Mathematical and Computational Applications

MDPI

Article

Reduced Order Modeling Using Advection-Aware Autoencoders

Sourav Dutta [1,*], Peter Rivera-Casillas [2], Brent Styles [1,3] and Matthew W. Farthing [1]

[1] Coastal and Hydraulics Laboratory, U.S. Army Engineer Research and Development Center, 3909 Halls Ferry Rd., Vicksburg, MS 39180, USA; brent.j.styles@usace.army.mil (B.S.); matthew.w.farthing@erdc.dren.mil (M.W.F.)
[2] Information Technology Laboratory, U.S. Army Engineer Research and Development Center, 3909 Halls Ferry Rd., Vicksburg, MS 39180, USA; peter.g.rivera-casillas@erdc.dren.mil
[3] Departmentof Electrical & Computer Engineering, University of Alabama in Huntsville, 301 Sparkman Drive, Huntsville, AL 35899, USA
* Correspondence: sourav.dutta@erdc.dren.mil

Abstract: Physical systems governed by advection-dominated partial differential equations (PDEs) are found in applications ranging from engineering design to weather forecasting. They are known to pose severe challenges to both projection-based and non-intrusive reduced order modeling, especially when linear subspace approximations are used. In this work, we develop an advection-aware (AA) autoencoder network that can address some of these limitations by learning efficient, physics-informed, nonlinear embeddings of the high-fidelity system snapshots. A fully non-intrusive reduced order model is developed by mapping the high-fidelity snapshots to a latent space defined by an AA autoencoder, followed by learning the latent space dynamics using a long-short-term memory (LSTM) network. This framework is also extended to parametric problems by explicitly incorporating parameter information into both the high-fidelity snapshots and the encoded latent space. Numerical results obtained with parametric linear and nonlinear advection problems indicate that the proposed framework can reproduce the dominant flow features even for unseen parameter values.

Keywords: deep autoencoder; advection-dominated flows; physics informed machine learning; LSTM; parametric model order reduction; non-intrusive reduced order modeling

Citation: Dutta, S.; Rivera-Casillas, P.; Styles, B.; Farthing, M.W. Reduced Order Modeling Using Advection-Aware Autoencoders. *Math. Comput. Appl.* **2022**, *27*, 34. https://doi.org/10.3390/mca27030034

Academic Editors: Simona Perotto, Gianluigi Rozza and Antonia Larese

Received: 2 February 2022
Accepted: 16 April 2022
Published: 21 April 2022

1. Introduction

Modern scientific computing relies on efficient numerical simulation of complex physical systems, especially for applications that seek solutions at different time or parameter instances. For these types of applications, the relevant physical system is typically described by a set of parameterized nonlinear partial differential equations (PDEs). Numerical discretizations of such systems using a high-fidelity (finite element, finite volume, or finite difference type) computational solver can be prohibitively expensive as they generate high-dimensional representations of the solution in order to accurately resolve multiple time and space scales and underlying nonlinearities [1]. However, there is compelling scientific evidence to suggest that the underlying dynamics often exhibit low-dimensional structure [2]. Reduced order models (ROMs) can replace such expensive high-fidelity solvers by exploiting the intrinsic, low-rank structure of the simulation data in order to create more tractable models for the spatiotemporal evolution dynamics of the PDE system [3,4].

Among the many different classes of ROM techniques that have been developed over the years, projection-based ROMs occupy a prominent place across a wide range of applications [5]. Formally, this class of methods is based on the identification of a reduced set of basis functions (or modes) such that their linear superposition spans an optimal low-rank approximation of the solution manifold. The Proper Orthogonal Decomposition (POD) is one of the widely popular methods of this class that leverages the singular value decomposition (SVD) to determine an empirical basis of dominant, orthonormal modes that can help define the best possible linear subspace in which to project the

PDE dynamics [6,7]. If the governing equations are known, Galerkin projection [8,9], or the Petrov–Galerkin projection [10,11], can be adopted to generate an interpretable ROM defined by the high-energy or dominant modes. For applications where the governing equations are not accessible, purely data-driven methods for non-intrusive ROM (NIROM) [12] have gained in popularity. In these methods, instead of a Galerkin-type projection, the expansion coefficients for the reduced solution are obtained via interpolation on the reduced basis space spanned by the set of dominant modes. However, since the reduced dynamics generally belong to nonlinear, matrix manifolds, a variety of interpolation and regression methods have been proposed, which are capable of enforcing the constraints characterizing those manifolds. Some notable examples in this class include dynamic mode decomposition (DMD) [13,14], radial basis function interpolation [15,16], and Gaussian process regression [17,18], to name a few. In addition, the emergence of modern machine learning (ML) methods has provided a transformative approach to effectively approximate and accelerate existing numerical models by leveraging the capabilities to incorporate multi-fidelity datastreams from diverse sources, seamlessly explore massive design spaces, and identify complex, multivariate correlations. A variety of data-driven, ML-based approximation frameworks have been proposed to model the propagation of system dynamics in the latent space. Some of the highly successful examples involve the use of deep neural networks (DNNs) [19], long-short-term memory (LSTM) networks [20,21], neural ordinary differential equations (NODE) [22,23], and temporal convolutional networks (TCNs) [24].

One fundamental assumption of linear reduced basis methods like POD is that any element in the solution manifold, $\mathcal{M}$ of the nonlinear PDE system can be accurately approximated using a linear combination of a small number of basis functions. Traditionally this concept is quantified by the Kolmogorov n-width, $\mathcal{D}_n$, which measures the error introduced by approximating any element f of $\mathcal{M}$ with an element g of a linear space E_n. A common heuristic approach to get a rough estimate of $\mathcal{D}_n$ for a particular discretized solution manifold is to examine the rate of decay of the singular values obtained by a SVD of the system snapshots. A fast rate of decay signifies a small $\mathcal{D}_n$, which indicates the existence of a low-dimensional space in which the high-fidelity nonlinear system can be approximated well. Many PDE systems of importance, however, exhibit transport-dominated behavior (e.g., advection-dominated flows, wave and shock propagation phenomena), which leads to a large Kolmogorov n-width. For instance, a stationary soliton wave solution can be perfectly captured by one spatial mode, as reflected in a rapid decay of the corresponding POD singular values, whereas a wave translating in time cannot be represented by a low-dimensional representation with POD/SVD. This is because the steep gradients and moving spatial discontinuities inherent to these problems often trigger temporal discontinuities. An accurate linear approximation of such temporal discontinuities requires a large number of basis functions, hindering the efficiency of a ROM [25,26], and often leads to an oscillatory approximation [27,28]. These inadequacies have inspired a growing number of works in recent years, which focus on constructing an efficient alternative—nonlinear ROMs for transport-dominated systems. A brief review will be provided in the following section. The focus of this work will be the study of ML-based, non-intrusive reduced order modeling strategies for transport-dominated systems.

2. Related Work

As the traditional approach for constructing a ROM for transport-dominated problems has proven to be ineffective due to the limitations of a linear subspace approximation, there has been significant interest in alternative modifications for improving the accuracy of ROM approximations in these applications. These can be roughly classified into two distinct approaches: transformations of the linear subspace to facilitate better mode extraction and improved stability of projection-based ROMs [28–34], and the construction of low-rank representations in terms of nonlinear manifolds [35–37]. Most of the previous work in the first class of methods has been focused on either (a) sparse sampling of nonlinear terms to enable efficient approximation in a reduced subspace like gappy POD [38], GNAT [27], DEIM [39],

or (b) pre-processing the linear subspace to embed the dominant advective features of the solution [28–31,40]. A collection of methods in this class has also been based around the concept of adaptivity. For instance, offline adaptive methods either extend [41] or create a weighted [42] snapshot database during the construction of the reduced model. Online adaptive methods, on the other hand, either rely on precomputed quantities to update the reduced basis online using interpolation, localization, and dictionary approaches [43–45], or allow for the incorporation of new data online [46,47]. Almost all of the above techniques from the first class of methods require some kind of problem-specific prior knowledge of the physical or numerical properties of the underlying nonlinear system, thus imposing some limitations on their applicability to experimental data or systems with no access to governing equations and closed-form solutions.

An alternative perspective on the limitations of linear subspace approximation in ROM design is based upon the observation that many PDE systems of importance, especially in fluids, contain symmetries such as rotations, translations, and scaling, which play a foundational role in the dynamics. Traditional ROM approaches such as the SVD-based methods are unable to handle these symmetries, and are only truly effective for dynamical systems where time and space interactions can be essentially decoupled through separation of variables [7]. To overcome these limitations, the second class of methods based on nonlinear manifold learning have recently gained a lot of research interest. Some of the earliest examples of methods in this class include Iso-map [48], Locally linear embedding (LLE) [49], Laplacian eigenmaps [50], and t-SNE [51]. However, these methods fail to provide a mapping from the low-dimensional nonlinear representation to the high-dimensional input, which is a crucial tool for dimension reduction applications. Many other novel approaches have been proposed to overcome this gap, such as self-organizing maps [52], kernel PCA [53], diffeormorphic dimensionality reduction [54], and autoencoders [55] (see [36] for a survey). In recent years, due to the tremendous progress in the development of high-performance software tools for the construction of neural networks based models, different types of autoencoder models [56,57] have emerged, as some of the most popular and powerful techniques for nonlinear manifold-based dimension reduction of PDEs. These have been successfully applied to different types of ROM applications such as deep fully-connected autoencoders [58,59], deep convolutional autoencoders (CAEs) [36,60], time-lagged autoencoders [61], shallow masked autoencoders [37], variational autoencoders [62], and deep delay autoencoders [63].

In this work, we propose an advection-aware (AA) autoencoder design, in which a high-fidelity system snapshot is mapped through a shared latent space to an approximation of itself and simultaneously to another arbitrary snapshot. For advection-dominated problems, this arbitrary snapshot can be chosen in a physics-guided manner to primarily represent the advective features, thus allowing the latent space to more efficiently identify reduced-representations of high-fidelity solution fields. We then employ LSTM neural networks to non-intrusively model the temporal evolution of these compressed latent representations. Moreover, our approach enables exploration of parametric search spaces by training on a combined parametric dataset of offline simulations. In contrast, the studies outlined in Refs. [36,60] use a convolutional autoencoder architecture to nonlinearly embed high-dimensional states, and this may pose problems when the high-fidelity simulation data is available on unstructured computational meshes. In addition, Ref. [36] adopts an intrusive approach by solving the governing equations on the nonlinear manifold defined by the CAE model, whereas Ref. [60] employs a similar idea of modeling latent dynamics using recurrent neural networks like an LSTM. Another interesting approach is proposed by Ref. [64], where the authors introduce the idea of imposing Lyapunov stability-preserving priors to the autoencoder-based model in order to improve the generalization performance for fluid flow prediction. While this approach is similar to ours in being motivated by the idea of physics-informed learning, the ultimate objective of the proposed design is different. Our approach also differs from the framework proposed in [65] as the system parameters such as shape of the profile, flow speed, and viscosity are explicitly

embedded in the input feature space and the latent space, thus allowing independent training of the AA autoencoder and the LSTM networks. In Ref. [66], a registration-based approach is proposed, which trains a diffeomorphic mapping between the physical space and a new parameter-varying, spatio-temporal grid on which the solution of the PDE can be expressed in the form of a low-rank linear decomposition. This low-rank time/parameter-varying grid or manifold is utilized as an autoencoder type layer for reducing the dimension of high-fidelity snapshots. This is an elegant approach, but involves solving optimization problems with nonlinear constraints and performing repeated 2D interpolation tasks, both of which may potentially lead to efficiency issues and introduce approximation errors for large-scale problems.

The rest of the article is organized as follows. In Section 3, we provide a high-level overview of undercomplete autoencoders, followed by details on the proposed AA autoencoder network design and training strategies. We also include a brief review of LSTM networks, which have been adopted in this work to model the system dynamics in the nonlinear latent space. In Section 4, we present numerical results obtained with the proposed AA autoencoder model on two different types of parametric problems characterized by advection-dominated flow features. Finally, in Section 5, we present some concluding remarks and discuss plans for future work.

3. Methodology

3.1. Autoencoders

An autoencoder is a type of neural network that is designed to learn an approximation of the identity mapping, $\chi : \mathbf{v} \mapsto \tilde{\mathbf{v}}$ such that $\tilde{\mathbf{v}} \approx \mathbf{v}$ and $\chi : \mathbb{R}^N \mapsto \mathbb{R}^N$. This is accomplished using a two-part architecture. Figure 1 shows an example of a fully connected autoencoder network with two distinct parts. The first part is called an encoder, χ_e, which maps a high-dimensional input vector $\mathbf{v}$ to a low-dimensional latent vector $\mathbf{z}$ as given by $\mathbf{z} = \chi_e(\mathbf{v}; \boldsymbol{\theta}_e)$ where $\mathbf{z} \in \mathbb{R}^m$ ($m \ll N$).

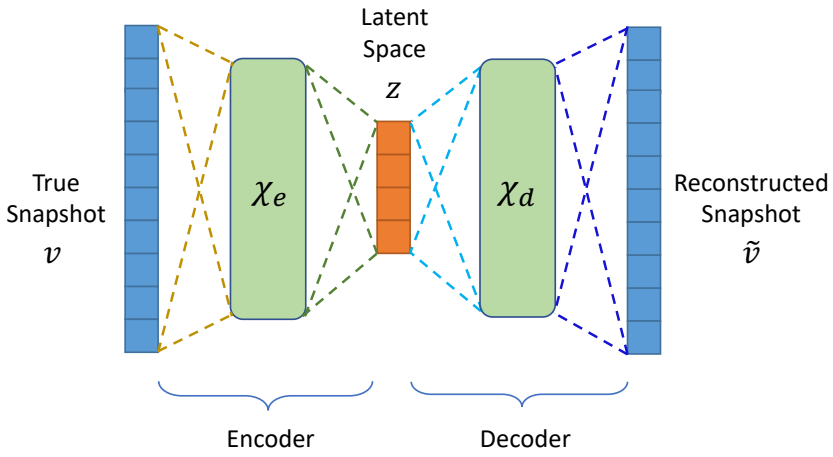


Figure 1. Fully connected autoencoder architecture.

The second part is called a decoder, χ_d, which maps the latent vector $\mathbf{z}$ to an approximation $\tilde{\mathbf{v}}$ of the high-dimensional input vector $\mathbf{v}$ and is defined as $\tilde{\mathbf{v}} = \chi_d(\mathbf{z}; \boldsymbol{\theta}_d)$. The combination of these two parts yields an autoencoder of the form

$$\chi : \mathbf{v} \mapsto \chi_d \circ \chi_e(\mathbf{v}). \tag{1}$$

This autoencoder network is trained by computing the optimal values of the parameters $(\boldsymbol{\theta}_e^*, \boldsymbol{\theta}_d^*)$ that minimize the reconstruction error over all the training data

$$\boldsymbol{\theta}_e^*, \boldsymbol{\theta}_d^* = \underset{\boldsymbol{\theta}_e, \boldsymbol{\theta}_d}{\operatorname{argmin}} \mathcal{L}(\mathbf{v}, \tilde{\mathbf{v}}), \tag{2}$$

where $\mathcal{L}(\mathbf{v}, \widetilde{\mathbf{v}})$ is a chosen measure of discrepancy between $\mathbf{v}$ and its approximation $\widetilde{\mathbf{v}}$. The restriction $\dim(\mathbf{z}) = m \ll N = \dim(\mathbf{v})$ forces the autoencoder model to learn the salient features of the input data via compression into a low-dimensional space and then reconstructing the input, instead of directly learning the identity function. Essentially, autoencoders can be thought of as a powerful generalization of the POD/SVD approach from learning a linear subspace to identifying an improved coordinate system on a nonlinear manifold. That is, with the choice of a linear, single-layer encoder of the form $\mathbf{z} = H_E\mathbf{v}$, and a linear, single-layer decoder of the form $\widetilde{\mathbf{v}} = \mathbf{H}_D\mathbf{z}$, where $\mathbf{H}_E \in \mathbb{R}^{m\times N}$, $\mathbf{H}_D \in \mathbb{R}^{N\times m}$, and a squared reconstruction error as the loss function $\mathcal{L}(\widetilde{\mathbf{v}}, \mathbf{v}) = \|\mathbf{v} - \widetilde{\mathbf{v}}\|_2^2$, the autoencoder model has been shown to learn the same subspace as that spanned by the first m POD modes if $\mathbf{H} = \mathbf{H}_E = \mathbf{H}_D$. However, additional constraints are necessary to ensure that the columns of $\mathbf{H}$ form an orthonormal basis and follow an energy-based hierarchical ordering [67].

3.2. Advection-Aware Autoencoder Design

In this work, inspired by the registration-based nonlinear manifold learning idea [66] and the physics-informed autoencoder model design [64], we develop a new advection-aware autoencoder model that incorporates physical knowledge of the advection-dominated flow features into the autoencoder neural network via both an inductive bias as well as soft constraints. As shown in Figure 2, this AA autoencoder architecture is composed of three sub-networks. The first part, as usual, is called an encoder, χ_e, which maps a high-dimensional input snapshot $\mathbf{v}$ to a low-dimensional latent vector $\mathbf{z}$, and is defined by $\mathbf{z} = \chi_e(\mathbf{v}; \boldsymbol{\theta}_e)$ where $\mathbf{z} \in \mathbb{R}^m$ ($m \ll N$).

Two independent decoder networks are also defined, which map the latent vector to (i) a transformed (or "shifted") version of the high-dimensional input snapshot, and (ii) back to the true high-dimensional input snapshot. The first of these two decoders is called a shift decoder, $\boldsymbol{\phi}_s$, that maps the latent vector $\mathbf{z}$ to an approximation $\widetilde{\mathbf{v}}_s$ of a suitably defined "shifted" snapshot $\mathbf{v}_s$ that encapsulates the dominant advective features of the flow problem, $\widetilde{\mathbf{v}}_s = \boldsymbol{\phi}_s(\mathbf{z}; \boldsymbol{\theta}_s)$. This can be achieved by following a registration-type approach, where a high-fidelity snapshot at a randomly chosen time point in the simulation is selected to be the candidate output target for the shift decoder. The arbitrariness of the choice could be partially resolved if a known physical characteristic of the flow like dissipation or multiscale oscillations indicates that a particular time point such as the initial solution or a time point at the beginning of a period is a more preferable candidate for the output target of $\boldsymbol{\phi}_s$. This approach is flexible, by design, as it does not require any additional knowledge of the dominant advection patterns of the flow, such as speed of propagation, in order to train the shift decoder, and has been adopted for the numerical experiments in this study. Alternatively, if some partial knowledge of the dominant advective flow features are available, then a transported snapshot could be defined, following the ideas in [29,30]. In this approach, a set of time-varying coordinates are defined by transporting the high-dimensional computational grid using the dominant advection properties such as speed and direction of propagation. Then, the true simulation snapshots are mapped onto the time-varying grid using a suitable interpolation technique to produce the time-varying output target for the shift decoder. The primary advantage of this approach is that it preserves some of the structural properties of any secondary features of the flow problem such as frictional dissipation or the wake patterns trailing a moving vessel. In this way, this approach allows an improved isolation of the dominant advective features of the flow. However, the additional requirements of physical knowledge about the flow, and the potential approximation errors introduced by the interpolation technique are some of the primary issues that need to resolved.

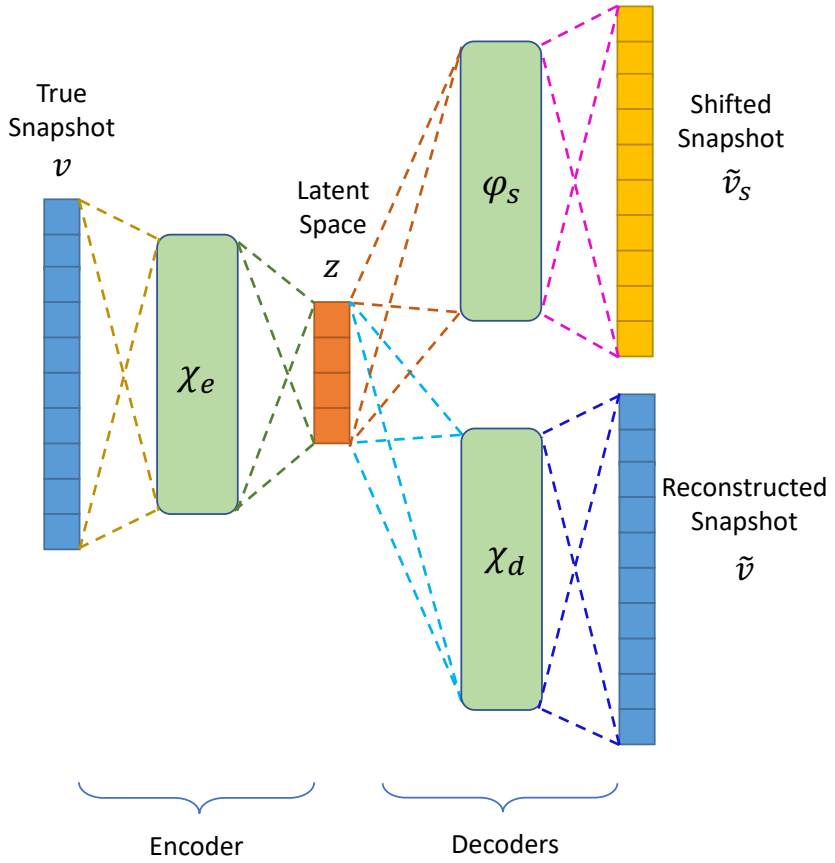


Figure 2. Advection-aware autoencoder architecture. An encoder network χ_e extracts the dominant features of $v \in \mathbb{R}^N$ into a compressed latent space $z \in \mathbb{R}^k$. One decoder network $\boldsymbol{\phi}_s$ maps the latent vector to the shifted snapshot, $v_s \in \mathbb{R}^N$. The second decoder network χ_d maps the latent vector back to an approximation of itself, $\tilde{v} \in \mathbb{R}^N$.

The third and final sub-network is called a true decoder, χ_d, which maps the latent vector $\mathbf{z}$ to an approximation $\tilde{\mathbf{v}}$ of the high-dimensional input snapshot $\mathbf{v}$, and is defined as a traditional decoder network as $\tilde{\mathbf{v}} = \chi_d(\mathbf{z}; \boldsymbol{\theta}_d)$. The combination of the encoder network χ_e and the true decoder network χ_d yields an autoencoder network given by (1).

Such a network enables us to express high-dimensional snapshots in terms of low-dimensional nonlinear manifolds, and can be employed in traditional non-intrusive reduced order modeling frameworks. On the other hand, the combination of the encoder network χ_e and the shift decoder network $\boldsymbol{\phi}_s$ creates a nonlinear mapping between the true snapshots $\mathbf{v}$ and the "shifted" snapshots $\mathbf{v}_s$ in the high-dimensional physical space, such that this transformation map learns about the dominant advective features of the flow.

$$\widehat{\chi} : \mathbf{v} \mapsto \boldsymbol{\phi}_s \circ \chi_e(\mathbf{v}). \tag{3}$$

The primary contribution of this AA autoencoder design is that simultaneous training of these two partially-coupled autoencoder and transformation maps, χ and $\widehat{\chi}$, respectively, endows the intermediate nonlinear, latent space manifold with the information about the dominant advection characteristics of the flow, as well. The simultaneous training can be performed by defining two separate loss functions. The shift loss, $\mathcal{L}_1$, is defined as a measure of discrepancy between the prediction of the shift decoder and the high-dimensional "shifted" snapshot, $\mathcal{L}_1 = \|\mathbf{v}_s - \boldsymbol{\phi}_s \circ \chi_e(\mathbf{v})\|_V$, where $\|\|_V$ denotes a chosen error norm. The second loss function is called the reconstruction loss, $\mathcal{L}_2$, and is defined as the error between the prediction of the true decoder and the high-dimensional true snapshot, $\mathcal{L}_2 = \|\mathbf{v} - \chi_d \circ \chi_e(\mathbf{v})\|_V$, in the same error norm $\|\|_V$. The AA autoencoder is

trained by computing the optimal values of the parameters $(\boldsymbol{\theta}_e^*, \boldsymbol{\theta}_d^*, \boldsymbol{\theta}_s^*)$ that simultaneously minimize a weighted combination of these two loss components over all the training data

$$\boldsymbol{\theta}_e^*,\ \boldsymbol{\theta}_d^*,\ \boldsymbol{\theta}_s^* = \underset{\boldsymbol{\theta}_e, \boldsymbol{\theta}_d, \boldsymbol{\theta}_s}{\operatorname{argmin}} \left\{ w_1 \mathcal{L}_1(\mathbf{v}, \widetilde{\mathbf{v}}_s) + w_2 \mathcal{L}_2(\mathbf{v}, \widetilde{\mathbf{v}}) \right\}, \tag{4}$$

where w_1, w_2 are the weights of the linear combination that could either be fixed during training or be a part of the trainable hyperparameters. In this work, AA autoencoder networks are trained to produce an advection-informed latent space representation of the high-fidelity numerical solution of a parametric linear advection problem and a parametric viscous advecting shock problem. LSTM networks are trained to model the temporal dynamics of the latent space coefficients. The AA autoencoder and the LSTM dynamics model are combined to construct a fully non-intrusive, physics-aware reduced order model for the advection-dominated test problems.

3.3. Long-Short-Term Memory (LSTM) Network

An LSTM network is a special type of recurrent neural network (RNN) that is well-suited for performing classification and regression tasks based on time series data. The main difference between the traditional RNN and the LSTM architecture is the capability of an LSTM memory cell to retain information over time, and an internal gating mechanism that regulates the flow of information in and out of the memory cell [68]. A very concise overview of LSTM networks as applied in the context of model reduction can be found in Ref. [60].

The LSTM cell consists of three parts, also known as gates, that have specific functions. The first part, called the forget gate, chooses whether the information coming from the previous step in the sequence is to be remembered or can be forgotten. The second part, called the input gate, tries to learn new information from the current input to this cell. The third and final part, called the output gate, passes the updated information from the current step to the next step in the sequence. The basic LSTM equations for an arbitrary input vector $\mathbf{u}$ are

$$\begin{aligned}
\text{input gate:} \quad & \zeta_i = \alpha_S \circ \mathcal{F}_i(\mathbf{u}), \\
\text{forget gate:} \quad & \zeta_f = \alpha_S \circ \mathcal{F}_f(\mathbf{u}), \\
\text{cell state:} \quad & \mathbf{c}_t = \zeta_f \odot \mathbf{c}_{t-1} + \zeta_i \odot (\alpha_T \circ \mathcal{F}_a(\mathbf{u})), \\
\text{output gate:} \quad & \zeta_o = \alpha_S \circ \mathcal{F}_o(\mathbf{u}), \\
\text{output:} \quad & \mathbf{h}_t = \zeta_o \circ \alpha_T(\mathbf{c}_t).
\end{aligned} \tag{5}$$

Here, $\mathcal{F}$ refers to a linear transformation defined by a matrix multiplication and bias addition, that is, $\mathcal{F}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, where $\mathbf{W} \in \mathbb{R}^{n \times m}$ is a matrix of layer weights, $\mathbf{b} \in \mathbb{R}^n$ is a vector of bias values, and $\mathbf{x} \in \mathbb{R}^m$ is a vector of layer activations. Also, α_S and α_T denote sigmoid and hyperbolic tangent activation functions, which are usually the default choices in an LSTM network, and $\mathbf{x} \odot \mathbf{y}$ denotes a Hadamard product of two vectors $\mathbf{x}$ and $\mathbf{y}$. In the context of reduced order modeling, the vector $\mathbf{u}$ represents a linear or nonlinearly encoded snapshot vector, with which the LSTM network is trained to advance with time. The core concept of an LSTM network is the cell state $\mathbf{c}_t$, which behaves as the "memory" of the network. It can either allow greater preservation of past information, reducing the issues of short-term memory, or it can suppress the influence of the past depending on the actions of the various gates during the training process.

LSTM networks have proven to be an effective tool in the development of reduced order models for physical systems and have shown that they can outperform alternate classical methods such as DMD and POD-Galerkin, as well as other flavors of RNNs that often suffer from issues with vanishing gradients and the transmission of long-term information [20,69–71]. Different ML methods for time series modeling, such as the neural ordinary differential equations (NODE) [23,59], spatial transformer networks [72], echo

state networks [73], and residual networks (ResNets) [74] have also been shown to be very accurate in various ROM applications for dynamical systems. Unfortunately, many of these newer approaches are not readily available as packages or modules inside well-known machine learning libraries such as TensorFlow and PyTorch. However, LSTM implementations are included as part of the core, highly-efficient, GPU-accelerated modules of all these libraries. Hence, owing to the ease of implementation and the well-known success stories of LSTM-based prediction models for dynamical systems, we have adopted it as our method of choice for modeling of latent space dynamics.

We train an LSTM network to independently learn the temporal evolution of the latent space coefficients generated by the encoder of a pre-trained AA autoencoder model, following a similar approach as in [60,69]. The decoupling of the AA autoencoder training for a nonlinear embedding and the LSTM training for latent space dynamics allows for greater flexibility in our non-intrusive ROM development. If alternate time series learning methods are available that better suit the needs of the problem in a future time, the nonlinear manifold defined by the pre-trained AA autoencoder will not need to be retrained. Moreover, an end-to-end, simultaneous training of an AA autoencoder and a time series learning method like LSTM requires the development of a carefully weighted loss function that appropriately penalizes both the reconstruction and the forecast accuracy. This can often lead to significant loss in both the training efficiency as well as in the overall robustness of the training algorithm.

4. Results

In this section, we demonstrate the capability of the advection-aware autoencoder architecture to generate a compressed representation for high-fidelity snapshots of two different advection-dominated problems. Furthermore, we present numerical results to illustrate the potential of training reduced order models for the system dynamics in the latent space generated by the pre-trained AA autoencoder models. In this study, LSTM architectures are chosen to build these dynamics models for the purposes of illustration. However, the methodology could be easily adapted to use any other approximation framework that might be more appropriate for a particular problem.

4.1. Linear Advection Problem

Consider the advection of a circular Gaussian pulse traveling in the positive $y-$direction through a rectangular domain, $\Omega = [-100, 100] \times [0, 500]$ at a constant speed, c. The analytical solution is given by

$$u(x, y) = \exp\left\{ -\left(\frac{(x - x_0)^2}{2\sigma_x^2} + \frac{(y - y_0 - ct)^2}{2\sigma_y^2} \right) \right\}, \tag{6}$$

where (x_0, y_0) is the initial location of the center of the pulse, σ_x and σ_y define the support of the pulse in the x and y directions, respectively. The domain is uniformly discretized into 201 grid points in the x-direction and 501 grid points in the y-direction using $\Delta x = 1$ and $\Delta y = 1$, respectively, and generating 100,701 computational nodes. A uniform time discretization of $\Delta t = 1$ is used to generate 460 high-fidelity time snapshots for a circular Gaussian pulse parametrized by different values of the size of the pulse profile $\sigma_x = \sigma_y \equiv \sigma = \{5, 8, 10, 16, 20\}$, and traveling at a constant speed $c = 1$ from an initial location $(x_0, y_0) = (0, 40)$. Figure 3 depicts the relative information content (RIC) for a different number of POD modes obtained by taking a SVD of the high-fidelity snapshots. As the singular values computed by SVD are arranged in the descending order of relative importance, the RIC values of the leading r POD modes can be defined as

$$\text{RIC}(\%) = \frac{\sum_{k=1}^{r} \lambda_k^2}{\sum_{k=1}^{M} \lambda_k^2} \times 100, \tag{7}$$

where λ_k is the kth singular value and M denotes the total number of time snapshots. For the set of snapshots generated with a given value of σ, the dotted vertical line indicates the number of leading POD modes required to attain a RIC value of 99.9%. For instance, $\sigma = 20$ signifies a flatter pulse profile and 17 POD modes contain 99.9% RIC for the corresponding system of snapshots, whereas 63 POD modes are needed to capture 99.9% RIC for a sharper pulse profile given by $\sigma = 5$. This illustrates the phenomena of relatively large Kolmogorov n-widths for even simple, linear advection problems, which severely limits the efficiency of low-dimensional approximation using SVD-generated linear subspaces.

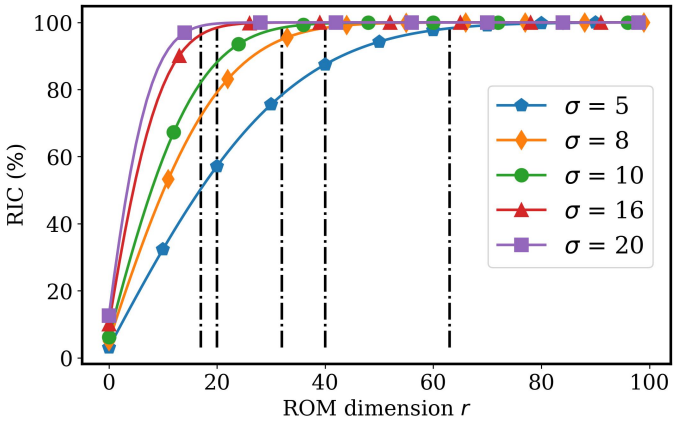


Figure 3. The relative information content for different number of retained POD modes. The singular values are computed by taking an SVD of the high-fidelity snapshots for an advecting circular Gaussian pulse (see Equation (6)) of varying width (σ) traveling at a constant speed $c = 1$.

In the first numerical example, the training dataset is constructed using 460 high-fidelity snapshots for each value of $\sigma_{train} = \{5, 10, 16\}$. The remaining snapshots corresponding to $\sigma_{test} = \{8, 20\}$ are used to create a test dataset. This creates a geometrically parameterized training and testing dataset.

The AA autoencoder network is trained on the parametric training set for 8000 epochs using the Adam optimizer with an initial learning rate of 1×10^{-4} that decays step-wise by 15% every 456 epochs. The training snapshots are all augmented by the value of the corresponding parameter. The training snapshots are divided into two sets—starting from the initial time point every alternate snapshot is used for training the AA autoencoder model, while the rest are reserved for validation during training. In this study, the losses computed on the validation points are solely used to monitor the extent of overfitting during training, and later to evaluate the accuracy of prediction on unseen time steps associated with a training parameter value. After exploring a large space of network design parameters, as described in Table 1, the results presented here are obtained with two of the most optimal AA autoencoder designs. In the first model (AA1), only the input feature is augmented by the parameter value; while in the second model (AA2), both the input feature and the output labels are augmented by the parameter value. The encoder network χ_e of both the models is constructed with three hidden layers composed of 629, 251 and 62 units that connect an input feature (i.e., augmented snapshot) of dimension $N = 100,702$ to an encoded latent vector representation of dimension k. For both the models, the decoder networks χ_d and ϕ_s are set up to be mirror images of the corresponding encoder network. The AA1 model uses the *selu* activation function for the hidden layers followed by a *linear* activation on the output layer, while the AA2 model uses the *swish* activation function for the hidden layers. The individual loss components $\mathcal{L}_1$ and $\mathcal{L}_2$ are defined as a weighted combination of the normalized mean square error (NMSE) loss and the pseudo-Huber loss, as defined below,

$$
\begin{aligned}
\text{NMSE Losses: } \mathcal{L}_1^{NMSE} &= \frac{1}{N}\sum_{k=1}^{N}\frac{\left\|\mathbf{v}_{k,s}-\boldsymbol{\phi}_s\circ\chi_e(\mathbf{v}_k)\right\|_2^2}{\|\mathbf{v}_k\|_2^2},\\
\mathcal{L}_2^{NMSE} &= \frac{1}{N}\sum_{k=1}^{N}\frac{\left\|\mathbf{v}_{k}-\chi_d\circ\chi_e(\mathbf{v}_k)\right\|_2^2}{\|\mathbf{v}_k\|_2^2},\\
\text{pseudo-Huber Losses: } \mathcal{L}_1^{H} &= \frac{1}{N}\sum_{k=1}^{N}\delta^2\left(\sqrt{1+\left(\frac{\mathbf{a}_{k,1}}{\delta}\right)^2}-1\right),\\
\mathcal{L}_2^{H} &= \frac{1}{N}\sum_{k=1}^{N}\delta^2\left(\sqrt{1+\left(\frac{\mathbf{a}_{k,2}}{\delta}\right)^2}-1\right),\\
\text{where}\quad \mathbf{a}_{k,1} &= \mathbf{v}_{k,s}-\boldsymbol{\phi}_s\circ\chi_e(\mathbf{v}_k) \text{ and } \mathbf{a}_{k,2}=\mathbf{v}_k-\chi_d\circ\chi_e(\mathbf{v}_k).
\end{aligned}
\tag{8}
$$

Table 1. Hyperparameters explored to design the AA autoencoders for the linear advection example.

Hyperparameters	AA1	AA2
Input/Output	Augmented Input, non-augmented output	Augmented input and output
Hidden Units (50–1500)	629, 251, 62	629, 251, 62
Batch Size (8–128)	32	24
Latent Dimension (5–50)	15	15
Activation (*ReLU*, *selu*, *linear*, *tanh*, *swish*)	*selu*	*swish*

The pseudo-Huber loss is a smooth approximation of the Huber loss function that behaves as a L_2 squared loss by being strongly convex near the desired minimum and as a L_1 absolute loss with reduced steepness near the extreme values. The scale at which this transition happens and the steepness near the extreme values is controlled by the δ parameter.

A piece-wise segmented training approach is adopted for both the models in which only the $\mathcal{L}_2$ component of the total loss is minimized for the first 2500 epochs, followed by a weighted combination of both the loss components $w_1\mathcal{L}_1 + w_2\mathcal{L}_2$ for the rest of the training. The AA1 model is trained with mini batches of size 32 and the AA2 model is trained with mini batches of size 24, while both models generate a latent space of dimension 15.

The training trajectories for the AA1 and AA2 models are shown in Figure 4. Even though both models are trained for the same number of epochs, the lower training and validation loss values for the AA2 model (see the left panel of Figure 4) indicates a higher level of expressivity and overfitting due to the augmented dimension of the decoder outputs and the resultant higher number of network hyperparameters (weights and biases). As a result, the prediction errors for the test parameter values are found to be higher using the AA2 model than those obtained with the AA1 model. Less overfitting is usually an indication of better generalization performance, and hence the AA1 model is used to generate the field predictions for both the seen and unseen data in the rest of this example. On the other hand, when extrapolatory predictions or predictions for unseen data are not required, a slightly overfit model such as AA2 can be considered preferable. This is supported by the evolution of the losses corresponding to each decoder: $\mathcal{L}_1$ for the prediction of shifted snapshots and $\mathcal{L}_2$ for the reconstruction of the true solution (see the right panel of Figure 4). As the $\mathcal{L}_1$ loss is associated with the network's ability to map the true snapshot to a fixed snapshot, it is relatively easier to minimize and both models perform equally well in this task. However, due to the higher expressivity of the AA2 model, it is able to minimize the $\mathcal{L}_2$ loss much more than the AA1 model, thus leading to higher accuracy in the approximation of the true snapshots using training data.

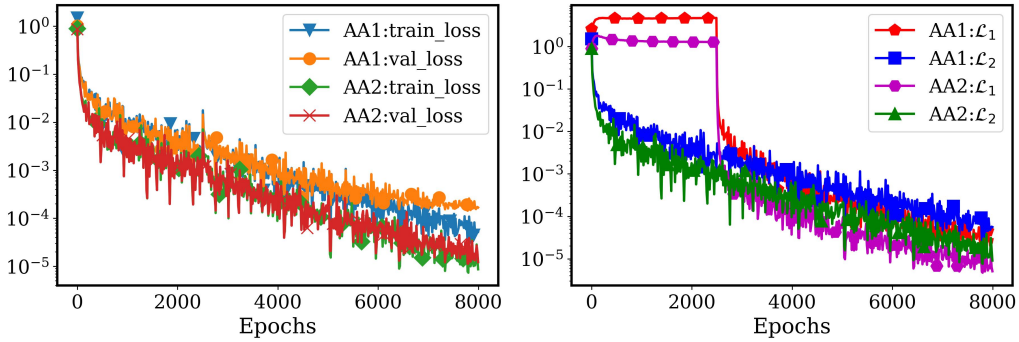
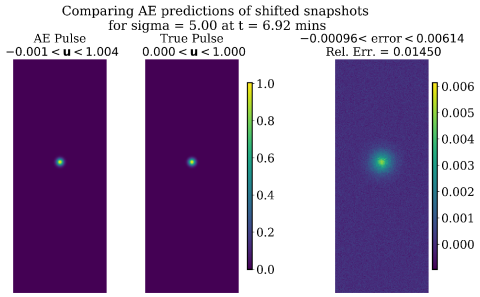


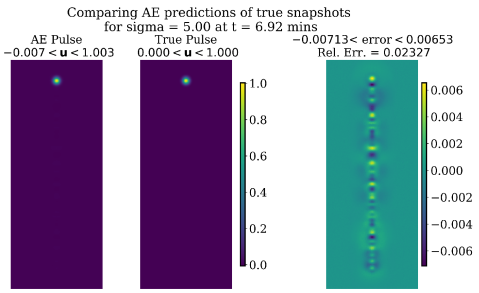
Figure 4. Training characteristics of two AA autoencoder networks trained using a parametric dataset of snapshots for a 1D advecting Gaussian pulse parameterized with varying support of the pulse profile, $\sigma = \{5, 10, 16\}$. AA1 denotes the model trained with the input features augmented by parameter values, while AA2 denotes the model where both input features and output labels are augmented. The left panel shows the decay of training and validation losses during training. The right panel shows the evolution of the loss components during training.

Figure 5 presents the predictions of the high-dimensional shifted and true snapshots obtained by the corresponding decoders $\boldsymbol{\phi}_s$ and χ_d, respectively, of model AA1 as well as a comparison of the prediction performance for different training parameter values in terms of the spatial relative errors of the full-order predictions. The decoder predictions are evaluated for the two parameter values at the boundaries of the training range $\sigma = 5$ and $\sigma = 16$ as they present distinct challenges. Autoencoders are known to struggle with the extraction of discontinuities in the input feature space. The snapshot data for $\sigma = 5$ features a very steep gradient in the shape of the pulse profile which poses some of the same challenges as a discontinuous profile. Moreover, a single encoder network χ_e is being tasked, by design, to combine with two independent decoder networks χ_d and $\boldsymbol{\phi}_s$ to map into both a stationary discontinuity as well as a moving discontinuity. Thus the spatial distribution of reconstruction error for the $\sigma = 5$ profile is more localized near the moving pulse, whereas that of the $\sigma = 16$ profile is more uniformly spread out across the spatial domain (see Figure 5b,d. Despite all of these minor differences in prediction performance, there is a high degree of agreement between the full order decoder predictions and the high-dimensional snapshots, with less than 4% relative error for all of the training parameter values (see Figure 5e).

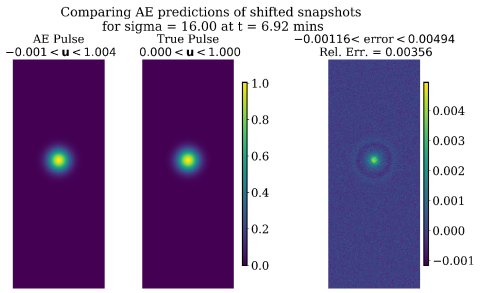
Finally, prediction performance results of the AA1 model for high-fidelity snapshots generated with an unseen parameter value $\sigma = 8 \in \sigma_{test}$ are presented in Figure 6. Loss in accuracy with extrapolatory predictions for a geometrically parameterized dataset is one of the well-known challenges faced by both intrusive and non-intrusive reduced order modeling approaches, which requires particular attention to resolve representation issues posed by the topology of the parametric solution manifold [75]. Thus, as expected, there is a noticeable drop in accuracy for the prediction of full order solutions, with the errors being especially localized near the moving pulse profile (see Figure 6b). This effect is also reflected in the relative error plots for the two unseen parameter values in σ_{test}. However, the quality of predictions are still quite encouraging considering that these are purely extrapolatory predictions on an unseen parameter instance, without any special treatment of the solution manifold or modification of the training process.



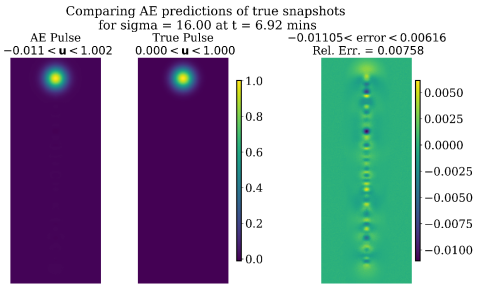
(**a**) Prediction of shifted snapshot



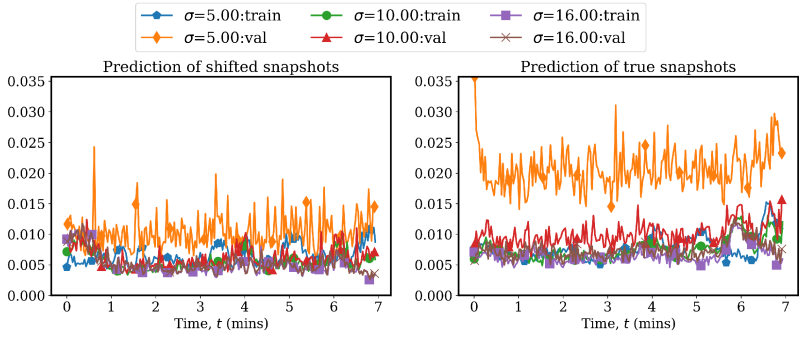
(**b**) Prediction of true snapshot



(**c**) Prediction of shifted snapshot

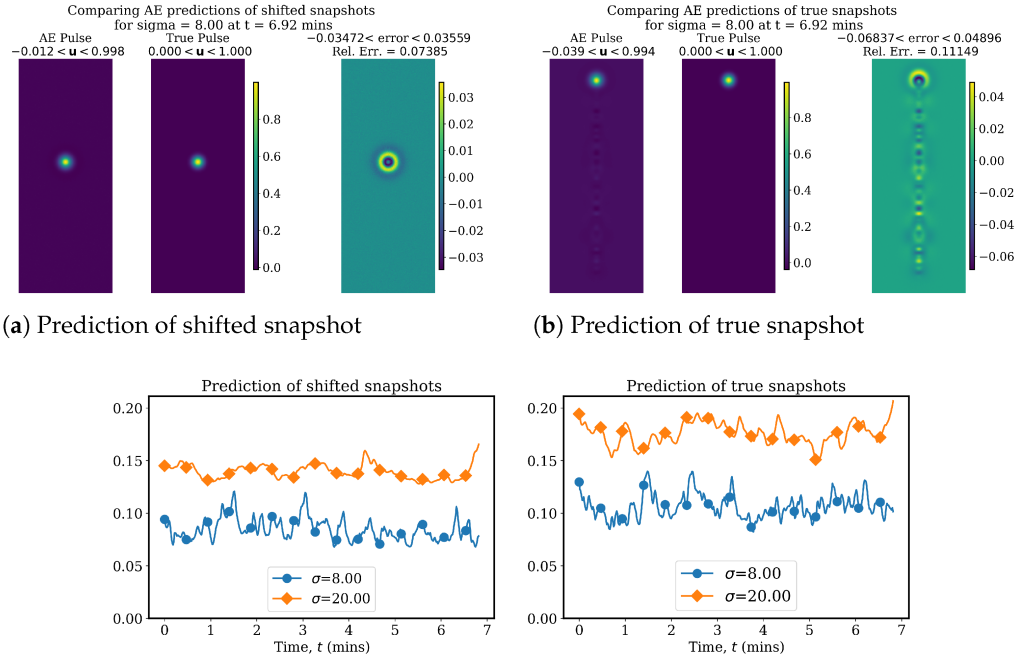


(**d**) Prediction of true snapshot



(**e**) Relative error plots

Figure 5. Prediction performance of ϕ_s and χ_d decoders on training data. (**a**,**b**) predictions of shifted and true snapshots, respectively, for pulse size $\sigma = 5$ and (**c**,**d**) predictions of shifted and true snapshots, respectively, for pulse size $\sigma = 16$ at an intermediate time $t = 6.92$ min using the AA1 model. (**e**) Relative errors for the decoder predictions using different values of the parameter from the training set.



(**a**) Prediction of shifted snapshot (**b**) Prediction of true snapshot

(**c**) Relative error plots

Figure 6. Prediction performance of ϕ_s and χ_d decoders on unseen data. (**a**,**b**) predictions of shifted and true snapshots, respectively, for unseen pulse size $\sigma = 8$ at time $t = 6.92$ min using the AA1 model. (**c**) Relative errors for the decoder predictions using two different values of the parameter from the unseen test set.

4.2. Advecting Viscous Shock Problem

The second numerical example is described by the one-dimensional viscous Burgers' equation (VBE) with Dirichlet boundary conditions [60] as given by

$$\begin{aligned} \dot{u} + u\frac{\partial u}{\partial x} &= \nu \frac{\partial^2 u}{\partial x^2}, \\ u(x,0) = u_0, \qquad x \in [0,L], \qquad u(0,t) &= u(L,t) = 0, \end{aligned} \tag{9}$$

where we set $L = 1$ and the maximum time $t_{max} = 2$. The solution of the above equation is capable of generating shock discontinuities even with smooth initial conditions if the viscosity ν is sufficiently small, due to the advection-dominated behavior. We consider the initial condition

$$u(x,0) \equiv u_0 = \frac{x}{1 + \sqrt{\frac{1}{t_0}}\exp\left(Re\frac{x^2}{4}\right)}. \tag{10}$$

An analytical solution of this problem is given by

$$u(x,t) = \frac{\frac{x}{t+1}}{1 + \sqrt{\frac{t+1}{t_0}}\exp\left(Re\frac{x^2}{4t+4}\right)}, \tag{11}$$

where $t_0 = \exp(Re/8)$ and $Re = 1/\nu$. The high-fidelity snapshot data is generated by directly evaluating the analytical solution over a uniformly discretized spatial domain containing 200 grid points and for 500 uniform time steps. Figure 7 shows a visualization of the time evolution of the initial condition for three different values of $Re = 50, 300, 600$.

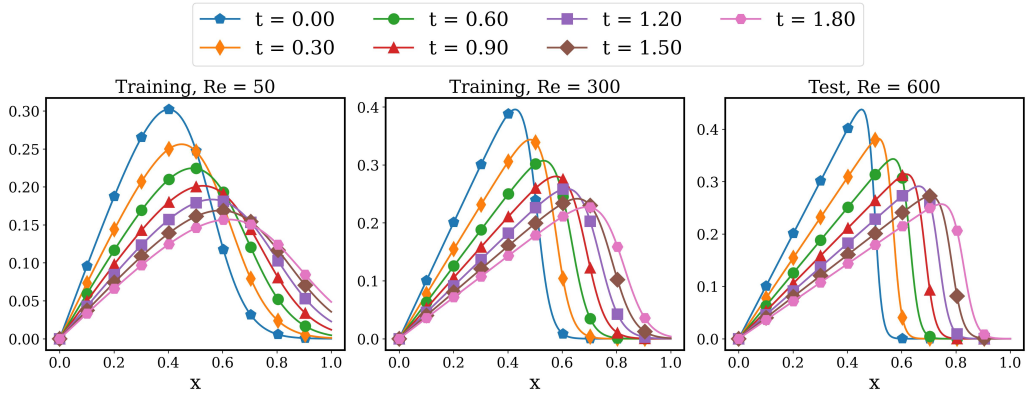


Figure 7. Time evolution of the high-fidelity snapshots for the advecting viscous shock problem (see Equation (9)), parameterized with variable Reynolds number, *Re*.

A parametric dataset is generated by collecting 500 high-fidelity snapshots for different values of the Reynolds number, $Re = \{50, 150, 300, 400, 500, 600\}$. The training dataset is constructed using snapshots for $Re_{train} = \{50, 150, 300, 500\}$. The remaining snapshots for $Re_{test} = \{400, 600\}$ constitute the test dataset. Figure 8 depicts the variation in RIC with the number of retained POD modes for snapshots corresponding to different *Re* values. The vertical dashed lines represent the number of POD modes required to attain 99.9% RIC for snapshots of a given *Re* value. The gradual rise in the number of POD modes required to attain 99.9% RIC with increasing values of *Re* clearly indicates how a growth of advection-dominated behavior raises the effective Kolmogorov n-width of the system.

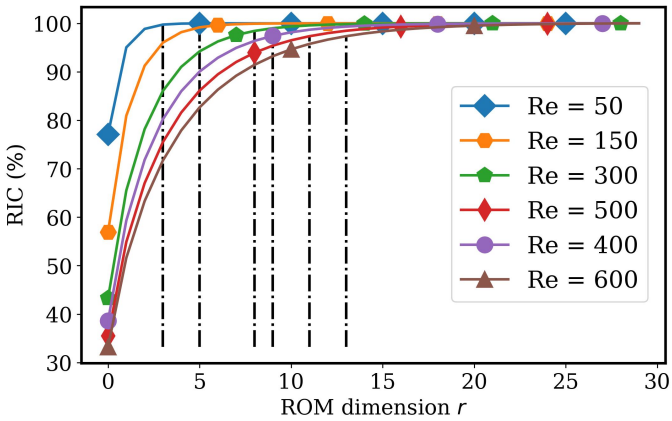


Figure 8. The relative information content for a different number of retained POD modes. The singular values are computed by taking an SVD of the high-fidelity snapshots for the advecting viscous shock problem (see Equation (9)), parameterized with variable Reynolds number, *Re*.

4.2.1. AA Autoencoder Models for Varying Advection Strength

In this section, we present the numerical results on the training of AA autoencoder networks for the viscous advecting shock problem parameterized with variable *Re* values, as discussed before. Following the idea of a registration-type approach, as discussed in Section 3.2, a high-fidelity simulation snapshot at roughly the midpoint of the simulation time period is chosen as the shifted snapshot for training the shift decoder. This choice is, however, arbitrary and any other high-fidelity snapshot could have been selected without affecting the effectiveness of the approach.

The results reported here are obtained with two different AA autoencoder models—AA3 and AA4. The primary objective of this comparison is to evaluate the ability of AA autoencoders not just to predict snapshots for unseen parameter values, but also to forecast solutions at time points not included in the time history of the training snapshots. With

that objective in mind, the AA3 model is trained using all of the time snapshots available for each training parameter value, while the AA4 model is trained using the first 90% of the time snapshots, i.e., until $t = 1.80$, for each training parameter value. Similar to the previous numerical example, the available high-fidelity snapshots for each training parameter value are divided into two sets—starting from the initial time point every alternate snapshot is used for training the AA autoencoder model, while the rest are reserved for validation purposes. As in the previous example, the losses computed on the validation data points during training are solely used to monitor the extent of overfitting, and later to evaluate the accuracy of prediction on unseen data points corresponding to a training parameter value.

After a careful exploration of the design space, a set of optimal values for the hyperparameters were obtained to construct models AA3 and AA4 (see Table 2). Both the models are trained for 5000 epochs using minibatches of size 24 and employing the Adam optimizer. The AA3 model training is initialized with a learning rate of 5×10^{-4} that decays stepwise by 10% every 330 epochs, whereas the initial learning rate for the AA4 model is chosen to be 3×10^{-4} and it is allowed to decay by 10% every 309 epochs. Both the input features and the output labels are constructed by augmenting the training snapshots with the corresponding scaled parameter values. The encoder network χ_e for model AA3 is constructed with a single hidden layer of size 50 that connects an input feature (i.e., augmented snapshot) of dimension $N = 201$ to an encoded latent vector representation of dimension k. On the other hand, the AA4 model is defined with two hidden layers of sizes 100 and 50. For both the models, the decoder networks χ_d and ϕ_s are set up to be mirror images of the encoder network. From Figure 8 it can be seen that at least a minimum of 3 POD modes are required to attain 99% RIC for any of the chosen *Re* values. However, while exploring a range of possible latent space dimensions, $3 \leq k \leq 10$, it was observed that a latent space of dimension $k = 5$ was adequate in capturing the essential dynamical features of the entire parametric training dataset. Hence, $k = 5$ is selected as the optimal latent space dimension for both models AA3 and AA4. All hidden layers are endowed with the *swish* activation function, while the output layers are designed to have a *linear* activation. The individual loss components $\mathcal{L}_1$ and $\mathcal{L}_2$ are defined by a weighted combination of the NMSE loss and the pseudo-Huber loss, as discussed in the previous numerical example.

Table 2. Hyperparameters to design the AA autoencoders for the viscous advecting shock example.

Hyperparameters	AA3	AA4
Input/Output	Augmented input and output	Augmented input and output
Hidden Units (50–150)	50	100, 50
Batch Size (8–128)	24	24
Latent Dimension (3–10)	5	5
Activation (*selu, tanh, swish*)	*swish*	*swish*
Initial Learning Rate (1×10^{-3}–1×10^{-5})	5×10^{-4}	3×10^{-4}

Figure 9 shows the salient features of the training process for models AA3 and AA4. The left plot shows the decay of the training and validation losses during training, and the right plot shows the decay of the two loss components, $\mathcal{L}_1$ and $\mathcal{L}_2$, for both models. Due to its higher capacity (more hidden layers and more neurons) model AA4 is capable of attaining lower values of training and validation losses as compared to model AA3. This is possibly an indication that model AA4 is able to learn the essential features of the high-dimensional state space more effectively, thus enabling improved prediction over data points that lie within the bounds of the training time history, as will be shown in the later experiments. On the other hand, this also causes the $\mathcal{L}_1$ loss component of model AA4 to have a sharper decay than the $\mathcal{L}_2$ component, whereas model AA3 shows a more balanced decay of the two loss components. The latter trait is considered more preferable, as the effectiveness of any latent space dynamics model is dependent upon the accuracy of the true decoder χ_d, that is measured by the $\mathcal{L}_2$ loss component. Therefore, in situations when the

entire time history is available for model training, a smaller capacity AA autoencoder model like AA3 is capable of achieving the desirable training outcomes. Hence, following the principle of parsimony, model AA3 is chosen to generate the latent space representations that are used to train LSTM dynamics models in the next two sections.

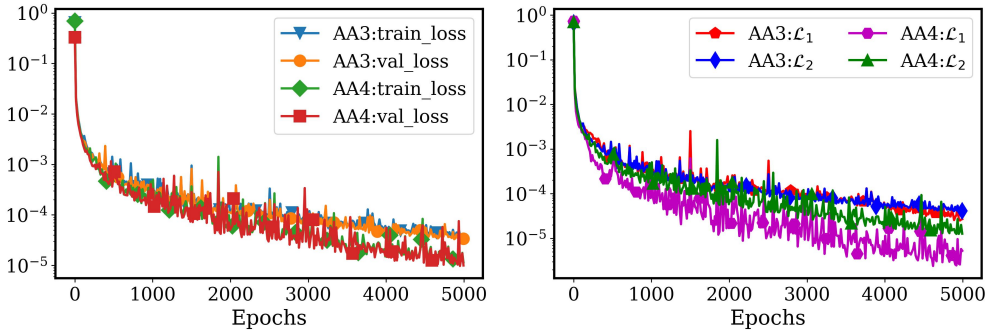
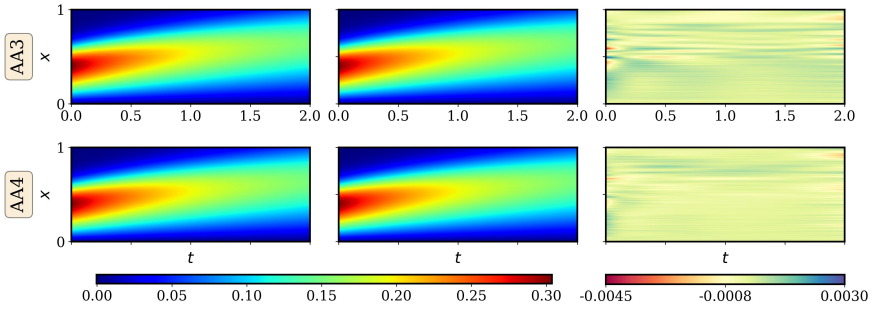


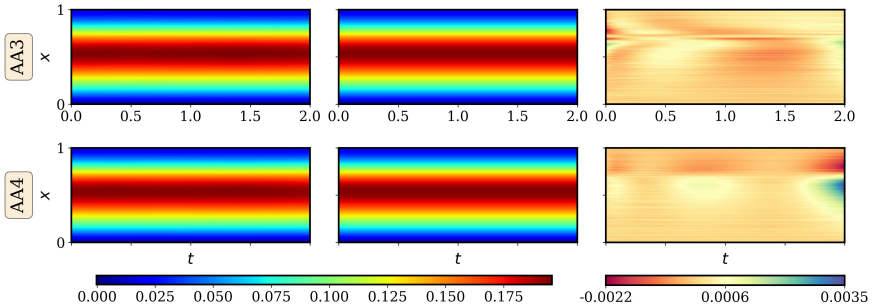
Figure 9. Training characteristics of two AA autoencoder networks trained using a parametric dataset of snapshots for the advecting viscous shock problem parameterized with variable Reynolds number, $Re = \{50, 150, 300, 500\}$. AA3 denotes the model trained with the entire time snapshot history for every parameter value, while AA4 denotes the model trained with the initial 90% of the time snapshot history for each parameter value. The left panel shows the decay of training and validation losses during training. The right panel shows the evolution of the loss components during training.

Figures 10 and 11 present a performance comparison of models AA3 and AA4 in predicting the true and shifted snapshots for two of the training parameter values, $Re = 50$ and $Re = 500$, respectively. Each individual plot shows the evolution of either a solution field or an error field in the $x - t$ space, where the x-axis represents time and the y-axis represents space. The plots in the left column depict the solution fields predicted by the AA3 and AA4 models, the plots in the middle column depict the high-dimensional solution fields, and the plots in the right column depict the pointwise error between the high-dimensional and the predicted solution fields. It is evident from the first two columns that models AA3 and AA4 are able to qualitatively capture both the true and shifted solution fields. A closer look at the right column reveals that the approximation error of model AA3 is randomly spread throughout the simulation time history (see Figures 10b and 11a,b, whereas the approximation error of model AA4 rises gradually as predictions are sought further away from the end point of the training time history, i.e., $t > 1.8$. This confirms the previously discussed observation that, due to the higher network capacity, model AA4 offers more accurate predictions than model AA3 for time points that lie within the bounds of the training time history that is common to both models, i.e., $0 < t \leq 1.8$. However, model AA4 gradually loses predictive capability over the unseen time points given by $t > 1.8$, whereas model AA3 still generates accurate predictions, despite its lower network capacity.

Figure 12 depicts the time trajectory of the spatial relative errors for the predictions obtained by models AA3 and AA4 over the parametric training dataset. The spatial relative errors are computed as the ratio of the spatial l^2-norm of the prediction error to the spatial l^2-norm of the high-dimensional solution at every computational time point. Figure 12a shows the spatial relative errors for the shift decoder predictions while Figure 12b shows the corresponding errors for the true decoder predictions, for every value in the parametric training dataset, $Re_{train} = \{50, 150, 300, 500\}$. The relative error plots not only validate the previously discussed observations about the prediction capabilities of models AA3 and AA4, but also highlight that even for $t > 1.8$, model AA4 offers encouraging extrapolatory predictions with a relative error of less than 4%.

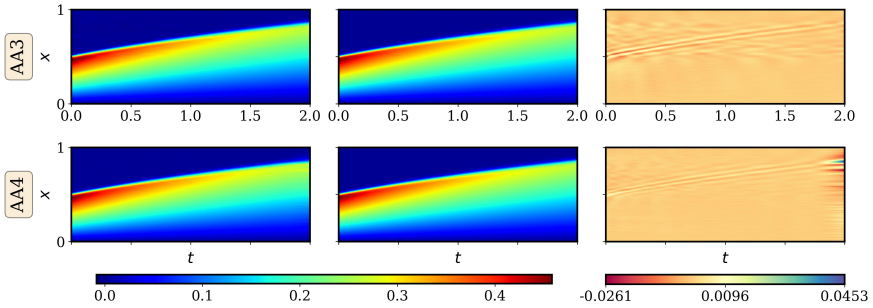


(**a**) Prediction of true snapshots for $Re = 50$ using models AA3 and AA4

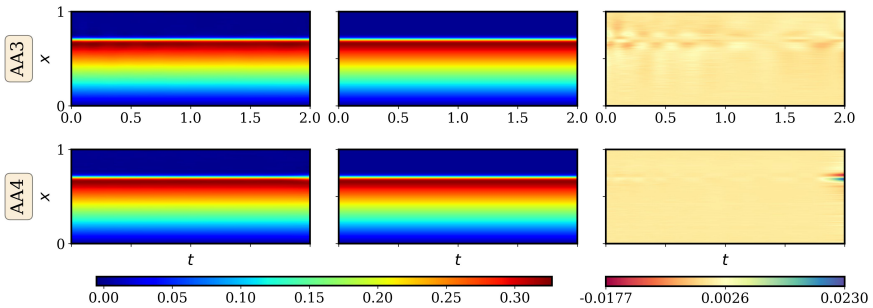


(**b**) Prediction of shifted snapshots for $Re = 50$ using models AA3 and AA4

Figure 10. Prediction performance of χ_d and ϕ_s decoders on training data. Predictions of (**a**) true and (**b**) shifted snapshots for a training parameter value, $Re = 50$, using models AA3 and AA4. The left column shows the predicted solutions, the center column shows the high-fidelity solutions, and the right column shows the error between the two.

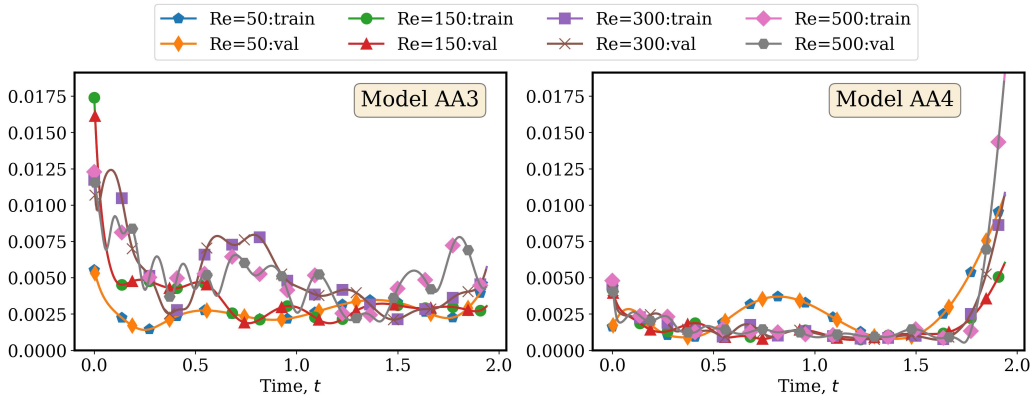


(**a**) Prediction of the true snapshots for $Re = 500$ using models AA3 and AA4

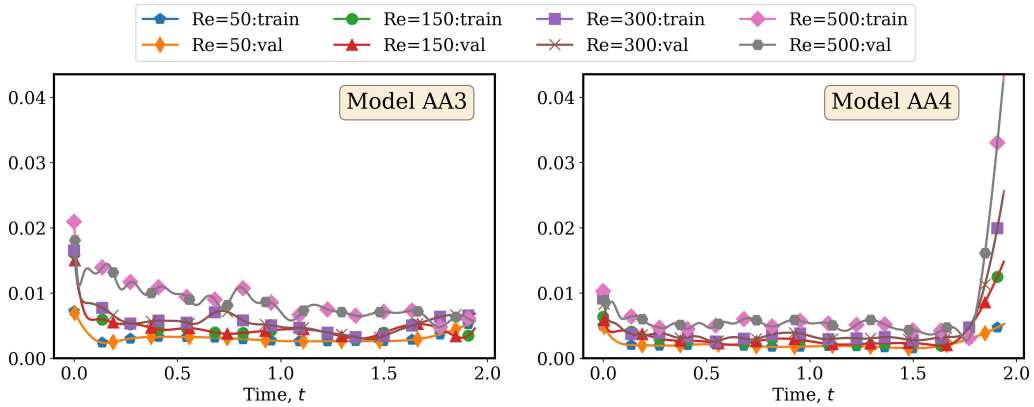


(**b**) Prediction of the shifted snapshots for $Re = 500$ using models AA3 and AA4

Figure 11. Prediction performance of χ_d and ϕ_s decoders on training data. Predictions of (**a**) true and (**b**) shifted snapshots for a training parameter value, $Re = 500$, using models AA3 and AA4. The left column shows the predicted solutions, the center column shows the high-fidelity solutions and the right column shows the error between the two.



(**a**) Relative errors of the shift decoder predictions



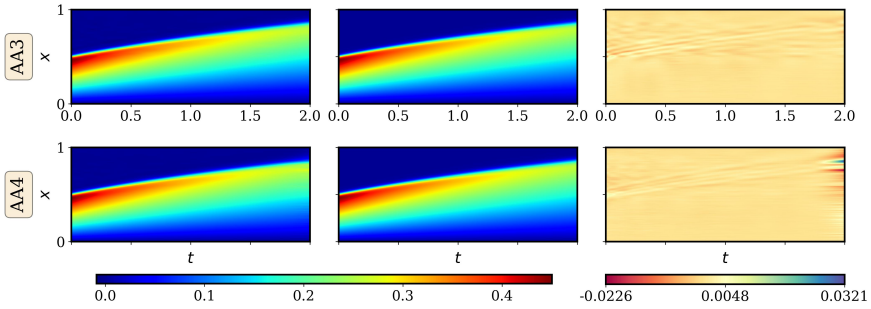
(**b**) Relative errors of the true decoder predictions

Figure 12. Relative errors of (**a**) ϕ_s and (**b**) χ_d predictions using the AA3 and AA4 models for snapshots generated with the training parameter values.

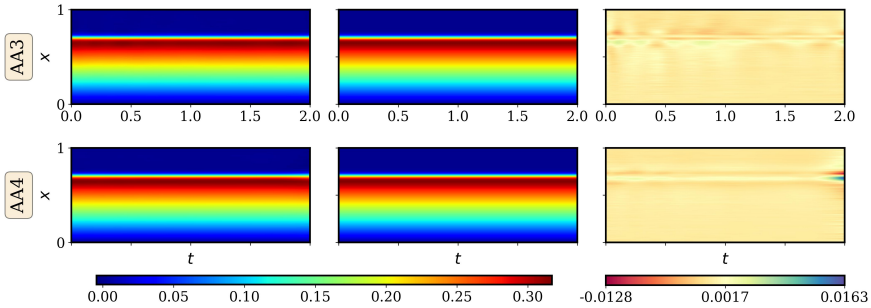
Figures 13 and 14 show the performance of models AA3 and AA4 while predicting the true and shifted snapshots using parameter values from the test dataset, $Re = 400$ and $Re = 600$, respectively. Figure 15 presents the spatial relative errors of the predictions made by models AA3 and AA4 for the two test parameter values. Even for the unseen test parameter values, the true and shifted solution fields computed by the AA autoencoder models are closely aligned with the high-dimensional solution fields. This is reflected in the error field plots as well as the spatial relative error plots, which are bounded below the 5% relative error. The results in this section demonstrate that even for a nonlinear advection-dominated problem, a trained AA autoencoder network can offer accurate extrapolatory predictions for unseen parameter instances as well as short-term extrapolatory predictions for unseen time.

4.2.2. LSTM Models for System Dynamics

In this section, numerical results are presented for the modeling of the temporal evolution of the latent space coefficients defined by a pre-trained AA autoencoder network for the advective viscous shock problem parametrized by variable Re. As the focus of this work is to demonstrate the efficiency and flexibility of the AA autoencoder architecture, hence for the sake of simplicity, the latent space dynamics are modeled in an autoregressive fashion using traditional LSTM networks.

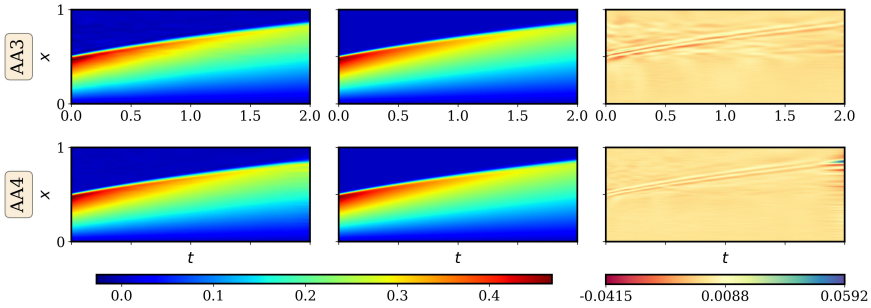


(**a**) Prediction of true snapshots for $Re = 400$ using models AA3 and AA4

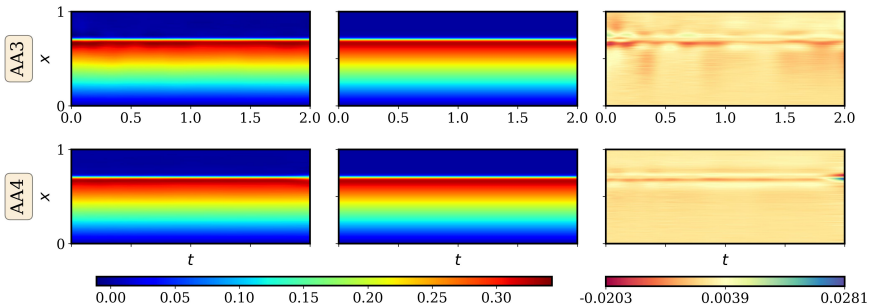


(**b**) Prediction of shifted snapshots for $Re = 400$ using models AA3 and AA4

Figure 13. Prediction performance of χ_d and ϕ_s decoders on unseen data. Predictions of (**a**) true and (**b**) shifted snapshots for a test parameter value, $Re = 400$, using models AA3 and AA4. The left column shows the predicted solutions, the center column shows the high-fidelity solutions and the right column shows the error between the two.



(**a**) Prediction of true snapshots for $Re = 600$ using models AA3 and AA4



(**b**) Prediction of shifted snapshots for $Re = 600$ using models AA3 and AA4

Figure 14. Prediction performance of χ_d and ϕ_s decoders on unseen data. Predictions of (**a**) true and (**b**) shifted snapshots for a test parameter value, $Re = 600$, using models AA3 and AA4. The left column shows the predicted solutions, the center column shows the high-fidelity solutions, and the right column shows the error between the two.

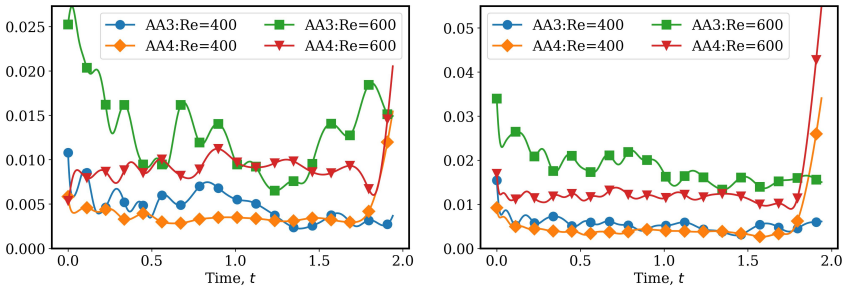
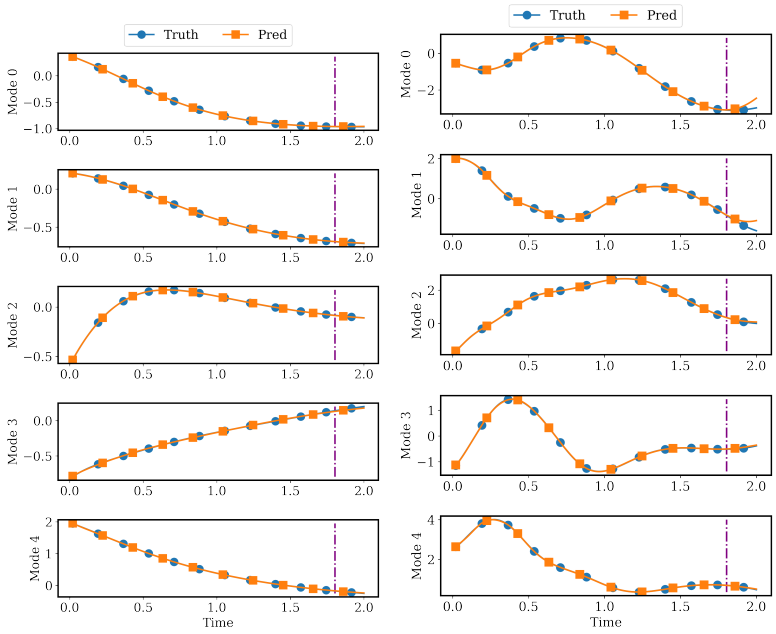


Figure 15. Relative errors of ϕ_s (left) and χ_d (right) predictions using the AA3 and AA4 models for snapshots generated with the unseen test parameter values.

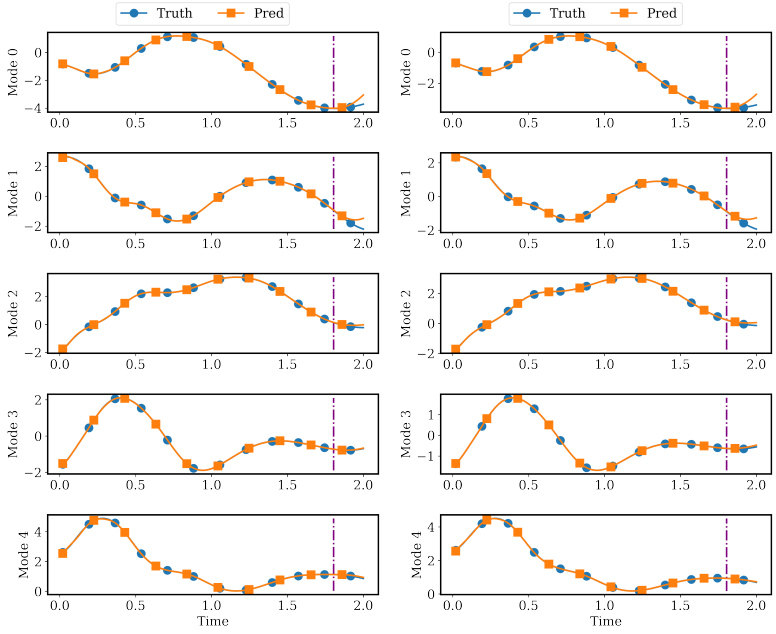
Two different approaches are adopted to construct these dynamics models. In the first method, an independent LSTM network is trained for the encoded snapshots corresponding to each parameter value in $Re_{train} = \{50, 150, 300, 500\}$. The AA3 model is chosen to compute the encoded latent representations of the high-dimensional snapshots. For the datasets characterized by weaker advection, i.e., $Re = 50$ and $Re = 150$, a smaller capacity LSTM network is defined using two stacked LSTM cells with 32 hidden dimensions each and *swish* activation. For the datasets with higher values of $Re = 300$ and $Re = 500$, a higher capacity network consisting of two stacked LSTM cells with 150 hidden units was found to be necessary to accurately capture the dynamics. The network is trained to read an input consisting of 5 time steps and predict the next element in the time series. The first 90% time steps are used for training the LSTM model and the remaining 10% are used for testing the extrapolatory predictive capability of the trained LSTM model. Training is performed for 4000 epochs using the Adam optimizer with minibatches of size 24 and with an initial learning rate of 2×10^{-5} that decays by 10% every 304 epochs. For minimization of the network hyperparameters, losses in the latent space predictions are computed using the NMSE loss. Scaling the input features used for training the LSTM model was found to offer no additional benefits to the training process or improved prediction accuracy. Some preliminary testing with the use of dropout layers also yielded inconclusive evidence to support or recommend their use for further training.

In Figure 16, the latent space coefficients of the high-dimensional snapshots as defined by the AA3 autoencoder model are compared with the predictions generated by the individual LSTM models. As the AA3 model defines a latent space of dimension 5, hence each plot depicts 5 latent space modes. The plots in panels (**a**)–(**c**) are obtained with LSTM models that are trained on the latent space coefficients corresponding to snapshots in the parametric training dataset, i.e., $Re = 50, 300, 500$, respectively. The modal trajectories in panel (**d**) are obtained by evaluating the latent space coefficients for a test parameter value $Re = 400$ using the AA3 model and then training a LSTM model to learn the evolution of these coefficients. As mentioned before, the LSTM models are trained on the first 90% time steps of each timeseries and the boundary of the training data is marked by a dashed vertical line in each plot. The encoded true snapshots and the LSTM predictions for both the training and even the test parameter values display a high degree of agreement, especially for time steps within the LSTM training time window. It is also encouraging to observe that for a short length outside the training time window, even the extrapolatory predictions obtained from the trained LSTM models are in agreement with the encoded true snapshots. This behavior can also be seen in Figure 17 where the true decoder of the AA3 model is applied to the LSTM predictions and the results are compared with the high-dimensional snapshots. These plots are populated with the predicted and high-dimensional solution snapshots at four different intermediate times with the x-axis representing the spatial grid. The plotting time steps are distributed uniformly throughout the simulation time window and are chosen in such a way that the final time step $t = 1.90$ lies outside the LSTM training time window. It is clear that the trained autoencoder(AE)-LSTM model captures the viscous advecting shock-like feature fairly well, even for the extrapolatory time step. This demonstrates the ease of constructing dynamics models in a latent space defined by

the parametric AA autoencoder model, even while adopting a standard implementation of a simple and lightweight LSTM network. While some discrepancies emerge with extrapolatory and longer-time prediction windows, this can be attributed to the well-known issues with the autoregressive modeling of time series data using standard LSTM networks [70]. However, for applications where time series predictions are desired over shorter time windows, the proposed AA autoencoder+LSTM approach shows the capacity for effective extrapolatory predictions.

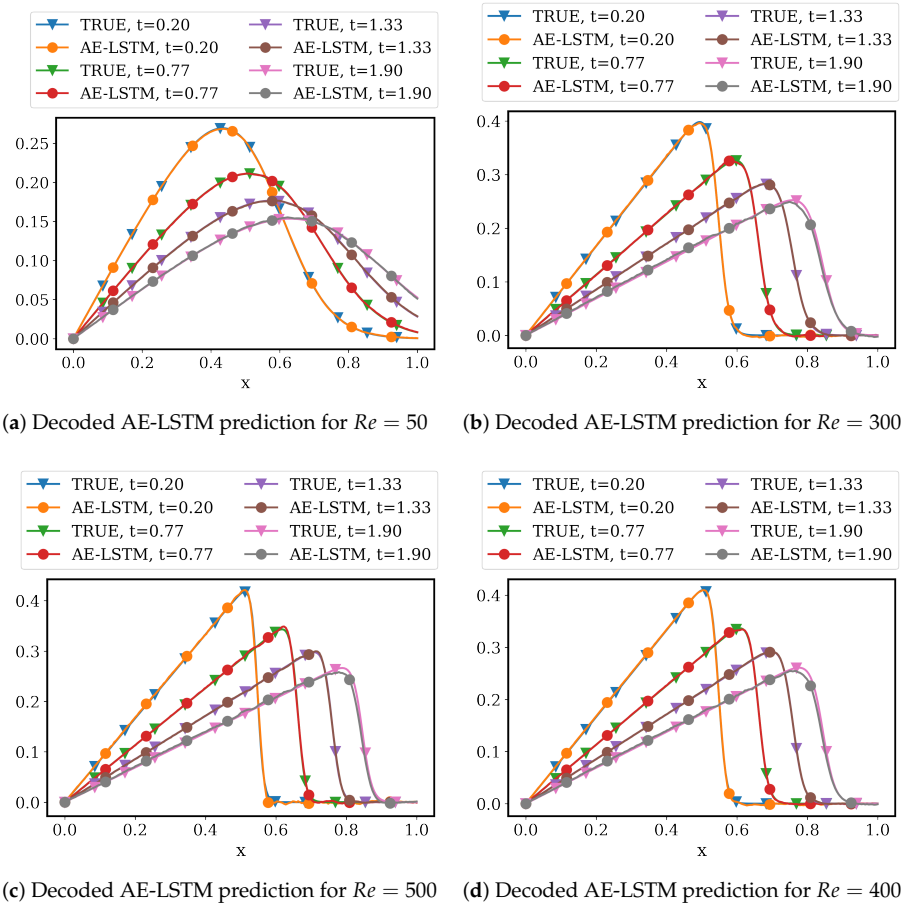


(**a**) LSTM prediction for $Re = 50$ (**b**) LSTM prediction for $Re = 300$



(**c**) LSTM prediction for $Re = 500$ (**d**) LSTM prediction for $Re = 400$

Figure 16. Comparing latent space predictions obtained using a parametric AA autoencoder and LSTM models. The LSTM models are trained separately for each training parameter value as presented in (**a**) $Re = 50$, (**b**) $Re = 300$, (**c**) $Re = 500$ and for a test parameter value shown in (**d**) $Re = 400$. All LSTM models are trained using the first 90% of the total time steps (as demarcated by the vertical lines in each figure), and the remaining time steps are used for evaluating extrapolatory predictions.



(**a**) Decoded AE-LSTM prediction for $Re = 50$ (**b**) Decoded AE-LSTM prediction for $Re = 300$

(**c**) Decoded AE-LSTM prediction for $Re = 500$ (**d**) Decoded AE-LSTM prediction for $Re = 400$

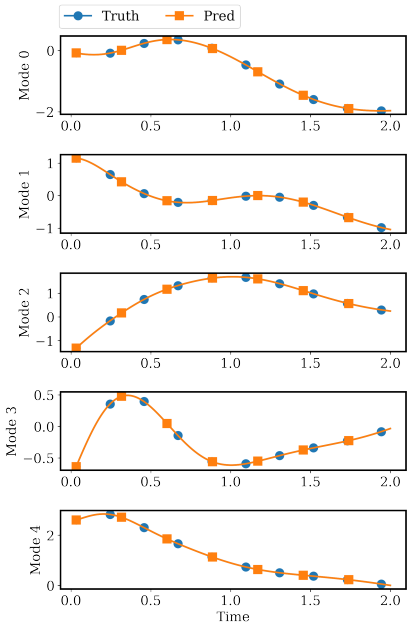
Figure 17. Comparing predictions of the high-dimensional solutions for the parametric advecting viscous shock problem using an AA autoencoder and LSTM models. The LSTM models are individually trained for each parameter value and are presented as (**a**) $Re = 50$, (**b**) $Re = 300$, (**c**) $Re = 500$, and (**d**) $Re = 400$.

In the second approach, a parametric LSTM (pLSTM) network is trained on the encoded snapshots obtained by the AA3 model. An encoded representation is obtained for every snapshot corresponding to the training parameter values, $Re_{train} = \{50, 150, 300, 500\}$. Moreover, the encoded snapshots are augmented by explicitly attaching the corresponding scaled parameter values as labels. The pLSTM network is defined using three stacked LSTM cells with 128 hidden units each and *swish* activation. The network is trained to read an input consisting of 8 time steps and predict the next element in the time series. Training is performed for 50,000 epochs using the Adam optimizer with minibatches of size 150, and with an initial learning rate of 1×10^{-3}, that decays by 20% every 2083 epochs. For optimization of the network hyperparameters, losses in the latent space predictions are computed using the NMSE loss. Similar to the previous approach, scaling the input features for the pLSTM network were not found to be beneficial.

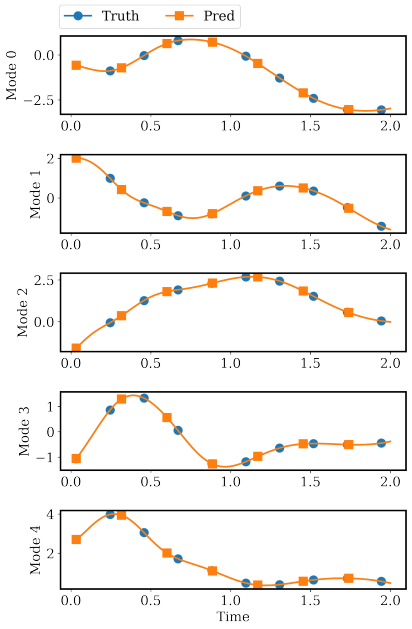
Figure 18a,b show the comparison between the encoded true snapshots and the latent space predictions of the pLSTM model for snapshots corresponding to training parameter values $Re = 150$ and $Re = 300$, respectively. The pLSTM model can be clearly seen to approximate the time trajectory of the latent space coefficients accurately. In Figure 18c,d, the true decoder of the AA3 model is applied to the latent space predictions at four intermediate time steps and the results are compared to the high-dimensional solution snapshots. The solutions predicted by the combined AA3+pLSTM model perfectly match the high-dimensional simulation snapshots.

In the next set of numerical experiments, the trained pLSTM model is deployed to emulate the evolution of the latent space coefficients in a recursive fashion, i.e., the

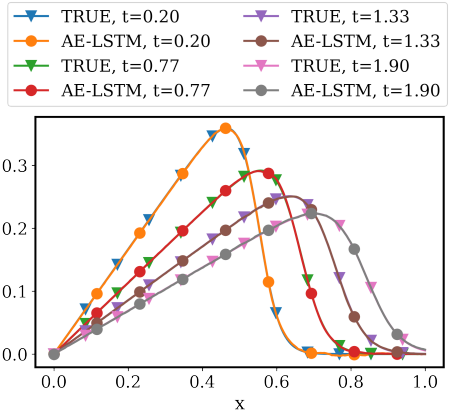
pLSTM model outputs for one time step are recursively rolled into the time step input window and used for pLSTM predictions at the next time step. Figure 19 shows two such examples of recursive pLSTM predictions for training parameter values $Re = 50$ and $Re = 500$, starting from randomly chosen initial time points. In panel (**a**), pLSTM predictions of the latent space evolution for $Re = 50$ are computed by randomly choosing the encoded high-dimensional solution at $t = 0.26$ as the initial data, and marching forward until $t = 2$ in a recursive fashion. Similarly, in (**b**), the initial point is chosen to be $t = 0.50$ and the latent space evolution for $Re = 500$ is computed recursively. In both cases, the predicted trajectories show remarkable agreement with the encoded high-dimensional solution trajectories. Finally, in (**c**) and (**d**), the pLSTM latent space predictions are decoded using model AA3 to compare with the high-dimensional solution snapshots at four intermediate time steps. Again, the predicted solutions are found to closely align with the true high-dimensional snapshot data.



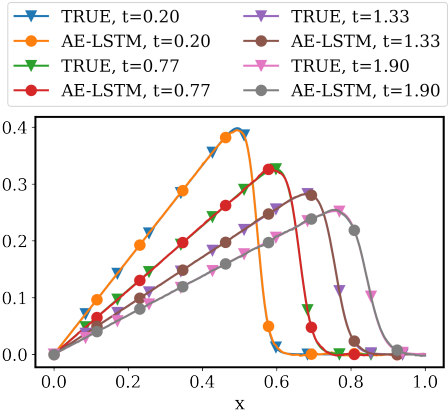
(**a**) pLSTM prediction for $Re = 150$



(**b**) pLSTM prediction for $Re = 300$

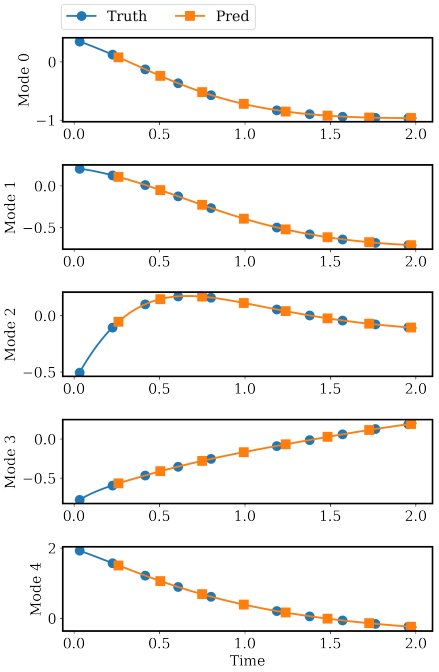


(**c**) Decoded pLSTM prediction for $Re = 150$

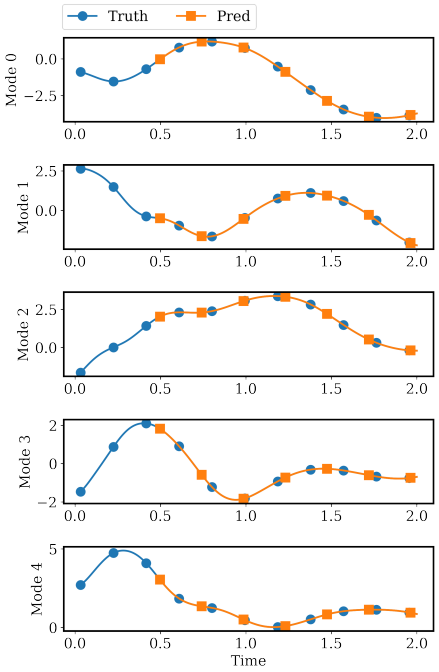


(**d**) Decoded pLSTM prediction for $Re = 300$

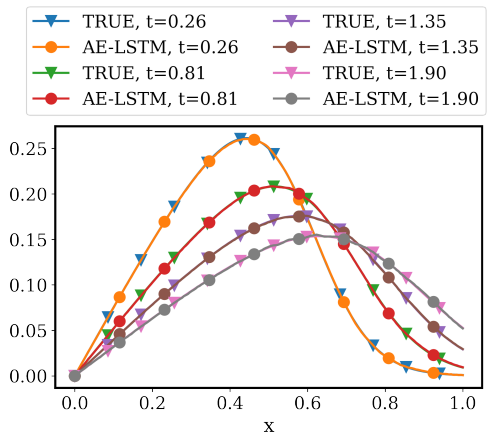
Figure 18. Comparing predictions obtained using a parametric AA autoencoder and a parametric LSTM (pLSTM) model. (**a**,**b**) show the latent space predictions using the pLSTM model and the encoded high-dimensional snapshots for training parameter values $Re = 150$ and $Re = 300$, respectively. (**c**,**d**) compare the corresponding decoded pLSTM predictions with the high-dimensional snapshots at four intermediate time steps.



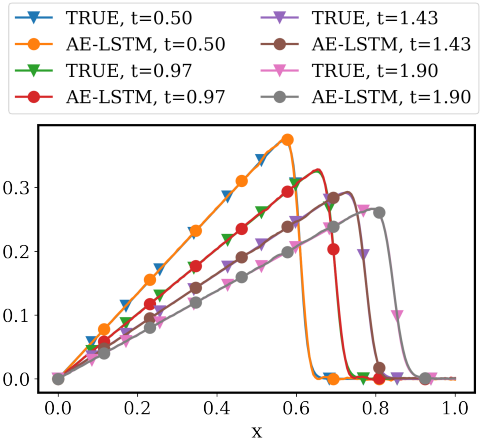
(**a**) pLSTM prediction for $Re = 50$



(**b**) pLSTM prediction for $Re = 500$



(**c**) Decoded pLSTM prediction for $Re = 50$



(**d**) Decoded pLSTM prediction for $Re = 500$

Figure 19. Comparing recursive predictions obtained using a parametric AA autoencoder and a parametric LSTM (pLSTM) model. (**a**,**b**) show the recursive latent space predictions using the pLSTM model and the encoded high-dimensional snapshots for training parameter values $Re = 50$ and $Re = 500$, respectively. (**c**,**d**) compare the corresponding decoded pLSTM predictions with the high-dimensional snapshots at four intermediate time steps.

5. Conclusions and Future Work

In this study, we propose a novel advection-aware autoencoder network that can find a low-dimensional nonlinear embedding of the salient physical features of advection-dominated transport problems. Such systems are known to exhibit instabilities and inefficient compression ratios when expressed in terms of a linear subspace defined by POD-Galerkin projection-based reduced order models. The novelty of the proposed design lies in the definition of a latent space vector that can simultaneously be efficiently mapped to the corresponding high-fidelity simulation snapshot as well as an arbitrary snapshot that effectively emulates the advective features of the high-fidelity snapshot. We demonstrate that for a linear advection problem that requires about 60 linear POD basis modes to accu-

rately capture solution features, the AA autoencoder model can achieve a stable nonlinear embedding using a latent representation of dimension 15, and for a viscous advecting shock problem that requires about 15 POD basis modes, the AA autoencoder model can produce accurate representations with a nonlinear embedding of dimension 5. We also develop a fully, non-intrusive ROM framework by combining the AA autoencoder architecture with a separately trained LSTM network that captures the temporal dynamics in the latent space defined by the AA autoencoder model. This non-intrusive ROM formulation is extended to parametric problems by concatenating the parameter information to both the high-dimensional input snapshots for the AA autoencoder as well as the low-dimensional latent states used for the LSTM training, and then training each model independently. The proposed parametric, non-intrusive ROM is numerically evaluated on the parametric test problem involving a viscous advecting shock. The results indicate that the framework is capable of learning essential underlying features by exhaustively exploring different types of parametric design spaces. Moreover, evaluations with unseen parameter values reveal that the model is also able to produce accurate extrapolatory predictions. The numerical examples presented here demonstrate that the proposed approach is capable of handling not just uniform advection, but also nonlinear problems involving viscous shocks. The key to extending this approach to a wider class of advection-dominated problems such as solitary waves, chaotic systems, and systems governed by multiple traveling waves, lies in the appropriate selection of a shifted snapshot that optimally endows the latent space with information about the underlying advective transport. The registration-type approach of selecting an intermediate high-fidelity simulation snapshot as the shifted snapshot, as adopted here, is a versatile strategy that can be extended even to problems with multiple traveling features. For the examples studied here, the efficacy of the approach was found to be independent of the choice of the particular intermediate time step. However, a systematic sensitivity analysis is required to understand the effect of this choice for systems characterized by multiple traveling waves and chaotic dynamics.

Currently, we are engaged in developing a robust, end-to-end algorithm for concurrent training of the AA autoencoder and the dynamics model, and formulating design guidelines to help the end user choose between concurrent and separately trained ROMs. One important step towards this goal is to explore the addition of physics-based regularizing constraints to the loss function, in order to resolve some of the instabilities in the training trajectory. We are also interested in investigating the use of other ML-based and classical time series learning strategies to model the dynamical features of the latent space coefficients. Another planned direction of future work is aimed at the development of a method that can efficiently map the high-fidelity solution on to a regular, logically rectangular grid. This will allow us to construct AA autoencoder models using convolutional and deconvolutional layers that can more efficiently extract localized spatial patterns in the input features.

Author Contributions: Conceptualization, S.D. and M.W.F.; methodology, S.D. and M.W.F.; software, P.R.-C. and S.D.; validation, S.D., P.R.-C., B.S., and M.W.F.; formal analysis, S.D. and M.W.F.; data curation, S.D. and B.S.; writing—original draft preparation, S.D.; writing—review and editing, M.W.F. and P.R.-C.; visualization, S.D. and B.S.; supervision, M.W.F.; project administration, M.W.F.; funding acquisition, M.W.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All the specific details of the AA autoencoder and the LSTM frameworks used in this study may be found in the supporting source code that will be made available at https://github.com/erdc/aa_autoencoder_mca, accessed on 1 February 2022.

Acknowledgments: The authors acknowledge all the helpful comments from Orie Cecil and Adam Collins. This research was supported in part by an appointment of the first author to the Postgraduate Research Participation Program at the U.S. Army Engineer Research and Development Center, Coastal and Hydraulics Laboratory (ERDC-CHL) administered by the Oak Ridge Institute for Science and

Education through an interagency agreement between the U.S. Department of Energy and ERDC. Permission was granted by the Chief of Engineers to publish this information.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kutz, J.N. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*; Oxford University Press, Inc.: Oxford, UK, 2013.
2. Holmes, P.; Lumley, J.L.; Berkooz, G. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*; Cambridge Monographs on Mechanics, Cambridge University Press: Cambridge, UK, 1996. [CrossRef]
3. Benner, P.; Gugercin, S.; Willcox, K. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM Rev.* **2015**, *57*, 483–531. [CrossRef]
4. Brunton, S.L.; Noack, B.R.; Koumoutsakos, P. Machine Learning for Fluid Mechanics. *Annu. Rev. Fluid Mech.* **2020**, *52*, 477–508. [CrossRef]
5. Hesthaven, J.S.; Rozza, G.; Stamm, B. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*; Springer: Cham, Switzerland, 2016; pp. 1–131. [CrossRef]
6. Rowley, C.W.; Dawson, S.T. Model Reduction for Flow Analysis and Control. *Annu. Rev. Fluid Mech.* **2017**, *49*, 387–417. [CrossRef]
7. Taira, K.; Brunton, S.L.; Dawson, S.T.; Rowley, C.W.; Colonius, T.; McKeon, B.J.; Schmidt, O.T.; Gordeyev, S.; Theofilis, V.; Ukeiley, L.S. Modal analysis of fluid flows: An overview. *AIAA J.* **2017**, *55*, 4013–4041. [CrossRef]
8. Berkooz, G.; Holmes, P.; Lumley, J.L. The proper orthogonal decomposition in the analysis of turbulent flows. *Annu. Rev. Fluid Mech.* **1993**, *25*, 539–575. [CrossRef]
9. Lozovskiy, A.; Farthing, M.; Kees, C.; Gildin, E. POD-based model reduction for stabilized finite element approximations of shallow water flows. *J. Comput. Appl. Math.* **2016**, *302*, 50–70. [CrossRef]
10. Lozovskiy, A.; Farthing, M.; Kees, C. Evaluation of Galerkin and Petrov–Galerkin model reduction for finite element approximations of the shallow water equations. *Comput. Methods Appl. Mech. Eng.* **2017**, *318*, 537–571. [CrossRef]
11. Carlberg, K.; Barone, M.; Antil, H. Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction. *J. Comput. Phys.* **2017**, *330*, 693–734. [CrossRef]
12. Dutta, S.; Rivera-Casillas, P.; Cecil, O.; Farthing, M. pyNIROM—A suite of python modules for non-intrusive reduced order modeling of time-dependent problems. *Softw. Impacts* **2021**, *10*, 100129. [CrossRef]
13. Alla, A.; Kutz, J.N. Nonlinear model order reduction via dynamic mode decomposition. *SIAM J. Sci. Comput.* **2017**, *39*, B778–B796. [CrossRef]
14. Wu, Z.; Brunton, S.L.; Revzen, S. Challenges in Dynamic Mode Decomposition. *J. R. Soc. Interface* **2021**, *18*, 20210686. [CrossRef] [PubMed]
15. Xiao, D.; Fang, F.; Pain, C.C.; Navon, I.M. A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications. *Comput. Methods Appl. Mech. Eng.* **2017**, *317*, 868–889. [CrossRef]
16. Dutta, S.; Farthing, M.W.; Perracchione, E.; Savant, G.; Putti, M. A greedy non-intrusive reduced order model for shallow water equations. *J. Comput. Phys.* **2021**, *439*, 110378. [CrossRef]
17. Guo, M.; Hesthaven, J.S. Data-driven reduced order modeling for time-dependent problems. *Comput. Methods Appl. Mech. Eng.* **2019**, *345*, 75–99. [CrossRef]
18. Xiao, D. Error estimation of the parametric non-intrusive reduced order model using machine learning. *Comput. Methods Appl. Mech. Eng.* **2019**, *355*, 513–534. [CrossRef]
19. Hesthaven, J.S.; Ubbiali, S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *J. Comput. Phys.* **2018**, *363*, 55–78. [CrossRef]
20. Wan, Z.Y.; Vlachas, P.; Koumoutsakos, P.; Sapsis, T. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PLoS ONE* **2018**, *13*, e0197704. [CrossRef]
21. Maulik, R.; Mohan, A.; Lusch, B.; Madireddy, S.; Balaprakash, P.; Livescu, D. Time-series learning of latent-space dynamics for reduced-order model closure. *Phys. D Nonlinear Phenom.* **2020**, *405*, 132368. [CrossRef]
22. Chen, R.T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D. Neural Ordinary Differential Equations. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Montréal, QC, Canada, 2–8 December 2018; pp. 6572–6583. [CrossRef]
23. Dutta, S.; Rivera-Casillas, P.; Farthing, M.W. Neural Ordinary Differential Equations for Data-Driven Reduced Order Modeling of Environmental Hydrodynamics. In Proceedings of the AAAI 2021 Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physical Sciences, Virtual Meeting, 22–24 March 2021; CEUR-WS: Stanford, CA, USA, 2021.
24. Wu, P.; Sun, J.; Chang, X.; Zhang, W.; Arcucci, R.; Guo, Y.; Pain, C.C. Data-driven reduced order model with temporal convolutional neural network. *Comput. Methods Appl. Mech. Eng.* **2020**, *360*, 112766. [CrossRef]
25. Taddei, T.; Perotto, S.; Quarteroni, A. Reduced basis techniques for nonlinear conservation laws. *ESAIM Math. Model. Numer. Anal.* **2015**, *49*, 787–814. [CrossRef]
26. Greif, C.; Urban, K. Decay of the Kolmogorov N-width for wave problems. *Appl. Math. Letters* **2019**, *96*, 216–222. [CrossRef]

27. Carlberg, K.; Farhat, C.; Cortial, J.; Amsallem, D. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *J. Comput. Phys.* **2013**, *242*, 623–647. [CrossRef]
28. Nair, N.J.; Balajewicz, M. Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks. *Int. J. Numer. Methods Eng.* **2019**, *117*, 1234–1262. [CrossRef]
29. Rim, D.; Moe, S.; LeVeque, R.J. Transport reversal for model reduction of hyperbolic partial differential equations. *SIAM/ASA J. Uncertain. Quantif.* **2018**, *6*, 118–150. [CrossRef]
30. Reiss, J.; Schulze, P.; Sesterhenn, J.; Mehrmann, V. The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena. *SIAM J. Sci. Comput.* **2018**, *40*, A1322–A1344. [CrossRef]
31. Rim, D.; Peherstorfer, B.; Mandli, K.T. Manifold approximations via transported subspaces: Model reduction for transport-dominated problems. *arXiv* **2019**, arXiv:1912.13024.
32. Cagniart, N.; Maday, Y.; Stamm, B. Model order reduction for problems with large convection effects. In *Contributions to Partial Differential Equations and Applications*; Springer International Publishing: Cham, Switzerland, 2019; pp. 131–150. [CrossRef]
33. Taddei, T. A registration method for model order reduction: Data compression and geometry reduction. *SIAM J. Sci. Comput.* **2020**, *42*, A997–A1027. [CrossRef]
34. Peherstorfer, B. Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *SIAM J. Sci. Comput.* **2020**, *42*, A2803–A2836. [CrossRef]
35. Kashima, K. Nonlinear model reduction by deep autoencoder of noise response data. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 5750–5755. [CrossRef]
36. Lee, K.; Carlberg, K.T. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *J. Comput. Phys.* **2020**, *404*, 108973. [CrossRef]
37. Kim, Y.; Choi, Y.; Widemann, D.; Zohdi, T. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *J. Comput. Phys.* **2022**, *451*, 110841. [CrossRef]
38. Willcox, K. Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. *Comput. Fluids* **2006**, *35*, 208–226. [CrossRef]
39. Chaturantabut, S.; Sorensen, D.C. Nonlinear model reduction via Discrete Empirical Interpolation. *SIAM J. Sci. Comput.* **2010**, *32*, 2737–2764. [CrossRef]
40. Mendible, A.; Brunton, S.L.; Aravkin, A.Y.; Lowrie, W.; Kutz, J.N. Dimensionality reduction and reduced-order modeling for traveling wave physics. *Theor. Comput. Fluid Dyn.* **2020**, *34*, 385–400. [CrossRef]
41. Haasdonk, B.; Ohlberger, M. Adaptive basis enrichment for the reduced basis method applied to finite volume schemes. In Proceedings of the Fifth International Symposium on Finite Volumes for Complex Applications, Aussois, France, 8–13 June 2008; pp. 471–479.
42. Chen, P.; Quarteroni, A.; Rozza, G. A weighted empirical interpolation method: A priori convergence analysis and applications. *ESAIM Math. Model. Numer. Anal.* **2014**, *48*, 943–953. [CrossRef]
43. Amsallem, D.; Farhat, C. An online method for interpolating linear parametric reduced-order models. *SIAM J. Sci. Comput.* **2011**, *33*, 2169–2198. [CrossRef]
44. Maday, Y.; Stamm, B. Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces. *SIAM J. Sci. Comput.* **2013**, *35*, A2417–A2441. [CrossRef]
45. Peherstorfer, B.; Butnaru, D.; Willcox, K.; Bungartz, H.J. Localized discrete empirical interpolation method. *SIAM J. Sci. Comput.* **2014**, *36*, A168–A192. [CrossRef]
46. Carlberg, K. Adaptive *h*-refinement for reduced-order models. *Int. J. Numer. Methods Eng.* **2015**, *102*, 1192–1210. [CrossRef]
47. Peherstorfer, B.; Willcox, K. Online adaptive model reduction for nonlinear systems via low-rank updates. *SIAM J. Sci. Comput.* **2015**, *37*, A2123–A2150. [CrossRef]
48. Tenenbaum, J. Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems*; Jordan, M., Kearns, M., Solla, S., Eds.; MIT Press: Cambridge, MA, USA, 1998; Volume 10.
49. Roweis, S.T.; Saul, L.K. Nonlinear dimensionality reduction by locally linear embedding. *Science* **2000**, *290*, 2323–2326. [CrossRef]
50. Belkin, M.; Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **2003**, *15*, 1373–1396. [CrossRef]
51. van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
52. Kohonen, T. Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **1982**, *43*, 59–69. [CrossRef]
53. Mika, S.; Schölkopf, B.; Smola, A.; Müller, K.R.; Scholz, M.; Rätsch, G. Kernel PCA and de-noising in feature spaces. In *Advances in Neural Information Processing Systems*; Kearns, M., Solla, S., Cohn, D., Eds.; MIT Press: Cambridge, MA, USA, 1999; Volume 11.
54. Walder, C.; Schölkopf, B. Diffeomorphic dimensionality reduction. In *Advances in Neural Information Processing Systems 21*; Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., Eds.; Curran Associates Inc.: Red Hook, NY, USA, 2009; pp. 1713–1720.
55. Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [CrossRef] [PubMed]
56. Masci, J.; Meier, U.; Cireşan, D.; Schmidhuber, J. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In *Artificial Neural Networks and Machine Learning—ICANN 2011*; Honkela, T., Duch, W., Girolami, M., Kaski, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 52–59.

57. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, Puerto Rico, 2–4 May 2016; pp. 1–16.
58. Champion, K.; Lusch, B.; Nathan Kutz, J.; Brunton, S.L. Data-driven discovery of coordinates and governing equations. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 22445–22451. [CrossRef]
59. Dutta, S.; Rivera-Casillas, P.; Cecil, O.M.; Farthing, M.W.; Perracchione, E.; Putti, M. Data-driven reduced order modeling of environmental hydrodynamics using deep autoencoders and neural ODEs. *arXiv* **2021**, arXiv:2107.02784.
60. Maulik, R.; Lusch, B.; Balaprakash, P. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Phys. Fluids* **2021**, *33*, 037106. [CrossRef]
61. Wehmeyer, C.; Noé, F. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *J. Chem. Phys.* **2018**, *148*, 241703. [CrossRef]
62. Nishizaki, H. Data augmentation and feature extraction using variational autoencoder for acoustic modeling. In Proceedings of the 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Kuala Lumpur, Malaysia, 12–15 December 2017; pp. 1222–1227. [CrossRef]
63. Bakarji, J.; Champion, K.; Kutz, J.N.; Brunton, S.L. Discovering Governing Equations from Partial Measurements with Deep Delay Autoencoders. *arXiv* **2022**, arXiv:2201.05136.
64. Erichson, N.B.; Muehlebach, M.; Mahoney, M.W. Physics-informed Autoencoders for Lyapunov-stable Fluid Flow Prediction. *arXiv* **2019**, arXiv:1905.10866.
65. Gonzalez, F.J.; Balajewicz, M. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv* **2018**, arXiv:1808.01346.
66. Mojgani, R.; Balajewicz, M. Low-Rank Registration Based Manifolds for Convection-Dominated PDEs. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), Vancouver, BC, Canada, 2–9 February 2021; pp. 399–407.
67. Plaut, E. From Principal Subspaces to Principal Components with Linear Autoencoders. *arXiv* **2018**, arXiv:1804.10253.
68. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2222–2232. [CrossRef]
69. Eivazi, H.; Veisi, H.; Naderi, M.H.; Esfahanian, V. Deep neural networks for nonlinear model order reduction of unsteady flows. *Phys. Fluids* **2020**, *32*, 105104. [CrossRef]
70. Maulik, R.; Lusch, B.; Balaprakash, P. Non-autoregressive time-series methods for stable parametric reduced-order models. *Phys. Fluids* **2020**, *32*, 087115. [CrossRef]
71. del Águila Ferrandis, J.; Triantafyllou, M.S.; Chryssostomidis, C.; Karniadakis, G.E. Learning functionals via LSTM neural networks for predicting vessel dynamics in extreme sea states. *Proc. R. Soc. A* **2021**, *477*, 20190897. [CrossRef]
72. Chattopadhyay, A.; Mustafa, M.; Hassanzadeh, P.; Kashinath, K. Deep Spatial Transformers for Autoregressive Data-Driven Forecasting of Geophysical Turbulence. In Proceedings of the 10th International Conference on Climate Informatics, Oxford, UK, 22–25 September 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 106–112. [CrossRef]
73. Pathak, J.; Hunt, B.; Girvan, M.; Lu, Z.; Ott, E. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. *Phys. Rev. Lett.* **2018**, *120*, 24102. [CrossRef]
74. Usman, A.; Rafiq, M.; Saeed, M.; Nauman, A.; Almqvist, A.; Liwicki, M. Machine learning-accelerated computational fluid dynamics. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e2101784118. [CrossRef]
75. Stabile, G.; Zancanaro, M.; Rozza, G. Efficient geometrical parametrization for finite-volume-based reduced order methods. *Int. J. Numer. Methods Eng.* **2020**, *121*, 2655–2682. [CrossRef]