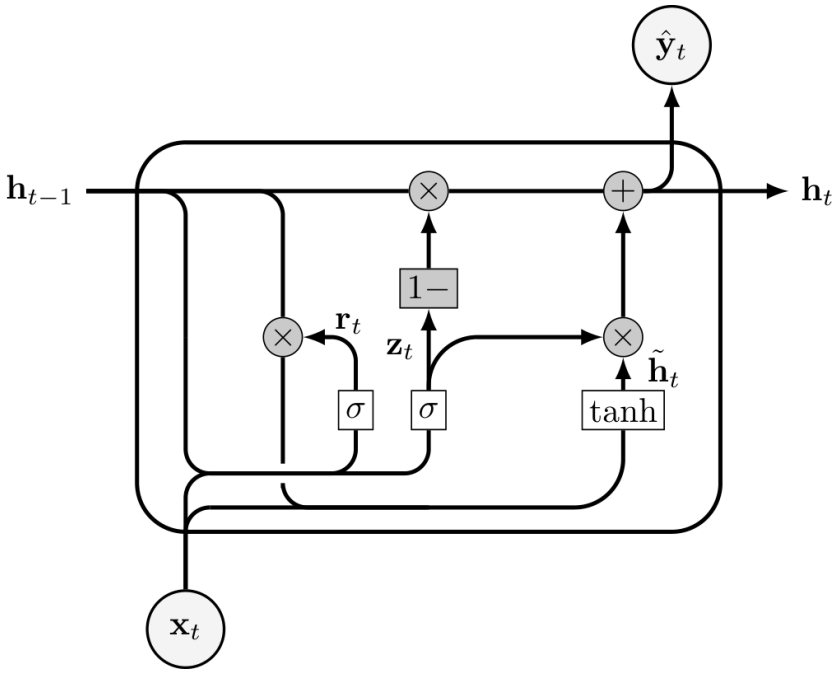

Figure 2.5: Architecture of a general GRU.

This candidate is then computed by first applying the reset gate element-wise to the prior activation and combining it with the current input, followed by a hyperbolic tangent:

$$\tilde{h}_t = \tanh\Big(W^h x_t + U^h\,(r_t \circ h_{t-1}) + b^h\Big) \tag{2.35}$$

Finally, the new hidden state h_t is formed by interpolating between the old state and the candidate according to the update gate:

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t \tag{2.36}$$

By adaptively learning when to reset, update, or retain information with just two gates, GRUs perform well on sequence modeling tasks while using fewer parameters and requiring less computation [13].

2.2.4 Backpropagation and Backpropagation Through Time

Backpropagation and Backpropagation Through Time are fundamental to training our model, so we devote a separate subsection to them. When training a neural network (MLP), we need to compute how each weight contributes to the overall loss by propagating an error signal backwards through the layers. We do this by taking gradients of the loss with respect to the weights and biases[6]. Once the MLP has produced

$$z^{(L+1)} = W^{(L+1)}\, a^{(L)} + b^{(L+1)} \tag{2.37}$$

$$y = \sigma_{\text{out}}\Big(z^{(L+1)}\Big) \tag{2.38}$$

we define a scalar loss

$$\mathcal{L} = \mathcal{L}\Big(y,\, \hat{y}\Big) \tag{2.39}$$

where $\hat{y}$ is the ground-truth target, and $\mathcal{L}(\cdot, \cdot)$ denotes the chosen loss function (e.g. cross-entropy for classification[4, 5] or mean-squared error for regression[6]).

The error at the output pre-activation is

$$\delta^{(L+1)} = \frac{\partial \mathcal{L}}{\partial z^{(L+1)}} = \frac{\partial \mathcal{L}}{\partial y} \;\circ\; \sigma'_{\text{out}}\Big(z^{(L+1)}\Big) \tag{2.40}$$

and for each hidden layer $\ell = L, \ldots, 1$

$$\delta^{(\ell)} = \Big(W^{(\ell+1)\top}\, \delta^{(\ell+1)}\Big) \;\circ\; \sigma'\Big(z^{(\ell)}\Big) \tag{2.41}$$

The gradients are then

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \delta^{(\ell)}\, a^{(\ell-1)\top} \tag{2.42}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(\ell)}} = \delta^{(\ell)} \tag{2.43}$$

and then gradient-descent updates the parameters as

$$W^{(\ell)} \;\leftarrow\; W^{(\ell)} - \eta\, \frac{\partial \mathcal{L}}{\partial W^{(\ell)}} \tag{2.44}$$

$$b^{(\ell)} \;\leftarrow\; b^{(\ell)} - \eta\, \frac{\partial \mathcal{L}}{\partial b^{(\ell)}} \tag{2.45}$$

where η is the learning rate [4].

In practice, these basic gradient-descent updates can be extended with a variety of optimization algorithms that accelerate convergence and improve generalization. Common choices include momentum SGD [17], RMSProp [18], and Adam [19], each of which adaptively modifies the step direction or size based on past gradient information.

Extending to Sequences: Backpropagation Through Time(BPTT)

Recurrent networks share the same weights W^x, W^h, and biases at each time step, so when you feed into a whole sequence, you effectively obtain a very deep network if you unfold it at all times[9]. Backpropagation Through Time (BPTT) applies the same backpropagation rule at each time slice and then sums the gradients across all steps.

When you unroll an L-layer RNN over T time-steps, you obtain a feed-forward network of depth $L \times T$. Let the per-step loss be

$$\ell_t = \ell\Big(h_t^{(L)},\, \hat{y}_t\Big), \tag{2.46}$$

where $h_t^{(L)}$ is the top-layer output at time t and $\hat{y}_t$ the corresponding target. The total sequence loss is then

$$\mathcal{L}_{\text{seq}} = \sum_{t=1}^{T} \ell_t. \tag{37}$$

Backpropagation Through Time applies the same chain-rule updates at each layer and each time slice, and then sums the contributions across t. In particular, for each RNN layer $\ell = 1, \ldots, L$, the gradient of the total loss with respect to its recurrent weight matrix $W^{h,(\ell)}$ is

$$\frac{\partial \mathcal{L}_{\text{seq}}}{\partial W^{h,(\ell)}} = \sum_{t=1}^{T} \delta_t^{(\ell)} \left(h_{t-1}^{(\ell)}\right)^\top, \tag{2.47}$$

and similarly the input-to-hidden weight gradient is

$$\frac{\partial \mathcal{L}_{\text{seq}}}{\partial W^{x,(\ell)}} = \sum_{t=1}^{T} \delta_t^{(\ell)} \left(h_t^{(\ell-1)}\right)^\top, \tag{2.48}$$

where

$$\delta_t^{(\ell)} = \frac{\partial \ell_t}{\partial z_t^{(\ell)}} = \left(W^{h,(\ell)\top}\, \delta_{t+1}^{(\ell)} + W^{x,(\ell+1)\top}\, \delta_t^{(\ell+1)}\right) \circ \sigma'\Big(z_t^{(\ell)}\Big) \tag{2.49}$$

with the understanding that $\delta_{T+1}^{(\ell)} = 0$ and $\delta_t^{(L+1)} = \partial \ell_t / \partial z_t^{(L+1)}$. Finally, each parameter is updated by stepping in the negative gradient direction (cf. (35)).

2.2.5 Attention mechanism

Bahdanau et al. [14] introduced attention to relieve the information bottleneck in the vanilla RNN encoder–decoder. In the original architecture the encoder reads the source sequence step-by-step, compressing everything into one fixed-length context vector. The decoder must then generate every target token using only this single summary plus its own recurrent state—a setup that degrades sharply as the source sentence grows longer. Attention removes the bottleneck by letting the decoder create a fresh context vector at each step: it learns alignment weights that highlight the most relevant encoder states for the current output token. Consequently the decoder can "look back" at the entire source history on demand instead of relying on one compressed vector, leading to much better performance on long sequences.

The attention mechanism solves this by letting the decoder focus on different parts of the input at each step. At decoding time t, we compute an unnormalized attention score

$$e_{t,i} = \left(v^a\right)^\top \tanh\left(W^a s_{t-1} + U^a h_i + b^a\right) \tag{2.50}$$

where s_{t-1} is the decoder's previous hidden state and h_i is the ith encoder state. We then apply the softmax function to normalize it:

$$\alpha_{t,i} = \frac{\exp\left(e_{t,i}\right)}{\sum_{j=1}^{T} \exp\left(e_{t,j}\right)} \tag{2.51}$$

which ensures $\sum_i \alpha_{t,i} = 1$ and lets us interpret each $\alpha_{t,i}$ as relative importance assigned to encoder position i [15]. Finally, the context vector

$$c_t = \sum_{i=1}^{T} \alpha_{t,i}\, h_i \tag{2.52}$$

This context c_t is passed into the decoder along with its recurrent state to produce next output token. When visualized, the attention scores $\alpha_{t,i}$ will form a relative map that reveal how relevant each encoder hidden state is for predicting the next output token. Once those scores are normalized into weights, they indicate how much emphasis the model places on each part of the input when building its context. This dynamic and content based weighting allows the decoder to selectively draw on the most informative elements of the input at every step.[14]

2.2.6 Physics Informed Neural Networks (PINNs):

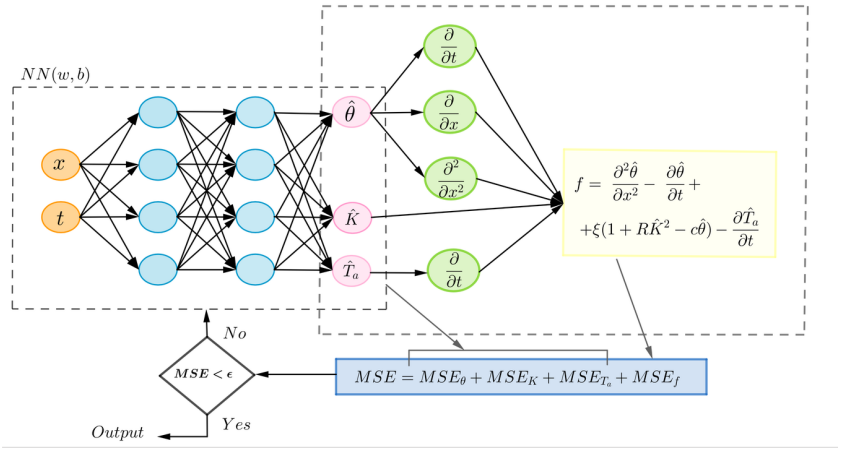

Figure 2.6: A representative PINN architecture illustrating the solution network, automatic differentiation of the PDE operator, and composite loss

Physics-Informed Neural Networks (PINNs)[16] provide a way to fuse data-driven learning with first-principles modeling. By treating differential equations and other constraints as "soft" penalties during training, they learn solution functions that

both fit observed data and rigorously satisfy the underlying physics. A PINN is a neural approximation $u_\theta(x)$ of an underlying physical field $u(x)$ that is trained not only to fit observational data but also to satisfy the governing laws

$$\mathcal{N}[u](x) = 0 \tag{2.53}$$

by enforcing them as soft constraints in the loss. Using automatic differentiation, one computes the physics residual

$$r(x;\theta) = \mathcal{N}\Big[u_\theta\Big](x) \tag{2.54}$$

and combines this with any measurement or boundary-condition data $\{(x_i, y_i)\}$ into the composite loss

$$\boxed{\mathcal{L}(\theta) = \underbrace{\frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \Big\| u_\theta(x_i) - y_i \Big\|^2}_{\text{data–fit}} + \underbrace{\frac{1}{N_{\text{phys}}} \sum_{j=1}^{N_{\text{phys}}} \Big\| r(x_j;\theta) \Big\|^2}_{\text{physics–residual}} + \mathcal{L}'} \tag{2.55}$$

where $\mathcal{L}'$ may include additional penalties for boundary conditions, conservation laws, or other constraints. By minimizing $\mathcal{L}(\theta)$, PINNs blend measurement data and first-principles physics in a mesh-free, end-to-end learning framework capable of both forward prediction and inverse parameter estimation[16].

3

Literature Review

In this chapter, we present a comprehensive review of state-of-the-art approaches for modeling and estimating the lateral dynamics of vehicles. We begin by examining joint parameter-and-state estimation frameworks that simultaneously estimate tire parameters and vehicle states. Then, we analyze pure machine learning techniques that leverage data-driven methodologies without explicit physical constraints. Finally, we explore how physics-informed neural networks are used for hybrid vehicle dynamics modeling, combining neural networks with physical models to improve prediction accuracy while maintaining consistency with fundamental vehicle dynamics principles.

3.1 Model Based State and Parameter Estimation in Vehicle Dynamics

Pure tire-coefficient estimation has been addressed using many methods, such as frequency-response testing, recursive least-squares identification, and adaptive neural-network models. But when it comes to joint estimation of vehicle states and parameters, a significant body of work exists in the area of vehicle parameter and state estimation using Kalman filters and their variants. In fact, Kalman filters remain the go-to approach in the automotive industry to estimate states such as lateral velocity. In this subsection, we review the literature that closely aligns with our work in lateral dynamics estimation, specifically focusing on research that addresses the joint estimation of lateral velocity, tire forces, side-slip angles, and tire coefficients.

In [20], the authors present a recursive, online approach for estimating front and rear tire stiffness within an enhanced single-track model. By integrating vertical, lateral, and yaw dynamics into a unified framework, their algorithm computes load transfer in real time and embeds it in a constrained recursive least-squares estimator with a forgetting factor to maintain realistic parameter bounds. However, the method relies on sufficient lateral or longitudinal excitation to remain accurate and may drift during steady, low-dynamic cruising unless deliberate maneuvers are introduced. Moreover, lacking direct ground-truth validation, the authors assess parameter validity solely through indirect model-fit metrics rather than by comparison with true stiffness values.

In [21], two EKFs run in parallel to jointly estimate both the dynamic states of the vehicle, such as velocities, yaw rate and accelerations, and a small set of key parameters, including mass, moment of inertia, and distance from the center of gravity to the front axle. One EKF propagates and corrects the dynamic states using a four-wheel 4-DOF vehicle model together with measured inputs and outputs while the other treats the vehicle parameters as slowly varying states and refines them over time. This dual EKF approach performed exceptionally well in ideal noise-free simulation data and remained robust in terms of state estimation accuracy and parameter convergence when applied to real-world vehicle measurements. However, the estimator is very sensitive to process noise covariance matrix settings and the performance depends heavily on initial parameter values and filter settings.

Whereas in [22], an adaptive UKF approach improves on standard methods by dynamically updating both noise covariance matrices based on vehicle dynamics. It estimates critical quantities such as velocities, yaw rate, wheel speeds, and tire–road friction. The regular UKF showed similar performance on high-grip surfaces, but the adaptive system proved its worth in challenging conditions instead. Testing in low-friction environments and during sudden friction changes revealed much lower estimation errors, and critically it maintained stability during extreme slip conditions where traditional systems typically fail. This robust performance under degraded model accuracy makes it especially valuable for real-world applications. Even though the covariance matrices are adaptively tuned, one must still carefully design the adaptation rules and heuristics, and sensitivity to initial parameter values remains, especially in extreme conditions.

3.2 Machine Learning Based Vehicle Dynamics Modeling

In [23], an end-to-end GRU + MLP model is trained on real-world vehicle data to predict the next vehicle state such as velocities, yaw rate, accelerations etc. every 4 ms using recent state history and current controls such as steering, torques, brake pressures etc. The model achieves lower prediction errors than the traditional physics-based approach across different grip conditions and remains stable in long open-loop simulations.

Another approach in [24], estimates vehicle sideslip angle with a dual network architecture with the first network predicting longitudinal velocity and the other network using this to predict side slip angle. This creates a virtual sensor that can replace expensive optical sensors typically used to measure sideslip angle directly. This model performed well on different track and tire conditions.

The problem with purely machine learning based approaches as seen above is that they require ground truth for every state we want to model. In lateral dynamics, several states like lateral velocity and parameters such as tire cornering stiffness and slip angles are difficult to measure directly, necessitating specialized devices for accurate estimates. These devices are expensive, hard to integrate, and often unavailable for

mass-market applications, making pure ML-based modeling impractical. Moreover, such models are brittle to domain shifts and require costly retraining or calibration when vehicle, tire, or environmental conditions change. They are also difficult to certify and explain under strict automotive regulations and pose long-term maintenance challenges in a production fleet.

3.3 PINNs for Vehicle Dynamics Modeling

In [25], a Physics Embedded Neural Network Vehicle Model is used which blends deep learning with physics models. It uses the Pacejka tire model and bicycle model to better predict vehicle states and parameters. The model estimates tire coefficients and forces directly from data. It then uses these to extract latent features like tire forces and cornering stiffness. The main drawback is testing only in simulation. They used IPG Carmaker instead of real-world driving data, and only tried a few driving scenarios.

On the otherhand, in [27] a PINN-UKF combination was used. The PINN module outputs IMU calibration values as pseudo-states that feed into the UKF-M module, enabling estimation of vehicle states including attitude, velocity, and position. The method incorporates physical constraints through ODEs in the loss function to handle unobservable states like lateral velocity. However, the PINN here is merely used to compensate for the IMU drift by returning pseudo-states to a UKF and does not produce standalone predictions which is different from what we are trying to achieve through this project, but is still an interesting direction.

Whereas [26] builds upon and addresses key limitations of the [25]. It enforces parameter constraints by implementing a Physics Guard layer, estimates vehicle moment of inertia rather than assuming it's known, and employs a more sophisticated architecture for the machine learning part compared to [25]. Deep Dynamics integrates directly with model-predictive control for both longitudinal and lateral control. There were huge improvements in performance when compared to [25], over 3 orders of magnitude in simulation and 26-56% on real-world data and it generalized better across different track conditions. While primarily designed for autonomous racing at extreme speeds many of its techniques could potentially be adapted for production vehicles with appropriate modifications, which is what we have tried to implement in our work.

3.3.1 Towards a deployable 'Gray Box' approach

The studies reviewed above expose three practical gaps that still separate the state of the art from a deployable estimator.

First, many solutions rely on sensors, such as optical sideslip cameras, wheel-load cells, or high-grade INS units, that are absent from cost-sensitive hardware.

Second, both classic observers and pure data-driven networks remain brittle when tyres, payload, or road grip drift away from the calibration domain.

Third, most hybrid PINN demonstrations are confined to simulation or a handful of manoeuvres, leaving real-world generality largely untested.

To overcome these limitations we introduce PhyAtteN, a grey-box architecture that couples a Pacejka-bicycle core and load-transfer physics to a lightweight, attention-guided RNN. A Physics Guard constrains all learned parameters to plausible ranges, enabling the network to infer latent states, lateral velocity, without ground-truth labels. Crucially, the design runs on standard measurements in the vehicle alone and is validated across both routine traffic and extreme manoeuvres.

The next chapter details this architecture, its design philosophy and state-of-the-art methodology that bridge the gap between research prototypes and deployable estimators for hardware where measure standard signals.

4

PhyAtteN

We introduce *PhyAtteN*, stands for **P**hysics and **Att**ention **Att**end in **N**etwork. A hybrid grey-box architecture that marries classical vehicle-dynamics blocks with a data-driven attention encoder. Guided by two production constraints (no on-board lateral-velocity sensor and ever-shifting wheel loads), PhyAtteN (i) estimate the full nine-term Pacejka tyre coefficients while estimating vertical-load transfer analytically, (ii) rolls the single-track model forward over a short horizon so it can infer sideslip implicitly, and (iii) uses a learnable "best-moment" self-attention to insert the initial hidden state where the sensor context is richest. A physics-guard layer keeps all parameters in lawful ranges, variance regularisers stop the network from cheating by over-tuning the tyre curve, and clipped, truncated-BPTT ensures training remains numerically stable. The result is a physically coherent, data-efficient network that runs in real time yet retains the flexibility to self-correct under aggressive maneuvers.

4.1 Motivation for PhyAtteN

In reality: the absence of lateral velocity sensors and the need for accurate force prediction under varying normal loads, without direct measurement of those loads. To address these, our model integrates a more detailed representation of the tire and a multi-step forward propagation of vehicle dynamics.

Multi-Step Integration Without v_y Input. Since direct measurements of lateral velocity v_y are not available in production vehicles, *PhyAtteN* estimates tire coefficients and implicitly reconstructs the slip angle, lateral force, and sideslip by unrolling the bicycle model over multiple time steps. Given the time series of observed states and driver inputs:

$$\mathbf{X}_t = \begin{bmatrix} v_x(t) \\ r(t) \\ \dots \\ a_x(t) \\ \delta_f(t) \end{bmatrix}, \tag{4.1}$$

PhyAtteN treats the lateral velocity v_y as a hidden state and propagates the single-track bicycle model over a horizon of T discrete time steps (sample time $\Delta t = 10\,\text{ms}$).

For every step $\tau = 0, \ldots, T-1$ the dynamics are advanced according to

$$\hat{v}_{y,t+\tau+1} = \hat{v}_{y,t+\tau} + \frac{\Delta t}{m}\Big(\hat{F}_{yf}(\tau) + \hat{F}_{yr}(\tau)\Big) \; - \; \Delta t\, \hat{r}_{t+\tau}\, v_{x,t+\tau}, \tag{4.2}$$

$$\hat{r}_{t+\tau+1} = \hat{r}_{t+\tau} + \frac{\Delta t}{I_z}\Big(l_f\, \hat{F}_{yf}(\tau) - l_r\, \hat{F}_{yr}(\tau)\Big), \tag{4.3}$$

$$\hat{a}_{y,t+\tau+1} = \frac{1}{m}\Big(\hat{F}_{yf}(\tau) + \hat{F}_{yr}(\tau)\Big). \tag{4.4}$$

Here m, I_z, l_f and l_r are the vehicle parameters supplied by the guarded-parameter head.

The lateral forces are delivered by guarded Pacejka layers

$$\hat{F}_{yf}(\tau) = F_y\Big(\hat{\alpha}_f(\tau); \hat{a}^{(f)}_{0:8}\Big), \qquad \hat{F}_{yr}(\tau) = F_y\Big(\hat{\alpha}_r(\tau); \hat{a}^{(r)}_{0:8}\Big),$$

with slip angles

$$\hat{\alpha}_f(\tau) = \delta_f(t+\tau) - \arctan\frac{\hat{v}_{y,t+\tau} + l_f\, \hat{r}_{t+\tau}}{v_{x,t+\tau}}, \tag{4.5}$$

$$\hat{\alpha}_r(\tau) = -\arctan\frac{\hat{v}_{y,t+\tau} - l_r\, \hat{r}_{t+\tau}}{v_{x,t+\tau}}. \tag{4.6}$$

After the rollout, the predicted state is

$$\hat{\mathbf{X}}_{t+\tau+1} = \begin{bmatrix} a_{y,t+\tau+1} \\ \hat{v}_{y,t+\tau+1} \\ \hat{r}_{t+\tau+1} \end{bmatrix}.$$

Define the measurement vectors

$$\mathbf{z}^{\text{obs}}_{t+\tau+1} = \begin{bmatrix} a^{\text{obs}}_{y,t+\tau+1} \\ r^{\text{obs}}_{t+\tau+1} \end{bmatrix}, \qquad \hat{\mathbf{z}}_{t+\tau+1} = \begin{bmatrix} \hat{a}_{y,t+\tau+1} \\ \hat{r}_{t+\tau+1} \end{bmatrix}.$$

The data-fit loss is then

$$\mathcal{L}_{\text{data}} = \frac{1}{T}\sum_{\tau=0}^{T-1} \Big\| \mathbf{z}^{\text{obs}}_{t+\tau+1} - \hat{\mathbf{z}}_{t+\tau+1} \Big\|_2^2. \tag{4.7}$$

This multi-step rollout offers two key benefits:

- **Self-correcting behaviorover time.** By penalizing accumulated error, the network learns to compensate for drift and model mismatch, even in the presence of unknown disturbances or transient dynamics.
- **Implicit reconstruction of sideslip.** Without ever observing v_y, the model learns to infer tire behaviour and lateral force effects that result in a trajectory consistent with the measured vehicle motion.

Detailed Tire Parameterisation.

Unlike previous works that regress the lumped coefficients (B, C, D, E) - which are sensitive to normal load F_z - we instead estimate the complete set of Pacejka lateral force parameters $\{a_0, a_1, \ldots, a_8\}$.

These parameters are treated as **intrinsic properties** of the tire, independent of time-varying dynamics such as load transfer. This approach is grounded in the huge assumption that the tire's fundamental behavior does not change within short time horizons. By separating the tire's physical characteristics from dynamic influences, we simplify the learning problem and improve generalisability across varying conditions. This huge assumption will be discussed in more detail in following context of chapter.

4.2 Design Philosophy of PhyAtteN

The design of *PhyAtteN* follows a hybrid philosophy inspired by *Axel Villandseie*, who introduced the concept at the 2025 SVEA Vehicle Dynamics Conference[1]. It reflects the target cascading process used in modern vehicle development: a bottom-up stack of physical modeling combined with a top-down learning framework. This dual-perspective philosophy enables *PhyAtteN* to remain both data-efficient and physically consistent.

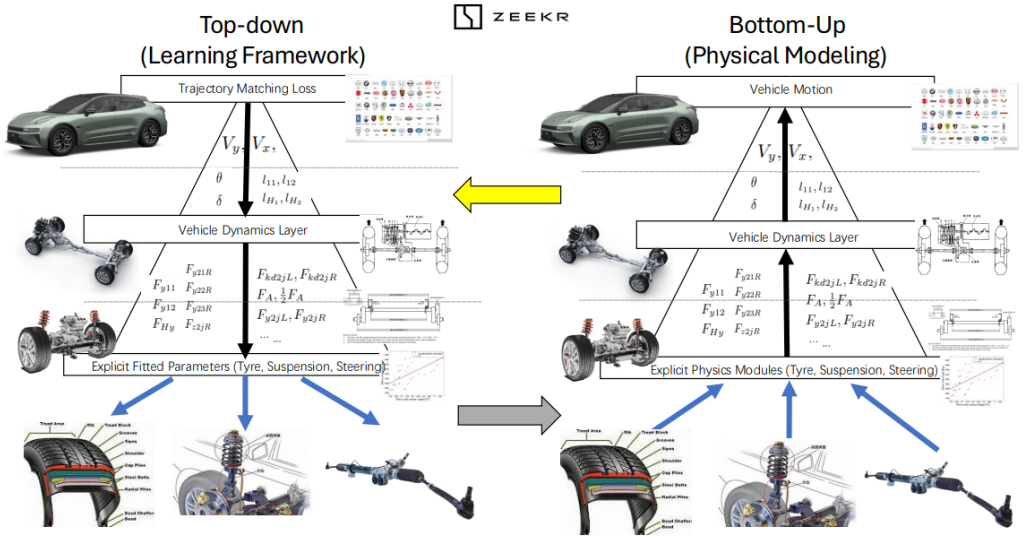

Figure 4.1: Philosophy of bottom-up and top-down

In *PhyAtteN*, the **bottom-up** component corresponds to the forward model: we explicitly encode the tire model, load transfer equations, steering system, and bicycle vehicle dynamics. These blocks form a layered structure from tire up to the full vehicle, capturing only the necessary physical couplings at each stage. This grounded modelling approach reduces the burden on the neural network to "discover" known

[1] `https://www.sveafordon.com`

physics, and ensures that the *PhyAtteN* estimated results remains interpretable and consistent with classical vehicle dynamics.

The **top-down** component comes into play during training. Since production vehicles lack sensors for lateral velocity or normal load, we cannot supervise tyre-level quantities directly. Instead, *PhyAtteN* learns by matching the predicted vehicle trajectory (e.g., yaw rate, lateral speed) to observed data. Through backpropagation, the model adjusts internal parameters—such as tyre coefficients or estimated load transfer—to minimize trajectory error. In doing so, it effectively closes the loop, propagating high-level deviations back into low-level corrections.

By combining strong prior structure with data-driven learning, it reconstructs latent states like lateral force and sideslip without needing additional sensors—making it especially suitable for deployment in production environments.

4.3 PhyAtteN

We introduce the *PhyAtteN*, a physics-informed neural network. Figure 4.2 gives an overview of the full *PhyAtteN* architecture.

The network consists of five macro–blocks, and we will go through each in following sections: 1.*Input Fusion*, 2.*Temporal Encoder & Attention Block*, 3.*Guarded Parameter Heads*, 4.*Physics Rollout Module* and 5.*Output& Loss Function*

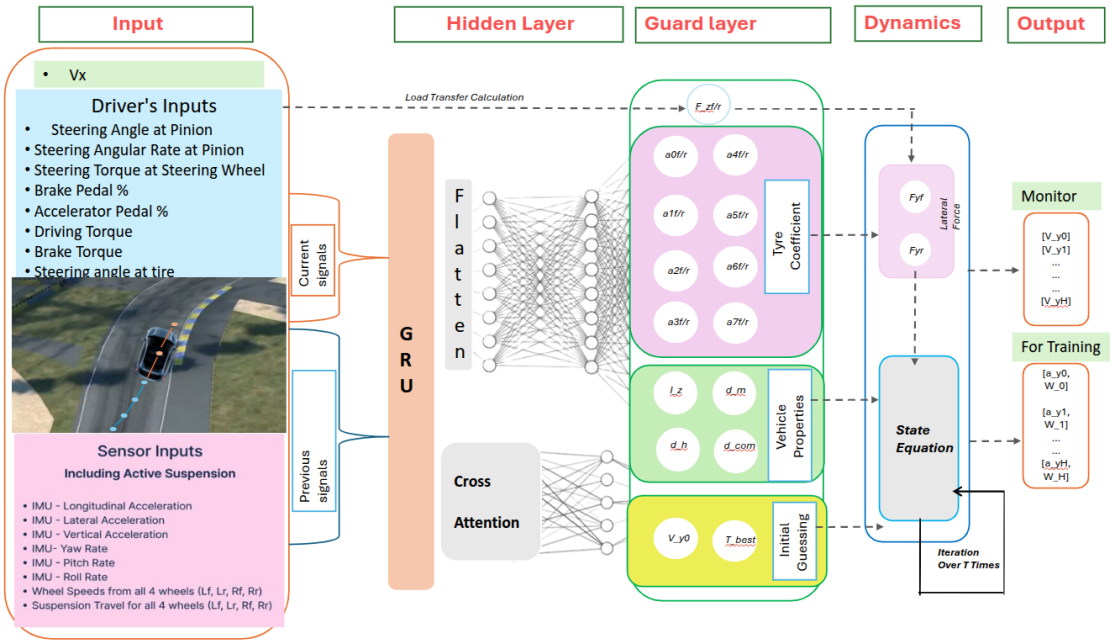

Figure 4.2: Overall architecture of *PhyAtteN*. Guard colors defined following: purple for *tyre coefficients*, green for *vehicle properties*, yellow for *initial state guesses*

1) Input Fusion.

The model ingests two groups of CAN signals sampled at 100 Hz:

- *Driver commands* (cyan block): steering angle / rate / torque, brake and accelerator pedal positions, wheel torques, *etc.*
- *Motion sensors* (magenta block): a_x, a_y, a_z, yaw rate r, pitch and roll, wheel speeds, and suspension travel $\Delta z_{\mathrm{Lf...Rr}}$.

Raw signals from the current time step (t) and the preceding $H-1$ steps are stacked to form a time series $\mathbf{S}_{t-H+1:t} \in \mathbb{R}^{H\times d\mathrm{sig}}$, where H denotes the temporal horizon. This sequence is normalised online using running statistics.

2) Temporal Encoder & Self-Attention.

A multi-layer GRU encodes past dynamics, and attention selects the key time step for accurate estimation.

A multi-layer **GRU** with $d_h = 128$ hidden units is used to extract a latent sequence, which maintains information from previous state. The recurrent structure allows the model to process sequences one time step at a time, maintaining a hidden state that encodes information from previous steps. By stacking multiple layers, the GRU learns hierarchical representations—lower layers capture immediate variations, while higher layers encode more abstract temporal patterns.

The GRU outputs a latent sequence $\{\mathbf{h}_{t-H+1}, \ldots, \mathbf{h}t\} \in \mathbb{R}^{H\times d_h}$, where each vector $\mathbf{h}\tau$ summarizes the information up to time τ. Most importantly, the GRU serves as a temporal feature extractor: it transforms the raw input sequence into a structured representation of system dynamics over the horizon. This latent sequence is then passed to a MLP, which interprets the extracted features to estimate the underlying physical parameters.

Learnable "best–moment" attention.

Figure 4.3 illustrates the attention block that follows the GRU encoder. The hidden sequence $\{\mathbf{h}_{t-H+1}, \ldots, \mathbf{h}_t\}$ from GRU is linearly projected into *keys* and *values*

$$\mathbf{k}_\tau = W^k\mathbf{h}_\tau, \quad \mathbf{v}_\tau = W^v\mathbf{h}_\tau, \quad \text{for } \tau = t-H+1, \ldots, t. \tag{4.8}$$

yielding the matrices $\mathbf{K} = [\mathbf{k}_{t-H+1}\ldots\mathbf{k}_t]^\top \in \mathbb{R}^{H\times d_k}$ and $\mathbf{V} = [\mathbf{v}_{t-H+1}\ldots\mathbf{v}_t]^\top \in \mathbb{R}^{H\times d_v}$.

The query vector $\mathbf{q} \in \mathbb{R}^{d_k}$ is computed as the mean of the GRU outputs across time:

$$\mathbf{Q} = W^q\left(\frac{1}{H}\sum_{\tau=t-H+1}^{t}\mathbf{h}_\tau\right).$$

Intuitively, **query** vector asks *"Which moment in the time series contains the most informative for accurate estimation on initial state"*.

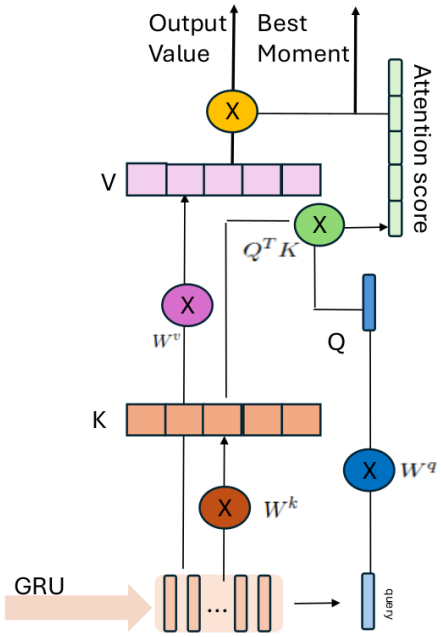

Figure 4.3: Architecture of the self-attention block. The GRU hidden sequence is projected into keys and values. A global query vector, computed as the average of GRU outputs, attends over the sequence to identify the most informative time step. The resulting context vector is passed to the MLP for parameter estimation.

The projection matrix W^k, W^k and W^v is learned jointly with the network, so the model discovers on its own which ambient cues are informative.

Scoring and aggregation. Scaled dot-product attention computes relevance scores

$$e_\tau = \frac{\mathbf{q}^\top \mathbf{k}_\tau}{\sqrt{d_k}}, \qquad \alpha_\tau = \frac{\exp(e_\tau)}{\sum_{\tau'} \exp(e_{\tau'})}, \tag{4.9}$$

and forms the context vector and best moment

$$\mathbf{c} = \sum_{\tau=t-H+1}^{t} \alpha_\tau \, \mathbf{v}_\tau, \qquad \textbf{BestMoment} = argmax(\alpha_\tau) \quad \text{for } \tau = t-H+1, \ldots, t,$$

Because α_τ forms a probability distribution, the attention mechanism acts like a spotlight that automatically concentrates on the *best moment*—ideally somewhere near the middle of the horizon, where contextual information from both past and future is most balanced and informative for estimation.

Output context of attention $\mathbf{c}$ that is passed to the downstream MLP for initial state estimation.

3) Guarded Parameter Heads.

The dense backbone branches into three heads:

(a) **Tyre-coefficient head** outputs raw logits parameters $\tilde{a}_{f/r} \in \mathbb{R}^9$ per axle.

(b) **Vehicle-property head** regresses delta mass, delta height of COM and delta position of COM (d_m, d_h, d_{com}) and yaw inertia $\tilde{I}_z$.

(c) **Initial-state head** proposes latent variables $[\tilde{v}_y^{(0)}, T_{\text{best}}]$ used to warm-start the rollout.

All raw logits are squashed through a *Physics Guard*:

$$\hat{\theta} = sigmoid(\tilde{\theta}) \ * \left(\theta_{\max} - \theta_{\min}\right) + \theta_{\min},$$

where $(\theta_{\min}, \theta_{\max})$ are literature ranges (Table 5.5). This guarantees physically plausible parameters throughout training.

4) Physics Roll-out Module (Discrete Integrator).

The final block advances the single–track dynamics by explicitly integrating lateral and yaw motions over T steps with a fixed sample time dt.

(i) Adaptive vehicle constants. The green head in Figure4.3 supplies batch-specific corrections $\{d_m, d_h, d_{\text{com}}, I_z\}$ to the nominal mass, CG height and axle distances. These slow-varying bias allow the model to reflect payload shifts. We will go to detail in section **??** to introduce how we constraint this.

(ii) Simple load transfer. At each time step the longitudinal acceleration a_x drives a simple analytic formula $\Delta F = m\, a_x\, h/L$ that redistributes the static axle loads into $F_{z\text{f}}$ and $F_{z\text{r}}$. Both loads are fed into the Guard layer, which converts the learned $\{a_0 : a_8\}$ into physically consistent (B, C, D, E) coefficients for the front and rear tyres.

(iii) Lateral force and yaw dynamics. Slip angles are evaluated with the ISO arctangent formulation, tyre forces follow the Pacejka '89 curve, and the single-track equations yield $a_y,\ \dot{r},\ \dot{v}_y = a_y - v_x r$. A forward-Euler step with d$t = 10\,\text{ms}$ advances v_y and the yaw rate.

(iv) Low-speed stabilisation. When $v_x < 3\,\text{m/s}$ the integrator zeroes $a_y, \dot{r}, \dot{v}_y$ to avoid division by a near-zero forward speed and to dampen numerical noise during stand-still manoeuvres.

(v) Per-sample "best-index" reset. Each sample in the batch may begin integration at a different index $t^\star$ (determined by the attention block). The initial lateral speed $v_y^{(0)}$ is injected at $t^\star$, and the integrator rolls forward for the remaining $T - t^\star$ steps. The last H frames are retained for the rollout loss, ensuring that gradients always flow through a fixed-length segment irrespective of $t^\star$.

5) Output & Loss Function.

At each forward pass the network produce state

$$\hat{\mathbf{X}}_{t+\tau+1} = \begin{bmatrix} a_{y,t+\tau+1} \\ \hat{v}_{y,t+\tau+1} \\ \hat{r}_{t+\tau+1} \end{bmatrix}$$

the guarded tyre/tire coefficients $\{a_0{:}a_8\}_{\mathrm{f/r}}$, and the vehicle-geometry deltas $\Big(d_m,\ d_h,\ d_{\mathrm{com}},\ I_z\Big)$.

During training we supervise *only* the observable channels— lateral acceleration $\hat{a}_y$ and yaw-rate $\hat{r}$. Sideslip v_y is allowed to roam free, so the model can reconcile its internal mechanics with the measured trajectory without being shackled to a noisy proxy of v_y, a pragmatic concession from a production point of view, where a reliable lateral-velocity sensor is absent.

To support batch learning we trim the first $t^\star$ steps of every roll-out (the sample-specific `best_idx`) and evaluate the discrepancy over the remaining fixed window of length $H - t^\star$. The data term is the sum of root-mean-square errors:

$$\mathrm{RMSE}_{a_y} = \mathrm{RMSE}\Big(\hat{a}_y,\, a_y\Big), \qquad \mathrm{RMSE}_r = \mathrm{RMSE}\Big(\hat{r},\, r\Big).$$

Regularisers.

- **Variance loss**
$$\mathcal{L}_{\mathrm{var}} = \mathrm{Var}\Big[a_{0:8}\Big]_{\mathrm{f/r}} + \mathrm{Var}\Big[I_z\Big]$$
keeps the 18 pacejka coefficients and the yaw inertia nearly invariant across the batch, respect to the physical belief that rubber properties and inertial mass hardly fluctuate within a few hundred milliseconds.

- **Delta-penalty**
$$\mathcal{L}_\Delta = w_{dm}\,\mathrm{Var}\Big(d_m^2\Big) + w_{dh}\,\mathrm{Var}\Big(d_h^2\Big) + w_{d_{\mathrm{com}}}\,\mathrm{Var}\Big(d_{\mathrm{com}}^2\Big)$$
discourages sample-to-sample drift in mass, CG height, and fore–aft balance.

Total objective.

$$\mathcal{L} = \mathrm{RMSE}_{a_y} + \mathrm{RMSE}_r + \mathcal{L}_{\mathrm{var}} + \mathcal{L}_\Delta.$$

The composite loss is minimised using *truncated Back-Propagation-Through-Time (BPTT)*: the discrete integrator unrolls the bicycle dynamics for T steps, and gradients are then propagated backwards through the entire unrolled graph before being clipped and applied. This truncated-BPTT strategy captures sufficient temporal context for the network to self-correct, while keeping memory usage reasonable for training.

However, BPTT can introduce troubles. Repeated Jacobian products across time may, in principle, amplify gradients and trigger exploding updates—especially when terms like $\arctan\left((v_y \pm l\, r)/v_x\right)$ are evaluated at low vehicle speeds, or when the learned tyre parameters push towards the edges of their admissible bounds.

To counteract this, we apply two safeguards: (i) a global gradient norm clipping operation (`clip_grad_norm()`) after each backward pass, and (ii) a low-speed mask

in the integrator that nullifies updates when $v_x < 3\,\mathrm{m/s}$. Together with the physics-based guard layers, these mechanisms keep training numerically stable without sacrificing convergence.

Additionally, we found that the **variance constraints** play a critical role in guiding the model's attention during learning. Without these regularisers, the model becomes overly flexible and tends to "solve" the trajectory matching task by dynamically adjusting the tyre parameters $\{a_0{:}a_8\}_{f/r}$, which are meant to remain constant across short time horizons—rather than properly inferring the unobserved latent variable $v_y^{(0)}$.

When Attention block becomes "LAZY", leads to unstable or uninterpretable coefficient profiles across batches and undermines the physical integrity of the model. To keep attention block "LIVELY", enforcing batch-level consistency through variance loss, we effectively *force the attention mechanism to focus on correcting the initial hidden state* (i.e., estimating the correct sideslip) rather than warping the physical layer. In this sense, the variance constraint not only reflects physical intuition but also acts as a soft form of supervision on the model's internal logic.

5

Experiment & Result

The experiments put PhyAtteN through both high-fidelity IPG CarMaker manoeuvres and real-world runs on two Zeekr EVs, stressing everything from double-lane changes to urban traffic, Across all cases the grey-box network tracks yaw-rate, lateral acceleration and the hidden sideslip more faithfully than a larger LSTM baseline. Ablation studies show the attention-based best-moment initialisation, the physics-guard layer and the lightweight parameter-reduction trick each play a critical role in keeping the model stable, compact and physically consistent.

5.1 DataSets

This section describes the two categories of data employed throughout the project: *simulation* data generated with IPG CarMaker, and *real-world* data acquired from *Zeekr vehicle D* and *Zeekr vehicle B*.

5.1.1 Simulation

The simulated vehicle in IPG CarMaker was configured using the default Demo_IPG_CompanyCar_EV_AssetInfo package. Key parameters of the vehicle model include a total mass of 2100kg and a wheelbase of 2.9m, with the center of gravity located 1.3m behind the front axle and 0.55m above ground.

Table 5.1: Key vehicle parameters used in IPG CarMaker simulation.

Parameter	**Symbol**	**Value**
Vehicle mass	m	2100 kg
Wheelbase	L	2.9 m
CG location (front axle)	l_f	1.3 m
CG height	h_{cg}	0.55 m
Track width (front/rear)	w_f/w_r	1.65 m / 1.65 m
Tire model	—	`RT_255_35R20_p2.50`
Drive type	—	Rear-wheel drive (RWD)
Steering	—	Electric power steering

A high-fidelity FTire model `RT_255_35R20_p2.50` was used to capture realistic

tire dynamics. The vehicle employs rear-wheel drive, electric power steering, and symmetric front/rear track widths of 1.65m. All simulations were conducted with the integrated vehicle dynamics control disabled to ensure unfiltered dynamic response.

Training and Evaluation Scenarios.

Table 5.2 lists the ten manoeuvres used to train the model, grouped into standard vehicle dynamics tests and public road driving scenarios. The vehicle dynamics tests—such as lane change procedures (ADAC, AMS, VDA), slalom, steady-state cornering, and steer step—are specifically designed to push the vehicle towards its handling limits. These aggressive manoeuvres are crucial for learning model parameters under near-limit conditions where nonlinearities become prominent.

In contrast, the normal public driving scenarios reflect unstructured, real-world-like environments, including country roads, city tunnel driving, and lap-time optimization laps. All manoeuvres were initially simulated under nominal dry asphalt conditions ($\mu = 1$), with additional sets generated for reduced friction ($\mu = 0.8$) by altering only the tyre–road friction coefficient, preserving all other simulation parameters.

Table 5.2: Manoeuvres used in the simulation training dataset. The **Vehicle Dynamics Tests** include standardized lane change scenarios defined by different organizations: *ADAC* (Allgemeiner Deutscher Automobil-Club), *AMS* (Auto Motor und Sport), and *VDA* (Verband der Automobilindustrie).

Category	**Scenario**
Vehicle Dynamics Tests	Lane Change (ADAC)
	Lane Change (AMS)
	Lane Change (VDA)
	Slalom 18 m
	Steady-State Circular (100 m, right)
	Steer Step (exclude $mu = 0.8$)
High-Defination Scenarios(exclude $mu = 0.8$**)**	Country Road
	City Tunnel
	Lap Time Optimization

Table 5.3 outlines the four manoeuvres held out for evaluation. These include the ISO double-lane change, sinusoidal steering input, a tighter steady-state circular test (42 m radius), and a high-demand urban test inspired by Chinese tollway traffic. As with the training set, each scenario was simulated under both $\mu = 1$ and $\mu = 0.8$ to assess generalisation across varying grip levels.

5.1.2 Real-World Datasets (Zeekr vehicle D & Zeekr vehicle B)

This dual-platform dataset challenges PhyAtteN's ability to generalise across both RWD and AWD powertrain dynamics, as well as from structured proving-ground

Table 5.3: Manoeuvres used in the simulation evaluation dataset.

Category	Scenario
Vehicle Dynamics Tests	Lane Change (ISO Double-Lane) Sinusoidal Steering Steady-State Circular (42 m, left)
High-Defination Scenario(exclude $mu = 0.8$**)**	Chinese Tollway

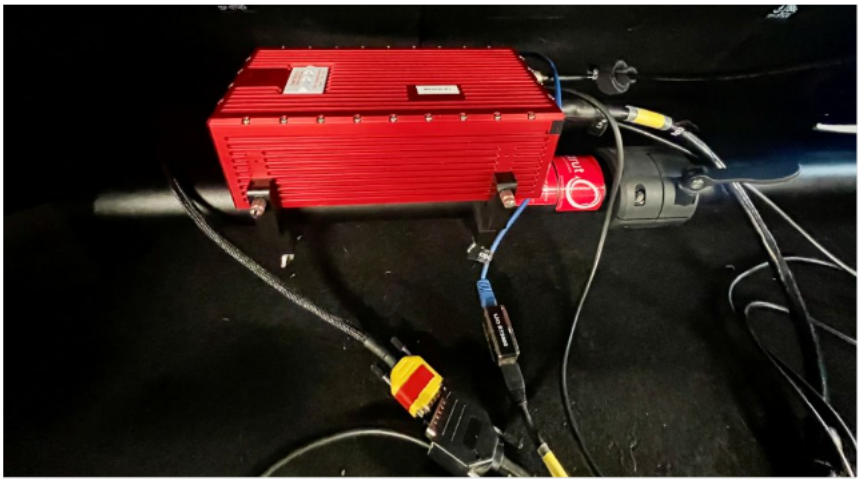

Figure 5.1: Mounted external OXTS RT3000 to measure high-accuracy ground truth

manoeuvres to the variability of real-world driving—all while relying solely on CAN and FlexRay sensor suites. High-accuracy ground truth is provided by the externally mounted OXTS RT3000 GNSS/INS system.

Vehicle platforms. Data were collected on two production-level electric vehicles from Zeekr:

- **Zeekr vehicle D** – rear-wheel-drive (RWD) long-range trim, curb mass 2340 kg;
- **Zeekr vehicle B** – dual-motor all-wheel-drive (AWD), curb mass 1960 kg.

Both cars run the same 19-inch winter tyres.

Instrumentation and logging.

Both cars stream their primary signals—driver inputs, wheel speeds, IMU, and suspension travel over the native CAN and FlexRay buses, which are logged synchronously with a *Vector CANoe 15.0* .

An external **OXTS RT3000** GNSS/INS unit was rigidly mounted on the trunk of each vehicle to provide high-accuracy ground truth. The RT3000 fuses RTK-corrected GPS with inertial measurements to output position, velocity, and orientation at 100 Hz, achieving centimetre-level accuracy in position and sub-degree heading resolution. Its 1PPS signal was used to synchronise all CAN and FlexRay data streams. This setup yields a single, tightly aligned dataset per run—with motion dynamics, control inputs, and reference states all synchronised and ready for post-processing.

Test locations.

Data were recorded at:

- **Proving ground**, dry asphalt, ambient 10 ± 5 °C;
- **Gothenburg public-road** – 3 km of urban 60 km h^{-1} traffic with ring-road on-ramps.

Driving Manoeuvres.

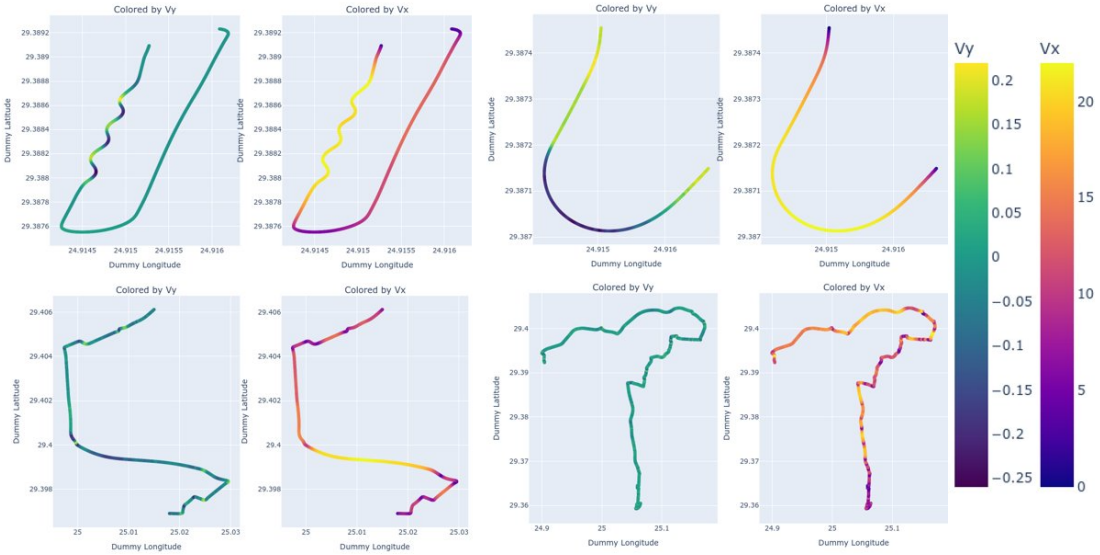

Figure 5.2: The top row shows vehicle trajectories during controlled dynamic maneuvers, while the bottom row depicts real-world public driving. The trajectories are colored based on lateral velocity v_y (left panels) and longitudinal velocity v_x(right panels). True X, Y label was edit due to confidential reason.

The real-world dataset covers both controlled vehicle-dynamics tests and unconstrained public-road driving. As shown in Table 5.4, the training set includes single-lane change and slalom runs conducted at Airport proving ground, along with a baseline urban-driving log from Gothenburg traffic. For evaluation, we use a double-lane change test, Steady Cornering and a second round of public driving under more varied surface conditions. This combination ensures that the model is exposed to both structured limit-handling scenarios and the noise and variability typical of production deployments.

Table 5.4: Manoeuvres used in the real-world dataset.

Set	Scenario
Training	Single-Lane Change
	Slalom
	Public Driving 1
Evaluation	Double-Lane Change
	Steady Cornering
	Public Driving 2

5.2 Evaluation Metrics

To give a balanced picture of performance we report two complementary families of metrics: *data–fit* and *computational*. All metrics are computed on the last $H-t^\star$ frames of each roll-out (cf. Section 4.3) and then averaged over the full test set.

5.2.1 Data–fit metrics.

- **RMSE**$_{a_y}$

$$\mathrm{RMSE}_{a_y} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{a}_{y,i} - a_{y,i}\right)^2}$$

  where N is the number of frames in the evaluation window and a_y is expressed in $\mathrm{m/s^2}$.

- **RMSE**$_r$ Same definition applied to yaw-rate r (unit: $\mathrm{rad\,s^{-1}}$).

- **RMSE**$_{v_y}$ Same definition applied to lateral velocity r (unit: m/s).

5.2.2 Computational metrics

- **Forward–pass latency (excluding roll–out)** Mean inference time per frame, benchmarked on an NVIDIA Quadro RTX4000. NVIDIA Quadro RTX4000 is a professional-grade Turing-architecture GPU with 8,GB GDDR6 memory, 2304 CUDA cores, and dedicated Tensor and RT cores.

- **Floating–point operations (FLOPs, excluding roll–out)** Calculated with `fvcore` to indicate scalability.

5.3 Simulation Results

5.3.1 Experimental Setup

Scenarios and grip levels. Evaluation is conducted on the four hold-out manoeuvres listed in Table 5.3, spanning both structured vehicle dynamics tests and a high-definition real-world scenario.

Each manoeuvre is simulated under two distinct surface conditions: nominal dry asphalt ($\mu = 1.0$) and reduced grip ($\mu = 0.8$). This dual-friction setup allows us to assess the model's ability to generalise across varying traction levels.

Evaluation metrics. We report performance using the metrics defined in Section 5.2. For completeness, the following signals are evaluated:

- **Lateral acceleration error** RMSE_{a_y}, expressed in $\mathrm{m/s^2}$;
- **Yaw-rate error** RMSE_r, in $\mathrm{rad\,s^{-1}}$;
- **Lateral velocity error** RMSE_{v_y}, in m/s.

Each RMSE is computed over the last steps of a rollout and averaged across all trajectories in the evaluation set.

Physical-parameter search space. Table 5.5 lists the prior bounds imposed on the Pacejka tyre coefficients and the vehicle yaw inertia I_z during simulation-level identification. All Δ-terms are fixed to 0, since we could get exactly value from CarMaker, therefore omitted.

Table 5.5: Search ranges for the physical parameters used in the simulation study. Values outside these bounds are *clipped* during optimisation to guarantee physically plausible behaviour.

Parameter	Front axle		Rear axle	
	Min	Max	Min	Max
a_0	1.0	3.0	1.0	3.0
a_1	−40	40	−40	40
a_2	0	3000	0	3000
a_3	0	3000	0	3000
a_4	0	5	0	5
a_5	0	2	0	2
a_6	−1	1	−1	1
a_7	−1	1	−1	1
a_8	0	2	0	2
I_z	1200 – 3000			

Model configuration. Unless otherwise stated, **PhyAtteN** is instantiated with a hidden state of **32** units.

Each forward pass ingests a fixed *temporal horizon* of **50** past frames (0.5 s at 100 Hz). The self–attention block is allowed to attend over the first **30** timestamps to capture the richest transient information, while the remaining **20** timestamps to processed through the integration head. The model use Adam($\mathrm{lr} = 10^{-4}$) and batch size of 64 for 500 epochs. All results reported below use this single hyper-parameter setting; no post-hoc tuning was performed on the evaluation set.

Baseline model. To illustrate the benefit of incorporating physical priors, we compare *PhyAtteN* against a *purely data-driven* sequence model.

A three-layer LSTM with 128 hidden units per layer and a linear head that predicts the two target signals $\{a_y, r\}$ at every time-step.[1]

The LSTM sees exactly the same input tensors and training splits as *PhyAtteN*; no additional tuning is permitted.

5.3.2 Aggregate Performance

All results reported in this section are obtained from the aggregate simulation–test set summarised in Table 5.3. Both the LSTM baseline and our *PhyAtteN* model are trained on the manoeuvre catalogue in Table 5.2 at a uniform friction level of $\mu = 1$.

Table 5.6: Root-mean-square error (RMSE) on the held-out simulation set. A dash indicates the quantity is not predictable by the model.

Model	RMSE_{v_y}	RMSE_{a_y}	RMSE_r
PhyAtteN (ours)	0.040	0.080	0.002
LSTM (purely data-driven)	–	0.497	0.041

Overall quantitative errors. Table 5.6 lists the root–mean–square errors (RMSE) achieved by each model.[2]

- Across all observable states, **PhyAtteN** attains the lowest error. In particular, it reduces the yaw–rate RMSE by ~95% (0.002 vs. 0.041) and the lateral–acceleration RMSE by ~84% (0.080 vs. 0.497) compared with the purely data-driven LSTM.
- Unlike the LSTM, **PhyAtteN** also recovers the hidden state v_y (lateral velocity) with an RMSE of only 0.040,ms^{-1}, highlighting the benefit of embedding vehicle dynamics priors.

Yaw-rate r and lateral-acceleration a_y (Figure5.3). Figure5.3 overlays the predicted trajectories (orange: **PhyAtteN**; green: LSTM) against the ground truth (black) for a representative double-lane-change manoeuvre.

1. **Global trend fidelity.** Both models follow the coarse evolution of the signals; however, the PhyAtteN trace stays visually glued to the ground-truth curve, whereas the LSTM begins to diverge as dynamic loading increases.
2. **Transient accuracy.** The red call-outs highlight sharp steering reversals that excite the vehicle's lateral dynamics. Here, the LSTM incurs large overshoots

[1] The network is trained with an ℓ_2 loss on normalised targets, using Adam ($\text{lr} = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$) and a batch size of 64 for 500 epochs.

[2] Lower values indicate better performance. A dash ("–") denotes that the quantity is ,*not* predicted by the corresponding model.

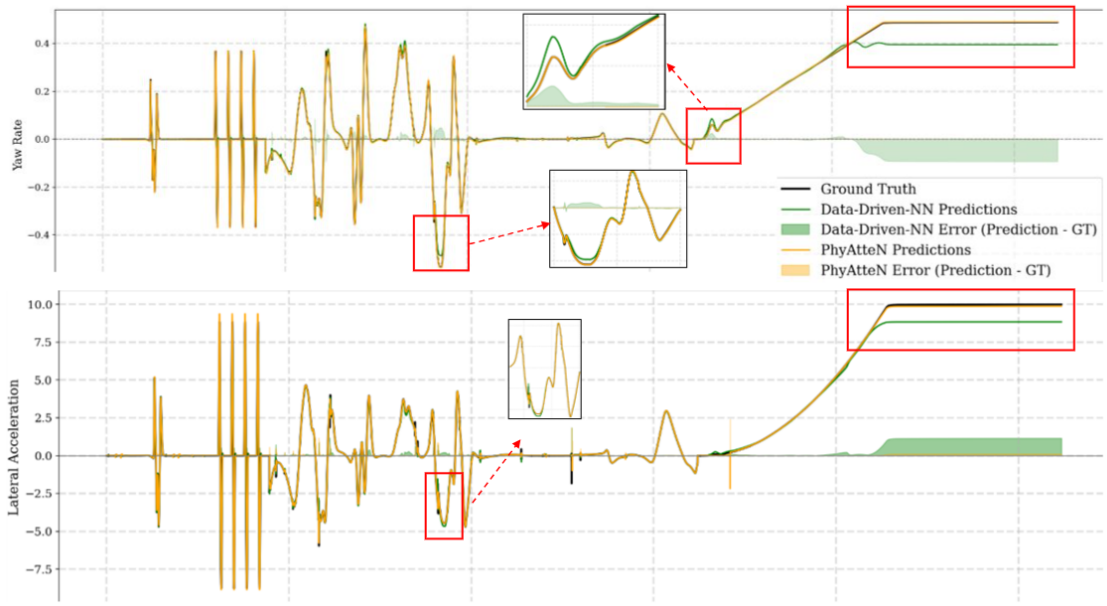

Figure 5.3: Yaw-rate r (top) and **lateral-acceleration** a_y (bottom) for the same run

and phase lags, resulting in the wide green error envelope. PhyAtteN, aided by its physics-aware attention mechanism, captures the amplitude and phase of these transients much more faithfully.

3. **Steady–state circular tracking.** In the final segment the vehicle transitions into a constant–radius, steady–state cornering phase (cf. ISO 4138 open–loop test). **PhyAtteN** sustains accurate estimation on lateral acceleration and yaw rate , whereas the LSTM gradually drifts and under–predicts both signals, causing the widening green envelope.

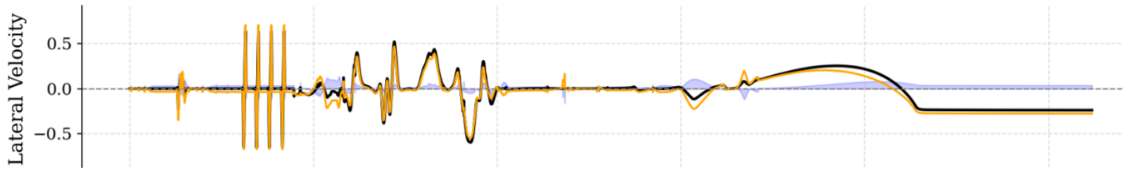

Figure 5.4: Lateral Velocity v_y for the same run.

Hidden-state recovery—lateral velocity v_y (Figure 5.4).

Because v_y is *not* supplied as a supervised target during LSTM training, the baseline cannot produce a prediction (hence the shaded blue confidence band only). PhyAtteN, on the other hand, reconstructs a physically plausible v_y trajectory that almost overlays the ground truth throughout the manoeuvre. The tight alignment confirms that the model's internal state remains consistent with vehicle dynamics.

5.3.3 Scenario-by-Scenario Breakdown

Table 5.7: Per-scenario performance at $\mu = 1.0$ on the four held-out manoeuvres. Best value per column in **bold**.

Scenario	$\mathbf{RMSE}_{a_y}$ [m/s^2]	$\mathbf{RMSE}_{r}$ [rad/s]	$\mathbf{RMSE}_{v_y}$ [m/s]
Lane Change (ISO Double-Lane)	**0.046**	**0.00186**	**0.036**
Sinusoidal Steering	0.144	0.00220	0.040
Steady-State Circular (42 m, left)	0.083	0.00189	0.046
Chinese Tollway (HD)	0.056	0.00246	0.040

Across all three state variables, the model performs best on the aggressive ISO double-lane change. Sinusoidal steering shows the highest a_y RMSE.

Notably, our model maintains strong performance in the steady-state cornering scenario, where inputs are quasi-constant and frequency content is low—demonstrating that PhyAtteN generalises not only to dynamic transients but also to stable-limit behaviour.

Performance on the unstructured Chinese Tollway scenario remains competitive, confirming the model's robustness in high-definition environments beyond standard tests.

5.3.4 Self-Correction after the Attention Horizon

Although the open-loop hand-off at $t = BestMoment$ may deliver a biased snapshot of the hidden state v_y, two design choices ensure that the bias is flushed out almost:

1. **Accurate parameters before accurate states.** The 30-frame attention block is optimised *primarily* to infer the *time-invariant* parameters that govern lateral dynamics—most notably the Pacejka coefficients $\{a_0{:}a_8\}$ and yaw inertia I_z. Table 5.8 shows these parameters converge to physically plausible values with low variance. By the time control passes to the integration head, the model is already marching under "the correct set of equations."

2. **Physics-guarded integration closes the loop.** During the last 20 timestamps integration horizon the GRU propagates the latent state forward while a lightweight guard layer enforces the coupling $\left(v_y,\ r,\ \alpha,\ F_y,\ a_y\right)$ implied by the learned coefficients.

   Any mismatch between the predicted lateral acceleration $\hat{a}_y$ and the measured a_y generates a residual $\varepsilon = a_y - \hat{a}_y$. Because $\hat{a}_y := F_y/m$ and F_y depends on v_y through the Magic-Formula, the residual back-propagates into the hidden state, nudging v_y toward the physically admissible trajectory. With correct coefficients, this feedback is strongly convergent, eliminating ε.

In short, the network first *locks in the physics* (accurate coefficients), then lets those physics *re-lock the state*. That is why a poor initial guess is harmless: the physically

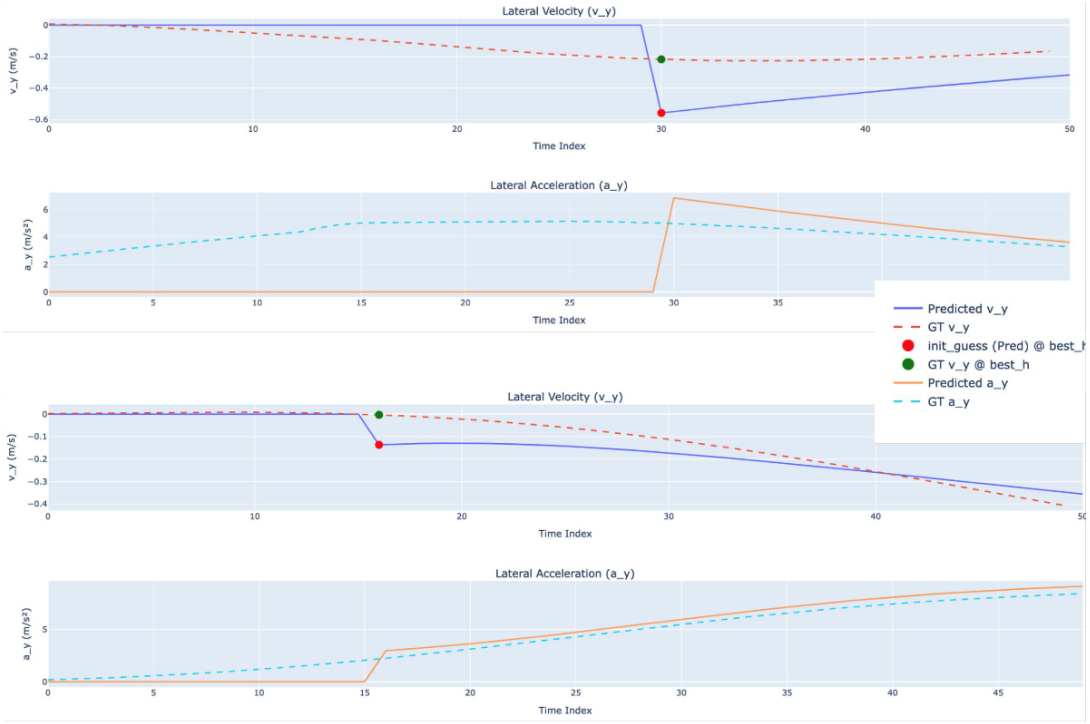

Figure 5.5: Two examples of self-correction. **Top:** Lateral velocity v_y (blue = prediction, red = initial guess at t=$BestMoment$, green = ground truth at t=$BestMoment$, dashed = ground truth over time). **Bottom:** Lateral acceleration a_y (orange = prediction, cyan = ground truth).

consistent integration phase simply "bleeds out" the error.

5.3.5 Model Coefficient Analysis

To investigate whether the model learns physically plausible behaviour, we extract the estimated Pacejka coefficients $\{a_0{:}a_8\}$ from the guarded output head across the evaluation set and reconstruct the corresponding lateral force–slip curves.

Interpretation.

- *Consistency.* Standard deviations are tiny—typically $<1\%$ of the mean—indicating the guard layer keeps parameters tightly constrained across the whole test corpus.
- *Front vs. rear.* The rear tyres exhibit a much steeper stiffness factor (a_1) and larger curvature magnitude (a_7), which aligns with an RWD vehicle where the rears run closer to their friction limit.

Figure 5.6 visualises the reconstructed lateral force–slip curves using the learned coefficients under three different vertical loads. The curves exhibit the characteristic S-shape seen in physical tyre models—initial linear buildup, followed by soft saturation and mild asymmetry. As vertical load increases from 4 kN to 8 kN, both

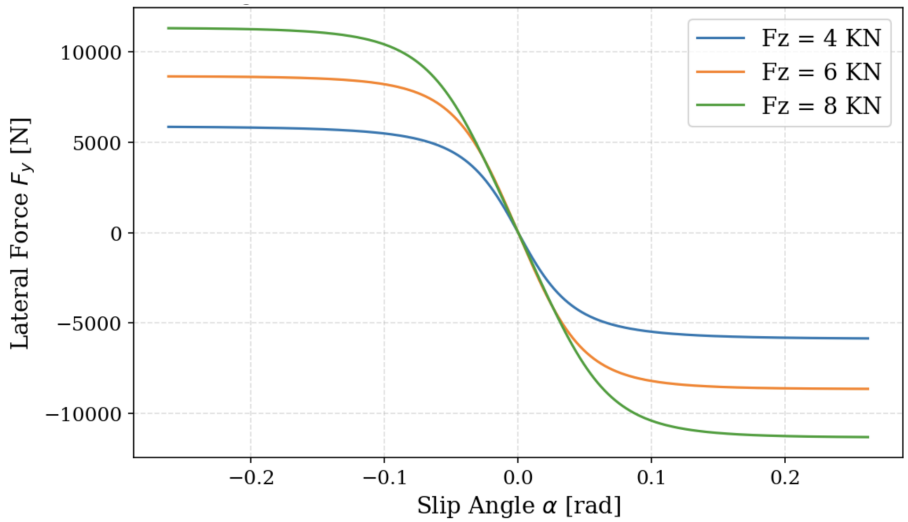

Figure 5.6: Reconstructed lateral force–slip curves from the learned $\{a_i\}$ at three vertical loads ($F_z = 4,\ 6,\ 8$ kN). The expected S-shape and load-dependent peak force are clearly preserved, confirming physical plausibility of the estimated parameters.

peak force and cornering stiffness increase, which aligns with known tire dynamics and matches the trends observed in empirical plots like Fig. 2.2. This consistency reinforces the physical plausibility of the estimated parameters, despite the absence of direct supervision on slip angle or lateral force.

Friction–dependent shape factors

Tyre tests and simulation studies [28, 29] consistently report that higher road friction yields a steeper initial cornering stiffness B and a larger peak-force scale D. To verify that the network has internalised the same trend, we split the evaluation set into **dry** ($\mu \sim 1.0$) and **wet/medium-μ** ($\mu \sim 0.8$) and recover the Magic-Formula parameters B, C, D, E for each tyre.

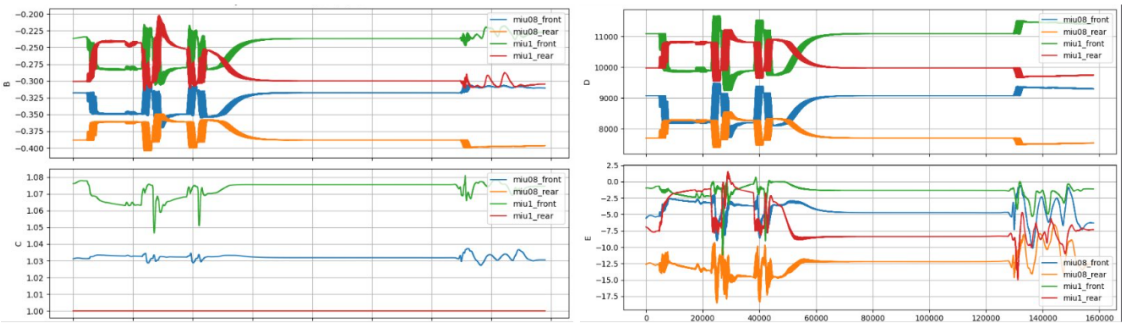

Figure 5.7: Time-series of estimated shape factors B, C, D, E under different friction conditions. The curves correspond to: μ=0.8 (front: blue, rear: orange) and μ=1.0 (front: blue, rear: red).

Interpretation.

Table 5.8: Learned Pacejka coefficients a_0:a_8 (mean ± std) on the evaluation set.

Coeff.	Front (mean ± std)	Rear (mean± std)
a_0	1.074 ± 0.006	1.000 ± 0.000
a_1	-13.14 ± 0.02	-39.97 ± 0.00
a_2	$1\,519.32 \pm 0.01$	$1\,582.36 \pm 0.01$
a_3	$2\,999.97 \pm 0.00$	$3\,000.0 \pm 0.00$
a_4	3.989 ± 0.019	4.998 ± 0.000
a_5	0.395 ± 0.012	0.173 ± 0.009
a_6	-0.014 ± 0.018	-0.035 ± 0.037
a_7	-0.133 ± 0.018	-0.501 ± 0.016
a_8	0.932 ± 0.008	0.329 ± 0.002

- *Stiffness factor B.* Dry asphalt produces a noticeably less negative (steeper) B than the medium-μ surface, corroborating the classic finding that $B \propto \mu$ for a fixed vertical load [28].
- *Peak scale D.* The dry-road D values are roughly 35% higher than on the wet laps, aligning with the simple Magic-Formula relation $D = \mu F_z$. In our single-track model, we combine 2 front/rear tyre to one, so leads bit high value in D.
- *Why the traces are thick.* The width of each coloured band stems from the analytic load-transfer estimate used inside the guard head: as load shifts, both B and D vary linearly with F_z, creating a visible envelope even though the underlying means stay stable.

5.4 Ablation Study

Parameter–reduction (`_RDS`)

Our baseline head ingests the *entire* sequence of GRU hidden states, which must be flattened before the final multilayer perceptron (MLP). To cut the parameter budget we introduce a switch, denoted `_RDS`, that forwards *only the last hidden vector*—effectively a "last-step pooling" of the recurrent output. The change eliminates most of the input channels to the MLP while leaving all recurrent dynamics and the attention block untouched.

Discussion.

- **Dramatic model compression.** Forwarding a single hidden vector slashes the parameter count by $\sim$77%.
- **Mixed accuracy trade-off.** The yaw-rate error remains identical, and lateral-velocity error *improves* from 0.0725 to 0.0422 . Lateral-acceleration RMSE,

Table 5.9: Impact of the parameter-reduction trick on the baseline configuration ($\mu = 1$). Horizon H=50, hidden size d=32, integration window t=20, batch-size BC=2048.

Model	Params [M]	Time [ms]	FLOPs[M]	RMSE$_{a_y}$	RMSE$_r$	RMSE$_{v_y}$
H50_HD32_LST20	0.519	2.563	17.2	0.0626	0.00219	0.0725
H50_HD32_LST20_**RDS**	0.117	2.541	4.48	0.0802	0.00219	0.0422

however, degrades by roughly 0.018.

- **Interpretation.** The full-sequence flattening evidently helps the MLP capture fine-grained variations in a_y. The tighter v_y improvement suggests that the compacthead may regularise over-fitting

Overall, `_RDS` delivers a four-times smaller network with near-identical yaw-rate fidelity and a gain in lateral-velocity accuracy, at the cost of a modest hit in lateral-acceleration. Depending on deployment constraints, the compressed variant is a strong candidate for real-time embedded implementation.

Hidden-size sweep

We examine how enlarging the GRU hidden state from $d = 32$ to $d = 128$ affects performance when the parameter-reduction (`_RDS`) is active. All other hyperparameters are held fixed: horizon $H = 50$, integration window $t = 20$, batch-size BC = 2048.

Table 5.10: Influence of GRU hidden size with `_RDS` enabled ($\mu = 1$).

Model	Params [M]	Time [ms]	FLOPs [M]	RMSE$_{a_y}$	RMSE$_r$	RMSE$_{v_y}$
H50_HD32_LST20_RDS	0.117	2.541	4.48	0.0802	0.00219	0.0422
H50_HD**128**_LST20_RDS	0.612	3.889	31.002	0.0626	0.00473	0.0395

Discussion.

- **Lateral-acceleration improves.** Increasing the hidden width cuts RMSE$_{a_y}$ from 0.0802 to 0.0626 (-22 %).
- **Yaw-rate degrades.** RMSE$_r$ more than doubles (0.00219 $\rightarrow$ 0.00473 rad s^{-1}), but it is still in very high accuracy.
- **Lateral-velocity gains are modest.** RMSE$_{v_y}$ tightens slightly (-6%).
- **Cost–benefit.** The wider GRU multiplies parameter count by 5.2$\times$ yet yields mixed accuracy: clear benefit for a_y, neutral for v_y, and harmful for r. Unless yaw-rate precision can be recovered, the smaller $d = 32$ model remains the more balanced choice.

Attention block

The baseline encoder is followed by a lightweight multi-head attention layer that re-weights the final GRU outputs before they enter the MLP. Here we ablate that layer (`_WTATT`); the variant still predicts the full state vector, including an explicit estimate of v_y at $t = 0$.

All remaining hyper-parameters match the previous experiment (horizon $H = 50$, hidden $d = 32$, window $t = 20$, parameter-reduction on, batch-size BC = 2048).

Table 5.11: Effect of removing the attention block ($\mu = 1$).

Model	Params [M]	RMSE$_{a_y}$	RMSE$_r$	RMSE$_{v_y}$
H50_HD32_LST20_RDS	0.117	0.0802	0.00219	0.0422
H50_HD32_LST20_RDS_**WTATT**	0.115	0.0810	0.00193	0.1000

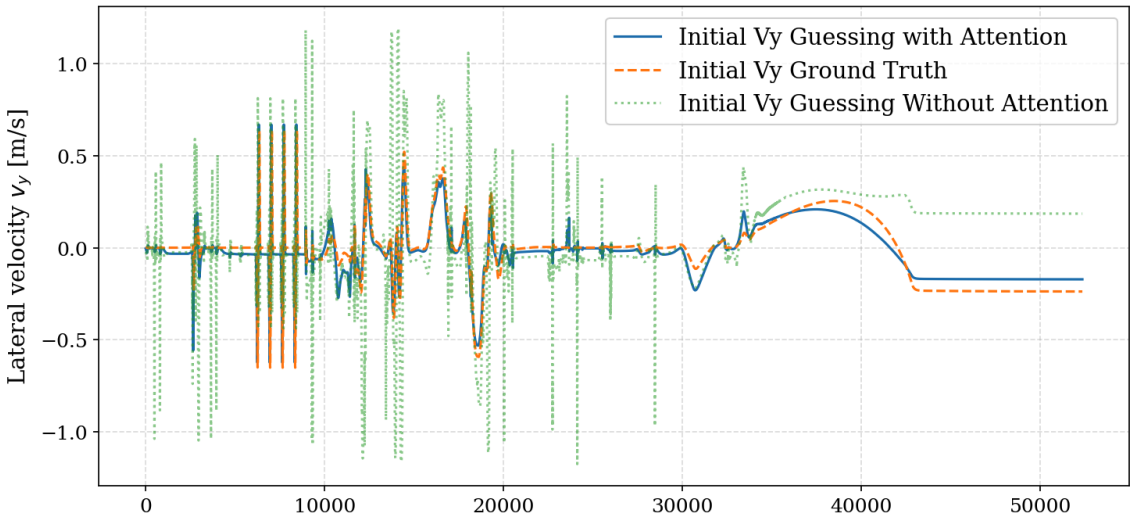

Figure 5.8: Initial lateral-velocity v_y reconstruction on a dry-asphalt lap. With attention (blue solid), the estimate tracks the ground truth (orange dashed) closely. Without attention (`_WTATT`, green dotted), large excursions appear and a steady bias builds up during the manoeuvre around samples 35 000–40 000.

Discussion.

- **Lateral-velocity suffers without attention.** RMSE$_{v_y}$ more than doubles, and Fig. 5.8 shows pronounced spikes and drift once dynamic loading increases. While the recurrent integration will *heals* the trajectory a bit, an accurate initial guess of $v_y\big(t{=}0\big)$ remains crucial—errors at this stage bias the subsequent optimisation of the Pacejka coefficients. The attention block is therefore essential for fusing early-sequence cues into a coherent and physically consistent v_y estimate.
- **Lateral-acceleration unchanged.** The 0.0008 rise in RMSE$_{a_y}$ is negligible relative to sensor noise.

- **Parameter footprint stable.** Dropping the attention layer saves only 1.8 % parameters because the layer was already extremely compact.

In summary, attention delivers a decisive benefit for the initial lateral-velocity estimate while keeping yaw-rate accuracy competitive and incurring virtually no parameter overhead. We therefore retain the attention block in the final architecture.

5.5 Real-World Results

Experimental SetUp

All experiments were carried out on two production electric vehicles—**Zeekr vehicle D** (RWD, 2340 kg) and **Zeekr vehicle B** (AWD, 1960 kg) , equipped with identical 19-inch winter tyres.

Vehicle states and driver inputs were sampled from the native CAN and FlexRay buses at 100 Hz using *Vector CANoe 15.0* to log on. An externally mounted **OXTS RT3000** GNSS/INS provided centimetre-level ground-truth motion.

Physical-Parameter Bounds. During every forward pass the guard-layer constrains the vehicle and tyre parameters to the priors in Table 5.12, reproduced below for clarity:

Table 5.12: Prior bounds used for real-world training. All symbols as in Table 5.5. Δm, Δh and Δx_{CoM} encode tolerated deviations from nominal mass, CG height and CG longitudinal position.

Front axle			**Rear axle**		
Coefficient	Min	Max	Coefficient	Min	Max
a_0	1.0	3.0	a_0	1.0	3.0
a_1	−40	40	a_1	−40	40
a_2	0	3000	a_2	0	3000
a_3	0	3000	a_3	0	3000
a_4	0	5	a_4	0	5
a_5	0	2	a_5	0	2
a_6	−1	1	a_6	−1	1
a_7	−1	1	a_7	−1	1
a_8	0	2	a_8	0	2
I_z: $2{-}5 \times 10^3$ kg m^2; Δm: [−300, 500] kg; Δh: [−0.15, 0.25] m; Δx_{CoM}: [−0.20, 0.20] m					

These tyre coefficient intervals are identical to the simulation study except for a wider yaw-inertia span to accommodate the different vehicle and deviation for vehicle properties.

Model and Training Configuration

Architecture. We deploy the **H50_HD32_LST20_RDS** instance of *PhyAtteN*:

- GRU encoder with hidden-state width $d = 32$ and horizon $H = 50$ timestamps (0.5 s);
- Additive attention over the first 30 encoder outputs;
- `RDS` "last-step pooling" to a 32-dimensional context vector;
- Physics-guard head that map parameters to the bounds in Table 5.12.

Total parameter count: ≈ 0.12 M.

The model use Adam(lr $= 10^{-4}$) and batch size of **2048** for 500 epochs. All results reported below use this single hyper-parameter setting; no post-hoc tuning was performed on the evaluation set.

We will discuss the reason for using such a huge and abnormal batch size in section 5.5.3

5.5.1 Aggregate Performance

For confidentiality reasons, all real-world result plots omit the y-axis labels.

As illustrated in Figure 5.9, the ground-truth trajectories (solid lines) and the corresponding PhyAtteN predictions (dashed) are almost indistinguishable, underscoring the network's robustness in both controlled vehicle-dynamics tests and representative public-road driving.

Moreover, PhyAtteN acts as an implicit low-pass filter: its outputs are appreciably smoother than the raw IMU channels, indicating effective suppression of high-frequency measurement noise without sacrificing temporal acuity.

Table 5.13 quantifies this impression. For the Zeekr vehicle D, the model attains good performance in lateral velocity (0.080), around 0.3 in lateral acceleration, and 2.45×10^{-3} in yaw-rate—values that lie below the noise floor of a typical automotive-grade IMU, whose yaw-rate and acceleration noise densities typically exceed 0.004 rad/s and 0.4m/s^2 [36, 37]. Performance on the AWD platform Zeekr vehicle B deteriorates marginally in v_y (0.108) but improves in yaw-rate (0.00147), indicating that the model generalises across platforms, ranging from rear-wheel-drive (RWD) to all-wheel-drive (AWD) configurations—without sacrificing platform-specific fidelity.

5.5.2 Scenario-wise Breakdown

Table 5.14 decomposes the aggregate metrics into three representative manoeuvres—an aggressive double-lane change, a steady-state corner, and mixed public-road

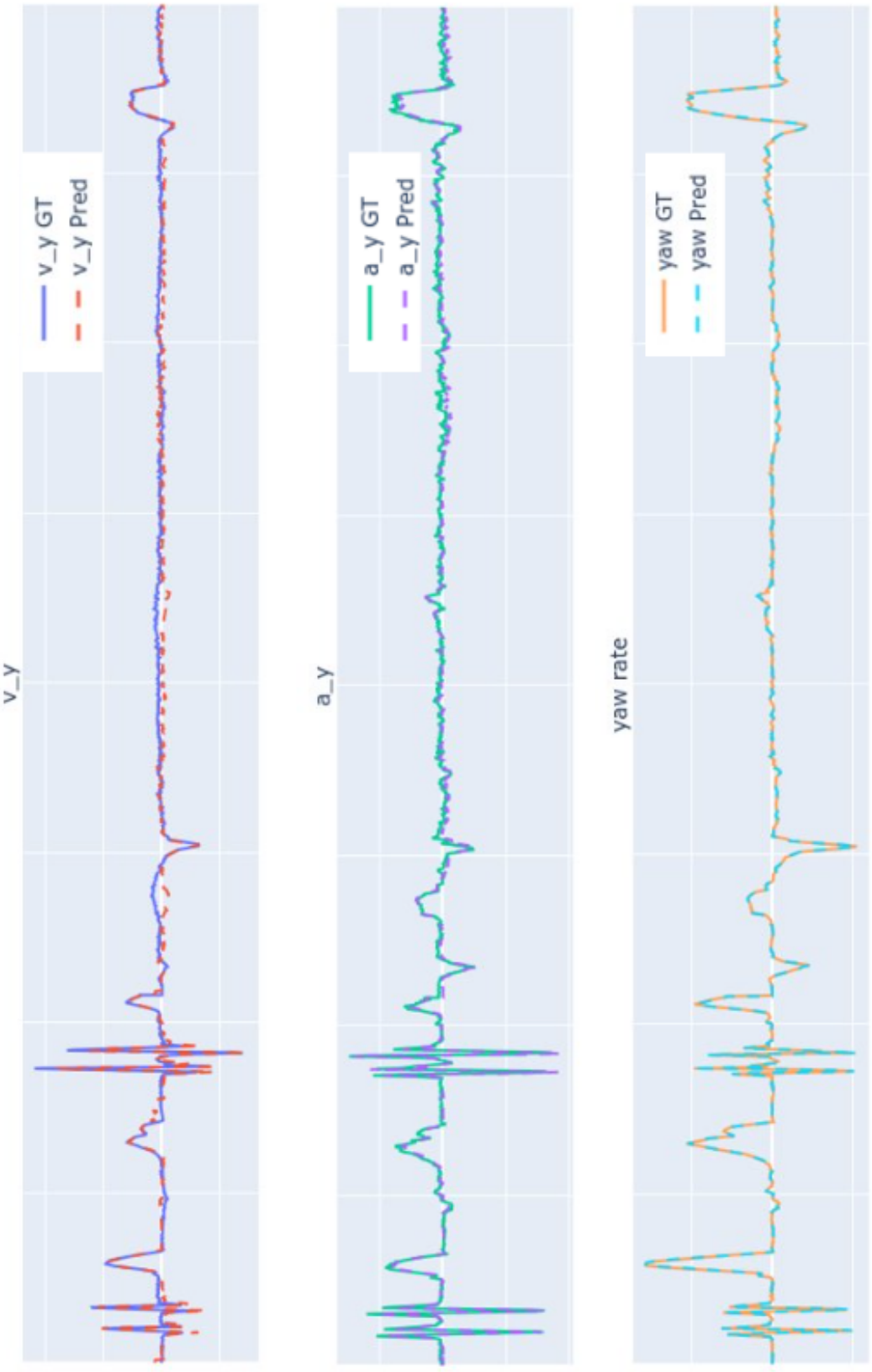

Figure 5.9: Real-world test run on Zeekr Vehicle D: comparison of PhyAtteN predictions (dashed) with ground truth (solid) for lateral velocity (v_y), lateral acceleration (a_y), and yaw-rate (r). The y-axis values have been hidden due to confidentiality requirements. The x-axis is preserved for time-based comparison.

Table 5.13: Root-mean-square error (RMSE) on the held-out *real-world* corpus.

Platform	RMSE_{v_y} [m/s]	RMSE_{a_y} [m/s^2]	RMSE_r [rad/s]
Zeekr vehicle D	0.080	0.297	0.00245
Zeekr vehicle B	0.108	0.284	0.00147

driving—on both test platforms.

Table 5.14: Scenario-wise root-mean-square error (RMSE) for each platform.

Platform	Scenario	RMSEv_y [m/s]	RMSEa_y [m/s^2]	RMSE_r [rad/s]
Zeerk vehicle D	Double lane change	0.0973	0.414	0.00452
	Steady cornering	0.0586	0.221	0.00410
	Public road	0.0725	0.239	0.000806
Zeerk vehicle B	Double lane change	0.185	0.497	0.00277
	Steady cornering	0.124	0.550	0.00139
	Public road	0.0967	0.203	0.00133

Figure 5.10and Figure 5.11 illustrates PhyAtten's performance on Zeekr vehicle D and Zeekr vehicle B under the different tested scenarios, focusing primarily on trends and transient behaviors. The absence of y-axis labels due to data confidentiality, and thus, the visual mismatch seen in steady cornering scenario is primarily due to the zoom-in level chosen to highlight subtle performance differences rather than actual inaccuracies.

From Table 5.14, the aggressive double-lane change scenario introduces the highest dynamic load, characterized by rapid steering inputs and combined-slip events. Despite these demanding conditions, the estimator maintains lateral velocity errors under 0.10 m/s and yaw-rate errors under 5 mrad/s for Zeekr vehicle D, and slightly higher values for Zeekr vehicle B.

During steady-state cornering, where vehicle dynamics are primarily steady and predictable, the estimator leverages its embedded physical priors effectively, reducing RMSE to lower levels—approximately 0.059 m/s for Zeekr vehicle D and 0.124 m/s for Zeekr vehicle B.

In mixed public-road driving scenarios, the performance metrics reflect robust real-world generalization, approaching the sensor's quantisation limits (around 7 cm/s for v_y and 1 mrad/s for r). This indicates PhyAtteN's strong ability to generalize and handle real-world variations such as moderate lateral maneuvers, braking events, and common road irregularities.

Impact of Simplified Model

Across all scenarios, the slightly higher RMSE on the Zeekr vehicle B appears structural rather than data-limited: In our simplified single-track model assumes symmetric axle force distribution and ignores the front torque-vectoring present on the

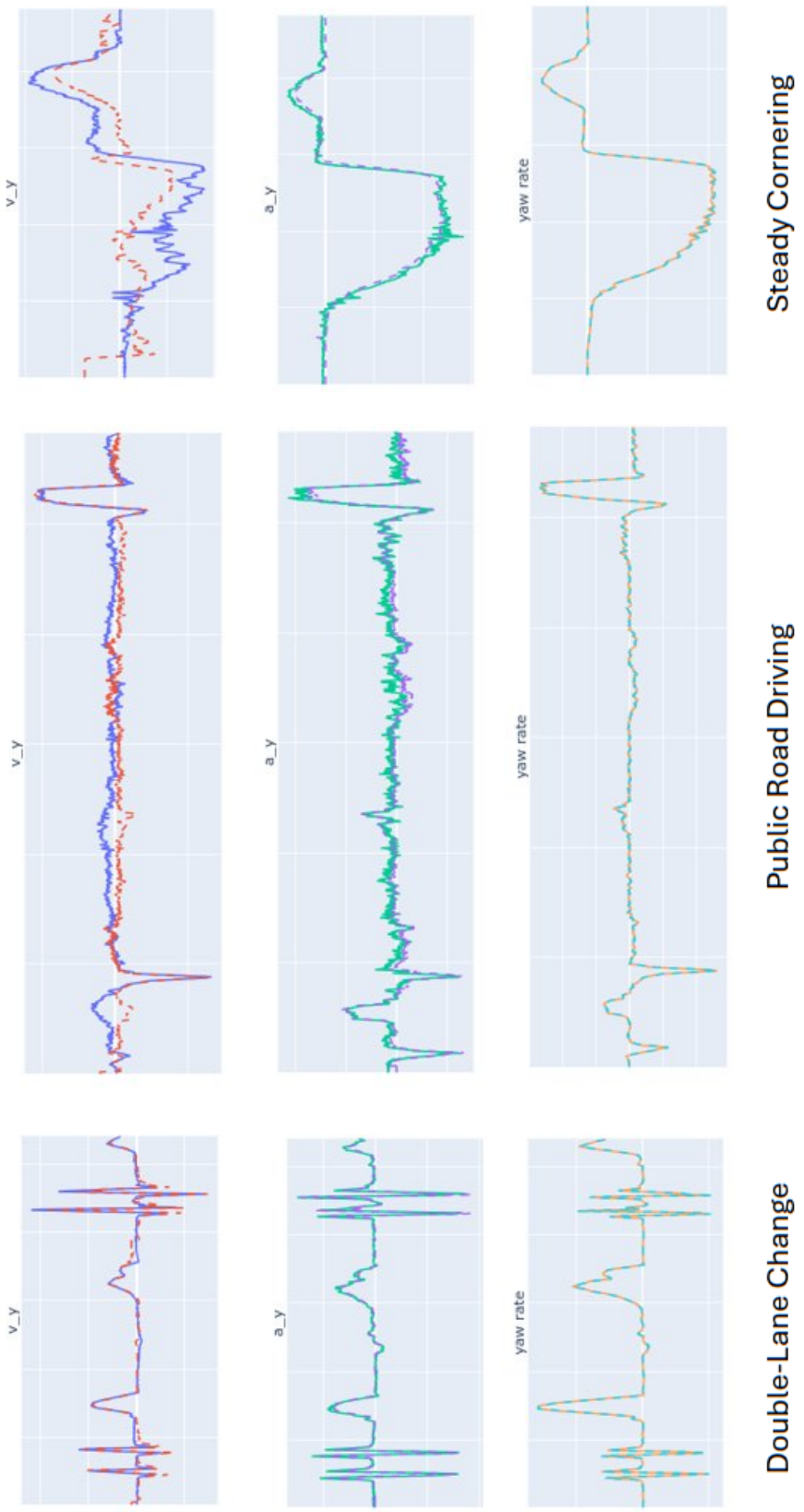

Figure 5.10: Real-world evaluations on Zeekr Vehicle D across three scenarios. Y-axis values are hidden due to confidentiality requirements.

AWD platform. Extending the physics backbone to a two-track (four-wheel) representation or appending a data-driven corrective term for wheel-torque asymmetry is therefore expected to close the remaining gap while maintain the strong noise filter properties of the estimator.

5.5.3 Discussion

Why Large Mini-Batches During Training

Contrary to the conventional wisdom that favours 32–256 sample mini-batches, we found that a batch size of 2048 was necessary when training on the real-world corpus. Small batches failed to learn the weak but critical interaction with the target component v_y, whereas large batches provided sufficiently low-variance gradient estimates to expose the underlying signal.

A key factor is that v_y is a latent variable and is never directly observed. The model must therefore infer it from weak, indirect cues whose gradient contributions are easily swamped by sampling noise when the mini-batch is small[30]. Increasing B averages out that noise and enhances the signal-to-noise ratio of the cross-covariance terms through which the latent component is learned.

The second factor is methodological. Our objective includes a regularisation term that penalises the within-batch variance of the coefficient vector. Estimating this variance from only 64 examples yields a high-variance statistic that can overwhelm the primary loss and destabilize optimization. Similar variance-based regularisers—e.g. Variance-Constancy Loss [31] and VCReg [32]—are likewise reported to require batches of large than 1024.

In the simulation study the data are strictly obey the underlying physical model, and are noise-free. Every 64-sample mini-batch is therefore an unbiased estimator of both the gradient and the coefficient-variance penalty, so optimisation converges even with B=64. The mismatch between simulation and practice arises because (i) the physical model is only an approximation of the real-world process, introducing systematic model-data mismatch, and (ii) the variance-based regulariser becomes unstable when estimated from small, noisy batches in the real corpus.

These observations align with the broader literature on large-batch training. Goyal et.al. (2017) [33]showed that ImageNet can be trained with B=8192 if the learning rate is scaled linearly and prefaced with a short warm-up. Hoffer et.al. (2017)[34] further demonstrated that the apparent generalisation gap associated with large batches disappears when training is measured in optimiser steps rather than epochs, and proposed Ghost Batch Normalisation to retain stable batch statistics. Our findings therefore support the view that mini-batch size must be co-tuned with any batch-wise statistic or regulariser, and that "large" is sometimes a prerequisite rather than a luxury.

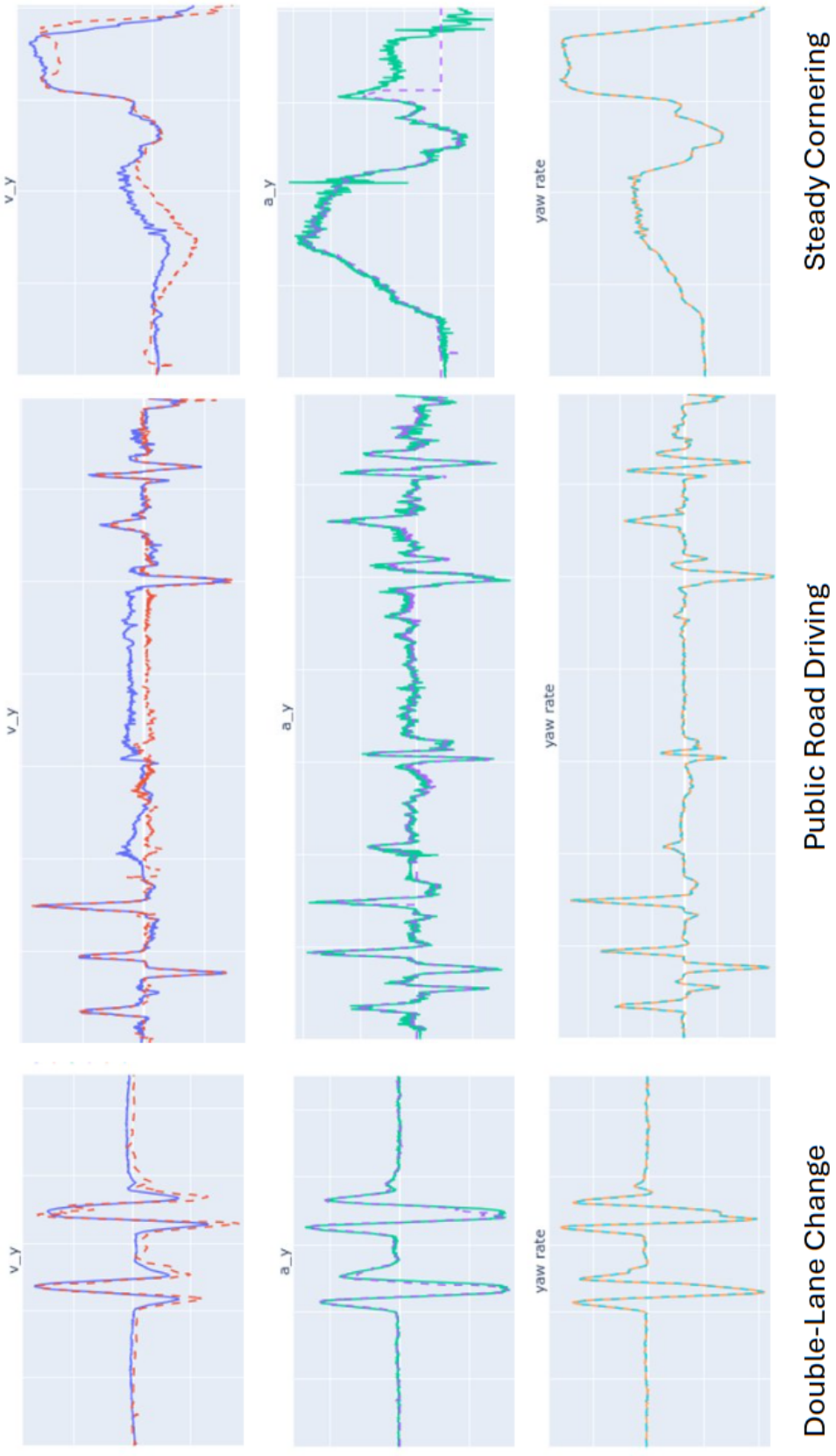

Figure 5.11: Real-world evaluation on Zeekr Vehicle B across representative driving scenarios in Table 5.4.

6

Research Question

6.1 Research Question

"All models are wrong, but some are useful." — George Box

6.1.1 Research Question 1

All models are wrong.

Motivation. The bicycle (single-track) model is a convenient first-order abstraction for lateral vehicle dynamics because it collapses the full 6 degree-of-freedom system to two planar states. In a body-fixed coordinate frame, the lateral–velocity derivative obeys the purely kinematic relation.

$$\dot{v}_y = a_y - r v_x, \tag{1}$$

where a_y is the lateral acceleration, r the yaw rate, and v_x the longitudinal velocity. Equation (1) follows directly from Newton's laws in a rotating frame and therefore holds on any locally flat surface, independent of tyre stiffness, suspension compliance, or load transfer effects.

In practice, however, the quantities in (1) are obtained from onboard sensors that are neither ideal nor perfectly aligned. Vehicle roll couples gravity into the lateral accelerometer, wheel speed sensors bias v_x under slip, and gyro drift contaminates r. Moreover, once the single-track template is promoted to a dynamic model, further idealisations—zero track width, symmetric tyres, negligible chassis compliance—introduce additional error.

Empirical evidence of failure. For every time-step we compute two independent estimates of lateral acceleration:

$$\underbrace{\frac{v_{y,t+1} - v_{y,t}}{\Delta t}}_{\textbf{(A) finite–difference } \dot{v}_y} \stackrel{?}{=} \underbrace{a_{y,t} - r_t\, v_{x,t}}_{\textbf{(B) single–track model}} \tag{2}$$

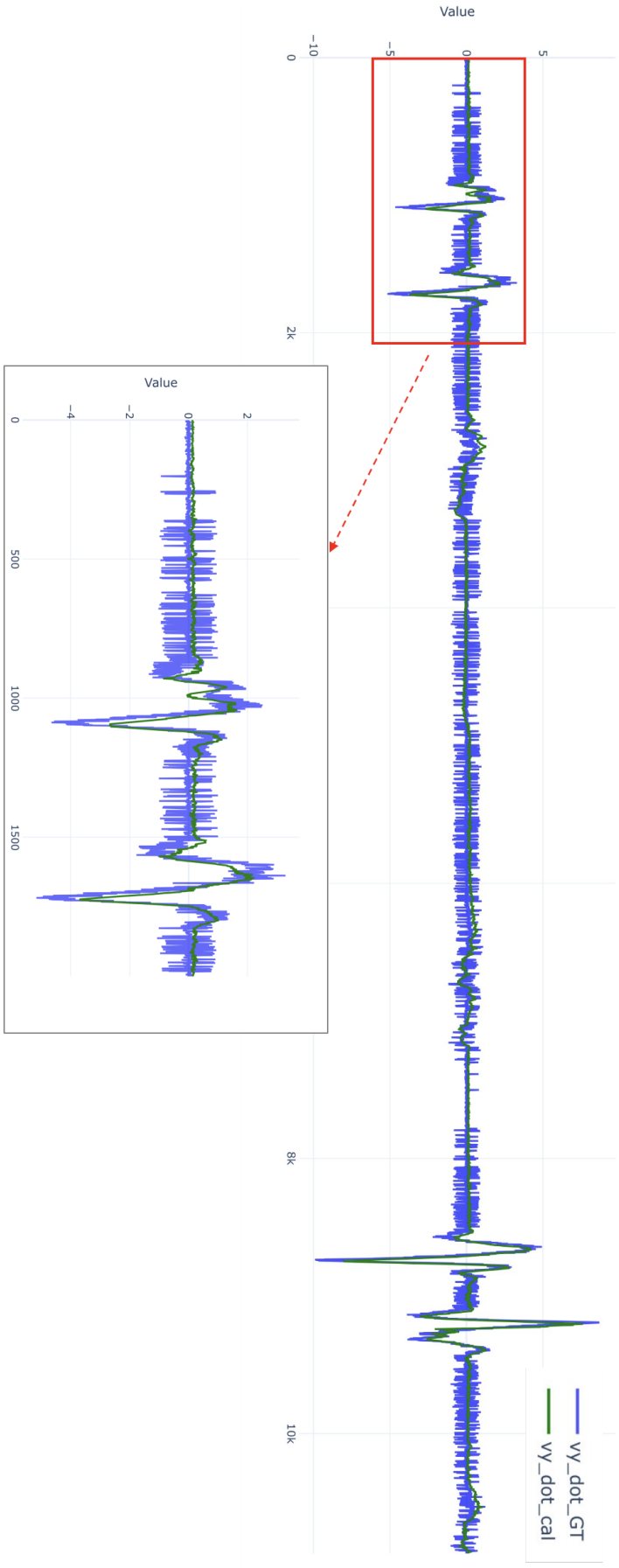

Figure 6.1: Ground-truth $\dot{v}_y$ (blue) versus single-track model (green).

where all signals on the right-hand side come from the OXTS ground-truth channel. Figure 6.1 overlays the raw ground-truth derivative $\dot{v}_y$ (blue) with the single-track prediction (green);

Fig. 6.2 condenses the point-wise error $e_t = \dot{v}y, t^{\mathrm{GT}} - \dot{v}_{y,t}^{\mathrm{cal}}$ over the entire test set into a 2-D histogram.

The overlay in Fig.6.1 together with the error heat-map in Fig.6.2 reveal systematic, structured residuals. These patterns indicate that the idealised assumptions—rigid tyres, flat un-banked road, perfect sensor alignment and negligible chassis compliance—are routinely violated in practice.

Consequently, the baseline single-track model cannot reproduce real-world lateral dynamics with adequate accuracy.

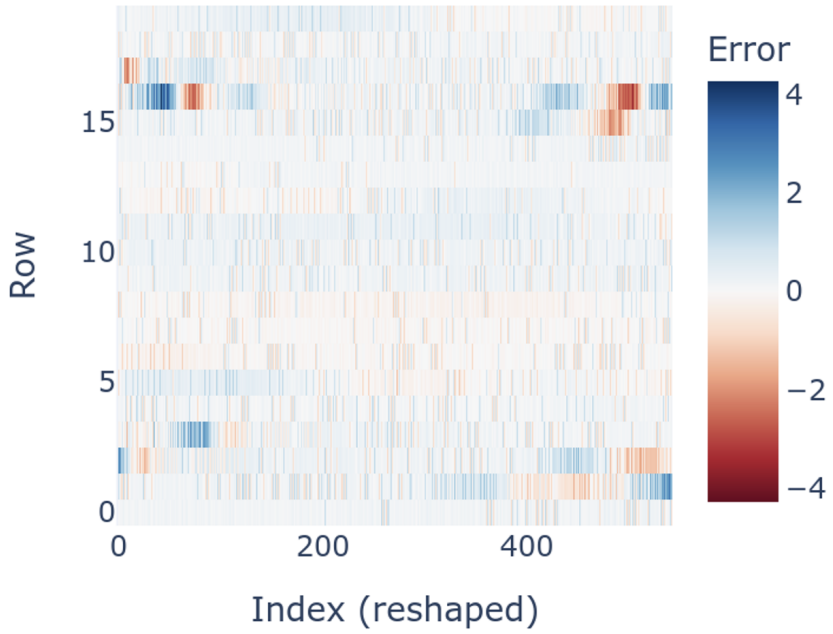

Figure 6.2: Heat-map of signed prediction error e_t for the real-world manoeuvre corpus (blue=over-prediction, red=under-prediction).

In short, the single-track model is "wrong", but the coherent structure of its error implies it may still be useful. *The natural next step is to endow the simplistic template with a small number of learnable parameters that can absorb the missing physics.*

6.1.2 Research Question 2

Some are useful: Simplified Model Forces Neural Parameters to Tell the "Truth"

Motivation. A grey-box approach promises the best of two worlds: it preserves the interpretability and real-time guarantees of the single-track model, while allow-

ing data-driven layers to repair its missing physics.

Instead of discarding the model, we keep its "Wrong" *form but allow a neural network to tuning* dummy coefficient *that bend it back toward the "Truth".*

This operationalises the second half of Box's aphorism—*"some models are useful"*—by testing whether a nominally wrong yet interpretable model can still deliver reliable predictions once its parameters are allowed to flex.

Link to classical observers. Reconciling model predictions with real-world measurements is a central theme in vehicle dynamics and control. The idea underpins Luenberger's pole-placement observer as well as the Kalman, extended Kalman (EKF) and unscented Kalman (UKF) filters: a first-principles prior is continuously corrected by a measurement-driven residual. Our neural parameter layer plays an analogous role, but expresses the correction through physically anchored coefficients instead of an additive observer gain.

"Truth" is relative. The gains supplied by the neural parameter layer are *not* expected to recover the *true* physical constants of the car; they exist to counter-balance the structural error of the single-track model. Hence we speak of *dummy-physical* parameters: every coefficient keeps the correct units, sign and fit in reasonable ranges, yet its numerical value should be interpreted as the *amount of model error routed through that channel* rather than a direct measurement of inertia, tyre scales or sensor scale.

Thus the PhyAtteN performs physics-respectful without claiming to recover tyre-rig test parameters or the car's exact properties.

Counter-example experiment. To make this distinction tangible we ran an ablation in which the network was fed an *under-reported* vehicle mass $\hat{m} = m_{\text{true}} - 200\,\text{kg}$. If the layer were seeking physical fidelity one would expect the learned mass offset Δm to increase accordingly.

Instead, the optimiser channelled ~80,% of the correction into the polynomial tyre-stiffness terms $a_0, \ldots, a_8$ while leaving Δm nearly untouched. Gradient inspection shows these terms offer a steeper descent direction for the lateral-acceleration loss than Δm, so the network "spends" its correction budget where it is mathematically most effective—even when that means nudging coefficients only loosely tied to the physical source of error. The episode underscores that the parameters are pragmatic proxies: dummy variables anchored in physics but tuned to mend a model that is knowingly incomplete.

Towards more faithful coefficient. The previous findings underscore that learnable coefficients in a simplified model serve primarily to compensate for structural error, not to recover true physical constants. Nonetheless, the more accurate the underlying physical scaffold, the more meaningful these learned coefficients become. If the backbone physical model were upgraded, for example, by incorporating lateral

load transfer, compliant suspension effects, then the same training process would likely yield parameters that not only improve prediction but also converge toward real-world values.

This motivates a hybrid modelling strategy: start simple, learn where the gaps are, and incrementally assert better physics to steer the learned parameters closer to physical truth.

7

Conclusion

7.1 Conclusion

Throughout this thesis, we developed, validated, and evaluated a physics-informed neural network (PINN) specifically designed to estimate vehicle lateral dynamics. One key contribution is successfully integrating physical vehicle dynamics models with data-driven neural networks. This integration ensures high accuracy, interpretability, and real-time performance. The proposed PINN consistently outperformed purely data-driven approaches. It showed strong robustness in various driving scenarios, including extreme maneuvers.

Another significant outcome of this thesis is demonstrating how simplified physical models can effectively bridge theory and real-world data. Despite limitations inherent to simplified physical models, the PINN framework provided practical and reliable results. Attention mechanisms and guarded parameter layers proved particularly useful. They ensured physically plausible outcomes and facilitated rapid training convergence.

Additionally, the thesis introduces methods for reconstructing and accurately estimating latent vehicle states. This improves observability and interpretability, even without direct measurement. As a result, downstream tasks such as state estimation and vehicle control benefit significantly.

Ultimately, this thesis emphasizes the importance of hybrid modeling in modern automotive control systems. It offers a scalable approach toward intelligent, adaptive, and continually improving frameworks for vehicle dynamics estimation and control.

7.2 Future Work

The current study has established a robust and efficient physics-informed neural network (PINN) framework capable of accurately estimating vehicle lateral dynamics. Building on this foundational research, several promising directions emerge:

- **On-board Deployment and Evaluation:**

  Transitioning the developed model onto vehicle control units and assessing real-time inference capability under production constraints. Specifically, em-

bedding the PINN in embedded hardware (e.g., automotive-grade ECUs) and rigorously evaluating performance metrics such as latency, computational load, and accuracy against conventional estimation methods.

- **Integration with Downstream State Estimation and Control Algorithms:**

  Exploring synergies with Extended Kalman Filters (EKF), Unscented Kalman Filters (UKF), and model predictive controllers (MPC). The PhyAtteN derived parameters and states would serve as accurate priors for these downstream algorithms, significantly enhancing their reliability, robustness, and predictive capabilities.

- **Extension to Enhanced Physical Models:**

  Expanding the backbone physical model of PhyAtteN beyond lateral dynamics by incorporating comprehensive high-fidelity dynamics, including:

  - Longitudinal Dynamics: Integrating driveline and powertrain dynamics to improve overall vehicle control strategies, particularly under acceleration and braking scenarios.
  - Double-Track Model: Extending the current single-track approximation to double-track models, capturing lateral load transfers more accurately and providing better predictions during aggressive maneuvering.
  - Suspension Dynamics: Incorporating vertical dynamics and suspension models to enable detailed road-tire interaction modeling, enhancing comfort, handling, and predictive maintenance capabilities.

Bibliography

[1] R. Rajamani. *Vehicle Dynamics and Control*. Springer, New York, 2006.

[2] E. Bakker, H. B. Pacejka, and L. Lidner. "A New Tire Model with an Application in Vehicle Dynamics Studies." In: *SAE Transactions* **98**.6: Journal of Passenger Cars (1989), pp. 101–113. Stable URL: `https://www.jstor.org/stable/44472260`. :contentReference[oaicite:0]index=0

[3] OptimumG. *Asymmetric Tire Data and OptimumTire*. 2012. Available at: `https://optimumg.com/asymmetric-tire-data-and-optimumtire/`.

[4] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors." In: *Nature* **323**.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[7] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators." In: *Neural Networks* **2**.5 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

[8] J. L. Elman. "Finding Structure in Time." In: *Cognitive Science* **14**.2 (1990), pp. 179–211. DOI: 10.1207/s15516709cog14021.

[9] P. J. Werbos. "Backpropagation Through Time: What It Does and How to Do It." In: *Proceedings of the IEEE* **78**.10 (1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[10] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult." In: *IEEE Transactions on Neural Networks* **5**.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[11] S. Hochreiter and J. Schmidhuber. "Long short-term memory." In: *Neural Computation* **9**.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. :contentReference[oaicite:0]index=0

[12] R. Pascanu, T. Mikolov, and Y. Bengio. "On the difficulty of training recurrent neural networks." In: *Proc. 30th Int. Conf. on Machine Learning (ICML)*, 2013, pp. 1310–1318.

[13] K. Cho, B. Van Merrienboer, D. Bahdanau, and Y. Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." In: *Proc. Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[14] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." In: *Proc. 3rd International Conference on Learning Representations (ICLR)*, 2015. arXiv:1409.0473 [cs.CL].

[15] M.-T. Luong, H. Pham, and C. D. Manning. "Effective Approaches to Attention-based Neural Machine Translation." In: *Proc. 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166.

[16] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations." *arXiv preprint arXiv:1711.10561*, 2017.

[17] B. T. Polyak. Some Methods of Speeding up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[18] Tijmen Tieleman and Geoffrey Hinton. RMSProp: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*, 2012.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[20] C. Lundquist and T. B. Schön. Recursive Identification of Cornering Stiffness Parameters for an Enhanced Single Track Model. In *Proceedings of the 15th IFAC Symposium on System Identification*, pages 1726–1731, Saint-Malo, France, July 6–8 2009. DOI: 10.3182/20090706-3-FR-2004.00287.

[21] T. A. Wenzel *et al.* "Dual extended Kalman filter for vehicle state and parameter estimation." In: *Vehicle System Dynamics* **44**.2 (2006), pp. 153–171. DOI: 10.1080/00423110500385949.

[22] Aymen Alshawi *et al.* "An Adaptive Unscented Kalman Filter for the Estimation of the Vehicle Velocity Components, Slip Angles, and Slip Ratios in Extreme Driving Manoeuvres." In: *Sensors* **24**.2 (2024), 436. DOI: 10.3390/s24020436.

[23] L. Hermansdorfer *et al.* "End-to-End Neural Network for Vehicle Dynamics Modeling." In: *Proc. 6th IEEE Congress on Information Science and Technology (CISST)*, 2020, pp. 407–411. DOI: 10.1109/CIST49399.2021.9357196. :contentReference[oaicite:2]index=2:contentReference[oaicite:3]index=3

[24] G. Napolitano Dell'Annunziata *et al.* "Four-Wheeled Vehicle Sideslip Angle Estimation: A Machine Learning-Based Technique for Real-Time Virtual Sensor Development." In: *Applied Sciences* **14**.3 (2024), 1036. DOI: 10.3390/app14031036. :contentReference[oaicite:0]index=0:contentReference[oaicite:1]index=1

[25] T. Kim; Lee, H.; Lee, W. *et al.* "Physics Embedded Neural Network Vehicle Model and Applications in Risk-Aware Autonomous Driving Using Latent Features." arXiv:2207.07920v1 [cs.RO], 16 Jul. 2022. :contentReference[oaicite:6]index=6:contentReference[oaicite:7]index=7

[26] J. Chrosniak; Ning, J.; Behl, M. *et al.* "Deep Dynamics: Vehicle Dynamics Modeling with a Physics-Constrained Neural Network for Autonomous Racing." arXiv:2312.04374v2 [cs.RO], 2 Dec. 2024. :contentReference[oaicite:4]index=4:contentReference[oaicite:5]index=5

[27] C. Tan; Cai, Y.; Wang, H.; Sun, X.; Chen, L. "Vehicle State Estimation Combining Physics-Informed Neural Network and Unscented Kalman Filtering on Manifolds." In: *Sensors* **23** (2023), 6665. DOI: 10.3390/s23156665. :contentReference[oaicite:8]index=8:contentReference[oaicite:9]index=9

[28] H. B. Pacejka. *Tire and Vehicle Dynamics*. 2nd ed., Butterworth-Heinemann, Oxford, UK, 2006. ISBN: 9780750669184.

[29] W. F. Milliken and D. L. Milliken. *Race Car Vehicle Dynamics*. 15th printing, SAE International, Warrendale, PA, 2019. Originally published 1995. ISBN: 9781560915263.

[30] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. arXiv:1609.04836.

[31] Littwin, E., & Wolf, L. (2018). *Regularizing by the Variance of the Activations' Sample-Variances*. arXiv:1811.08764.

[32] Mialon, G., Balestriero, R., & LeCun, Y. (2024). *Variance Covariance Regularization Enforces Pairwise Independence in Self-Supervised Representations*. arXiv:2209.14905.

[33] Goyal, P., Doll'ar, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., & He, K. (2017). *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*. arXiv:1706.02677.

[34] Hoffer, E., Hubara, I., & Soudry, D. (2017). *Train Longer, Generalize Better: Closing the Generalization Gap in Large Batch Training of Neural Networks*. arXiv:1705.08741.

[35] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[36] STMicroelectronics, *ASM330LHH: Automotive grade 6-axis IMU datasheet*, 2020. Available: `https://www.st.com/resource/en/datasheet/asm330lhh.pdf`

[37] Bosch Sensortec, *BMI160: Low power inertial measurement unit datasheet*, 2019. Available: `https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi160/`

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY