

MOX-Report No. 80/2026

Deep Invertible Autoencoders for Dimensionality Reduction of Dynamical Systems

Botteghi, N; Glas, S; Brune, C

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it https://mox.polimi.it

Deep Invertible Autoencoders for Dimensionality Reduction of Dynamical Systems

Nicolò Botteghi[a], Silke Glas[b], Christoph Brune[b]

[a] *MOX – Department of Mathematics, Politecnico di Milano, Piazza Leonardo da Vinci 32, I-20133, Milano, Italy*
[b] *Department of Applied Mathematics, University of Twente, Enschede, Netherlands*

Abstract

Constructing reduced-order models (ROMs) capable of efficiently predicting the evolution of parameter-dependent high-dimensional dynamical systems is crucial in many applications in engineering and applied sciences. A popular class of projection-based ROMs projects the high-dimensional full-order model (FOM) dynamics onto a low-dimensional manifold. These projection-based ROMs approaches often rely on classical model reduction techniques such as proper orthogonal decomposition (POD) or, more recently, on neural network architectures such as autoencoders (AEs). In the case that the ROM is constructed by the POD, one has approximation guaranteed based based on the singular values of the problem at hand. However, POD-based techniques can suffer from slow decay of the singular values in transport- and advection-dominated problems. In contrast to that, AEs allow for better reduction capabilities than the POD, often with the first few modes, but at the price of theoretical considerations. In addition, it is often observed, that AEs exhibits a plateau of the projection error with the increment of the dimension of the trial manifold.
In this work, we propose a deep invertible AE architecture, named *inv-AE*, that computationally improves upon the stagnation of the reconstruction error typical of traditional AE architectures, e.g., convolutional, and the reconstructions quality. Inv-AE is composed of several invertible neural network layers that allows for gradually recovering more information about the FOM solutions the more we increase the dimension of the reduced manifold. Through the application of inv-AE to a parametric 1-dimensional Burgers' equation, a parametric 2-dimensional fluid flow around an obstacle with variable geometry, and a parametric 3-dimensional Korteweg-de Vries, we show that (i) inv-AE mitigates the issue of the characteristic plateau of (convolutional and fully connected) AEs and (ii) inv-AE can be combined with popular autoencoder-based ROM approaches, e.g., DL-ROM and POD-DL-ROM, to improve their accuracy.

1. Introduction

Many applications in engineering and applied science, e.g., fluid dynamics, chemical and process engineering, climate science, and (soft) robotics, require accurate physical modeling of the systems by means of partial differential equations (PDEs). Often, these systems do not yield closed-form solutions such that numerical simulations are needed to provide approximations. Usually, the solution of these high-dimensional models, which we refer to as full-order models (FOMs), is often computational demanding, limiting the application of such models in all the scenarios requiring real-time or iterative integration, e.g., in optimal control, reinforcement learning, or uncertainty quantification [1]. Reducing the computational cost of the FOMs by investigating new strategies for efficient model reduction (MOR) has been a research focus for decades and, more recently, the advances of machine learning had a significant influence on the development of data-driven reduced-order models (ROMs).

In case that the underlying equations are available, one popular option for building ROMs is by projection of the FOM equations onto a trial manifold, see, e.g., [1, 2, 3, 4, 5]. The trial manifold can be obtained using linear or nonlinear projection-based techniques such as proper orthogonal decomposition (POD) or autoencoders (AE) [6]. While linear projection techniques based on the singular value decomposition (SVD) have solid mathematical foundations and implementations, they may suffer from slow decay of the singular values in transport- and advection-dominated problems [7]. The slow decay of the singular values negatively affects the reduction capabilities of the linear-subspace ROMs, making the dimension of the trial manifold

extremely large, if high accuracy is required, and thus might not allow a drastic reduction of the computational cost of the ROM compared to the FOM. This is due to the fact, that linear-subspace ROMs have a lower bound for the approximation quality, the so-called Kolmogorov n-widths. The Kolmogorov n-widths denote the best-possible error for a linear-subspace projection-based ROM of size n, however, in most cases, there is no explicit construction available how this best-possible approximation error can be attained.

The recent advances of machine learning have made nonlinear projection-based techniques, such as AEs, more and more appealing. AE-based methods can improve the reduction capabilities of linear-subspace MOR methods in challenging problems, showing better decay of the approximation errors, especially within the first few eigenfunctions [8, 9, 10, 5, 11, 12]. However, the use of neural networks (NNs) often complicated theoretical consideration of the approximation bounds of the techniques. Additionally, as observed in e.g., [13, 14, 15], the reconstruction error of AEs, namely the error due to the reduction of dimensionality, with respect to the dimension of the trial manifold, exhibits a plateau (often in the order of 10^{-3}) after approximately reaching the intrinsic dimension of the underlying problem. This critical drawback of AE architecture poses a strict lower bound on the approximation capabilities of the resulting ROMs.

AEs utilize two NNs to approximate the reconstruction mappings that are commonly referred to in the machine learning community as the *encoder* and the *decoder*. However, the encoder is (usually) a non-invertible function that does not allow the decoder to properly recover the inputs accurately. The non-invertibility of the encoder makes the decoder process ambiguous and causes loss of information [16]. Increasing the latent dimension does not mitigate this drawback and it explains the plateau observed in [13] when AEs are used for building ROMs. Building symmetric AEs by construction has been shown by, e.g., [17, 18] to be an effective strategy to improve upon traditional AE architectures in terms of performance and theoretical guarantees. However, the construction of such symmetric AEs comes at the price of relying on specialized cost functions or additional computational complexity.

The construction of *analytically invertible* NNs layers, such as coupling layers, invertible residual networks, and Glow-type layers [19, 20, 21, 22, 23, 24] has gained attention due to the numerous methods and applications that can benefit from them, e.g., generative models, density estimation, normalizing flows, generative adversarial [25, 26, 27, 28, 29], bi-directional training and efficient backpropagation [21], inverse problems [30, 31], fundamental analysis [32, 33, 34, 35], and, recently, dimensionality reduction with autoencoders [16, 36, 37]. However, it is worth mentioning that the analytical invertibility of invertible NNs do not necessarily translates into their *numerical invertibility*. As highlighted in [35], invertible NNs often experience exploding inverses due to numerical instabilities and errors. The Lipschitz constant of the invertible NNs determines how much these errors are magnified or attenuated. Therefore, controlling the Lipschitz constant is crucial for the numerical stability and invertibility of invertible NNs. Different invertible NN layers have different stability properties, but in general numerical errors become more severe the more we increase the depth of the models. Spectral normalization [38], sigmoid scaling [29], and weight initialization [39] can be used to control the local stability of the invertible NNs. However, global stability of the forward and backward pass is often hard to guarantee [35].

In this work, we focus on enhancing nonlinear projection-based ROMs. In particular, we propose an analytically and numerically invertible AE (*inv-AE*) architecture, tailored to MOR, that overcomes the reconstruction error plateau characterizing traditional AE architectures composed of feedforward or convolutional NN layers. To construct an inv-AE, we stack multiple *invertible affine coupling layers* [19, 20] to build an invertible NN. The forward pass through the invertible NN represents the encoder $\boldsymbol{\phi}$, while the backward pass represents the decoder $\boldsymbol{\psi}$ that in this case is equal to the inverse of the encoder by construction, i.e., $\boldsymbol{\psi} = \boldsymbol{\phi}^{-1}$ (restricted to its image). In this way, through *parameter sharing* of encoder and decoder, we are capable of drastically reducing the number of learnable parameters, namely weights and biases of the NN, and consequently the training cost. With reference to Figure 1, a generic snapshot $\boldsymbol{x}_k^{\boldsymbol{\mu}}$, where k denotes the time step and $\boldsymbol{\mu}$ the parametric dependency of the snapshot, is projected by the invertible layers, with learnable parameters $\boldsymbol{\theta}_{\boldsymbol{\phi}}$, to $\boldsymbol{\phi}(\boldsymbol{x}_k^{\boldsymbol{\mu}}; \boldsymbol{\theta}_{\boldsymbol{\phi}})$. An invertible NN requires the inputs to be of the same dimension of its outputs. Thus, to reduce the dimensionality and promote the compression of information onto a lower-dimensional manifold, we zero-mask $\boldsymbol{\phi}(\boldsymbol{x}_k^{\boldsymbol{\mu}}; \boldsymbol{\theta}_{\boldsymbol{\phi}})$ up to the chosen latent dimensionality to obtain $\boldsymbol{z}_k^{\boldsymbol{\mu}}$. After zero-masking, $\boldsymbol{z}_k^{\boldsymbol{\mu}}$ is used fed to the inverse mapping $\boldsymbol{\phi}^{-1}(\boldsymbol{z}_k^{\boldsymbol{\mu}}; \boldsymbol{\theta}_{\boldsymbol{\phi}})$, i.e., the decoder of traditional AEs, to recover the reconstruction $\hat{\boldsymbol{x}}_k^{\boldsymbol{\mu}}$ of $\boldsymbol{x}_k^{\boldsymbol{\mu}}$. With this simple zero-masking, inspired by [16], we have transformed an invertible NN into an inv-AE.

In many complex problems, such as in fluid dynamics, the snapshots can be extremely high-dimensional.

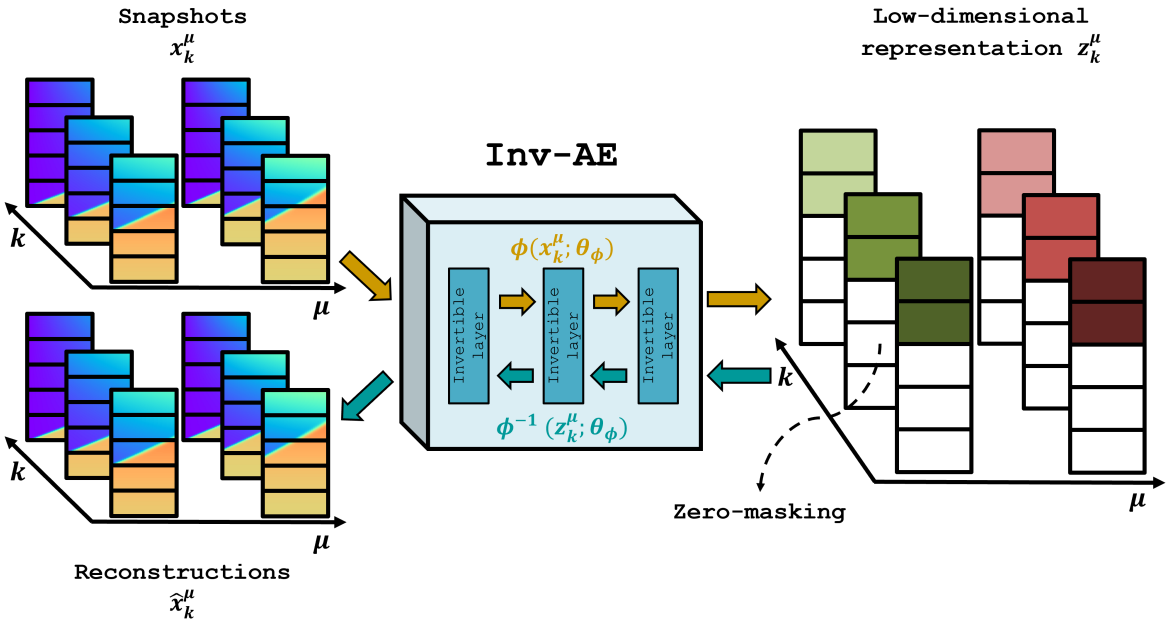


Figure 1. Dimensionality reduction with inv-AE. The snapshots $\boldsymbol{x}_k^{\boldsymbol{\mu}}$ are encoded using the encoder $\boldsymbol{\phi}(\boldsymbol{x}_k^{\boldsymbol{\mu}};\boldsymbol{\theta}_{\boldsymbol{\phi}})$, successively zero-masked up to a chosen latent dimension, and eventually decoded using the inverse of the encoder $\boldsymbol{\phi}^{-1}(\boldsymbol{z}_k^{\boldsymbol{\mu}};\boldsymbol{\theta}_{\boldsymbol{\phi}})$.

However, it is possible to first perform dimensionality-reduction using SVD on the data matrix

$$\boldsymbol{X}_{\text{train}} \approx \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

where $\boldsymbol{U} \in \mathbb{R}^{N\times r}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$, $\boldsymbol{V} \in \mathbb{R}^{(N_t\times|\mathcal{P}_{\text{train}}|)\times r}$, with r being the chosen number of modes used to represent the data. Using $\boldsymbol{U}^\top$ we can obtain the POD coefficient $\boldsymbol{h}_k^{\boldsymbol{\mu}}$ from $\boldsymbol{x}_k^{\boldsymbol{\mu}}$ and rely on the inv-AE to recover the POD coefficients $\boldsymbol{h}_k^{\boldsymbol{\mu}}$. Eventually, using $\boldsymbol{U}$ we can recover the reconstruction $\hat{\boldsymbol{x}}_k^{\boldsymbol{\mu}}$ of $\boldsymbol{x}_k^{\boldsymbol{\mu}}$ from the reconstructed POD coefficients $\hat{\boldsymbol{h}}_k^{\boldsymbol{\mu}}$. We refer to this approach as *POD-inv-AE* and we depict it in Figure 2.

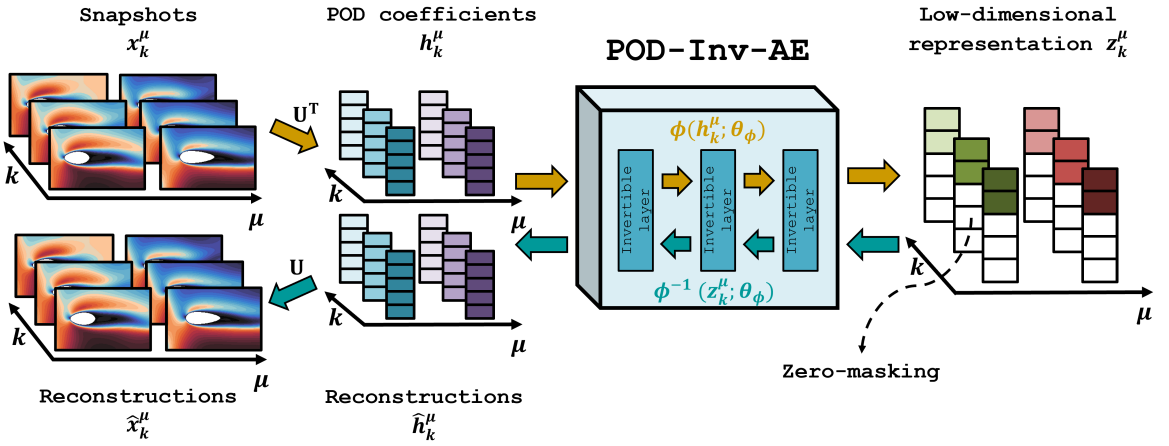


Figure 2. Dimensionality reduction with POD-inv-AE. The snapshots dimensionality is first reduced using POD, then the POD coefficients are further compressed by zero-masking output of encoder $\boldsymbol{\phi}(\boldsymbol{x}_k^{\boldsymbol{\mu}};\boldsymbol{\theta}_{\boldsymbol{\phi}})$. The POD coefficients are recovered by decoding the latent variables $\boldsymbol{\phi}^{-1}(\boldsymbol{z}_k^{\boldsymbol{\mu}};\boldsymbol{\theta}_{\boldsymbol{\phi}})$, and eventually the snapshots are reconstructed from the POD coefficients using U.

We numerically study the reduction capabilities of inv-AE and POD-inv-AE on a parametric Burgers' equation, analogously to [13], on a parametric fluid flow around a obstacle with variable geometry, and a

parametric Korteweg-de Vries equation. Eventually, we show that inv-AE and POD-inv-AE can replace the traditional AE architectures of popular deep learning-based ROM techniques such as DL-ROM [5] and POD-DL-ROM [40] to improve the accuracy of the predictions and consequently the performance of the resulting ROMs.

The rest of the paper is organized as follows: Section 2 introduces problem settings, linear and nonlinear strategies for dimensionality reduction, and deep learning-based ROM. Section 3 presents the proposed invertible AE architecture and its combination with deep learning-based ROM. Section 4 shows the numerical experiments, results, and discusses the findings. Eventually, Section 5 concludes the paper.

2. Preliminaries

In this section, we summarize existing background knowledge needed for further comprehension of the paper. We begin by stating the parametric initial value problem, its MOR, the limitations of linear-subspace MOR, and AEs in Section 2.1. Further, in Section 2.2 we introduce (POD) DL-ROMs based on AEs.

2.1. Problem Setting and Limitations of Linear-subspace Model Reduction

We consider parametric initial value problems as our FOM model: for given $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^p, p \in \mathbb{N}$ and initial condition $\boldsymbol{x}_0(\boldsymbol{\mu}) \in \mathbb{R}^N$, we seek the solution $\boldsymbol{x}(\cdot;\mu) \in C^1(\mathcal{I};\mathbb{R}^N)$ such that

$$\frac{\mathrm{d}}{\mathrm{dt}}\boldsymbol{x}(t;\boldsymbol{\mu}) = \boldsymbol{f}(t,\boldsymbol{x}(t;\boldsymbol{\mu});\boldsymbol{\mu}), \qquad \boldsymbol{x}(t_0;\boldsymbol{\mu}) = \boldsymbol{x}_0(\boldsymbol{\mu}) \in \mathbb{R}^N, \tag{1}$$

using time interval $\mathcal{I} = (t_0, t_f]$ and $t_0 < t_f < \infty$, and right-hand side $\boldsymbol{f} : \mathcal{I} \times \mathbb{R}^N \times \mathcal{P} \to \mathbb{R}^N$. By solving the FOM with a numerical integration scheme in time, we obtain a collection of snapshots of the type $\boldsymbol{x}(t;\boldsymbol{\mu}) \to \boldsymbol{x}(t_k;\boldsymbol{\mu}) = \boldsymbol{x}_k^{\boldsymbol{\mu}}$ for $k = 1, \cdots, N_t$ for each parameter $\boldsymbol{\mu}$ in a discrete training set $\mathcal{P}_{\text{train}} \subseteq \mathcal{P}$, where we set $N_{\text{train}} := |\mathcal{P}_{\text{train}}|$. The collection of snapshots compose our data matrix such that we arrive at

$$\boldsymbol{X}_{\text{train}} = \begin{bmatrix} | & | & \cdots & | & | \\ \boldsymbol{x}_1{}^{\boldsymbol{\mu}_1} & \boldsymbol{x}_2^{\boldsymbol{\mu}_1} & \cdots & \boldsymbol{x}_{N_t-1}^{\boldsymbol{\mu}_{N_{\text{train}}}} & \boldsymbol{x}_{N_t}^{\boldsymbol{\mu}_{N_{\text{train}}}} \\ | & | & \cdots & | & | \end{bmatrix} \in \mathbb{R}^{N \times (N_t \times N_{\text{train}})}.$$

Given the FOM in (1), a ROM can be constructed via reconstruction with the two mappings previously denoted as encoder $\boldsymbol{\phi} : \mathbb{R}^N \to \mathbb{R}^n$ and decoder $\boldsymbol{\psi} : \mathbb{R}^n \to \mathbb{R}^N$ such that $\boldsymbol{\phi} \circ \boldsymbol{\psi} = \mathrm{id}_{\mathbb{R}^n}$ and $n \ll N$. Two terms commonly used in MOR are the computational intense offline phase, which is dependent on the FOM dimension N and in which the ROM is built, as well as the cost-effective online phase, where the ROM (ideally only depending on the latent dimension n) is evaluated. Linear-subspace MOR methods employs linear mappings for $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$, while in the case of MOR on manifolds, the mappings $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are nonlinear functions.[1]

In the case, that the mapping $\boldsymbol{\phi}, \boldsymbol{\psi}$ are linear the quality of the resulting ROM approximation can be bounded from below by the Kolmogorov n-widths. In many applications, such as in elliptic and parabolic partial differential equations, see, e.g., [42, 43], it is observed, that Kolmogorov n-widths can obtain an exponential decay. However, in the case of transport-dominated problems, the Kolmogorov n-widths may decay slowly [7]. One possibility on how to break this Kolmogorov barrier is to employ nonlinear mapping for $\boldsymbol{\phi}, \boldsymbol{\psi}$ such as autoencoders.

AEs [44, 6] are special types of NN architectures used for dimensionality reduction. An AE (i) maps the high-dimensional data onto a low-dimensional latent variable representation using the encoder and (ii) maps the low-dimensional latent variables back to the original high-dimensional data space using the decoder, i.e, $\boldsymbol{x} \approx \hat{\boldsymbol{x}} = \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{\theta}_{\boldsymbol{\phi}});\boldsymbol{\theta}_{\boldsymbol{\psi}})$. We denote the learnable parameters, namely weights and biases, of the encoder and decoder by $\boldsymbol{\theta}_{\boldsymbol{\phi}}$ and $\boldsymbol{\theta}_{\boldsymbol{\psi}}$, respectively. These learnable parameters $\boldsymbol{\theta}_{\boldsymbol{\phi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}$ of the AE are jointly optimized by minimizing the reconstruction loss over a (train) set X on M training samples that we indicate as $\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(\mathrm{M})}\}$, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}_{\boldsymbol{\phi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}) = \frac{1}{M}\sum_{i=1}^{M} ||\boldsymbol{x}^{(i)} - \hat{\boldsymbol{x}}^{(i)}||_2^2 = \frac{1}{M}\sum_{i=1}^{M} ||\boldsymbol{x}^{(i)} - \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x}^{(i)};\boldsymbol{\theta}_{\boldsymbol{\phi}});\boldsymbol{\theta}_{\boldsymbol{\psi}})||_2^2, \tag{2}$$

[1] For a general differential geometric formulation of MOR on manifolds, we refer the reader to [41].

where $\boldsymbol{x}^{(\mathrm{i})}$ indicates the ith training data and $\hat{\boldsymbol{x}}^{(\mathrm{i})}$ its reconstruction using the AE. The dimension of the latent space is a hyper-parameter of the AE model that can be chosen using prior knowledge or through a trail-and-error approach. Moreover, for $\boldsymbol{\psi}, \boldsymbol{\phi}$ being nonlinear mappings, adjusted variants of limiting widths can be investigated, where we refer the interested reader to [45] for an overview of nonlinear widths.

2.2. (POD) Deep Learning-based ROMs

The method of Deep Learning-based ROM (DL-ROM) [5] is a popular and effective approach for learning ROMs from data. In this context, the generic i^{th} data $\boldsymbol{x}^{(\mathrm{i})}$ corresponds to the ith FOM snapshot $\boldsymbol{x}_k^{\boldsymbol{\mu}}$ that might have temporal and parametric dependencies. DL-ROM is composed of a (convolutional) encoder $\boldsymbol{\phi}(\boldsymbol{x}_k^{\boldsymbol{\mu}}; \boldsymbol{\theta}_{\boldsymbol{\phi}})$ and a (convolutional) decoder $\boldsymbol{\psi}(\boldsymbol{z}_k^{\boldsymbol{\mu}}; \boldsymbol{\theta}_{\boldsymbol{\psi}})$ that learn a low-dimensional representation $\boldsymbol{z}_k^{\boldsymbol{\mu}}$ of the snapshots and their reconstructions $\hat{\boldsymbol{x}}_k^{\boldsymbol{\mu}}$ in parallel. To efficiently predict the system's dynamics, DL-ROM uses a feedforward NN $\boldsymbol{\xi}(k, \boldsymbol{\mu}; \boldsymbol{\theta}_{\boldsymbol{\xi}})$, with learnable parameters $\boldsymbol{\theta}_{\xi}$, to predict the latent vector $\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}$ from the timestep t and parameter vector $\boldsymbol{\mu}$. We can then retrieve the approximated FOM snapshots by decoding $\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}$. We can optimize the DL-ROM parameters, namely $\boldsymbol{\theta}_{\boldsymbol{\phi}}$, $\boldsymbol{\theta}_{\boldsymbol{\psi}}$, and $\boldsymbol{\theta}_{\boldsymbol{\xi}}$, by minimizing over the loss function

$$
\begin{aligned}
\mathcal{L}_{\text{DL-ROM}}(\boldsymbol{\theta}_{\boldsymbol{\phi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}, \boldsymbol{\theta}_{\boldsymbol{\xi}}) &= \frac{1}{M} \sum_{i=1}^{M} \left(\frac{\alpha}{2} ||\boldsymbol{x}^{(i)} - \tilde{\boldsymbol{x}}^{(i)}||_2^2 + \frac{1-\alpha}{2} ||\boldsymbol{z}^{(i)} - \tilde{\boldsymbol{z}}^{(i)}||_2^2 \right) \\
&= \frac{1}{M} \sum_{i=1}^{M} \left(\frac{\alpha}{2} ||\boldsymbol{x}^{(i)} - \boldsymbol{\psi}(\tilde{\boldsymbol{z}}^{(i)}; \boldsymbol{\theta}_{\boldsymbol{\psi}})||_2^2 + \frac{1-\alpha}{2} ||\boldsymbol{\phi}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}_{\boldsymbol{\phi}}) - \tilde{\boldsymbol{z}}^{(i)}||_2^2 \right) \\
&= \frac{1}{M} \sum_{i=1}^{M} \left(\frac{\alpha}{2} ||\boldsymbol{x}^{(i)} - \boldsymbol{\psi}(\boldsymbol{\xi}(k^{(i)}, \boldsymbol{\mu}^{(i)}; \boldsymbol{\theta}_{\boldsymbol{\xi}}); \boldsymbol{\theta}_{\boldsymbol{\psi}})||_2^2 + \frac{1-\alpha}{2} ||\boldsymbol{\phi}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}_{\boldsymbol{\phi}}) - \boldsymbol{\xi}(k^{(i)}, \boldsymbol{\mu}^{(i)}; \boldsymbol{\theta}_{\boldsymbol{\xi}})||_2^2 \right),
\end{aligned}
\tag{3}
$$

where $0 \leq \alpha \leq 1$. In the online phase, we use $\boldsymbol{\xi}(k, \boldsymbol{\mu}; \boldsymbol{\theta}_{\boldsymbol{\xi}})$ to predict the evolution of the system at an arbitrary timestep k for a given $\boldsymbol{\mu}$ and decode the latent vectors $\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}$ using the decoder to retrieve approximations to the FOM solutions.

The POD-DL-ROM method [40] extends DL-ROM with an additional POD reduction step before the AE neural network to drastically reducing the number of learnable parameters of the AE and consequently reduce the training times. First the POD is computed for a chosen dimension r, $n \leq r \leq N$. Using the POD basis $\boldsymbol{U} \in \mathbb{R}^{N \times r}$ the respective POD coefficients $\boldsymbol{h}_k^{\boldsymbol{\mu}} := \boldsymbol{U}^\top \boldsymbol{x}_k^{\boldsymbol{\mu}}$ on the intermediate linear subspace $\mathbb{R}^r$ are computed. Subsequently, the DL-ROM is trained using (3) by replacing $\boldsymbol{x}_k^{\boldsymbol{\mu}}$ with the POD coefficients $\boldsymbol{h}_k^{\boldsymbol{\mu}}$. In the online phase, we follow the DL-ROM steps and afterwards recover the FOM solutions from the reconstructed POD coefficients, i.e., $\tilde{\boldsymbol{x}}_k^{\boldsymbol{\mu}} = \boldsymbol{U}\tilde{\boldsymbol{h}}_k^{\boldsymbol{\mu}} = \boldsymbol{U}\boldsymbol{\psi}(\boldsymbol{\xi}(k, \boldsymbol{\mu}; \boldsymbol{\theta}_{\boldsymbol{\xi}}); \boldsymbol{\theta}_{\boldsymbol{\psi}})$. Note, that for the POD-DL-ROM, an adjusted version of the Kolmogorov n-widths holds, due to the linear reduction step introduced by the POD, see [46].

3. Invertible Autoencoders for Dimensionality and Model Reduction

Traditional AE architectures utilize (deep) NNs to parametrize the encoder and decoder and minimize the reconstruction loss (2) to optimize the learnable parameters of the two architectures. However, the encoder is usually non-invertible and, therefore, it cannot be inverted by the decoder, resulting in a loss of information [16]. The loss of information is typically not reduced by increasing the dimension of the manifold, i.e., using more latent variables, and it is correlated with the reconstruction-error plateau firstly observed by [13] in the context of data-driven ROM. To tackle the problem of (non-)invertibility of AE architectures, inspired by [16], we propose an *analytically and numerically invertible* AE (*inv-AE*) architecture capable of going beyond the reconstruction-error plateau of traditional AEs.

The rest of the section is organized as follows: Section 3.1 presents the building blocks of inv-AE, namely the invertible NN layers, while Section 3.2 presents the inv-AEs. Eventually, Section 3.3 introduces a new class of DL-ROMs named invertible (POD) DL-ROMs.

3.1. Invertible Neural Networks and Layers

Similarly to invertible NNs, the inv-AE is composed of multiple invertible layers. Inv-AE uses invertible affine coupling layers [19, 20], but other invertible layers can be simply plugged-in without changing the core principles. Each affine coupling layer is composed of four nonlinear learnable functions, scaling and translating the input $\boldsymbol{x}$:

1. $\boldsymbol{s}_1 : \mathbb{R}^{N/2} \to \mathbb{R}^{N/2}$,
2. $\boldsymbol{s}_2 : \mathbb{R}^{N/2} \to \mathbb{R}^{N/2}$,
3. $\boldsymbol{t}_1 : \mathbb{R}^{N/2} \to \mathbb{R}^{N/2}$, and
4. $\boldsymbol{t}_2 : \mathbb{R}^{N/2} \to \mathbb{R}^{N/2}$,

where N indicates the dimension of the input $\boldsymbol{x}$. Each function is parametrized by NN with learnable parameter vectors $\boldsymbol{\theta}_{\boldsymbol{s}_1}$, $\boldsymbol{\theta}_{\boldsymbol{s}_2}$, $\boldsymbol{\theta}_{\boldsymbol{t}_1}$, $\boldsymbol{\theta}_{\boldsymbol{t}_2}$, respectively. With reference to Figure 3, an affine coupling layer first

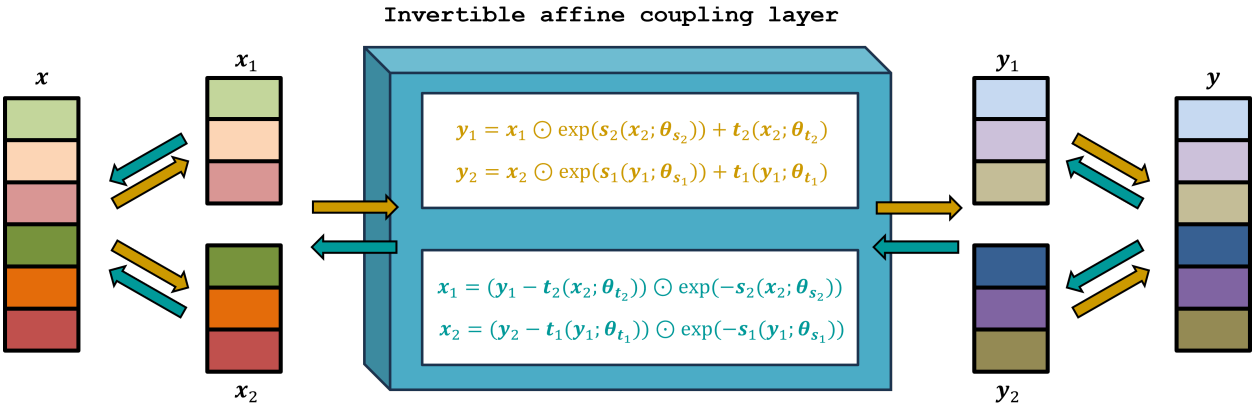


Figure 3. Invertible affine coupling layer. The forward pass through the layer is indicated by dark khaki arrows, while the backward pass by teal arrows.

splits the input $\boldsymbol{x}$ of dimension N in two part of equal dimension, i.e., $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{N/2}$ such that $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^\top$. Then, the transformations $\boldsymbol{s}_1(\cdot; \boldsymbol{\theta}_{\boldsymbol{s}_1})$, $\boldsymbol{s}_2(\cdot; \boldsymbol{\theta}_{\boldsymbol{s}_2})$, $\boldsymbol{t}_1(\cdot; \boldsymbol{\theta}_{\boldsymbol{t}_1})$, and $\boldsymbol{t}_2(\cdot; \boldsymbol{\theta}_{\boldsymbol{t}_2})$ are applied to $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to obtain the outputs $\boldsymbol{y}_1 \in \mathbb{R}^{N/2}$ and $\boldsymbol{y}_2 \in \mathbb{R}^{N/2}$ of the layer. The forward pass through an invertible coupling layer can be written by

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{x}_1 \odot \exp(\boldsymbol{s}_2(\boldsymbol{x}_2); \boldsymbol{\theta}_{\boldsymbol{s}_2}) + \boldsymbol{t}_2(\boldsymbol{x}_2; \boldsymbol{\theta}_{\boldsymbol{t}_2}) \\ \boldsymbol{y}_2 &= \boldsymbol{x}_2 \odot \exp(\boldsymbol{s}_1(\boldsymbol{y}_1); \boldsymbol{\theta}_{\boldsymbol{s}_1}) + \boldsymbol{t}_1(\boldsymbol{y}_1; \boldsymbol{\theta}_{\boldsymbol{t}_1}) \end{aligned} ,$$

where $\odot$ indicates the Hadamard product and the output $\boldsymbol{y}$ is obtained by concatenation of $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, i.e., $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2)^\top \in \mathbb{R}^N$. The coupling layer is analytically invertible, and we can write its backward pass from $\boldsymbol{y}$ to $\boldsymbol{x}$ by

$$\begin{aligned} \boldsymbol{x}_1 &= (\boldsymbol{y}_1 - \boldsymbol{t}_2(\boldsymbol{x}_2; \boldsymbol{\theta}_{\boldsymbol{t}_2})) \odot \exp(-\boldsymbol{s}_2(\boldsymbol{x}_2; \boldsymbol{\theta}_{\boldsymbol{s}_2})) , \\ \boldsymbol{x}_2 &= (\boldsymbol{y}_2 - \boldsymbol{t}_1(\boldsymbol{y}_1; \boldsymbol{\theta}_{\boldsymbol{t}_1})) \odot \exp(-\boldsymbol{s}_1(\boldsymbol{y}_1; \boldsymbol{\theta}_{\boldsymbol{s}_1})) . \end{aligned}$$

To prevent numerical instabilities, we clamp and scale the exponential function as proposed in [29]. Therefore, we obtain

$$\begin{aligned} \exp(\boldsymbol{s}_2(\boldsymbol{x}_2); \boldsymbol{\theta}_{\boldsymbol{s}_2}) &= \exp\left(\arctan\left(\boldsymbol{s}_2(\boldsymbol{x}_2); \boldsymbol{\theta}_{\boldsymbol{s}_2}\right)\right) , \\ \exp(\boldsymbol{s}_1(\boldsymbol{y}_1); \boldsymbol{\theta}_{\boldsymbol{s}_1}) &= \exp\left(\arctan\left(\boldsymbol{s}_1(\boldsymbol{y}_1); \boldsymbol{\theta}_{\boldsymbol{s}_1}\right)\right) . \end{aligned} \tag{4}$$

3.2. From Invertible NNs to Invertible AEs

Each invertible layer does not perform any form of dimensionality reduction because to achieve (analytical) invertibility, the layers require the same dimensionality of inputs and outputs, i.e., $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$. Therefore, using an invertible NN composed of multiple affine coupling layers is not sufficient for building an invertible

AE. However, to achieve dimensionality reduction, we use a technique proposed in [16]. We assume $\boldsymbol{x}$ to be the input of the first layer and $\boldsymbol{y}$ to be the output of the last layer of the invertible NN. In particular, after applying the forward pass transformations, we can mask with zeros the output of the final layer $\boldsymbol{y}$ up to an arbitrary latent dimension to obtain $\boldsymbol{z}$:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{\phi}(\boldsymbol{x}; \boldsymbol{\theta}_{\boldsymbol{\phi}}) \\ \boldsymbol{z} &= \texttt{zero-masking}(\boldsymbol{y})\,. \end{aligned} \tag{5}$$

In particular, given the vector $\boldsymbol{y}$, the zero-masking procedure is defined as follows:

$$\begin{aligned} \boldsymbol{y} &= [y_1, \dots, y_n, y_{n+1}, \dots, y_N] \\ &\downarrow \texttt{zero-masking} \\ \boldsymbol{z} &= [y_1, \dots, y_n, 0, \dots, 0]\,, \end{aligned} \tag{6}$$

where N indicates the dimensionality of $\boldsymbol{y}$, and n the dimension of the reduced space. The zero $(N-n)$-elements of $\boldsymbol{z}$ have no attached gradient to disable backpropagation thought the corresponding weights and biases. Zero masking is an effective dimensionality-reduction mechanism, forcing the encoder network to compress data into n-dimensional vectors, while maintaining the output dimensionality requirement imposed by the invertible layers. Then, the zero-masked vector $\boldsymbol{z}$ is passed through the backward pass to recover the reconstruction of the input $\hat{\boldsymbol{x}}$:

$$\hat{\boldsymbol{x}} = \boldsymbol{\psi}(\boldsymbol{z}; \boldsymbol{\theta}_{\boldsymbol{\psi}})\,, \tag{7}$$

where $\boldsymbol{\psi} = \boldsymbol{\phi}^{-1}$ (see Figure 1). The zero-masking promotes the compression of the data into a limited number of relevant dimensions, making the invertible NN and inv-AE. Analogously to traditional AEs, we train our inv-AE to minimize the reconstruction loss

$$\mathcal{L}(\boldsymbol{\theta}_{\boldsymbol{\phi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}) = \frac{1}{M} \sum_{i=1}^{M} ||\boldsymbol{x}^{(i)} - \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}_{\boldsymbol{\phi}}); \boldsymbol{\theta}_{\boldsymbol{\psi}})||_2^2\,, \tag{8}$$

where encoder and decoder correspond to the forward and backward pass through the invertible layers, and that $\boldsymbol{\theta}_{\boldsymbol{\psi}} = \boldsymbol{\theta}_{\boldsymbol{\phi}}$.

In addition to the reconstruction loss, it is possible to utilize additional terms such a zero-masking regularization loss [16], or a Lipschitz-constant regularization loss [35] to improve numerical stability and convergence properties of the architecture. However, in our case, the choice of an affine coupling layer with clamped exponential (see Equation (4)) is sufficient for preventing numerical instabilities. As shown in Figure C.14 in Appendix C, spectral normalization [38] is only mildly beneficial for our inv-AE, demonstrating the intrinsic numerical stability of our architecture. Implementation details of the proposed architecture are discussed in Appendix B.

3.3. Invertible (POD) Deep Learning-based ROMs

In this work, we do not only test the dimensionality-reduction capabilities of the inv-AE, but we also empirically evaluate the advantages of inv-AE in the context of projection-based ROM. Therefore, we propose to enhance the DL-ROM and the POD-DL-ROM (see Section 2.2) with our inv-AE replacing the convolutional AE. We call these two approaches as *inv-DL-ROM* and *POD-inv-DL-ROM*. Analogously to DL-ROM, inv-DL-ROM is composed of an inv-AE and a feedforward NN $\boldsymbol{\xi}$ predicting the latent vector $\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}$ from the timestep k and the parameter vector $\boldsymbol{\mu}$:

$$\begin{aligned} \tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}} &= \boldsymbol{\xi}(k, \boldsymbol{\mu}; \boldsymbol{\theta}_{\boldsymbol{\xi}}) \\ \tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}} &= \texttt{zero-padding}(\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}) \\ \tilde{\boldsymbol{x}}_k^{\boldsymbol{\mu}} &= \boldsymbol{\psi}(\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}; \boldsymbol{\theta}_{\boldsymbol{\xi}})\,, \end{aligned}$$

where $\boldsymbol{\psi} = \boldsymbol{\phi}^{-1}$ and $\boldsymbol{\theta}_{\boldsymbol{\psi}} = \boldsymbol{\theta}_{\boldsymbol{\phi}}$. In this case, a zero-padding procedure is used to have the dimensionality of $\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}$ equal to $\tilde{\boldsymbol{x}}_k^{\boldsymbol{\mu}}$ as required by the inv-AE architecture and it is defined as follows:

$$\begin{aligned} \tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}} &= [z_1, \dots, z_n] \\ &\downarrow \texttt{zero-padding} \\ \tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}} &= [z_1, \dots, z_n, 0, \dots, 0]\,. \end{aligned} \tag{9}$$

Zero-padding involves concatenating $N - n$ zeros to the output of the the model $\boldsymbol{\xi}$ to obtain a vector of again dimension N. On the other side, POD-inv-DL-ROM simply adds the POD (re)projection of the POD coefficients to the snapshots:

$$\begin{aligned}
\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}} &= \boldsymbol{\xi}(k, \boldsymbol{\mu}; \boldsymbol{\theta}_{\boldsymbol{\xi}}) \\
\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}} &= \texttt{zero-padding}(\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}) \\
\tilde{\boldsymbol{h}}_k^{\boldsymbol{\mu}} &= \boldsymbol{\psi}(\tilde{\boldsymbol{z}}_k^{\boldsymbol{\mu}}; \boldsymbol{\theta}_{\boldsymbol{\xi}}) \\
\tilde{\boldsymbol{x}}_k^{\boldsymbol{\mu}} &= U \tilde{\boldsymbol{h}}_k^{\boldsymbol{\mu}} ,
\end{aligned}$$

where U is the SVD matrix computed on the train set and $\tilde{\boldsymbol{h}}_k^{\boldsymbol{\mu}}$ is the reconstructed POD coefficient using $\boldsymbol{\xi}$ and $\boldsymbol{\psi}$. Due to the parameter sharing of encoder and decoder, (POD) inv-DL-ROM does not require the latent-space regularization of traditional DL-ROM architectures (second term of Equation (3)) and can be trained with a simplified training objective

$$\begin{aligned}
\mathcal{L}_{\text{inv-DL-ROM}}(\boldsymbol{\theta}_{\boldsymbol{\phi}}, \boldsymbol{\theta}_{\boldsymbol{\psi}}, \boldsymbol{\theta}_{\boldsymbol{\xi}}) &= \frac{1}{M} \sum_{i=1}^{M} ||\boldsymbol{x}^{(i)} - \tilde{\boldsymbol{x}}^{(i)}||_2^2 \\
&= \frac{1}{M} \sum_{i=1}^{M} ||\boldsymbol{x}^{(i)} - \boldsymbol{\psi}(\boldsymbol{\xi}(k^{(i)}, \boldsymbol{\mu}^{(i)}; \boldsymbol{\theta}_{\boldsymbol{\xi}}); \boldsymbol{\theta}_{\boldsymbol{\psi}})||_2^2 .
\end{aligned} \tag{10}$$

While the DL-ROM objective in Equation (3) requires tuning of the coefficient α to balance the contribution of the two terms of the loss function, the parameters of inv-DL-ROM are optimized solely to reconstruct the data $\boldsymbol{x}$ from $\boldsymbol{\mu}$ and t.

4. Numerical Results

To investigate the capabilities on the invertible AEs, we introduce three numerical examples, namely a parametric 1-dimensional Burgers' equation (Section 4.1), a parametric 2-dimensional fluid flow around an obstacle with variable geometry (Section 4.2), and a parametric 3-dimensional Korteweg-de Vries equation (Section 4.3). Using these three numerical examples, we aim to test *(i)* the dimensionality reduction capabilities of (POD)inv-AE and *(ii)* the accuracy of the resulting ROM, namely (POD)inv-DL-ROM. For reconstruction, we compare different architectures:

- *POD* – proper orthogonal decomposition,
- *conv. AE* – convolutional AE with $\sim 0.25M$ parameters,
- *fully-conn. AE* – fully-connected AE and with $\sim 2.5M$, parameters.
- *inv-AE* – invertible AE with $\sim 2.5M$ parameters, and
- *inv-AE (light)* – invertible AE with $\sim 1.25M$ parameters.

For reduction, instead, we compare:

- *POD-NN* – proper orthogonal decomposition with neural network-based surrogate model [47],
- *DL-ROM (conv. AE)* – DL-ROM with convolutional AE,
- *DL-ROM (fully-conn. AE)* – DL-ROM with fully-connected AE,
- *inv-DL-ROM* – DL-ROM with inv-AE, and
- *inv-DL-ROM (light)* – DL-ROM with inv-AE (light).

For fairness of comparison, all the AEs share the same number of layers, and the same architecture for $\boldsymbol{\xi}(k, \boldsymbol{\mu}; \boldsymbol{\theta}_{\boldsymbol{\xi}})$, composed of 2 hidden layers with 256 neurons each and leaky ReLU activations. For the 2-dimensional and 3-dimensional test cases, a POD step with $N \gg r \gg n$ is included before the AEs to reduced the data dimensionality without compromising the accuracy of the reconstructions and consequently reducing the number of learnable parameters of the neural network-based models. The complete architectural and implementation details are discussed in Appendix B.

After training the models using the train set $\boldsymbol{X}_{\text{train}}$, we evaluate their performance on the test set $\boldsymbol{X}_{\text{test}}$, where we set $N_{\text{test}} := |\mathcal{P}_{\text{test}}|$.

$$\boldsymbol{X}_{\text{test}} = \begin{bmatrix} | & | & \cdots & | & | \\ \boldsymbol{x}_1^{\boldsymbol{\mu}_1} & \boldsymbol{x}_2^{\boldsymbol{\mu}_1} & \cdots & \boldsymbol{x}_{N_t-1}^{\boldsymbol{\mu}_{N_{\text{test}}}} & \boldsymbol{x}_{N_t}^{\boldsymbol{\mu}_{N_{\text{test}}}} \\ | & | & \cdots & | & | \end{bmatrix} \in \mathbb{R}^{N \times (N_t \times N_{\text{test}})}.$$

In particular, we utilize the mean reconstrution error given by

$$\frac{1}{|\boldsymbol{X}_{\text{test}}|} \sum_{\boldsymbol{\mu} \in \mathcal{P}_{\text{test}}} \sqrt{\frac{\sum_{k=1}^{N_t} ||\boldsymbol{x}_k^{\boldsymbol{\mu}} - \hat{\boldsymbol{x}}_k^{\boldsymbol{\mu}}||_2^2}{\sum_{k=1}^{N_t} ||\boldsymbol{x}_k^{\boldsymbol{\mu}}||_2^2}}, \tag{11}$$

and the mean reduction error

$$\frac{1}{|\boldsymbol{X}_{\text{test}}|} \sum_{\boldsymbol{\mu} \in \mathcal{P}_{\text{test}}} \sqrt{\frac{\sum_{k=1}^{N_t} ||\boldsymbol{x}_k^{\boldsymbol{\mu}} - \tilde{\boldsymbol{x}}_k^{\boldsymbol{\mu}}||_2^2}{\sum_{k=1}^{N_t} ||\boldsymbol{x}_k^{\boldsymbol{\mu}}||_2^2}}, \tag{12}$$

where $\boldsymbol{x}_k^{\boldsymbol{\mu}}$ indicate the discretized FOM snapshot at the timestep k, $\hat{\boldsymbol{x}}_k^{\boldsymbol{\mu}}$ the reconstructed snapshot when passed through encoder and decoder, and $\tilde{\boldsymbol{x}}_k^{\boldsymbol{\mu}}$ the decoded solution of the DL-ROM. The reconstruction error measures the best-possible error made by the corresponding chosen autoencoder realization, while the reduction error shows the error of the corresponding ROM.

4.1. Burgers' Equation

We study a time-dependent 1-dimensional Burgers' equation as introduced in [13] given by

$$\begin{aligned} \frac{\partial x(y,t;\boldsymbol{\mu})}{\partial t} + \frac{\partial f(x(y,t;\boldsymbol{\mu}))}{\partial y} - 0.02 \exp^{\mu_2 y} = 0, \quad & \forall y \in [0, 100], \forall t \in (t_0, t_f], \\ x(0,t;\boldsymbol{\mu}) = \mu_1, \quad & \forall t \in [t_0, t_f], \\ x(y,0) = 1, \quad & \forall y \in [0, 100], \end{aligned}$$

where the flux is $f(x) = 0.5x^2$ and $\boldsymbol{\mu} = (\mu_1, \mu_2)$ is the parameter vector, t indicates the time variable, and y the spatial variable. Analogously to [13], we derive the FOM snapshpts by numerically solving the Burgers' equation from $t_0 = 0.0$, to $t_f = 35$, where we use equidistant time steps of $\Delta t = 0.07$s resulting in $N_t = 500$ and discretize in time with the implicit Euler scheme. For the spatial discretization, we apply a Godunov's scheme with $N = 256$ spatial degrees of freedom. The parameter domain is $\mathcal{P} = [4.25, 5.5] \times [0.015, 0.03]$, where we choose a training set of parameters $\mathcal{P}_{\text{train}}$ by discretizing the parameter intervals into $10 \times 8 = 80$ bins to replicate [13]. The training set X_{train} is composed of the 80 FOM solutions generated for the parameters $\mathcal{P}_{\text{train}}$. Further, we keep 8 of the 80 FOM solutions for the validation set X_{valid}, while the test set X_{test} is composed of 8 FOM solutions generated with 8 unseen parameter values.. In this way, we are able to test the generalization capabilities of the proposed AE architectures on unseen instances of the parameter $\boldsymbol{\mu}$.

In this first experiment, we compare the reconstruction error achieved by our inv-AE and the convolutional AE introduced in [13] for different manifold dimensions $\in \{1, 2, 3, 4, 5, 10, 20, 50, 100, 150, 200\}$. In the second experiments, we compare instead the reduction error achieved by inv-DL-ROM with respect to DL-ROM (with the convolution AE of [13] and with a fully connected AE with similar amounts of parameters) for different dimensions of the latent space $\in \{1, 2, 3, 4, 5, 10, 20\}$. For fairness of the comparison, we train the different models for 1000 epochs with the AdamW optimizer [48]. In addition, we use the same learning rate of $1e^{-4}$ and weight decay of $1e^{-4}$, and we employ early stopping if the loss on the validation set does not

decrease after 500 epochs of training. Same settings were used in [13]. After training the models using two independent seeds, we analyze the reconstruction errors achieved by the different AE architectures and the reduction errors achieved by the different DL-ROM architectures.

In Figure 4 we show the reconstruction errors obtained with the different architectures with respect to the chosen dimension of the trial manifold and in Table A.1 we report the best values. We compare the reconstruction error of the inv-AEs with POD, the convolutional AE from [13], and a fully connected AE. The blue-shaded area highlights values up to manifold dimension equal to 50, while the red-shaded area showed the region where the POD achieves comparable or superior performance to the AEs. As reported in

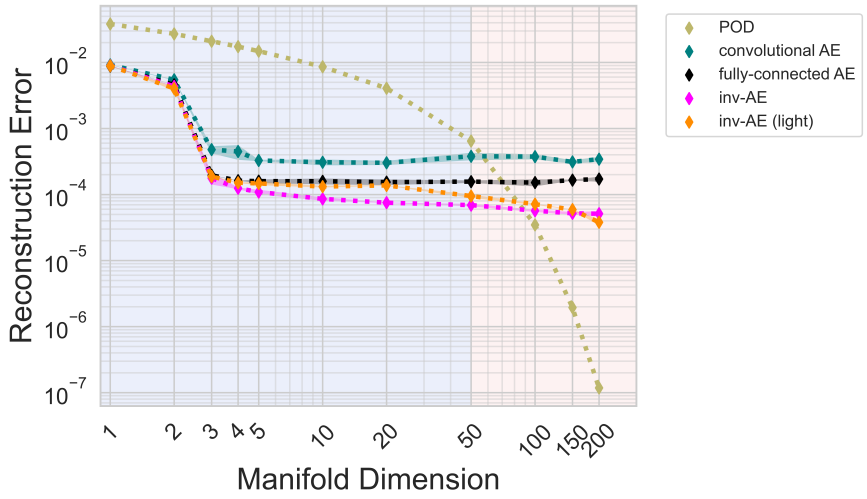


Figure 4. Reconstruction errors (11) calculated on the test set $\boldsymbol{X}_{\text{test}}$ for different dimensions of the trial manifold. The dashed line represents the mean and the shaded area the standard deviation across two different random seeds.

[13], the convolutional AE exhibits a plateau of the reconstruction error when the dimension of the manifold is 5. A similar plateau is achieved by the fully connected AE, while the reconstruction error of our inv-AEs continuously decreases with the increment of the manifold dimension and it is consistently smaller than the one of the convolutional AE for a manifold dimension > 2. The POD has higher reconstruction error than both variants of the AEs for manifold dimensions smaller that 50, while above 100 it outperforms all AEs.

Further, we report in Figure 5 the reduction errors achieved by DL-ROMs and inv-DL-ROMs for different dimension of the trial manifold $\in \{1, 2, 3, 4, 5, 10, 20\}$ and in Table A.2 we report the best values. The dimensions of the manifold were chosen to balance reduction capabilities and accuracy of the learned ROMs. By solely replacing the convolutional AE with the inv-AE, the inv-DL-ROM is capable of improving the reduction errors approximately by a factor 2.5. We also show that all the AEs achieves superior performance to POD-NN.

Eventually, in Figure 6 we show a representative example of reconstruction errors of the different AE architectures for a manifold dimension of 20, where it is possible to notice the improved reconstruction capabilities of the inv-AEs compared to convolutional and fully-connected AEs.

4.2. Flow Around an Obstacle

As a second challenging and high-dimensional test case, we use the fluid flow around an obstacle. We consider a 2-dimensional channel of dimension $[0, 10] \times [0, 2]$ with a circular obstacle placed in $(1, 1)$ with a radius of 0.2. The obstacle can be deformed along the x-axis on its left and right side by $\gamma_l \in [0.2, 0.6]$ and $\gamma_r \in [0.2, 1.0]$, respectively. We also assume incompressible flow, whose dynamics is described by the

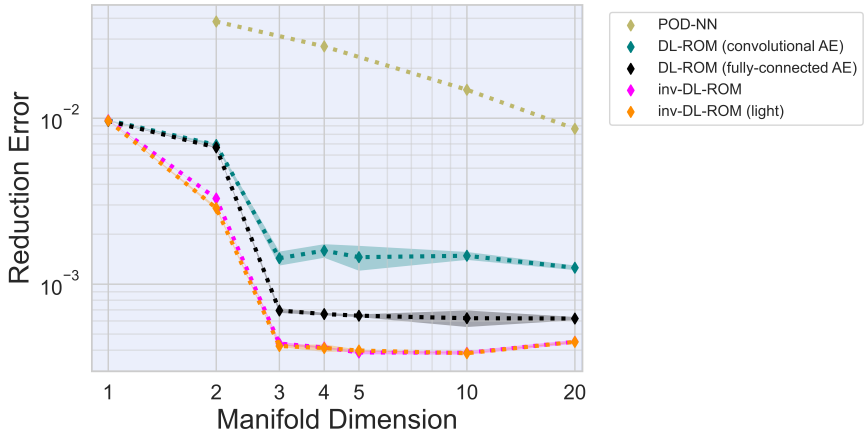


Figure 5. Reduction errors (12) calculated on the test set $\boldsymbol{X}_{\text{test}}$ for different dimensions of the trial manifold of DL-ROMs and inv-DL-ROMs. The dashed line represents the mean and the shaded area the standard deviation across two different random seeds.

unsteady Navier-Stokes equations

$$\begin{cases} \dfrac{\partial \boldsymbol{v}(\boldsymbol{y},t)}{\partial t} - \mu \Delta \boldsymbol{v}(\boldsymbol{y},t) + (\boldsymbol{v}(\boldsymbol{y},t)\cdot\nabla)\boldsymbol{v}(\boldsymbol{y},t) + \nabla p(\boldsymbol{y},t) = 0 & \text{in } \Omega(\boldsymbol{\mu})\times(t_0,t_f) \\ \text{div } \boldsymbol{v}(\boldsymbol{y},t) = 0 & \text{in } \Omega(\boldsymbol{\mu})\times(t_0,t_f) \\ \boldsymbol{v}(\boldsymbol{y},t) = \boldsymbol{v}_{\text{in}}(\boldsymbol{y},t;\boldsymbol{\mu}) & \text{on } \Gamma_{\text{in}}\times(t_0,t_f) \\ \boldsymbol{v}(\boldsymbol{y},t) = \boldsymbol{0} & \text{on } \Gamma_{\text{obs}}(\boldsymbol{\mu})\times(t_0,t_f) \\ \boldsymbol{v}(\boldsymbol{y},t)\cdot\boldsymbol{n}(\boldsymbol{y}) = 0 & \text{on } \Gamma_{\text{walls}}\times(t_0,t_f) \\ (\mu\nabla\boldsymbol{v}(\boldsymbol{y},t) - p(\boldsymbol{x},t)I)\boldsymbol{n}(\boldsymbol{y})\cdot\boldsymbol{\tau}(\boldsymbol{x}) = 0 & \text{on } \Gamma_{\text{walls}}\times(t_0,t_f) \\ (\mu\nabla\boldsymbol{v}(\boldsymbol{y},t) - p(\boldsymbol{y},t)I)\boldsymbol{n}(\boldsymbol{x}) = 0 & \text{on } \Gamma_{\text{out}}\times(t_0,t_f) \\ \boldsymbol{v}(\boldsymbol{y},0) = \boldsymbol{0} & \text{in } \Omega(\boldsymbol{\mu})\times\{t=t_0\} \end{cases} \tag{13}$$

where $\Omega(\boldsymbol{\mu}) = \Omega(\gamma_l,\gamma_r)$ is the domain of interest depending on the parameters γ_l and γ_r, $t_0 = 0$ is the initial time, $t_f > 0$ is the final time, $\boldsymbol{y} = (y_1, y_2)$ indicates the spatial variables, $\boldsymbol{v}(\boldsymbol{y},t) : \Omega(\gamma_l,\gamma_r)\times[0,t_f] \to \mathbb{R}^2$ is the velocity, $p(\boldsymbol{y},t) : \Omega(\gamma_l,\gamma_r)\times[0,t_f] \to \mathbb{R}$ is the pressure, $\boldsymbol{v}_{\text{in}}(\boldsymbol{y},t;\boldsymbol{\mu}) : \Gamma_{\text{in}}\times[0,t_f] \to \mathbb{R}^2$ is the time and parameters-dependent inflow datum, $\mu = 0.01$ is the kinematic viscosity, $\boldsymbol{n}(\boldsymbol{y})$ and $\boldsymbol{\tau}(\boldsymbol{y})$ are the normal and tangential tensors. The fluid starts with zero velocity at $t_0 = 0$ and enters the domain from the left boundary with intensity $\gamma_{\text{in}} \in [1.0, 10.0]$ and angle of attack $\alpha_{\text{in}} \in [-1.0, 1.0]$ as defined in

$$\boldsymbol{v}((0,y_2),t) = (\gamma_{\text{in}}\cos\alpha_{\text{in}}, y_2(2-y_2)\gamma_{\text{in}}\sin\alpha_{\text{in}})\,,$$

where the parabolic profile $y_2(2-y_2)$ is used to prevent discontinuities as in [49]. We use on a mesh with $N = 80592$ degrees of freedom for the velocities and solve (13) through the incremental Chorin-Temam reconstruction method for $t_f = 5$s, and $\Delta t = 0.05$s using the finite element solver in `fenics` [50] (see Figure 7).

Our goal is to recover the velocity field $\boldsymbol{v}$ for different values of the parameters $\boldsymbol{\mu}(\gamma_l,\gamma_r,\gamma_{\text{in}},\alpha_{\text{in}})$. The parameter domain is $\mathcal{P} = [0.2,0.6]\times[0.2,1.0]\times[1.0,10.0]\times[-1.0,1.0]$, corresponding to Reynolds numbers in the range $[40,1600]$, where we choose a training set of parameters $\mathcal{P}_{\text{train}}$ by randomly sampling 80 times from $\mathcal{P}$, the validation set of parameters $\mathcal{P}_{\text{valid}}$ by randomly sampling 10 times from $\mathcal{P}$, and the test set of parameters $\mathcal{P}_{\text{test}}$ by randomly sampling 10 times from $\mathcal{P}$. Therefore, the training set X_{train} is composed of the 80 FOM solutions generated for the parameters $\mathcal{P}_{\text{train}}$. Further, we keep 10 FOM solutions for the validation set X_{valid}, while the test set X_{test} is composed of 10 FOM solutions generated with 10 unseen parameter values.

Due to the high dimensionality of the fluid velocities, we first apply the POD to the snapshots, treated as a vector, to reduce the dimensionality from 80592 to $r = 512$, namely 256 dimensions per velocity

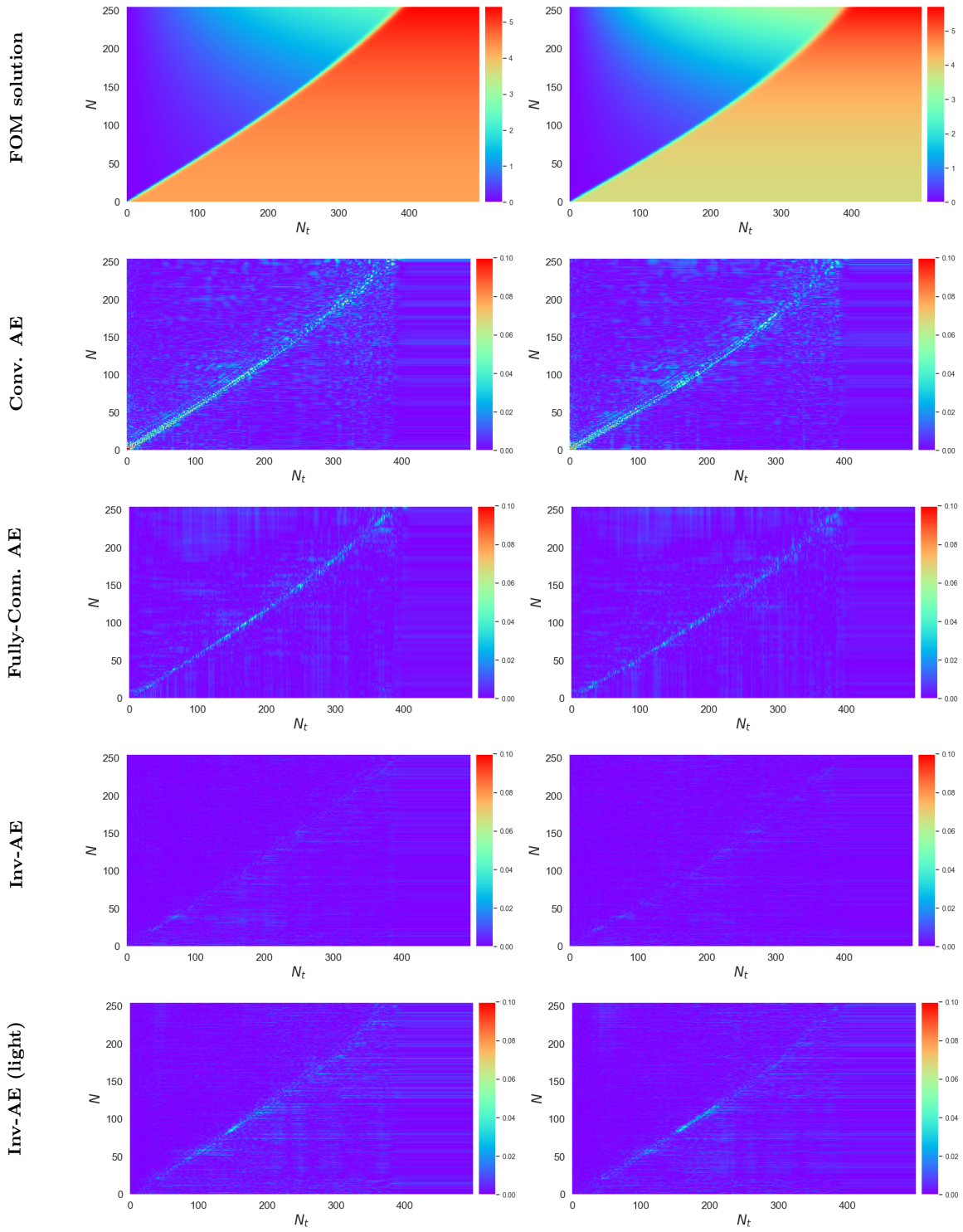


Figure 6. Qualitative comparison of the different architecture for a fixed manifold dimension of 20. The first row shows two FOM solutions from the test set, while the remaining rows the absolute reconstruction error achieved by the different AEs.

component, posing a lower bound on the reconstruction error of $3.0e^{-4}$. It is worth highlighting that with the POD pre-processing step we do not aim to find the smallest number of basis functions, but simply to decrease the snapshot dimensionality to the point in which we can use AEs with limited numbers of learnable parameters that are less computationally demanding to train and do not require larger amounts of data. A

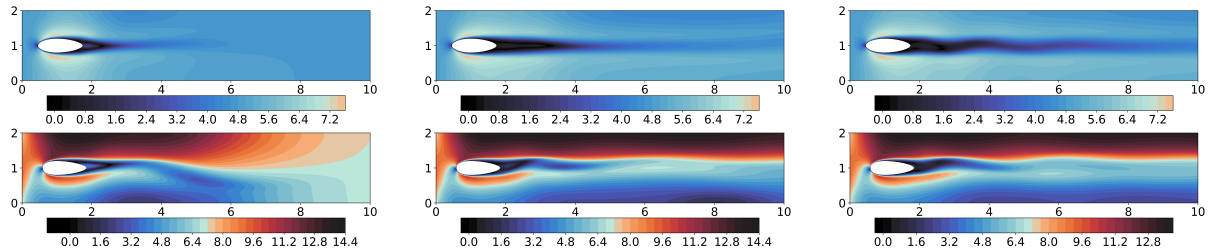


Figure 7. Examples of magnitude of the velocity fields of the flow around an obstacle at different timesteps and parameter instances.

similar procedure was used in [40]. From Figure 8, it is possible to notice that no AEs is capable of getting close to the lower bound posed by the POD. Therefore, the choice of considering $r = 512$ dimension is not detrimental to the AE and can be considered suitable for this numerical example.. We compare our inv-AEs with the convolutional AE introduced in [5] for a similar test case and a fully connected AE. stwhere We train the different AE models for 1500 epochs with AdamW optimizer with different manifold dimensions $\in \{1, 2, 3, 4, 5, 10, 20, 50, 100\}$. For all the models, we use learning rate equal to $1e^{-4}$, weight decay equal to $1e^{-4}$, and we employ early stopping if the validation loss does not decrease after 500 epochs of training.

In Figure 8, we show the reconstruction error (11) obtained using the convolutional AE, fully connected AE, inv-AEs, and POD, and in Table A.3 we report the best values. In addition, we report the reconstruction error when truncating the POD to $r = 512$. Even in this case, both inv-AE versions reduce the reconstruction error compared to the convolutional AE and fully connected AE. Moreover, for both of the inv-AE, we observe an almost monotonic decay of the reconstruction error, which is not the case for the convolutional or fully connected AE.

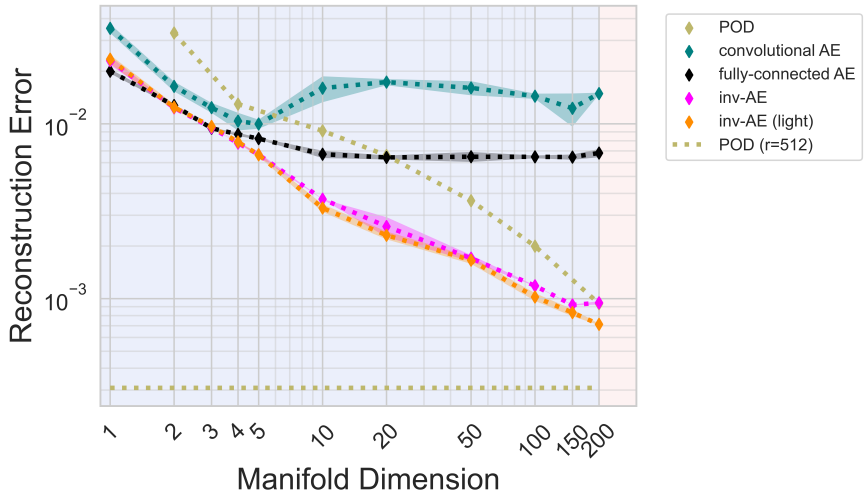


Figure 8. Reconstruction errors (11) for different dimensions of the trial manifold. The dashed line represents the mean and the shaded area the standard deviation across two different random seeds.

Finally, in Figure 9, we show the reduction errors (12) of POD-DL-ROMs and the POD-inv-DL-ROMs and in Table A.4 we report the best values. Even in this case, by simply replacing the convolutional or fully connected AE with the inv-AEs, we are capable of mildly improving the reduction error of the POD-DL-ROM architecture.

4.3. Korteweg-de Vries

As a third numerical example, we consider a 3-dimensional Korteweg-de Vries (KdV) equation. The KdV equation is a nonlinear dispersive PDE describing a wide range of physical systems, including internal ocean

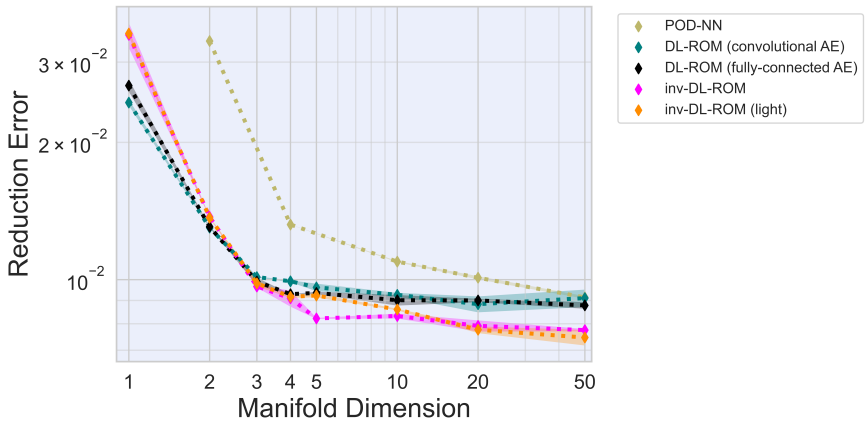


Figure 9. Reduction errors (12) of DL-ROMs and inv-DL-ROMs. The dashed line represents the mean and the shaded area the standard deviation across two different random seeds.

waves, plasma physics, lattice dynamics, and nonlinear optics [51]. The KdV can be written as follows:

$$\frac{\partial x(\boldsymbol{y},t)}{\partial t}+\frac{1}{2}\Big(\frac{\partial x(\boldsymbol{y},t)^2}{\partial y_1}+\frac{\partial x(\boldsymbol{y},t)^2}{\partial y_2}+\frac{\partial x(\boldsymbol{y},t)^2}{\partial y_3}\Big)+\Big(\frac{\partial^3 x(\boldsymbol{y},t)}{\partial y_1^3}+\frac{\partial^3 x(\boldsymbol{y},t)^2}{\partial y_2^3}+\frac{\partial^3 x(\boldsymbol{y},t)^2}{\partial y_3^3}\Big) = -\mu f(x(\boldsymbol{y},t))\,, \quad (14)$$

where $x(\boldsymbol{y},t) = x(y_1, y_2, y_3)$ is a scalar field defined on a three-dimensional periodic domain, μ is the hyper-diffusivity parameter of interest and

$$f(x(\boldsymbol{y},t)) = \frac{\partial^4 x(\boldsymbol{y},t)}{\partial y_1^4} + \frac{\partial^4 x(\boldsymbol{y},t)}{\partial y_2^4} + \frac{\partial^4 x(\boldsymbol{y},t)}{\partial y_3^4}\,. \quad (15)$$

We obtain the FOM snapshots by solving (14) for different values of the parameter $\mu \in [0.0005, 0.05]$ for 40 time steps with $\Delta t = 0.05$s using `Exponax` [52]. `Exponax` employs a Fourier pseudo-spectral discretization in space combined with an exponential time-differencing Runge-Kutta scheme for temporal integration. The 3-dimensional spatial domain is discretized in $N = 100$ bins per dimension, resulting in a one million-dimensional snapshots. We compute 100 FOM solutions in total, which are distributed into disjoint sets for training, validation and testing. The training set X_{train} is composed of the 80 FOM solutions. Further, we keep 10 of the 100 FOM solutions for the validation set X_{valid}, while the test set X_{test} is composed of the remaining 10 FOM solutions generated with 10 unseen parameter values. Examples of test solutions can be found in Figure 10.

Similarly to the previous numerical example, we first apply POD to the snapshots to reduce the dimensionality from one million to $r = 512$, posing a lower bound on the reconstruction error of $1.92e^{-7}$. Again, it is worth noting that with the POD pre-processing step we simply want to decrease the snapshot dimensionality to be able to use light-weight AEs. We compare our inv-AEs with the convolutional AE introduced in [5] and a fully connected AE. We train the different AE models for 1000 epochs with AdamW optimizer and manifold dimensions $\in \{1, 5, 10, 50, 100\}$. For all the models, we use a learning rate of $1e^{-4}$, weight decay equal to $1e^{-4}$, and we employ early stopping if the validation loss does not improve for 500 consecutive epochs.

In Figure 11, we show the reconstruction error (11) obtained using the convolutional AE, the fully connected AE, the inv-AEs, and the POD, and in Table A.5 we report the best values. In addition, we report the reconstruction error when truncating the POD to $r = 512$. Also in this case, both inv-AE variants reduce the reconstruction error compared to the convolutional AE and fully connected AE. Moreover, for both of the inv-AE, we observe an almost monotonic decay of the reconstruction error, which is neither the case for the convolutional nor the fully connected AE.

Eventually, in Figure 12, we show the reduction errors (12) of POD-DL-ROMs and the POD-inv-DL-ROMs, while in Table A.6 we report the best values. Even in this case, by simply replacing the convolutional or fully connected AE with the inv-AEs, we are capable of significantly improving the reduction error of the POD-DL-ROM architecture.

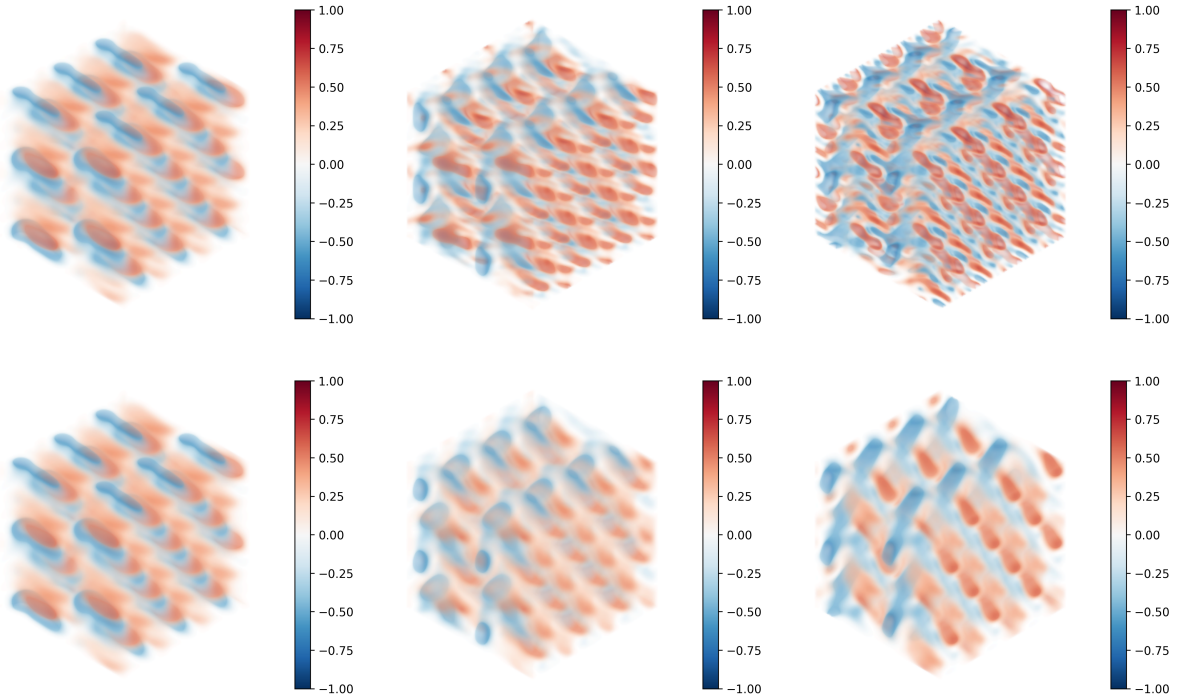


Figure 10. Examples of solution snapshots of the Korteweg-de Vries equation for the two different instances of the parameter and timesteps.

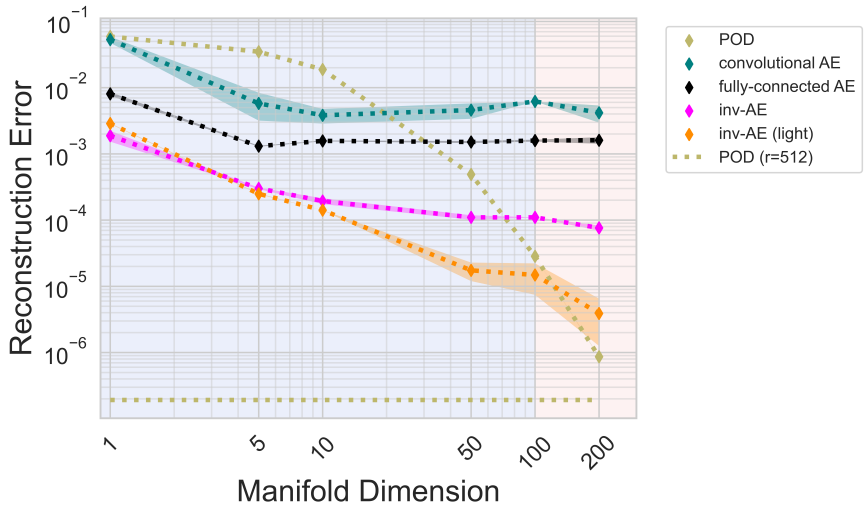


Figure 11. Reconstruction errors (11) for different dimensions of the trial manifold. The dashed line represents the mean and the shaded area the standard deviation across two different random seeds.

Eventually, in Figure 13 we show the reconstruction errors of the inv-AE for increasing manifold dimensions, where we highlight the decrease trend of the absolute error.

5. Conclusion

In this work, we propose an invertible AE architecture, named inv-AE, that allows for solving challenging dimensionality reduction problems. Inv-AE relies on *(i)* an invertible neural network to effectively learn an

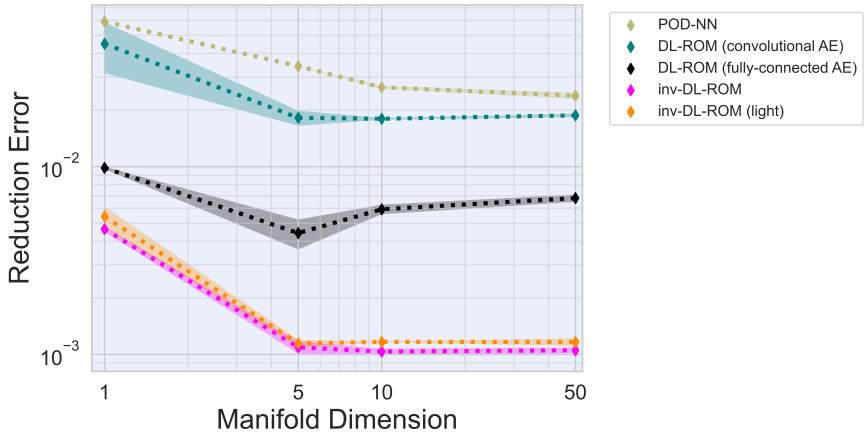


Figure 12. Reduction errors (12) of DL-ROMs and inv-DL-ROMs. The dashed line represents the mean and the shaded area the standard deviation across two different random seeds.

analytically invertible encoder-decoder scheme, on *(ii)* affine coupling layers to ensure numerical stability, and *(iii)* parameter sharing between encoder and decoder to reduce the number of learnable parameters. Moreover, we propose an integration of the inv-AE with popular data-driven MOR strategies such as DL-ROM and POD-DL-ROM, that we named inv-DL-ROM and POD-inv-DL-ROM.

In numerical experiments dealing with two challenging classes of high-dimensional problems, namely the Burgers' equation, the flow around an obstacle, and a Korteweg-de Vries equation, we show that inv-AE is capable of achieving a lower reconstruction error than traditional convolutional and fully connected AEs, and that its use in inv-DL-ROM and POD-inv-DL-ROM is beneficial for improving the reduction error of DL-ROM and POD-DL-ROM. Throughout our results, we show that inv-AE can mitigate the reconstruction-error plateau of traditional AEs architectures and the quality of its reconstruction gradually improves with the increment of the manifold dimension. However, the reduction error does not necessarily improves in the same way. This limitation might be intrinsic to the DL-ROM architecture and improving upon is an interesting future research direction.

While the inv-AE is analytically invertible by construction, the choice of the affine coupling layers is critical for its numerical invertibility. We experience that our affine coupling layers with clamping are not prone to training instabilities and the use of spectral normalization (see Appendix C) appears to be only mildly beneficial. Future research will investigate these properties of the inv-AE to derive theoretical results and bounds for the proposed architectures.

The current Inv-AE architecture relies on fully connected neural networks as the basic building blocks of the affine coupling transformations. Fully connected layers do not explicitly exploit any spatial structure of the (discretized) physical domain. This limitation may become particularly relevant when considering problems characterized by complex geometries, unstructured meshes, or varying spatial resolutions. In such settings, extending the reversible coupling blocks to incorporate more advanced architectures, such as graph neural networks, represents a promising research direction toimprove the scalability and generalization capabilities of inv-AE for large-scale problems.

Acknowledgements

NB acknowledges the Project "Reduced Order Modeling and Deep Learning for the real- time approximation of PDEs (DREAM)" (Starting Grant No. FIS00003154), funded by the Italian Science Fund (FIS) - Ministero dell'Università e della Ricerca.

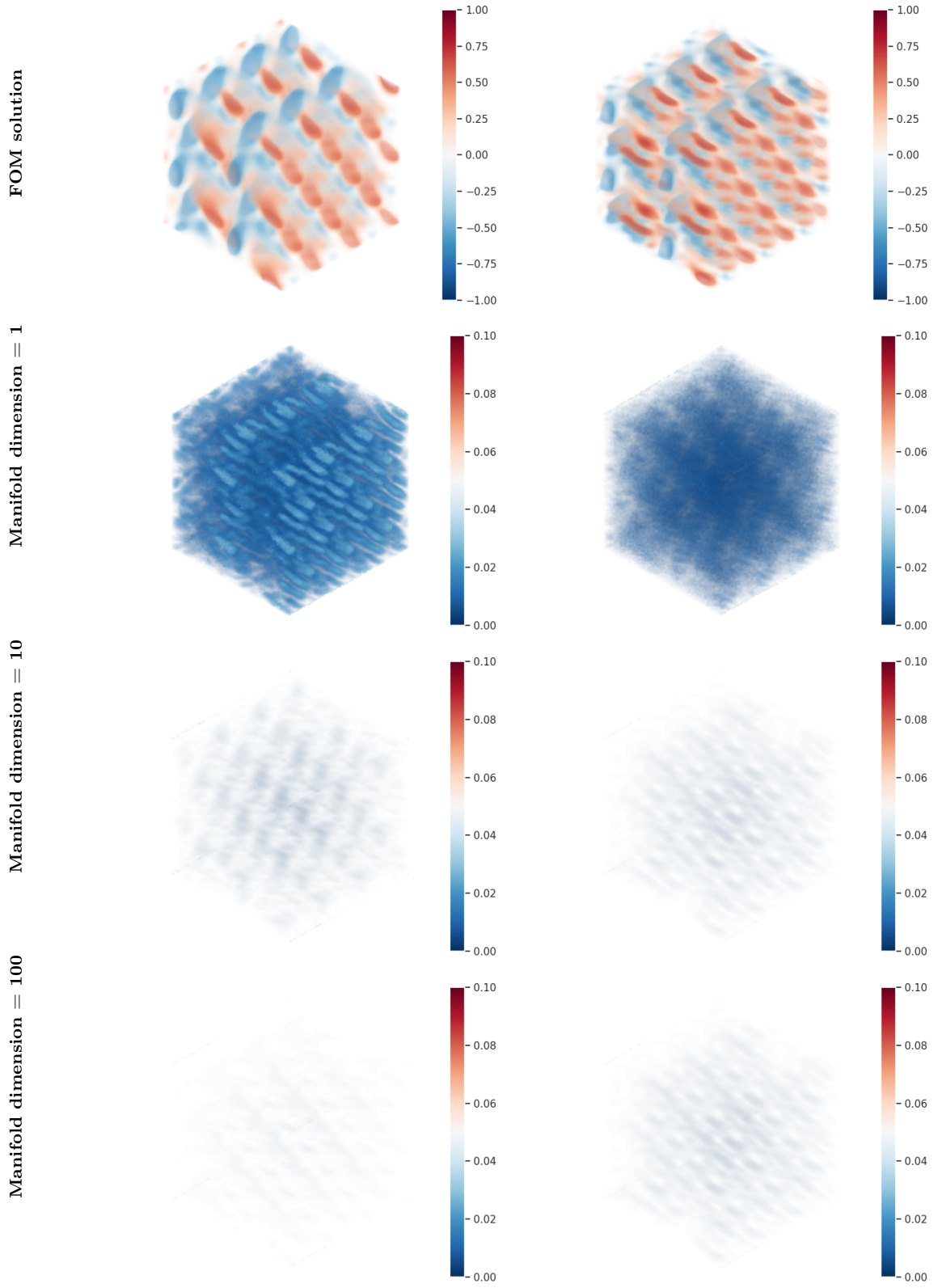


Figure 13. Qualitative comparison of the reconstructions of the inv-AE for different manifold dimensions. The first row shows two FOM solutions from the test set, while the remaining rows the absolute reconstruction error achieved by the different inv-AEs.

References

[1] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

[2] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531, 2015.

[3] Jan S Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.

[4] Peter Benner, Mario Ohlberger, Albert Cohen, and Karen Willcox. *Model reduction and approximation: theory and algorithms*. SIAM, 2017.

[5] Stefania Fresca, Luca Dede', and Andrea Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs. *Journal of Scientific Computing*, 87:1–36, 2021.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[7] M. Ohlberger and S. Rave. Reduced basis methods: Success, limitations and future challenges. *Proceedings of the Conference Algoritmy*, pages 1–12, 2016.

[8] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[9] Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3), 2021.

[10] Kai Fukami, Takaaki Murata, Kai Zhang, and Koji Fukagata. Sparse identification of nonlinear dynamics with low-dimensionalized flow representations. *Journal of Fluid Mechanics*, 926:A10, 2021.

[11] Joseph Bakarji, Kathleen Champion, J Nathan Kutz, and Steven L Brunton. Discovering governing equations from partial measurements with deep delay autoencoders. *Proceedings of the Royal Society A*, 479(2276):20230422, 2023.

[12] Nicola Rares Franco, Daniel Fraulin, Andrea Manzoni, and Paolo Zunino. On the latent dimension of deep autoencoders for reduced order modeling of PDEs parametrized by random fields. *Advances in Computational Mathematics*, 50(5):96, 2024.

[13] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[14] Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

[15] Patrick Buchfink, Silke Glas, and Bernard Haasdonk. Symplectic model reduction of Hamiltonian systems on nonlinear manifolds and approximation with weakly symplectic autoencoder. *SIAM Journal on Scientific Computing*, 45(2):A289–A311, 2023.

[16] The-Gia Leo Nguyen, Lynton Ardizzone, and Ullrich Köthe. Training invertible neural networks as autoencoders. In *German Conference on Pattern Recognition*, pages 442–455. Springer, 2019.

[17] Samuel E Otto, Gregory R Macchio, and Clarence W Rowley. Learning nonlinear projections for reduced-order modeling of dynamical systems using constrained autoencoders. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(11), 2023.

[18] Simone Brivio and Nicola Rares Franco. Deep symmetric autoencoders from the Eckart-Young-Schmidt perspective. *arXiv preprint arXiv:2506.11641*, 2025.

[19] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[20] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

[21] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

[22] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International conference on machine learning*, pages 573–582. PMLR, 2019.

[23] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-RevNet: Deep invertible networks. In *ICLR 2018-International Conference on Learning Representations*, 2018.

[24] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[25] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[26] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[27] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

[28] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

[30] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2019.

[31] Andrew Glaws, Ryan N King, Ganesh Vijayakumar, and Shreyas Ananthan. Invertible neural networks for airfoil design. *AIAA journal*, 60(5):3035–3047, 2022.

[32] Jens Behrmann, Sören Dittmer, Pascal Fernsel, and Peter Maaß. Analysis of invariance and robustness via invertibility of relu-networks. *arXiv preprint arXiv:1806.09730*, 2018.

[33] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[34] Joern-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *International Conference on Learning Representations*, 2019.

[35] Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Jörn-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.

[36] Yunfei Teng and Anna Choromanska. Invertible autoencoder for domain adaptation. *Computation*, 7(2):20, 2019.

[37] Siyuan Li, Haitao Lin, Zelin Zang, Lirong Wu, Jun Xia, and Stan Z Li. Invertible manifold learning for dimension reduction. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III 21*, pages 713–728. Springer, 2021.

[38] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[39] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[40] Stefania Fresca and Andrea Manzoni. POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388:114181, 2022.

[41] Patrick Buchfink, Silke Glas, Bernard Haasdonk, and Benjamin Unger. Model reduction on manifolds: a differential geometric framework. *Physica D: Nonlinear Phenomena*, 468:134299, 2024.

[42] Yvon Maday, Anthony T. Patera, and Gabriel Turinici. A priori convergence theory for reduced-basis approximations of single-parameter elliptic partial differential equations. *SIAM Journal on Scientific Computing*, 17(1):437–446, 2002.

[43] Haasdonk, Bernard. Convergence rates of the POD–greedy method. *ESAIM: M2AN*, 47(3):859–873, 2013.

[44] Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and Helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

[45] Albert Cohen, Charbel Farhat, Yvon Maday, and Agustin Somacal. Nonlinear compressive reduced basis approximation for PDE's. *Comptes Rendus Méchanique*, 351(S1):357–374, 2023.

[46] Patrick Buchfink, Silke Glas, and Bernard Haasdonk. Approximation bounds for model reduction on polynomially mapped manifolds. *Comptes Rendues. Mathématique*, 362:1881–1891, 2024.

[47] Jan S Hesthaven and Stefano Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[48] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[49] Matteo Tomasetto, Jan P Williams, Francesco Braghin, Andrea Manzoni, and J Nathan Kutz. Reduced order modeling with shallow recurrent decoder networks. *Nature Communications*, 16(1):10260, 2025.

[50] Martin Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E Rognes, and Garth N Wells. The FEniCS project version 1.5. *Archive of numerical software*, 3(100), 2015.

[51] Diederik Johannes Korteweg and Gustav De Vries. Xli. on the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 39(240):422–443, 1895.

[52] Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. Apebench: A benchmark for autoregressive neural emulators of pdes. *Advances in Neural Information Processing Systems*, 37:120252–120310, 2024.

[53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Appendix A. Additional Results

Appendix A.1. Burgers Equation

Manifold Dimension	POD	Convolutional AE	Fully-connected AE	Inv-AE	Inv-AE (light)
1	$3.82e^{-2}$	$9.13e^{-3}$	$\mathbf{8.71e^{-3}}$	$8.91e^{-3}$	$8.89e^{-3}$
2	$2.70e^{-2}$	$5.38e^{-3}$	$4.57e^{-3}$	$4.27e^{-3}$	$\mathbf{3.77e^{-3}}$
3	$2.09e^{-2}$	$4.19e^{-4}$	$1.94e^{-4}$	$\mathbf{1.42e^{-4}}$	$1.83e^{-4}$
4	$1.73e^{-2}$	$3.36e^{-4}$	$1.61e^{-4}$	$\mathbf{1.20e^{-4}}$	$1.49e^{-4}$
5	$1.48e^{-2}$	$3.20e^{-4}$	$1.49e^{-4}$	$\mathbf{1.00e^{-4}}$	$1.39e^{-4}$
10	$8.60e^{-3}$	$2.77e^{-4}$	$1.42e^{-4}$	$\mathbf{8.08e^{-5}}$	$1.28e^{-4}$
20	$4.06e^{-3}$	$2.80e^{-4}$	$1.39e^{-4}$	$\mathbf{7.45e^{-5}}$	$1.36e^{-4}$
50	$6.46e^{-4}$	$3.31e^{-4}$	$1.50e^{-4}$	$\mathbf{6.61e^{-5}}$	$8.92e^{-5}$
100	$\mathbf{3.46e^{-5}}$	$3.41e^{-4}$	$1.33e^{-4}$	$5.45e^{-5}$	$6.96e^{-5}$
150	$\mathbf{1.94e^{-6}}$	$2.91e^{-4}$	$1.62e^{-4}$	$4.94e^{-5}$	$5.87e^{-5}$
200	$\mathbf{1.18e^{-7}}$	$3.15e^{-4}$	$1.56e^{-4}$	$4.65e^{-5}$	$3.40e^{-5}$

Table A.1. Best reconstruction errors calculated on the test set achieved by the different approaches. We use **bold** to highlight the best values.

Manifold Dimension	POD-NN	DL-ROM (Conv. AE)	DL-ROM (Fully-conn. AE)	Inv-DL-ROM	Inv-DL-ROM (light)
1	-	$9.70e^{-3}$	$\mathbf{9.60e^{-3}}$	$9.67e^{-3}$	$9.66e^{-3}$
2	$3.82e^{-2}$	$6.78e^{-3}$	$6.61e^{-3}$	$3.28e^{-3}$	$\mathbf{2.82e^{-3}}$
3	-	$1.29e^{-3}$	$6.71e^{-4}$	$4.27e^{-4}$	$\mathbf{4.18e^{-4}}$
4	$2.71e^{-2}$	$1.44e^{-3}$	$6.49e^{-4}$	$4.12e^{-4}$	$\mathbf{3.91e^{-4}}$
5	-	$1.21e^{-3}$	$6.33e^{-4}$	$\mathbf{3.78e^{-4}}$	$3.93e^{-4}$
10	$1.48e^{-2}$	$1.40e^{-3}$	$5.52e^{-4}$	$\mathbf{3.76e^{-4}}$	$3.83e^{-4}$
20	$8.63e^{-3}$	$1.22e^{-3}$	$6.06e^{-4}$	$\mathbf{4.37e^{-4}}$	$4.44e^{-4}$

Table A.2. Best reduction errors calculated on the test set achieved by the different approaches. We use **bold** to highlight the best values.

Appendix A.2. Flow Around an Obstacle

Manifold Dimension	POD	Convolutional AE	Fully-connected AE	Inv-AE	Inv-AE (light)
1	-	$3.31e^{-2}$	$\mathbf{1.94e^{-2}}$	$2.16e^{-2}$	$2.21e^{-2}$
2	$3.31e^{-2}$	$1.53e^{-2}$	$1.25e^{-2}$	$1.23e^{-2}$	$\mathbf{1.19e^{-2}}$
3	-	$1.16e^{-2}$	$\mathbf{9.36e^{-3}}$	$9.46e^{-3}$	$9.66e^{-3}$
4	$1.30e^{-2}$	$9.17e^{-3}$	$8.52e^{-3}$	$\mathbf{7.74e^{-3}}$	$7.85e^{-3}$
5	-	$9.38e^{-3}$	$8.13e^{-3}$	$\mathbf{6.48e^{-3}}$	$6.50e^{-3}$
10	$9.12e^{-3}$	$1.33e^{-2}$	$6.37e^{-3}$	$3.70e^{-3}$	$\mathbf{3.13e^{-3}}$
20	$6.59e^{-3}$	$1.66e^{-2}$	$6.26e^{-3}$	$2.25e^{-3}$	$\mathbf{2.18e^{-3}}$
50	$3.63e^{-3}$	$1.46e^{-2}$	$6.05e^{-3}$	$1.63e^{-3}$	$\mathbf{1.58e^{-3}}$
100	$1.99e^{-3}$	$1.39e^{-2}$	$6.35e^{-3}$	$1.18e^{-3}$	$\mathbf{9.70e^{-4}}$
150	-	$9.63e^{-3}$	$6.23e^{-3}$	$9.16e^{-4}$	$\mathbf{7.93e^{-4}}$
200	$9.36e^{-4}$	$1.47e^{-2}$	$6.44e^{-3}$	$9.25e^{-4}$	$\mathbf{7.06e^{-4}}$
512	$3.09e^{-4}$	-	-	-	-

Table A.3. Best reconstruction errors calculated on the test set achieved by the different approaches. We use **bold** to highlight the best values.

Manifold Dimension	POD-NN	DL-ROM (Conv. AE)	DL-ROM (Fully-conn. AE)	Inv-DL-ROM	Inv-DL-ROM (light)
1	-	$\mathbf{2.40e^{-2}}$	$2.58e^{-2}$	$3.23e^{-2}$	$3.41e^{-2}$
2	$3.34e^{-2}$	$1.30e^{-2}$	$\mathbf{1.30e^{-2}}$	$1.37e^{-2}$	$1.36e^{-2}$
3	-	$1.02e^{-2}$	$9.70e^{-3}$	$\mathbf{9.61e^{-3}}$	$9.65e^{-3}$
4	$1.32e^{-2}$	$9.78e^{-3}$	$9.29e^{-3}$	$\mathbf{8.72e^{-3}}$	$9.12e^{-3}$
5	-	$9.53e^{-3}$	$9.29e^{-3}$	$\mathbf{8.20e^{-3}}$	$9.14e^{-3}$
10	$1.09e^{-2}$	$9.12e^{-3}$	$8.77e^{-3}$	$\mathbf{8.22e^{-3}}$	$8.56e^{-3}$
20	$1.00e^{-2}$	$8.44e^{-3}$	$8.92e^{-3}$	$7.69e^{-3}$	$\mathbf{7.63e^{-3}}$
50	$9.08e^{-3}$	$8.71e^{-3}$	$8.63e^{-3}$	$7.70e^{-3}$	$\mathbf{7.17e^{-3}}$

Table A.4. Best reduction errors calculated on the test set achieved by the different approaches. We use **bold** to highlight the best values.

Appendix A.3. Korteweg-de Vries

Manifold Dimension	POD	Convolutional AE	Fully-connected AE	Inv-AE	Inv-AE (light)
1	$5.92e^{-2}$	$4.75e^{-2}$	$7.53e^{-3}$	$\mathbf{1.53e^{-3}}$	$2.83e^{-3}$
5	$3.49e^{-2}$	$3.18e^{-3}$	$1.29e^{-3}$	$2.94e^{-4}$	$\mathbf{2.46e^{-4}}$
10	$1.88e^{-2}$	$2.88e^{-3}$	$1.55e^{-3}$	$1.75e^{-4}$	$\mathbf{1.41e^{-4}}$
50	$4.89e^{-4}$	$3.39e^{-3}$	$1.46e^{-3}$	$1.01e^{-4}$	$\mathbf{1.20e^{-5}}$
100	$2.84e^{-5}$	$6.18e^{-3}$	$1.53e^{-3}$	$1.10e^{-4}$	$\mathbf{7.45e^{-6}}$
200	$\mathbf{8.48e^{-7}}$	$2.97e^{-3}$	$1.42e^{-3}$	$6.94e^{-5}$	$1.26e^{-6}$
512	$1.92e^{-7}$	-	-	-	-

Table A.5. Best reconstruction errors calculated on the test set achieved by the different approaches. We use **bold** to highlight the best values.

Manifold Dimension	POD-NN	DL-ROM (Conv. AE)	DL-ROM (Fully-conn. AE)	Inv-DL-ROM	Inv-DL-ROM (light)
1	$5.92e^{-2}$	$3.15e^{-2}$	$9.70e^{-3}$	$4.49e^{-3}$	$\mathbf{4.73e^{-3}}$
5	$3.42e^{-2}$	$1.65e^{-2}$	$3.63e^{-3}$	$\mathbf{1.01e^{-3}}$	$1.09e^{-3}$
10	$2.59e^{-2}$	$1.78e^{-2}$	$5.58e^{-3}$	$\mathbf{9.97e^{-4}}$	$1.16e^{-3}$
50	$2.29e^{-2}$	$1.84e^{-2}$	$6.49e^{-3}$	$\mathbf{1.02e^{-3}}$	$1.11e^{-3}$

Table A.6. Best reduction errors calculated on the test set achieved by the different approaches. We use **bold** to highlight the best values.

Appendix B. Neural Network Architectures

Appendix B.1. Invertible AE

The inv-AE is composed of 5 invertible layers L. Each reversible layer is based on a multiplicative affine coupling transformation, where the input vector is split into two parts $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$. The forward transformation is defined as

$$\begin{aligned} \boldsymbol{y}_1 &= \boldsymbol{x}_1 \odot \exp(\boldsymbol{s}_2(\boldsymbol{x}_2); \boldsymbol{\theta}_{\boldsymbol{s}_2}) + \boldsymbol{t}_2(\boldsymbol{x}_2; \boldsymbol{\theta}_{\boldsymbol{t}_2}) \\ \boldsymbol{y}_2 &= \boldsymbol{x}_2 \odot \exp(\boldsymbol{s}_1(\boldsymbol{y}_1); \boldsymbol{\theta}_{\boldsymbol{s}_1}) + \boldsymbol{t}_1(\boldsymbol{y}_1; \boldsymbol{\theta}_{\boldsymbol{t}_1}) \end{aligned},$$

while the inverse transformation can be analytically computed as

$$\begin{aligned} \boldsymbol{x}_1 &= (\boldsymbol{y}_1 - \boldsymbol{t}_2(\boldsymbol{x}_2; \boldsymbol{\theta}_{\boldsymbol{t}_2})) \odot \exp(-\boldsymbol{s}_2(\boldsymbol{x}_2; \boldsymbol{\theta}_{\boldsymbol{s}_2}))\,, \\ \boldsymbol{x}_2 &= (\boldsymbol{y}_2 - \boldsymbol{t}_1(\boldsymbol{y}_1; \boldsymbol{\theta}_{\boldsymbol{t}_1})) \odot \exp(-\boldsymbol{s}_1(\boldsymbol{y}_1; \boldsymbol{\theta}_{\boldsymbol{s}_1}))\,. \end{aligned}$$

In the case of the inv-AE, the scale and translation operators $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t}_1, \boldsymbol{t}_2$ are represented by fully connected neural networks $N/2 \to 512 \to N/2$ with GeLU activation. In the case of the inv-AE (light), instead, we use 3 fully connected layers $N/2 \to 128 \to 128 \to N/2$ with GeLU actuvation to compensate for the reduction of the paramenters. The resulting architecture combines nonlinear dimensionality reduction with an explicitly invertible latent representation, enabling the learning of compact reduced-order coordinates while preserving a stable mapping between the latent and full-order spaces.

Appendix B.2. 1-dimensional Convolutional AE

For one-dimensional spatial fields, namely the 1-dimensional Burgers' equation, we employ a convolutional autoencoder composed of a four-layer convolutional encoder and a transposed-convolutional decoder. The same architecture was used in [13]. The encoder is composed of 4 1-dimensional convolutional layers with ELU activations:

$$\begin{aligned} &\text{Conv1:} \quad C \to 8 \qquad k = 25, s = 2, p = 1, \\ &\text{Conv2:} \quad 8 \to 16, \qquad k = 25, s = 2, p = 1, \\ &\text{Conv3:} \quad 16 \to 32, \qquad k = 25, s = 1, p = 1, \\ &\text{Conv4:} \quad 32 \to 64, \qquad k = 25, s = 2, p = 1, \end{aligned} \tag{B.1}$$

where C denotes the number of channels, k the kernel size, s the stride length, and p the padding dimension. The feature maps produced by the final convolutional layer are flattened and projected onto a low-dimensional latent space through a fully connected layer. The decoder mirrors the encoder architecture. The latent vector is first mapped through a fully connected layer and reshaped into a tensor with 64 channels, which is then processed by four transposed convolutional layers:

$$\begin{aligned} &\text{Deconv1:} \quad 64 \to 32, \qquad k = 25, s = 1, p = 0, \\ &\text{Deconv2:} \quad 32 \to 16, \qquad k = 25, s = 1, p = 1, \\ &\text{Deconv3:} \quad 16 \to 8, \qquad k = 25, s = 2, p = 0, \\ &\text{Deconv4:} \quad 8 \to C, \qquad k = 25, s = 2, p = 0, \end{aligned} \tag{B.2}$$

with ELU activations applied after each layer except the final output layer. The relatively large kernel size ($k = 25$) allows the network to capture long-range spatial correlations, while the strided convolutions progressively reduce the spatial resolution and extract increasingly abstract features before projection onto the latent manifold.

Appendix B.3. 2-dimensional Convolutional AE

For the 2-dimensional and 3-dimensional test cases, we employ the convolutional autoencoder introduced in [40]. The encoder consists of 4 2-dimensional convolutional layers, each employing (3×3) kernels and

ELU activation functions. The architecture is given by

$$\begin{aligned}
&\text{Conv1:} \quad 2 \to 32, \qquad k = 3, s = 1, \\
&\text{Conv2:} \quad 32 \to 32, \qquad k = 3, ; s = 1, \\
&\text{BatchNorm}, \\
&\text{Conv3:} \quad 32 \to 32, \qquad k = 3, s = 1, \\
&\text{Conv4:} \quad 32 \to 32, \qquad k = 3, s = 1, \\
&\text{BatchNorm}.
\end{aligned} \tag{B.3}$$

No pooling or strided convolutions are employed. Instead, the dimensionality reduction is performed through the fully connected layers following the convolutional feature extractor. After the final convolutional layer, the feature maps are flattened and projected onto the chosen manifold of dimension n through 2 fully connected layers. The latent vector is first projected back to the convolutional feature space using a fully connected layer and then reconstructed through 4 transposed convolutional layers

$$\begin{aligned}
&\text{Deconv1:} \quad 32 \to 32, \qquad k = 3, s = 1, \\
&\text{Deconv2:} \quad 32 \to 32, \qquad k = 3, s = 1, \\
&\text{BatchNorm}, \\
&\text{Deconv3:} \quad 32 \to 32, \qquad k = 3, s = 1, \\
&\text{Deconv4:} \quad 32 \to 2, \qquad k = 3, s = 1,
\end{aligned} \tag{B.4}$$

with ELU activations applied after all layers except the final output layer. The use of small (3×3) kernels enables the network to efficiently capture local spatial correlations the underlying field. Moreover, by avoiding aggressive downsampling operations, the architecture retains fine-scale spatial information that is essential for accurately reconstructing high-dimensional dynamical states.

Appendix B.4. Fully Connected AE

The encoder is composed of five fully connected layers, mapping the input state from dimension N to a lower-dimensional latent space of dimension n through the sequence $N \to 2048 \to 1024 \to 512 \to 256 \to n$. The decoder expands the latent representation back to the original state dimension through the sequence $n \to 256 \to 512 \to 1024 \to 2048 \to N$. Leaky-ReLU activation functions are applied after each hidden layer.

Appendix B.5. Fully Connected Reduced-order Model

In the proposed implementation, the ξ is composed of 3 fully connected layers with architecture $(\boldsymbol{\mu}, t) \to 256 \to 256 \to n$, where p denoted the dimension of the parameter space. The first two layers are followed by Leaky-ReLU nonlinear activation functions, while the final layer directly outputs the latent variables. Unlike the encoder, which extracts latent coordinates from observed states, this network learns a direct approximation of the latent manifold as a function of the governing parameters and time.

Appendix C. Spectral Normalization

The training stability of NNs is heavily affected by the Lipschitz constant of networks themselves. Therefore, controlling the Lipschitz constant and preventing its rapid growth is crucial from avoiding exploding gradients and for the loss convergence. Spectral normalization [38] is a type of weight normalization that can improve performance of the NN architectures suffering from training instabilities, such as generative adversarial networks [53].

Given a NN layer with weight matrix W, the spectral norm of W equals the largest singular value $\sigma_{\max}(W)$, i.e., $\sigma_{\max}(W) = \|W\|_2$. Spectral normalization then uses the spectral norm of W for normalization such that $\sigma(W) = 1$, i.e.,

$$W_{\text{SN}} = \frac{W}{\sigma_{\max}(W)},$$

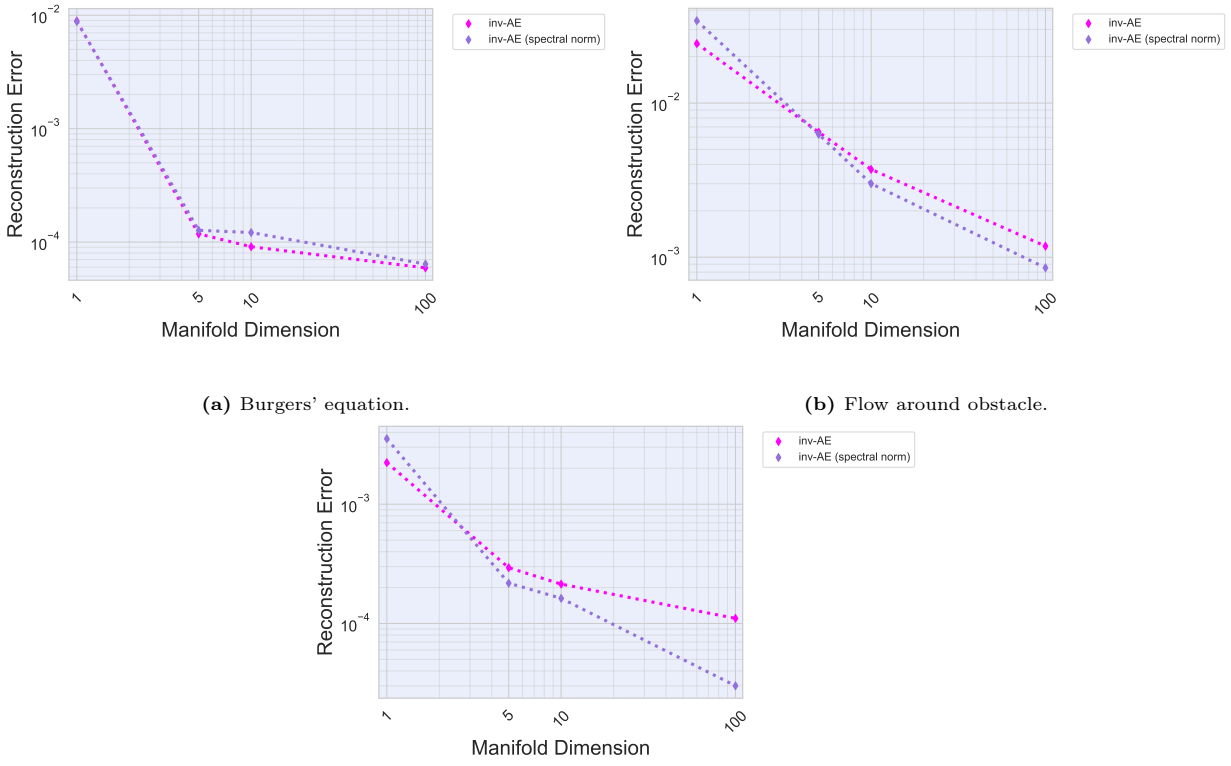
where W_{SN} is the spectrally normalized weight matrix.

If the activation functions have Lipschitz constant equal to 1,[2] the Lipschitz constant of a NN can be bounded from about by the product of the spectral norms of the the weight matrices of each layer l

$$L \leq \prod_{l=1}^{N_L} \sigma_{\max}(W^l)\,, \tag{C.1}$$

where N_L indicates the number of layers of the network and W^l the weight matrix associated with l^{th} layer. Therefore, if all layers of the network are spectrally normalized, then the Lipschitz constant of the network is bounded from above by 1.

In Figure C.14 we show the effect of spectral normalization on the reconstruction errors of the AE for the three different test cases.



(a) Burgers' equation.

(b) Flow around obstacle.

(c) Korteweg-de Vries.

Figure C.14. Projection error with and without spectral normalization for the three different test cases.

[2]This is the case for many activation functions such as ReLU and leaky ReLU.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

78/2026 Botteghi, N; Brook, O; Fasel, U; Califano, F
Interconnection and Damping Assignment Passivity-Based Control using Sparse Neural ODEs

79/2026 Vitullo, P.; Macaluso, C.; Gazzoni, S.; Cassanmagnago, A.; Dimola, N.; Possenti, L.; Zunino, P.
A computational model for phenotype-structured tumour dynamics with reduced microvascular oxygen and event-based radiotherapy

77/2026 Renzi, F.; Pagani, S.; Antonietti, P.F.; Vergara, C.
A Neural-Network Approach for the Stability of Loosely-Coupled FSI Simulations in Hemodynamics

75/2026 Botteghi, N; Tomasetto, M; Fasel, U; Braghin, F; Manzoni, A
HypeMARL: Multi-Agent Reinforcement Learning For High-Dimensional, Parametric, and Distributed Systems

76/2026 Pizzamiglio, S.M.; Pagani, S.; Regazzoni, F.
Learning PDE solution operators with variable initial conditions via Latent Dynamics Networks

74/2026 Brivio, S.; Franco, N. R.
Deep Symmetric Autoencoders from the Eckart-Young-Schmidt Perspective

73/2026 Corti, M.
A weighted polygonal discontinuous Galerkin method for hierarchically coupled reaction–diffusion systems with cubic interactions

72/2026 Lorenzon, G.; Regazzoni, F.
Optimal Transport Dropout for Structured Predictive Uncertainty

70/2026 Micheletti, S.
Longitudinal versus quasi-P waves in weakly anisotropic elasticity: operator comparison and long-time dephasing

71/2026 Radisic, I.; Regazzoni, F.; Dede', L.; Quarteroni, A.
A mathematical model for irreversible damage of the collagen scaffold in the myocardium