Beyond Static Models: Hypernetworks for Adaptive and Generalizable Forecasting in Complex Parametric Dynamical Systems

Pantelis R. Vlachas[a,*], Konstantinos Vlachas[a], Eleni Chatzi[a]

[a] *Department of Civil, Environmental, and Geomatic Engineering, ETH Zürich, Stefano-Franscini Platz 5, 8049, Zürich, Switzerland*

Abstract

Dynamical systems play a key role in modeling, forecasting, and decision-making across a wide range of scientific domains. However, variations in system parameters, also referred to as parametric variability, can lead to drastically different model behavior and output, posing challenges for constructing models that generalize across parameter regimes. In this work, we introduce the Parametric Hypernetwork for Learning Interpolated Networks (PHLieNet), a framework that simultaneously learns: (a) a global mapping from the parameter space to a nonlinear embedding and (b) a mapping from the inferred embedding to the weights of a dynamics propagation network. The learned embedding serves as a latent representation that modulates a base network, termed the *hypernetwork*, enabling it to generate the weights of a *target network* responsible for forecasting the system's state evolution conditioned on the previous time history. By interpolating in the space of models rather than observations, PHLieNet facilitates smooth transitions across parameterized system behaviors, enabling a unified model that captures the dynamic behavior across a broad range of system parameterizations. The performance of the proposed technique is validated in a series of dynamical systems with respect to its ability to extrapolate in time and interpolate and extrapolate in the parameter space, i.e., generalize to dynamics that were unseen during training. In all cases, our approach outperforms or matches state-of-the-art baselines in both short-term forecast accuracy and in capturing long-term dynamical features, such as attractor statistics.

Keywords: parametric dynamical systems, hypernetworks, forecasting, neural networks, nonlinear systems

1. Introduction

Accurate modeling and inference of the behavior of complex dynamical systems is essential for understanding, predicting, and controlling real-world phenomena across disciplines, including physics [1], biology [2], and neuroscience [3]. Applications span tasks related to weather modeling and forecasting [4, 5], extreme event prediction [6], fluid dynamics [7], financial markets modeling [8], the spread of diseases [9], and the operation of engineered systems [10]. While recent work in data-driven modeling has advanced our ability to learn the governing dynamics of complex systems, typically represented by ordinary and partial differential equations (ODEs and PDEs), most approaches focus mainly on capturing variations due to the influence of initial conditions [11, 12]. Yet, an equally critical aspect influencing dynamic response is *parametric variability*, which arises from changes in intrinsic system properties or external excitation characteristics [13]. Examples include the influence of the Reynolds number on flow regime transitions in fluid dynamics [14], and the role of carbon dioxide levels or solar radiation in shaping long-term climate trends [15]. These examples underscore the ubiquity of parametric systems, where dynamics are governed not only by initial conditions but also by smoothly or abruptly varying parameters.

Traditional physics-based approaches for modeling and forecasting complex dynamical systems of parametric nature rely on the derivation of mathematical models based on first principles, that is, established physical laws [16].

*Corresponding author

Email addresses: pvlachas@ethz.ch (Pantelis R. Vlachas), vlachas@ibk.baug.ethz.ch (Konstantinos Vlachas), chatzi@ibk.baug.ethz.ch (Eleni Chatzi)

To enable efficient simulation of parametric dynamical systems, the corresponding frameworks often employ model reduction techniques that approximate the full-order dynamics. These include projection-based methods [17, 18, 19], which construct reduced-order models via subspace projections (e.g., POD-Galerkin), as well as decomposition-based strategies [20, 21, 22]. In addition, meshless and interpolation-based approaches, including radial basis function methods and tensor decomposition techniques like Proper Generalized Decomposition (PGD), are also widely used, particularly when aiming to represent solution manifolds across a broad parameter space [16, 23, 21]. However, such strategies often face limitations when applied in chaotic or strongly nonlinear systems, particularly those exhibiting intricate interactions or multiple parametric dependencies. In addition, capturing and propagating the dynamics in many real-world applications requires resolving a wide range of scales, rendering the application of equation-based models computationally expensive or even intractable.

Despite recent advances in enhancing physics-based frameworks through scientific machine learning techniques [24, 25, 26], many modeling challenges persist, particularly in capturing complex dynamical systems characterized by strong nonlinearity, multiscale behavior, and parametric variability. In this context, hybrid approaches that integrate data with governing equations [27, 28] have gained traction for their ability to combine the expressiveness of data-driven models with the interpretability and structure of physical laws [23]. Among these, Physics-Informed Neural Networks (PINNs)[29, 30, 31] and their meta-learning extensions, such as Meta-PDE[32], HyperPINN [33], and Meta-Auto-Decoder frameworks [34]—represent prominent efforts to encode the PDE structure directly into the learning process. More recently, diffusion-based generative models have also been proposed as a means of incorporating physics-driven constraints into probabilistic modeling [35].

These techniques typically embed physical constraints, governing equations, or inductive biases tailored to specific PDEs, thereby enhancing model generalization and interpretability [36, 37]. However, they often rely on explicit knowledge of the underlying dynamics or the availability of the Jacobian, and may require intrusive access to integration schemes during training. As a result, their applicability can be limited in large-scale, high-dimensional systems, or in settings involving complex boundary conditions and broad parametric variability [38, 39, 40]. These scalability and mesh-dependency issues motivate the search for more flexible, non-intrusive alternatives capable of generalizing across diverse system configurations.

Parallel to hybrid methods, purely data-driven approaches have also been explored for modeling dynamical systems, particularly in the context of autoregressive time-series modeling. Classical architectures such as Reservoir Computers [41, 42, 43], Recurrent Neural Networks (RNNs) including Long Short-Term Memory (LSTM) networks [44] and Gated Recurrent Units (GRUs) [45, 46, 47, 48], and Transformers [49] are commonly employed for modeling dynamical systems. Furthermore, DeepONets [50], Neural Operators (NOs), including Fourier Neural Operators [51, 52], spectral NOs [53], and convolutional NOs [54, 55] offer a functional perspective by learning mappings between infinite-dimensional function spaces, thereby enabling efficient modeling of high-dimensional PDE systems. Similarly, neural ordinary differential equations (neural ODEs) [56, 57, 58] extend neural architectures to continuous-time domains by parameterizing the underlying dynamics through differentiable solvers.

Despite their promise, most data-driven methods train a single forecasting model per parametrization and struggle to generalize across unseen system parameters. As a result, the developed frameworks often struggle to generalize or extrapolate reliably across unseen dynamical regimes. Recent works in deep learning for parametric PDEs [59, 60, 61], PINNs [62], Neural ODEs [63], and Echo State Networks (ESNs) [64, 65, 66] also identify this gap and attempt to address it by stacking vector embeddings that combine state and parameter information [61] or by directly augmenting the state with the parameter, following earlier practices [65]. However, these methods still rely on shared model weights across all parameter configurations, which limits expressiveness and generalization when dynamics vary significantly across parameter space. We argue that this pitfall might evolve to a limiting factor hindering generalization, especially if the dynamics of the problem exhibit a wide variability, e.g., from simple oscillatory to fully chaotic behavior.

To overcome these limitations, we introduce *Parametric Hypernetwork for Learning Interpolated Networks (PHLieNet)*, a novel framework that explicitly conditions a hypernetwork on a learned embedding of the system parameters. PHLieNet maps each parameter vector to a continuous latent embedding by interpolating over learned embedding vectors associated with fixed anchor points. This embedding is then passed to a hypernetwork, which generates the weights of a forecaster network. The forecaster models the temporal evolution of the system's state, conditioned on short-term history. By dynamically adapting the weights of the target network to reflect the input system parameters, PHLieNet provides a unified and flexible modeling framework capable of capturing a wide range of dynamical

regimes. Crucially, the proposed approach allows differentiation through the hypernetwork with respect to the input parameters, enabling the computation of parameter-aware gradients. This capability, which is not standard in many existing frameworks, allows PHLieNet to support gradient-based training and inference over both state and parameter spaces. This stands in contrast to most state-of-the-art approaches, which either require training separate models for each parametrization or lack mechanisms for explicit parametric generalization.

Hypernetworks are neural networks that output the parameters of another neural network, known as the target network [67]. Originally introduced in the context of meta-learning[68, 69], hypernetworks have been used to generate initial weights or learning rules that enable rapid adaptation in low-data regimes or few-shot learning scenarios. Their success in this domain has spurred broader adoption across applications such as neural architecture search [70, 71, 72] and across a range of parametric tasks, including image retouching [73], style transfer [74], and differentiable pruning [75]. Despite this growing interest, the application of hypernetworks to the modeling of dynamical systems, particularly those governed by ODEs and PDEs, remains relatively nascent [76]. Early works demonstrate promising directions, including dynamic convolutions, implemented as hypernetworks, which have achieved promising results in short-range weather forecasting [77]. Furthermore, Berman et al. [78] proposed CoLoRA, which adapts low-rank weights of neural networks for new parameters and initial conditions, and Zheng et al. [79] introduced HyperCAN for modeling mechanical meta-materials under varying conditions. More recent innovations include Hypersolvers[80], which leverage hypernetworks to approximate higher-order terms in PDE solvers, and HyperPINNs[81, 82], which combine hypernetworks with physics-informed learning to improve generalization across varying PDE conditions. The latter employ meta-learning strategies and low-rank architectures to enable efficient and scalable approximations of solution manifolds.

Closest to our approach are frameworks such as Context-Informed Dynamics Adaptation (CoDA) [83], which uses a hypernetwork to condition a dynamics model on inferred environment-specific context vectors, and LEADS, by Yin et al. [84], which explicitly decomposes dynamics into shared and environment-specific components to generalize across environments. In contrast, PHLieNet leverages known system parameters directly, enabling more precise and expressive modeling across regimes.

Our method, supported by recent theoretical work that demonstrates the advantages of hypernetworks in terms of modularity and expressiveness [85], provides a scalable and effective alternative for data-driven inference on complex dynamical systems with parametric variability. By avoiding precomputed reduced spaces, explicit physical constraints, and fixed model architectures, PHLieNet adapts flexibly to a wide range of system behaviors, from fixed points and periodic orbits to chaos. A distinctive feature of our approach is its ability to interpolate and extrapolate across parameter space via learned embeddings, enabling coherent transitions between dynamical regimes.

In contrast to PHLieNet, approaches that embed parameters into fixed architectures often struggle to scale across wide parametric ranges or require extensive retraining for each new configuration. Similarly, context-based methods that infer latent representations from observed trajectories aim for parameter-agnostic generalization but are limited when precise parametric information is available and can be explicitly leveraged. By directly incorporating parametric variations into the embedding, PHLieNet ensures continuous and accurate forecasting across parameter space without compromising adaptability. This design supports a principled and scalable means of interpolating between distinct dynamical regimes within a single unified framework. Consequently, a direct comparison with approaches such as CoDA [83] and LEADS [84] is not entirely appropriate, as these methods are tailored for settings where parametric information is inferred from the data. In contrast, PHLieNet operates under the assumption that parameter values are explicitly available, leveraging them directly to enable more precise and flexible modeling across diverse dynamical regimes.

To validate our approach, we benchmark PHLieNet against parameter-agnostic temporal dynamics models and parameter-augmented models, including LSTMs and temporal CNNs with causal dilated convolutions. We evaluate performance on a diverse set of dynamical systems, including the Van der Pol oscillator, the Lorenz system, the Rössler attractor, and Chua's circuit. Across all systems, PHLieNet consistently outperforms or matches state-of-the-art models in short-term forecasting accuracy, while also improving long-term statistical fidelity, as measured by histogram errors on state trajectories and errors in the power spectral density. Moreover, we demonstrate that PHLieNet is expressive and capable of learning the complete spectrum of dynamics across parametric regimes, enabling it to extrapolate in time on seen parametric dynamics and generalize to parameters unseen during training. These results highlight the robustness and flexibility of PHLieNet, advancing the state of the art in data-driven, parametric modeling of dynamical systems and opening new avenues for real-world applications.

This paper is organized as follows: In Section 2, the PHLieNet framework is presented along with the considered baseline models. Section 3 presents the comparison metrics. In Section 4, we describe the benchmark dynamical systems, the models used for comparison, and the numerical results that highlight the efficiency and effectiveness of PHLieNet. Section 5 concludes the paper by summarizing our contributions, offering insights and suggesting directions for future research.

2. Methods

2.1. Parametric Dynamical Systems

A parametric dynamical system can be represented by a system of equations whose evolution depends on a parameter vector $\mathbf{p} \in \mathbb{R}^{D_\mathbf{p}}$, whose components may include physical constants, external conditions, or control inputs. Such systems can be formulated in continuous time as either ordinary differential equations (ODEs) or partial differential equations (PDEs). For ODEs, the state $\mathbf{x}(t) \in \mathbb{R}^{D_\mathbf{x}}$ evolves according to:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}(t), \mathbf{p}), \tag{1}$$

where $f : \mathbb{R}^{D_\mathbf{x}} \times \mathbb{R}^{D_\mathbf{p}} \to \mathbb{R}^{D_\mathbf{x}}$ defines the dynamics of the system as a function of the state $\mathbf{x}$ and parameters $\mathbf{p}$.

In the discrete-time setting, the evolution of the system is modeled as a sequence of state transitions governed by a nonlinear update function:

$$\mathbf{x}_{t+1} = \Phi(\mathbf{x}_t, \mathbf{p}), \tag{2}$$

where $\Phi : \mathbb{R}^{D_\mathbf{x}} \times \mathbb{R}^{D_\mathbf{p}} \to \mathbb{R}^{D_\mathbf{x}}$ represents a possibly learned or explicitly defined discrete-time transition map. This formulation encompasses a wide range of integration schemes, including explicit or implicit methods (e.g., Euler or Runge-Kutta [86]), as well as data-driven alternatives such as Neural ODEs [87]. Our method is compatible with any such formulation; however, for simplicity and clarity, we adopt a first-order integration scheme in this work expressed as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t \cdot f(\mathbf{x}_t, \mathbf{p}), \tag{3}$$

where $\mathbf{x}_t$ is the state at time t, $\mathbf{p}$ is the parameter vector, $f(\mathbf{x}_t, \mathbf{p})$ is the time derivative (i.e., the gradient of the state), and Δt is the discretization step size. This formulation corresponds to an explicit Euler integration of the continuous-time dynamics.

The parameter vector $\mathbf{p}$ can vary across simulations, resulting in a diverse set of phenomena such as fixed points, periodic orbits, and chaotic dynamics. This variation enables the study of how changes in $\mathbf{p}$ influence the evolution of the system. A key challenge in parametric dynamical systems is efficiently capturing the relationship between $\mathbf{p}$ and the resulting dynamics, especially across a wide range of parameter values or when $\mathbf{p}$ itself varies over time.

2.2. Learning Temporal Dynamics

In forecasting dynamical systems, we want to learn an approximation of f, using a parametrized model f^{w_f} by minimizing some reconstruction or prediction error. The variable w_f represents the parameters of the approximator f. In case of a neural network, for example, w_f is the set of all weights and biases of the network. To capture non-Markovian effects, account for missing information, and improve performance in long-term forecasting, the approximator often incorporates information from the previous history of the state. Models that are explicitly designed to process sequential data, such as recurrent neural networks (RNNs)[88, 89] and Temporal Convolutional Neural Networks with causal dilated convolutions (TCNN CD)[90], are natural choices for this task because they can effectively capture and leverage temporal patterns and dependencies in time series. In what follows, we use the term Temporal Dynamic Networks (TDNs) to refer to such models, grouping together architectures specifically designed for learning from sequential data. Such approaches align with Takens' theorem [91], which demonstrates that a system's dynamics can be reconstructed from a time-delayed embedding of its state under certain conditions.

In the case of TDNs, the state evolution is approximated by:

$$\tilde{\mathbf{x}}_{t+1} = \int \tilde{\dot{\mathbf{x}}}_t dt, \quad \tilde{\dot{\mathbf{x}}}_t = f^{w_f}(\underbrace{\mathbf{x}_t, \ldots, \mathbf{x}_{t-\mathrm{ISL}+1}}_{\text{history}}\,;\, \mathbf{p}). \tag{4}$$

where $\tilde{\bullet}$ denotes inferred quantities, the networks are used to approximate the dynamics (time derivative of the state), and we truncate the dependence on the previous states after ISL timesteps (state-less formulation).

To capture the influence of parametric variability, expressed by $\mathbf{p}$, the network needs to be fitted to trajectories from the parametrized dynamics. Let us assume we have response data from a set of parameters $P_{train} = \{p_1, \ldots, p_{N_p}\}$ with total $|P_{train}| = N_P$ parametrizations. For each parameterization, we assume N_{ics} trajectories, each one representing different initial conditions (after eliminating initial transients) and consisting of N_T timesteps. Thus, the train data are $\mathbf{X} \in \mathbb{R}^{N_p \times N_{ics} \times N_T}$. In turn, the neural network employed as an approximator f is trained to minimize the prediction loss across time. Specifically, for a given batch of a trajectory of the data $x^{j,k}_{t-\mathrm{ISL}+1}, \ldots, x^{j,k}_t, x^{j,k}_{t+1}$, corresponding to parameter $p_j \in P_{train}$, and initial condition $k \in \{1, \ldots, N_{ics}\}$, the loss is defined as:

$$\mathcal{L}_{j,k,t} = \|\dot{\mathbf{x}}_t - \tilde{\dot{\mathbf{x}}}_t\|^2 = \|\dot{\mathbf{x}}_t - f^{w_f}(\mathbf{x}_t, \ldots, \mathbf{x}_{t-\mathrm{ISL}+1}; \mathbf{p})\|^2. \tag{5}$$

The network parameters are optimized by minimizing the loss over the entire parameter set, across all initial conditions and timesteps, as follows:

$$w_f = \arg \min_{\substack{p_j \in P_{train} \\ k \in \{1,\ldots,N_{ics}\} \\ t \in \{1,\ldots,N_T\}}} \mathcal{L}_{j,k,t}. \tag{6}$$

Given a sufficiently diverse parameter set P_{train} to capture the system's behavior, the trained network can be used to forecast unseen dynamics, extrapolate in time, and even generalize to unseen parameters. The latter is a significantly more challenging task, as varying parameters can profoundly alter the attractor structure and overall dynamics.

2.2.1. Recurrent Neural Networks

A natural choice to approximate the time derivative $f^{w_f}(\mathbf{x}_t, \mathbf{p})$ is a Long Short-Term Memory (LSTM) network. LSTMs are particularly effective at capturing long-range dependencies through a gated mechanism that controls the flow of information over time steps [44]. Such models have been successfully applied in learning dynamical system representations [45]. The LSTM updates its hidden state $\mathbf{h}_t$ and the cell state $\mathbf{c}_t$ at each time step t based on the previous states $\mathbf{h}_{t-1}$, $\mathbf{c}_{t-1}$, and the current input $\mathbf{x}_t$. The update equations are given by:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \tag{7}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f), \tag{8}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o), \tag{9}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c), \tag{10}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{11}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{12}$$

where σ denotes the sigmoid activation function and $\odot$ represents element-wise multiplication. The vectors $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the input, forget, and output gates, respectively. All parameters $\mathbf{W}_*$, $\mathbf{U}_*$, and $\mathbf{b}_*$ are learned during training and collectively constitute the parameter set w_f, which is not to be confused with the parametric dependencies of the model $\mathbf{p}$. The LSTM approximates the time derivative of the state f^{w_f} as in Equation (4). During training, the model minimizes the loss defined in Equation (6) across all parameterizations and initial conditions, allowing f^{w_f} to learn the parameter-dependent dynamics of the system.

2.2.2. Causal Dilated Temporal CNN (CD-TCNN)

Another effective solution to model sequential data, while ensuring causality, is a Causal Dilated Temporal Convolutional Network (CD-TCNN). This architecture leverages dilated convolutions to efficiently capture long-range dependencies without relying on recurrent structures. Specifically, each convolutional layer uses a dilation factor that grows exponentially with the layer index:

$$d_i = 2^i, \quad i = 0, 1, \ldots, L-1,$$

where L is the number of layers. This exponentially increasing dilation pattern ensures that the receptive field grows rapidly, enabling the model to capture temporal dependencies over large windows. For a 1D temporal convolution at

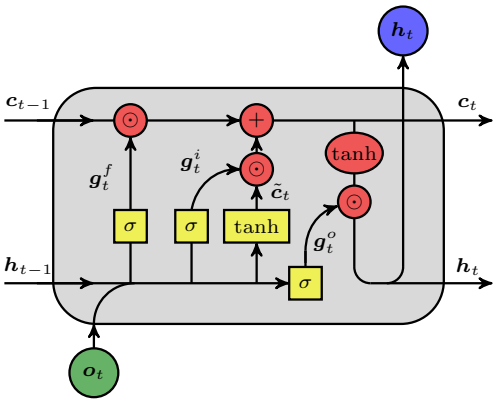


Figure 1: Information flow of a Long Short-Term Memory (LSTM) Cell.

time step t, the output y_t is computed as:

$$y_t = \sum_{i=0}^{k-1} w_i \cdot x_{t-i\cdot d},$$

where w_i are the learned convolutional weights, k is the kernel size, and d is the dilation factor. This formulation allows each output y_t to aggregate information from a causal receptive field of past states $(x_t, x_{t-1}, \ldots, x_{t-\mathrm{ISL}})$, without accessing future inputs. Causal padding is used in each convolutional layer to prevent information leakage from future time steps. For a convolutional kernel of size k, the causal padding at layer i is computed as:

$$\text{padding}_i = d_i \cdot (k-1).$$

The total receptive field R of the network is then:

$$R = 1 + \sum_{i=0}^{L-1} (k-1) \cdot d_i.$$

After each convolution, a smooth nonlinearity (SiLU activation) is applied:

$$\mathrm{SiLU}(x) = x \cdot \sigma(x),$$

where $\sigma(x)$ is the sigmoid function. Unlike recurrent networks, CD-TCNN omits fully connected layers, relying instead on a final 1×1 convolution to project the learned features to the output dimension. This approach reduces the parameter count while maintaining sufficient expressiveness for temporal data modeling.

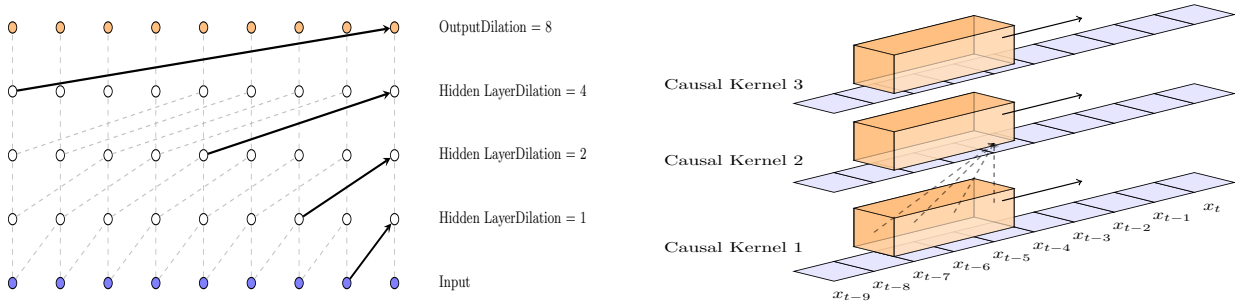
In this work, the number of layers L is automatically determined by the length of the input sequence and the size of the kernel to ensure that the receptive field covers the necessary temporal context. To determine the minimum number of layers L required to cover an input sequence of length ISL, we analytically invert the receptive field formula. Given a kernel size $k > 1$, the receptive field grows as $R = 1 + (k-1)(2^L - 1)$. Solving for L, we obtain:

$$L = \left\lceil \log_2 \left(\frac{\mathrm{ISL} - 1}{k-1} + 1 \right) \right\rceil.$$

This ensures that the receptive field spans at least ISL time steps, allowing the network to access the full temporal context during training. For example, with $k = 5$, this results in $L = 3$ layers for sequences of length $\mathrm{ISL} = 16$, and $L = 4$ layers for $\mathrm{ISL} = 32$. More information on the hyperparameters of the models used is reported in Appendix Appendix B. The dilation pattern is illustrated in Figure 2a. The causal temporal kernel is illustrated in Figure 2b.

2.3. Modeling the Parametric Dependency

The main challenge addressed in this work lies in formulating an expressive functional representation for Equation (4). We distinguish between three modeling paradigms: (i) a trivial, parameter-agnostic formulation; (ii) the established approach of state augmentation for parametric modeling, as reviewed in section 1; and (iii) the proposed PHLieNet framework, which introduces a principled alternative.



(a) Visualization of the exponentially increasing dilation pattern. Each hidden layer employs a convolutional kernel with a dilation factor $d_i = 2^i$, which enables the network to efficiently capture long-range dependencies across input sequences. The final output aggregates information from a broad receptive field that spans multiple temporal scales.

(b) Illustration of the causal convolutional kernels. Each kernel processes the current and past input states, thereby preserving the temporal order and preventing information leakage from future time steps. Different layers have different dilation factors, allowing the receptive field to expand and integrate long-range dependencies while respecting causality.

Figure 2: Illustration of the exponentially increasing dilation pattern and the causal convolutional kernels in the Causal Dilated Temporal Convolutional Network (CD-TCNN).

2.3.1. Parametric-Agnostic Case

A straightforward way to handle parametric dependency is to treat all parametric dynamics uniformly, assuming that $\mathbf{p}$ does not significantly change the functional form, effectively ignoring the parametric dependency. Alternatively, we may approximate the complete f^{w_f} without assuming explicit knowledge or dependence on $\mathbf{p}$. This form is referred to as the parametric-agnostic model. The functional form then becomes:

$$\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \Delta t \cdot f^{w_f}(\mathbf{x}_t, \ldots, \mathbf{x}_{t-\mathrm{ISL}+1}). \tag{13}$$

Any temporal dynamics model such as an LSTM, a GRU, or a TCN-CD can then be used to model Equation (13).

2.3.2. State Augmentation

Another straightforward way to handle parametric dependency is to augment the hidden state. As a result, f^{w_f} becomes a neural network that receives as input a vector with the state concatenated to the parameters. Thus, the augmented state is given by:

$$\mathbf{u}_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{p} \end{bmatrix} \in \mathbb{R}^{D_{\mathbf{x}}+D_{\mathbf{p}}} \tag{14}$$

and the state evolution is approximated by

$$\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \Delta t \cdot f^{w_f}(\mathbf{u}_t, \ldots, \mathbf{u}_{t-\mathrm{ISL}+1}). \tag{15}$$

2.4. Parametric Hypernetwork with Learned Interpolation Embedding

In systems where the dynamics heavily depend on the external parameters $\mathbf{p}$, learning a single set of network coefficients w_f, like proposed in Section 2.3.1, may not adequately capture the full range of behaviors induced by different values of $\mathbf{p}$. Moreover, appending the parameters to the state of the system, as in Section 2.3.2, may not be adequate, as the parameters might affect the structural form of f. Instead, concatenation hinders flexibility in the expressiveness of f. To address this challenge, in this work, we utilize a hypernetwork which dynamically generates the coefficients w_f of the network(s) used to model f conditioned on the input parameter vector $\mathbf{p}$.

Hypernetworks, as introduced by Ha et al. [67], provide a framework for generating the coefficients w_f of another neural network, meaning the corresponding weights and biases. Instead of directly learning a function f that models the system's dynamics for each possible value of $\mathbf{p}$, a hypernetwork can be used to generate the coefficients of f conditioned on $\mathbf{p}$. However, directly conditioning the hypernetwork on the raw parameters $\mathbf{p}$ presents significant challenges, as the network might struggle to distinguish between qualitatively different dynamical regimes, especially when transitions are nonlinear or discontinuous. This leads to poor generalization across regimes and necessitates some form of representation learning or clustering to structure the parameter space. A related approach using linear

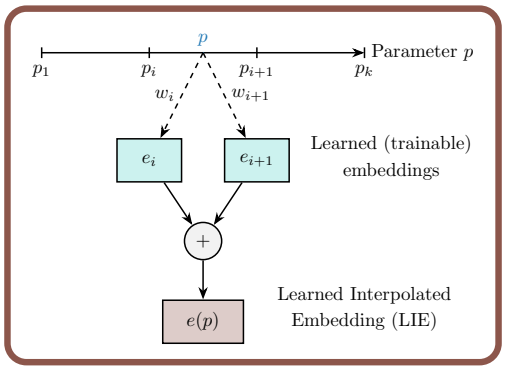


Figure 3: Overview of the learned interpolation mechanism used in PHLieNet. An input parameter vector $\mathbf{p}$ is used to compute interpolation weights $\{a_i(\mathbf{p})\}$ over a set of anchor points $\{\mathbf{p}^{(i)}\}$. Each anchor is associated with a learned embedding $\mathbf{e}^{(i)}$. The final embedding $\mathbf{e}(\mathbf{p}) = \sum_i a_i(\mathbf{p})\mathbf{e}^{(i)}$ is a convex combination of the learned embeddings, which is then used as input to the hypernetwork to generate the coefficients of the target network. This structure enables generalization across parameter space by smoothly interpolating between known regimes.

RNNs and a linear embedding of the parameter vector was proposed in [92]. Extending such approaches to nonlinear systems and nonlinear target networks remains an open challenge.

In this work, we adopt a different approach. We capture the parametric dependence of the system through a structured two-stage mechanism, although the entire architecture is trained end-to-end. First, the input parameters $\mathbf{p}$ are mapped to a continuous embedding via linear interpolation over a set of learned anchor embeddings. Second, the continuous embedding is passed to a hypernetwork that generates the coefficients of the target network. We refer to this method as *Parametric Hypernetwork with Learned Interpolated Embedding (PHLieNet)*. A detailed description of its implementation follows.

2.4.1. Step 1: Learned Interpolated Embedding

Let $\{\mathbf{p}^{(i)}\}_{i=1}^{N_\mathbf{e}}$ be a set of anchor parameter vectors and let $\{\mathbf{e}^{(i)}\}_{i=1}^{N_\mathbf{e}} \subset \mathbb{R}^{D_\mathbf{e}}$ be their corresponding learned embeddings. The embeddings are learned, so the matrix $w_e = [\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(N_\mathbf{e})}]$ represents the weights of this layer. Given a new parameter vector realization $\mathbf{p}^j$, we compute the interpolation weights $\{\alpha_i(\mathbf{p}^j)\}_{i=1}^{N_\mathbf{e}}$ such that:

$$\sum_{i=1}^{N_\mathbf{e}} \alpha_i(\mathbf{p}^j) = 1, \quad \alpha_i \geq 0, \tag{16}$$

and define the embedding of $\mathbf{p}^j$ as:

$$\mathbf{e}(\mathbf{p}^j) = \sum_{i=1}^{N_\mathbf{e}} \alpha_i(\mathbf{p}^j)\, \mathbf{e}^{(i)}. \tag{17}$$

In our setting, the parameter space is one-dimensional and we use simple linear interpolation between learned anchor embeddings. In higher-dimensional parameter spaces, the interpolation weights can be generalized to ensure convex combinations of nearby anchors. For instance, the weights can be designed to reflect proximity in parameter space while maintaining smoothness and stability in the resulting embedding. This allows the method to interpolate and, to some extent, extrapolate across a wide range of parameterized dynamical regimes in a continuous and differentiable manner. The learned interpolated embedding layer is illustrated in Figure 3.

The number of anchor embeddings $N_\mathbf{e}$ plays a crucial role in determining the expressiveness and generalization ability of the PHLieNet framework. On the one hand, $N_\mathbf{e}$ must be sufficiently smaller than the number of parametrization included in the training data, ensuring that the network is forced to learn meaningful interpolations in embedding space rather than memorizing the dynamics associated with each training parameter. This promotes generalization and encourages the model to capture shared structure across parametric regimes. On the other hand, if $N_\mathbf{e}$ is too small, the resulting embedding space may lack the capacity to represent the diversity of dynamics present in the dataset, particularly in systems exhibiting rich or highly nonlinear behaviors. In such cases, the model may fail to generate sufficiently expressive target networks, limiting its ability to accurately forecast dynamics across parameter space. Therefore, $N_\mathbf{e}$

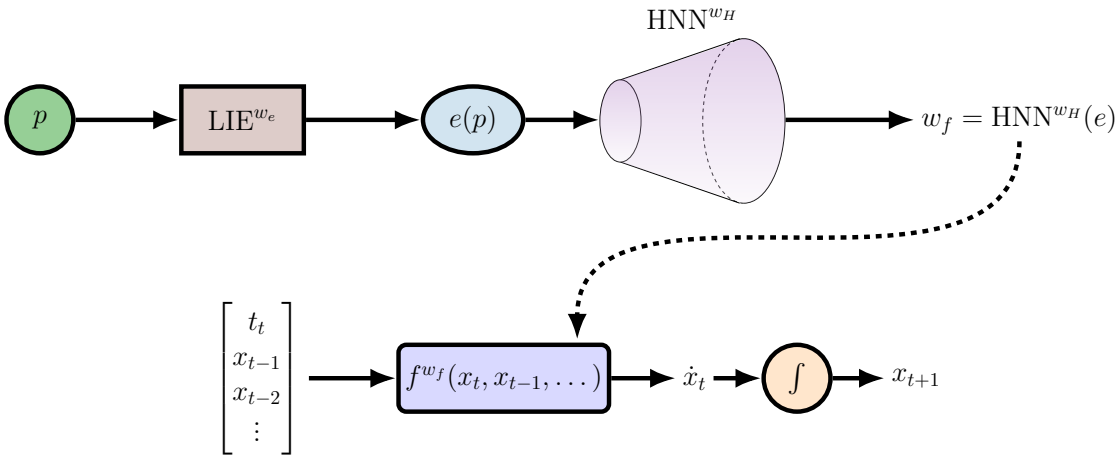


Figure 4: PHLieNet framework: The parameter p is passed through the Learned Interpolated Embedding (LIE) layer to produce an embedding $e(p)$. This embedding is used by the Hypernetwork to generate the weights of a target network (e.g., a causal dilated CNN or LSTM), which is then used to model and integrate the system's temporal dynamics.

must be carefully chosen to balance the trade-off between interpolation capacity and model expressiveness, ensuring robust generalization while retaining sufficient representational power.

2.4.2. Step 2: Parameter Generation via Hypernetwork.

The second stage of our framework involves the generation of the coefficients w_f of the target temporal dynamics model using a hypernetwork. The hypernetwork, denoted by HNN, takes as input the embedding $\mathbf{e}(\mathbf{p}^j) \in \mathbb{R}^{D_\mathbf{e}}$ produced in Step 1 and outputs the parameters $w_f \in \mathbb{R}^{|w_f|}$ of network f. Formally, this mapping is defined as:

$$w_f = \mathrm{HNN}\left(\mathbf{e}(\mathbf{p}^j); w_H\right), \tag{18}$$

where $\mathrm{HNN} : \mathbb{R}^{D_\mathbf{e}} \to \mathbb{R}^{|w_f|}$ is the hypernetwork parameterized by coefficients w_H, and $|w_f|$ denotes the number of weights of the target model f. The target temporal dynamics network f^{w_f}, with coefficients w_f generated by the hypernetwork, is then used to model the time derivative of the system's state based on a history of observations, as in Equation (4):

$$\tilde{\dot{\mathbf{x}}}_t = f^{w_f}\left(\mathbf{x}_t, \dots, \mathbf{x}_{t-\mathrm{ISL}+1}\right) = f^{\mathrm{HNN}\left(\mathbf{e}(\mathbf{p}^j); w_H\right)}\left(\mathbf{x}_t, \dots, \mathbf{x}_{t-\mathrm{ISL}+1}\right). \tag{19}$$

The proposed PHLieNet framework is illustrated in Figure 4. By conditioning the coefficients w_f of the temporal dynamics model on the system parameters $\mathbf{p}$, the hypernetwork enables flexible and continuous adaptation to a wide range of dynamical regimes. This architecture allows a single, unified model to generalize across different parameter configurations, eliminating the need to train separate models for each regime. More details on the hypernetwork architecture are provided in Appendix Appendix A.

Figure 5 offers a methodological perspective on PHLieNet, emphasizing interpolation in the weight space. The process begins with the linear combination of task-specific embeddings, representing different dynamical regimes, in a shared latent space. These interpolated embeddings are then mapped by a hypernetwork to generate model weights, effectively performing interpolation in the weight space. Crucially, this nonlinear interpolation induces meaningful transitions in the phase space dynamics of the resulting models. By focusing on the weight space (model space), rather than the state or parametric space, this approach enables coherent and controllable blending of dynamical behaviors across tasks or parameter regimes.

3. Evaluation Metrics

To evaluate the forecasting performance of parametric models across different dynamical systems, we employ complementary metrics that capture different aspects of predictive accuracy. Namely, we will employ the time evo-

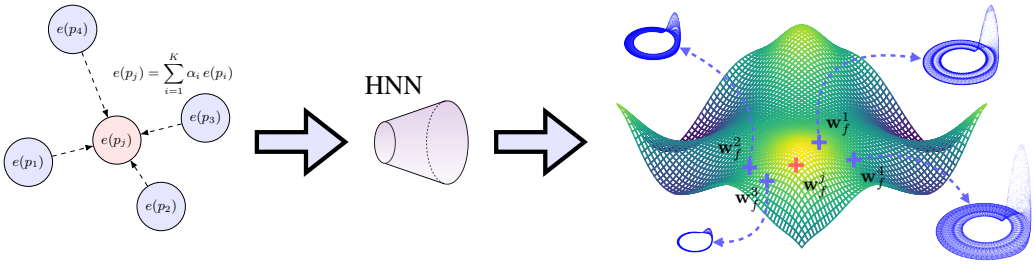


Figure 5: Overview of the three-step modeling process: (1) Linear combination of task embeddings in the shared embedding space; (2) Transformation of the embedding through a hypernetwork to generate corresponding model weights; (3) Nonlinear interpolation in the model (weight) space, which induces interpolation in the resulting phase space dynamics. This procedure enables smooth transitions across parametric tasks while preserving expressive dynamical behavior.

lution of the normalized root-mean-square error (NRMSE), the total root-mean-square error (RMSE), the Time-to-Threshold (TtT), the power spectrum error, and the histogram L1 norm. These metrics jointly assess short-term prediction accuracy, frequency content, and long-term statistical behavior, and are briefly presented below.

3.1. Time Evolution of the Normalized RMSE

To evaluate the accuracy of model predictions over time, we computed the normalized root mean squared error (NRMSE) as a function of time. The NRMSE at time t is defined as:

$$\mathrm{NRMSE}(t) = \frac{\|\widetilde{\mathbf{x}}(t) - \mathbf{x}(t)\|_2}{\sqrt{\sigma^2 + \varepsilon}}, \tag{20}$$

where $\widetilde{\mathbf{x}}(t)$ and $\mathbf{x}(t) \in \mathbb{R}^{D_\mathbf{x}}$ are the predicted and true states, and σ^2 is the variance of the ground truth states aggregated across all parameters, initial conditions, times, and dimensions. The small constant ε ensures numerical stability. We calculate the mean of the NRMSE across initial conditions to characterize the predictive performance of different models. This time-resolved error curve provides information on how the accuracy degrades during extrapolation in time.

3.2. Root Mean Square Error (RMSE)

The RMSE quantifies the average magnitude of the prediction error over time and across dimensions. Given predicted trajectories $\widetilde{\mathbf{X}} \in \mathbb{R}^{N_p \times N_{ics} \times N_T \times D_\mathbf{x}}$ and true trajectories $\mathbf{X}$, it is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_p N_{ics} N_T D_\mathbf{x}} \sum_{q=1}^{N_p} \sum_{i=1}^{N_{ics}} \sum_{t=1}^{N_T} \sum_{d=1}^{D_\mathbf{x}} \left(\tilde{x}^q_{i,t,d} - x^q_{i,t,d}\right)^2}, \tag{21}$$

where N_p is the number of total parameters in the dataset, N_{ics} is the number of initial conditions per parameter, N_T is the number of time steps and $D_\mathbf{x}$ is the dimensionality of the state.

3.3. Time-to-Threshold (TtT)

The Time-to-Threshold (TtT) metric quantifies the duration for which the predicted trajectory remains within an acceptable error margin relative to the ground truth. We define the TtT based on the NRMSE defined in Section 3.1. It measures the maximum continuous time during which the normalized error stays below a specified threshold θ_{rel}. The Time-to-Threshold *ttt* is given by:

$$\boxed{\mathrm{TtT}_{\theta_{\mathrm{rel}}} = \max\left\{t \;\middle|\; \mathrm{NRMSE}(t') < \theta_{\mathrm{rel}} \quad \forall\, t' \le t\right\} \cdot \Delta t}, \tag{22}$$

where θ_{rel} is the predefined relative error threshold and Δt is the simulation time step. In practice, the TtT is calculated as the maximum continuous time before the relative error first exceeds the threshold θ_{rel}, averaged (or otherwise aggregated) over multiple initial conditions to obtain a robust measure of predictive performance.

3.4. Power Spectrum Error

The power spectrum error measures the discrepancy between the frequency content of predicted and true trajectories. For each dimension, we compute the power spectral density (PSD) using a real-valued Fast Fourier Transform (FFT). Given a signal $x(t) \in \mathbb{R}$, which is a component of the state evolution $\mathbf{x} \in \mathbb{R}^{D_{\mathbf{x}}}$ sampled at frequency $f_s = 1/\Delta t$, the frequency spectrum in decibels (dB) is calculated as:

$$\text{PSD}(f) = 20 \log_{10}\left(\frac{2}{N}|\mathcal{F}[x](f)|\right), \tag{23}$$

where $\mathcal{F}[x](f)$ denotes the FFT of the signal $x(t)$, and N is the number of time steps. The power spectrum error is then defined as the average ℓ_1-distance between the predicted and true spectra across dimensions:

$$\text{Spectrum Error} = \frac{1}{D_{\mathbf{x}}} \sum_{d=1}^{D_{\mathbf{x}}} \frac{1}{F} \sum_{f=1}^{F} \left|\text{PSD}_{\text{pred}}^{(d)}(f) - \text{PSD}_{\text{true}}^{(d)}(f)\right|, \tag{24}$$

where F is the number of frequency bins.

3.5. Histogram L1 Norm

To assess the long-term statistical behavior of the system, we compute the L1 distance between histograms of predicted and true state values. Given flattened state trajectories histograms are computed using a common binning strategy, and the L1 norm between the normalized histograms is calculated:

$$\text{L1}_{\text{Hist}} = \sum_{b=1}^{B} \left|T(b) - \widetilde{T}(b)\right|, \tag{25}$$

where B is the number of bins, and T, $\widetilde{T}$ are the normalized bin frequencies of the true and predicted states, respectively. For multivariate systems discussed in this work, the L1 distance is computed per dimension and averaged.

4. Applications

Our framework is implemented in PyTorch [93]. We have developed the PHLieNet framework implementation on the hypernetwork library [94], extending it to suit our needs. Our runs are conducted on the Euler supercomputing cluster at ETH Zurich. Each run utilizes an RTX 3090 GPU, with 10 CPUs per task and 1 GB of RAM per CPU. For benchmarking, we consider the networks summarized in Table 1.

Table 1: Reference table for compared networks.

Reference name	Description
FFNN-A	A feedforward neural network agnostic to the parameter.
FFNN-P	A feedforward neural network, augmented on the state with the parameter.
LSTM-A	An LSTM neural network agnostic to the parameter.
LSTM-P	An LSTM neural network, augmented on the state with the parameter.
TCNN-CD-A	A causal dilated temporal CNN agnostic to the parameter.
PHLieNet	The proposed Parametric Hypernetwork with Learned Interpolation Embedding

Although the architecture of the TCNN-CD is quite effective, as it leverages temporal invariances in the data, it is not straightforward to design a state-augmented TCNN-CD. In fact, for complex architectures where the input

modality requires spatial invariance (handled by convolutions), it is not straightforward to incorporate other modalities. In our case, the parameter can be seen as an additional modality. Here, the proposed PHLieNet framework offers a compelling alternative: TCNN-CD can be used as the target network, while the parameter modulates the kernels via the hypernetwork. Thus, in the following, we use the TCNN-CD as the target network, combining the best of both worlds: parameter-based modulation that interpolates over the parametric space, and the TCNN-CD as the dynamics propagator. Regarding hyperparameters, we did not perform an exhaustive search for optimal values. Our aim is not to achieve the absolute best possible benchmark performance, but rather to demonstrate the viability of our proposed modeling framework.

The networks are trained on a dataset that contains trajectories $\mathbf{X} \in \mathbb{R}^{N_p \times N_{ics} \times N_T \times D_{\mathbf{x}}}$ generated from a set of parameter values denoted as P_{train}, with cardinality N_p. For validation, we use the same set of parameters P_{train} but with trajectories initialized from different initial conditions. We evaluate the methods in the auto-regressive forecasting (AR) setting. For testing, we consider two distinct tasks. In the first task, we evaluate the extrapolation in the time domain on the values of the seen parameters (AR-T), so $P_{\text{test}}^{\text{AR-T}} = P_{\text{train}}$. In the second task (AR-P), we assess the networks' ability to generalize to unseen parameter values, a significantly more challenging task. Here, trajectories are sampled from a testing set of parameters $P_{\text{test}}^{\text{AR-P}}$.

We now apply the proposed framework to a diverse set of dynamical systems chosen to reflect a broad range of behaviors and modeling challenges. These systems include the Van der Pol oscillator, the Lorenz 3D system, the Rössler attractor, and the dynamics of Chua's circuit. Together, they span nonlinear oscillations, chaotic attractors, and complex bifurcation patterns. For each system, we report results on both the AR-T and AR-P tasks. The following sections provide details on each system and the corresponding experimental setup.

4.1. The Van der Pol Oscillator

The Van der Pol oscillator is a second-order non-linear dynamical system originally introduced by Balthasar van der Pol in the 1920s while studying electrical circuits containing vacuum tubes [95]. The Van der Pol oscillator is governed by the following second-order differential equation:

$$\frac{d^2 x_1}{dt^2} - \mu(1 - x_1^2)\frac{dx_1}{dt} + x_1 = 0, \tag{26}$$

which can be rewritten as a system of first-order ODEs:

$$\frac{dx_1}{dt} = x_2, \tag{27}$$

$$\frac{dx_2}{dt} = \mu(1 - x_1^2)x_2 - x_1, \tag{28}$$

where $\mathbf{x} = [x_1, x_2]^T$ is the state variable with dimensionality $D_{\mathbf{x}} = 2$ and $\mu \in \mathbb{R}^+$ is a scalar parameter that controls the degree of nonlinearity and the damping intensity. The system exhibits qualitatively different behaviors depending on the value of μ. For small μ, it behaves like a near-harmonic oscillator, whereas for larger μ, it transitions to nonlinear relaxation oscillations with slow dynamics punctuated by rapid jumps. This rich variety of dynamics renders the Van der Pol oscillator an important model for studying non-equilibrium phenomena and oscillatory behavior in biological, chemical, and engineering systems [96].

In our work, we vary the parameter μ in the range $P_{\text{train}} = \{1, 2, \ldots, 8\}$ to explore various dynamical regimes. We use a fourth-order Runge-Kutta integrator with a solver time step $\delta t = 0.001$ and sample observations every $\Delta t = 0.05$ units of time. For the training and validation datasets, we generate a single long trajectory with $N_{ics}^{\text{train}} = N_{ics}^{\text{val}} = 1$, each simulated up to 200 time units, i.e., $N_T^{\text{train}} = N_T^{\text{val}} = 4\text{K}$ timesteps. Train and validation data are therefore $\mathbf{X} \in \mathbb{R}^{8\times1\times4\text{K}\times2}$. The noise level during training is set to $\sigma_{\text{noise}} = 10\%$

In contrast, for the test data, we simulate shorter trajectories up to 20 time units, i.e. $N_T^{\text{test}} = 400$, using $N_{ics}^{\text{test}} = 100$ distinct initial conditions per parameter value. The initial conditions are sampled independently from a uniform distribution in the square $[-5, 5]^2 \subset \mathbb{R}^2$. This setup enables robust evaluation across a broad range of initial conditions and supports both temporal and parametric generalization.

As elaborated in Section 4, we define two test sets to evaluate the performance of the examined algorithms in autoregressive testing scenarios: a time extrapolation set using the same μ values as in training, but with unseen initial

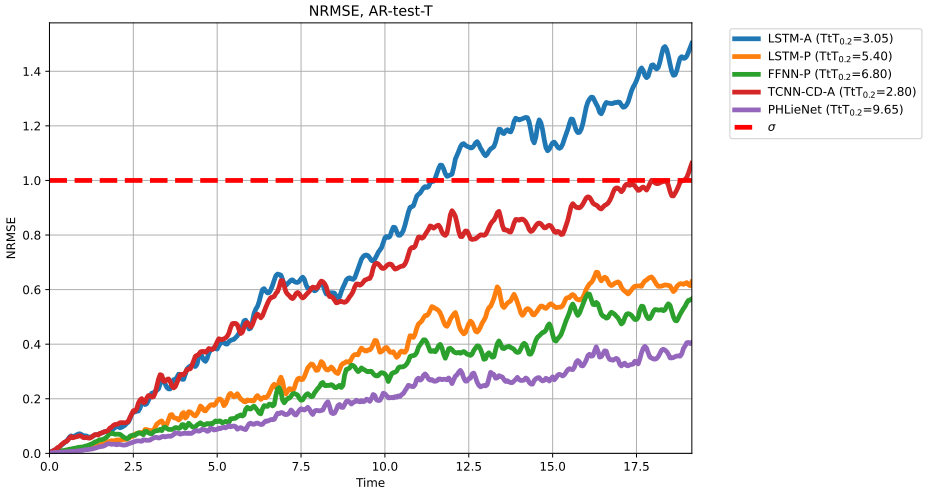
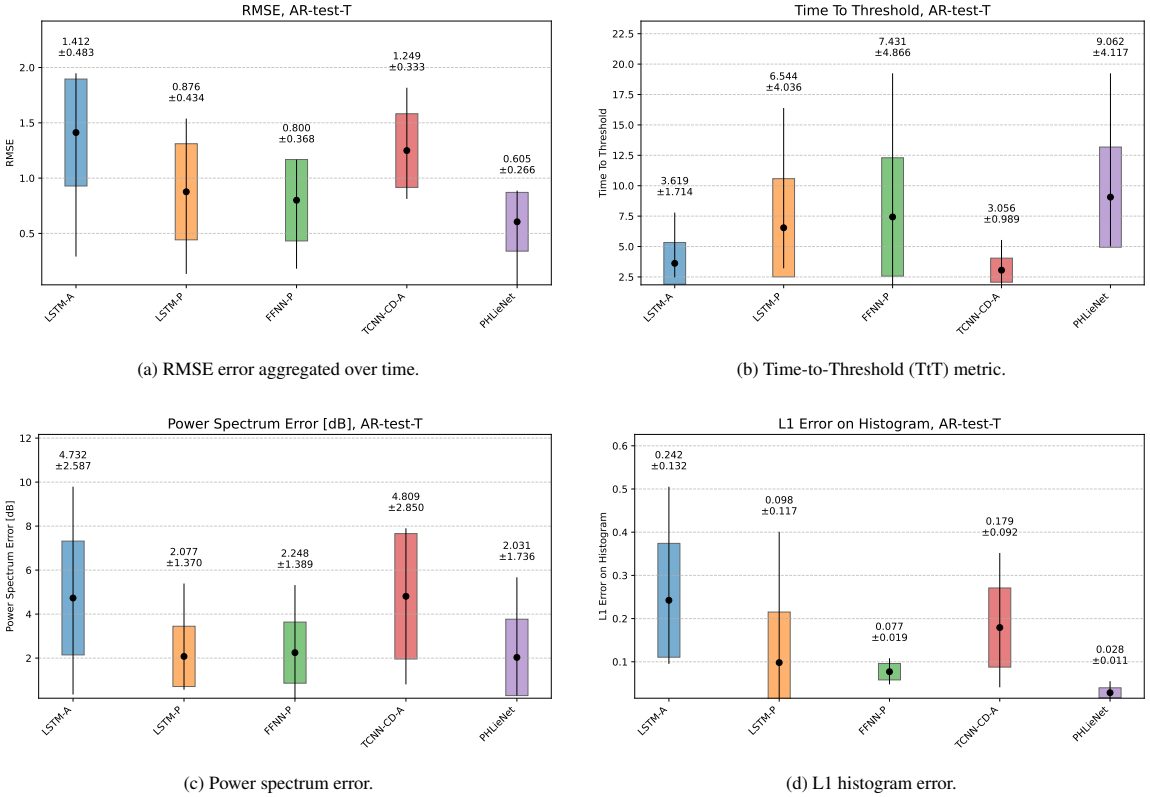


Figure 6: Normalized Root Mean Squared Error (NRMSE) evolution in time for extrapolation on seen parameters (AR-T).



(a) RMSE error aggregated over time.

(b) Time-to-Threshold (TtT) metric.

(c) Power spectrum error.

(d) L1 histogram error.

Figure 7: Model performance on the Van der Pol oscillator dynamics for extrapolation in time on seen parameters (AR-T). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

conditions, and a parameter extrapolation set with unseen μ values $P_{\text{test}}^{\text{AR-P}} = \{1.5, 2.5, \ldots, 8.5\}$. This setup allows us to assess both temporal generalization and robustness to unseen dynamical regimes.

In Figures 6 and 7, we benchmark the performance of the networks summarized in Table 1 on the autoregressive

testing task with previously seen parameters (AR-T). FFNN-A is omitted because its predictions quickly diverge. PHLieNet demonstrates lower NRMSE and RMSE errors in Figures 6 and 7a, as well as a marginally higher time-to-threshold in Figure 7b, indicating its superior short-term forecasting performance. Furthermore, the proposed PHLieNet shows excellent performance with respect to the power spectrum error in Figure 7c and a significantly lower histogram L1 norm error in Figure 7d, underscoring its ability to better capture the long-term statistics of the attractors. In general, we observe that the parameter-agnostic models of Table 1 exhibit larger errors and a faster divergence in their NRMSE, resulting in a worse short-term forecasting performance. Although augmenting LSTMs and FFNNs with the parameter in the state reduces errors somewhat, PHLieNet consistently outperforms these alternatives across all metrics, indicating superior performance.

In Figures 8 and 9, the forecasting performance of the networks summarized in Table 1 is benchmarked in the challenging task of autoregressive testing on unseen parameters (AR-P), with the key detail that these parameters were not used during training.

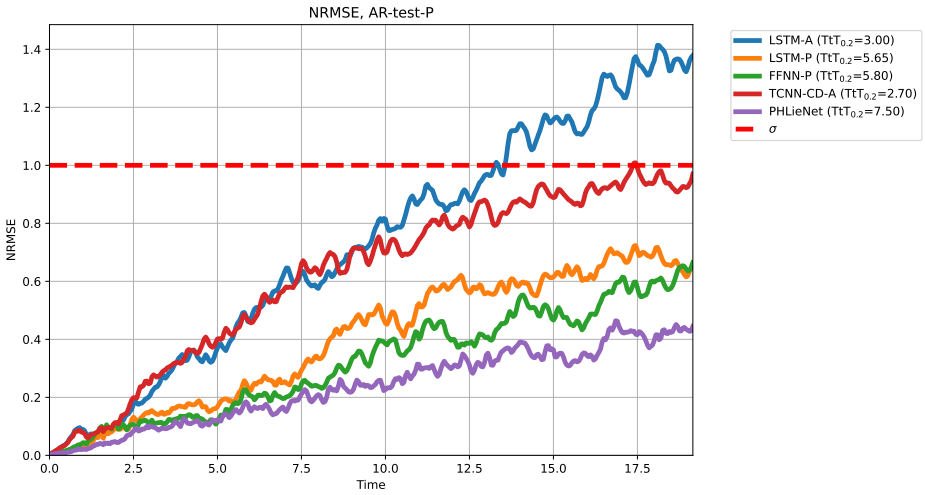
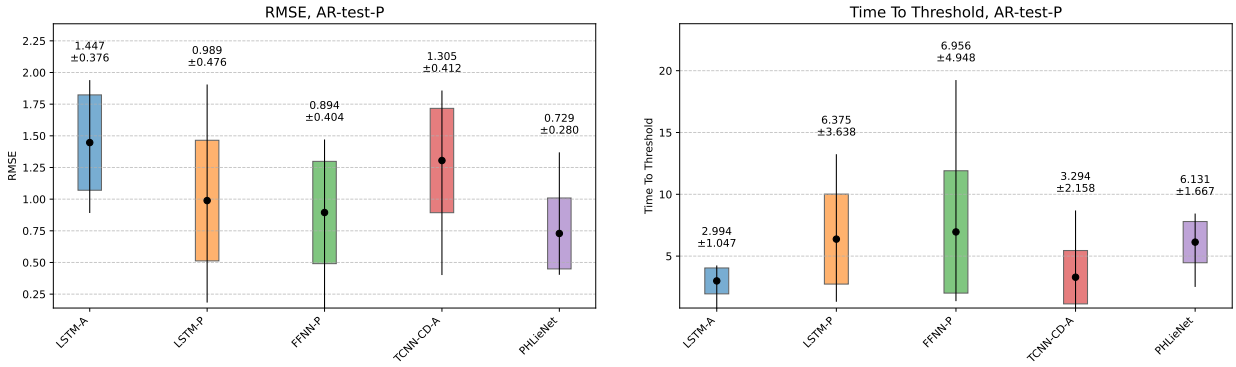


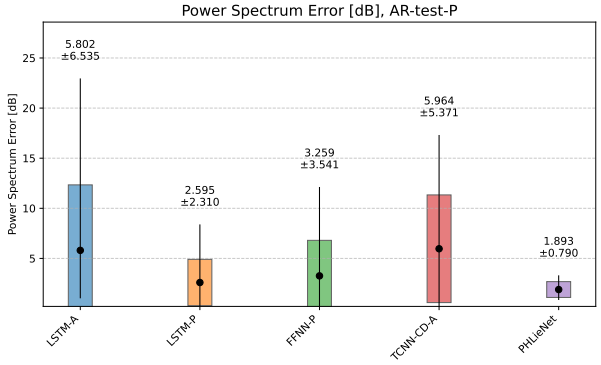
Figure 8: Normalized Root Mean Squared Error (NRMSE) evolution in time for extrapolation on unseen parameters (AR-P).

Similarly to the AR-T testing case, the proposed PHLieNet stands out by achieving low power spectrum error, low L1 histogram error, low RMSE, and a high time-to-threshold, demonstrating its ability to accurately extrapolate to unseen parameters by interpolating in the weight space of the target networks. Compared to the benchmarking models of Table 1, PHLieNet consistently yields lower NRMSE and RMSE errors, highlighting its superior short-term forecasting capabilities. Furthermore, it exhibits lower power spectrum and L1 histogram errors, further underscoring its effectiveness in capturing the long-term statistical properties of dynamics. As expected, parameter-informed models outperform agnostic ones, achieving lower errors and better overall performance.

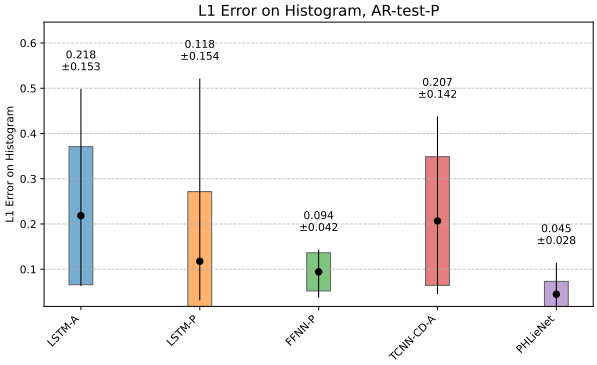


(a) RMSE error aggregated over time.

(b) Time-to-Threshold (TtT) metric.



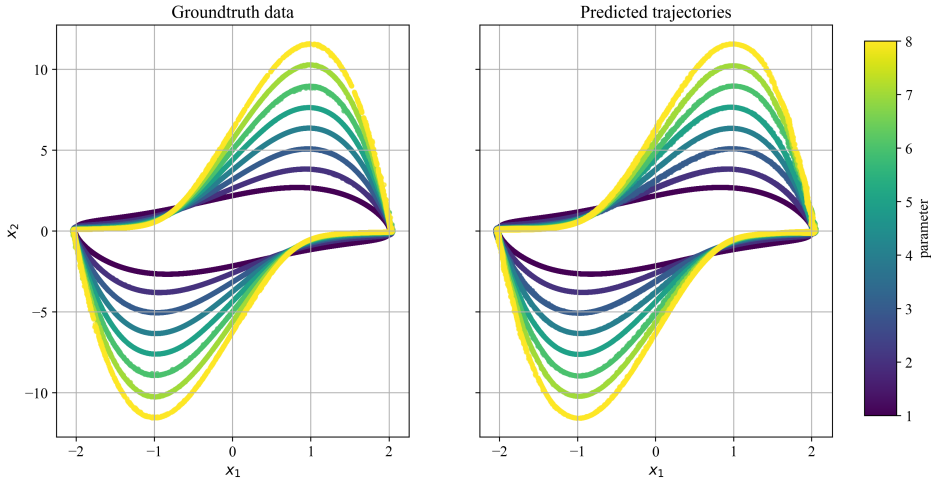
(c) Power spectrum error.



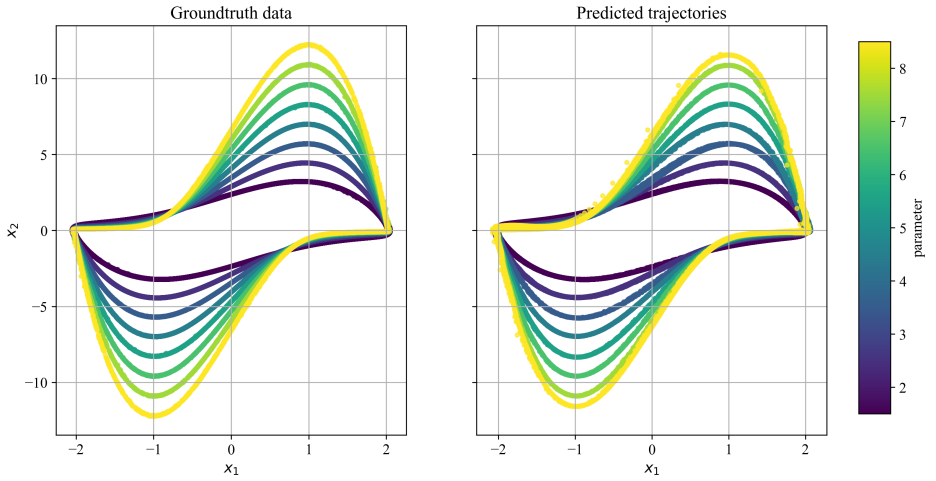
(d) L1 histogram error.

Figure 9: Model performance on the Van der Pol oscillator dynamics for extrapolation on unseen parameters (AR-P). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

The performance of the implemented PHLieNet is further demonstrated in Figure 10, where we plot ground truth data along with predicted trajectories in different parameter values for both testing autoregressive tasks (AR-T and AR-P). We observe that PHLieNet qualitatively captures the attractors over a broad range of parameter values, demonstrating its ability to reproduce the system dynamics in diverse dynamic regimes.



(a) Extrapolation in time testing (AR-T)



(b) Generalization on unseen parameters testing (AR-P)

Figure 10: Ground truth data and predicted trajectories from PHLieNet for the Van der Pol oscillator across different parameters.

4.2. The Lorenz 3D System

The Lorenz system, introduced by Edward Lorenz in 1963 [97], is a classical three-dimensional nonlinear dynamical system that exhibits deterministic chaos. Originally derived as a simplified model for atmospheric convection, it has become a cornerstone in the study of chaotic systems and strange attractors. The Lorenz system is defined by the following set of coupled differential equations:

$$\dot{x_1} = \sigma(x_2 - x_1), \tag{29}$$

$$\dot{x_2} = x_1(\rho - x_3) - x_2, \tag{30}$$

$$\dot{x_3} = x_1 x_2 - \beta x_3, \tag{31}$$

where $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$ is the system state, and $\sigma, \beta, \rho \in \mathbb{R}$ are scalar parameters that govern the dynamics. In our experiments, we fix $\sigma = 10$ and $\beta = \frac{8}{3}$, and vary the parameter ρ.

We use a fourth-order Runge–Kutta integrator (`RK45`) with a solver time step of $\delta t = 0.001$, and sample observations every $\Delta t = 0.01$ time units. The parameter set of the training and validation data is $P_{\text{train}} = \{28, 36, 44, 52, 60, 68\}$. For both training and validation data, we simulate $N_{ics}^{\text{train}} = N_{ics}^{\text{val}} = 10$ initial conditions for each parameter up to 50 time units leading to trajectories with $N_T^{\text{train}} = N_T^{\text{val}} = 5\text{K}$ timesteps. For each trajectory, the first 100 time units are

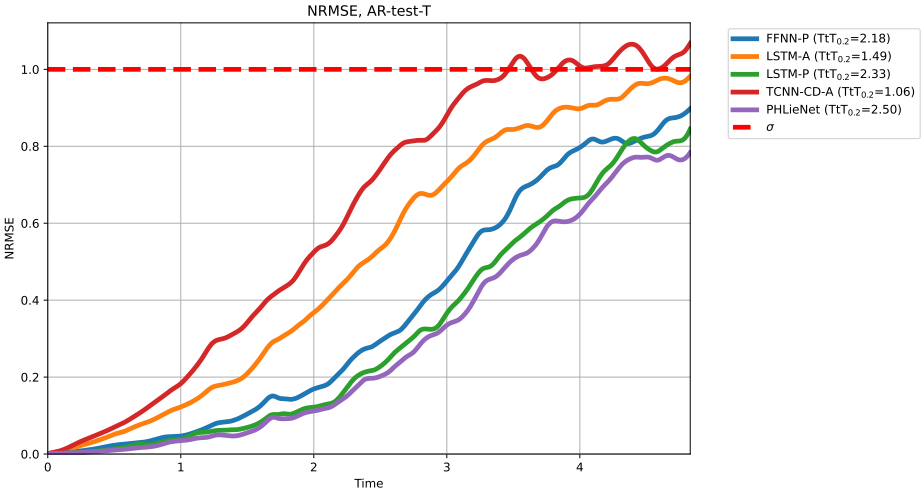
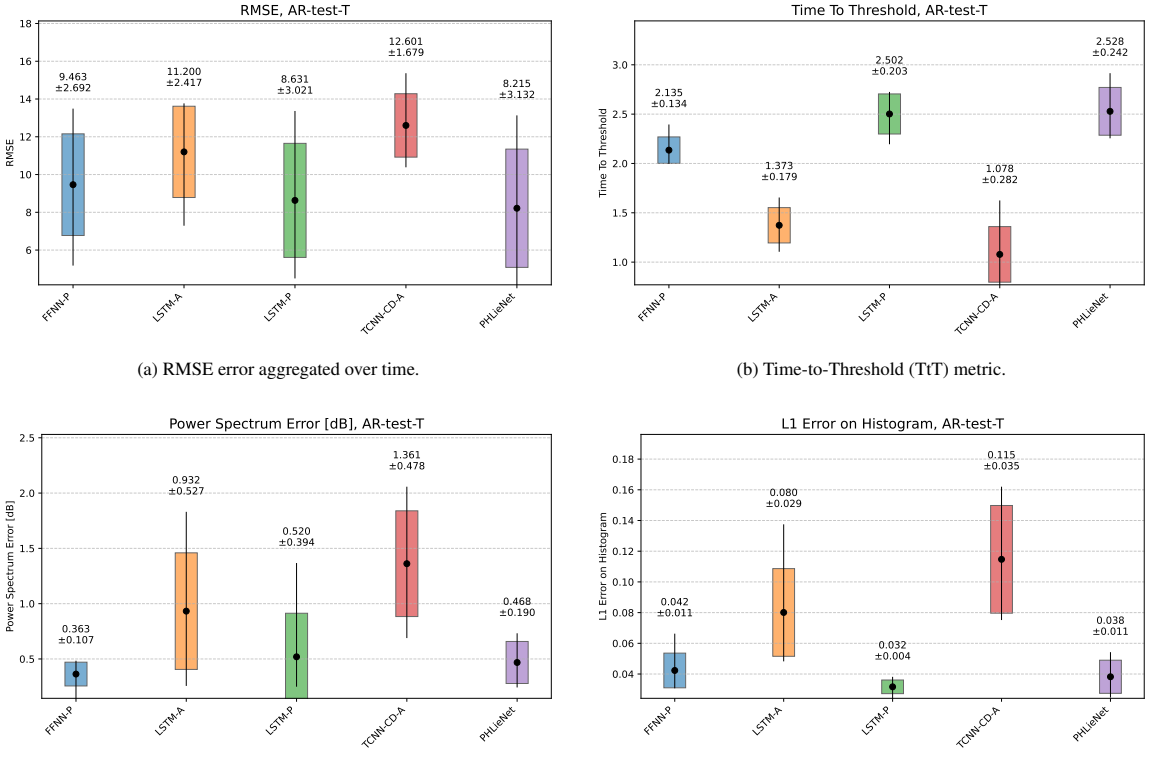


Figure 11: Root Mean Squared Error (RMSE) evolution in time for extrapolation on seen parameters (AR-T).

discarded to avoid initial transient effects. The initial conditions are sampled uniformly from the cube $[-5, 5]^3 \subset \mathbb{R}^3$. Noise during training is set to $\sigma_{\text{noise}} = 10\%$ For the autoregressive testing datasets we simulate $N_{ics}^{\text{test}} = 20$ initial conditions for 5 time units ($N_T^{\text{test}} = 500$). For parameter extrapolation, we use unseen values $P_{\text{test}}^{\text{AR-P}} = \{32, 40, 48, 56, 64\}$.



(a) RMSE error aggregated over time.

(b) Time-to-Threshold (TtT) metric.

(c) Power spectrum error.

(d) L1 histogram error.

Figure 12: Model performance on the Lorenz dynamics for extrapolation in time on seen parameters (AR-T). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

In Figures 11 and 12, we benchmark the performance of the inference models summarized in Table 1 in the autoregressive testing task with seen parameters (AR-T). The FFNN-A variant is omitted here because its predictions diverge quickly. In the NRMSE evolution plot in Figure 11, we observe that PHLieNet achieves slightly lower errors than LSTM-P, with a marginally higher $\text{TtT}_{0.2}$ in Figure 12b. Additionally, the parameter-agnostic models show a more rapid increase in the reconstruction error, indicating poorer performance in capturing the involved dynamics. Similarly to the van der Pol case study, the proposed PHLieNet exhibits slightly lower RMSE and higher TtT compared to LSTM-P, as illustrated in Figures 12a and 12b. Finally, in Figures 12c and 12d, PHLieNet and LSTM-P both display low power spectrum and L1 histogram errors, indicating their ability to capture the state statistics of the dynamics.

Next, we validate the performance of the networks of Table 1 in the autoregressive testing task on unseen parameters (AR-P). In the evolution of NRMSE in Figure 13, we observe that PHLieNet achieves slightly lower errors than LSTM-P, with a marginally higher $\text{TtT}_{0.2}$, similarly to the time extrapolation case in Figure 11. Parameter-agnostic models, in contrast, exhibit a more rapid increase in error. Furthermore, PHLieNet consistently achieves slightly lower RMSE and higher TtT compared to LSTM-P, as illustrated in Figure 14a and Figure 14b Finally, in Figure 14c and Figure 14d, both PHLieNet and LSTM-P maintain a low power spectrum and L1 histogram errors, indicating that they effectively capture the statistical properties of the system dynamics.

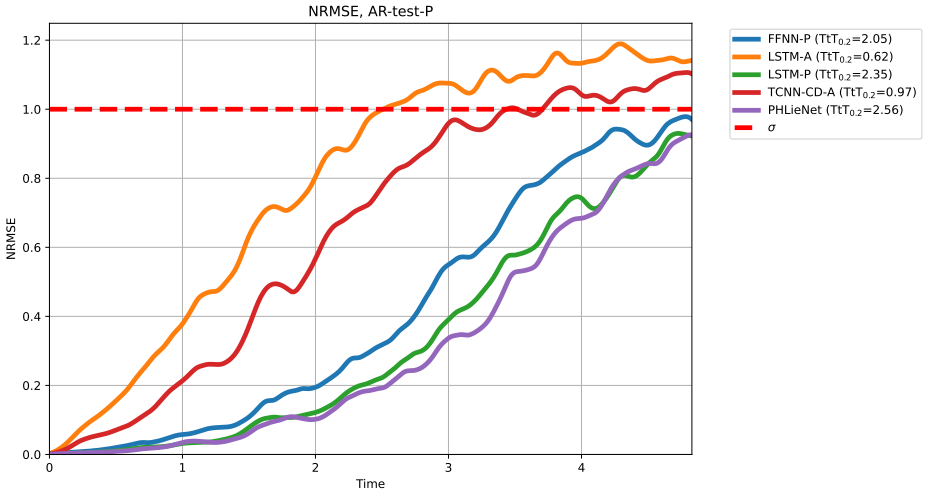
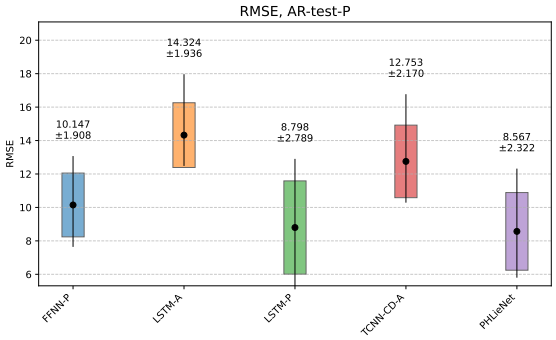
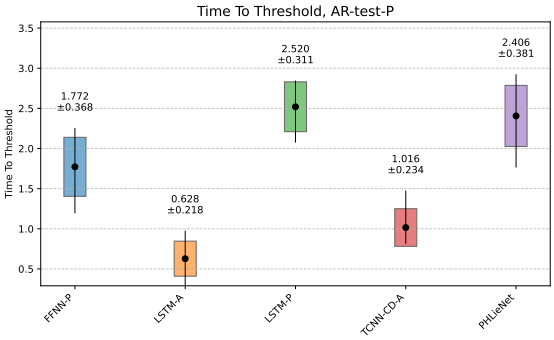


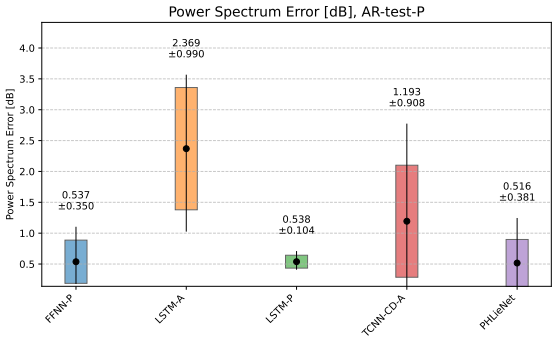
Figure 13: Root Mean Squared Error (RMSE) evolution in time for extrapolation on unseen parameters (AR-P).



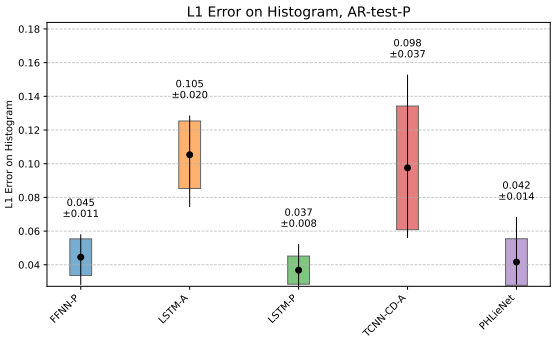
(a) RMSE error aggregated over time.



(b) Time-to-Threshold (TtT) metric.



(c) Power spectrum error.



(d) L1 histogram error.

Figure 14: Model performance on the Lorenz dynamics for extrapolation on unseen parameters (AR-P). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

In Figure 15, we visualize the ground truth data attractors of the Lorenz system and compare them with those formed by the predicted trajectories. In Figure 16, we do the same for the case of generalization to unseen parameters in the training dataset. These results visually reinforce the quantitative findings discussed earlier, demonstrating that PHLieNet successfully reconstructs the dynamics of the attractors across different dynamical regimes, including those with unseen parameters during training. Despite the slight variations in the shape and characteristics of the attractors caused by different parameterizations, the network accurately captures and reproduces the underlying dynamics in each case, highlighting the crucial role that the parameter plays in shaping the attractor.

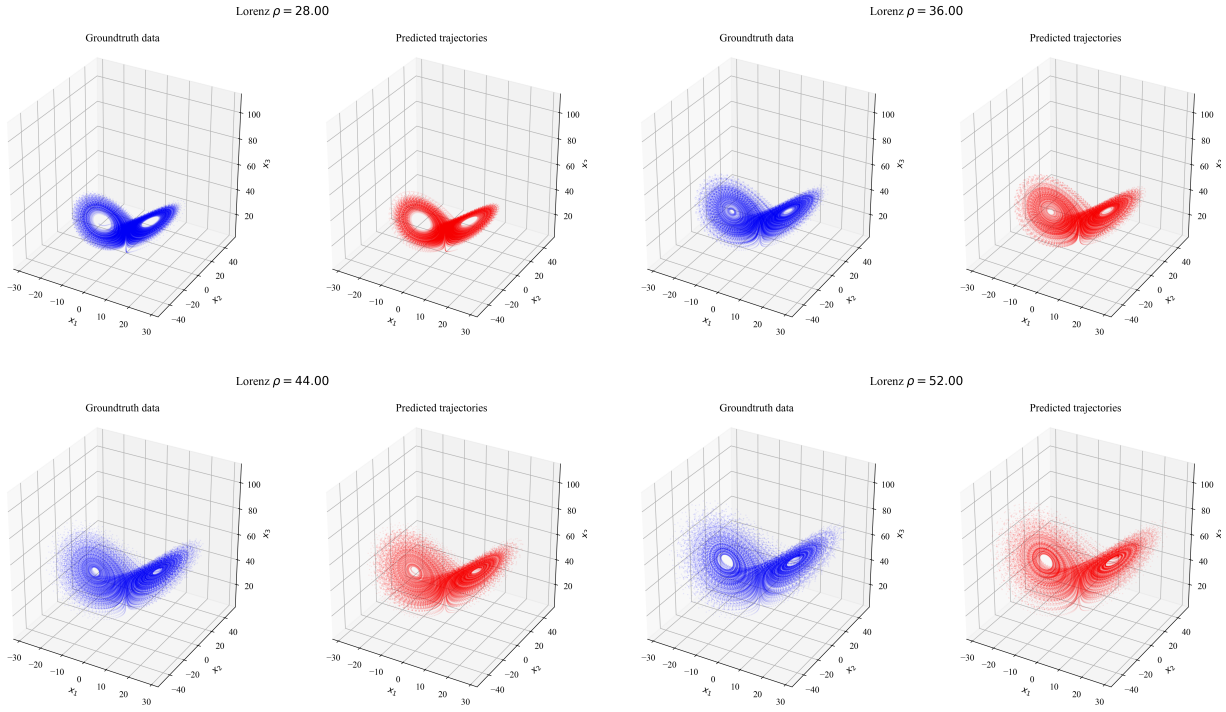


Figure 15: Extrapolation in time: Comparison of ground truth and predicted attractors across different parameter regimes.

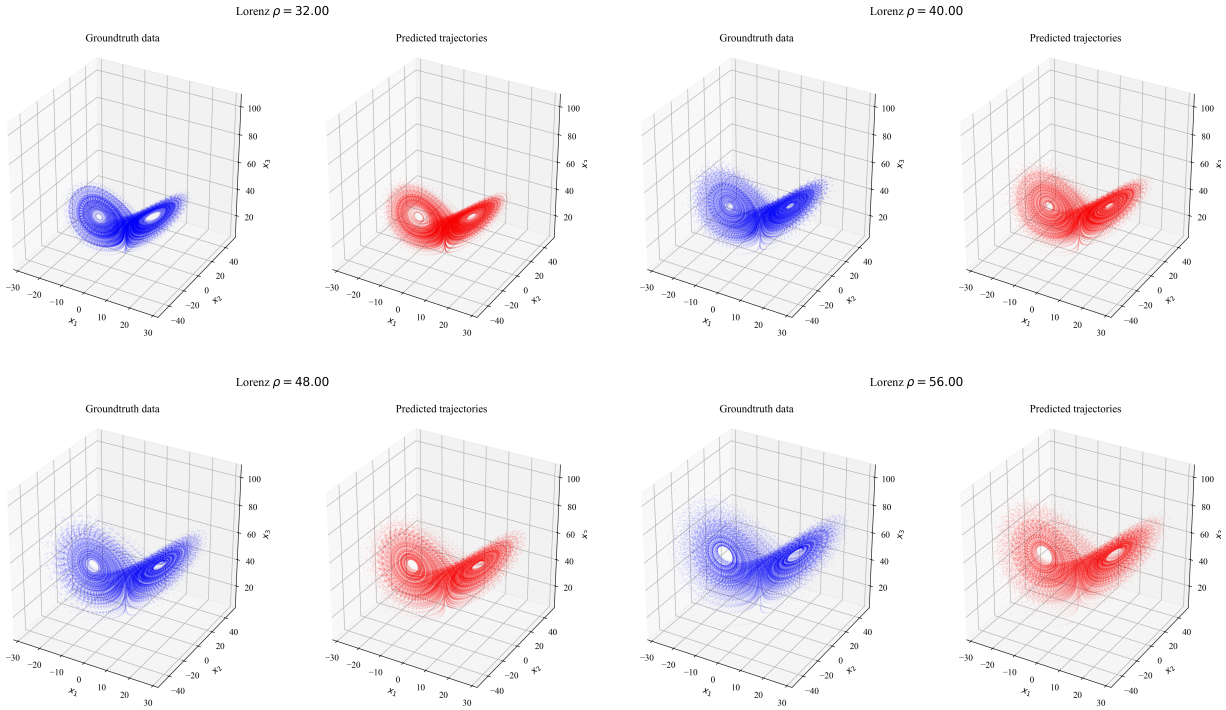


Figure 16: Generalization on unseen parameters: Comparison of ground truth and predicted attractors across different parameter regimes.

4.3. The Rössler System

The Rössler system, introduced by Otto Rössler in 1976 [98], is a set of three coupled nonlinear ordinary differential equations (ODEs) that exhibit chaotic behavior. Rössler initially developed this system as a simplified model to explore chaos in continuous dynamical systems. Its simplicity, both in terms of form and computational requirements, has made it a widely studied example in the field of chaos theory.

The system is defined by the following set of ODEs:

$$\dot{x_1} = -x_2 - x_3, \tag{32}$$

$$\dot{x_2} = x_1 + ax_2, \tag{33}$$

$$\dot{x_3} = b + x_3(x_1 - c), \tag{34}$$

where $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$ is the state and a, b, c are scalar parameters controlling the dynamics. The parameters a, b, and c in the Rössler system play distinct roles in shaping its dynamics: a controls the linear damping in the y-equation, b introduces a constant drift in the z-equation, and c modulates the nonlinearity in the z-equation through coupling with x. As these parameters are varied, the system transitions from simple periodic oscillations to chaotic behavior, which is often visualized in the form of attractors. For specific parameter values, the system generates a fractal attractor known as the Rössler attractor, one of the most iconic examples of deterministic chaos. Despite its simplicity, the Rössler system exhibits rich dynamical behavior, including bifurcations, periodic orbits, and chaotic attractors, depending on the values of the chosen parameters. Rössler's work has been extended to various fields such as chemical reactions, biological systems, and electronics, where chaos plays a critical role, making the system a valuable testbed for both theoretical studies and practical applications [96].

In our experiments, we fix $a = b = 0.1$ and vary c to explore different regimes, from periodic motion to deterministic chaos. We simulate trajectories using a fourth-order Runge–Kutta integrator (RK4) with a solver time step $\delta t = 0.001$, and sample the solution every $\Delta t = 0.1$ time units. To generate training data, we simulate $N_{ics}^{\text{train}} = N_{ics}^{\text{val}} = 10$ trajectories up to 100 time units (preceded by a transient period of 100 time units, which is discarded), which means $N_T^{\text{train}} = N_T^{\text{val}} = 1\text{K}$ timesteps. The initial conditions are sampled uniformly from the cube $[-1, 1]^3 \subset \mathbb{R}^3$. The parameter set for the training and validation data is $P_{\text{train}} = \{4, 6, 8, 10, 12, 14, 16, 18\}$. For the test datasets, we generate $N_{ics}^{\text{test}} = 100$ trajectories per parameter value, each with the same duration of 100 time units ($N_T^{\text{test}} = 1\text{K}$). For parameter extrapolation, we use unseen values $P_{\text{test}}^{\text{AR-P}} = \{5, 7, 9, 11, 13, 15, 17\}$ to evaluate generalization across dynamical regimes.

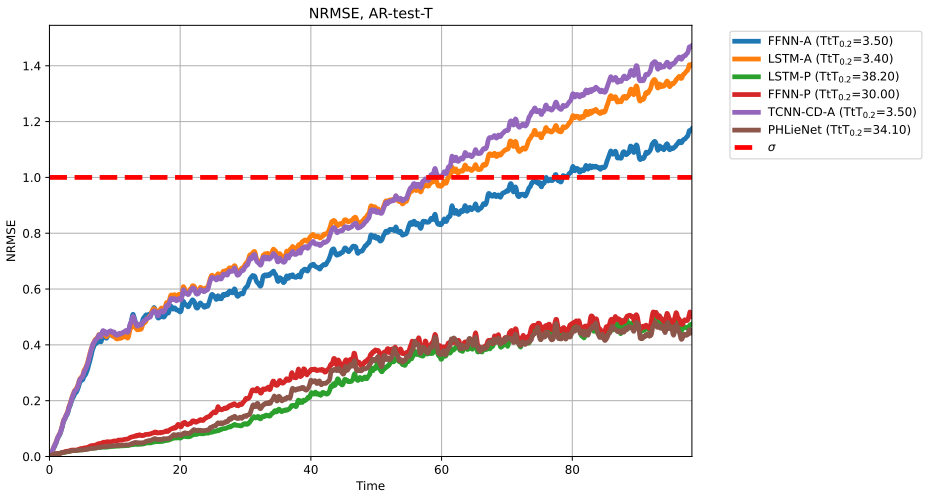
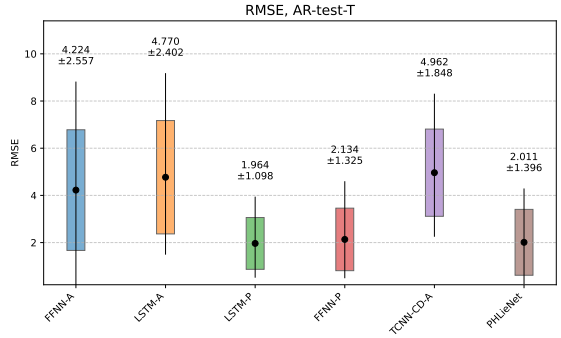


Figure 17: Root Mean Squared Error (RMSE) evolution in time for extrapolation on seen parameters (AR-T).

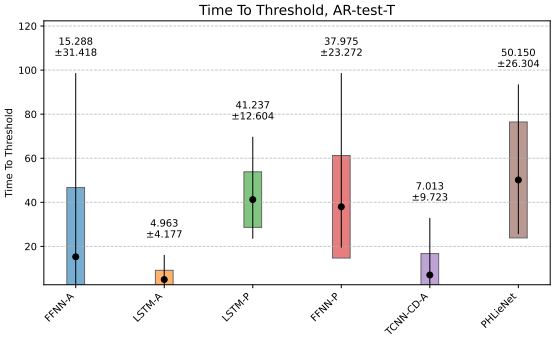
In Figure 18, we first benchmark the performance of the networks in Table 1 in the autoregressive testing task for time extrapolation on trajectories from seen parameters (AR-T). The proposed PHLieNet consistently delivers low

NRMSE errors, while it exhibits slightly higher errors than LSTM-P, with a slightly lower $TtT_{0.2}$ on the NRMSE. Both these parameter-informed approaches achieve lower errors than FFNN-P, and, as expected, all parameter-informed models exhibit a slower increase in error compared to the agnostic ones. This trend is also apparent in the cumulative RMSE plot in Figure 18a.

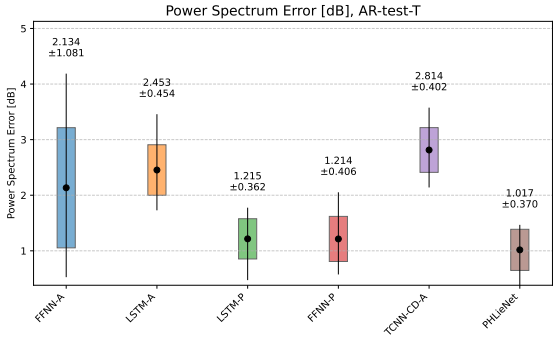
Interestingly, in the average TtT per trajectory shown in Figure 18b PHLieNet actually achieves a higher TtT than LSTM-P. This suggests that the slightly worse performance in the cumulative NRMSE is mainly due to a few outlier trajectories. On average per trajectory, PHLieNet is more accurate in the short term compared to all other methods, but these outliers contribute to the cumulative NRMSE observed in Figure 17. Finally, in Figure 18c and Figure 18d, PHLieNet achieves the lowest power spectrum and L1 histogram errors, indicating that it effectively captures the statistical properties of the dynamics. LSTM-P is on par with PHLieNet in this regard.



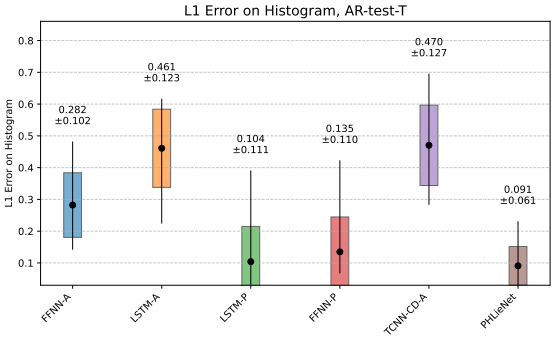
(a) RMSE error aggregated over time.



(b) Time-to-Threshold (TtT) metric.



(c) Power spectrum error.



(d) L1 histogram error.

Figure 18: Model performance on the Rössler dynamics for extrapolation in time on seen parameters (AR-T). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

Next, we present the results for the parametric extrapolation task (AR-P) in Figures 19 and 20. From the NRMSE evolution in Figure 19 and the cumulative RMSE in Figure 20a, we observe that PHLieNet performs comparably to LSTM-P, although LSTM-P exhibits slightly lower errors on average. Both models significantly outperform all other methods. In terms of short-term prediction, as reflected by the mean $TtT_{0.2}$ per trajectory in Figure 20b, PHLieNet and LSTM-P achieve longer prediction horizons than the other models. Meanwhile, all agnostic models show very high RMSEs and very low $TtT_{0.2}$ values.

In Figure 20c, we note that PHLieNet performs on par with FFNN-P but does not reach the low power spectrum errors of LSTM-P. Overall, PHLieNet faced more challenges in extrapolating the dynamics of this system compared to the other cases considered in this study. Nonetheless, PHLieNet achieves the lowest L1 histogram error, as shown in Figure 20d. Once again, the superiority of parameter-informed models is evident, with all such models demonstrating lower errors than their agnostic counterparts.

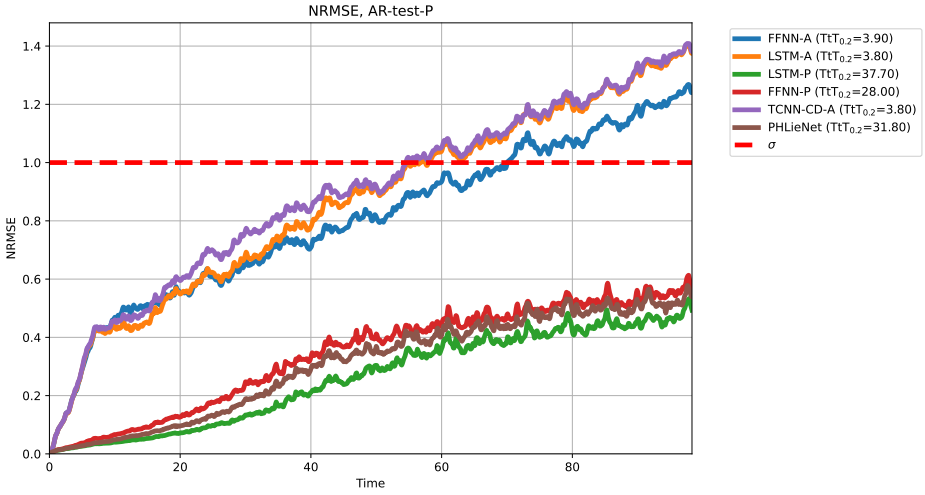
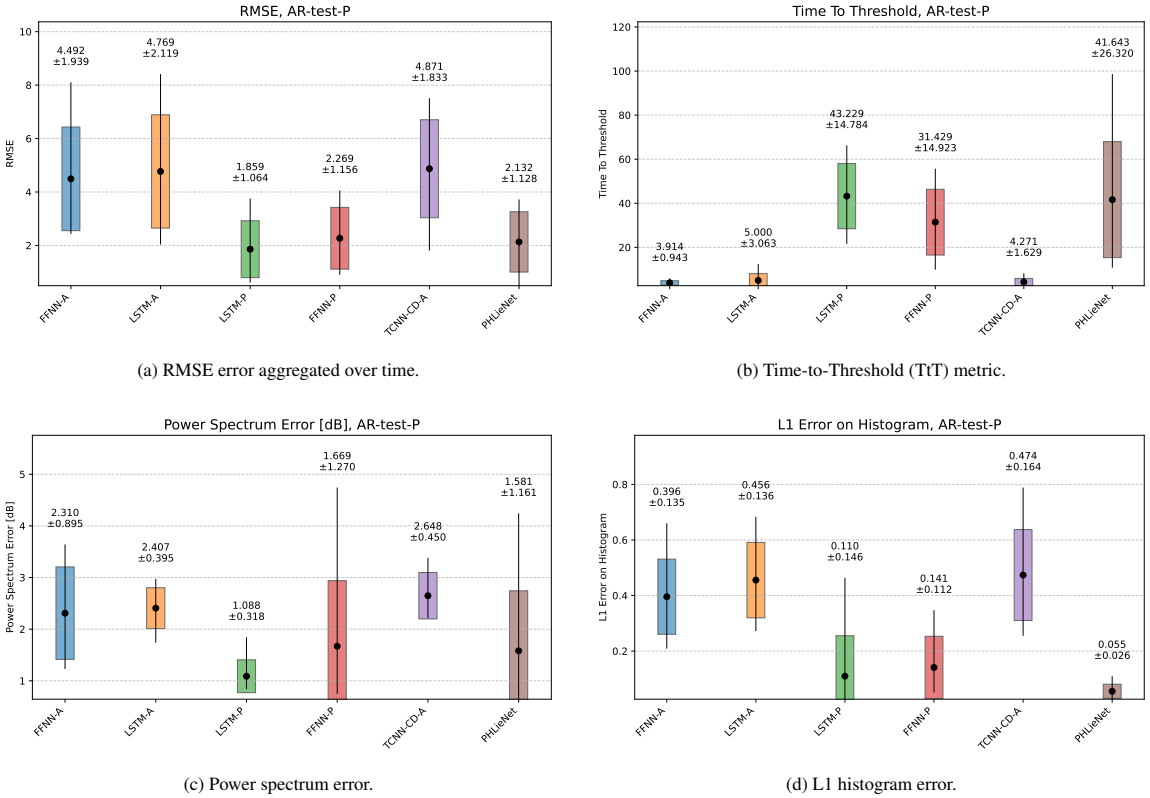


Figure 19: Root Mean Squared Error (RMSE) evolution in time for extrapolation on unseen parameters (AR-P).



(a) RMSE error aggregated over time.

(b) Time-to-Threshold (TtT) metric.

(c) Power spectrum error.

(d) L1 histogram error.

Figure 20: Model performance on the Rössler dynamics for extrapolation on unseen parameters (AR-P). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

In Figure 21, we visualize the ground truth data attractors of the Roessler system and compare them with those formed by the predicted trajectories across various parameter regimes. In Figure 22, we present similar comparisons for the case of generalization to unseen parameters that were not included in the training dataset. These visual

comparisons complement the quantitative results discussed earlier, providing compelling evidence that PHLieNet is capable of accurately reconstructing the dynamics of the Roessler system across a diverse range of parameter values. Although the shape and structure of the attractors vary significantly due to the influence of the system parameter, the network consistently reproduces the key dynamical features of each attractor. This highlights the network's ability to extrapolate and generalize, underscoring its predictive capabilities in complex systems where parametric variability strongly influences the underlying attractor dynamics.

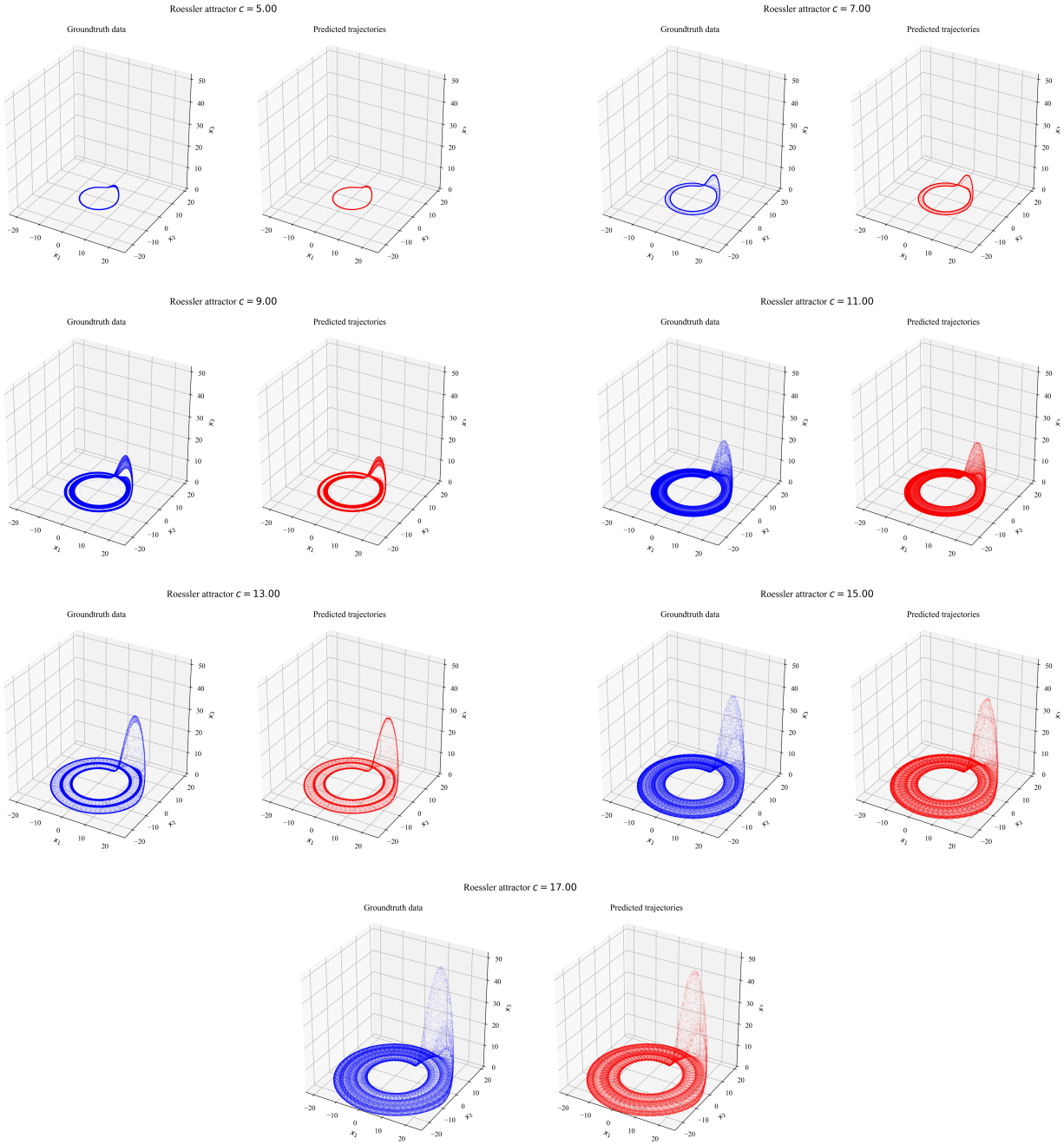


Figure 21: Extrapolation in time for the Roessler system: Comparison of ground truth and predicted attractors across different parameter regimes.

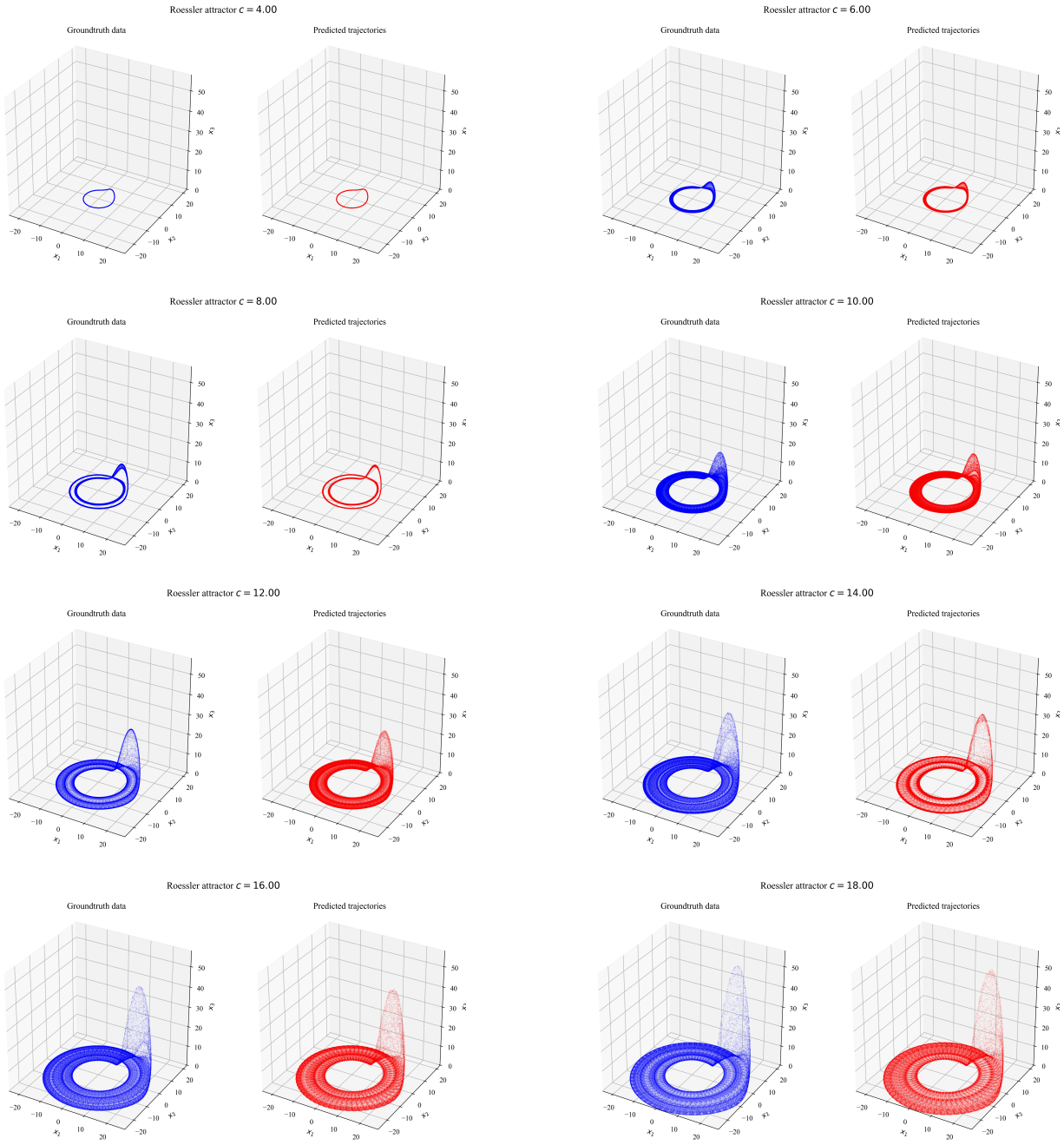


Figure 22: Generalization on unseen parameters for the Roessler system: Comparison of ground truth and predicted attractors.

4.4. Chua's circuit

Chua's circuit, first introduced by Leon O. Chua in 1983 [99], is a nonlinear electronic system that exhibits chaotic behavior through a simple configuration of linear elements (resistors, capacitors, and inductors) and a nonlinear component, the Chua diode, which introduces a piecewise-linear characteristic. The dynamics of the circuit are described by a system of three first-order ordinary differential equations governing the voltage and current, which give rise to a wide range of dynamical phenomena such as periodic orbits, bifurcations, and chaotic attractors.

The equations governing the dynamics of Chua's circuit are given by:

$$\dot{x_1} = a(x_2 - x_1 - h(x_1)), \tag{35}$$

$$\dot{x_2} = x_1 - x_2 + x_3, \tag{36}$$

$$\dot{x_3} = -bx_2, \tag{37}$$

Here, $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$ is the state that represents the normalized variables corresponding to the circuit's voltages and currents, while a and b are parameters related to the circuit's components. The nonlinearity in the system is introduced by the piecewise linear function $h(x_1)$, which models the behavior of the Chua diode.

$$h(x) = \mu_1 x + 0.5(\mu_0 - \mu_1)(|x + 1| - |x - 1|) \tag{38}$$

where μ_0 and μ_1 are parameters that control the slope of the diode's characteristic. This piecewise function contributes to the rich and diverse dynamical behavior of the system, including the emergence of chaotic attractors, bifurcations, and complex trajectories in phase space. In the following, we vary the parameter a as it crucially determines the balance between linear and nonlinear dynamics through its coupling of nonlinearity to the difference $x_1 - x_2$. Varying a allows us to explore a broad range of dynamic regimes, including stable fixed points, limit cycles, and chaotic behavior, providing a rich and informative spectrum of system responses. In contrast, parameters such as b (oscillation damping) and μ_0, μ_1 (nonlinearity shaping) have more specialized effects. We fix $b = 100/7 \approx 14.28$, $\mu_0 = -8/7 \approx -1.14$, and $\mu_1 = -5/7 \approx -0.71$, following previous studies [100] that extensively analyzed the dynamics of the double-scroll attractor.

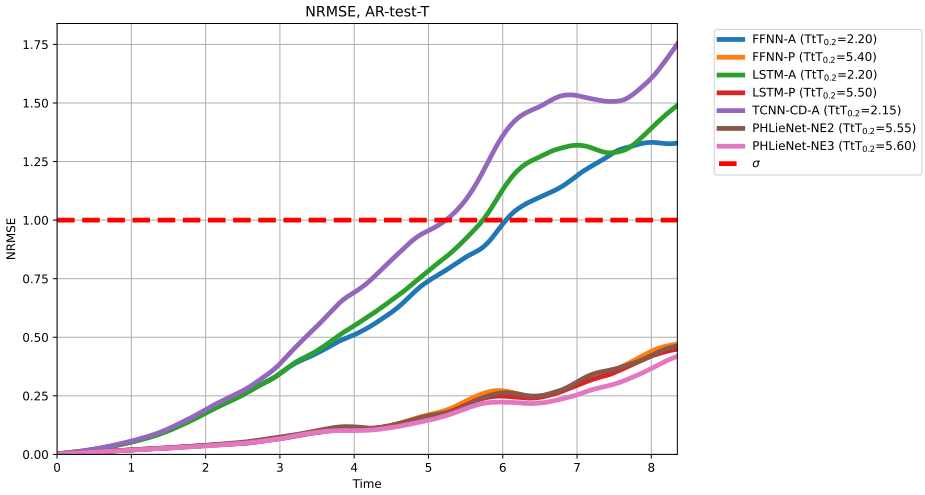
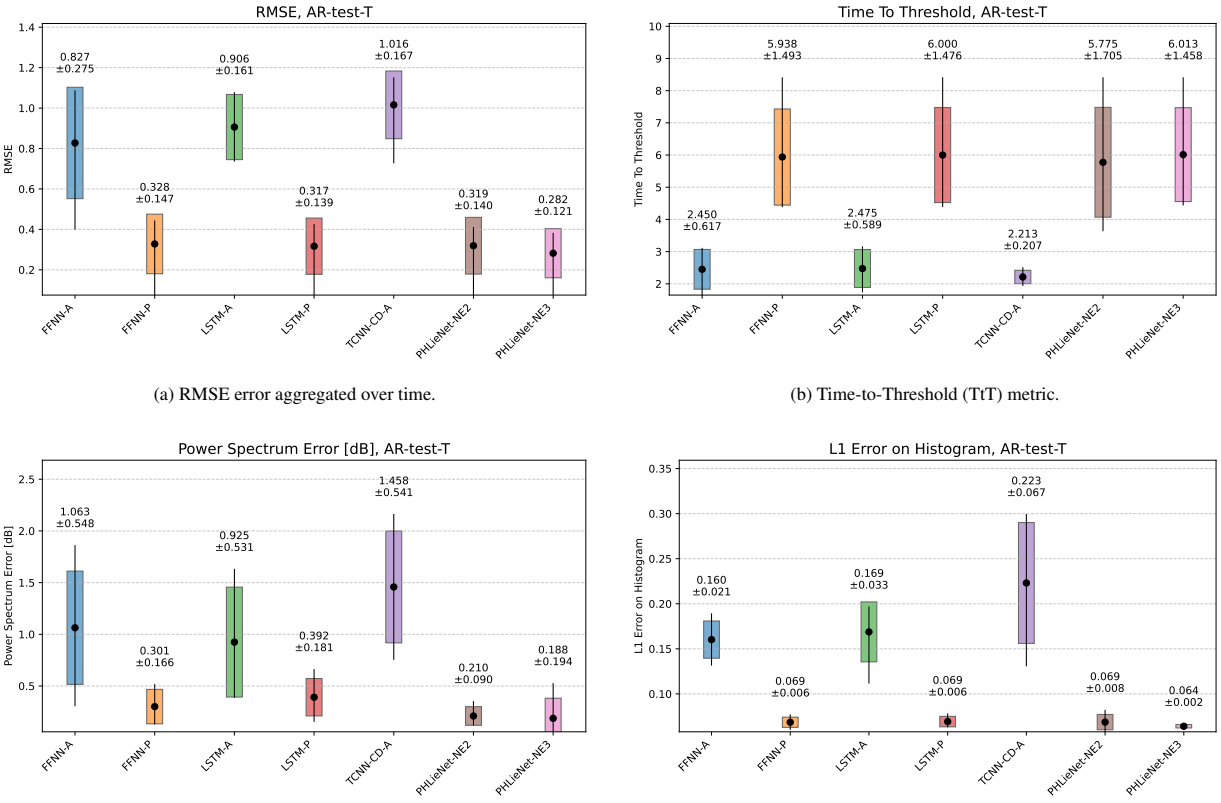


Figure 23: Root Mean Squared Error (RMSE) evolution in time for extrapolation on seen parameters (AR-T).

Trajectories of the state evolution are obtained by discretizing and simulating Equation (37) with $\delta t = 0.001$, using a fourth-order Runge-Kutta integrator. Trajectories are subsampled to $\Delta t = 0.05$ time units. The initial conditions $[x_0, y_0, z_0]^T$ are sampled from random uniform distributions: $x_1 \sim U[-0.5, 0.5]$, $x_2 \sim U[-0.5, 0.5]$, $x_3 \sim U[-0.1, 0.1]$. The first 100 time units are truncated to avoid initial transient effects. The train and validation data contain trajectories from the parametric set $P_{\text{train}} = \{8.5, 9, 9.5, 10\}$. In the training data, we simulate $N_{ics}^{\text{train}} = 10$ initial conditions for 50 time units corresponding to $N_T^{\text{train}} = 1\text{K}$ timesteps. A similar dataset is constructed for validation, although starting from different initial conditions. For testing we simulate $N_{ics}^{\text{train}} = 50$ initial conditions per parameter value for 10 time units corresponding to $N_T^{\text{train}} = 200$ timesteps. For the parameter extrapolation task we use unseen values $P_{\text{test}}^{\text{AR-P}} = \{8.75, 9.25, 9.75\}$.

The comparison metrics for the test time extrapolation task (AR-T) are presented in Figures 23 and 24. From the evolution of NRMSE in Figure 23 and the cumulative RMSE in Figure 24a, we observe that all parameter-informed models achieve lower errors than the parameter-agnostic ones. Among them, the PHLieNet variant with



(a) RMSE error aggregated over time.

(b) Time-to-Threshold (TtT) metric.

(c) Power spectrum error.

(d) L1 histogram error.

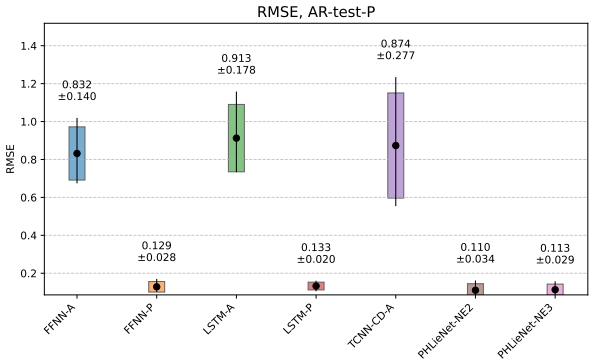
Figure 24: Model performance on the Chua dynamics for extrapolation in time on seen parameters (AR-T). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

three embeddings has the lowest error, although by a small margin. Similar results are observed in the TtT metric in Figure 24b, where the proposed PHLieNet network achieves the highest TtT, although the differences between all parameter-informed models are minor. Furthermore, as shown by the power spectrum error and the L1 error in the state histogram in Figures 24c and 24d respectively, PHLieNet consistently demonstrates lower errors, with the three-embedding variant outperforming the two-embedding variant in three out of four metrics. These results highlight that PHLieNet effectively captures the dynamics of the Chua circuit and accurately extrapolates in time.

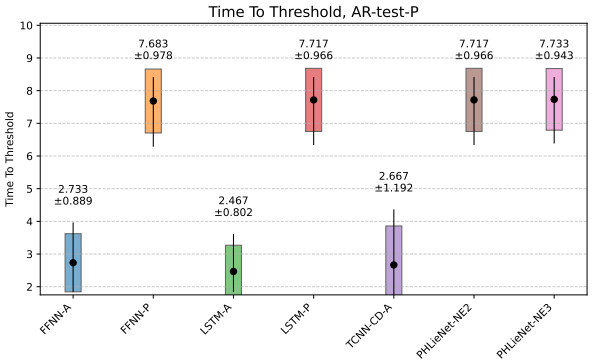
The improved performance of the three-embedding PHLieNet variant compared to its two-embedding counterpart highlights the importance of selecting an adequate number of anchor embeddings. In this experiment, each anchor embedding acts as a representative of a distinct dynamical regime in parameter space. Increasing the number of anchors from two to three enables the interpolation mechanism to capture the transitions more accurately between different behaviors of Chua's circuit. This added flexibility allows the hypernetwork to generate more expressive and well-adapted forecasting models, particularly in regimes where the dynamics change rapidly with respect to the parameter a. At the same time, the use of only three embeddings compared to four training parameter values ensures that the model must still interpolate rather than memorize, preserving generalization capabilities. These findings provide empirical support for the principle of architectural design discussed in Section 2.4, emphasizing the trade-off between expressiveness and interpolation capacity in the choice of the number of anchor embeddings.

Moving on to the task of parameter extrapolation, we plot the NRMSE error in Figure 25, where we observe a behavior similar to that seen in the time extrapolation case. Here too, the parameter-informed models exhibit lower RMSE errors (Figure 26a), as well as lower power spectrum errors (Figure 26c) and L1 histogram errors (Figure 26d). Among these, the PHLieNet variant with three embeddings achieves the best performance, albeit by a small mar-

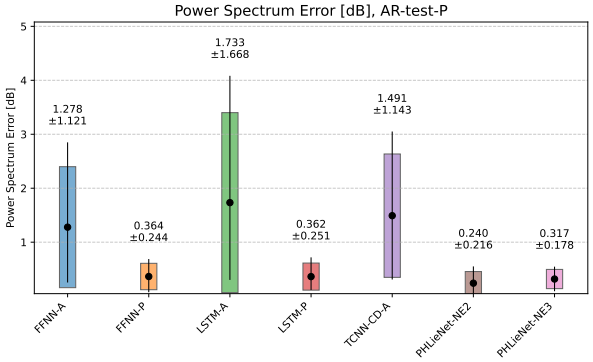
gin. These results indicate that PHLieNet is capable of efficiently extrapolating not only in time but also across the parametric space.



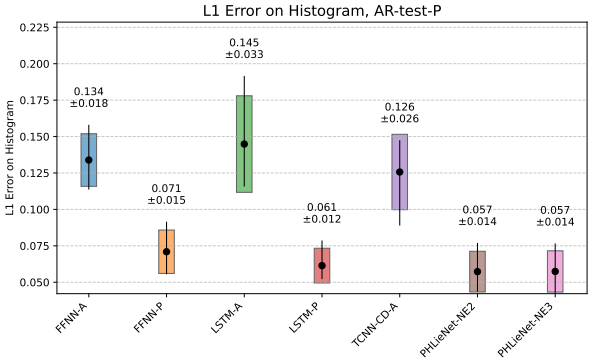
(a) RMSE error aggregated over time.



(b) Time-to-Threshold (TtT) metric.



(c) Power spectrum error.



(d) L1 histogram error.

Figure 26: Model performance on the Chua oscillator dynamics for extrapolation on unseen parameters (AR-P). (a) RMSE error aggregated over time. (b) Time-to-Threshold (TtT) metric. (c) Power spectrum error. (d) L1 histogram error.

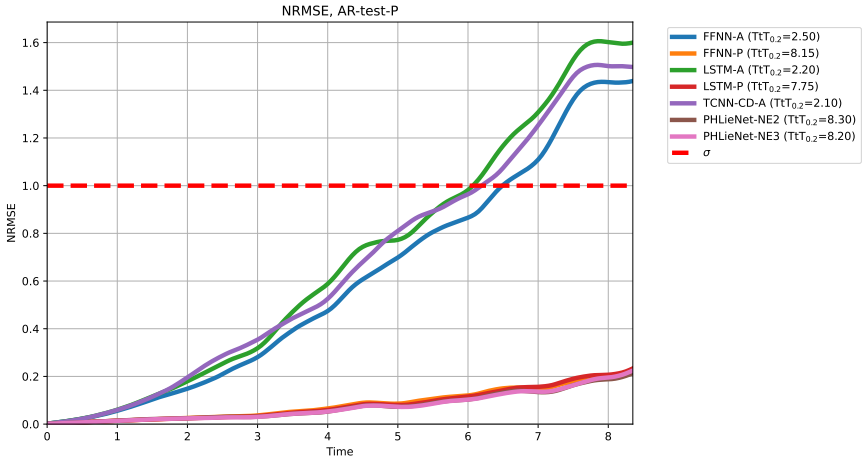


Figure 25: Root Mean Squared Error (RMSE) evolution in time for extrapolation on unseen parameters (AR-P).

In Figure 27, we present visual comparisons between the ground truth attractors of Chua's circuit and those reconstructed from the predicted trajectories for several parameter regimes. Similarly, in Figure 28, we evaluate the model's ability to generalize to attractors corresponding to previously unseen parameter values. These visual results complement our earlier quantitative analysis, clearly illustrating that PHLieNet is capable of capturing the intricate and parameter-dependent dynamics of Chua's circuit. Despite the substantial differences in attractor shapes driven by changes in system parameters, the network consistently reproduces the defining features of each attractor, highlighting the central role that the parameter plays in determining the system's dynamic behavior.

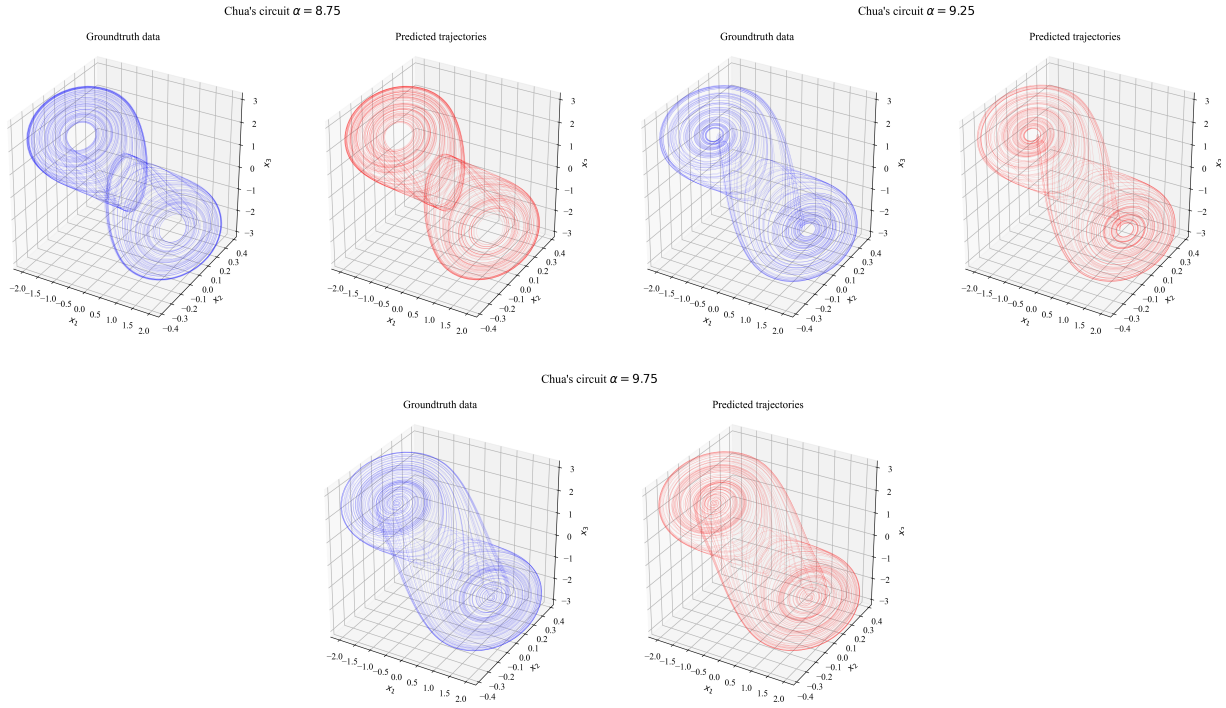


Figure 27: Extrapolation in time for Chua's circuit: Comparison of ground truth and predicted attractors across different parameter regimes.

5. Discussion

Modeling dynamical systems that exhibit differentiated responses to varying external stimuli or parameters is a central challenge with broad practical relevance. Previous approaches often fail to capture the full spectrum of behaviors within a unified framework. To address this, we propose PHLieNet - Parametric Hypernetwork for Learning Interpolated Networks, a novel architecture which a) learns a continuous embedding of the parametric space and b) uses hypernetworks to map this embedding to the parameters of a latent dynamics network. Unlike existing methods, PHLieNet does not impose theoretical constraints on the class of dynamics that can be represented. By adjusting the target network's complexity, the hypernetwork can learn to effectively interpolate within the weight space, enabling the generation of diverse dynamic behaviors. In our implementation, we adopt a causal dilated TCNN as the target network.

We demonstrate the efficiency of the proposed framework on four complex parametric dynamical systems that exhibit nonlinear dynamics: the Van der Pol oscillator, the Lorenz system, the Chua circuit, and the Rössler system. We benchmark our approach against other state-of-the-art methods, including parameter-agnostic models based on feedforward neural networks, LSTM RNNs, and causal dilated TCNNs, as well as their adaptations that augment the hidden state with the parameter. Evaluation metrics span both short-term prediction performance, such as time-to-threshold and RMSE evolution, and the ability to capture the attractor's statistics and reproduce the long-term climate, including power spectrum error and L1 norm error of the histogram. Across all benchmarks, our approach

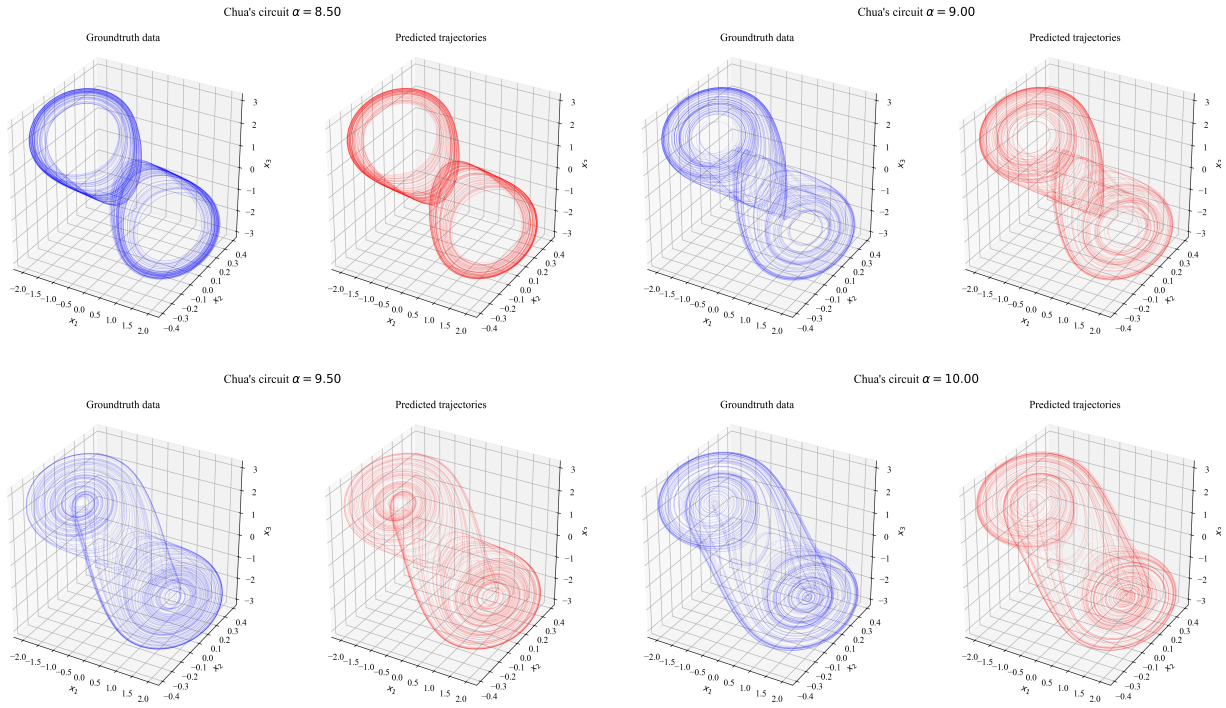


Figure 28: Generalization on unseen parameters for Chua's circuit: Comparison of ground truth and predicted attractors.

consistently outperforms or matches the performance of other models while also exhibiting qualitatively distinct properties compared to existing methods.

While the use of hypernetworks for modeling complex parametric dynamics remains relatively nascent, the results presented in this work with PHLieNets demonstrate strong potential. Future research could extend the proposed method to high-dimensional parametric spaces, where multiple interacting parameters govern the system's behavior. Such an extension is conceptually straightforward within our framework and could be achieved through barycentric interpolation in the parameter space, enabling even more flexible and expressive modeling capabilities.

Another promising direction is to eliminate the interpolation step altogether and directly learn the mapping from parameters to embedding. In an early version of our work, we explored this using a simple linear network. We obtained promising preliminary results, although for a more limited range of dynamics compared to this work. A similar linear mapping approach was also used in [92], which also demonstrated limited generalization in various dynamical regimes.

Although PHLieNet demonstrates promising results for modeling parametric dynamics, we recognize a potential bottleneck in our framework. PHLieNet relies on neural networks with smooth activation functions, enabling it to interpolate over parametric dynamics and weight space, but this relies on the assumption that the dynamics themselves vary smoothly with the parameters. In scenarios involving abrupt transitions, such as bifurcations, shocks, or regime shifts, our framework would require adaptations to capture these discontinuities accurately. Additionally, for such scenarios, a denser sampling of the parametric space would be necessary to adequately represent these rapid transitions.

In this work, we used the same set of parameters for training and validation. However, we also tested the ability of our approach to generalize to unseen parametric dynamics. Generalization to unseen parameters could be further improved by evaluating whether using a distinct parameter set for validation would enhance generalization. A comprehensive analysis of how to select the validation set is left for future work.

The proposed framework poses no limitations on the structure or selection of the target network. The target network in our framework could be a Neural ODE, a neural operator, a PINN, or any other dynamics propagator. Improving the proposed approach and benchmarking these different types of target network in PHLieNet while gaining

a deeper understanding of their differences remain an open area of investigation. It remains unclear which parametric weight space is more amenable to interpolation, as these methods exhibit distinct characteristics. In our experiments, we observed that interpolating in the parametric space of temporal CNNs was easier than with RNNs such as LSTMs.

Another interesting avenue for future research is to apply PHLieNet to online adaptive modeling of dynamical systems. Rather than training the model offline, this approach would account for parameters that change in real-time. Such a framework would require online anomaly detection, the ability to detect when the dynamic regime has shifted, and automatic recalibration of the model. Achieving this would likely involve combining PHLieNet with a state estimation framework.

PHLieNet marks a departure from the reductionist practice of training distinct models for each parameter setting to a more holistic and unified approach that learns the interplay between parameter and system dynamics. This framework not only enables interpolation across a wide range of parameter regimes but also opens the door to adaptive and generalizable modeling of complex dynamical systems. We believe that such a perspective, one that embraces the structure of the full parametric space, can inspire further research toward more flexible, robust, and insightful models of the dynamic phenomena that govern real-world systems.

Appendix A. Hypernetwork Architecture Details

We now describe the functional form of the hypernetwork used to generate the parameters of the target forecasting network, as defined in Equation (18).

Let the target network parameters be denoted by w_f, which includes all weights and biases. By flattening and concatenating these, we obtain a vector of total length $|w_f|$. The goal of the hypernetwork is to generate w_f as a function of the system parameter vector $\mathbf{p} \in \mathbb{R}^{D_p}$.

Appendix A.1. Embedding Interpolation Mechanism

To enable smooth generalization across parameter space, we employ an interpolation mechanism over a fixed number of learned embedding vectors. Specifically, we define a set of $N_{\mathbf{e}}$ learnable anchor embeddings $\{\mathbf{e}^{(i)}\}_{i=1}^{N_{\mathbf{e}}} \subset \mathbb{R}^{D_{\mathbf{e}}}$, implemented as a standard embedding layer.

For a given input parameter $p \in [0, 1]$, we identify two anchor indices, lower and upper, and compute a scalar interpolation weight $\alpha \in [0, 1]$ using linear interpolation:

$$\mathbf{e}(p) = (1 - \alpha) \cdot \mathbf{e}^{(\text{lower})} + \alpha \cdot \mathbf{e}^{(\text{upper})},$$

resulting in a continuous embedding $\mathbf{e}(p) \in \mathbb{R}^{D_z}$.

Our method is inherently scalable to high-dimensional parameter vectors ($D_p \gg 1$), with no architectural limitations. The main consideration lies in adapting the interpolation mechanism: simple linear interpolation may no longer suffice. Alternatives such as barycentric interpolation over sparse simplices or kernel-based schemes can be employed to enable smooth generalization across complex parameter spaces.

Appendix A.2. Weight Generation via MLP

The embedding is then passed through a multi-layer perceptron (MLP) to generate the flattened weight vector of the target network:

$$w_f = \text{MLP}\left(\mathbf{e}(p)\right) \in \mathbb{R}^{|w_f|}, \tag{A.1}$$

where the MLP is composed of several hidden layers with activation functions such as SiLU and a final output layer of size $|w_f|$. This architecture allows efficient and expressive mapping from the embedding space to the parameter space of the target network.

Appendix A.3. Training

The entire system consisting of the embedding layer, the MLP-based hypernetwork, and the target temporal forecasting model, is trained end-to-end using gradient-based optimization. The hypernetwork remains fully differentiable with respect to the input parameter **p**, allowing backpropagation through both the embedding interpolation and the weight generation process.

This architecture enables dynamic generation of forecasting models tailored to each parameter configuration without requiring retraining for every new value of **p**. As such, it provides a flexible and efficient approach for modeling dynamical systems across wide parametric domains.

Appendix B. Hyperparameters of Network Architectures Used

The network architectures implemented for benchmarking in this study are summarized in Table 1. For feedforward neural networks (FFNNs), we use three hidden layers with 64 neurons each. For LSTMs, a single recurrent layer with 64 hidden units is employed. The TCNN-CD (Temporal Convolutional Neural Network with Causal Dilations) is configured with a channel size of 64 and a kernel size of 5. We experiment with input sequence lengths of 16 and 32, corresponding to 3 and 4 layers, respectively.

The target network in PHLieNet is also a TCNN-CD with the same configuration (channel size 64, kernel size 5). The hypernetwork of PHLieNet incorporates a Learned Interpolated Embedding (LIE) layer with an embedding size of 64. The number of anchor embeddings is tuned between 2 and 4, depending on the specific parameters and dynamics of the system. This choice is adapted for each system based on the number of parameters observed in the training data and the complexity of the dynamics: four for the Van der Pol and Rössler systems, three for the Lorenz system, and two or three for Chua's circuit.

The weight-generating component of the hypernetwork is a small fully connected network with a single hidden layer of size 32. We did not observe substantial changes in performance when varying these hyperparameters within reasonable ranges, suggesting that the core architecture is the primary factor in the model's effectiveness.

An input sequence length of ISL = 16 was sufficient to achieve satisfactory performance on the Van der Pol oscillator, Lorenz system, and Rössler system. However, for Chua's circuit, which exhibits longer periodic behavior, ISL = 16 proved insufficient, and we therefore increased the input sequence length to ISL = 32.

While additional performance gains could potentially be achieved through further hyperparameter optimization, such tuning would come at increased computational cost and is beyond the scope of this work.

All networks were trained using truncated backpropagation through time with a batch size of 256 and 6 parallel data-loading workers. Training was run for a maximum of 1000 epochs. The initial learning rate was set to 10^{-2}, and reduced by a factor of 0.25 if the minimum validation loss did not decrease by at least 10^{-4} over 15 consecutive epochs. If the validation loss failed to improve by this threshold for 30 consecutive epochs, early stopping was triggered to prevent overfitting.

All input features were standardized using a feature-wise standard scaler. To improve robustness, additive Gaussian noise with standard deviation σ_{noise} was applied during training. All models were trained using the Ranger optimizer [101].

References

[1] S. Gupta, N. Mastrantonas, C. Masoller, J. Kurths, Perspectives on the importance of complex systems in understanding our climate and climate change—the nobel prize in physics 2021, Chaos: An Interdisciplinary Journal of Nonlinear Science 32 (5) (2022).

[2] N. J. Linden, B. Kramer, P. Rangamani, Bayesian parameter estimation for dynamical models in systems biology, PLoS computational biology 18 (10) (2022) e1010651.

[3] D. Durstewitz, G. Koppe, M. I. Thurm, Reconstructing computational system dynamics from neural data with recurrent neural networks, Nature Reviews Neuroscience 24 (11) (2023) 693–710.

[4] B. Bonev, T. Kurth, C. Hundt, J. Pathak, M. Baust, K. Kashinath, A. Anandkumar, Spherical fourier neural operators: Learning stable dynamics on the sphere, in: International conference on machine learning, PMLR, 2023, pp. 2806–2823.

[5] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, Q. Tian, Accurate medium-range global weather forecasting with 3d neural networks, Nature 619 (7970) (2023) 533–538.

[6] M. Farazmand, T. P. Sapsis, Extreme events: Mechanisms and prediction, Applied Mechanics Reviews 71 (5) (2019) 050801.

[7] T. Lassila, A. Manzoni, A. Quarteroni, G. Rozza, Model order reduction in fluid dynamics: challenges and perspectives, Reduced Order Methods for modeling and computational reduction (2014) 235–273.
[8] M. M. Kumbure, C. Lohrmann, P. Luukka, J. Porras, Machine learning techniques and data for stock market forecasting: A literature review, Expert Systems with Applications 197 (2022) 116659.
[9] A. L. Bertozzi, E. Franco, G. Mohler, M. B. Short, D. Sledge, The challenges of modeling and forecasting the spread of covid-19, Proceedings of the National Academy of Sciences 117 (29) (2020) 16732–16738.
[10] V. K. Dertimanis, E. Chatzi, S. E. Azam, C. Papadimitriou, Input-state-parameter estimation of structural systems from limited output information, Mechanical Systems and Signal Processing 126 (2019) 711–746.
[11] R. Vinuesa, S. L. Brunton, Enhancing computational fluid dynamics with machine learning, Nature Computational Science 2 (6) (2022) 358–366.
[12] S. L. Brunton, J. N. Kutz, Promising directions of machine learning for partial differential equations, Nature Computational Science 4 (7) (2024) 483–494.
[13] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM review 57 (4) (2015) 483–531.
[14] M. P. Schultz, K. A. Flack, Reynolds-number scaling of turbulent channel flow, Physics of Fluids 25 (2) (2013).
[15] V. Eyring, P. M. Cox, G. M. Flato, P. J. Gleckler, G. Abramowitz, P. Caldwell, W. D. Collins, B. K. Gier, A. D. Hall, F. M. Hoffman, et al., Taking climate model evaluation to the next level, Nature Climate Change 9 (2) (2019) 102–110.
[16] P. Benner, M. Ohlberger, A. Cohen, K. Willcox, Model reduction and approximation: theory and algorithms, SIAM, 2017.
[17] B. Peherstorfer, K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, Computer Methods in Applied Mechanics and Engineering 306 (2016) 196–215.
[18] D. Amsallem, B. Haasdonk, Pebl-rom: Projection-error based local reduced-order models, Advanced Modeling and Simulation in Engineering Sciences 3 (2016) 1–25.
[19] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, L. Miguel Silveira, Model Order Reduction: Volume 2: Snapshot-Based Methods and Algorithms, De Gruyter, 2020.
[20] J. L. Proctor, S. L. Brunton, J. N. Kutz, Dynamic mode decomposition with control, SIAM Journal on Applied Dynamical Systems 15 (1) (2016) 142–161.
[21] A. Pasquale, M.-J. Kazemzadeh-Parsi, D. Di Lorenzo, V. Champaney, A. Ammar, F. Chinesta, Modular parametric pgd enabling online solution of partial differential equations, Computers & Mathematics with Applications 176 (2024) 244–256.
[22] J. S. Hesthaven, G. Rozza, B. Stamm, et al., Certified reduced basis methods for parametrized partial differential equations, Vol. 590, Springer, 2016.
[23] V. Champaney, F. Chinesta, E. Cueto, Engineering empowered by physics-based and data-driven hybrid models: A methodological overview, International Journal of Material Forming 15 (3) (2022) 31.
[24] J. Barnett, C. Farhat, Y. Maday, Neural-network-augmented projection-based model order reduction for mitigating the kolmogorov barrier to reducibility, Journal of Computational Physics 492 (2023) 112420.
[25] S. Fresca, A. Manzoni, Pod-dl-rom: Enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition, Computer Methods in Applied Mechanics and Engineering 388 (2022) 114181.
[26] K. Vlachas, T. Simpson, A. Garland, D. D. Quinn, C. Farhat, E. Chatzi, Reduced order modeling conditioned on monitored features for response and error bounds estimation in engineered systems, Mechanical Systems and Signal Processing 226 (2025) 112261.
[27] Z. Y. Wan, P. Vlachas, P. Koumoutsakos, T. Sapsis, Data-assisted reduced-order modeling of extreme events in complex dynamical systems, PloS one 13 (5) (2018) e0197704.
[28] X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach, V. Kumar, Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles, ACM/IMS Transactions on Data Science 2 (3) (2021) 1–26.
[29] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational physics 378 (2019) 686–707.
[30] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, Journal of Scientific Computing 92 (3) (2022) 88.
[31] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (6) (2021) 422–440.
[32] T. Qin, A. Beatson, D. Oktay, N. McGreivy, R. P. Adams, Meta-pde: Learning to solve pdes quickly without a mesh, arXiv preprint arXiv:2211.01604 (2022).
[33] F. de Avila Belbute-Peres, Y.-f. Chen, F. Sha, Hyperpinn: Learning parameterized differential equations with physics-informed hypernetworks, The symbiosis of deep learning and differential equations 690 (2021).
[34] X. Huang, Z. Ye, H. Liu, S. Ji, Z. Wang, K. Yang, Y. Li, M. Wang, H. Chu, F. Yu, et al., Meta-auto-decoder for solving parametric partial differential equations, Advances in Neural Information Processing Systems 35 (2022) 23426–23438.
[35] A. Shysheya, C. Diaconu, F. Bergamin, P. Perdikaris, J. M. Hernández-Lobato, R. Turner, E. Mathieu, On conditional diffusion models for pde simulations, Advances in Neural Information Processing Systems 37 (2024) 23246–23300.
[36] M. Haywood-Alexander, W. Liu, K. Bacsa, Z. Lai, E. Chatzi, Discussing the spectrum of physics-enhanced machine learning: a survey on structural mechanics applications, Data-Centric Engineering 5 (2024) e30.
[37] A. Cicirello, Physics-enhanced machine learning: a position paper for dynamical systems investigations, in: Journal of Physics: Conference Series, Vol. 2909, IOP Publishing, 2024, p. 012034.
[38] F. F. de la Mata, A. Gijón, M. Molina-Solana, J. Gómez-Romero, Physics-informed neural networks for data-driven simulation: Advantages, limitations, and opportunities, Physica A: Statistical Mechanics and its Applications 610 (2023) 128415.
[39] A. J. Huang, S. Agarwal, On the limitations of physics-informed deep learning: Illustrations using first-order hyperbolic conservation law-based traffic flow models, IEEE Open Journal of Intelligent Transportation Systems 4 (2023) 279–293.
[40] T. Iwata, Y. Tanaka, N. Ueda, Meta-learning of physics-informed neural networks for efficiently solving newly given pdes, arXiv preprint

arXiv:2310.13270 (2023).
[41] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, E. Ott, Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data, Chaos: An Interdisciplinary Journal of Nonlinear Science 27 (12) (2017).
[42] J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Physical review letters 120 (2) (2018) 024102.
[43] Z.-M. Zhai, J.-Y. Huang, B. D. Stern, Y.-C. Lai, Reconstructing dynamics from sparse observations with no training on target system, arXiv preprint arXiv:2410.21222 (2024).
[44] S. Hochreiter, Long short-term memory, Neural Computation MIT-Press (1997).
[45] P.-R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, P. Koumoutsakos, Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics, Neural Networks 126 (2020) 191–217.
[46] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, P. Koumoutsakos, Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474 (2213) (2018) 20170844.
[47] P. R. Vlachas, G. Arampatzis, C. Uhler, P. Koumoutsakos, Multiscale simulations of complex systems by learning their effective dynamics, Nature Machine Intelligence 4 (4) (2022) 359–366.
[48] I. Kičić, P. R. Vlachas, G. Arampatzis, M. Chatzimanolakis, L. Guibas, P. Koumoutsakos, Adaptive learning of effective dynamics for online modeling of complex systems, Computer Methods in Applied Mechanics and Engineering 415 (2023) 116204.
[49] N. Geneva, N. Zabaras, Transformers for modeling physical systems, Neural Networks 146 (2022) 272–289.
[50] L. Lu, P. Jin, G. E. Karniadakis, Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, arXiv preprint arXiv:1910.03193 (2019).
[51] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Learning maps between function spaces with applications to pdes, Journal of Machine Learning Research 24 (89) (2023) 1–97.
[52] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895 (2020).
[53] V. S. Fanaskov, I. V. Oseledets, Spectral neural operators, in: Doklady Mathematics, Vol. 108, Springer, 2023, pp. S226–S232.
[54] B. Raonic, R. Molinaro, T. De Ryck, T. Rohner, F. Bartolucci, R. Alaifari, S. Mishra, E. de Bézenac, Convolutional neural operators for robust and accurate learning of pdes, Advances in Neural Information Processing Systems 36 (2023) 77187–77200.
[55] B. Raonic, R. Molinaro, T. Rohner, S. Mishra, E. de Bezenac, Convolutional neural operators, in: ICLR 2023 Workshop on Physics for Machine Learning, 2023.
[56] Z. Lai, W. Liu, X. Jian, K. Bacsa, L. Sun, E. Chatzi, Neural modal ordinary differential equations: Integrating physics-based modeling with neural ordinary differential equations for modeling high-dimensional monitored structures, Data-Centric Engineering 3 (2022) e34.
[57] E. Dupont, A. Doucet, Y. W. Teh, Augmented neural odes, Advances in neural information processing systems 32 (2019).
[58] C. R. Constante-Amores, A. J. Linot, M. D. Graham, Data-driven prediction of large-scale spatiotemporal chaos with distributed low-dimensional models, arXiv preprint arXiv:2410.01238 (2024).
[59] C. G. Wagner, Stacked tensorial neural networks for reduced-order modeling of a parametric partial differential equation, arXiv preprint arXiv:2312.14979 (2023).
[60] N. Franco, A. Manzoni, P. Zunino, A deep learning approach to reduced order modelling of parameter dependent partial differential equations, Mathematics of Computation 92 (340) (2023) 483–524.
[61] H. R. Karbasia, W. M. van Rees, A parametric lstm neural network for predicting flow field dynamics across a design space, in: Proceedings A, Vol. 481, The Royal Society, 2025, p. 20240055.
[62] W. Cho, M. Jo, H. Lim, K. Lee, D. Lee, S. Hong, N. Park, Parameterized physics-informed neural networks for parameterized pdes, arXiv preprint arXiv:2408.09446 (2024).
[63] N. Farenga, S. Fresca, S. Brivio, A. Manzoni, On latent dynamics learning in nonlinear reduced order modeling, Neural Networks (2025) 107146.
[64] H. Luo, Y. Du, H. Fan, X. Wang, J. Guo, X. Wang, Reconstructing bifurcation diagrams of chaotic circuits with reservoir computing, Physical Review E 109 (2) (2024) 024210.
[65] G. Langer, U. Parlitz, Modeling parameter dependence from time series, Physical Review E—Statistical, Nonlinear, and Soft Matter Physics 70 (5) (2004) 056217.
[66] M. Roy, S. Mandal, C. Hens, A. Prasad, N. Kuznetsov, M. Dev Shrimali, Model-free prediction of multistability using echo state network, Chaos: An Interdisciplinary Journal of Nonlinear Science 32 (10) (2022).
[67] D. Ha, A. Dai, Q. V. Le, Hypernetworks, arXiv preprint arXiv:1609.09106 (2016).
[68] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International conference on machine learning, PMLR, 2017, pp. 1126–1135.
[69] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning, in: International conference on learning representations, 2017.
[70] B. Zoph, Neural architecture search with reinforcement learning, arXiv preprint arXiv:1611.01578 (2016).
[71] H. Pham, M. Guan, B. Zoph, Q. Le, J. Dean, Efficient neural architecture search via parameters sharing, in: International conference on machine learning, PMLR, 2018, pp. 4095–4104.
[72] H. Liu, K. Simonyan, Y. Yang, Darts: Differentiable architecture search, arXiv preprint arXiv:1806.09055 (2018).
[73] Y. Chai, R. Giryes, L. Wolf, Supervised and unsupervised learning of parameterized color enhancement, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 992–1000.
[74] Y. Alaluf, O. Tov, R. Mokady, R. Gal, A. Bermano, Hyperstyle: Stylegan inversion with hypernetworks for real image editing, in: Proceedings of the IEEE/CVF conference on computer Vision and pattern recognition, 2022, pp. 18511–18521.
[75] Y. Li, S. Gu, K. Zhang, L. Van Gool, R. Timofte, Dhp: Differentiable meta pruning via hypernetworks, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16, Springer, 2020, pp. 608–624.
[76] V. K. Chauhan, J. Zhou, P. Lu, S. Molaei, D. A. Clifton, A brief review of hypernetworks in deep learning, Artificial Intelligence Review

57 (9) (2024) 1–29.
[77] B. Klein, L. Wolf, Y. Afek, A dynamic convolutional layer for short range weather prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 4840–4848.
[78] J. Berman, B. Peherstorfer, Colora: Continuous low-rank adaptation for reduced implicit neural modeling of parameterized partial differential equations, arXiv preprint arXiv:2402.14646 (2024).
[79] L. Zheng, D. M. Kochmann, S. Kumar, Hypercan: Hypernetwork-driven deep parameterized constitutive models for metamaterials, Extreme Mechanics Letters (2024) 102243.
[80] M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, Hypersolvers: Toward fast continuous-depth models, Advances in Neural Information Processing Systems 33 (2020) 21105–21117.
[81] R. Majumdar, V. Jadhav, A. Deodhar, S. Karande, L. Vig, V. Runkana, Hyperlora for pdes, arXiv preprint arXiv:2308.09290 (2023).
[82] W. Cho, K. Lee, D. Rim, N. Park, Hypernetwork-based meta-learning for low-rank physics-informed neural networks, Advances in Neural Information Processing Systems 36 (2023) 11219–11231.
[83] M. Kirchmeyer, Y. Yin, J. Donà, N. Baskiotis, A. Rakotomamonjy, P. Gallinari, Generalizing to new physical systems via context-informed dynamics model, in: International Conference on Machine Learning, PMLR, 2022, pp. 11283–11301.
[84] Y. Yin, I. Ayed, E. de Bézenac, N. Baskiotis, P. Gallinari, Leads: Learning dynamical systems that generalize across environments, Advances in Neural Information Processing Systems 34 (2021) 7561–7573.
[85] T. Galanti, L. Wolf, On the modularity of hypernetworks, Advances in Neural Information Processing Systems 33 (2020) 10409–10419.
[86] Y. Liu, J. N. Kutz, S. L. Brunton, Hierarchical deep learning of multiscale differential equation time-steppers, Philosophical Transactions of the Royal Society A 380 (2229) (2022) 20210200.
[87] R. T. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, Advances in neural information processing systems 31 (2018).
[88] I. Sutskever, Training recurrent neural networks, University of Toronto Toronto, ON, Canada, 2013.
[89] A. Graves, Generating sequences with recurrent neural networks, arXiv preprint arXiv:1308.0850 (2013).
[90] S. Bai, J. Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, arXiv preprint arXiv:1803.01271 (2018).
[91] F. Takens, Dynamical systems and turbulence, Warwick, 1980 (1981) 366–381.
[92] M. Brenner, E. Weber, G. Koppe, D. Durstewitz, Learning interpretable hierarchical dynamical systems models from time series data, arXiv preprint arXiv:2410.04814 (2024).
[93] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Advances in neural information processing systems 32 (2019).
[94] S. S. Sudhakaran, hyper-nn, `https://github.com/shyamsn97/hyper-nn` (2022).
[95] B. Van der Pol, Lxxxviii. on "relaxation-oscillations", The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (11) (1926) 978–992.
[96] S. H. Strogatz, Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering, CRC press, 2018.
[97] E. Lorenz, Deterministic nonperiodic flow, Journal of Atmospheric Sciences 20 (2) (1963).
[98] O. E. Rössler, An equation for continuous chaos, Physics Letters A 57 (5) (1976) 397–398.
[99] L. O. Chua, Global unfolding of chua's circuit, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 76 (5) (1993) 704–734.
[100] L. Chua, M. Komuro, T. Matsumoto, The double scroll family, IEEE transactions on circuits and systems 33 (11) (1986) 1072–1118.
[101] L. Wright, N. Demeure, Ranger21: a synergistic deep learning optimizer, arXiv preprint arXiv:2106.13731 (2021).