

Article

A Hybrid Physics-Augmented Neural Network for Dynamic System Modeling with Partially Known Dynamics

Laurin Ludmann ^{1,*}, Jaeyoun Choi ², Jens Neubeck ³, Andreas Wagner ^{1,3} and Chuchu Fan ²

¹ Institute of Automotive Engineering, University of Stuttgart, Pfaffenwaldring 12, 70569 Stuttgart, Germany; andreas.wagner@ifs.uni-stuttgart.de

² Reliable Autonomous Systems Lab, Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA; cjy0051@mit.edu (J.C.); chuchu@mit.edu (C.F.)

³ Forschungsinstitut für Kraftfahrwesen und Fahrzeugmotoren Stuttgart (FKFS), Pfaffenwaldring 12, 70569 Stuttgart, Germany; jens.neubeck@fkfs.de

* Correspondence: laurin.ludmann@ifs.uni-stuttgart.de

Abstract

This paper introduces Hybrid Physics-Augmented Neural Network (HyPA-Net), a hybrid modeling framework that integrates physics-based linear time-invariant models with artificial neural networks (ANNs) to address dynamic system modeling when only partial physical knowledge is available. The approach leverages the interpretability and robustness of established physical models while using ANNs—such as long short-term memory architectures—to capture unknown or nonlinear system behaviors. The methodology normalizes state and input variables for compatibility with ANN training and expands traditional recursive state-space equations for efficient backpropagation over sequences. Vehicle dynamics, specifically using a rear-wheel steering test case, validate the proposed framework. Various HyPA-Net configurations are benchmarked against pure physics-based and pure data-driven models, demonstrating improved prediction accuracy and model flexibility. The experimental results in this application confirm that hybrid models yield superior performance over strict physical approaches and can implicitly approximate submodel dynamics within a unified, yet modular, architecture, opening avenues for applications in domains where partial physics-based knowledge is available but insufficient on its own.

Keywords: dynamic system modeling; hybrid modeling; physics-augmented neural networks; vehicle dynamics; state space models

1. Introduction

The field of dynamic system modeling has seen a significant shift towards data-driven approaches in recent years. This is mainly due to the increasing availability of computing power and the success of machine learning techniques in various application domains. In this context, artificial neural networks (ANNs) have emerged as a powerful tool for modeling complex systems [1–7]. ANNs are capable of learning complex patterns from data and can be used to model systems with unknown dynamics.

Traditional ANNs process inputs through interconnected layers of neurons with non-linear activation functions, excelling at modeling nonlinear relationships through weighted connections and backpropagation [8,9]. For sequential data, recurrent neural networks (RNNs) introduced memory via hidden states that propagate temporal information, enabling applications in speech recognition and time series forecasting [10,11]. Long short-



Received: 2 June 2026

Revised: 20 July 2026

Accepted: 23 July 2026

Published: 27 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

term memory (LSTM) networks, introduced by [12], addressed the vanishing gradient limitations of RNNs through gated mechanisms, achieving breakthroughs in machine translation and long-context tasks by selectively retaining information across long sequences. LSTM networks are well-suited for time series data and have proven to be able to capture long-term dependencies in the input sequence [13–16]. Modern architectures, on the other hand, like Transformers, fundamentally changed sequence modeling through self-attention mechanisms that capture global dependencies without recurrence, though their quadratic complexity prompts adaptations for time series applications [17–19].

Although primary and popular fields of research are natural language processing with large language models (LLMs), image processing, and general agents, ANNs have also been successfully applied in the field of dynamic system modeling [6,20]. However, since ANNs generally only model the input-output relationship without providing insights into the underlying physics or internal dynamics of the system, they are considered black-box models, which limits their applicability in safety-critical systems, where understanding the underlying dynamics is crucial.

In this paper, we propose a novel approach to dynamic system modeling that combines the strengths of physics-based models and artificial neural networks. The approach, called Hybrid Physics-Augmented Network (HyPA-Net), uses a physical model as a foundation and augments it with an artificial neural network to model unknown parts or nonlinearities of the system. The novelty of the proposed HyPA-Net lies in four aspects. First, a hybrid modeling framework for dynamic systems with partially known dynamics is introduced, in which an explicit LTI state-space model of the known physics is embedded directly into a neural sequence architecture rather than being imposed only through the loss or replaced by a fully black-box model. Second, a state-of-the-art normalization-consistent formulation of the state-space equations together with a fixed-horizon expansion is derived, that enables efficient gradient-based training while preserving the physical meaning of states and inputs. Third, the framework allows unknown subsystem dynamics to be approximated implicitly from overall system-level input-output data, without requiring direct supervision of the subsystem itself. Fourth, this concept is demonstrated on a vehicle-dynamics case with rear-wheel steering and shows that the hybrid model improves substantially over the pure first-principles model while remaining competitive with end-to-end neural baselines.

2. Related Work

In this section, related work in the field of dynamic system modeling is reviewed with a focus on approaches that combine physical modeling with artificial neural networks.

2.1. State Space Models with Neural Networks

The class of structured state space sequence models (S4) recently emerged as a promising approach to sequence modeling inspired by the mathematical foundations of LTI systems and related to RNNs. In the seminal High-Order Polynomial Projection Operators (HiPPO) framework, a principled approach to online signal compression through dynamic memory mechanisms was introduced by [21]. The subsequent Mamba architecture demonstrates that state space models can outperform attention-based transformers in both accuracy and computational efficiency [22]. It has further been shown by [23] that S4 models in general are a variation in the transformer architecture, which is more efficient and effective for long sequences, leveraging the implicit knowledge of the system dynamics. This line of work has been extended by many others, leading to different variations in the S4 architecture, such as H3 [24], Hyena [25], RetNet [26] and others. However, these models all learn the dynamics of the system from data, without explicit incorporation of known physical laws or constraints. In contrast, HyPA-Net embeds an explicit, physically

parameterized LTI state-space model as a structural backbone rather than learning the full state transition from data, which constrains the known-physics component to remain interpretable and reduces the data burden on the ANN to only the residual dynamics.

2.2. Physics-Informed Neural Network Approaches

Physics-Informed Neural Networks represent a class of neural networks that incorporate physical laws, such as partial differential equations, directly into their training processes. They can handle a variety of complex equations by minimizing a loss function that combines data fitting and physics residuals [27,28]. These approaches have been successfully applied to a wide range of problems, including fluid dynamics [29], heat transfer [5], and power systems [30,31]. To some extent, they have demonstrated their ability to be more feasible than classical numerical methods. Training such hybrid models presents unique challenges, particularly in balancing the contribution of physical constraints with data-driven learning objectives. While the physics is integrated, e.g., in the loss function, the ANN itself still remains a black box. Unlike PINNs, where physical laws enter only as a soft penalty in the loss function while the network architecture itself is unconstrained, HyPA-Net hard-wires the known dynamics as an explicit computational sub-block in the forward pass, so the physical structure constrains the model's computation directly rather than only its training objective.

2.3. Neural Ordinary Differential Equations

Neural Ordinary Differential Equations (Neural ODEs) have introduced a novel approach to modeling dynamic systems by approximating continuous-time dynamics using neural networks. Instead of specifying a discrete sequence of hidden layers, the derivative of the hidden state is parameterized using a neural network [32]. The advantage of Neural ODE approaches lies in their ability to learn complex temporal dependencies while maintaining temporal consistency across different sampling rates. Variations such as Graph Neural ODEs [25], Stiff Neural ODEs [33] and Bayesian Neural ODEs [34] have been successfully proposed to address specific problems. Challenges remain, for example, regarding the numerical integration of ODEs, particularly in terms of stability and convergence [35] and their robustness is still unclear [36]. Furthermore, Neural ODEs remain a black box.

2.4. Koopman Neural Networks

In Koopman theory, nonlinear dynamical systems are represented as linear dynamics on functions of the observable state, and their spectral decomposition characterizes the behavior of the nonlinear system. Finding finite-dimensional representations of the Koopman operator means finding coordinate transformations where the nonlinear dynamics appear linear [37]. Dynamic Mode Decomposition is a numerical method to approximate the spectral objects of the Koopman operator, e.g., eigenvalues and modes, whereas a Koopman neural network learns these representations [38–41]. All methods recast nonlinear dynamics as approximately linear evolution in an enlarged space of observables or latent coordinates, but they typically treat the underlying physics only implicitly, if at all, and operate in abstract feature spaces rather than in the original physical state space. In contrast, our method starts with an explicit LTI state-space model of the known physics and augments only the unknown or nonlinear residual dynamics with a neural network so that the core representation remains firmly anchored in physically meaningful states and inputs. This design yields several important advantages. The physics grounding enforces structural consistency with known laws, the separation between modeled and learned components enhances interpretability of both the global model and embedded submodels, and the focus on learning only residual dynamics can substantially improve data efficiency compared to approaches that must infer the entire dynamical map from data alone.

3. Problem Statement

The aim of this paper is to develop a hybrid modeling framework that integrates partially known physical dynamics with data-driven learning in order to accurately capture the behavior of complex dynamic systems. Specifically considered are systems where the governing equations can be described in a state-space formulation, but where only a subset of the dynamics is analytically available. In many real-world cases, the system dynamics can be decomposed into a known and an unknown residual component. Traditional physics-based models often approximate the real dynamics, which leads to systematic errors, whereas purely data-driven methods attempt to learn the full mapping, often requiring large datasets, suffering from poor generalization outside the training domain, and offering no insights into the internal dynamics of the system if they are not trained directly. The central problem is therefore to jointly exploit partial physical knowledge and data-driven learning to approximate the residual dynamics while retaining interpretability, stability, and data efficiency. In particular, this work aims to design a hybrid model based on a first-principles physical model in combination with a parameterized neural augmentation mechanism that can achieve the following:

- Faithfully reproduce the observed trajectories.
- Generalize to unseen inputs and operating regimes.
- Remain consistent with the known physical structures.
- Implicitly approximate subsystems in dynamic systems without isolating them.

4. Methodology

In this section, we present our approach and the general network architecture of HyPA-Net, consisting of two main components: a physics-based model and an ANN. Those two components can be combined in different ways to strike the desired balance between interpretability and flexibility, always taking into account the specific requirements of the application domain. The physics-based model represents the known system dynamics, while the ANN complements the physics-based model by modeling the unknown or neglected nonlinearities in the system.

An example of a general network architecture is shown in Figure 1. In this case, an arbitrary input u is passed to two different sub-models. The first is known and modeled using a first-principles physical model. However, the second one is either unknown or too complex to model directly. Therefore, an ANN is introduced to capture its dynamics. The respective outputs of both sub-models γ_1 and γ_2 are then passed on to another sub-model, which is again modeled using a first-principles physical model to generate the overall output x . While there are various ways to represent dynamic systems, this work sets a focus on the discrete state-space representation of LTI systems. An overview of the workflow is given in Figure 2.

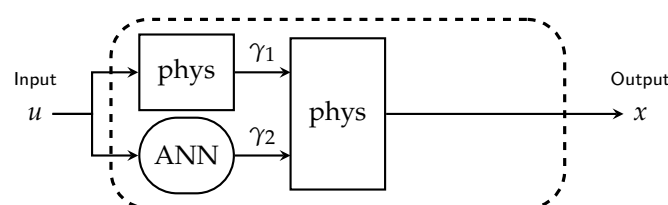


Figure 1. One general example variant of the HyPA-Net architecture. Two known components of the system are modeled using LTI systems (sharp edges with “phys” label), while the unknown part is modeled using an ANN (rounded edges).

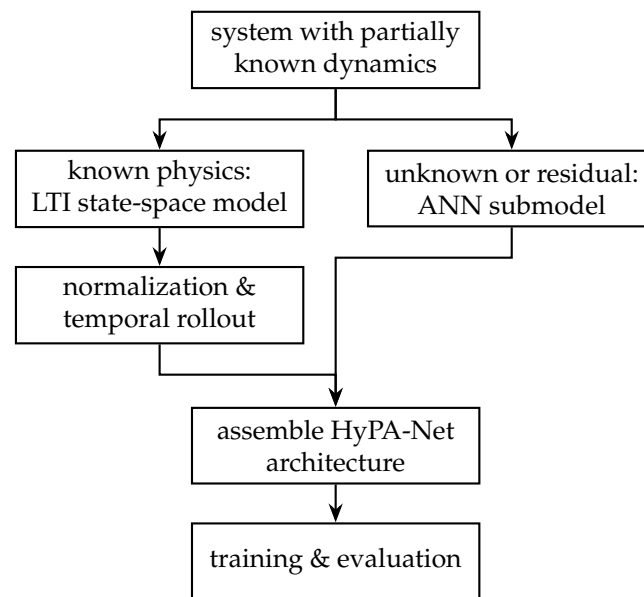


Figure 2. Overview of the HyPA-Net methodology workflow, from the decomposition of physical and residual dynamics through normalization, horizon expansion, hybrid model assembly, training and evaluation.

Equation (1) describes the transition of the system's state x from one time step to the next, indicated by the index k [42]. The state matrix A describes the system's dynamics, while the input matrix B describes how the input u influences the state.

$$x_{k+1} = Ax_k + Bu_k \quad (1)$$

Discrete-time state-space models are found in a variety of literature, e.g., [42–45]. Linear state space models have proven particularly valuable in a wide range of applications from economic forecasting [46,47] to aerospace systems [48,49]. These models excel at representing higher-order linear systems, predicting future system states, and serving as foundational components in more complex estimation frameworks such as Kalman filtering [50]. However, their linear nature inherently limits their ability to capture nonlinear phenomena that are ubiquitous in real-world systems.

With a given initial state x_0 and a sequence of inputs $u_0, u_1, \dots$, the state trajectory $x_1, x_2, \dots$ can be computed iteratively. However, two problems arise when using Equation (1) to compute the state trajectory:

1. The state space equations are based on physical units, i.e., the state and input variables have physical meanings. ANNs operate on normalized data, i.e., the input and output variables are normalized, e.g., to zero mean and unit variance.
2. The state trajectory is computed recursively, i.e., the state at time step $k + 2$ depends on the state at time step $k + 1$. This recursion interferes with the backpropagation algorithm used to train the ANN.

To address these problems, we propose a novel approach that combines the state space representation of LTI systems with ANNs while solving the aforementioned issues. The key idea is to normalize the state and input variables as well as the state space equations before feeding them into the ANN and to compute the state trajectory in parallel using the state space equations. As a first step, the state and input variables are normalized using the mean and standard deviation of the training data

$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \text{ and } \hat{u} = \frac{u - \mu_u}{\sigma_u} \quad (2)$$

with μ_x and σ_x being the mean and standard deviation of the state variables, and μ_u and σ_u respectively. Within this paper, all variables with a hat $\hat{\cdot}$ are normalized variables. Using the backtransformation from Equation (2) and inserting into the state space Equation (1) yields the normalized state space equation

$$x_{k+1} = A(\hat{x}_k \sigma_x + \mu_x) + B(\hat{u}_k \sigma_u + \mu_u) \quad (3)$$

With the help of this equation, the physical states can be calculated based on the normalized states and inputs. To calculate the normalized states, Equation (2) has to be incorporated on the left-hand side of Equation (3) as well as on the right-hand side and rearranged. This yields

$$\hat{x}_{k+1} = \frac{A(\hat{x}_k \sigma_x + \mu_x) + B(\hat{u}_k \sigma_u + \mu_u) - \mu_x}{\sigma_x} \quad (4)$$

Because normalization is an element-wise operation, the associativity and distributivity laws apply. Subsequently converting the state space matrices to the normalized domain as well, Equation (4) can be rewritten as

$$\begin{aligned} \hat{x}_{k+1} &= \frac{A\sigma_x}{\sigma_x} \hat{x}_k + \frac{A\mu_x}{\sigma_x} + \frac{B\sigma_u}{\sigma_x} \hat{u}_k + \frac{B\mu_u}{\sigma_x} - \frac{\mu_x}{\sigma_x} \\ &= \hat{A}\hat{x}_k + \hat{B}\hat{u}_k + \hat{H} \end{aligned} \quad (5)$$

with

$$\hat{A} = \frac{A\sigma_x}{\sigma_x} \quad (6)$$

$$\hat{B} = \frac{B\sigma_u}{\sigma_x} \quad (7)$$

$$\hat{H} = \frac{A\mu_x}{\sigma_x} + \frac{B\mu_u}{\sigma_x} - \frac{\mu_x}{\sigma_x} \quad (8)$$

where H represents a constant normalization offset in the normalized domain. Equation (5) is the final form of the normalized state space equation. It requires normalized states and inputs and calculates the normalized states at the next time step. It is therefore able to compute the state trajectory in the scale of the ANN. This solves the scale problem mentioned above.

For the second problem, the recursive nature of the LTI state space equation, the state space equation is expanded to a fixed number of time steps to precalculate the recursion. Introducing a target sequence length N and expanding state and input to

$$\hat{X} = \begin{bmatrix} \hat{x}_{k+1} \\ \hat{x}_{k+2} \\ \vdots \\ \hat{x}_{k+N} \end{bmatrix} \in \mathbb{R}^{N \times n} \text{ and } \hat{U} = \begin{bmatrix} \hat{u}_k \\ \hat{u}_{k+1} \\ \vdots \\ \hat{u}_{k+N-1} \end{bmatrix} \in \mathbb{R}^{N \times m} \quad (9)$$

requires Equation (5) to be expanded to

$$\begin{aligned}
 \hat{\mathbf{X}} &= \begin{bmatrix} \hat{\mathbf{A}} \\ \hat{\mathbf{A}}^2 \\ \vdots \\ \hat{\mathbf{A}}^N \end{bmatrix} \hat{\mathbf{x}}_k + \begin{bmatrix} \hat{\mathbf{B}} \\ \hat{\mathbf{A}}\hat{\mathbf{B}} \\ \vdots \\ \hat{\mathbf{A}}^{N-1}\hat{\mathbf{B}} \end{bmatrix} \hat{\mathbf{u}}_k + \begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{B}} \\ \vdots \\ \hat{\mathbf{A}}^{N-2}\hat{\mathbf{B}} \end{bmatrix} \hat{\mathbf{u}}_{k+1} + \cdots + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \hat{\mathbf{B}} \end{bmatrix} \hat{\mathbf{u}}_{k+N} \\
 &+ \begin{bmatrix} \hat{\mathbf{H}} \\ \hat{\mathbf{A}}\hat{\mathbf{K}} + \hat{\mathbf{H}} \\ \vdots \\ \hat{\mathbf{A}}^{N-1}\hat{\mathbf{H}} + \hat{\mathbf{A}}^{N-2}\hat{\mathbf{H}} + \cdots + \hat{\mathbf{H}} \end{bmatrix} \\
 &= \begin{bmatrix} \hat{\mathbf{A}} \\ \hat{\mathbf{A}}^2 \\ \vdots \\ \hat{\mathbf{A}}^N \end{bmatrix} \hat{\mathbf{x}}_k + \begin{bmatrix} \hat{\mathbf{B}} & \mathbf{0} & \cdots & \mathbf{0} \\ \hat{\mathbf{A}}\hat{\mathbf{B}} & \hat{\mathbf{B}} & \cdots & \mathbf{0} \\ \hat{\mathbf{A}}^2\hat{\mathbf{B}} & \hat{\mathbf{A}}\hat{\mathbf{B}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{A}}^{N-1}\hat{\mathbf{B}} & \hat{\mathbf{A}}^{N-2}\hat{\mathbf{B}} & \cdots & \hat{\mathbf{B}} \end{bmatrix} \hat{\mathbf{U}} \\
 &+ \begin{bmatrix} \hat{\mathbf{H}} \\ \hat{\mathbf{A}}\hat{\mathbf{K}} + \hat{\mathbf{H}} \\ \vdots \\ \hat{\mathbf{A}}^{N-1}\hat{\mathbf{H}} + \hat{\mathbf{A}}^{N-2}\hat{\mathbf{H}} + \cdots + \hat{\mathbf{H}} \end{bmatrix} \quad (10)
 \end{aligned}$$

or in short

$$\hat{\mathbf{X}} = \hat{\mathbf{A}}\hat{\mathbf{x}}_k + \hat{\mathbf{B}}\hat{\mathbf{U}} + \hat{\mathbf{H}} \quad (11)$$

Equation (11) represents the final form of the normalized state space equation for a fixed number of time steps. This enables the parallel computation of the state trajectory in the scale of the ANN and solves the recursion problem mentioned above.

A concern for safety-critical applications such as vehicle dynamics is the theoretical behavior of the hybrid architecture. While general convergence proofs and tight generalization bounds for recurrent neural network components remain open problems and equally apply to the purely data-driven baselines considered here, several structural properties of HyPA-Net provide meaningful theoretical grounding. First, the stability of the LTI backbone can be verified explicitly: in discrete time, stability requires the eigenvalues of the state matrix $\mathbf{A}$ to lie strictly within the unit circle, which can be checked and should be enforced during model construction independently of the neural network component. Similarly, controllability and observability of the LTI backbone can be verified directly via the classical Kalman rank conditions, since these matrices are fixed and known a priori. Extending these notions rigorously to the full hybrid system, in which the neural network introduces a nonlinear, recurrent coupling, remains an open problem in control theory and is left for future work. Second, since LSTM gating mechanisms employ bounded activation functions, the neural network outputs are bounded by design. Combined with a stable LTI system receiving bounded inputs, this ensures that the coupled hybrid system produces bounded state trajectories for any bounded input sequence. Third, and distinct from purely data-driven approaches, HyPA-Net preserves the physical meaning of all state variables throughout the architecture in the latent space, enabling domain experts to monitor whether the model operates within physically plausible regimes at inference time. This form of interpretable oversight complements formal guarantees rather than replacing them. We acknowledge that deriving tight generalization bounds for the full hybrid system remains an open challenge and an important direction for future work.

5. Experiments

This paper uses vehicle dynamics as a test case to demonstrate the effectiveness of the proposed approach. With modern vehicles featuring an increasing number of active systems that influence their behavior, inferring an exact model becomes increasingly difficult. However, this modeling technique can easily be adapted to accommodate other systems and robots. Since this is only a test case, the formulation of the vehicle dynamics is briefly described in this chapter. The single-track model from [51] of a passenger car is chosen, because it is a well-known and widely used model in the field of vehicle dynamics. It is a simple yet well-established model, validated for small side slip angles or lateral acceleration of up to around $a_y = 4 \text{ m/s}^2$. For further details, the reader is referred to the literature, e.g., to [52–54].

As our test vehicle has rear-wheel steering, the standard bicycle model from [54] is extended to include rear-wheel steering dynamics in case of physical modeling of the rear-wheel steering, assuming a linear relationship between the steering wheel angle and the rear-wheel angle. This bicycle model is a two-degree-of-freedom model that describes the vehicle dynamics in the lateral and yaw directions, having the side slip angle β and the yaw rate $\dot{\psi}$ as states $x = [\dot{\psi}, \beta]^T$. The side slip angle is the angle between the direction of movement of the vehicle and the longitudinal axis of the body frame coordinate system, and the yaw rate is the rate of turning around the vertical axis. The input to the model is the steering angle of both wheels, represented as $u = [\delta_f, \delta_r]^T$ if the vehicle has a front-wheel steering (FWS) and rear-wheel steering (RWS). The state space matrices of the vehicle dynamics for the continuous model is given by

$$A = \begin{bmatrix} -\frac{c_{\alpha 0, f} l_f^2 + c_{\alpha 0, r} l_r^2}{v I_{zz}} & -\frac{c_{\alpha 0, f} l_f - c_{\alpha 0, r} l_r}{v m} \\ -1 - \frac{c_{\alpha 0, f} l_f - c_{\alpha 0, r} l_r}{v m^2} & -\frac{I_{zz}}{v m} \end{bmatrix} \quad (12)$$

$$B = \begin{bmatrix} \frac{c_{\alpha 0, f} l_f}{v m} & -\frac{c_{\alpha 0, r} l_r}{v m} \\ \frac{I_{zz}}{v m} & \frac{I_{zz}}{v m} \end{bmatrix} \quad (13)$$

with $c_{\alpha 0, f}$ and $c_{\alpha 0, r}$ being the cornering stiffness of the front and rear axles, l_f and l_r are the distances from the center of gravity to the front and rear axles, v the vehicle velocity, I_{zz} the yaw moment of inertia, and m the vehicle mass. The resulting state space representation is discretized using a zero-order hold approach for the implementation. For a comparison of different variations in the model, a variant without rear-wheel steering is also used. In this case, the second column of the input matrix B is removed, resulting in a single-input system which only uses the front-wheel angle $u = \delta_f$ as input.

Vehicle dynamics data were captured on the Handling Roadway (HRW) test bench [55,56]. The HRW, depicted in Figure 3, allows for the simultaneous capture of longitudinal, lateral, and vertical dynamics of the vehicle. For this study, the vehicle is set to a constant velocity, while the steering wheel robot excites the lateral degree of freedom. In total, three different dedicated experiments were conducted for training and validation of the network, resulting in three different dedicated datasets. Both training data sets consisted of stochastic excitation with an excitation spectrum typical for a human driver. The maximum steering wheel angle is $\delta_{sw, \max} = 35$, which results in a static lateral acceleration of around $a_y = 7 \text{ m/s}^2$ for the vehicle velocity of $v_x = 33.3 \text{ m/s}$. The power spectral density of the steering wheel excitation realizes $\delta_{sw, \max}$ up to the break frequency of 1 Hz and subsequently exhibits a $1/f^2$ roll-off up to the cut-off frequency of 5 Hz. The validation set also used stochastic excitation with a different random seed and a frequency of 0.8 Hz, but additionally included specific deterministic maneuvers like a lane change.

With a maximum lateral acceleration of around $a_y = 7 \text{ m/s}^2$, the data is out of validity for the single-track model. This is intentional, because the neglected dynamics is learned by the ANN part of the HyPA-Net, demonstrating its capabilities.

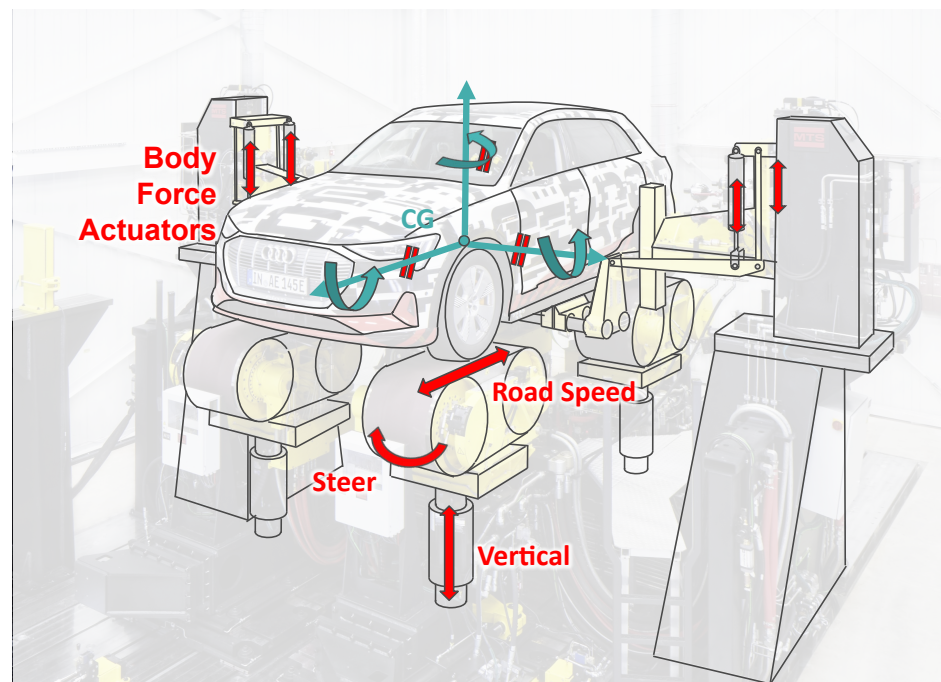


Figure 3. The schematics of the Handling Roadway test bench. The vehicle is fastened in the center of gravity. Roll, pitch, and vertical degrees of freedom are free, while longitudinal, lateral and yaw are fixed and virtually addressed using load cells (green arrows). Each corner module is able to excite vertically, set a steer angle, and control road speed individually (red arrows, lower center). Body force actuators can overlay virtual forces like aerodynamics (red arrows, left and right).

Three purely data-driven approaches were used as baseline benchmarks: an LSTM, a Transformer, and a Mamba, all of which were trained end-to-end. Additionally, the same data were used to parametrize a first-principal physics-based LTI model. The LSTM baseline is a unidirectional many-to-many LSTM with input size 2, hidden size 10, 2 stacked recurrent layers, and a linear output layer mapping the hidden state to 3 output variables. The Transformer baseline is implemented as an encoder-only Transformer with linear input and output projections, fixed sinusoidal positional encoding, model width $d_{\text{model}} = 64$, 4 attention heads, and 2 stacked encoder layers; all unspecified encoder-layer settings follow the PyTorch default configuration. The Mamba baseline is implemented with an input linear layer, two stacked residual Mamba blocks of width $d_{\text{model}} = 64$, and an output linear layer. Each block uses a 2D model input projection with sigmoid gating and GELU state activation, an output projection, and LayerNorm.

Conversely, three different HyPA-Net setups were trained and tested. In all cases, the steering wheel angle δ_{sw} is used as input to the network. For the first variant (V1), the FWS model calculates the corresponding front-wheel steering angle δ_f as a function of δ_{sw} using an LTI system. Since the FWS in the test vehicle is a fixed-gear-ratio mechanical transmission, a linear relationship with a constant steering ratio is a very close approximation and can be experimentally verified. The RWS model, on the other hand, calculates the rear-wheel steering angle δ_r as a function of δ_{sw} using an ANN. The rear-wheel steering in the test vehicle is an active system controlled by an electronic control unit with no mechanical transmission line. It uses mainly the steering wheel angle, but also changes its behavior based on other influences like vehicle velocity or yaw rate. Furthermore, the exact control algorithm is unknown, so a simple linear relationship is therefore not sufficient to

model the rear-wheel steering. Instead, an LSTM, a Transformer, and a Mamba network are used to model the unknown dynamics of the rear-wheel steering, so V1 is tested with three different ANN architectures. However, there is no explicit training of the ANN for the rear-wheel steering. Training is done implicitly by training the whole model. A flow chart of this variant is shown in Figure 4a.

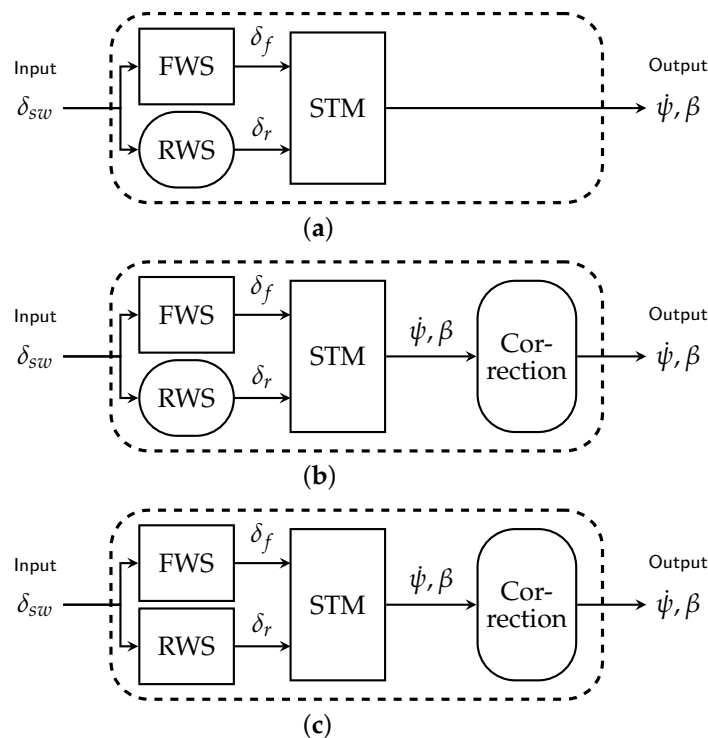


Figure 4. Different variants for the HyPA-Net with LTI systems (sharp edges) and ANNs (rounded edges): (a) V1, FWS and STM as physical models and RWS as ANN; (b) V2, additional correction ANN; (c) V3, FWS, RWS and STM as physical models; only a correction model as ANN.

As the training data contains lateral accelerations of up to $a_y = 7 \text{ m/s}^2$, the small side slip angle assumption is violated, meaning the single-track model is not necessarily valid anymore. Deviations between the ground truth and the prediction are to be expected. It is, furthermore, expected that these deviations will be also addressed by the ANN modeling the rear-wheel steering, incorporating not only the behavior of the true RWS but also the deviations from the STM. To address these deviations, a correction model can be introduced. This correction model is placed after the single-track model to ensure a better fit to the training data. In this case, only LSTM networks are used. A flow chart of this version of the HyPA-Net is shown in Figure 4b. For this variant (V2), the training process is done in two stages: first, training the RWS LSTM network and then training the correction model.

The third variant uses a physical RWS model as a simplification and only corrects for the unknown dynamics (see Figure 4c). Using a fixed velocity for the test setup, a ratio between the steering wheel angle δ_{sw} and the rear-wheel steering angle δ_r can be calculated, which is incorporated in the LTI system. However, it is clear from the data, that there is a neglected temporal relationship between the steering wheel angle and the rear-wheel steering angle. Therefore, an LSTM network is used to correct for unknown parts of the vehicle dynamics after the STM model.

The HyPA-Net training process is the same as that for a standard ANN. It is implemented in PyTorch (v2.6.0) and the training is done on a GeForce RTX 3090 GPU. At a sampling rate of 1024 Hz, the data was down sampled to 32 Hz to ensure larger differences between two time steps. A separate data set was used to determine the following hyper-

parameter. The NAdam optimizer [57] was used with a learning rate of 5×10^{-4} and a batch size of 32. The MSE is used as a loss function. The output of the network is the yaw rate $\dot{\psi}$ and the side slip angle β based on the input steering wheel angle δ_{sw} . The sequence length was set to 100, meaning that the network processes 100 time steps at once. Selecting a proper sequence length is a tradeoff of requirements and should be chosen by application, because smaller sequence lengths generally yield more accurate results while larger ones represent a larger horizon. In all setups, the training was performed for a maximum of 50 epochs with early stopping based on the validation loss. The results presented in the next section are obtained using a separate test set.

6. Results

Different metrics are used to evaluate the performance of the different setups. Table 1 shows the results of the different setups for the validation datasets. The setups are compared using the following metrics: mean squared error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE) and R^2 score. As expected, the physical single-track model shows the worst performance across all metrics because the single-track model is not capable of capturing the full vehicle dynamics. Conversely, the end-to-end LSTM model shows the best performance, while Transformer and Mamba also perform well. The hybrid models (V1, V2, V3) fall in between and enjoy very similar and competitive performance. V1 is the model with the implicit RWS training. The results show that the implicit RWS model is able to capture the dynamics of the rear-wheel steering without explicit training and therefore improve the overall performance. In particular, the combination of V1 with the two S4 models yields a similar performance to the V1 LSTM combination, where MAE and MSE are better for the S4 combinations and MAPE and R^2 are better for the LSTM, although Transformer and Mamba perform worse than the LSTM as end-to-end networks. Due to similar performance and behavior, the results of Transformer and Mamba are not presented as they do not add significant information. All end-to-end models are able to capture the system dynamics while not yielding information at subsystem level. V2 is the model that uses both RWS and the correction model as LSTM networks. This model performs the best among the hybrid models, even being on par with the end-to-end LSTM model for the R^2 score. An ANN as an output of the model improves the performance as expected because it is able to match the training data as close as possible. Similar behavior is observable with other ANNs instead of an LSTM. V3 is the model with the explicit RWS and a correction LSTM model. This variant also shows improved performance in comparison to the first-principle physics model because the LSTM at the end is able to capture all neglected dynamics and also compensate for an ill-defined rear-wheel steering angle. While this variant is certainly better performing, there is no benefit with regard to the characterization of the potential unknown subsystem of the RWS. In general, the results show that the hybrid approach is able to improve the performance of the model compared to the purely physics-based model because of the addition of the residual neural network, while still benefiting from the interpretability of the physical model. Furthermore, the hybrid approach is also competitive with end-to-end ANN with regard to prediction accuracy.

While for the training of the networks, no information about the measured rear-wheel steering angle δ_r is used, it is used as an additional variable for the evaluation of the setups to evaluate the performance of the implicit training of submodels within the HyPA-Net framework. Figure 5 shows the comparison of the ground truth and the different predictions for the rear-wheel steering angle. The plot shows that the implicit RWS model (V1, orange line) is able to predict the rear-wheel steering angle quite well. The same is true for V2 (red line). This confirms that the HyPA-Net framework is able to implicitly approximate the dynamics of a submodel, in this case the rear-wheel steering, without

explicit training of the submodel and within the context of a larger model. A critical limitation of the present study is that the single-track model is used partly outside its nominal validity range. While it is typically valid only for small slip angles or lateral accelerations up to about $a_y = 4 \text{ m/s}^2$, data include maneuvers up to approximately $a_y = 7 \text{ m/s}^2$. Consequently, part of the neural augmentation may compensate not only for unknown rear-wheel-steering dynamics but also for structural mismatch of the underlying bicycle model, including neglected tire and high-slip nonlinearities. It is visible in Figure 5 that both learned RWS models also incorporate dynamic behavior, that is not related to the RWS but still improves the overall performance in the context of the larger model. This is due to the fact that the overall vehicle dynamics are not fully captured by the single-track model because of the neglected tire dynamics and the violation of the small side slip angle assumption. The RWS model is able to capture some of these neglected dynamics, which is why it is able to improve the overall performance of the model. Incorporating a correction model or even a dedicated submodel for the tire partially addresses this issue, but isolating the tire dynamics is difficult as the required measurement of forces and torques at the tire is not available in the current setup and is a future point to improve.

Table 1. Result metrics of the different setups.

Setup	MAE	MSE	MAPE	R^2
physical	2.45×10^{-3}	2.1×10^{-5}	3.44	0.962
LSTM	1.09×10^{-3}	0.3×10^{-5}	2.90	0.993
Transformer	1.43×10^{-3}	0.8×10^{-5}	2.65	0.981
Mamba	1.48×10^{-3}	1.4×10^{-5}	3.03	0.973
V1 LSTM	2.17×10^{-3}	1.7×10^{-5}	2.67	0.984
V1 Transformer	1.95×10^{-3}	1.5×10^{-5}	2.71	0.974
V1 Mamba	1.95×10^{-3}	1.3×10^{-5}	3.35	0.980
V2	1.12×10^{-3}	0.4×10^{-5}	2.56	0.993
V3	1.34×10^{-3}	0.5×10^{-5}	3.07	0.991

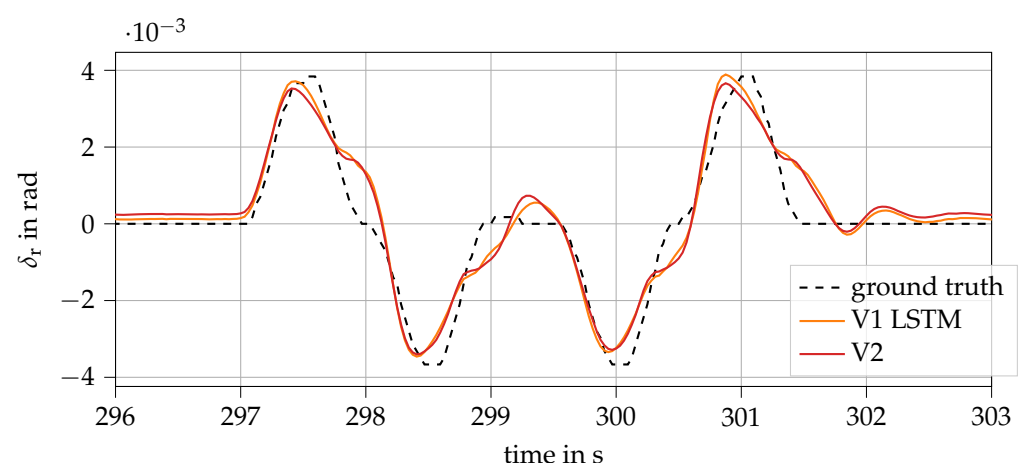


Figure 5. The rear-wheel steering angle over time. The ground truth is shown as a black dashed line, while the different predictions are solid lines.

In the domain of vehicle dynamics, transfer functions are often used as a way of characterizing a vehicle's dynamic behavior. The yaw transfer function describes the relationship between the steering input δ_{sw} and the yaw rate $\dot{\psi}$ of the vehicle. Figure 6 shows the yaw transfer function for the different setups. The single-track model shows in its transfer function a more damped behavior compared to the ground truth. It is not able to capture the full dynamics of the system. The learned RWS (V1 LSTM) improves

on this, resulting in smaller deviations from the ground truth. The other hybrid models with the correction (V2, V3), as well as the baseline ANNs, show a transfer function that is even closer to the ground truth, supporting the above results. This indicates that the correction part at the end is able to adjust the model's predictions to better match the true system dynamics.

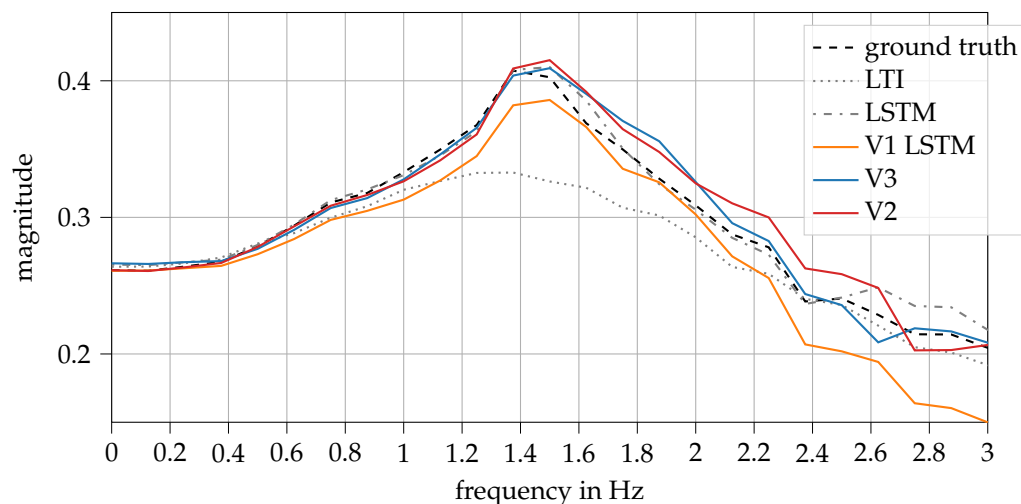


Figure 6. The yaw transfer function for the different setups. The ground truth is shown as a dashed black line, while the different predictions are solid lines.

7. Conclusions

This study presents a novel way to combine first-principles models with ANN in a way that the ANN operates in the normalized data domain, the physical meaning of the values is preserved, and subsystem modeling is possible through implicit training within the ANN. The results show that the hybrid approach is able to improve the performance of the model compared to the purely physics-based model through a data-driven approach for this application, while still benefiting from the interpretability of the physical model. While ANN models need to be retrained if the underlying system dynamics change, traditional methods can often be adapted to new conditions with minimal adjustments. However, those methods are often found to be less accurate than ANNs, especially in complex systems with nonlinear dynamics. The hybrid approach in this paper combines the strengths of both ANNs and classical control methods, allowing for a more flexible and accurate modeling of complex systems. This highlights the potential for hybrid approaches that leverage the strengths of both ANNs and classical methods. Although not empirically tested here, the modular design and vehicle-agnostic formulation of HyPA-Net suggest promising potential for transfer to other vehicle segments and systems. However, the results shown in this study only reflect the capabilities of the current implementation with one specific architecture and one specific dataset. This limits the generalization of the findings, so further research is needed to validate and extend these findings. While the presented variants demonstrate the flexibility of the framework across different levels of physical knowledge availability, a dedicated ablation study isolating individual architectural choices would further strengthen the experimental validation and is identified as a priority for future work. Furthermore, the implicit submodel approximation demonstrated for the rear-wheel steering inherently absorbs residual dynamics beyond the targeted subsystem when multiple unmodeled effects share the same signal pathway. In an ongoing work, we also aim to generalize the HyPA-Net framework to incorporate nonlinear and time-variant physical models to allow for a more flexible and accurate modeling of complex systems with

nonlinear dynamics. Furthermore, we plan to apply the introduced method to different systems to show the generalization capabilities of the HyPA-Net framework.

Author Contributions: Conceptualization, L.L.; methodology, L.L.; software, L.L.; validation, L.L. and J.C.; formal analysis, L.L.; resources, A.W. and C.F.; writing—original draft preparation, L.L.; writing—review and editing, J.C., J.N. and A.W.; visualization, L.L.; supervision, J.N., A.W. and C.F.; project administration, J.N.; funding acquisition, L.L., A.W. and C.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the MIT International Science and Technology Initiatives Seed Fund.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: The datasets presented in this article are not readily available because of a non disclosure agreement on parts of the data. Requests to access the datasets should be directed to corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
FWS	Front-Wheel Steering
HRW	Handling Roadway
HyPA-Net	Hybrid Physics-Augmented Neural Network
LLM	Large Language Model
LSTM	Long Short-Term Memory
LTI	Linear Time Invariant
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
ODE	Ordinary Differential Equation
RNN	Recurrent Neural Networks
RWS	Rear-Wheel Steering
S4	Structured State Space Sequence

References

1. Serrano, G.; Jacinto, M.; Ribeiro-Gomes, J.; Pinto, J.; Guerreiro, B.J.; Bernardino, A.; Cunha, R. Physics-Informed Neural Network for Multirotor Slung Load Systems Modeling. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*; IEEE: New York, NY, USA, 2024; pp. 12592–12598. <https://doi.org/10.1109/ICRA57147.2024.10610582>.
2. Fabiani, F.; Stellato, B.; Masti, D.; Goulart, P.J. A neural network-based approach to hybrid systems identification for control. *arXiv* **2024**, arXiv:2404.01814. <https://doi.org/10.48550/arXiv.2404.01814>.
3. Krätschmer, A.; Lutchen, R.; Reuss, H.C. AI-Based Diagnostic Tool for Offline Evaluation of Measurement Data on Test Benches. In *21. Internationales Stuttgarter Symposium*; Bargende, M., Reuss, H.C., Wagner, A., Eds.; Proceedings; Springer: Wiesbaden, Germany, 2021; pp. 203–214. https://doi.org/10.1007/978-3-658-33521-2_15.
4. Milojevi, S.; Bodza, S.; Cimniak, V.; Angerbauer, M.; Rether, D.; Grill, M.; Bargende, M. *Data-Driven Modeling: An AI Toolchain for the Powertrain Development Process*; SAE Technical Paper Series; SAE International: Warrendale, PA, USA, 2022. <https://doi.org/10.4271/2022-01-0158>.
5. Cai, S.; Wang, Z.; Wang, S.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks for Heat Transfer Problems. *J. Heat Transf.* **2021**, *143*, 060801. <https://doi.org/10.1115/1.4050542>.

6. Da Lio, M.; Bortoluzzi, D.; Rosati Papini, G.P. Modelling longitudinal vehicle dynamics with neural networks. *Veh. Syst. Dyn.* **2020**, *58*, 1675–1693. <https://doi.org/10.1080/00423114.2019.1638947>.
7. Zhou, T.; Gao, S.; Wang, J.; Chu, C.; Todo, Y.; Tang, Z. Financial time series prediction using a dendritic neuron model. *Knowl.-Based Syst.* **2016**, *105*, 214–224. <https://doi.org/10.1016/j.knosys.2016.05.031>.
8. Rabczuk, T. *Machine Learning in Modeling and Simulation: Methods and Applications*, 1st ed.; Computational Methods in Engineering and the Sciences Series; Springer International Publishing AG: Cham, Switzerland, 2023.
9. Karim, S.A.A. (Ed.) *Intelligent Systems Modeling and Simulation III: Artificial Intelligent, Machine Learning, Intelligent Functions and Cyber Security*; Studies in Systems, Decision and Control; Springer: Cham, Switzerland, 2024; Volume 553.
10. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv* **2024**, arXiv:1506.00019.
11. Salehinejad, H.; Sankar, S.; Barfett, J.; Colak, E.; Valaee, S. Recent Advances in Recurrent Neural Networks. *arXiv* **2017**, arXiv:1801.01078.
12. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>.
13. Hermansdorfer, L.; Trauth, R.; Betz, J.; Lienkamp, M. End-to-End Neural Network for Vehicle Dynamics Modeling. In *Proceedings of the 6th International Congress on Information Science and Technology*; El Mohajir, M., Ed.; IEEE: Piscataway, NJ, USA, 2020; pp. 407–412. <https://doi.org/10.1109/CiSt49399.2021.9357196>.
14. Sarker, I.H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Comput. Sci.* **2021**, *2*, 420. <https://doi.org/10.1007/s42979-021-00815-1>.
15. Wang, Z.; Ho, D.W.C.; Liu, X. State estimation for delayed neural networks. *IEEE Trans. Neural Netw.* **2005**, *16*, 279–284. <https://doi.org/10.1109/TNN.2004.841813>.
16. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Comput.* **2019**, *31*, 1235–1270. https://doi.org/10.1162/neco_a_01199.
17. Lim, B.; Arik, S.O.; Loeff, N.; Pfister, T. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *Int. J. Forecast.* **2019**, *37*, 1748–1764.
18. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
19. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 11106–11115.
20. Ludmann, L.; Zeitvogel, D.; Krantz, W.; Neubeck, J.; Wagner, A. Detection of Driving Dynamics Anomalies Using Deep Learning. In *Proceedings of the 2024 Stuttgart International Symposium on Automotive and Engine Technology*; Casal Kulzer, A., Reuss, H.C., Wagner, A., Eds.; ATZ Live; Springer: Wiesbaden/Heidelberg, Germany, 2024; pp. 382–393. https://doi.org/10.1007/978-3-658-45010-6_24.
21. Gu, A.; Dao, T.; Ermon, S.; Rudra, A.; Re, C. HiPPO: Recurrent Memory with Optimal Polynomial Projections. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1474–1487.
22. Gu, A.; Goel, K.; Ré, C. Efficiently Modeling Long Sequences with Structured State Spaces. *arXiv* **2021**, arXiv:2111.00396.
23. Dao, T.; Gu, A. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. *arXiv* **2024**, arXiv:2405.21060.
24. Fu, D.Y.; Dao, T.; Saab, K.K.; Thomas, A.W.; Rudra, A.; Ré, C. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. *arXiv* **2022**, arXiv:2212.14052.
25. Poli, M.; Massaroli, S.; Nguyen, E.; Fu, D.Y.; Dao, T.; Baccus, S.; Bengio, Y.; Ermon, S.; Ré, C. Hyena Hierarchy: Towards Larger Convolutional Language Models. In *Proceedings of the 40th International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2021; pp. 28043–28078.
26. Sun, Y.; Dong, L.; Huang, S.; Ma, S.; Xia, Y.; Xue, J.; Wang, J.; Wei, F. Retentive Network: A Successor to Transformer for Large Language Models. *arXiv* **2023**, arXiv:2307.08621.
27. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:2307.08621.
28. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:1711.10561.
29. Cai, S.; Mao, Z.; Wang, Z.; Yin, M.; Karniadakis, G.E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mech. Sin.* **2021**, *37*, 1727–1738. <https://doi.org/10.1007/s10409-021-01148-1>.
30. Huang, B.; Wang, J. Applications of Physics-Informed Neural Networks in Power Systems—A Review. *IEEE Trans. Power Syst.* **2023**, *38*, 572–588. <https://doi.org/10.1109/TPWRS.2022.3162473>.
31. Misyris, G.S.; Venzke, A.; Chatzivasileiadis, S. Physics-Informed Neural Networks for Power Systems. In *Proceedings of the 2020 IEEE Power Energy Society General Meeting (PESGM)*; IEEE: New York, USA, 2020. <https://doi.org/10.1109/PESGM41954.2020.9282004>.

32. Chen, R.T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D.K. Neural ordinary differential equations. In Proceedings of the 32nd Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018.
33. Kim, S.; Ji, W.; Deng, S.; Ma, Y.; Rackauckas, C. Stiff neural ordinary differential equations. *Chaos Interdiscip. J. Nonlinear Sci.* **2021**, *31*, 093122. <https://doi.org/10.1063/5.0060697>.
34. Dandekar, R.; Chung, K.; Dixit, V.; Tarek, M.; Garcia-Valadez, A.; Vemula, K.V.; Rackauckas, C. Bayesian Neural Ordinary Differential Equations. *arXiv* **2022**, arXiv:2012.07244.
35. Zhu, A.; Jin, P.; Zhu, B.; Tang, Y. On Numerical Integration in Neural Ordinary Differential Equations. In *Proceedings of the 39th International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2022; pp. 27527–27547.
36. Yan, H.; Du Jiawei.; Tan, V.Y.F.; Feng, J. On Robustness of Neural Ordinary Differential Equations. *arXiv* **2022**, arXiv:1910.05513.
37. Brunton, S.L.; Budišić, M.; Kaiser, E.; Kutz, J.N. Modern Koopman Theory for Dynamical Systems. *arXiv* **2021**, arXiv:2102.12086. <https://doi.org/10.48550/arXiv.2102.12086>.
38. Tu, J.H. *Dynamic Mode Decomposition: Theory and Applications*; Princeton University: Princeton, NJ, USA, 2013.
39. Colbrook, M.J. The Multiverse of Dynamic Mode Decomposition Algorithms. *arXiv* **2023**, arXiv:2312.00137. <https://doi.org/10.48550/arXiv.2312.00137>.
40. Yeung, E.; Kundu, S.; Hodas, N. Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems. In *Proceeding of the 2019 American Control Conference (ACC)*; IEEE: New York, NY, USA, 2019; pp. 4832–4839. <https://doi.org/10.23919/ACC.2019.8815339>.
41. Xiao, Y.; Zhang, X.; Xu, X.; Liu, X.; Liu, J. Deep Neural Networks With Koopman Operators for Modeling and Control of Autonomous Vehicles. *IEEE Trans. Intell. Veh.* **2023**, *8*, 135–146. <https://doi.org/10.1109/TIV.2022.3180337>.
42. Puente León, F.; Jäkel, H. *Signale und Systeme*, 7th ed.; Überarbeitete Auflage; De Gruyter Oldenbourg Studium, De Gruyter Oldenbourg: Berlin, Germany; Boston, MA, USA, 2019.
43. Heer, B.; Maußner, A., Eds. *Dynamic General Equilibrium Modeling: Computational Methods and Applications*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2009. <https://doi.org/10.1007/978-3-540-85685-6>.
44. Isermann, R.; Münchhof, M. *Identification of Dynamic Systems: An Introduction with Applications*; Springer: Berlin/Heidelberg, Germany, 2011.
45. Unbehauen, H. *Klassische Verfahren zur Analyse und Synthese Linearer Kontinuierlicher Regelsysteme, Fuzzy-Regelsysteme*, 15th ed.; Überarbeitete und Erweiterte Auflage; Studium Automatisierungstechnik; Vieweg + Teubner: Wiesbaden, Germany, 2008; Volume 1. <https://doi.org/10.1007/978-3-8348-9491-5>.
46. Granger, C.; Newbold, P. *Forecasting Economic Time Series*; Elsevier: Amsterdam, The Netherlands, 1986. <https://doi.org/10.1016/C2013-0-10756-8>.
47. Choutri, S.E.; Djehiche, B.; Mazhar, O. Empirical validation of novel non-asymptotic bounds on the least squares estimator for LTI systems with applications in economics. *Asian J. Econ. Bank.* **2025**, *9*, 491–509. <https://doi.org/10.1108/AJEB-12-2024-0147>.
48. Lopez, M.S.; Prasad, J.V.R. Linear Time Invariant Approximations of Linear Time Periodic Systems. *J. Am. Helicopter Soc.* **2017**, *62*, 1–10. <https://doi.org/10.4050/JAHS.62.012006>.
49. Lavretsky, E.; Wise, K.A., Eds. *Robust and Adaptive Control*; Advanced Textbooks in Control and Signal Processing; Springer International Publishing: Cham, Switzerland, 2024. <https://doi.org/10.1007/978-3-031-38314-4>.
50. Kalman, R.E. A New Approach to Linear Filtering and Prediction Problem. *J. Basic Eng.* **1960**, *82*, 35–45.
51. Riekert, P.; Schunck, T.E. Zur Fahrmechanik des gummibereiften Kraftfahrzeugs. *Ingenieur-Archiv* **1940**, *11*, 210–224. <https://doi.org/10.1007/BF02086921>.
52. Küçükay, F. *Grundlagen der Fahrzeugtechnik*; Springer: Wiesbaden, Germany, 2022. <https://doi.org/10.1007/978-3-658-36727-5>.
53. Schindler, E. *Fahrdynamik: Grundlagen des Lenkverhaltens und ihre Anwendung für Fahrzeugregelsysteme*; Kontakt & Studium, expert-Verl.: Renningen, Germany 2007; Volume 685.
54. Schramm, D.; Hiller, M.; Bardini, R. *Modellbildung und Simulation der Dynamik von Kraftfahrzeugen*; Springer: Berlin/Heidelberg, Germany, 2018. <https://doi.org/10.1007/978-3-662-54481-5>.
55. Zeitvogel, D.; Ahlert, A.; Neubeck, J.; Krantz, W.; Wiedemann, J.; Boone, F.; Kan, W. *An Innovative Test System for Holistic Vehicle Dynamics Testing*; SAE Technical Paper Series; SAE International: Warrendale, PA, USA, 2019. <https://doi.org/10.4271/2019-01-0449>.
56. Zeitvogel, D.; Krantz, W.; Neubeck, J.; Wagner, A. Holistic vehicle parametrization on a handling roadway. *Automot. Engine Technol.* **2022**, *7*, 209–216. <https://doi.org/10.1007/s41104-022-00107-7>.
57. Dozat, T. Incorporating Nesterov Momentum into Adam. In Proceedings of the 4th International Conference on Learning Representations, San Juan, PR, USA, 2–4 May 2016.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.