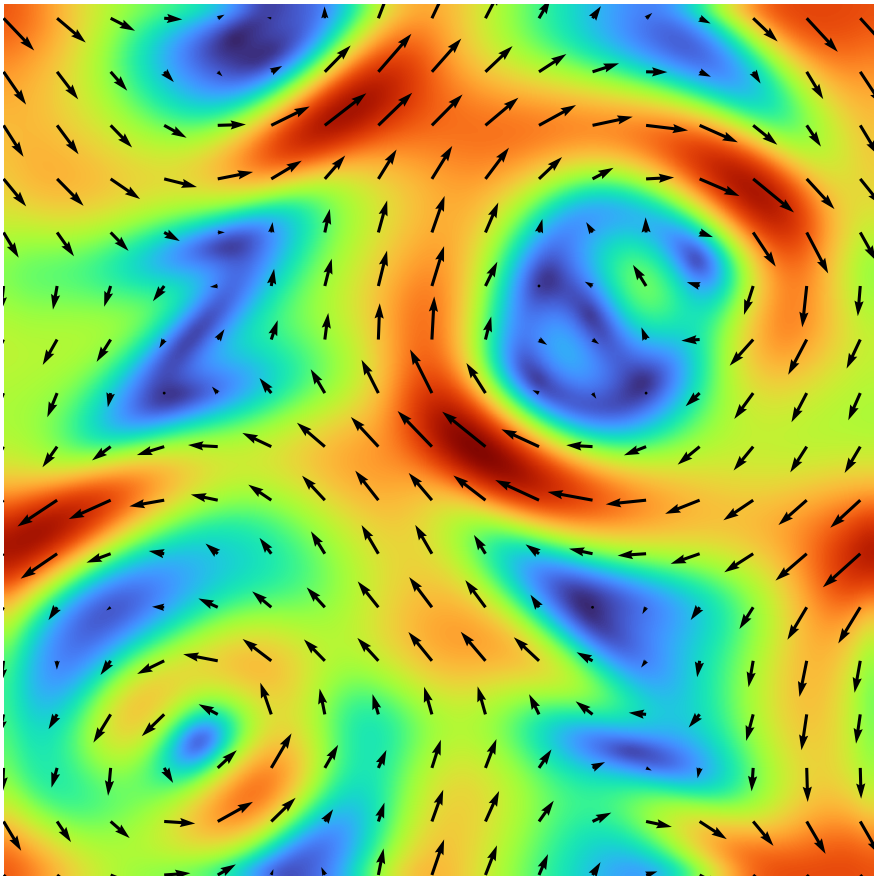
Master of Science Thesis

Hybrid knowledge-based/deep learning reduced order modelling of high dimensional chaotic systems

J. A. Veerman

October 27, 2023



Aerodynamics

TU Delft

Faculty of Aerospace Engineering

Delft University of Technology

Hybrid knowledge-based/deep learning reduced order modelling of high dimensional chaotic systems

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

J. A. Veerman

October 27, 2023

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF AERODYNAMICS

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance the thesis entitled **"Hybrid knowledge-based/deep learning reduced order modelling of high dimensional chaotic systems"** by **J. A. Veerman** in fulfillment of the requirements for the degree of **Master of Science**.

Defence date : November 16, 2023
Project duration : January 2023 - November 2023
Student number : 5410800

Supervisor: Dr. N.A.K. Doan

Chair: Dr. ir. M.I. Gerritsma

External examiner: Dr. I. Langella

Preface

This Master thesis marks the end of an incredible and rather unique educational journey, which started back in 2009 in high school at the VMBO-TL. After graduating from the VMBO-TL and later as well from the HAVO and the VWO, I decided to study Earth Science at the University of Utrecht, which I later combined into a dual bachelor with Physics. During my time at Physics, it were the courses in fluid mechanics and turbulence that sparked my interest into (numerical) small scale fluid dynamics, from which I decide to continue with a master in Aerospace Engineering, focusing on (numerical) aerodynamics.

During my masters education, I have discovered my passion for machine learning and after learning more about neural networks I was sure that I wanted to conduct my thesis within this topic. I am glad that I was able to learn and gain first hand experience on with a variety of machine learning techniques ranging from dense neural networks to more advanced tools such as convolutional neural networks and recurrent neural networks. My excitement on machine learning only grew stronger over the course of this thesis.

There are a few people without whom I would not have been able to deliver this thesis. First, I would like to thank my supervisor Dr. (Anh Khoa) Doan for his supervision and his help along the project. The independence he provided me to work on this thesis combined with his insightful tips and our discussions were very helpful. I also would like to thank Mathias Lesjak for providing his numerical codes on knowledge-based (for Kolmogorov flow), deep learning and hybrid reduced order models for low dimensional chaotic systems as well as more information which lay the foundation of my thesis. Furthermore, I would like to thank George Popi and Menno Veerman for proof reading my thesis and providing me with useful feedback. Finally, I would like to thank my family for their unconditional support.

I hope you enjoy reading my work!

Jochem Veerman, October 27, 2023

Table of Contents

List of Figures

List of Tables

Abstract

Chaotic systems are widespread and can be found everywhere, from small scale processes inside the human body to the large scale dynamics of the entire atmosphere. However, modelling these high dimensional chaotic systems is a difficult task due to the intrinsic nonlinear nature of chaos as well as the accompanied computational cost. Therefore, in this research, the predictive potential of hybrid reduced order models consisting of a knowledge-based (proper orthogonal decomposition Galerkin projection) and a deep learning reservoir computing component was studied for high dimensional chaotic systems, for which two hybrid architectures were designed. In the first, the blending of the knowledge-based and deep learning components occurred within the reservoir of the data-driven component (Hybrid-A) and its performance was assessed through the short-term prediction horizon and long-term statistical predictions and compared to the pure knowledge-based and pure deep learning reduced order models. For the second model (Hybrid-FFNN), the blending occurred in a separate feed-forward neural network and the performance was compared to the Hybrid-A model. In order to obtain efficient deep learning predictions for both models, a low dimensional latent space representation of the physical system was obtained using autoencoders. Both models were tested on two-dimensional Kolmogorov flow at a Reynolds number of $Re = 20$ (periodic regime) and $Re = 34$ (chaotic regime).

For the periodic case, the short-term performance of the Hybrid-A and deep learning model were comparable, while the knowledge-based model was outperformed due to instabilities as a result of the truncation of the number of modes from proper orthogonal decomposition. Furthermore, an increase in latent space and/or deep learning reservoir size had no consistent influence on the performance of the Hybrid-A model, due to the periodic and thus simple to learn dynamics for the deep learning component. Finally, little influence was found for the accumulation of (de)compression errors occurring in the Hybrid-A model, which was compensated for by the additional information from the knowledge-based model and/or the Hybrid-A model was able to restore the errors due to the periodic nature of the system.

Increasing the Reynolds number to the chaotic regime resulted in the Hybrid-A model to underperform compared to the knowledge-based and deep learning models for smaller latent space due to the (de)compression error accumulation. In addition, the Hybrid-A model was

unable to perform better than the knowledge-based model for larger latent spaces and number of retained modes, from which it was concluded that too much information on the physical system was lost in latent space for the Hybrid-A model to benefit from both components, while the knowledge-based model operated on the (full) physical system. For the Hybrid-FFNN it was shown that the feed-forward neural network introduced a too large error to be beneficial over the Hybrid-A model.

For the long-term statistics of both test cases, the importance of the design for the tuning of the hyperparameters of the deep learning model/component was showed. In general, no clear influence was found in terms of the performance of the Hybrid-A model as a function of the latent space and reservoir size. Furthermore, the comparative performance of the knowledge-based, deep learning and Hybrid-A models showed no relation. This behaviour was expected to be the result of a design strategy that biased the short-term performance over the long-term performance of the deep learning model/component. Furthermore, for the periodic regime, such behaviour could also have originated from the periodicity of the flow.

Up to the authors knowledge, this work proposed the first hybrid knowledge-based/deep learning reduced order model using autoencoders for efficient predictions of high dimensional chaotic systems. Even though it was found that the proposed Hybrid models were unable to show increased performance compared to the knowledge-based and deep learning models, the results can be viewed as a start to design other hybrid architectures for potential performance improvement.

Nomenclature

Abbreviations

AN	Artificial neuron
CNN	Convolutional neural network
DNN	Dense neural network
DoF	Degrees of freedom
EI	Expected Improvement
ESN	Echo state network
FFNN	Feed-forward neural network
GPR	Gaussian process regression
GP	Gaussian process
IC	Initial conditions
KNMI	Royal Dutch Meteorological Institute
LES	Large Eddy Simulations
LSTM	Long short-term memory
LT	Long term
MSE	Mean square error
POD-GL	proper orthogonal decomposition Galerkin
POD	Proper orthogonal decomposition
RANS	Reynolds Averaged Navier Stokes
RMSE	Root mean square error
RNN	Recurrent neural network
ROM	Reduced order model(ling)
ST	Short term
TCNN	Transpose convolutional neural network

Deep learning

α	Leaking rate
β	Tikhonov factor
$\boldsymbol{\beta}$	Array containing Tikhonov factors for grid-search
$\tilde{\boldsymbol{H}}(t)$	ESN training predictions
$\tilde{\boldsymbol{h}}(t)$	ESN prediction, single snapshot
$\tilde{\boldsymbol{Q}}$	Output data TCNN
$\boldsymbol{H}(t)$	Training data ESN
$\boldsymbol{h}(t)$	True data ESN, single snapshot
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{Q}$	Input data CNN
$\boldsymbol{Q}_{\text{dec}}$	Decoded image
$\boldsymbol{Q}_{\text{lat}}$	Latent space image
$\boldsymbol{Q}_F$	Feature map
$\boldsymbol{q}_F$	Filter kernel
$\boldsymbol{W}$	Internal matrix ESN
$\boldsymbol{W}^{\text{in}}$	Input matrix ESN
$\boldsymbol{W}^{\text{out}}$	Output matrix ESN
$\boldsymbol{x}_{\text{res}}(t)$	Internal state ESN
$\boldsymbol{X}_{\text{res}}$	Reservoir state matrix during training
η_{dec}	Number of filters per decoder layer
η_{enc}	Number of filters per encoder layer
$\langle d \rangle$	Sparsity
$\rho(\boldsymbol{W})$	Spectral radius
σ_{in}	Input scaling
$\tilde{y}_{\text{an}}$	Linear output artificial neuron
b	Bias of artificial neurons
D_f	Number of filters in CNN
f_{ac}	Nonlinear activation function
L_{train}	Training loss
L_{val}	Validation loss
$N_{\boldsymbol{Q}}$	First/Second dimension input data CNN

N_{h}	DoF in ESN single training snapshot
N_{lat}	Latent space dimension
N_{layers}	Number of layers in encoder/decoder
N_c	Third dimension input data CNN
N_f	First/Second dimension of filter in CNN
$N_{\boldsymbol{Q}_F}$	First/second dimensions feature map
N_{an}	Number of input data artificial neurons
$N_{\text{lat},x}$	number of x grid points latent space
$N_{\text{lat},y}$	number of y grid points latent space
N_{res}	Number of neurons in ESN
s	Stride
T_{h}	Number of ESN training snapshots
$w_{\text{an},j}$	Weights of artificial neurons
$x_{\text{an},j}$	Input data of artificial neurons
y_{an}	Nonlinear output of artificial neurons

Hybrid

$\tilde{\boldsymbol{y}}_{\text{hy, in}}$	Concatenated encoded Hybrid-A input
$\tilde{\boldsymbol{y}}_{\text{hy, out}}(t)$	Output deep learning component Hybrid-A
$\tilde{\boldsymbol{y}}_{\text{hy, lat}}(t-\delta t)$	Encoded Hybrid-A input data
$\tilde{\boldsymbol{y}}_{\text{hy,PODGL-lat}}(t)$	Encoded prediction POD-GL in Hybrid-A model
$\tilde{\boldsymbol{y}}_{\text{hy,PODGL}}(t)$	Prediction POD-GL in Hybrid-A model
$\tilde{\boldsymbol{y}}_{\text{hy}}(t)$	Prediction Hybrid-A model

Knowledge-based

$\boldsymbol{\Phi}(\boldsymbol{x})$	POD modes
$\boldsymbol{a}(t)$	Time dependent POD coefficient
$\boldsymbol{C}$	Correlation matrix
$\boldsymbol{f}_{for}(\boldsymbol{x},t)$	Forcing term
$\boldsymbol{y}'(\boldsymbol{x},t)$	Velocity fluctuations
$\boldsymbol{Y}'$	Velocity fluctuations matrix
$\boldsymbol{y}(\boldsymbol{x},t)$	State vector
Λ_{pod}	Matrix of POD eigenvalues
λ_{pod}	Eigenvalue of a POD mode
$\mathfrak{L}$	Generalized linear operator
$\mathfrak{N}$	Generalized nonlinear operator
E_{ret}	Energy of retained POD modes
N_{ret}	Retained number of POD modes
N_{pod}	Number of POD modes
q	Turbulent kinetic energy
T_{pod}	Number of fluctuation snapshots
Tr	Trace of matrix

Other symbols

$(\cdot,\cdot)$	Inner product
$\hat{\boldsymbol{y}}(\boldsymbol{x},t)$	Perturbed state
$\boldsymbol{f}_{\text{KF}}$	Kolmogorov forcing
$\boldsymbol{g}_{\mathbf{bb}}(\boldsymbol{\zeta}_{\mathbf{in}})$	Initial data Bayesian optimisation
$\boldsymbol{u}(\boldsymbol{x},t)$	Flow field $[u,\, v]$
$\boldsymbol{x}$	Spatial coordinates $[x,\, y]$
χ	Wave number magnitude
$\delta\boldsymbol{y}(\boldsymbol{x},t)$	Perturbations
$\delta\boldsymbol{y}_{\text{norm}}$	Normalized perturbations
δt	Time interval
$\delta\boldsymbol{y}'(\boldsymbol{x},t)$	Orthogonalized perturbations
ϵ_{Lyap}	Perturbation factor
$\epsilon_{hor}(t)$	Short-term prediction horizon error
$\epsilon_{thres}(t)$	Short-term prediction horizon threshold error
Γ_{ESN}	Total contribution ESN in Hybrid-A
Γ_{PODGL}	Total contribution POD-GL in Hybrid-A
Γ_j	Mean normalized contribution in Hybrid-A
$\hat{\boldsymbol{u}}$	Flow field in Fourier space
κ_{ij}	Entries of the output matrix
λ_{Lyap}	Lyapunov exponent
$\langle\cdot\rangle$	Ensemble average
$\langle u\rangle$	Mean flow in u
$\langle u'\rangle$	Mean fluctuations in u
$\langle v\rangle$	Mean flow in v
$\langle v'\rangle$	Mean fluctuations in v
$\mathbb{E}$	Expectation value
$\mathcal{N}()$	Multi-variate Gaussian distribution
$\mu_0(\zeta)$	Mean function

$\mu_n(\zeta_+)$	Mean function of posterior
$\Sigma_0(\zeta, \zeta)$	Covariance function/kernel
$\Sigma_n(\zeta_+)$	Covariance function/kernel of posterior
$\tilde{g}_{\text{bb}}(\zeta_+)$	Multi-variate Gaussian with mean and covariance from posterior
$\lvert \cdot \rvert$	Absolute value
$\lvert\lvert \cdot \rvert\rvert$	Euclidean norm
ζ	Hyperparameter space
ζ_+	Next point of hyperparameter space in Bayesian optimisation
$D(t)$	Domain averaged dissipation
D_{KY}	Kaplan-Yorke dimension
$E(t)$	Domain averaged kinetic energy
$g_{\text{bb}}(\zeta)$	Black-box objective function
$g_{\text{bb}}(\zeta_+)\lvert[\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}, \zeta_+]$	Posterior distribution Bayesian optimisation
N_{bo}	Number of evaluation points Bayesian optimisation
N_{ic}	Number of initial conditions in Bayesian optimisation
n_{KF}	Frequency Kolmogorov forcing
N_{podtrain}	Number of training data snapshots POD
N_{resync}	Number of resync snapshots
N_{rmd}	Number of random evaluation points before Bayesian optimisation for initial data
N_{train}	Number of training data snapshots
N_{val}	Number of validation data snapshots
N_{wash}	Number of washout snapshots
N_k	Number of symmetric wave numbers in KolSol
N_{phys}	DoF of physical system
$P(M\lvert O)$	Posterior
$P(M)$	Prior
$P(O\lvert M)$	Likelihood
p	Pressure
Re	Reynolds number
t	Temporal coordinate
T_{Lyap}	Lyapunov time
T_{per}	Time of a single period
t_{pred}	Short-term prediction horizon
t_{train}	Time steps of training data
t_{val}	Time steps of validation data
T_{orth}	Time interval orthonormalisation
$u(\boldsymbol{x}, t)$	x-component flow field
$v(\boldsymbol{x}, t)$	y-component flow field
X_{rand}	Random normalized matrix

Chapter 1

Introduction

Dynamical systems displaying chaotic behaviour form an important part of the daily life [1]. Chaos occurs in a wide variety of processes and scales, from small processes inside the human body [2] to the aerodynamics of an aircraft and from the flow of pedestrians [3] to the dynamics of the entire atmosphere [4]. All chaotic systems are characterized by the fact that a small perturbation in the initial conditions can results in vastly different outcomes over time [5]. For example, due to the chaotic behaviour of the atmosphere, the Dutch Royal Meteorological Institute (KNMI) performs for the daily weather forecast one high resolution weather simulations as well as 50 low resolutions runs with slightly perturbed initial conditions and model setting to account for the uncertainty arising from chaos [6].

Due to the impact of chaotic systems, it is of profound importance to be able to simulate their evolution and thus gain insight into the governing processes. However, due to the intrinsic nonlinearity and often the high dimensionality associated with chaotic systems, performing temporal predictions is often not straightforward and results into a large computational cost [7]. In addition, (complex) forcing terms in the governing equations might be unknown, preventing high accuracy simulations [7].

Fluid flows are known for their often chaotic behaviour and high dimensionality as a result of turbulence, which is often limiting for the exact (direct) numerical simulations [8]. Therefore, in order to reduce the computational cost [8], (part of) the scales of turbulence can be modelled rather than solved exactly through methods such as Reynolds Averaged Navier Stokes (RANS) [9] and Large Eddy Simulations (LES) [10]. However. as a consequence, the quality of the numerical solutions are lower.

Apart from modelling rather than solving exactly (part of) the turbulence, a significant reduction in computational cost can be obtained through reduced order models (ROMs), which aim to preserve only the dominant features in the flow [11, 12]. Such models can be constructed using initial data from experiments or from high-fidelity simulations [11, 12]. Reduced order modelling is used in for example online parameter estimation and active flow

control [13, 14, 15], and can be constructed using three approaches: A knowledge-based, a data-driven or a hybrid type of architecture.

Traditionally, reduced order models have been constructed using a knowledge-based approach in which the governing equations of the dynamical system were utilized to reflect the dominant features of the flow [11, 12]. The main idea is to decompose initially known data from the system into a set of modes and project the governing equations onto these modes through Galerkin projection [16]. Then, a significant reduction in computational cost can be obtained through truncation of the number of modes considered in the reduced order model [11, 16].

A common decomposition method is proper orthogonal decomposition (POD) as introduced by Lumley [17] into the field of fluid mechanics in 1967. A major advantage of this technique is the optimal property, which states that a linear basis obtained through proper orthogonal decomposition is optimal in the sense that it requires the least amount of modes compared to any other linear basis to represent a given dynamical system [16, 17]. However, despite the optimal property several drawbacks exists [12]. For example, no higher order correlations are captured with this method and the modes are ranked based on the amount of energy each reflect in the system rather than the dynamical importance [12]. For these reasons, other decomposition methods have been proposed, such as dynamic mode decomposition [18] and spectral proper orthogonal decomposition [19]. However, in the rest of this work, the focus will be on proper orthogonal decomposition in the context of knowledge-based reduced order modelling as it is a proven method that has often been used combination with Galerkin projection in literature [20, 21, 22, 23].

Using proper orthogonal decomposition, combined with Galerkin projection, knowledge-based reduced order models based on the governing equations can be constructed. Initial work on this topic within the field of fluid mechanics applied the knowledge-based reduced order modelling approach on relatively simple geometries, such as two-dimensional incompressible (grooved) channel flow [20], cylinder flow [20] and lid-driven cavity flow [21]. However, over the years, the extension was made to three dimensional fluid mechanical systems such as incompressible channel flow [22, 23] and incompressible cylinder flow [24, 25]. In addition, knowledge-based reduced order models were constructed for compressible flow [26, 27] and visoelastic turbulence [28, 29, 30]. These studies showed that the reduced order model was able to predict the dynamics of the systems for some time until divergence occurred [20, 24, 28, 29] as well as being robust for small changes in model parameters [20, 23]. However, it was also shown that the POD-Galerkin models suffered from instabilities due to a high dependency on the number of modes retained [24, 29, 31]. In addition, the errors grew significantly with prediction time [24, 28, 29] and additional closure models for sufficient energy drain from lower order modes were sometimes required [21, 22, 32].

As previously mentioned, the full governing equations might not be available, which is disadvantageous for knowledge-based reduced order modelling. Therefore, to avoid utilizing the governing equations, reduced order models can be constructed in a pure data-driven manner such as recurrent neural networks (RNN). Using such techniques is motivated by the property of recurrent neural networks to serve as a universal approximator for dynamical systems [33] and the recent advances in recurrent deep learning such as long short-term memory (LSTM) [34] and echo state networks (ESN) [35]. Up to the authors knowledge, one of the earliest stud-

ies using pure simple recurrent neural networks for temporal predictions of a fluid mechanical system was performed by Faller and Schreck [36] for the 2D flow around a NACA-0015 airfoil. However, simple RNN are known for possible vanishing and exploding gradients in the training phase [34, 37]. The vanishing gradient problem was tackled through the introduction of the more advanced LSTM [34, 37]. Furthermore, the echo state network addresses both the vanishing and exploding gradients by introducing a methodology not dependent on gradient descent [35]. The plain recurrent neural networks for temporal prediction on chaotic systems [36, 38, 39], however, are often limited by the high dimensionality of the chaotic systems [40]. For this reason, similar to knowledge-based reduced order modelling, data reduction is desired.

Similar to the Galerkin approach, data reduction can be obtained using decomposition techniques such as proper orthogonal decomposition [16, 17]. Wang et al. [41] applied a long-short term memory network to perform temporal predictions on the temporal POD coefficient for the three-dimensional incompressible non-hydrostatic Navier Stokes equation, showing that reasonable accuracy was obtainable at a much lower computational cost. Other studies using a POD-LSTM approach were performed by Rahman et al. [42], Mohan et al. [43] and Deng et al. [44]. Alternatively to proper orthogonal decomposition, other decomposition techniques such as Fourier [45] and spectral proper orthogonal decomposition [46] have been combined with LSTM cells to predict system dynamics. In addition, echo state networks can be utilized instead of LSTM [47, 48, 49]. The aforementioned studies obtained dimensional reduction through linear decomposition techniques. Therefore, by projecting chaotic systems onto a linear basis, errors are introduced due to missing nonlinear interactions [50].

A different methodology for data reduction is through a nonlinear deep learning approach with convolutional autoencoders, where a lower dimensional latent space representation is obtained through an encoder and the latent space is transformed back into full dimensions by a decoder [50, 51, 52, 53, 54]. In order to improve the performance of the dimensional reduction, more advanced autoencoders have been proposed and combined with deep learning prediction tools, such as the multi-scale autoencoder [55, 56, 57], self-attention autoencoder [58], advection-aware autoencoder [59] and the hierarchical autoencoder [60].

Thus, reduced order models can be constructed through a knowledge-based approach as well as a pure data driven methodology. A third technique is to combine these two approaches into a hybrid architecture, to benefit from both the governing equations as well as data from the chaotic system [61], such that the machine learning component is able to retrieve physical information lost due to modal truncation of the knowledge-based component (e.g. [7, 61, 62, 63]).

The study performed by Pathak et al. [61] was among the first to propose a hybrid model architecture, where an imperfection was added to the governing equations of a low-dimensional chaotic system. This imperfect knowledge-based component was combined with an echo state network to perform temporal predictions and it was shown that the hybrid model was able to perform better compared to the knowledge-based and deep learning components alone. In the same year, other hybrid models for low dimensional systems were proposed by Wan et al. [62] and Vlachas et al. [64]. The former combined a Galerkin model with a LSTM, such that the LSTM was able to encapsulate dynamics removed from the knowledge-based part,

while Vlachas et al. [64] utilized a mean stochastic model for the knowledge-based component. Later, Pawar et al. [63] proposed a model where the machine learning component was trained to predict the error between the true data and the results from the knowledge-based component, which was added up to the knowledge-based solution to retrieve lost dynamics. Finally, Lesjak and Doan [7] compared the architectures of Pathak et al. [61] and Pawar et al. [63], where the knowledge-based component was a POD-Galerkin model and the machine learning consisted of an echo state network, for several low dimensional chaotic systems. It was found that the architecture by Pathak et al. [61] was able to outperform the model from Pawar et al. [63].

Thus, the aforementioned showed the potential of hybrid reduced order modelling over knowledge-based and deep learning models for low dimensional chaotic systems. However, up the to authors knowledge, no research exists on a hybrid architecture using autoencoders that enables efficient predictions of high dimensional chaotic systems as previous studies only focused on low dimensional systems. This leads to the main research question of this study:

How can a hybrid knowledge-based/deep learning reduced order model approach be designed to enable the accurate short term and long term statistical predictions of high dimensional chaotic systems?

In order to construct the hybrid reduced order model such that it is able to handle high dimensional chaotic systems, data reduction is needed [11, 12]. The knowledge-based components is constructed through the POD-Galerkin approach [16]. In addition, to allow the recurrent neural network to efficiently handle the chaotic systems data, dimensional reduction using autoencoders is desired to reduce the spatial dimensional size [50, 51, 52, 53, 54]. Therefore, the first sub-question of this thesis reads:

- How can we enable an efficient way for the hybrid architecture to handle high dimensional data?
  - How can that be achieved with an autoencoder-like architecture?
  - What is the performance in terms of the short term prediction horizon and long term statistics of a chaotic system?
  - How does the latent space size of the autoencoder influence the above-mentioned performances?

In previous studies, its was shown that hybrid models have predictive benefits over the pure data driven and knowledge-based models alone [7, 61]. Therefore, similar interest is in the performance of the high dimensional hybrid architecture compared to its separate components, leading to the second sub question:

- What is the comparative performance of the hybrid model in terms of (i) short-term prediction horizon and (ii) long-term statistics of the knowledge-based ROM, data-only deep learning model and the hybrid ROM on chaotic systems of increasing complexity?

These questions will be addressed in this work. First, theoretical background on knowledge-based and deep learning reduced order modelling is provided in Chapter 2 and Chapter 3 respectively. Next, in Chapter 4, the high dimensional test cases are presented. Now, in this work, two hybrid models will be presented: Hybrid-A and Hybrid-FFNN. However, for the latter only preliminary results were obtained. Therefore, in Chapter 5 the proposed Hybrid-A model and methodology are discussed, followed by the results and discussion of this model in Chapter 6. Then, in Chapter 7, the Hybrid-FFNN model is introduced and some brief preliminary results are shown. Finally, the conclusions of this work are provided in Chapter 8.

Chapter 2

Theory: Knowledge-based reduced order modelling

In this chapter, the fundamentals of knowledge-based reduced order modelling are discussed. First, decomposition of the data through proper orthogonal decomposition is outlined in section 2.1. Next, Galerkin projection of the governing equations onto the POD modes is presented in section 2.2.

2.1 Proper orthogonal decomposition

The aim of proper orthogonal decomposition [17] is to decompose sequential data of a dynamical system, where snapshots at each time step are represented as a state vector $\boldsymbol{y}(\boldsymbol{x},t) \in \mathbb{R}^{N_{\text{phys}}}$, into a set of $N_{\text{pod}} = N_{\text{phys}}$ orthonormal modes $\boldsymbol{\Phi}(\boldsymbol{x}) \in \mathbb{R}^{N_{\text{pod}} \times N_{\text{pod}}}$ and corresponding temporal coefficients, such that the data is represented as [16, 65]

$$\boldsymbol{y}(\boldsymbol{x},t) = \sum_{j=1}^{N_{\text{pod}}} a_j(t)\Phi_j(\boldsymbol{x}) = \boldsymbol{a}\boldsymbol{\Phi}^T. \tag{2.1}$$

Here, t is the temporal coordinate, $\boldsymbol{x}$ are the spatial coordinates, N_{phys} is the length of the state vector indicating the number of degrees of freedom in the dynamical system, N_{pod} is the number of modes, $a_j(t)$ is the j-th temporal coefficient and $\Phi_j(\boldsymbol{x})$ is the j-th corresponding spatial-dependent mode. These modes are constructed to form the best linear basis such that the representation of the dynamical system requires the least amount of modes compared to any other linear basis [16].

Thus, in order to obtain the optimal basis, the error between the true data $\boldsymbol{y}(\boldsymbol{x},t)$ and its projection onto the POD modes must be minimized such that [16]

$$\left\langle \left|\left| \boldsymbol{y}(\boldsymbol{x},\boldsymbol{t}) - \frac{(\boldsymbol{y}(\boldsymbol{x},t),\boldsymbol{\Phi}(\boldsymbol{x}))}{||\boldsymbol{\Phi}(\boldsymbol{x})||^2}\boldsymbol{\Phi}(\boldsymbol{x}) \right|\right|^2 \right\rangle. \tag{2.2}$$

Here $\langle\cdot\rangle$ denotes the ensemble average, $(\cdot,\cdot)$ the inner product, $||\cdot||$ the Euclidean norm [16]. Equivalently, one can state to maximize [16]

$$\frac{\langle|(\boldsymbol{y}(\boldsymbol{x},t),\boldsymbol{\Phi}(\boldsymbol{x}))|^2\rangle}{||\boldsymbol{\Phi}||^2}. \tag{2.3}$$

Here, $|\cdot|$ is the absolute value [16]. Through variational calculus and the constraint that the modes should be orthonormal $||\boldsymbol{\Phi}||^2 = 1$ (for the derivation reference, is made to Holmes et al [16]) it is found that the POD modes correspond to the eigenvectors of the correlation matrix $\boldsymbol{C}$ of the fluctuations $\boldsymbol{y'}(\boldsymbol{x},t)$, namely [16, 65]

$$\boldsymbol{C} = \frac{1}{N_{\text{pod}}-1}\boldsymbol{Y'}\boldsymbol{Y'}^T, \tag{2.4}$$

where the fluctuations $\boldsymbol{y'}(\boldsymbol{x},t)$ are obtained by locally subtracting the temporal mean from the data. Here, $\boldsymbol{Y'} \in \mathbb{R}^{T_{\text{pod}}\times N_{\text{phys}}}$ is a matrix containing a concatenation of $T_{\text{pod}} \in \mathbb{N}_{>0}$ fluctuation snapshots, where each snapshot of the dynamical system is represented as a single row in the matrix [65]. In addition, the reason to use the correlation matrix can also be explained in a more intuitive manner. The essential idea of proper orthogonal decomposition is to study coherent structures in a dynamical system [65]. Therefore, correlations and thus the correlation matrix $\boldsymbol{C}$ between fluctuations at different spatial and temporal locations are of interest as these give indications of such coherent features [65].

Furthermore, upon considering the correlation matrix $\boldsymbol{C}$, it is observed that the trace equals the turbulent kinetic energy q [65]

$$q = \frac{1}{2}\boldsymbol{y}'_i\boldsymbol{y}'_i = \frac{1}{2}Tr(\boldsymbol{C}). \tag{2.5}$$

Now, the correlation matrix $\boldsymbol{C}$ is a symmetric matrix by definition and thus can be diagonalized as [65]

$$\boldsymbol{C} = \boldsymbol{\Phi}\Lambda_{\text{pod}}\boldsymbol{\Phi}^T, \tag{2.6}$$

where Λ_{pod} is the eigenvalue matrix containing the eigenvalues on the diagonal. Next, consider the trace of the correlation matrix and plug in the diagonalized form, such that

$$Tr(\boldsymbol{C}) = Tr(\boldsymbol{\Phi}\Lambda_{\text{pod}}\boldsymbol{\Phi}^T). \tag{2.7}$$

To provide a proof of concept, assume that the correlation matrix is a 2×2 square matrix. As a result, $\boldsymbol{\Phi}$ is a two dimensional matrix containing the eigenvectors and Λ_{pod} is a diagonal matrix of the corresponding eigenvalues. Therefore, the diagonal form of $\boldsymbol{C}$ can be written as

$$\boldsymbol{C} = \boldsymbol{\Phi}\Lambda_{\text{pod}}\boldsymbol{\Phi}^T = \begin{pmatrix} \Phi_{11} & \Phi_{21} \\ \Phi_{12} & \Phi_{22} \end{pmatrix} \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix} \begin{pmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{pmatrix}. \tag{2.8}$$

Writing out the matrix-matrix multiplication and computing the trace of the correlation matrix results in

$$Tr(\boldsymbol{C}) = [\Phi_{11}^2 + \Phi_{12}^2]\Lambda_{\text{pod},1} + [\Phi_{22}^2 + \Phi_{21}^2]\Lambda_{\text{pod},2}. \tag{2.9}$$

Now, since the modes are orthonormal, the inner product of a mode with itself is unity. The factor corresponding to $\Lambda_{\text{pod},1}$ is the self inner product of the first eigenvector and similar for $\Lambda_{\text{pod},2}$ and the second eigenvector. Thus, it is observed that

$$Tr(\boldsymbol{C}) = \Lambda_{\text{pod},1} + \Lambda_{\text{pod},2} = Tr(\Lambda_{\text{pod}}). \tag{2.10}$$

This shows that the trace of the eigenvalue matrix equals the trace of the correlation matrix and thus the turbulent kinetic energy [65, 66]

$$q = Tr(\boldsymbol{C}) = Tr(\Lambda_{\text{pod}}). \tag{2.11}$$

This results into an appealing aspect of the eigenvalues, because each POD mode (eigenvector) is connected to a single eigenvalue. Therefore, a certain portion of the turbulent kinetic energy can be assigned to each POD mode and thus all the modes can be ranked based on the amount of energy represented by them.

2.2 Galerkin Projection

In the previous section, orthonormal modes were obtained from data of a dynamical system through proper orthogonal decomposition. Here, Galerkin projection will be used to project the governing equations of a generalized dynamical system onto the set of obtained set of modes (following [7, 50, 67]).

Assume a generalized first order partial differential equation in discretized space with a linear $\mathfrak{L}$, a nonlinear component $\mathfrak{N}$ and a forcing term $\boldsymbol{f}_{for}(\boldsymbol{x},t)$ dependent on $\boldsymbol{x}$ and/or t, without writing down their specific forms (Eq. 2.12)

$$\dot{\boldsymbol{y}}(\boldsymbol{x},t) = \mathfrak{L}(\boldsymbol{y}(\boldsymbol{x},t)) + \mathfrak{N}(\boldsymbol{y}(\boldsymbol{x},t)) + \boldsymbol{f}_{for}(\boldsymbol{x},t). \tag{2.12}$$

In the previous section it was shown that data on the dynamical system can be decomposed into a set of optimal orthonormal modes and corresponding temporal coefficients through proper orthogonal decomposition. In order to Galerkin project Eq. 2.12 onto the set of modes from proper orthogonal decomposition, perform the inner product between Eq. 2.12 and the m-th POD mode $\boldsymbol{\Phi}_m(\boldsymbol{x})$, such that

$$\dot{\boldsymbol{y}}(\boldsymbol{x},t)\cdot\Phi_m(\boldsymbol{x}) = \mathfrak{L}(\boldsymbol{y}(\boldsymbol{x},t))\cdot\Phi_m(\boldsymbol{x}) + \mathfrak{N}(\boldsymbol{y}(\boldsymbol{x},t))\cdot\Phi_m(\boldsymbol{x}) + \boldsymbol{f}_{for}(\boldsymbol{x},t)\cdot\Phi_m(\boldsymbol{x}). \tag{2.13}$$

Now, $\boldsymbol{y}(\boldsymbol{x},t)$ can also be rewritten in terms of the temporal coefficients and POD modes according to Eq. 2.1. Plug the decomposition into Eq. 2.13 to obtain

$$(\dot{a}_j(t)\Phi_j(\boldsymbol{x}))\cdot\boldsymbol{\Phi}_m(\boldsymbol{x}) = \mathfrak{L}\left(a_j(t)\Phi_j(\boldsymbol{x})\right)\cdot\Phi_m(\boldsymbol{x}) + \mathfrak{N}\left(a_j(t)\cdot\Phi_j(\boldsymbol{x})\right)\Phi_m(\boldsymbol{x}) + \boldsymbol{f}_{for}(\boldsymbol{x},t)\cdot\Phi_m(\boldsymbol{x}). \tag{2.14}$$

Here, Einstein summation convention was used to avoid writing down all the summations. Next, an important property of the modes obtained through proper orthogonal decomposition is that they are orthonormal. Therefore, the inner product between the j-th modes and the m-th mode is always zero except when $j = m$ which returns unity. As a result, Eq. 2.14 reduces to

$$\dot{a}_j(t) = \mathfrak{L}\left(a_j(t)\Phi_j(\boldsymbol{x})\right)\cdot\Phi_m(\boldsymbol{x}) + \mathfrak{N}\left(a_j(t)\Phi_j(\boldsymbol{x})\right)\cdot\Phi_m(\boldsymbol{x}) + \boldsymbol{f}_{for}(\boldsymbol{x},t)\cdot\Phi_m(\boldsymbol{x}) \tag{2.15}$$

or in vector form

$$\dot{\boldsymbol{a}}(t) = \boldsymbol{\Phi}^T\mathfrak{L}\left(\boldsymbol{a}(t)\boldsymbol{\Phi}(\boldsymbol{x})\right) + \boldsymbol{\Phi}^T\mathfrak{N}\left(\boldsymbol{a}(t)\boldsymbol{\Phi}(\boldsymbol{x})\right) + \boldsymbol{\Phi}^T\boldsymbol{f}_{for}(\boldsymbol{x},t). \tag{2.16}$$

The latter two expressions (Eq. 2.15 and Eq. 2.16) represent the Galerkin projected equations of Eq. 2.12 onto a set of N_{pod} modes obtained through proper orthogonal decomposition. Then, Eq. 2.15 and Eq. 2.16 can be integrated in time through a numerical integration technique such as Euler Forward or a Runge-Kutta scheme to obtain the prediction $\boldsymbol{a}(t + \delta t)$. In order to transform the (predicted) temporal coefficient back into the physical system, a matrix vector multiplication with the time independent POD modes is performed, according to Eq. 2.1.

Chapter 3

Theory: Neural Networks

In the previous chapter, the fundamentals of knowledge-based reduced order modelling were discussed. However, aside from knowledge-based reduced order modelling, deep learning techniques can also be utilized to construct purely data-driven reduced order models. In this chapter, the fundamentals of deep learning reduced order modelling are presented. First, the artificial neuron and the neural network are discussed in section 3.1. Next, for the temporal predictions recurrent neural networks and more specifically echo state networks were used, which is outlined in section 3.2. Finally, convolutional neural networks are highlighted in section 3.3, which form the basis of autoencoders (section 5.2).

3.1 Artificial neuron

Before discussing more complex deep learning tools such as echo state networks and convolutional autoencoders, the building block of every neural network, the artificial neuron (Fig. 3.1), is presented. A typical artificial neuron allows N_{an} data inputs $x_{\text{an},j}$ with $j = 1...N_{\text{an}}$, which are each multiplied by a weight $w_{\text{an},j}$ to determine the relative importance of that data input. Next, all weight-input combinations are added up and a bias b is added, such that output $\tilde{y}_{\text{an}}$ of a single artificial neuron is expressed by Eq. 3.1

$$\tilde{y}_{\text{an}} = \sum_{j=1}^{N_{\text{an}}} w_{\text{an},j} x_{\text{an},j} + b. \tag{3.1}$$

So far, the artificial neuron only performs linear operators. Therefore to introduce nonlinearity, a nonlinear activation function f_{ac} is utilized such that the new nonlinear output of the

artificial neuron becomes

$$y_{\text{an}} = f_{ac}(\tilde{y}_{\text{an}}). \tag{3.2}$$

Eqs. 3.1 and 3.2 represent only a single artificial neuron, suitable for small and simple tasks. Therefore, in order to accommodate more complex systems, artificial neurons are stacked up over one or more layers to form artificial neural networks. It was shown that a sufficiently large neural network is a universal approximator of any continuous function [68].

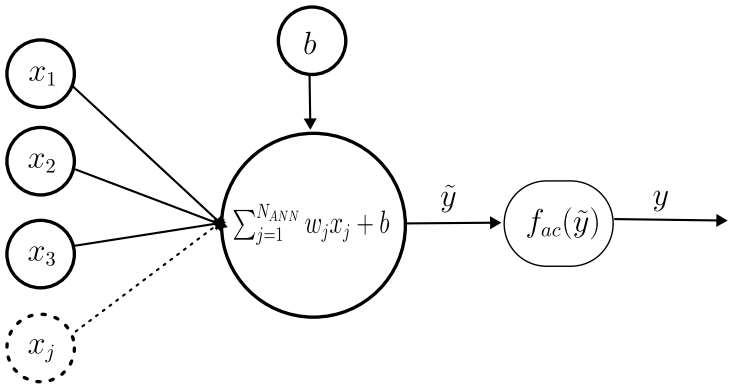


Figure 3.1: Image of the artificial neuron including the nonlinear activation function f_{ac}

A relatively simple type of neural network is the dense neural network (DNN) as depicted in Fig. 3.2. The first part of the DNN consists of the input layer that takes the inputs. Then, the second part consists of the hidden layers with a number of artificial neurons in each layer. Finally, the last layer is the output layer that takes the output from the last hidden layer and project its to the final output of the dense neural network. A neural network is called dense when the output of each neuron from a (hidden) layer connects to every neuron of the next layer. Finally, since information is only allowed to travel from the left to the right through the network, this is a type of feed-forward neural networks (FFNN).

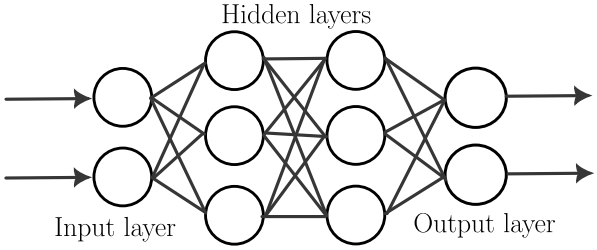


Figure 3.2: Image of a typical dense neural network. Here, the neural networks takes two input data point and consists of two hidden layers of three neurons each. Finally, the dense neural networks also outputs two values.

3.2 Recurrent neural network

In Chapter 2, the fundamentals on knowledge-based reduced order modelling to perform temporal predictions on chaotic system were discussed. However, such methods rely on full information on the governing equations. Thus, if such knowledge is not obtainable, predictions can be performed in a purely data-driven manner using recurrent neural networks, which are designed to handle sequential data such as time series. However, in order to train common recurrent neural networks such as the simple RNN and the LSTM [34], back propagation is required. This can result in vanishing and/or exploding gradients in the training process [34, 37]. However, such issues can be prevented through using a different type of recurrent neural networks, namely the echo state networks as proposed by Jaeger in 2001 [35] and is shown in Fig. 3.3. The echo state network is used in the data-driven models in this work and will be further elaborated upon in the section 3.2.1, based on the guidelines provided by Lukoševičius [69].

3.2.1 Echo state network

The echo state network (ESN) is a reservoir computing architecture [35] and is depicted in Fig. 3.3. Similar to all machine learning techniques, the primary goal in using an echo state network is to train the ESN with training data $\boldsymbol{H}(t-\delta t) \in \mathbb{R}^{T_\mathrm{h} \times N_\mathrm{h}}$ (Fig. 3.3A), such that the error between the predictions $\tilde{\boldsymbol{H}}(t) \in \mathbb{R}^{T_\mathrm{h} \times N_\mathrm{h}}$ and the true value $\boldsymbol{H}(t) \in \mathbb{R}^{T_\mathrm{h} \times N_\mathrm{h}}$ is minimal. Here, similar to section 2.1, $\boldsymbol{H}(t)$ is a concatenation of T_h time steps of the true data $\boldsymbol{h}(t) \in \mathbb{R}^{N_h}$, where N_h are the number of degrees of freedom (DoF) in the input data [69]. As observed in Fig. 3.3A, during training, data only flows in one direction through the network. Now, in order to perform predictions in time using input data $\tilde{\boldsymbol{h}}(t-\delta t) \in \mathbb{R}^{N_\mathrm{h}}$, the echo state network operates in prediction mode (Fig. 3.3B), where a feedback loop is added such that the ESN re-uses its output $\tilde{\boldsymbol{h}}(t) \in \mathbb{R}^{N_\mathrm{h}}$ as the input for the next prediction to allow for autonomous predictions [69].

The echo state network consists of three major building blocks (Fig. 3.3) [69]. First is the input matrix $\boldsymbol{W}^\mathrm{in} \in \mathbb{R}^{N_\mathrm{res} \times N_\mathrm{h}}$ that projects the input data $\tilde{\boldsymbol{h}}(t-\delta t)$ onto the high dimensional nonlinear internal state (expansion) $\boldsymbol{x}_\mathrm{res}(t) \in \mathbb{R}^{N_\mathrm{res}}$ of the reservoir consisting of N_res sparsely connected neurons, where the connectivity is enclosed in the internal matrix $\boldsymbol{W} \in \mathbb{R}^{N_\mathrm{res} \times N_\mathrm{res}}$ (second component). In addition, the internal state $\boldsymbol{x}_\mathrm{res}(t)$ is updated over time and thus also serves as the memory of the ESN. Finally, the third component is the output matrix $\boldsymbol{W}^\mathrm{out} \in \mathbb{R}^{N_\mathrm{h} \times N_\mathrm{res}}$ which projects the internal state into the prediction of the echo state network $\tilde{\boldsymbol{h}}(t)$ [69]. An important property during training and predicting is the echo state property, which states that a well configured ESN should, over time, not depend anymore on the initial reservoir conditions. Rather, it will be able to work autonomously purely driven by the input data [69].

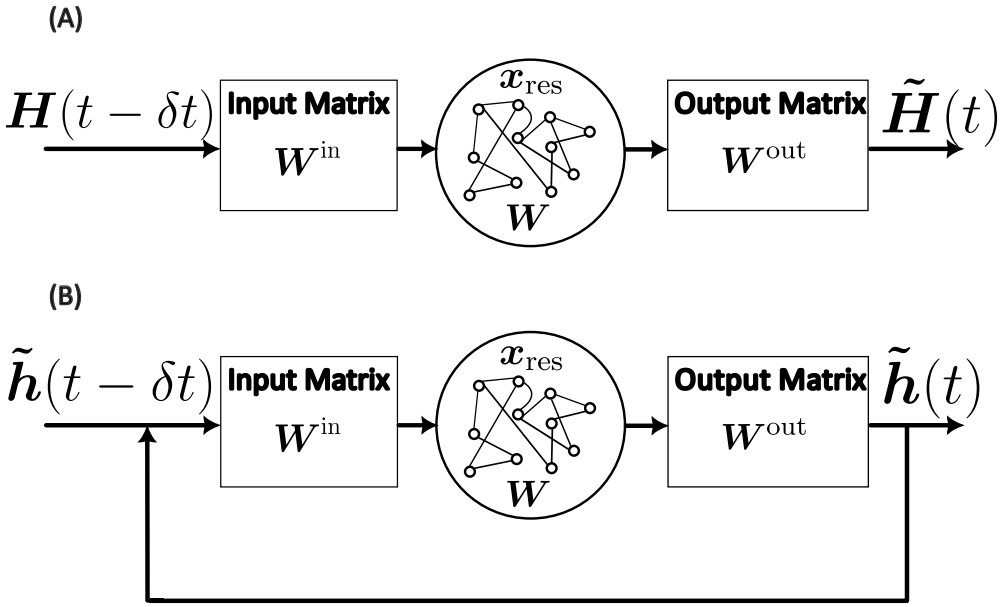


Figure 3.3: Architecture of an echo state network showing prediction mode (A) on top and training mode (B) on the bottom. Here, the reservoir is depicted in between the input matrix W^{in} and output matrix $\boldsymbol{W}^{\text{out}}$ and consists of the internal state $\boldsymbol{x}_{\text{res}}(t)$ and N_{res} connected neurons

The echo state network is able to perform temporal predictions by updating/integrating its internal state $\boldsymbol{x}_{\text{res}}(t)$ over time using [69]

$$\boldsymbol{x}(t) = (1-\alpha)\boldsymbol{x}_{\text{res}}(t-\delta t) + \alpha \tanh\left(\boldsymbol{W}^{\text{in}}\tilde{\boldsymbol{h}}(t-\delta t) + \boldsymbol{W}\boldsymbol{x}_{\text{res}}(t-\delta t)\right) \tag{3.3}$$

as the update equation. Here $\alpha \in [0,1]$ is the leaking integration rate that determines how much of the internal state of the previous time step directly influences the new state [69]. After a prediction in time, in accordance with Pathak et al. [61] and Lesjak and Doan [7], a quadratic transformation is applied to the internal state, such that all even numbered neurons are squared, while the odd numbered neurons remain the same as it was shown to provide improved performance for other chaotic systems [61]. Finally, the transformed internal state is multiplied with the output matrix to obtain the single time step prediction of the ESN [69]

$$\tilde{\boldsymbol{h}}(t) = \boldsymbol{W}^{\text{out}}\boldsymbol{x}_{\text{res}}(t). \tag{3.4}$$

In order to construct an echo state network, the input matrix $\boldsymbol{W}^{\text{in}}$ and the internal matrix $\boldsymbol{W}$ are prescribed. Therefore, these are not altered during training and while performing temporal predictions on the dynamical system [69]. The reservoir state is updated in time and the output matrix $\boldsymbol{W}^{\text{out}}$ is obtained through training. Furthermore, the echo state network is optimized using five hyperparameters, namely the sparsity $\langle d \rangle$, spectral radius $\rho(\boldsymbol{W})$, input scaling σ_{in}, Tikhonov factor β and the leaking rate α. The first four are used for the input and internal matrices, whereas the Tikhonov factor is used during training. In

the following paragraph, the details behind constructing the input and internal matrix are presented.

Construction of $\boldsymbol{W}^{in}$ and $\boldsymbol{W}$
For the construction of the input and internal matrix, the guidelines by Lukoševičius [69] were followed and the echo state network numerical model of Lesjak and Doan [7, 66] was used. As mentioned before, $\boldsymbol{W}^{in}$ and $\boldsymbol{W}$ define the structure of the ESN and are not altered during the training and prediction phase [69].

The first step is to fix the size of the reservoir and thus the number of artificial neurons [69]. In general, a larger reservoir is able to capture the dynamics better and thus results in more accurate predictions [69]. Therefore, the reservoir size should theoretically be at least equal to the amount of input data $T_\mathrm{h} \cdot N_\mathrm{h}$. However, in practice, the size of the reservoir is limited by computational resources and a reservoir as large as possible should be used. In addition, often correlations exist within the data reducing the required amount of neurons in the reservoir [69].

The input matrix is constructed, similar to Pathak et al. [61] and Lesjak and Doan [7, 66] such that each input variable is mapped to one randomly selected neuron in the reservoir. Hence, each row in the input matrix only contains a single nonzero element (see Eq. 3.3 for the matrix-vector multiplication of the input matrix with the input data). Furthermore, during this mapping, the input variable is multiplied with a randomly selected value from the uniform distribution between $[-\sigma_\mathrm{in}, \sigma_\mathrm{in}]$ to allow for scaling of the inputs.

Next, the internal matrix is constructed through randomly sampling from a uniform distribution between [-1, 1]. Here, the connectivity of the neurons $\langle d \rangle$ is generally low, such that each neuron is only connected to a few other neurons, resulting in a sparse internal matrix [69]. Then, in order to construct the internal matrix, also the spectral radius is required, which is the largest absolute eigenvalue of the internal matrix. This hyperparameter is of profound importance as it determines whether the ESN obeys the echo state property [35, 69]. Typically, an ESN with a spectral radius lower than unity will display the echo state property [69]. Furthermore, if the spectral radius is larger than one and the inputs are non-zero, the echo state property is often obeyed as well [69]. So, in order to set the spectral radius, the randomly sampled internal matrix is divided by its largest absolute eigenvalue and then multiplied by a desired constant spectral radius determined through hyperparameter tuning as discussed in section 5.7.1 [7, 66].

Training of ESN / Computing $\boldsymbol{W}^{\mathrm{out}}$
The primary advantage of an echo state network over other recurrent neural networks such as the simple RNN and the LSTM is its convenient training procedure [35, 57, 69]. As shown in Eq. 3.4, the output of the ESN is obtained through a matrix-vector multiplication of the output matrix with the (updated) internal state of the reservoir $\boldsymbol{x}_\mathrm{res}(t)$.

During training, the training data $\boldsymbol{H}(t-\delta t) \in \mathbb{R}^{T_\mathrm{h} \times N_\mathrm{h}}$ is used as input for the echo state network, which is used to obtain the temporal evolution of the internal state $\boldsymbol{x}_\mathrm{res}(t)$ at each time step, where all internal states are concatenated into a matrix $\boldsymbol{X}_\mathrm{res}(t) \in \mathbb{R}^{T_\mathrm{h} \times N_\mathrm{res}}$. Following Eq. 3.4, the output matrix $\boldsymbol{W}^{\mathrm{out}}$ can be determined from [69]

$$\boldsymbol{H} = \boldsymbol{W}^{\text{out}} \boldsymbol{X}_{\text{res}}. \tag{3.5}$$

To avoid problems caused by either overfitting or feedback instability, ridge regression (or Tikhonov regularisation) is used [69]

$$\boldsymbol{W}^{\text{out}} = \boldsymbol{H}\boldsymbol{X}_{\text{res}}^{T} \left(\boldsymbol{X}_{\text{res}}\boldsymbol{X}_{\text{res}}^{T} + \beta \boldsymbol{I}\right)^{-1}. \tag{3.6}$$

Here, $\boldsymbol{X}_{\text{res}} \in \mathbb{R}^{T_h \times N_{\text{res}}}$ is a matrix containing the reservoir states during training and $\boldsymbol{I}$ is the identity matrix.

The plain echo state network, however, is not suitable for temporal predictions on high dimensional systems for two reasons. First, in the Tikhonov regularisation (Eq. 3.6) a matrix inversion is required, resulting in a large (often unfeasible) memory usage. Secondly, as the spatial dimensions of the chaotic systems increases, a larger reservoir is needed to accurately capture the dynamics. However, the computational costs of the echo state network increases with increasing reservoir size. Thus, for large system the computational cost becomes large. Therefore, dimensional reduction of the data is required similarly to the modal reduction used in the knowledge-based reduced order models. In the next section, such a nonlinear deep learning dimensional reduction using convolutional neural networks is discussed. Finally, as aforementioned, the echo state network is defined by the five hyperparameters, which must be tuned for optimal performance. In this work, this tuning is obtained through Bayesian Optimisation, which is further elaborated upon in Appendix A.

3.3 Deep learning data (de)compression

In the previous section, the echo state network was described, which is the recurrent neural network used in this research to perform data driven predictions. However, echo state networks are not suitable for high dimensional data due to Tikhonov regularisation, because a matrix inversion is required resulting in an unpractical large memory use. Furthermore, for high dimensional data, a very large number of neurons inside the reservoir would be required resulting in large memory use as well as long computational time. Therefore, in order to allow the echo state network to perform efficient temporal predictions of high dimensional‘chaotic systems, nonlinear data compression and decompression is utilized through convolutional neural networks and transpose convolutional neural networks, which are further outlined in section 3.3.1.

3.3.1 (Transpose) Convolutional neural networks

Convolutional neural networks [70] is a successful deep learning tool finding many applications in fields such as computer vision and image recognition [71] and they form the basis of more complex machine learning methods such as convolutional autoencoders [51, 53, 54, 57]. Similar to the neural network outlined in section 3.1, a convolutional neural network (CNN) consists of layers of neurons with weights and biases [71]. However, in contrast to dense neural works, the CNN is used to deal with images data as it can handle and learn correlations on a local level [71]. In this section, the convolutional neural network is presented [71] together with the implementation and architecture in this thesis following Racca et al. [57]. In this work, the (transpose) convolutional neural networks were the building blocks of the autoencoder (section 5.2) to allow for a nonlinear deep learning dimensional reduction.

The primary task of a convolutional neural network is to extract feature maps through the use of D_f (trainable) filter kernels $\boldsymbol{q}_k \in \mathbb{R}^{N_f \times N_f \times D_f}$ (N_f gives filter kernel first and second dimension) that are convoluted over the original input image $\boldsymbol{Q} \in \mathbb{R}^{N_{\boldsymbol{Q}} \times N_{\boldsymbol{Q}} \times N_c}$ [71]. Here, N_c are the number of parallel images in the input, for example for two dimensional fluid flow $N_c = 2$. Such filter kernels are commonly square matrices and are trained to extract certain patterns from the data, where each filter specializes on single pattern [71]. In the convolutional process, the filter kernel is placed on the top left of the image and is convolved row by row from top to bottom over the image with a prescribed stride s (step size). At each position, an element-wise multiplication occurs between the filter kernel and the local region of the image, after which these multiplications are summed up and stored in the feature map $\boldsymbol{Q}_F \in \mathbb{R}^{N_{\boldsymbol{Q}_F} \times N_{\boldsymbol{Q}_F} \times D_f}$, which is depicted in Fig. 3.4 [71]. Here N_F gives the size of the feature map and note that the number of features maps is equal to the number of filters in the CNN. Thus, the stride s allows for dimensional reduction of the input image to a lower dimensional feature map. Another common dimensional reduction technique is max-pooling [71].

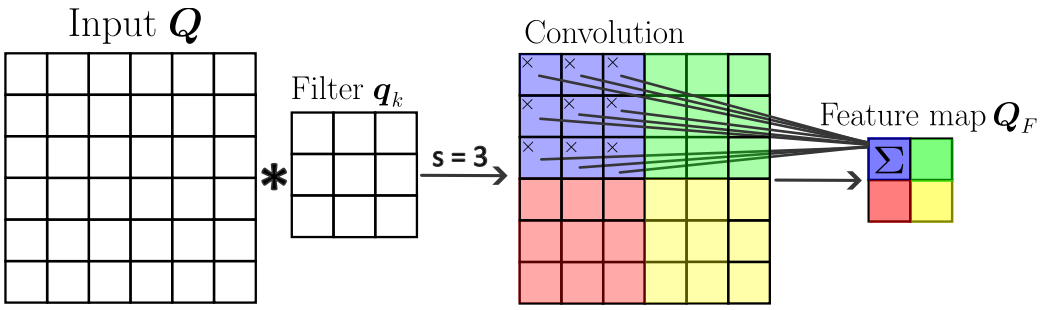


Figure 3.4: Image showing the working principle of a convolutional neural network. The filter kernel is convoluted over the input image $\boldsymbol{Q}$ with a stride of three. At each filter location, an element wise multiplication occurs between the filter kernel and the selected local region of $\boldsymbol{Q}$ after which they are summed up to form 1 entry of the feature map.

The essence of the convolutional neural network is to train the entries of the filter kernel to extract specific features from the image. This also shows the link with the artificial neuron of section 3.1. Here, the entries of the filter kernel are the "neurons" weights, which are multiplied with the local regions of $\boldsymbol{Q}$. Additionally, a bias b can be added to each filter kernel. Note here, that during the convolution process between the input image and a single

kernel, the kernel does not change [57, 71]. Therefore, compared to a fully connected neural network, significantly less weights are required [57]. Finally, in order to add nonlinearity, each entry of the feature map is also applied to a tangent hyperbolic function [57], resulting in the following equation for each entry of the feature map (from Racca et al. [57])

$$\boldsymbol{Q}_{F(i,j,m)} = \tanh\left(\sum_{p_1=1}^{N_f}\sum_{p_2=1}^{N_f}\sum_{l=1}^{N_c}\boldsymbol{Q}_{s(i-1)+p_1,s(j-1)+p_2,l}w_{p_1,p_2,l,m} + b_m\right). \tag{3.7}$$

Here, w is the weight of each entry of the filter.

Now consider Fig. 3.4, if the stride s would have been equal to one, the feature map $\boldsymbol{Q}_f$ would become a 4×4 matrix, which is smaller than the input image $\boldsymbol{Q}$. This is due to the convolution process near the boundaries of $\boldsymbol{Q}$ since the filter kernel cannot "move outside" the input image and thus information near the edges would be lost as the boundary element of the input matrix are not taken into account in the convolution. To prevent this, a typical technique is to apply padding to the input image as shown in Fig. 3.5 [57, 71]. In padding, one or multiple layer(s) of additional matrix elements are added around the input image. Typically, to avoid a contribution of these elements in the convolution each entry of the padding layers is set to zero (zero-padding). However, when a dynamical system has periodic boundary conditions, the padding layers can utilize this periodicity as shown in Fig. 3.5.

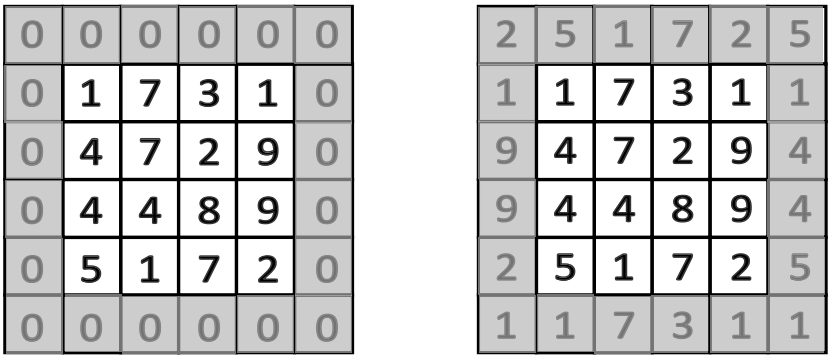


Figure 3.5: Image showing the working principle of zero-padding (left) and periodic-padding (right) for an input image $\boldsymbol{Q}$. Here, the padding layers are shown in gray. Here the numbers of the matrix were chosen randomly and do not represent the test cases considered.

A convolutional neural network is used to extract feature maps from the input image and to reduce the dimensionality. However, the inverse procedure to extract feature maps while increasing the dimensionality of the data is possible through the use of transpose convolutional neural networks (TCNN) as depicted in Fig. 3.6 [57]. The idea is to obtain an image $\tilde{\boldsymbol{Q}}$ in the dimensionality of the input data $\boldsymbol{Q}$ from the feature maps through the transpose convolutions neural networks. Here, each element of the feature map is multiplied element-wise with the filter kernel $\boldsymbol{q}_k$. The resulting 3×3 matrix of each feature map element is placed in a new matrix $\tilde{\boldsymbol{Q}}$ in the same order of the feature map elements. If the stride is smaller than 3 and thus the coloured blocks overlap, their contribution will be locally summed up [57].

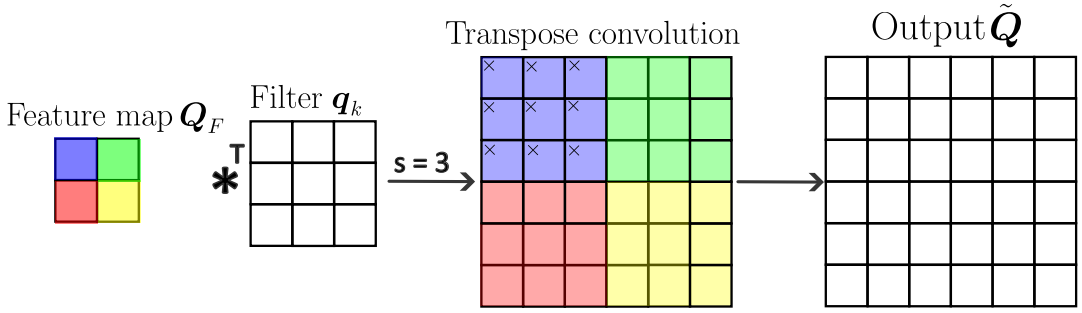


Figure 3.6: Image showing the principle of a transpose convolutional neural network.

Chapter 4

Test case: Kolmogorov Flow

Fluid dynamical systems display chaotic behaviour (turbulence) above a certain Reynolds number Re [8]. Furthermore, due to the wide range of spatial scales of turbulence, such chaotic behaviour is often concurrent with high dimensionality. For these reasons, in this research, the constructed reduced order models was applied to a fluid dynamical system, namely the two dimensional Kolmogorov flow as it was shown to possess a wide variety of flow regimes [72]. In addition, Chandler and Kerswell [73] showed the presence of three stages in the global dissipation and the kinetic energy of the velocity fluctuations as a function of the Reynolds number. First, at lower Reynolds numbers, a laminar stage was observed. Increasing the Reynolds number pushes the flow into the transitional regime and finally for large Reynolds number an asymptotic regime was found [73]. Therefore, in this work, the Kolmogorov flow was considered at Reynolds numbers $Re = 20$ and 34. Here, the Kolmogorov flow at $Re = 20$ was shown to be before the onset of turbulence [73]. In contrast, $Re = 34$ was located in the transitional regime [73] and thus chaotic behaviour was present [57, 73]. Thus, by increasing the Reynolds number of the flow, the constructed models were tested on dynamical and chaotic systems of increasing complexity. Much larger Reynolds numbers would have resulted in significantly more computational resources.

4.1 2D Kolmogorov flow

The two dimensional Kolmogorov flow is described by the incompressible Navier Stokes equations with an additional sinusoidal forcing term in the x-direction as given in Eq. 4.1.

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} &= -\nabla p + \frac{1}{Re}\nabla^2\boldsymbol{u} + \boldsymbol{f}_{\mathrm{KF}}, \\ \nabla\cdot\boldsymbol{u} &= 0, \\ f_{\mathrm{KF},x} &= \sin(n_{\mathrm{KF}}y), \\ f_{\mathrm{KF},y} &= 0. \end{aligned} \tag{4.1}$$

Here x and y were the spatial coordinates, $\boldsymbol{u} = [u, v]$ was the two dimensional flow with components u and v, p was the pressure, $\boldsymbol{f}_{\mathrm{KF}}$ represented the Kolmogorov (body) forcing with x-component $f_{\mathrm{KF},x}$ and y-component $f_{\mathrm{KF},y}$ and $n_{\mathrm{KF}} = 4$ was the frequency of the forcing. The Kolmogorov flow is double periodic on $x \in [0, 2\pi]$ and $y \in [0, 2\pi]$ domain, with N_x grid points in x and N_y grid points in y. To provide an impression, in Fig 4.1, snapshots of u, v and the absolute velocity $\sqrt{u^2 + v^2}$ are presented for the two-dimensional Kolmogorov flow at $Re = 20$ and $Re = 34$.

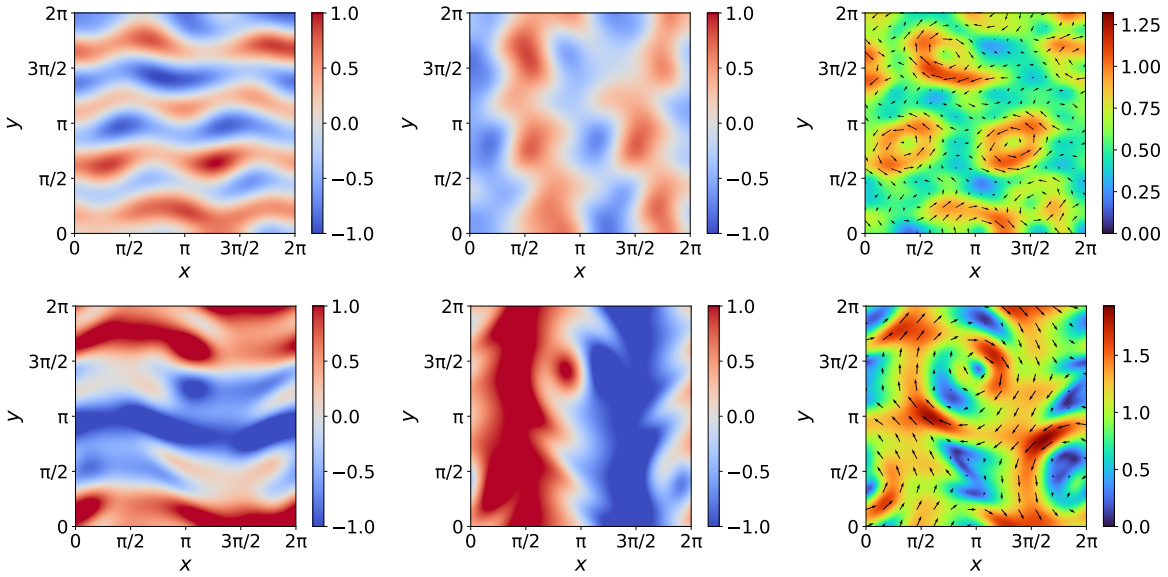


Figure 4.1: Images showing a snapshot in time of the u-component (left), v-component (middle) and their norm (right) of two-dimensional Kolmogorov flow at $Re = 20$ (top row) and $Re = 34$ (bottom row).

Furthermore, the temporal mean flow of u and v for $Re = 34$ are depicted in Fig. 4.2, which were computed over 300,000 integer time steps [57]. It was shown for the mean u-component of the velocity follows the sinusoidal pattern of the forcing with 4 periods in the domain.

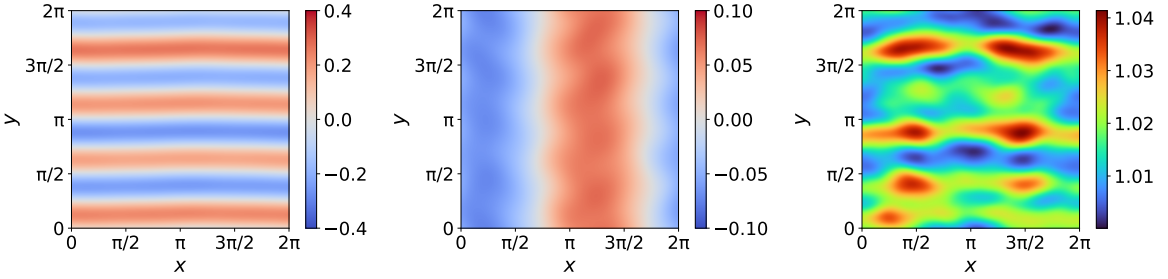


Figure 4.2: Three plots showing the mean flow of the two-dimensional Kolmogorov flow for $Re = 34$. Here, the left plot is for the u-component, the middle plot for v and the right plot depict the norm of the velocity.

To obtain the training, validation as well as the reference data set, Eq. 4.1 was solved through direct numerical simulation using the open-source KolSol library [74] in Python, which is a

pseudo-spectral solver using the Fourier-Galerkin method [75] with a fourth order Runge-Kutta scheme. KolSol solved the system for a prescribed N_k symmetric wave numbers and thus $(2N_k + 1)$ modes in each direction. Then, the integrated solution in Fourier space is transformed onto a physical grid of $(2N_k + 2)$ grid points in each direction. An even number of physical grid points was used due to the architecture of the autoencoder used in this work (section 5.2).

In order to provide a measure of the characteristic chaotic behaviour the Lyapunov exponent λ_{Lyap} was used, which indicates the average rate at which trajectories with slightly different initial conditions separate over time [76]. The exponents were computed using the numerical code by Racca et al. [57] and further details are provided in Appendix B. Furthermore, the characteristic time scale, the Lyapunov time, was defined as the inverse of the largest obtained Lyapunov exponent λ_{Lyap}. In this work, 15 Lyapunov exponents were computed per flow case and the largest was used to provide the time scale of the chaotic behaviour. In addition, the computed Lyapunov exponents were also used to obtain a measure of the latent space dimensions of the autoencoder (section 3.3) through the Kaplan-Yorke dimension D_{KY} [77] (Eq. 4.2)

$$D_{\text{KY}} = k + \frac{\sum_{i=1}^{k} \lambda_{\text{Lyap},i}}{\lambda_{\text{Lyap},k+1}}. \tag{4.2}$$

Here, the summation was performed over the k largest Lyapunov exponents. The idea was that the Kaplan-Yorke dimension provided the upper bound of the amount of dimensions that still described the chaotic attractor. Therefore, the latent space dimensions in the autoencdoer should not be lower than the Kaplan-Yorke dimension [57].

***Re* = 20**

The Kolmogorov flow at $Re = 20$ was solved on a grid consisting of $N_k = 8$ symmetric wave numbers. Here, the symmetric wave number was selected through the convergence of the mean domain averaged kinetic energy in the system as a function of the symmetric wave number (left plot of Fig. 4.3), where the kinetic energy was computed as

$$E(t) = \frac{1}{(2\pi)^2} \int_0^{2\pi} \int_0^{2\pi} \frac{1}{2} u_i u_i dx dy. \tag{4.3}$$

It was observed that the mean kinetic energy became approximately constant at $N_k = 8$. Furthermore, since $N_k = 8$ symmetric wave numbers were chosen, the system was solved using 17 modes and 18 physical grid points in each direction. In addition, a time interval $\delta t = 0.1$ was chosen for the temporal discretization and the data storing frequency. The integration occurred over 50,000 integer time steps after an initial 3,000 integer time step to integrate past the transient, during which no data was stored. Despite being a periodic flow and thus strictly no Lyapunov exponent is defined, 15 Lyapunov exponents were computed to obtain the Kaplan-Yorke dimension using Eq. 4.2. Here, 15 different initial conditions were used such that 15 exponents were obtained from which $D_{\text{KY}} = 3.2$ was computed.

Furthermore, the kinetic energy in the flow as a function of time was considered as depicted in the right plot of Fig. 4.3 for $N_k = 8$. It was observed that the flow showed periodic behaviour with a period of approximately $T_{\text{per}} = 170$.

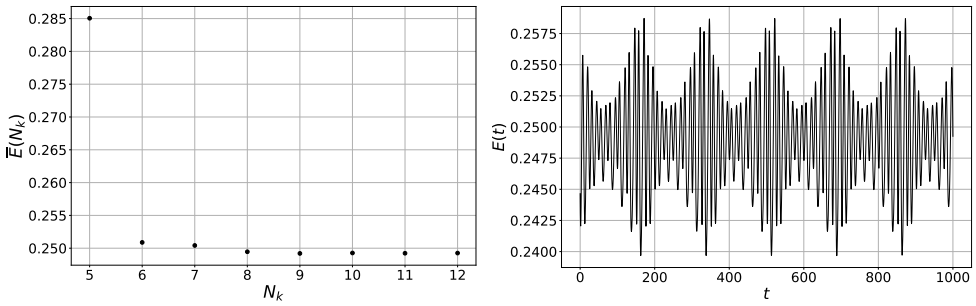


Figure 4.3: Left: Convergence of the temporal mean of the domain averaged kinetic energy $\overline{E}(N_k)$ as a function of the number of symmetric wave numbers used in the solver for $Re = 20$. Right: Domain averaged kinetic energy $E(t)$ of the Kolmogorov flow as a function of time for $N_k = 8$.

$Re = 34$

For $Re = 34$, the flow was solved on a finer grid consisting of 32 symmetric wave numbers and thus 65 modes in each direction and 66 grid points in each spatial direction with a temporal discretization of $\delta t = 0.01$ for 100,000 integer time steps. In addition, 1,000 integer time steps were used to compute past the initial transient regime and this data was not stored. The spatial discretization was determined from the temporal mean of the kinetic energy spectrum (Fig. 4.4) for a given number of modes since the kinetic energy should go to zero for the smallest scales (highest wave numbers magnitudes) in a turbulent flow. The kinetic energy spectrum is computed in Fourier space as (from the KolSol library)

$$E\chi = \frac{1}{(2N_k)^2} \sum_{\chi} \frac{1}{2} \hat{\boldsymbol{u}}(\chi) \cdot \hat{\boldsymbol{u}}^*(\chi). \tag{4.4}$$

Here, $\hat{\boldsymbol{u}}$ is the flow field in Fourier space, the asterisk represented its complex conjugate and χ is the wave number magnitude at which the kinetic energy was computed. The mean was computed over 1,000 integer time steps, where the data was stored every $\delta t = 0.1$ and thus 10,000 samples were used.

In order to compute the Kaplan-Yorke dimension of a given chaotic attractor (Eq. 4.2) for the latent space of the autoencoders, the system with prescribed initial condition together with 15 simulations with slightly perturbed initial conditions were integrated over for 10,000 time steps to obtain 15 Lyapunov exponents. Next, Eq. 4.2 was used to computed the Kaplan-Yorke dimension $D_{\text{KY}} = 9.5$ and the largest Lyapunov exponent $\lambda_{\text{Lyap}} = 0.067$ provided the characteristic Lyapunov time $T_{\text{Lyap}} = 1/0.067 \approx 14.9$. Both these values coincide with those obtained by Racca et al. [57].

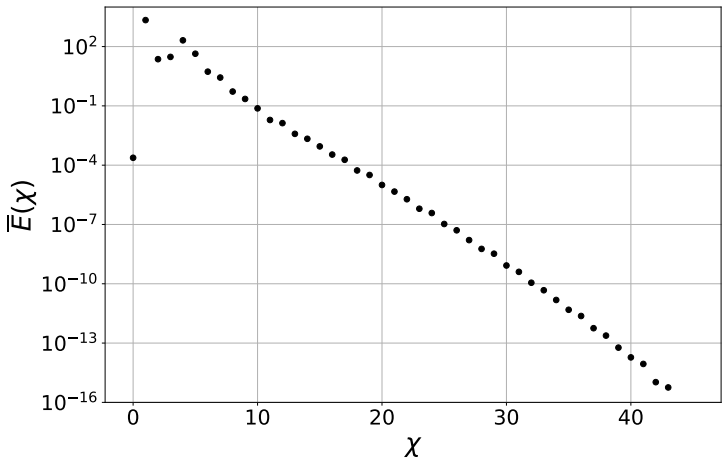


Figure 4.4: Plot showing the temporal mean of the kinetic energy spectrum as a function of the wave number magnitude for two dimensional Kolmogorov flow at $Re = 34$

Chapter 5

Methodology and Hybrid-A architecture

In the previous two chapters, the fundamental theory and techniques behind knowledge-based and pure deep learning reduced order modelling were presented. The primary aim of this thesis was to construct a hybrid knowledge-based/deep learning reduced order model to allow for time-accurate short-term and long-term statistical predictions. Therefore, in this chapter, the proposed Hybrid-A model and methodology are presented. First, in section 5.1, the derivation of the knowledge-based reduced order model for two-dimensional Kolmogorov flow (chapter 4) is presented. Next, the deep learning nonlinear dimensional (de)compression technique based on (transpose) convolutional neural networks (section 3.3.1), the multi-scale autoencoder [55, 56, 57] is discussed in section 5.2. Then, the combination of the multi-scale autoencoder and the echo state network (section 3.2.1) into a pure data-driven reduced order model is outlined in section 5.3 (following the work of Racca et al. [57]). Fourthly, the proposed Hybrid-A model are presented in section 5.4. Next, the comparison criteria to quantify and compare the performances of the reduced order models (section 5.5) are discussed. Then, in section 5.6 information is provided on the reduction of the number of POD modes for the knowledge-based and the Hybrid models. The implementation of the hyperparameter tuning as well as the validation strategies [78] and the settings are discussed in section 5.7.1. Finally, the model settings for the temporal predictions are presented in section 5.7.2

5.1 Knowledge-based reduced order Kolmogorov flow

In Chapter 2, the theory behind the knowledge-based reduced order model was presented, such as proper orthogonal decomposition and Galerkin projection of the governing equations onto a set of orthonormal POD modes. The latter was shown for a generalized partial differential equation in section 2.2. Therefore, in this section, the knowledge-based reduced order model for the two-dimensional Kolmogorov flow was derived using the work of Wang et al. [79].

As presented in Chapter 4, the governing equation of two-dimensional Kolmogorov flow were written as

$$\begin{aligned}\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} &= -\nabla p + \frac{1}{Re}\nabla^2\boldsymbol{u} + \boldsymbol{f}_{\mathrm{KF}},\\ \nabla\cdot\boldsymbol{u} &= 0,\\ f_{\mathrm{KF},x} &= \sin(ny)\\ f_{\mathrm{KF},y} &= 0.\end{aligned}$$

and solved on a double periodic domain. Following the notation from proper orthogonal decomposition in section 2.1, the velocity vector $\boldsymbol{u} = [u, v]$ was represented in the knowledge-based model as a one-dimensional vector $\boldsymbol{y}(\boldsymbol{x}, t) \in \mathbb{R}^{N_{\mathrm{phys}}}$, where $N_{\mathrm{phys}} = 2 \cdot N_x \cdot N_y$, with the first $N_x \cdot N_y$ elements corresponding to u and the second $N_x \cdot N_y$ corresponding to v. Furthermore, assume that orthonormal modes $\boldsymbol{\Phi}(\boldsymbol{x})$ obtained from the true data are available through proper orthogonal decomposition. First, discretize the governing equations and thus replace $\boldsymbol{u}$ with $\boldsymbol{u}_h$, where the latter is the discretized variable, and discretize the forcing term such that

$$\begin{aligned}\dot{\boldsymbol{u}}_h + (\boldsymbol{u}_h\cdot\nabla)\boldsymbol{u}_h &= -\nabla p + \frac{1}{Re}\nabla^2\boldsymbol{u}_h + \boldsymbol{f}_{\mathrm{KF},h},\\ \nabla\cdot\boldsymbol{u}_h &= 0,\\ f_{\mathrm{KF},x} &= \sin(ny)\\ f_{\mathrm{KF},y} &= 0.\end{aligned}$$

Utilizing the results of Wang et al. [79], it followed that the Galerkin projected governing equations for each component of the POD temporal coefficient from $j = 1$ to $j = N_{pod}$ became (using Einstein summation convention)

$$\dot{a}_j = -\frac{2}{Re}\left[\frac{\nabla\Phi_p + (\nabla\Phi_p)^T}{2}\right]\cdot(\nabla\Phi_j)a_p(t) - [\Phi_p\cdot\nabla\Phi_q]\cdot\Phi_j a_p(t)a_q(t) + \mathbf{f}_{\mathrm{KF}}(\mathbf{x},t)\cdot\Phi_j. \quad (5.1)$$

Eq. 5.1 represented the knowledge-based reduced order model for the two-dimensional Kolmogorov flow. Now, the forcing term $\boldsymbol{f}(\boldsymbol{x}, t)$ only consisted of a x-component. Therefore, in the inner product, only the x component of mode Φ_j remained, resulting in

$$\dot{a}_j = -\frac{2}{Re}\left[\frac{\nabla\Phi_p + (\nabla\Phi_p)^T}{2}\right]\cdot(\nabla\Phi_j)a_p(t) - [\Phi_p\cdot\nabla\Phi_q]\cdot\Phi_j a_p(t)a_q(t) + \sin(ny)\Phi_j^x, \quad (5.2)$$

where Φ_j^x is the x-component of the modes. This set of ordinary differential equations for the time-dependent POD coefficients were numerically integrated in time using the fourth-order Runge-Kutta scheme combined with exponential time differencing [80, 81, 82]. Then, Eq. 2.1 was used to compute the physical system from the predicted temporal coefficient and the time independent POD modes at each time step.

Now, Eqs. 5.1 and 5.2 still contained all the modes from proper orthogonal decomposition. For this reason, the number of modes were truncated to obtain a higher computational efficiency. However, as a cost, the accuracy of the model compared to the true solution decreased. Therefore, a retained number of modes must be chosen properly, which will be further elaborated upon in section 5.6.

5.2 Multi-scale autoencoder

As discussed in section 3.3.1, convolutional and transpose convolutional neural networks are typically used to extract feature maps and to compress and decompress images [71]. Now, in order to efficiently and accurately compress the data from the physical system such as the Kolmogorov flow into a latent space format for the echo state network, a multi-scale autoencoder consisting of a series of (transpose) convolutional neural networks was used. The multi-scale autoencoder was proposed by Hasegawa et al. [55] and later used by Nakamura et al. [56] and Racca et al. [57]. In this work, the numerical code of the multi-scale autoencoder from Racca et al. [57] was utilized.

The primary idea of an autoencoder (Fig. 5.1) is to first use an encoder to compress the original data $\boldsymbol{Q} \in \mathbb{R}^{N_x \times N_y \times 2}$ into a latent space format $\boldsymbol{Q}_{\text{lat}} \in \mathbb{R}^{N_{\text{lat},x} \times N_{\text{lat},y} \times N_c}$ of dimensions

$$\left[N_{lat,x}, N_{lat,y}, \frac{N_{lat}}{N_{lat,x} \cdot N_{lat,y}} \right],$$

where $N_{lat,x} \ll N_x$, $N_{lat,x} \ll N_y$ and $N_{lat} \ll N_{\text{phys}}$ through the use of multiple layers of convolutional neural networks (section 3.3.1) [55, 56, 57]. Here, the last dimension of the input image represented the two velocity components. Simultaneously, a decoder is trained to decompress the latent space back into the original dimensional using transpose convolutional neural networks to obtain the decode data $\boldsymbol{Q}_{\text{dec}} \in \mathbb{R}^{N_x \times N_y \times 2}$ with a minimal loss between the original and decoded images [55, 56, 57], where the last dimensions represented the flow velocity components. Furthermore, in a multi-scale autoencoder, multiple encoders and decoders are operated in parallel and all individual contributions are added up to produce the latent space and decompression respectively. By using multiple encoders/decoders more features at different spatial dimensions can be extracted from the input data [55, 56, 57].

In this work, following Racca et al. [57], three parallel encoders and three parallel decoders were chosen. The first encoder/decoder had a filter kernel size of (3×3), the second (5×5) and the third (7×7) for each test case. In the following of this section, the general structure of the encoder and decoders are outlined based on the numerical code by Racca et al. [57].

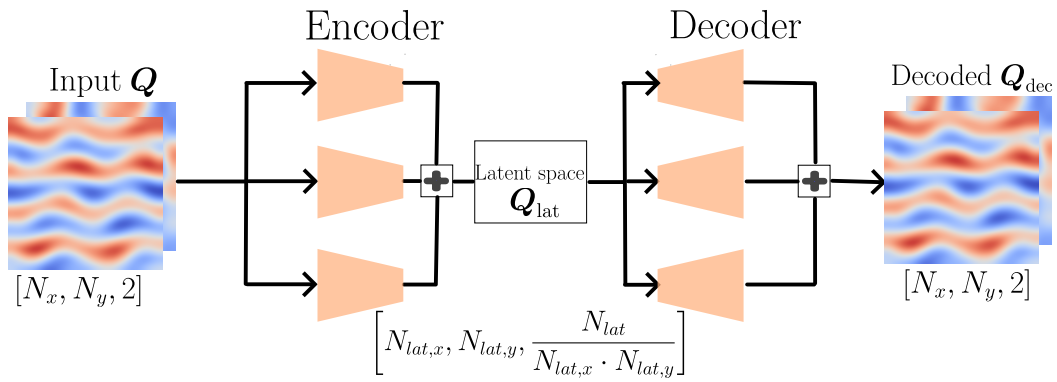


Figure 5.1: Multi-scale autoencoder consisting of three encoders to produce the latent space $\boldsymbol{Q}_{\text{lat}}$ from input data $\boldsymbol{Q}$ and 3 decoders to decompress the latent space into the decoded format $\boldsymbol{Q}_{\text{dec}}$.

Structure of the encoders

A single encoder from the multi-scale autoencoder consisted of a series of N_{layers} layers with a fixed filter kernel size ($N_k \times N_k$) but different number of filter kernels D_f [55, 56, 57], where each encoder had the same general architecture. Furthermore the vector $\eta_{\text{enc}} \in \mathbb{R}^{N_{\text{layers}}}$ contained the number of filters D_f for each layer.

The encoder was composed of two parts. The first part consisted of $N_{\text{layers}} - 1$ layers with the same architecture. In the first step of each of these layers a padding was applied around the image, where the dimensions of the padding depended on the filter kernel size of that encoder. Then, a convolutional neural network with $D_f = \eta_{\text{enc},i}$ filter kernels and a stride equal to 2 was used, where $\eta_{\text{enc},i}$ was the i-th element of η_{enc} and $i = 1...N_{\text{layers}}$. As a result of the stride, the dimensions of the output image was halved compared to the input image for the CNN in that layer. Then, padding was again applied around the output of the first CNN and a second convolutional neural network was used with $s = 1$. As a result of the second CNN, the dimensionality was not reduced and only the encoding performance was increased [57]. This (first) part of the encoder architecture was repeated $N_{\text{Layers}} - 1$ times.

Now, in the second part of the encoder a padding was applied, after which a CNN with $s = 2$ was used to reduce dimensionality to the latent space size. Furthermore, the number of filter $\eta_{\text{enc},N_{\text{layers}}}$ was equal to the desired number of feature maps in the latent space. The schematic representation to construct the encoders is also depicted in the flowchart of Fig. 5.2A.

Structure of the decoders

Similar to the encoder, the general architecture of the three parallel decoders were the same and only their filter kernel sizes were different. In addition, the general architecture consisted of N_{layers} layers subdivided in two parts (Fig. 5.2B) [57], where the number of filters D_f per layer was represented in $\eta_{\text{dec}} \in \mathbb{R}^{N_{\text{layers}}}$. .

Initially, before the first of the layer, a padding was applied around the latent space image, where the dimensions of the padding depended on the filter kernel size [57]. After that, $N_{\text{layers}} - 1$ layers were present consisting of first a transpose CNN with a stride equal to

two and $\eta_{\text{dec},i}$ filter kernels, where $i = 1...N_{\text{layers}}$. As a result, the size of an output image of the transpose CNN was twice as large as that of an input image. Then, a CNN was present with a stride of 1 to further increase the performance of the decoder without altering the dimensionality [57]. Finally, after the repeated $N_{\text{layers}} - 1$ layers, the second component consisted of a single transpose convolutional neural network with $s = 2$ and $\eta_{\text{dec},N_{\text{layers}}}$ filter kernels, followed by a centercrop layer and a CNN with a stride of 1 and 2 filter kernels- (both velocity components). Here, the centercrop layers removed the boundaries of the images and keeps only the centre of the image to obtain an image size equal to the input, in contrast to padding that adds along the boundaries of the image [57].

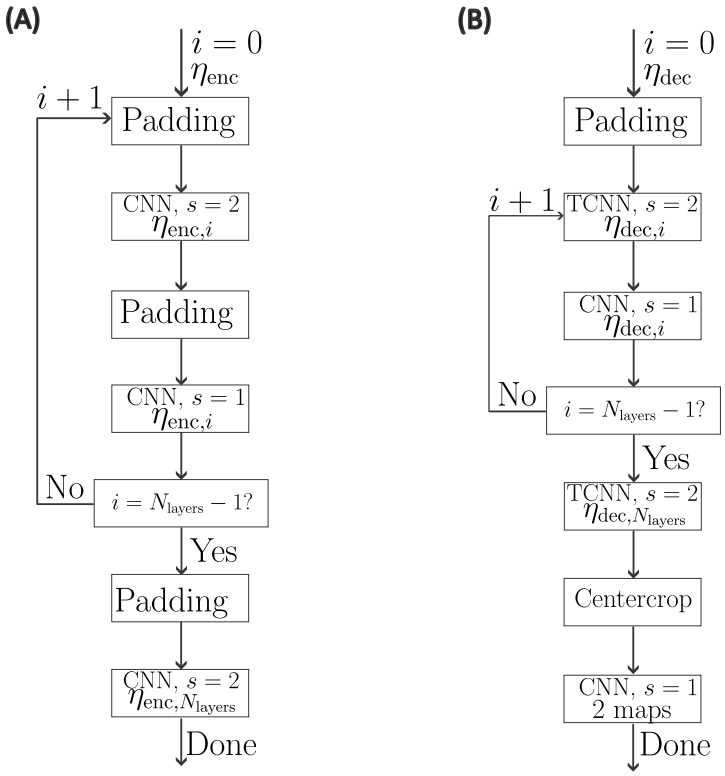


Figure 5.2: Flowcharts showing schematically the steps of constructing the encoder (A) and decoder (B) of the multi-scale autoencoder. Architecture based on the work of Racca et al. [57].

Multi-scale autoencoder settings

A lower bound for the latent space dimensions of the autoencoder for a specific test case was provided through the Kaplan-Yorke dimension of the dynamical system. To summarize, the Kaplan-Yorke dimension D_{KY} was 3.24 and 9.5 for the $Re = 20$ and $Re = 34$ test cases respectively. The dimensions of a single feature map in latent space for $Re = 20$ was a 4×4 matrix (16 degrees of freedom) and for $Re = 34$ an 8×8 matrix (64 degrees of freedom). The latent space dimensions of a single feature map was chosen such that they were much larger than the Kaplan-Yorke dimensions of the corresponding test case and a sufficiently low reconstruction error was obtained. In order to study the effect of latent space dimensions on the performance of the Hybrid-A architecture, three multi-scale autoencoders with a different number of latent space feature maps per Reynolds number were constructed (Tab. 5.1). Here, for each encoder η_{enc} and decoder η_{dec}, the number of feature maps in every layer is provided.

Table 5.1: The number of feature maps in the convolutional neural network layers for each autoencoder with changing latent space size for 2D Kolmogorov flow at $Re = 20$ and $Re = 34$. For each vector η_{enc}, the last entry represented the number of filter maps in latent space.

Re	η_{enc}	η_{dec}	N_{lat}
20	[24, 1],[24, 3],[24, 5]	[24, 12]	[16, 48, 80]
34	[12, 24, 1],[12, 24, 3],[12, 24, 5]	[24, 12, 6, 3]	[64, 192, 320]

Multi-scale autoencoder training

The presented multi-scale autoencoders were trained for 2001 epochs for both Reynolds numbers, where the initial weights of the (transpose) convolutional neural networks were prescribed following Glorot and Bengio [83]. For $Re = 20$, the training data set consisted of 12,500 snapshots from the flow with a time interval of $\delta t = 0.5$ between each snapshot. Furthermore, the autoencoders were validated on a set of 2,500 snapshots with the same time interval as the training data. For $Re = 34$, similar to Racca et al. [57], the autoencoders were trained on 25,000 snapshots and validated on 5,000 snapshots with a time interval of $\delta t = 1.0$. The training and validation data were split into batches containing 50 samples. The training occurred by minimizing the mean square error (MSE)

$$\text{MSE} = \left\langle (\boldsymbol{Q} - \boldsymbol{Q}_{\text{dec}})^2 \right\rangle \tag{5.3}$$

between the reference and the decode solution through AMSgrad stochastic gradient descent [57, 84, 85]. The final training losses L_{train} and validation losses L_{val} for each test case and latent space are presented in Tab. 5.2.

Table 5.2: Table showing the the training loss L_{train} and the validation loss L_{val} of the trained autoencoders for 2D Kolmogorov flow at $Re = 20$ and $Re = 34$ for each latent space dimension.

Re	N_{lat}	L_{train}	L_{val}
20	[16, 48, 80]	$[7.1, 2.2, 1.3]\cdot 10^{-5}$	$[7.5, 2.3, 1.4]\cdot 10^{-5}$
34	[64, 192, 320]	$[9.0, 2.4, 1.1]\cdot 10^{-5}$	$[11.7, 2.8, 1.2]\cdot 10^{-5}$

5.3 Data-driven reduced order modelling

Aside from the knowledge-based approach of section 5.1, which depend on the formulation of governing equations of dynamical systems, recent advances in machine learning can be utilized to construct pure deep learning reduced order models based on recurrent neural networks such as echo state networks (section 3.2) and multi-scale autoencoders (section 5.2) to handle high dimensional data [57].

As mentioned in section 3.2.1, the plain ESN is not able to handle high dimensional data efficiently due to requirement of very large reservoir sizes as well as the memory cost as a result of Tikhonov regularisation during training. Therefore, to allow for a pure data driven reduced order model, the echo state network can be trained to perform prediction on

the latent space obtained by the multi-scale autoencoder rather than the chaotic system in physical space [57]. Thus, for the data-driven reduced order model in this work, the echo state network was combined with a multi-scale autoencoder as proposed by Racca et al. [57]. The main idea was that the initial conditions in physical space were encoded into their latent space representation. Then, the trained ESN performed temporal predictions on the dynamical system in latent space format [57]. Finally, after the latent space predictions were performed, each predicted time step was decompressed back into physical space [57]. In this work, both the multi-scale autoencoder and the echo state network were implemented using the TensorFlow library [86].

5.4 Hybrid-A architecture

The aim of this Master thesis was to design a hybrid knowledge-based/deep-learning reduced order model that enabled efficient and time-accurate short-term and long-term statistical prediction for high dimensional chaotic systems. In this work, it was presented that knowledge-based models can be constructed through proper orthogonal decomposition combined with Galerkin projection (section 5.1). Furthermore, following the work by Racca et al. [57], a deep learning reduced order model was obtained through combining a recurrent neural network (ESN) with a convolutional autoencoder (section 5.3). In order to construct the Hybrid-A model, these two architectures were combined following the approach of Pathak et al. [61] and Lesjak and Doan [7]. However, the hybrid model by Pathak et al. [61] and later adapted by Lesjak and Doan [7] was only applicable to low-dimensional chaotic systems. Thus, to enable the Hybrid-A model proposed in this research to efficiently work on high dimensional chaotic system, the multi-scale autoencoder as discussed in section 5.2 was implemented. The schematic overview of the proposed Hybrid-A model is presented in Fig. 5.3. Here, the model is shown in prediction model and similar to the plain echo state network, training mode is obtained by removing the feedback loops [7].

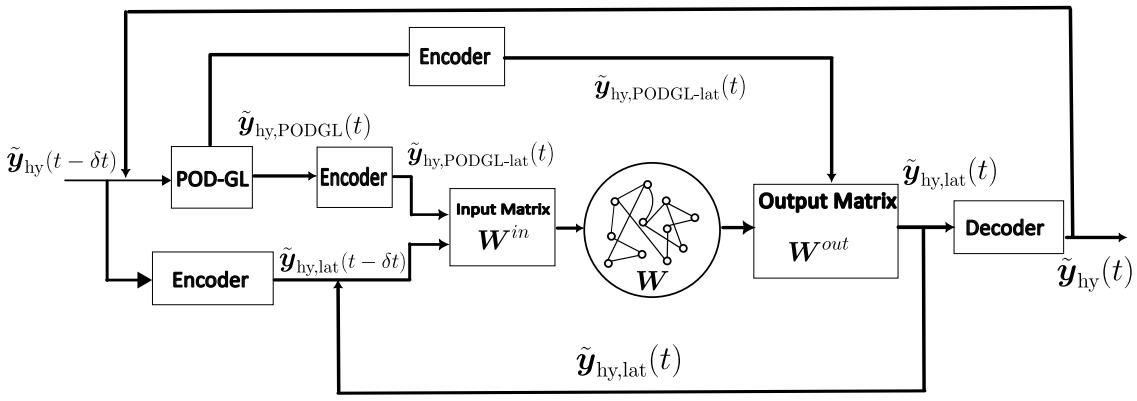


Figure 5.3: Proposed Hybrid-A model architecture allowing for temporal predictions on high dimensional chaotic system.

The true data of the system (following section 2.1 and section 5.1) was written as the one dimensional vector $\boldsymbol{y}(\boldsymbol{x},t) \in \mathbb{R}^{N_{\text{phys}}}$, where $N_{\text{phys}} = 2 \cdot N_x \cdot N_y$, with the first $N_x \cdot N_y$ elements

corresponding to u and the second $N_x \cdot N_y$ corresponding to v. Furthermore, the prediction of the Hybrid-A model was written as $\tilde{\boldsymbol{y}}_{\text{hy}}(t) \in \mathbb{R}^{N_{\text{phys}}}$.

The first step of the proposed Hybrid-A model was to obtain the input $\tilde{\boldsymbol{y}}_{\text{hy}}(t-\delta t) \in \mathbb{R}^{N_{\text{phys}}}$. Then, the knowledge-based model (POD-GL, proper orthogonal decomposition Galerkin) performed a single prediction in time to obtained the POD-GL solution $\tilde{\boldsymbol{y}}_{\text{hy,PODGL}}(t) \in \mathbb{R}^{N_{\text{phys}}}$ and this prediction was encoded into $\tilde{\boldsymbol{y}}_{\text{hy,PODGL-lat}}(t) \in \mathbb{R}^{N_{\text{lat}}}$. Simultaneously, the input $\tilde{\boldsymbol{y}}_{\text{hy}}(t-\delta t)$ was also encoded into the latent space format $\tilde{\boldsymbol{y}}_{\text{hy, lat}}(t-\delta t) \in \mathbb{R}^{N_{\text{lat}}}$. Then, both the encoded knowledge-based prediction and the encoded input data were concatenated into the input data vector of size $2N_{\text{lat}}$, such that [7, 61]

$$\tilde{\boldsymbol{y}}_{\text{hy, in}} = \begin{bmatrix} \tilde{\boldsymbol{y}}_{\text{hy,PODGL-lat}}(t) \\ \tilde{\boldsymbol{y}}_{\text{hy, lat}}(t-\delta t) \end{bmatrix} \in \mathbb{R}^{2N_{\text{lat}}}, \tag{5.4}$$

which was fed as input data into the echo state network. Furthermore, $\tilde{\boldsymbol{y}}_{\text{hy,PODGL-lat}}(t)$ was also fed into the output matrix of the echo state network. Doing so, the ESN was enabled to make a mix between its own prediction and the one from the knowledge-based model. Then, the echo state network performed a single prediction in the latent space by updating its internal state vector $\boldsymbol{x}_{\text{res}}$. The newly updated internal reservoir state was then concatenated with the knowledge-based prediction to obtain

$$\tilde{\boldsymbol{y}}_{\text{hy, out}}(t) = \begin{bmatrix} \tilde{\boldsymbol{y}}_{\text{hy,PODGL-lat}}(t) \\ \boldsymbol{x}_{\text{res}}(t) \end{bmatrix} \in \mathbb{R}^{N_{\text{lat}}+N_{\text{res}}}. \tag{5.5}$$

Using the trained output matrix $\boldsymbol{W}^{\text{out}} \in \mathbb{R}^{N_{\text{lat}} \times N_{\text{lat}}+N_{\text{res}}}$ the prediction $\tilde{\boldsymbol{y}}_{\text{hy,lat}}(t) \in \mathbb{R}^{N_{\text{lat}}}$ was obtained through the matrix-vector multiplication [7, 61]

$$\tilde{\boldsymbol{y}}_{\text{hy,lat}}(t) = \boldsymbol{W}^{\text{out}} \tilde{\boldsymbol{y}}_{\text{hy, out}} \tag{5.6}$$

Finally, the predicted flow field $\tilde{\boldsymbol{y}}_{\text{hy}}(t) \in \mathbb{R}^{N_{\text{phys}}}$ was obtained by decoding $\tilde{\boldsymbol{y}}_{\text{hy,lat}}(t)$. Now, in prediction mode, the Hybrid-A model was able to run autonomously such that it re-used its output as input for the next prediction. For the data-driven component, similar to Racca et al. [57], the latent space prediction was re-used directly without the need for decoding since the echo state networks operated in latent space. However, since the full physical system was required for the POD-GL reduced order model, the decoded prediction was also looped back to serve as input data for the next time step.

In the proposed Hybrid-A model, the encoder used to encode the data from the knowledge-based component was the same as for the data-driven component. Furthermore, for the POD-GL model (similar to section 5.1) the number of POD modes was truncated to obtain computational efficiency while reflecting dominant features in the dynamical system as will be discussed in section 5.6.

5.5 Comparison criteria

In order to assess the performance of the hybrid architectures and allow for comparison with the knowledge-based and pure deep learning models, several criteria were used. The first benchmark was the short-term prediction horizon of the models. Secondly, the long-term flow statistics were considered.

Short-term prediction horizon
In order to provide insight into the short-term prediction horizon, the error $\epsilon_{hor}(t)$ as a function of time was considered

$$\epsilon_{hor}(t) = \frac{|\boldsymbol{u}_{\text{rom}} - \boldsymbol{u}_{\text{ref}}|}{\sqrt{\langle |\boldsymbol{u}|^2 \rangle}}. \tag{5.7}$$

Here, $\epsilon_{hor}(t)$ represented the error between a given reduced order model $\boldsymbol{u}_{\text{rom}}$ and the true data (subscript ref for reference) $\boldsymbol{u}_{\text{ref}}$ obtained from solving the equations with direct numerical simulations. Then, the short-term prediction horizon t_{pred} was defined as the time that the error was below a prescribed threshold ϵ_{thres} [7]. In this work, the threshold error was fixed to $\epsilon_{thres} = 0.4$ for all experiments.

Long-term flow statistics
Since the two-dimensional Kolmogorov flow was a fluid dynamical system also the capability of the models to represent long-term statistics was of interest to gain insights into the long-term behaviour of the model. In this work, the first statistic considered was the temporal mean of the flow $\langle \boldsymbol{u} \rangle$. In order to assess the performance of the reduced order models, the root mean square error was considered:

$$\text{RMSE}_u = \sqrt{\langle (\langle u_{\text{rom}} \rangle - \langle u_{\text{ref}} \rangle)^2 \rangle} \quad \text{and} \quad \text{RMSE}_v = \sqrt{\langle (\langle v_{\text{rom}} \rangle - \langle v_{\text{ref}} \rangle)^2 \rangle}. \tag{5.8}$$

Secondly, due to turbulent flow, also the preservation of the the mean velocity fluctuations $\langle \boldsymbol{u}' \rangle$ is of interest. Similar to the mean velocity, the performances was assessed through the root mean squared error using Eq. 5.8, where $\langle \boldsymbol{u}_{\text{ref}} \rangle$ was swapped for $\langle \boldsymbol{u}'_{\text{ref}} \rangle$ and $\langle \boldsymbol{u}_{\text{rom}} \rangle$ was replaced for $\langle \boldsymbol{u}'_{\text{rom}} \rangle$

Thirdly, the phase space spanned by the domain-averaged kinetic energy $E(t)$ (see Eq. 4.3) and the domain-averaged dissipation $D(t)$ was of interest to gain insight into whether the models were able stay in the confined domain of the true solution or would migrate to a different region in phase space (or to a single fixed point). Here, both quantities were computed in physical space using [57] (in Einstein summation convention)

$$E(t) = \frac{1}{(2\pi)^2} \int_0^{2\pi} \int_0^{2\pi} \frac{1}{2} u_i u_i dx dy \tag{5.9}$$

$$D(t) = \frac{Re^{-1}}{(2\pi)^2} \int_0^{2\pi} \int_0^{2\pi} \frac{\partial u_i}{\partial x_j} \frac{\partial u_i}{\partial x_j} dx dy. \tag{5.10}$$

Finally, albeit strictly not a comparison criteria but important for further analysis, the contribution of the ESN and the knowledge-based components in the Hybrid-A model was considered [7]. As mentioned in section. 5.4, the encoded knowledge-based prediction of the input was fed into the input and output matrix of the echo state network. This way, a blending of the ESN and the knowledge-based prediction was obtained. Therefore, in order to delve deeper into the understanding of the proposed Hybrid-A model, it was of interest to gain insight into the (relative) contribution of both the pure deep learning and the knowledge-based components.

These contributions were obtained through the output matrix $\boldsymbol{W}^{\text{out}}$ [7]. It was observed in Eq. 5.6 that the output matrix operated on a concatenated vector of the knowledge-based model prediction with the updated internal state of the reservoir. Since this operation only entailed a matrix-vector multiplication (Eq. 5.6), each row from the output matrix was multiplied element-wise with the output vector $\tilde{\boldsymbol{y}}_{\text{hy, out}}$. Thus, the sum over all rows for each column of $\boldsymbol{W}^{\text{out}}$ provided an indication of the contribution of the corresponding element of $\tilde{\boldsymbol{y}}_{\text{hy, out}}$, which was either from the knowledge-based or deep learning component [7]. In this work, similar to Lesjak and Doan [7], the mean normalized contribution Γ_j were considered using

$$\Gamma_j = \frac{1}{N_{\text{lat}}} \sum_{i=1}^{N_{\text{lat}}} \frac{\kappa_{ij}^2}{\sum_k \kappa_{ik}^2}, \tag{5.11}$$

where κ_{ij} are the elements from the output matrix. Particular interest was in the total average normalized contribution of the knowledge-based Γ_{PODGL} and the deep learning model Γ_{ESN} defined as [7]

$$\Gamma_{\text{PODGL}} = \frac{\sum_{i=1}^{N_{\text{lat}}} \Gamma_i}{\sum_{i=1}^{N_{\text{lat}}+N_{\text{res}}} \Gamma_i} \quad \Gamma_{\text{ESN}} = \frac{\sum_{i=N_{\text{lat}}+1}^{N_{\text{lat}}+N_{\text{res}}} \Gamma_i}{\sum_{i=1}^{N_{\text{lat}}+N_{\text{res}}} \Gamma_i}. \tag{5.12}$$

5.6 Truncation of POD modes

As aforementioned in section 5.1 and section 5.4, in order to obtain the knowledge-based reduced order model and the POD-GL component of the hybrid models, the number of POD modes must be truncated. In this work, the choice of the number of POD modes was based on two factors. Firstly, since POD is a data reduction technique similar to autoencoders, it was decided that the number of POD modes must be approximately equal to the latent space size N_{lat} to allow for a good comparison between the reduced order models. Secondly, the chosen number of modes must have adequate performances in terms of the short-term prediction horizon of the knowledge-based reduced order model (section 5.5). In the proceeding of this section, the selection of the POD modes for both Reynolds numbers is further described.

***Re* = 20**

For the two-dimensional Kolmogorov flow $Re = 20$ test case, three different multi-scale autoencoders were used with an increasing latent space size to allow for increasing (de)compression performance. In this work, the latent space sizes were $N_{\text{lat}} = [16, 48, 80]$ (Tab. 5.1). Thus, following the first criterion for the modal reduction, the number of POD modes was chosen to be approximately equal to each N_{lat}. Therefore, for each multi-scale autoencoder, a different number of POD modes were selected in the knowledge-based and hybrid reduced order models. Secondly, the short-term prediction horizon was considered for POD modes in the vicinity of the number of latent space dimensions N_{lat} as shown in Fig. 5.4. Then, the short-term prediction horizons were computed for all considered modes and this was performed for 20 runs per mode using different initial conditions. From these runs, the mean and the standard deviation were obtained as shown in Fig. 5.4.

Starting with the lowest latent space $N_{\text{lat}} = 16$, it was observed in the top left image of Fig. 5.4 that from 12 to 21 POD modes the short-term prediction horizon increased. However, for increasing POD modes also the standard deviation increased significantly, indicating that the model became more dependent on initial conditions. Therefore, it was chosen to use 18 retained POD modes for the knowledge-based and hybrid reduced order models as it shows locally a good performance. Next, for the second multi-scale autoencoder with a latent space of $N_{\text{lat}} = 48$ (top right of Fig. 5.4) it was observed that at 51 retained modes the performance of the knowledge-based model was the best. Furthermore, an improved in performance compared to 18 POD modes was observed. Finally, for the largest latent space of $N_{\text{lat}} = 80$ the mean short-term prediction horizon increased significantly from 77 modes till 84 modes, after which it decreased again. For this reason, 84 modes were chosen for the knowledge-based and hybrid models corresponding to the third multi-scale autoencoder.

Thus, in this work, the POD modes equal to 18, 51 and 84 were selected for $N_{\text{lat}} = [16, 48, 80]$. Now, as discussed in section 2.1, the eigenvalues of the POD modes provided a measure of the amount of energy that a given mode represented in the system [65, 16]. In the left image of Fig. 5.6, the relative contribution of each mode to the total energy is depicted. It was observed that the amount of energy that a POD mode represented decreased fast with increasing modes. Therefore, only a relative small portion of modes were required to still encapsulate most of the energy in the system. In addition, the sum of all the eigenvalues provided the total amount of kinetic energy in the system (because the mean was not substracted for the POD).

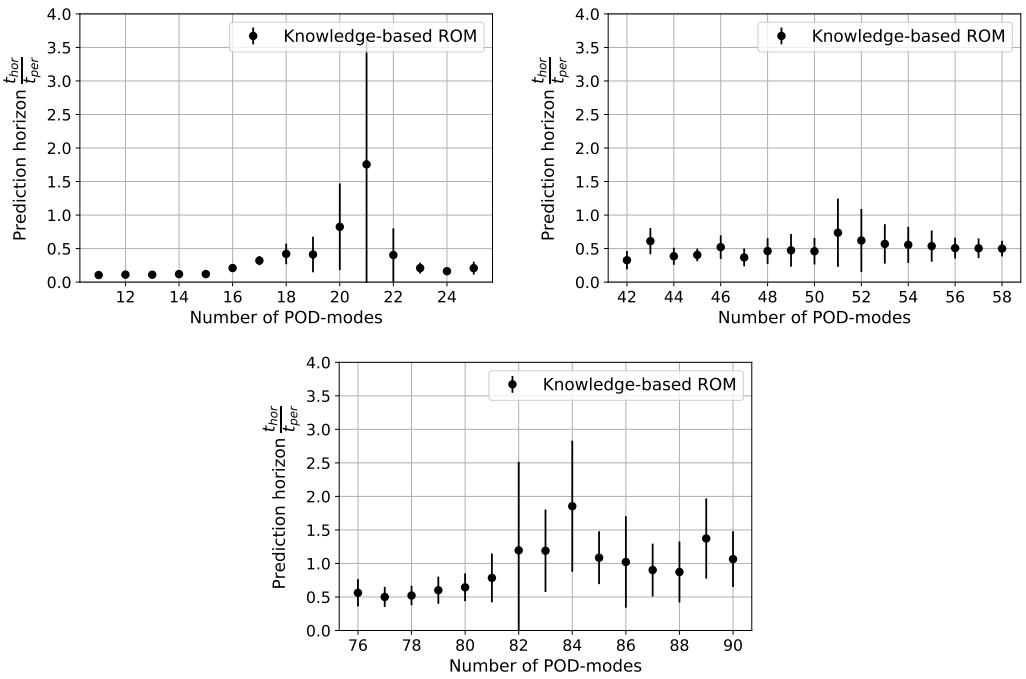


Figure 5.4: Images showing the mean short-term prediction horizon t_{pred} and standard deviation of 20 simulations as a function of the number of POD modes of the knowledge-based reduced order model, for three regions near the vicinity of the latent space sizes [16,48,80] of the multi-scale autoencoders for $Re = 20$. Here, $t_{\text{per}} = 170$.

Therefore, the percentage energy of the retained number of POD modes E_{ret} compared to the full systems energy was computed through

$$E_{\text{ret}} = \frac{\sum_{i=1}^{N_{\text{ret}}} \lambda_{\text{pod},i}}{\sum_{j=1}^{N_{\text{pod}}} \lambda_{\text{pod},j}} \cdot 100\%. \tag{5.13}$$

Here, λ_{pod} were the eigenvalues of the diagonal of Λ_{pod} in order of decreasing eigenvalue and N_{ret} are the number of retained modes. Using Eq. 5.13, it was obtained that retaining 18, 51 and 84 POD modes encapsulated $E_{\text{ret}} = 99.08\%$, 99.91% and 99.98% of the energy in the system.

$Re = 34$

The latent space sizes for $Re = 34$ were $N_{\text{lat}} = [64, 192, 320]$ (section 5.2). The plots for the prediction horizon as a function of the number of POD modes in the vicinity of the latent space size are depicted in Fig. 5.5, where the prediction time was scaled with the Lyapunov time of the system.

For the smallest latent space $N_{\text{lat}} = 64$, it was observed that the best performance was found for 68 retained POD modes. For the large two latent space size ($N_{\text{lat}} = [192, 320]$), it was chosen to set the number of retained POD modes to be much smaller than the latent space

size, because a too large number of modes would be too computationally inefficient. Still, to reflect the increase in latent space size, an amount of POD modes was chosen that showed significantly better performance compared to the 68 modes. In the right graph of Fig. 5.5, the mean prediction horizon over an interval of POD modes is depicted near 90 retained modes. Here, due to the good performance 90 retained modes were selected for $N_{\text{lat}} = [192, 320]$.

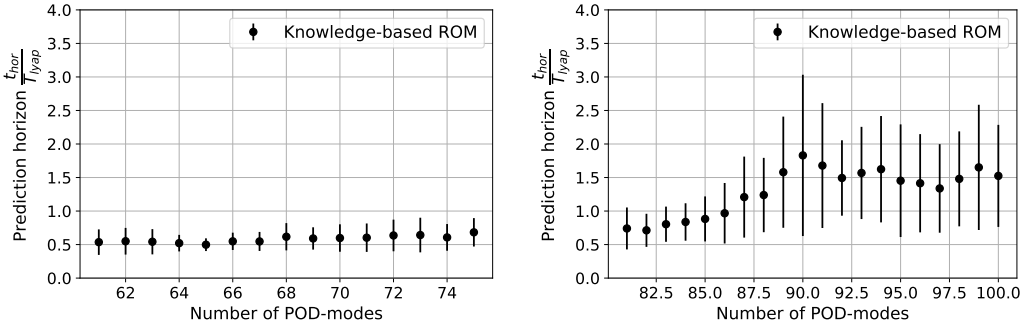


Figure 5.5: Images showing the mean short-term prediction horizon t_{pred} and standard deviation of 20 simulations as a function of the number of POD modes of the knowledge-based reduced order model, for three regions near the vicinity of the latent space sizes [64, 192, 320] of the multi-scale autoencoders for $Re = 34$.

Furthermore, the relative contribution of each of POD modes to the total amount of energy in the dynamical system is presented in the right graph of Fig. 5.6. Similar to $Re = 20$, it was observed the first few most contributed the most to the total amount of energy in the system. As a result, the encapsulated energy with respect to the full system was 99.4% and 99.8% for the 68 and 90 retained POD modes.

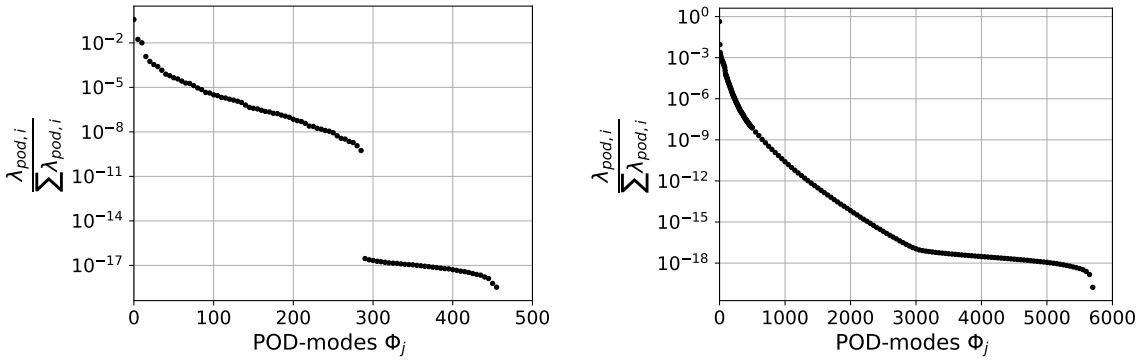


Figure 5.6: Plots showing the relative energy contribution of each POD mode for both test case.

5.7 Model settings

In the previous sections, the knowledge-based, deep learning and the proposed hybrid reduced order models were presented. Furthermore, the comparison criteria for the performance of the models and the truncation of the number of POD modes were outlined. In this section, the model settings for validation/hyperparameter tuning and the actual experiments are described. First, the tuning of the hyperparameters (also referred to as validation [78]) is discussed. Next, the model settings for the predictions are outlined.

The comparison of the performance in terms of short-term horizon and long-term statistics of the three models was of interest as a function of the number of neurons in the reservoir and the latent space dimensions of the multi-scale autoencoder. For this reason, it was chosen to keep the number of POD modes fixed for a given autoencoder (see section 5.6) and only vary the number of neurons in the ESN reservoir. Here, the reservoir size of [500, 1000, 1500, 2000, 3000, 4000, 5000] were considered.

5.7.1 Hyperparameter tuning and validation settings

As detailed in section 3.2.1, hyperparameter tuning was required to obtain good performance from the echo state network [69]. In this work, Bayesian optimisation (Appendix A) combined with a grid search was used to provide a guided tuning methodology as the evaluation of the hybrid models was expensive. The Bayesian optimisation (Scikit-Learn implementation [87]) was used for tuning the spectral radius ρ, input scaling σ_{in} and the leaking rate α, while grid search was used for the Tikhonov factor β. Furthermore, the sparsity $\langle d \rangle = 3.5$ was fixed. During the tuning process, the inverse of the short-term prediction horizon was minimized (and thus the short-term prediction horizon was maximized). In this section, first the general routine of the grid search-Bayesian optimisation is provided, after which validation strategies for $Re = 20$ (single shot validation) and $Re = 34$ (recycle validation) [78] are discussed together with the optimisation settings for each test case (Tab. 5.3). Furthermore, the obtained hyperparameters are presented in Appendix C for all experiments.

Grid Search-Bayesian optimisation routine
In the first step of the hyperparameter tuning, the search space for the spectral radius $[\rho^l, \rho^u]$, input scaling $[\sigma_{\text{in}}^l, \sigma_{\text{in}}^u]$ and leaking rate $[\alpha^l, \alpha^u]$ were defined, where the super script l and u indicate lower and upper bound. Furthermore, the list of Tikhonov factors $\boldsymbol{\beta}$ was defined over which the grid search was performed [57]. Then, N_{rdm} points in the search space spanned by $[\rho, \sigma_{\text{in}}, \alpha]$ were selected randomly. For each search space point, the mean inverse of the short-term prediction horizon were computed over N_{ic} initial conditions (IC) for each Tikhonov factor from $\boldsymbol{\beta} = \left[1.0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\right]$.

After the initial random search through the search space, N_{bo} points were evaluated through Bayesian optimisation with an ARD 5/2 Matern Kernel for the covariance matrix in the Gaussian process and the grid search over $\boldsymbol{\beta}$. Furthermore, Expected Improvement acquisition function was used. Then, the optimal hyperparameters were selected by maximizing the mean

short-term prediction horizon over 2 initial conditions. Here, only 2 initial conditions were chosen due to the large computational cost.

The generic procedure described above was used for both test. However, a profound difference between the $Re = 20$ and the $Re = 34$ test case was the validation strategies used for the optimisation, namely single shot validation ($Re = 20$) and recycle validation ($Re = 34$) [57, 78].

Single shot validation ($Re = 20$)
In single shot validation for the $Re = 20$ test case, the echo state network of both the pure deep learning and hybrid models were trained from $t = 0$ to $t = t_{\text{train}}$ using N_{train} snapshots and a washout of N_{wash} snapshots equispaced by δt. Furthermore, the proper orthogonal decomposition was based on N_{podtrain} snapshots. The washout was used during training to allow the ESN to update its internal state over time without storing those predictions such that the ESN was initialized using the echo state property [69]. The tuning of the hyperparameters occurred for 2 initial conditions with 100 time steps in between, such that both snapshots were within the same periodic cycle of the flow. For the validation part, the reduced order model performed predictions over t_{val} time steps, corresponding to N_{val} snapshots (equispaced by δt) and the short-term prediction horizon was determined for every run. Now, in the single shot validation method, the validation data set was chosen such that there exists no overlap with the training data set [78]. Thus, the validation data set was obtained for every run from $t_{\text{train}} < t < t_{\text{val}}$. Furthermore, similar to training, N_{resync} resync snapshots were chosen before the initial conditions to allow the echo state network to initialize similar to the washout [69]. The single shot validation settings for $Re = 20$ are provided in Tab. 5.3.

Recycle validation ($Re = 34$)
For the second test case at a Reynolds number of 34, the hyperparameter tuning during validation was obtained through recycle validation [78]. The primary difference between single shot validation and recycle validation was the validation data set. As mentioned in the previous part, in single shot validation the validation data set was located directly after the training set. In recycle validation, however, the validation sets were small portions of the training data set and thus the echo state network was trained and validated on the same data [78]. This was possible because training the ESN occurred in a different configuration as prediction, since no feedback loop was not present when training [78]. Thus, additional knowledge from the system was extracted by both training and validating on the same data set [78]. It was shown by Racca et al. [78] that for chaotic systems the recycle validation was able to outperform other validation approaches.

In this work, the echo state network for $Re = 34$ was trained for $t = t_{\text{train}}$, corresponding to N_{train} snapshots equispaced by δt. Furthermore, the proper orthogonal decomposition was based on N_{podtrain} snapshots. Similar to the single shot validation, during training, N_{wash} snapshots were omitted as washout for the initialisation of the internal state of the reservoir. For the validation, two initial conditions were selected and temporal predictions were obtained for t_{val} time steps (N_{val} snapshots), from which the mean short-term prediction horizon was computed through the grid search-Bayesian optimisation approach. Thus, in recycle validation the training and validation data sets overlapped. Therefore, the selected initial conditions were chosen such that the full validations sets fitted within the training set

[78]. The validation settings for $Re = 34$ are shown in Tab. 5.3

Parameter	Setting ($Re = 20$)	Parameter	Setting ($Re = 34$)
$[\rho^l, \rho^u]$	[0.1, 5.0]	$[\rho^l, \rho^u]$	[0.01, 5.0]
$[\sigma^l_{\text{in}}, \sigma^u_{\text{in}}]$	[0.1, 5.0]	$[\sigma^l_{\text{in}}, \sigma^u_{\text{in}}]$	[0.01, 5.0]
$[\alpha^l, \alpha^u]$	[0.1, 1.0]	$[\alpha^l, \alpha^u]$	[0.01, 1.0]
N_{rdm}	7	N_{rdm}	7
N_{bo}	7	N_{bo}	7
t_{train}	500	t_{train}	$500T_{\text{Lyap}}$
N_{train}	1000	N_{train}	74626
N_{PODtrain}	92000	N_{PODtrain}	90000
N_{wash}	100	N_{wash}	100
δt	0.5	δt	0.1
t_{val}	10.000	t_{val}	$20T_{\text{Lyap}}$
N_{val}	20.000	N_{val}	2985
N_{resync}	40	N_{resync}	2985 ($20T_{\text{Lyap}}$)

Table 5.3: The models settings of the validation parameters for the tuning of the hyperparameters for the $Re = 20$ (left) and $Re = 34$ (right) test case.

5.7.2 Model settings experiments

In the previous section, the values for the model parameters for the tuning of the hyperparameters in this research were presented. After obtaining the tuned hyperparameters, the model experiments were performed to compute the short-term prediction horizon and long-term statistics. In this section, these parameter values are presented. For most of the parameters, no distinction was made between the short-term predictions and the long-term statistics. Therefore, these are discussed simultaneously and it will be mentioned when difference were applied.

For both test cases, similar to the grid search-Bayesian optimisation [57, 78] of the previous section, the echo state network was trained on N_{train} snapshots, corresponding to t_{train} time steps where the snapshots were sampled with time interval δt. In addition, the proper orthogonal decomposition was based on N_{podtrain} snapshots. Furthermore, during training a washout of N_{wash} was used. Then, after training the echo state network, the temporal prediction were performed. Since the short-term prediction horizon and the long-term statistics were dependent on the initial condition, 20 simulations were performed for each number of neurons in the reservoir to obtain a mean and standard deviation of the quantities of interest. First, the echo state networks were initialized using N_{resync} snapshots. Then, the temporal predictions for the experiments were performed over N_{pred} integer time steps, corresponding to N_{pred} predictions in total. Here, for the long-term statistics, the prediction time was chosen such that it is much larger than the short-term prediction horizon of each simulation. Finally, the short-term prediction horizon and the long-term statistics were computed from the predictions of the reduced order models.

Parameter	Setting ($Re = 20$)
t_{train}	500
N_{train}	1000
N_{PODtrain}	92000
N_{wash}	100
δt	0.5
t_{pred}	10000 (ST) / 20000 (LT)
N_{pred}	20000 (ST) / 40000 (LT)
N_{resync}	40

Parameter	Setting ($Re = 34$)
t_{train}	$500T_{\text{Lyap}}$
N_{train}	74626
N_{PODtrain}	90000
N_{wash}	100
δt	0.1
t_{pred}	$20T_{\text{Lyap}}$ (ST) / $40T_{\text{Lyap}}$ (LT)
N_{pred}	2985 (ST) / 5970 (LT)
N_{resync}	2985 ($20T_{\text{Lyap}}$)

Table 5.4: The models settings of the temporal predictions parameters for the tuning of the hyperparameters for the $Re = 20$ (left) and $Re = 34$ (right) test case. Here, ST and LT are abbreviations for short term and long term respectively.

Finally, the $Re = 20$ experiments were performed on a workstation with a 14-core Intel(R) Xeon(R) W-2275 CPU @ 3.30Ghz with a Nvidia Quadro RTX 8000 GPU. Furthermore, for the $Re = 34$ a 20-core Intel(R) Xeon(R) w7-2475X and a Nvidia RTX A4500 GPU workstation was used.

Chapter 6

Hybrid A: Results and discussion

As stated in the research questions, the primary aim of this work was to construct a hybrid knowledge-based/deep learning reduced order model that allowed for time-accurate short-term and long-term statistical predictions on high dimensional chaotic systems. In this chapter, the results of the proposed Hybrid-A model for the two test cases will be presented. First, in section 6.1, the results on the short-term prediction horizon will be shown and discussed, after which the long-term statistics are presented in section 6.2. At the end of each section, also a brief summary of the results is provided. As outlined in section 5.5, the performance of the long-term statistics were assessed using the temporal mean of the velocity ($\langle u \rangle$ and $\langle v \rangle$), temporal mean of the velocity fluctuations ($\langle u' \rangle$ and $\langle v' \rangle$) and the trajectories in the phase space spanned by the domain averaged kinetic energy $E(t)$ and the domain averaged dissipation $D(t)$. Note however that since $Re = 20$ was not turbulent, the mean of the velocity fluctuations were not considered for this test case. For the discussion, the results of this study will be placed in perspective compared to the works of Racca et al. [57] and Lesjak and Doan [7] whom proposed a deep learning reduced order model for high dimensional chaotic systems and a hybrid model without autoencoders for low dimensional chaotic systems respectively. Finally, the limitations of the Hybrid-A model of this research will be discussed in section 6.3.

6.1 Short-term prediction horizon

In this section, the results on the short-term prediction horizon (as defined in section 5.5) are shown for the knowledge-based, deep learning and Hybrid-A reduced order models. First, the results for the $Re = 20$ test case are presented, followed by the $Re = 34$ chaotic Kolmogorov flow.

$Re = 20$

The short-term predictions horizons for the knowledge-based, deep learning and Hybrid-A reduced order model for the two-dimensional Kolmogorov flow at $Re = 20$ are presented in Fig. 6.1. Here, the top left plot corresponds to the multi-scale autoencoder with $N_{\text{lat}} = 16$, the top right to $N_{\text{lat}} = 48$ and the bottom plot to $N_{\text{lat}} = 80$.

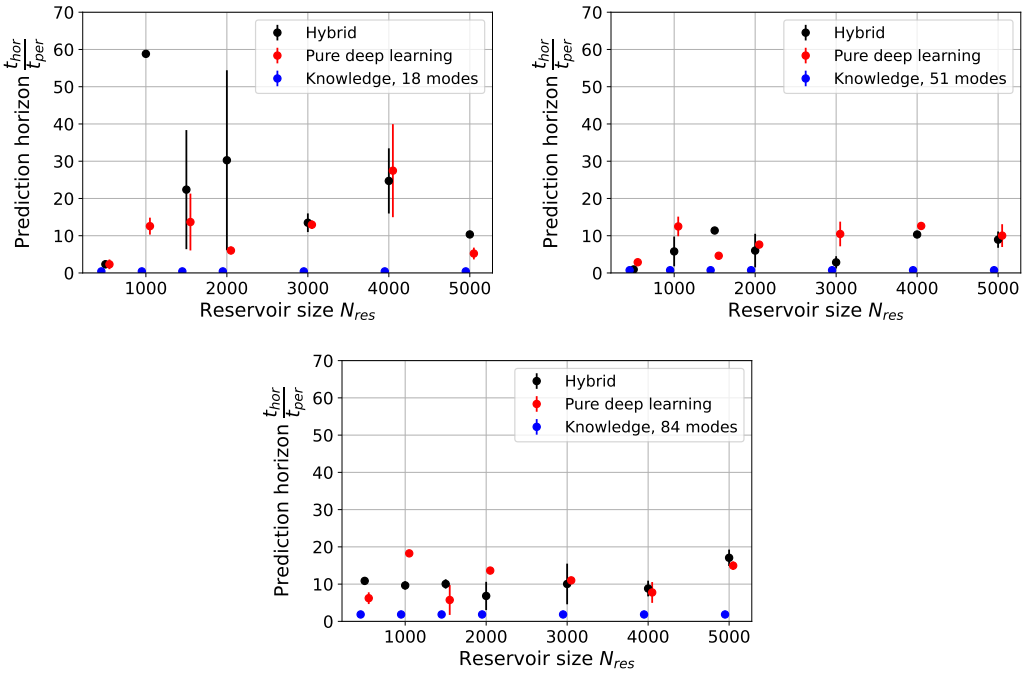


Figure 6.1: The short-term prediction horizon of the knowledge-based, deep learning and Hybrid-A reduced order models as a function of the number of neurons in the reservoir at $Re = 20$ Kolmogorov flow. Here, the top left plot is for the multi-scale autoencoder with $N_{\text{lat}} = 16$, the top right for $N_{\text{lat}} = 48$ and the bottom plot for $N_{\text{lat}} = 80$. The knowledge-based and deep learning results are displaced slightly in the horizontal direction for clarity.

For the knowledge-based model, the performance slightly improved with increasing number of POD modes as a result of the encapsulation of more dynamics from the system. However, despite using 84 modes and hence representing over 99% of the systems energy, the model was only able to predict around two periodic cycles in time. This was likely due to the introduction of instabilities in the model through the truncation of the number of modes. In contrast to the knowledge-based model, the deep learning and the Hybrid-A reduced order models displayed significant better performance for all three latent spaces and an average prediction horizon of at least 5 periodic cycles was obtainable for almost all experiments. Moreover, for many experiments much larger prediction horizons were observed.

In order to compare the performance between the knowledge-based and the deep learning/Hybrid-A models, it was important to consider that the mean square error introduced by the multi-scale autoencoders was of the order of 10^{-5} (Tab. 5.2). Since $Re = 20$ Kolmogorov flow was periodic in physical space, the latent space representation was also approximately periodic due to the small (de)compression errors. As a result, the dynamics in latent space were still simple and hence easy to learn by the echo state network. This thus ex-

plained the overall better performance of the deep learning and Hybrid-A models compared to the knowledge-based model. This was supported by the fact that no increase in performance of the deep learning and Hybrid-A models was observed for larger reservoir sizes, indicating that the echo state network was saturated. The latter might also have been an effect of too little training data (three periodic cycles) used for the echo state network [7], but that was not expected as still good performance was observed.

For the comparison of the performance of the deep learning and the Hybrid-A reduced order models for all latent sizes, it was observed that typically a larger performance difference was present for small reservoirs while for larger reservoirs the prediction horizons were comparable (Fig. 6.1). It was also shown that increasing the latent space dimensions from $N_{\text{lat}} = 48$ to $N_{\text{lat}} = 80$ that no consistent improvement was observed for both the deep learning and the Hybrid-A model. This was likely the result of the simple dynamics and hence larger latent spaces were not able to represent more dynamics. Surprisingly, for the Hybrid-A and deep learning models at $N_{\text{lat}} = 16$, in general, a much larger prediction horizon was observed compared to $N_{\text{lat}} = 48$ and $N_{\text{lat}} = 80$ for $N_{\text{res}} = [1000, 1500, 2000]$. More specifically, for $N_{\text{res}} = 1000$, the prediction even exceeded the model prediction setting of 10,000 integer time steps. However, it was unclear what caused this behaviour for $N_{\text{lat}} = 16$.

To further assess the Hybrid-A model, using Eq. 5.12, the total contributions of the deep learning and knowledge-based components were computed for each experiment as shown in Fig. 6.2. For the smallest and middle latent space, the relative contribution of the deep learning component (ESN) increased as a function of the reservoir size and thus consequently the knowledge-based part decreased. This also explained that for $N_{\text{lat}} = [16, 48]$ the short-term prediction horizon of the deep learning and Hybrid-A became more similar for larger reservoirs as the contribution of the ESN was more profound. On the other hand, for the largest latent space, initially an increase in deep learning contribution was observed, after which it started to fluctuate between a small and a large contribution. Still, as shown in Fig. 6.1, the performance became more similar for larger reservoirs. It was not clear what caused this behaviour for $N_{\text{lat}} = 80$.

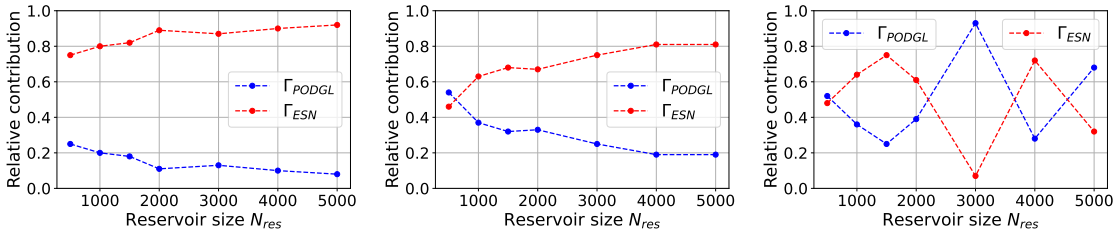


Figure 6.2: Plots showing the total relative contributions of the knowledge-based Γ_{PODGL} and deep learning Γ_{ESN} components in the Hybrid-A architecture as a function of reservoir size for $Re = 20$ Kolmogorov flow. The smallest latent space $N_{\text{lat}} = 16$ is depicted on the left, $N_{\text{lat}} = 48$ in the middle and $N_{\text{lat}} = 80$ on the right.

To continue, consider the proposed Hybrid-A architecture (section 5.4) and compare it to the knowledge-based and deep learning models. A profound difference between the three models was that the knowledge-based model operated in physical space, while the deep learning model performed predictions in latent space and each predicted latent space was decoded at the end

of the time series. Thus, during the iterations over time no compression/decompression of the data was required for both models. Now, for Hybrid-A architecture, (de)compression occurred in each iteration since the two components operated on different representations of the system. Hence, in each iteration, a small error was introduced into the Hybrid-A solution. Since no consistent performance difference was observed between the deep learning and the hybrid-A models, this indicated that the (de)compression error accumulation was not of importance for the $Re = 20$ test case. This was hypothesized to be the result of the periodic nature of the flow and thus allowing the Hybrid-A model to restore from the slight perturbations added in each iteration over time and the knowledge-based component compensating for the accumulation of errors from the autoencoder.

$Re = 34$

The second test case was conducted with Kolmogorov flow at $Re = 34$, which was in the turbulent (chaotic) regime [57, 73]. The results for the short-term prediction horizon for $Re = 34$ are displayed in Fig. 6.3. Similar to $Re = 20$, the top left plot is the smallest latent space $N_{\text{lat}} = 64$, the top right plot is $N_{\text{lat}} = 192$ and finally the bottom plot for $N_{\text{lat}} = 320$.

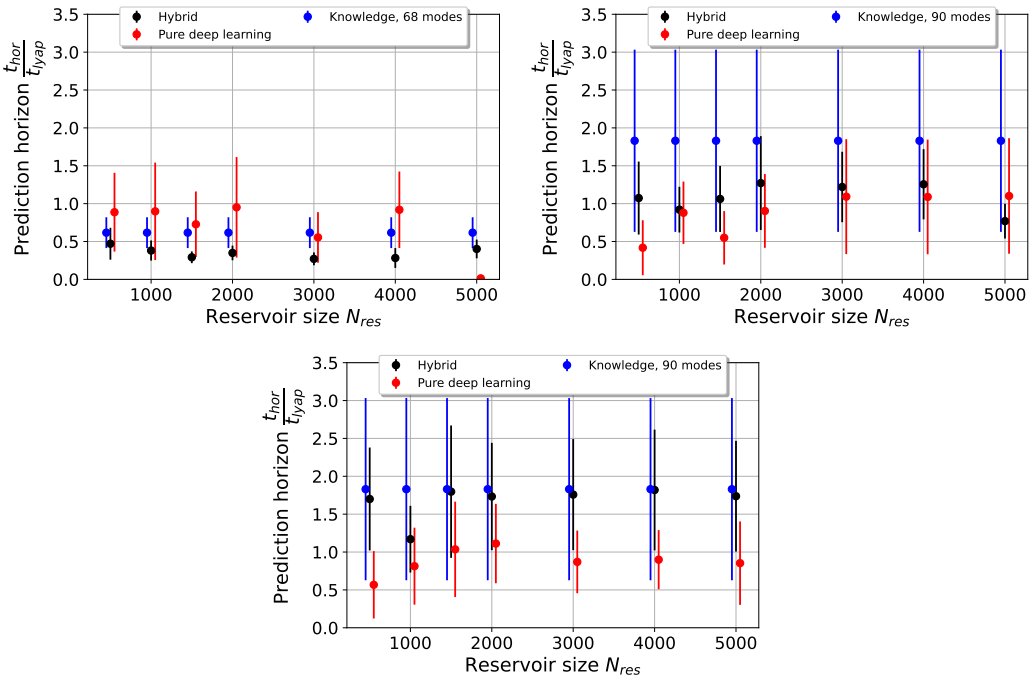


Figure 6.3: The short-term prediction horizon of the knowledge-based, deep learning and Hybrid-A reduced order models as a function of the number of neurons in the reservoir at $Re = 34$ Kolmogorov flow. Here, the top left plot is for the multi-scale autoencoder with $N_{\text{lat}} = 64$, the top right for $N_{\text{lat}} = 192$ and the bottom plot for $N_{\text{lat}} = 320$. The knowledge-based and deep learning results are displaced slightly in the horizontal direction for clarity.

Firstly, the models were able to predict time-accurately for a shorter amount of time and with a larger dependence on initial conditions compared to the periodic case, which was a result of the chaotic nature of the $Re = 34$ test case. For $N_{\text{lat}} = 64$, all the models were on average not able to sufficiently predict the flow for longer than one Lyapunov time, where the Lyapunov time provided a characteristic time scale of the chaos in the system. Here, the pure

deep learning model showed the best performance for all reservoirs sizes except $N_{\text{res}} = 5000$ (it was unknown why the performance of the deep learning model at this reservoir size was deteriorated). Then, the knowledge-based model showed the second best performance, while the hybrid model was the least accurate. Furthermore, for the hybrid and deep learning models no performance increase with increasing reservoir size was observed. This was thought to be caused by the small latent space size and thus a too small portion of the dynamics was represented. Hence, the ESN did not benefit from an increase in reservoir size to learn more latent space dynamics.

Upon increasing the latent space size to $N_{\text{lat}} = 192$ and thus retaining 90 modes in the knowledge-based model (top right plot of Fig. 6.3), the performance of the knowledge-based model increased significantly and was able to on average outperform the deep learning and Hybrid-A reduced order models. This performance increase was the result of (similar to $Re = 20$) the encapsulation of more dynamics with more POD modes. For the deep learning model no consistent performance improvement was observed. On the other hand, the Hybrid-A model showed significantly better performance compared to the smallest latent space. Thus, given no consistent improvement of the deep learning model, this was due to the better performing knowledge-based component and a larger latent space that reflected more dynamics. Furthermore, opposed to $N_{\text{lat}} = 64$, the deep learning reduced order model showed saturation behaviour for $N_{\text{lat}} = 192$ as the prediction horizon increased from $N_{\text{res}} = 500$ to $N_{\text{res}} = 2000$ after which stagnation/saturation occurred. Similar saturation was also observed for the Hybrid-A model but to a lesser extent.

Finally, further increasing the latent space size to $N_{\text{lat}} = 320$ showed that the predictive capabilities of the deep learning model did overall not improve for increasing latent space size. Similar behaviour was found by Racca et al. [57], where a changing latent space did not consistently influence the short-term prediction horizon. Furthermore, similar to $N_{\text{lat}} = 192$, an asymptotic short-term prediction horizon as a function of the reservoir size was observed for the deep learning model. This was potentially caused by a limited training data set and thus the model was not able to learn more dynamics for larger reservoirs [7]. In contrast to the deep learning model, a significant increase in predictive capabilities was observed for the Hybrid-A reduced order model as the short-term prediction horizon almost matched the knowledge-based model. For the Hybrid-A model no saturation was observed compared to the deep learning model.

Similar to the $Re = 20$ test case, the total relative contribution of the knowledge-based and deep learning components of the Hybrid-A model were computed as a function of the reservoir size for each multi-scale autoencoder as depicted in Fig. 6.4. For the smallest latent space, it was observed for $N_{\text{res}} = 500$ that the knowledge-based component had a slightly larger contribution likely the result of the small reservoir used. For larger echo state networks, the contribution of the deep learning component gradually became more important. For $N_{\text{lat}} = 192$, the increasing reservoir resulted in a larger contribution of the echo state network, such that for $N_{\text{res}} = [3000, 4000]$ the knowledge-based part was almost completely overruled by the deep learning component, which was surprising as the knowledge-based model displayed better performance. Finally, for the largest latent space, the opposite was true where the deep learning component had almost no contribution and thus the Hybrid-A model was effectively similar to the knowledge-based model with the inclusion of the encoders and decoders. As

a consequence, as shown in Fig. 6.3, the knowledge-based and Hybrid-A models had nearly similar performance.

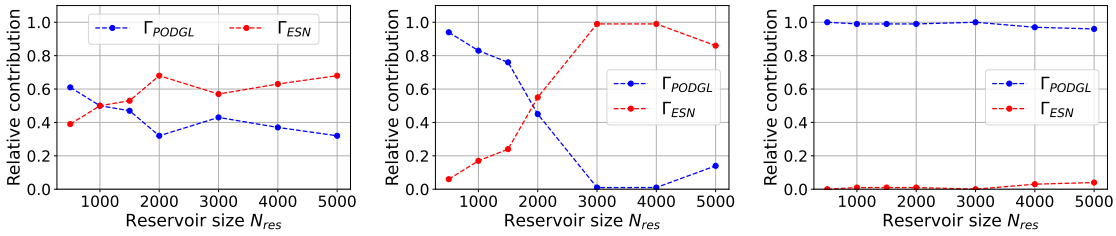


Figure 6.4: Plots showing the total relative contributions of the knowledge-based Γ_{PODGL} and deep learning Γ_{ESN} components in the Hybrid-A architecture as a function of reservoir size for $Re = 34$ Kolmogorov flow. The smallest latent space $N_{\mathrm{lat}} = 64$ is depicted on the left, $N_{\mathrm{lat}} = 192$ in the middle and $N_{\mathrm{lat}} = 320$ on the right.

Now, the Hybrid-A model was not able to outperform both the knowledge-based and the deep learning reduced order models for all latent space sizes. For the smallest latent space, the Hybrid-A model was outperformed by both the knowledge-based and the deep learning model. Given that Fig. 6.4 showed that both components contributed in the Hybrid-A architecture for a significant amount, the reduced performance of Hybrid-A was interpreted to be the result of the accumulation of errors by the compression and decompression of the data in the multi-scale autoencoder in each iteration. In addition, since $Re = 34$ Kolmogorov flow was chaotic and thus highly dependent on the initial conditions, each introduced error by the (de)compression could have resulted in the knowledge-based component to push away the Hybrid-A solution further away the true solution, from which the model could not recover (as opposed to $Re = 20$).

As the latent space increased, the accumulation of errors appeared to become less significant. For $N_{\mathrm{lat}} = 192$ the short-term performance of Hybrid-A became more similar to the performance of the deep learning model, coinciding with the findings of Fig. 6.4 that the ESN became more influential for larger reservoirs. Since Hybrid-A was able to outperform the deep learning model for most reservoirs, this indicated that despite the little contribution of the knowledge-based component for $N_{\mathrm{lat}} = [3000, 4000]$, it still served as a meaningful addition. However, it was difficult to assess the influence of the errors accumulation by the autoencoder against the benefits of the small portion of the knowledge-based model. Finally for the largest latent space, for all reservoirs except $N_{\mathrm{res}} = 1000$, the Hybrid-A model slightly under performed compared to the knowledge model. This coincided with Fig. 6.4 that displayed for all reservoirs that the knowledge-based component of the Hybrid-A model was dominant, from which it followed that the influence of the error accumulation by the autoencoder was small but present. Now, the dominance of the knowledge-based component for larger latent spaces was hypothesized to be the effect of the independence of the deep learning model on increasing latent space combined with more knowledge-based dynamics being captured in larger latent spaces. Therefore, the knowledge-based component gained in relative contribution in the Hybrid-A model.

Thus, that the Hybrid-A model was not able to outperform the knowledge-based and deep learning ROMs for all experiments. This was in contrast to the results by Lesjak and Doan

[7] for low dimensional chaotic systems with a similar hybrid model architecture, but without the multi-scale autoencoders. It was shown by them that their hybrid model was able to outperform the knowledge-based and the deep learning reduced order models. Furthermore, a good performing knowledge-based model combined with a small deep learning reservoir already significantly improved the results of the hybrid architecture. Evidently from Fig. 6.3 such behaviour was not observed using the Hybrid-A model. This thus indicated that the inclusion of the autoencoder to allow the hybrid model to efficiently handle high dimensional data was not beneficial in terms of the short-term prediction horizon. This was likely due to the fact that to much information was lost in the latent space representation, while the pure knowledge-based model operated on the physical system combined with the deep learning performance not increasing for larger latent spaces.

Summary of short-term prediction horizon
For the performance in terms of short-term prediction horizon for the periodic test case, it was found that the Hybrid-A reduced order model was able to time-accurately predict the flow for at least 5 periodic times for most experiments. Furthermore, in terms of the influence of the latent space on the predictive capabilities of the Hybrid-A model, for an latent space increase from $N_{\text{lat}} = 16$ to $N_{\text{lat}} = 48$ a significant influence was observed, however the cause was unknown. For larger latent space from $N_{\text{lat}} = 48$ to $N_{\text{lat}} = 80$ no influence was found as was expected due to the simple dynamics in latent space, which were easy to learn by the ESN. For the same reason, the short-term prediction horizon did not show a general increase for larger reservoirs. In terms of the comparative performance, the Hybrid-A model was able to significantly outperform the knowledge-based model. This was hypothesized to be the result of the simple dynamics in latent space, while instabilities occurred in the pure knowledge-based model. It was shown for the deep learning model that for larger reservoirs the results were similar compared to the Hybrid-A, resulting from the increasing ESN contribution with reservoir size for the two smaller latent spaces. However, for $N_{\text{lat}} = 80$, such behaviour was also observed, but no clear relation with the ESN contribution was found. Finally, since the performance of the Hybrid-A model did not under perform compared to the knowledge-based/deep learning ROMs, the (de)compression error accumulation by the multi-scale autoencoder was not of significance for all latent spaces, which was hypothesized to be the result of the periodic nature of the flow allowing the model to restore itself from slight perturbations and the included knowledge-based component was able to compensate for these errors.

Increasing the Reynolds number to $Re = 34$, such that chaotic behaviour was observed in the system resulted in the Hybrid-A model to be able to time-accurately predict the Kolmogorov flow for about 0.5 to close to 1.8 Lyapunov time depending on the latent space and reservoir size. Furthermore, a profound influence of the latent space dimensions of the autoencoders was observed for the Hybrid-A model. For the smallest latent space, the low Hybrid-A performance was interpreted to be the result of the accumulation of (de)compression errors by the autoencoders in each iteration. Then, increasing the latent space to $N_{\text{lat}} = 192$ (and thus 90 POD modes), showed a significant increase in Hybrid-A performance likely the result of the better performing knowledge-based component as little influence of latent space dimensions on the pure deep learning model was found. Similarly, as the latent space increased from $N_{\text{lat}} = 192$ to $N_{\text{lat}} = 320$ while the number of POD modes remained the same, also resulted in a significant increase in Hybrid-A model performance. This was explained by

the increase in knowledge-based contribution for all reservoirs effectively making the Hybrid-A model similar to the knowledge-based model with the inclusion of the autoencoder. This was a result of a better representation of the knowledge-based component predictions combined with no consistent performance increase of the deep learning model as a function of latent space. Now, in terms of the comparative performance between the Hybrid-A and the knowledge-based/deep learning ROMs, for all latent space sizes the Hybrid-A model was not able to outperform both the knowledge-based and the deep learning models. This was interpreted to be the result of the latent space representation of the physical system in which too much information was lost to benefit from both components. Furthermore, for smaller latent spaces, the accumulation of (de)compression errors in each iteration of the Hybrid-A model was expected to be of importance, while for larger latent space a smaller influence due to lower reconstruction errors was interpreted.

6.2 Long-term flow statistics

In the previous section of this chapter, the performance of the knowledge-based, deep learning and Hybrid-A reduced order models were discussed in terms of the short-term prediction horizon. As two-dimensional Kolmogorov flow was used to test the models on, the long-term statistics were also of interest. As described in section 5.5, the focus of this section will be on the mean velocities ($\langle u \rangle$ and $\langle v \rangle$), mean velocity fluctuations ($\langle u' \rangle$ and $\langle v' \rangle$) and the phase space described by the domain averaged kinetic energy $E(t)$ and the domain averaged dissipation $D(t)$ for both test cases. In particular, for the mean velocities and fluctuations, the root mean square error between the reduced order models and the reference solution was considered. In the proceeding of the section, first the statistical results for $Re = 20$, after which $Re = 34$ are presented.

***Re* = 20**

In Fig. 6.5, the results for the root means square error for the $\langle u \rangle$ (top row) and $\langle v \rangle$ (bottom row) between the reduced order models and the reference solutions are presented for the three latent space sizes. Here the right plot represent the smallest latent space, the middle plot corresponds to $N_{\text{lat}} = 48$ and the right plot depicts the largest latent space. In these plots, the data point for the $N_{\text{lat}} = 80$ and $N_{\text{res}} = 500$ experiment was removed for readability as it was much larger than the rest of the errors. The figures including this data point are presented in Appendix D. However, the origin of this significantly larger error was unclear. Here, the statistics were computed over 20.000 integer time steps (section 5.7.2), corresponding to roughly 143 periodic cycles. Furthermore, the mean of the velocity fluctuations were not considered as this flow was not turbulent/chaotic.

Starting with the knowledge-based model, for $\langle u \rangle$ the performance at 18 was slightly better compared to 84 POD modes, while a lesser performance was obtained for 51 retained POD modes. This was in contrast with the short-term prediction horizon, where a performance increase was observed with the inclusion of more modes. Similar to $\langle u \rangle$, the knowledge-based model with 51 retained POD modes showed the least good performance for $\langle v \rangle$. The lesser performance for 51 retained modes might have been a result of an instability present in the

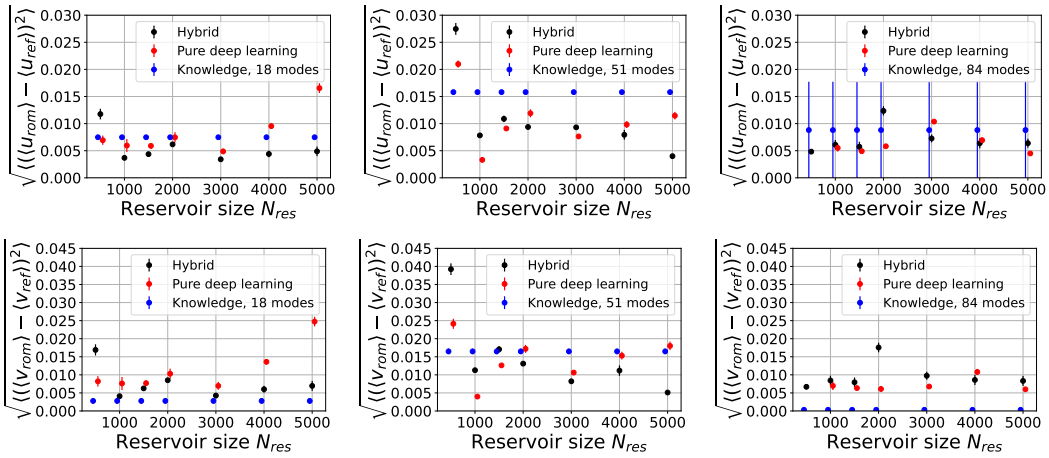


Figure 6.5: The root mean square error of $\langle u \rangle$ (top row) and $\langle v \rangle$ (bottom row) of the flow between the reduced order models and the reference solution for the knowledge-based, deep learning and Hybrid-A reduced order models as a function of the number of neurons in the reservoir at $Re = 20$. Here, the top left plot is for the multi-scale autoencoder with $N_{\text{lat}} = 16$, the top right for $N_{\text{lat}} = 48$ and the bottom plot for $N_{\text{lat}} = 80$. The knowledge-based and deep learning results are displaced slightly in the horizontal direction for clarity.

model that developed over longer time scales and thus did not affect the short-term prediction horizon. Furthermore, the performance in $\langle v \rangle$ for 84 POD modes was better compared to 18 retained modes. Finally, the results of $\langle u \rangle$ at $N_{\text{lat}} = 80$ a large standard deviation was observed, which was not present for all other experiments.

Continuing with the deep learning and the Hybrid-A architectures, for $\langle u \rangle$ both models were able to outperform the knowledge-based model for most experiments, coinciding with the short-term observations of Fig. 6.1. However, in general, the differences between the performances of the three models was less pronounced compared to the short-term prediction horizon. Here, the better comparative performance of the mean flow from the knowledge-based was hypothesized to be the result of the fact that in the proper orthogonal decomposition of this research, the averaged flow was not subtracted and thus the first mode fully represented the mean. Therefore, the mean flow was well represented by the knowledge-based model. Now, for $N_{\text{lat}} = 16$, the Hybrid-A model was able to outperform the deep learning model in terms of $\langle u \rangle$ for most reservoirs. However, for the middle and largest latent space, no clear pattern was observed as for some reservoir sizes the Hybrid-A model performed better while for others the opposite was true. Finally for $N_{\text{lat}} = 16$ and $N_{\text{lat}} = 51$, in contrast to the short-term horizon results, the deep learning statistics of $\langle u \rangle$ tended to diverge from the Hybrid-A solution for increasing reservoir sizes even though the relative contribution of the ESN in the Hybrid-A increased for larger reservoirs (Fig. 6.2).

Similar to the $\langle u \rangle$, the experiments showed for $\langle v \rangle$ that for $N_{\text{lat}} = 16$ the Hybrid-A model performed better than deep learning model for most experiments, however both models were outperformed by the 18 modes knowledge-based model. Furthermore, for larger reservoirs, similar to $\langle u \rangle$, the Hybrid-A model diverged from the deep learning solution. Increasing the latent space size to $N_{\text{lat}} = 48$, also a better performance by the Hybrid-A model was observed

for reservoirs equal and larger than 2000 neurons and both models performed in general better than the knowledge-based model, similar to $\langle u \rangle$. However, a similar divergence of Hybrid-A and deep learning results is depicted. Upon considering the largest latent space $N_{\text{lat}} = 80$, the knowledge-based model outperformed both the deep learning and the Hybrid-A models. Furthermore, the deep learning model showed better performance than Hybrid-A.

Thus, the above described results on the long-term statistics of the $\langle u \rangle$ and $\langle v \rangle$ mean flows showed that no clear pattern was present in the comparative performance between the Hybrid-A, knowledge-based and deep learning reduced order models. This also followed from the relative contributions shown in Fig, 6.2, where an increase in contribution by the ESN was observed for larger reservoirs, while for both $\langle u \rangle$ and $\langle v \rangle$ at $N_{\text{lat}} = [16, 48]$ the Hybrid-A and deep learning models showed greater differences for larger reservoirs. Similar to the short-term prediction horizon, no influence of the latent space dimensions on the performance of the Hybrid-A model was found, which might be the result of the simple dynamics. For $\langle v \rangle$ at $N_{\text{lat}} = [16, 80]$, the lesser performance of the Hybrid-A model was also unclear. A likely reason for the lack of clear pattern in comparative performance could have been the design of the hyperparameter tuning. As shown in section 5.7.1, the validation procedure was performed by maximizing the short-term prediction horizon. Hence, the deep learning and Hybrid-A models were not optimized to perform predictions over longer times. Furthermore, the periodic nature of the flow could have been the origin as the deep learning and Hybrid-A model were able to sufficiently represent the dynamics already for small reservoir. As a result, an increase in latent space and/or reservoir size was not beneficial, similar to the short-term prediction horizon.

Next, the performance of the three models in terms of the domain averaged kinetic energy $E(t)$ and dissipation $D(t)$ in the system was studied. For this analysis, only the reservoir sizes compromised of 500 and 2000 neurons are presented in Fig. 6.6 and Fig. 6.7. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 16$, second row $N_{\text{lat}} = 48$ and the third row $N_{\text{lat}} = 80$. Furthermore, the model settings are also repeated in the title of each plot and the red dot represents $t = 0$. In addition, the results for $N_{\text{res}} = [1000, 1500, 3000, 4000, 5000]$ are depicted in Appendix E. For the knowledge-based model, 18 retained POD modes was not sufficient to accurately reflect the kinetic energy and dissipation of the system. Still, the solution was confined to the same domain as the reference solution and increasing the number of modes showed better performance. For the deep learning model, the predictions on the kinetic energy and the dissipation of also remained in the same domain as the reference solution. However only for the largest latent space with 500 neurons in the reservoir a divergence towards a different region in phase space was present. Typically, for the smallest latent space, the deep learning model performed better than the knowledge-based model, while the opposite was true for the two larger latent spaces. Furthermore, for the Hybrid-A model, the kinetic energy and the dissipation remained confined to a domain similar to one observed for the reference solution for most model settings. In addition, an increase in the number of POD modes and latent space had a positive effect on the representation of the trajectories compared to the reference for some experiments, while a decrease in performance was also observed for reservoirs of size 1500, 4000 and 5000 neurons. In addition, the increase in the number of neurons in the reservoirs appeared to be beneficial as well but to a lesser extent.

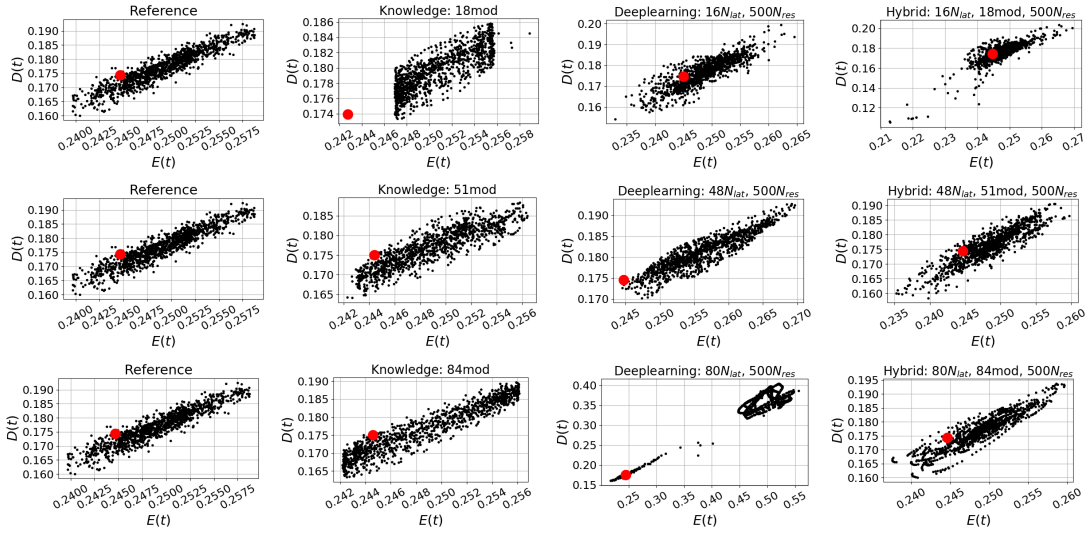


Figure 6.6: Plots showing the phase space trajectories of the dissipation $D(t)$ and the kinetic energy $E(t)$ for a reservoir of 500 neurons for the $Re = 20$ test case. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 16$, second row $N_{\text{lat}} = 48$ and the third row $N_{\text{lat}} = 80$ and the red dot represents $t = 0$.

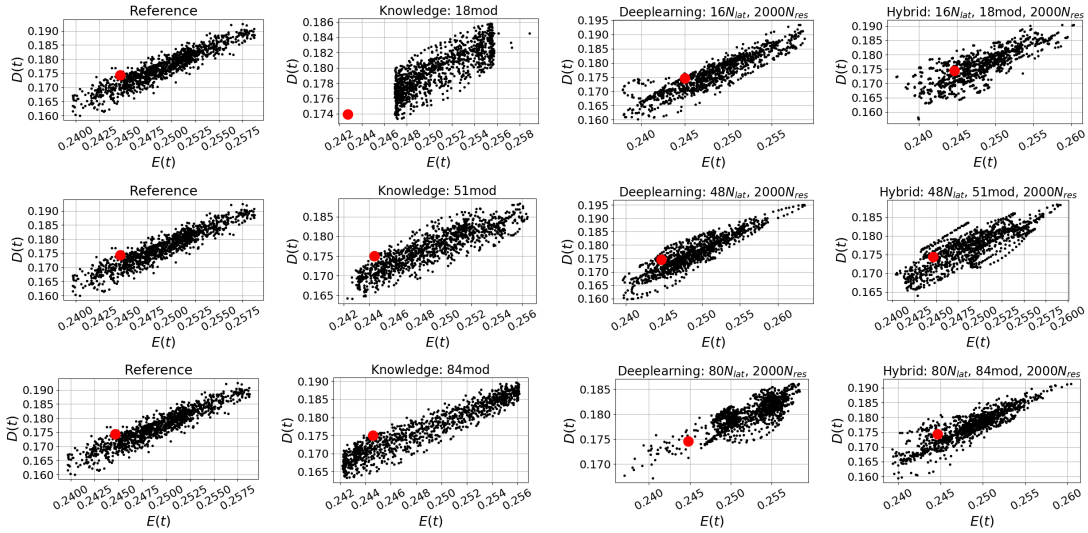


Figure 6.7: Plots showing the phase space trajectories of the dissipation $D(t)$ and the kinetic energy $E(t)$ for a reservoir of 2000 neurons for the $Re = 20$ test case. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 16$, second row $N_{\text{lat}} = 48$ and the third row $N_{\text{lat}} = 80$ and the red dot represents $t = 0$.

$Re = 34$

Similar to $Re = 20$, the root mean square error between the reduced order model and the reference solution for the long-term statistics were obtained as a function of the number of neurons in the reservoir. For $Re = 34$, the root mean square errors of $\langle u \rangle$ (top) and $\langle v \rangle$ (bottom) are depicted in Fig. 6.8. Here, similar to the previous figures on $Re = 20$, the left plot represents the smallest latent space $N_{\text{lat}} = 64$, the middle plot $N_{\text{lat}} = 192$ and the right plot corresponds to the largest latent space $N_{\text{lat}} = 320$. Furthermore, as detailed in section 5.7.2 the statistics were computed over 40 Lyapunov times, which was significantly longer than the average short-term prediction horizon.

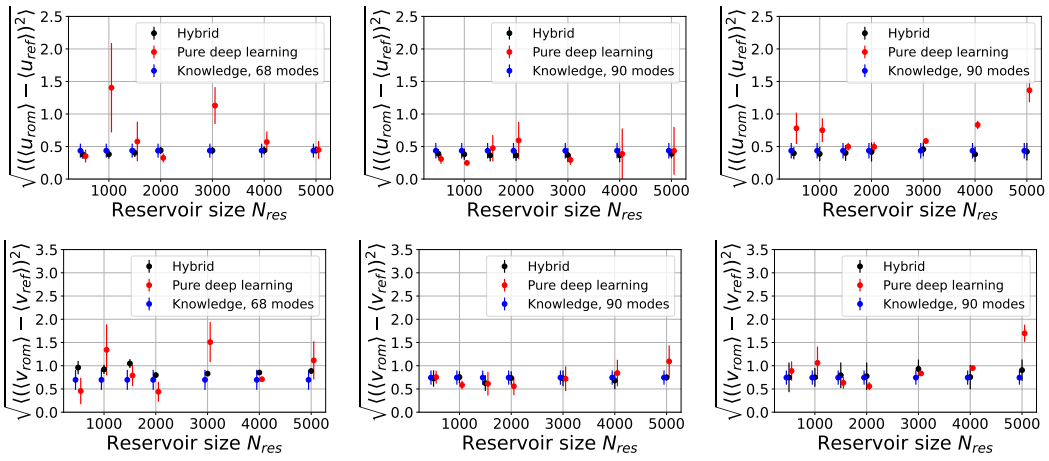


Figure 6.8: The root mean square error of $\langle u \rangle$ and $\langle v \rangle$ between the reduced order models and the reference solution for the knowledge-based, deep learning and Hybrid-A reduced order models as a function of the number of neurons in the reservoir at $Re = 34$. Here, the top left plot is for the multi-scale autoencoder with $N_{\text{lat}} = 64$, the top right for $N_{\text{lat}} = 192$ and the bottom plot for $N_{\text{lat}} = 320$. The knowledge-based and deep learning results are displaced slightly in the horizontal direction for clarity.

In Fig. 6.8 the errors for the $Re = 34$ were significantly larger than for the periodic test case for both $\langle u \rangle$ and $\langle v \rangle$ as a result of the chaotic nature of this test case. Furthermore, compared to short-term prediction horizon, the relative standard deviations were overall smaller for $\langle u \rangle$ and $\langle v \rangle$, indicating that for these statistics the initial condition had less influence. For the knowledge-based model, an increase in the number of modes did not influence the prediction errors (as opposed to the short-term prediction horizon), which was likely an effect of the fact that the first POD mode already fully represented the mean flow. Furthermore, slightly larger errors were found for $\langle v \rangle$ compared to $\langle u \rangle$.

For the deep learning reduced order model at the smallest latent space size $N_{\text{lat}} = 64$ for both $\langle u \rangle$ and $\langle v \rangle$, a large variation in performance as a function of the reservoir size was present compared to the knowledge-based and the Hybrid-A models. Furthermore, when taking into account the short-term prediction horizon of Fig. 6.3, a larger prediction horizon did not generally indicate a more accurate long-term statistics. For $N_{\text{lat}} = 192$, the variations in the root mean square error of $\langle u \rangle$ and $\langle v \rangle$ was slightly reduced, however for the deep learning model the error in $\langle v \rangle$ increased slightly for larger reservoirs as opposed to the increasing ESN contribution. For the largest latent space, a similar behaviour was found where the

performance tended decreased for increasing number of neurons. For both latent space sizes, such patterns were not observed for the short-term prediction horizon for the deep learning model (Fig. 6.3).

For the Hybrid-A reduced order model at $N_{\text{lat}} = 64$, it was shown for $\langle u \rangle$ that for smaller reservoirs the Hybrid-A model showed slightly better performance compared to the knowledge-based model. However, increasing the number of neurons resulted in a performance similar to the knowledge-based model. In contrast, for $\langle v \rangle$ at the smallest latent space, the Hybrid-A model was always outperformed by the knowledge-based model. However, compared to the deep learning model, no clear pattern was observed as a result of the large variations in the deep learning statistics. Then, for an increase of the latent space to $N_{\text{lat}} = 192$, the Hybrid-A model showed slight improvement over the knowledge-based model for $\langle u \rangle$, while similar performance was observed for $\langle v \rangle$. However, in comparison with the deep learning model, the Hybrid-A model showed less dependence on the reservoir size. Finally, for the largest latent space, still the performance of the Hybrid-A and knowledge-based were similar while the Hybrid-A model did not coincided well with the deep learning model for most experiments.

Aside from the root mean square error for $\langle u \rangle$ and $\langle v \rangle$ between the reduced order models and the reference solution, also the RMSE for the mean velocity fluctuations $\langle u' \rangle$ and $\langle v' \rangle$ were considered as depicted in Fig. 6.9. For both velocity fluctuation components, the error between the true and the predicted solution is of the order 10^{-14} since the mean fluctuations of the reference solution were also very small. Still, it showed that the fluctuations were well represented by all three reduced order models.

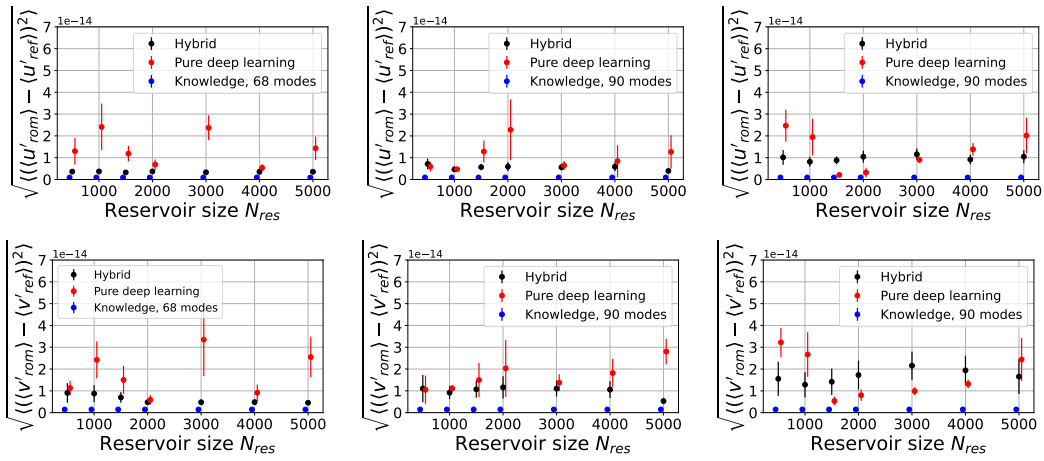


Figure 6.9: The root mean square error of the mean $\langle u' \rangle$ and $\langle v' \rangle$ between the reduced order models and the reference solution for the knowledge-based, deep learning and hybrid reduced order models as a function of the number of neurons in the reservoir at $Re = 34$. Here, the top left plot is for the multi-scale autoencoder with $N_{\text{lat}} = 64$, the top right for $N_{\text{lat}} = 192$ and the bottom plot for $N_{\text{lat}} = 320$. The knowledge-based and deep learning results are displaced slightly in the horizontal direction for clarity.

Finally, similar to $Re = 20$, the phase space trajectories of the dissipation $D(t)$ against kinetic energy $E(t)$ were plotted for all latent space and reservoirs size for the reference, knowledge-based, deep learning and Hybrid-A models. For this analysis, only the reservoir

sizes compromised of 500 and 2000 neurons are presented in Fig. 6.10 and Fig. 6.11. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 64$, second row $N_{\text{lat}} = 192$ and the third row $N_{\text{lat}} = 320$. Furthermore, the model are also repeated in the title of each plot. The results for $N_{\text{res}} = [1000, 1500, 3000, 4000, 5000]$ are depicted in Appendix F.

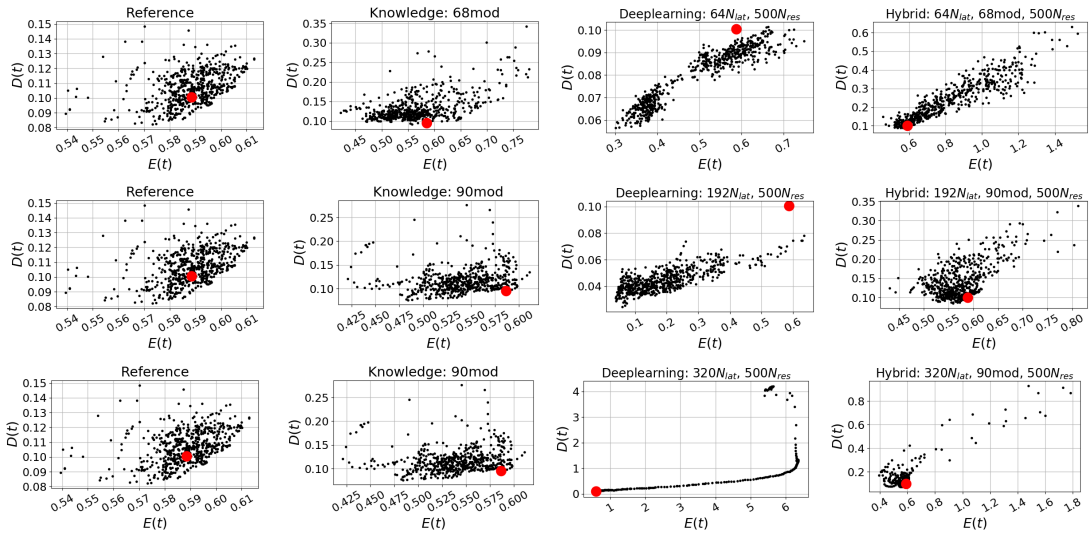


Figure 6.10: Plots showing the phase space trajectories of the dissipation $D(t)$ and the kinetic energy $E(t)$ for a reservoir of 500 neurons for the $Re = 34$ test case. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 64$, second row $N_{\text{lat}} = 192$ and the third row $N_{\text{lat}} = 320$ and the red dot represents $t = 0$.

For the knowledge-based reduced order model, the trajectories in phase space were confined to a domain larger than for the reference solution, however still most of the reference domain overlapped with the knowledge-based domain. In addition, the increase of the number of modes slightly improved the trajectories.

On the other hand, for the deep learning reduced order model, the long term kinetic energy and dissipation did not remain in the confined domain of the reference phase space. Rather, for most experiments the phase space trajectories diverged towards a distant region of much larger domain-averaged kinetic energy and in some cases also significantly larger dissipation. In these region, the trajectories either stagnated in a single fixed point or formed a point cloud. Furthermore, for some experiments the phase space trajectories moved to a region of lower kinetic energy and dissipation (for example $N_{\text{lat}} = 64$ and $N_{\text{res}} = 2000$). Now, for experiments such as $N_{\text{lat}} = 64$ with 2000 neurons and $N_{\text{lat}} = [192, 320]$ with 3000 and 4000 neurons a good performance in terms of $\langle u \rangle$ and $\langle v \rangle$ was found. However from the phase space trajectory it actually followed that these experiments moved towards a fixed point and thus were not accurate on longer time scales. Hence, for the mean flow and fluctuations, either the deep learning errors were much larger compared to the other models or of comparable performance. When comparing these results with the phase trajectories, good deep learning

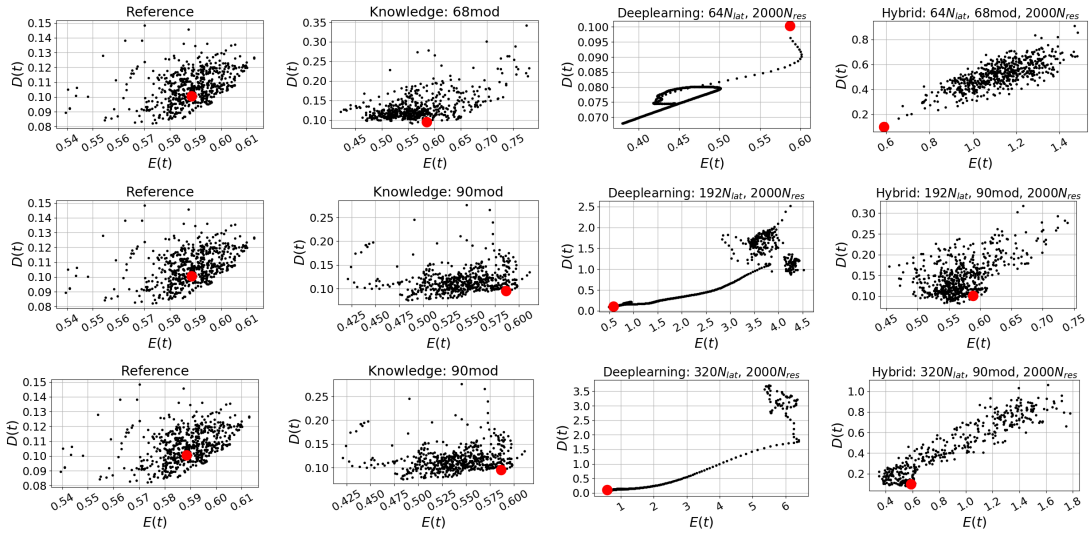


Figure 6.11: Plots showing the phase space trajectories of the dissipation $D(t)$ and the kinetic energy $E(t)$ for a reservoir of 2000 neurons for the $Re = 34$ test case. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 64$, second row $N_{\text{lat}} = 192$ and the third row $N_{\text{lat}} = 320$ and the red dot represents $t = 0$.

results for the temporal mean coincided for most experiments with a single point convergence close to the confined domain in phase space and thus these predictions actually did not perform well. Thus, comparing these findings with the short-term prediction horizon, no correlation in terms of short-term and long-term performance was found for the deep learning model.

The missing correlation between the short-term prediction horizon and long-term statistics however was surprising as Racca et al. [57] studied a similar deep learning reduced order model for which correlation between the short-term prediction horizon and the long-term statistics was observed. This was thought to be the result of the Bayesian optimisation of the hyperparameters since in this work tuning occurred by maximizing the prediction horizon and thus biasing focusing on that performance criteria. On the other hand, in the work of Racca et al. [57] the mean error between the predicted and reference solution was minimized over a much longer time frame. Thus, the ESN was made more suitable for longer term predictions. This was expected to be the most important factor for the lack of correlation between the short-term and long-term results in this work. Furthermore, Racca et al. [57] used an ensemble of 10 echo state networks to reduce the effect of the random initializations of the input and the internal matrices, while this work only used a single echo state network in view of computation cost of the Hybrid-A model. Other primary differences were that Racca et al. [57] used a significantly more grid points in the search space of the grid search-Bayesian optimisation for the hyperparameters and the leaking rate hyperparameter was not tuned but fixed. Thus, their tuned hyper parameter could have been more optimal compared to this work and thus worked better for prediction over longer time series.

Finally, for the phase space trajectories of the Hybrid-A model, in contrast to the deep

learning reduced order model, no divergence to a single fixed point was observed. This indicated that the inclusion of the knowledge-based component allowed to stabilize the model to a certain extend on longer prediction time scales. However, still for almost all experiments the trajectories were confined to a domain which was significantly larger than for the reference solution, where the dissipation was sometimes over 10 times larger and the kinetic energy was about 3 times larger. However, despite no fixed point convergence occurred, both the kinetic energy and the dissipation were significantly larger for most experiments. In contrast, for the hybrid architecture without autoencoders for low dimensional chaotic systems by Lesjak and Doan [7], the hybrid model was able to stay confined to the same phase space domain as the reference solution, further indicating the influence of the added autoencoders. Surprisingly, their hyperparameters were also determined from the short-term prediction horizon, possibly indicating the influence of the latent space representation and the accompanied lost dynamics.

Summary of long-term statistics

To summarize the long-term statistics, for the mean flow of the periodic regime ($Re = 20$) no consistent differences in performance were found for the Hybrid-A at larger reservoirs as well as larger latent spaces. Similar results were found for the phase space trajectories of the kinetic energy and dissipation, where an increase in autoencoder latent space dimensions had no profound influence. In terms of the comparative performance of the Hybrid-A model and the knowledge-based/deep learning models no clear pattern was found. Both were hypothesized to be the result of a non-optimal validation design for long term predictions and/or from the periodic nature of the flow. Still, due to the preservation of the mean flow in only the first POD mode, the knowledge-based model was able to show better relative performance compared to the short-term prediction horizon.

Now, increasing the Reynolds number to the $Re = 34$ chaotic regime, it was found that for the Hybrid-A reduced order model, the multi-scale autoencoder had little influence on the long-term statistics. Furthermore, for the comparative performance of mean flow, the Hybrid-A model mostly followed the knowledge-based solution hypothesized to be the result of the well preservation of the mean flow by the first POD mode. However, for the deep learning ROM most experiments tended towards a single point convergence as observed from the phase space trajectories. Similar to the knowledge-based, for the Hybrid-A model no fixed points were observed indicating that the knowledge-based component was able to somewhat stabilize the Hybrid-A model slightly. Still, the Hybrid-A phase space trajectories moved to a distant region of the phase space indicating less well performance compared to the knowledge-based model. Here, the poor performance of the deep learning model and consequently also the Hybrid-A model was likely a result of the hyperparameter tuning that favored the short-term prediction horizon.

6.3 Limitations

In the previous two sections, the results on the performance of the short-term prediction horizon and the long-term statistics of the Hybrid-A model were discussed. However, in the research, several limitations were present.

The first limitation of this work was the computational cost to evaluate the Hybrid-A architectures, which was a consequence of the (de)compression in every iteration over time. As a result, only a small amount of evaluations were performed in the Bayesian optimisation and thus it was anticipated that better performance could have been obtainable. In addition, the boundaries of the search space in Bayesian optimisation were limiting as the search space cannot be too wide for only a limiting number of evaluation points. However, as a result, more often than not one or multiple hyperparameters were tuned to these boundaries.

The second limitation of the model was the multi-scale autoencoder and more specifically the need to compress and decompress the physical system into a latent space format in every iteration. This requirements resulted from that fact that the ESN must operate on a lower dimensional representation of the physical system due to computational restrictions. However this compression/decompression added a small error in every iteration, which decreased with increasing latent space size. In addition, by compressing the physical into a latent space format, a lot of information on the system was removed resulting in a reduction of the predictive capabilities of the models operating on the latent space. The third limitation was the result of the construction of the knowledge-based model. Similar to Lesjak and Doan [7] an imperfect physical model was obtained through proper orthogonal decomposition and the projection of the governing equations onto the set of POD modes. Through a truncation of the number of modes, a certain amount of dynamics was removed from the system. However, since the POD modes were ranked based on their energy content, it was not controllable which dynamics were removed, only the amount energy that was represented. A second issue with the truncation of the number of modes was that instabilities were introduced into the reduced order model, which errors grew quickly over time.

The fourth limitation was in terms of hyperparameter tuning design where maximizing the short-term prediction horizon was chosen, whereas minimizing the error between the reference and reduce order solutions would have been more suitable for the long-term statistical predictions. Furthermore, the number of snapshots from which the POD modes were computed was larger than the number of training snapshots for the ESN and the POD mode data set also contained the reference data set. This could have resulted in slightly more biased performance of the knowledge-based component compared to the deep learning component. Finally, the last limitation was that the hybrid models were only applied one chaotic system and therefore its performance was not tested over a variety of different high dimensional chaotic systems. Here, the Reynolds number was also limiting as the spatial dimensions of a fluid mechanical systems increase fast with increasing Reynolds number [8]. Therefore, highly turbulent systems were not possible due to restriction in computational resources. In addition, larger Reynolds number would require larger latent spaces to retain low reconstruction errors by the autoencoder and thus the deep learning component would also be required to operate on increasingly large latent space dimensions.

Chapter 7

Hybrid-FFNN: Results and discussion

As stated earlier, also a second hybrid architecture was proposed in this work. However, only preliminary results were obtained for this model and for that reason it is treated separately from the results (Chapter 6) of the first (main) hybrid reduced order model introduced in section 5.4. In the first part of this chapter the proposed second hybrid architecture is introduced, which is based on the techniques described in Chapters 2, 3 and 5. Then, the model settings for the short-term predictions are briefly outlined and finally in the last part the preliminary results are shown. For this model, no long-term statistics were considered.

7.1 Hybrid model

The architecture of the second proposed hybrid model is presented in Fig. 7.1. The primary difference between this hybrid architecture and the Hybrid-A model (section 5.4) was the methodology of combining the predictions made by the knowledge-based and deep learning components, which did not occur in the echo state network. Rather, a separate feed-forward neural network was implemented that took the encoded knowledge-based and deep learning predictions as input and returned their combined contribution as the prediction in physical space. Here, the feed-forward neural network consisted of a dense neural network followed by the decoder from the multi-scale autoencoder.

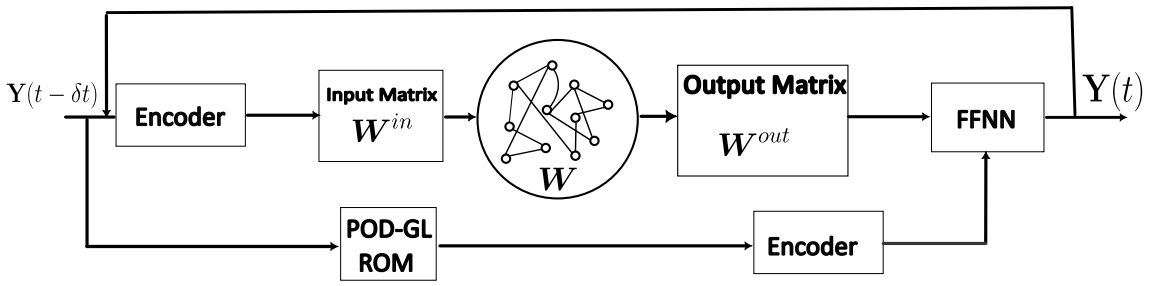


Figure 7.1: Image showing the second proposed hybrid based on a feed-forward neural network (FFNN) to combine the predictions by the knowledge-based and the deep learning components. Here, the feed-forward neural network consists of both the dense neural network followed by the decoder of the multi-scale autoencoder.

7.2 Feed-forward neural network

In the previous section, the proposed second Hybrid-FFNN model was presented and the primary differences with the Hybrid-A architecture were highlighted. In this section, more information is provided on the feed-forward neural network used to combine the knowledge-based and deep learning components.

Architecture
The architecture of the feed-forward neural network is presented in Appendix G and consisted of 8 layers. The first layer was the input layer with $2N_{\text{lat}}$ input parameters for the knowledge-based and deep learning predictions. Next, three dense neural layers each with 300 neurons and a tangent hyperbolic activation function were used, followed by a dense neural layer with 320 neurons also with a tangent hyperbolic activation function. The final dense neural layers also consisted of 320 neurons (corresponding to the latent space output dimensions) with a linear activation function. The aforementioned neural layers was composed of over 570,000 trainable parameters.

In order to train the feed-forward network, the physical space reference solution was used as target data. Therefore, the output of the last dense layer was reshaped by a reshaped layer since the decoder operated on two dimensional data and the output of the dense layers was one dimensional. Finally, the eighth layer was the multi-scale decoder which decompressed output of the last dense layer into physical space. In contrast to the dense layers, the parameters of the decoder were set to be non-trainable and were not altered.

Training and methodology
The feed-work network was trained on 65,000 snapshots equispaced by $\delta t = 0.4$ and thus coincide with 162,500 integer time units. For the validation 15,000 snapshots (37,500 integer time units) were used. For each snapshot, a one-step in time prediction was performed for the knowledge-based and the deep learning components. These solutions were combined in latent space and served as input data for the feed-forward neural network.

Training and validation data was obtained for $N_{\text{res}} = [500, 1000, 1500, 2000, 3000]$ for only

the multi-scale autoencoder with $N_{\text{lat}} = 320$ ($Re = 34$). The training and validation loss computed by the mean square error are shown Tab. 7.1.

Table 7.1: Table showing the training L_{train} and validation L_{val} losses of the feed-forward neural network used in the Hybrid-FFNN model.

N_{res}	L_{train}	L_{val}
500	0.0048	0.0067
1000	0.0048	0.0064
1500	0.0047	0.0060
2000	0.0046	0.0064
3000	0.0047	0.0060

In this Hybrid-FFNN architecture, the echo state network was pre-trained and thus the hyper parameters were already tuned using the grid search-Bayesian optimisation (section 5.7.1). Here, the same hyper parameter were used as obtained for the pure deep learning reduced order model (Appendix G). For the predictions, the same settings were used as for the Hybrid-A model as presented in section 5.4.

7.3 Results and discussion

For the Hybrid-FFNN only the largest latent space of $N_{\text{lat}} = 320$ was considered for the reservoirs sizes from 500 to 3000 neurons for only the short-term prediction horizon, to provide a demonstration of concept of this architecture. The short-term prediction horizon of the Hybrid-A and the Hybrid-FFNN model are depicted in Fig. 7.2

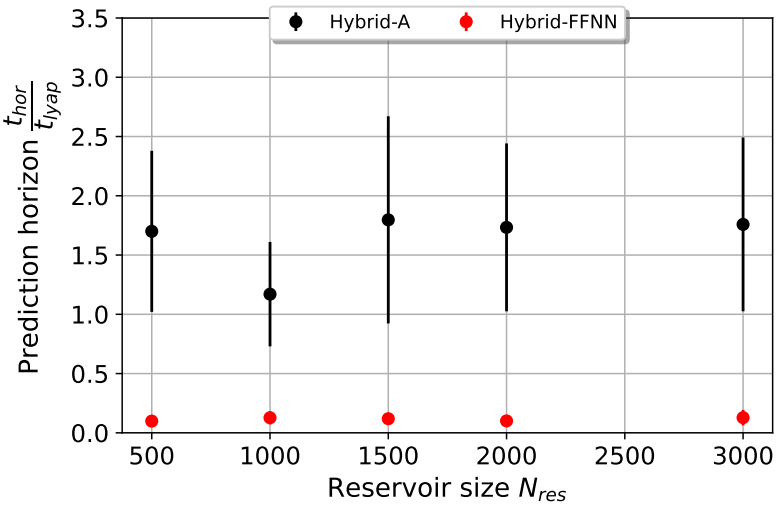


Figure 7.2: Plots showing the short-term prediction horizons of the hybrid-A and the hybrid-FFNN models as a function of the reservoir size for $N_{\text{lat}} = 320$.

The gap in performance was thought to be the consequence of the feed-forward neural network utilized by Hybrid-FFNN as the training and validation losses was several orders of magnitude larger than the multi-scale autoencoder (de)compression errors (Tab. 5.2). Thus in each iteration over time, a significant error was introduced into the Hybrid-FFNN solution.

Limitations

The first limitation concerned the training and validation losses of the feed-forward neural network. It was shown that performance of the Hybrid-FFNN was limited due to the large errors introduced by the FFNN. Hence, to reduce these errors, more dense layers and neurons in each layer should be added and as a result more training/validation data would have been required. However, the current network already consisted of over half a million trainable parameter and in view of the computational cost it was decided to not further increase the size of the network. Thus, the result in this work should be considered as a demonstration of concept of using a feed-forward neural network for the blending of the knowledge-based and deep learning predictions.

Furthermore, for the Hybrid-FFNN model an important limitation was that if the knowledge-based components performed a prediction in time, a certain amount of this physics is lost as a result of the truncation of the amount of modes. However, during a prediction of the deep learning component, also information was lost. Thus, when mixing the predictions by both components, the lost information cannot be retrieved. On the other hand, for the hybrid-A model, the blending occurred within the echo state network. Hence, the ESN was effectively trained to fill the gap between the true encoded solution and the knowledge-based encoded solution [7] such that it compensated for the lost physics.

Chapter 8

Conclusion

In order to answer the primary research question of thesis "How can a hybrid knowledge-based/deep learning reduced order model approach be designed to enable the accurate short term and long term predictions of high dimensional chaotic systems?" two main sub-questions were considered. In the first sub-question, two hybrid architectures using autoencoders were proposed, their performance in terms of short-term prediction horizon and long-term statistics and the influence of the latent space on the performance were addressed. In the second sub-question the hybrid model performance was compared to a pure knowledge-based model and a pure deep learning reduced order model. The knowledge-based model consisted of the Galerkin projection of the governing equations onto a set of orthonormal model obtained through proper orthogonal decomposition. The deep learning component was an echo state network that operated on the latent space representation of the physical system obtained through a multi-scale autoencoder.

In this work, the Hybrid-A and the Hybrid-FFNN architectures were proposed. In the Hybrid-A model, the one-step-ahead prediction of the knowledge-based component was combined with the Hybrid-A output of the previous time step and both were encoded into latent space representation and fed into the deep learning component in which a blending of the predictions occurred. Therefore, the deep learning component effectively learned the dynamics that were lost in the knowledge-based reduced order model. In contrast, in the Hybrid-FFNN model the blending of the predictions of the two components occurred in a separate feed-forward neural network. In order to study the performances, the Hybrid-A model was applied to two-dimensional Kolmogorov flow at a Reynolds number of $Re = 20$ (periodic regime) and $Re = 34$ (chaotic regime), while preliminary results were obtained for the Hybrid-FFNN model for the chaotic test case.

Short-term prediction horizon
For the Kolmogorov flow at $Re = 20$, the Hybrid-A model was able to predict for most experiments at least 5 periodic cycles. Since this test case was periodic, the dynamics were relatively simple and due to the low (de)compression errors by the multi-scale autoencoder also the la-

tent space representation would have been (approximately) periodic. For this reason, the Hybrid-A and deep learning were able to perform well in terms of the short-term prediction horizon compared to the knowledge-based technique that suffered from instabilities due to modal reduction. Furthermore, due to the simple dynamics, small reservoirs were already capable of reflecting the dynamics and hence no consistent improvement of performance was present for larger echo state networks and latent space representations. The relative contribution of the ESN in the Hybrid-A tended to increase for larger reservoir, but this pattern was only observed for the smaller two latent spaces. For this reason, the Hybrid-A and deep learning model predictions became similar for larger reservoirs. However, for smaller reservoirs no clear comparative performance between the two models was present, likely under influence of the larger contribution of the knowledge-based component. Finally, in the Hybrid-A model an accumulation of (de)compression errors occurred, which showed not to be of importance for this test case. This was due to the periodic nature of the flow and hence the model able to restore itself and the knowledge-based component was able to compensate for the errors.

Increasing the Reynolds number to $Re = 34$, such that the flow displayed chaotic behaviour significantly reduced the short-term prediction horizon of the Hybrid-A model. In addition, a larger latent space size was required for (de)compression errors similar to the $Re = 20$ test case. For $Re = 34$, the model was able to time-accurately predict for a significant shorter duration (about 0.5 to 1.8 Lyapunov times) and a larger dependence on initial conditions was present due to the chaotic nature. Furthermore, the influence of the latent space became more profound compared to $Re = 20$. For smaller latent spaces, a deteriorated Hybrid-A performance was the result of sufficiently large (de)compression errors by the autoencoder combined with the chaotic nature of the system and thus that in each iteration the knowledge-based component pushed the model solution further away from the reference solution. As the latent space increased, the (de)compression error decreased and was shown to become less significant for the Hybrid-A performance. Furthermore, the deep learning model performance did not consistently depend on the latent space size, resulting in an increasing contribution of the knowledge-based component in the Hybrid-A model as larger latent spaces were able to preserve more dynamics from the knowledge-based prediction. Still, the Hybrid-A model was not able to outperform the knowledge-based model for larger latent spaces as well, which was likely caused by the latent space representation through which too much information was lost to benefit from both components.

Finally, the Hybrid-FFNN model was only applied to $Re = 34$ Kolmogorov flow at the largest latent space and had a significantly less performance compared to the Hybrid-A model due to the large error accumulation of the feed-forward neural network.

Long-term statistics

For the long-term statistics of the $Re = 20$ and $Re = 34$ test case a less consistent pattern in terms of performance was observed. For $Re = 20$, the mean flow of the knowledge-based model, in contrast to the short-term prediction horizon, was similar to that of the other two models. This was explained by the fact that the mean was fully captured by the first POD mode and thus well represented. Furthermore, for the Hybrid-A model no influence was found on performance was observed for increasing latent space and reservoir size. Furthermore, for the comparative performance of the three models between the short-term prediction horizon and the long-term statistics no clear pattern was observed. These outcomes were hypothesized

to be the result of the tuning of the hyperparameters as it was chosen to optimized for the short-term prediction horizon and hence statistics over longer time series were less well represented and the periodicity of the flow.

Similar results were found for the long-term statistics of the $Re = 34$ test case. For the mean flow, the knowledge-based model showed similar performance to the Hybrid-A model as a result from the preservation of the mean in the first POD mode. On the other hand, the deep learning model showed a larger variation in mean flow. However, when the phase space trajectories of the dissipation and kinetic energy were consider, it was found that the deep learning model tended to move to either a fixed point or a point cloud in a distant region of the phase space. Hence, despite showing good performance on the mean, the trajectory showed that the solution tended to a stagnation point, which was not observed in literature using a similar model and test case [57], indicating the importance of the hyperparameter tuning method. However, for the hybrid model such stagnation point was not observed showing that the addition of the knowledge-based component was able to stabilize the model to a certain extent. Still, most experiments with the hybrid model tended toward a phase space region distant from the reference solution. Thus, similar to $Re = 20$, the results on $Re = 34$ showed that the chosen validation design was unsuitable for predictions over longer time series.

To conclude both the short-term prediction horizon and long-term statistics. The extension of the hybrid model by Pathak et al. [61] and Lesjak and Doan [7] to be compatible with high dimensional chaotic system through the use of autoencoders for dimensional reduction did not allow the hybrid architecture to consistently outperform both the knowledge-based and deep learning reduced order model. This was in contrast to the results obtained by Lesjak and Doan [7] for a similar hybrid model design without the autoencoders for low dimensional chaotic systems. Thus, combining a proper orthogonal decomposition Galerkin reduced order model with autoencoder-based deep learning model resulted in the addition of too much errors such that the additional information provided by the knowledge-based component was not beneficial. These errors originated from the iterative (de)compression error by the autoencoder as well as lost dynamics in the latent space. Finally, increasing the Reynolds number resulted in the need for a larger latent space to keep the (de)compression errors low. This could be potentially limiting for higher Reynolds number flow for which the spatial resolution continues to increase, since the latent space would become larger to keep the (de)compression errors low. As a result, the echo state network would have to operate on increasingly large latent space representations.

Up to the authors knowledge, this work proposed the first hybrid knowledge-based/deep learning reduced order model using autoencoders for efficient predictions of high dimensional chaotic systems. Even though it was found that the proposed Hybrid models were unable to show increased performance compared to the knowledge-based and deep learning models, the results should be viewed as a start to design other hybrid architectures for potential performance improvement.

Recommendations for future work

In the present work, the performance of a newly proposed hybrid knowledge-based/deep learning reduced order model was assessed. It was shown that the extension of the hybrid model of Pathak et al. [61] and Lesjak and Doan [7] to efficiently handle high dimensional

chaotic systems by including autoencoders for dimensional reduction did not result in a better performance compared to the knowledge-based and deep learning ROMs. Therefore, from this research, future studies can focus on several open questions.

Firstly, future research can focus on other hybrid architecture designs using either autoencoders or other dimensional reduction techniques to allow the hybrid model to show better performance compared to the pure knowledge-based and pure deep learning reduced order model. Furthermore, the autoencoder (or other dimensional reduction methods) can be improved by attempting to allow the latent space to reflect more dynamics from the physical system, while keeping the latent space low-dimensional. Next, in the validation of the ESN, the hyperparameters were tuned by maximizing the short-term prediction horizon, which was found to be less optimal for the long-term statistics. Hence, future research can focus on different validation methods and how these influence the performance of the hybrid models.

The proposed hybrid architectures were only applied to a single high dimensional chaotic system. Thus, future research can also study the architecture on other chaotic systems such as linearly forced isotropic turbulence [88]. However, as the complexity of the chaotic system increases, the knowledge-based type of approach through proper orthogonal decomposition combined with Galerkin projection might proof to not be sufficient due to instabilities in the model. Thus further research in stabilizing or applying different types of the knowledge-based approaches would also be of interest.

Finally, for the Hybrid-FFNN model a slight adaption to the current architecture can be studied where the echo state network is trained to perform prediction on the difference between the knowledge-based and the true solution and thus attempts to fill the gap of the truncation of the POD-modes as well as that these dynamics are easier to learn.

Bibliography

[1] D. Toker, F. T. Sommer, and M. D'Esposito. A simple method for detecting chaos in nature. *Communications biology*, 3(1):11, 2020.

[2] M. S. Demir, R. N. Ahmet Karaman, and S. D. Oztekin. Chaos Theory and Nursing. *International Journal of Caring Sciences*, 12(2):1–4, 2019.

[3] D. Helbing, A. Johansson, and H. Z. Al-Abideen. Dynamics of crowd disasters: An empirical study. *Physical review E*, 75(4):046109, 2007.

[4] E. N. Lorenz. Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, 20(2):130–141, 1963.

[5] É. Ghys. The Lorenz Attractor, a Paradigm for Chaos. In *Chaos: Poincaré Seminar 2010*, pages 1–54. Springer, 2013.

[6] The Royal Dutch Meteorological Institue (KNMI). Achtergrondinformatie Weer- en klimaatpluim en Expertpluim. https://www.knmi.nl/kennis-en-datacentrum/achtergrond/over-de-weer-en-klimaatpluim-en-expertpluim, 2023. Accessed: 23-05-2023.

[7] M. Lesjak and N. A. K. Doan. Chaotic systems learning with hybrid echo state network/proper orthogonal decomposition based model. *Data-Centric Engineering*, 2:e16, 2021.

[8] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[9] O. Reynolds. Iv. on the Dynamical Theory of Incompressible Viscous Fluids and the Determination of the Criterion. *Philosophical Transactions of the Royal Society of London.(a.)*, (186):123–164, 1895.

[10] J. Smagorinsky. General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review*, 91(3):99–164, 1963.

[11] C. W. Rowley and S. T. M. Dawson. Model Reduction for Flow Analysis and Control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

[12] K. Taira, S. L. Brunton, S. T. M. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley. Modal Analysis of Fluid Flows: An Overview. *AIAA Journal*, 55(12):4013–4041, 2017.

[13] W. Keiper, A. Milde, and S. Volkwein. *Reduced-order modeling (ROM) forSsimulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing.* Springer, 2018.

[14] B. R. Noack, M. Morzynski, and G. Tadmor. *Reduced-order modelling for flow control*, volume 528. Springer Science & Business Media, 2011.

[15] A. Quarteroni, G. Rozza, et al. *Reduced Order Methods for Modeling and Computational Reduction*, volume 9. Springer, 2014.

[16] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry.* Cambridge University Press, 2012.

[17] J. L. Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric Turbulence and Radio Wave Propagation*, pages 166–178, 1967.

[18] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[19] M. Sieber, C. O. Paschereit, and K. Oberleithner. Spectral proper orthogonal decomposition. *Journal of Fluid Mechanics*, 792:798–828, 2016.

[20] A. E. Deane, I. G. Kevrekidis, G. E. Karniadakis, and S. A. Orszag. Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders. *Physics of Fluids A: Fluid Dynamics*, 3(10):2337–2354, 1991.

[21] W. Cazemier, R. W. C. P. Verstappen, and A. E. P. Veldman. Proper orthogonal decomposition and low-dimensional models for driven cavity flows. *Physics of Fluids*, 10(7):1685–1699, 1998.

[22] A. Omurtag and L. Sirovich. On Low-Dimensional Modeling of Channel Turbulence. *Theoretical and Computational Fluid Dynamics*, 13(2):115–127, 1999.

[23] P. S. Johansson, H. I. Andersson, and E. M. Rønquist. Reduced-basis modeling of turbulent plane channel flow. *Computers & Fluids*, 35(2):189–207, 2006.

[24] X. Ma and G. E. Karniadakis. A low-dimensional model for simulating three-dimensional cylinder flow. *Journal of Fluid Mechanics*, 458:181–190, 2002.

[25] M. Buffoni, S. Camarri, A. Iollo, and M. V. Salvetti. Low-dimensional modelling of a confined three-dimensional wake flow. *Journal of Fluid Mechanics*, 569:141–150, 2006.

[26] C. W. Rowley, T. Colonius, and R. M. Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004.

[27] R. Bourguet, M. Braza, and A. Dervieux. Reduced-order modeling for unsteady transonic flows around an airfoil. *Physics of Fluids*, 19(11):111701, 2007.

[28] J. Chen, D. Han, B. Yu, D. Sun, and J Wei. A POD-Galerkin reduced-order model for isotropic viscoelastic turbulent flow. *International Communications in Heat and Mass Transfer*, 84:121–133, 2017.

[29] Y. Wang, H. Ma, W. Cai, H. Zhang, J. Cheng, and X. Zheng. A POD-Galerkin reduced-order model for two-dimensional Rayleigh-Bénard convection with viscoelastic fluid. *International Communications in Heat and Mass Transfer*, 117:104747, 2020.

[30] J. Chen, D. Han, B. Yu, D. Sun, and J. Wei. POD-Galerkin reduced-order model for viscoelastic turbulent channel flow. *Numerical Heat Transfer, Part B: Fundamentals*, 72(3):268–283, 2017.

[31] A. E. Deane and C. Mavriplis. Low-Dimensional Description of the Dynamics in Separated Flow past Thick Airfoils. *AIAA journal*, 32(6):1222–1227, 1994.

[32] Sk. M. Rahman, S. E. Ahmed, and O. San. A dynamic closure modeling framework for model order reduction of geophysical flows. *Physics of Fluids*, 31(4):046602, 2019.

[33] L. Gonon and J.-P. Ortega. Reservoir Computing Universality With Stochastic Inputs. *IEEE transactions on neural networks and learning systems*, 31(1):100–112, 2019.

[34] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[35] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an Erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[36] W. E. Faller and S. J. Schreck. Unsteady Fluid Mechanics Applications of Neural Networks. *Journal of Aircraft*, 34(1):48–55, 1997.

[37] A. Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[38] H. Jaeger and H. Haas. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667):78–80, 2004.

[39] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[40] A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, and E. Ott. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(5):053111, 2020.

[41] Z. Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, and Y. Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[42] Sk. M. Rahman, S. Pawar, O. San, A. Rasheed, and T. Iliescu. Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. *Physical Review E*, 100(5):053306, 2019.

[43] A. T. Mohan and D. V. Gaitonde. A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. *arXiv preprint arXiv:1804.09269*, 2018.

[44] Z. Deng, Y. Chen, Y. Liu, and K. C. Kim. Time-resolved turbulent velocity field reconstruction using a long short-term memory (LSTM)-based artificial intelligence framework. *Physics of Fluids*, 31(7):075108, 2019.

[45] P. A. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter, and R. Vinuesa. Predictions of turbulent shear flows using deep neural networks. *Physical Review Fluids*, 4(5):054603, 2019.

[46] A. Lario, R. Maulik, O. T. Schmidt, G. Rozza, and G. Mengaldo. Neural-network learning of SPOD latent dynamics. *Journal of Computational Physics*, 468:111475, 2022.

[47] S. Pandey and J. Schumacher. Reservoir computing model of two-dimensional turbulent convection. *Physical Review Fluids*, 5(11):113506, 2020.

[48] F. Heyder and J. Schumacher. Echo state network for two-dimensional turbulent moist Rayleigh-Bénard convection. *Physical Review E*, 103(5):053107, 2021.

[49] A. Racca and L. Magri. Data-driven prediction and control of extreme events in a chaotic flow. *Physical Review Fluids*, 7(10):104402, 2022.

[50] R. Maulik, B. Lusch, and P. Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, 2021.

[51] A. Shrestha and A. Mahmood. Review of Deep Learning Algorithms and Architectures. *IEEE access*, 7:53040–53065, 2019.

[52] R. Han, Y. Wang, Y. Zhang, and G. Chen. A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Physics of Fluids*, 31(12):127101, 2019.

[53] T. Murata, K. Fukami, and K. Fukagata. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics*, 882:A13, 2020.

[54] S. Pandey, P. Teutsch, P. Mäder, and J. Schumacher. Direct data-driven forecast of local turbulent heat flux in Rayleigh-Bénard convection. *Physics of Fluids*, 34(4):045106, 2022.

[55] K. Hasegawa, K. Fukami, T. Murata, and K. Fukagata. Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. *Theoretical and Computational Fluid Dynamics*, 34:367–383, 2020.

[56] T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, and K. Fukagata. Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Physics of Fluids*, 33(2):025116, 2021.

[57] A. Racca, N. A. K. Doan, and L. Magri. Modelling spatiotemporal turbulent dynamics with the convolutional autoencoder echo state network. *arXiv preprint arXiv:2211.11379*, 2022.

[58] P. Wu, S. Gong, K. Pan, F. Qiu, W. Feng, and C. Pain. Reduced order model using convolutional auto-encoder with self-attention. *Physics of Fluids*, 33(7):077107, 2021.

[59] S. Dutta, P. Rivera-Casillas, B. Styles, and M. W. Farthing. Reduced Order Modeling Using Advection-Aware Autoencoders. *Mathematical and Computational Applications*, 27(3):34, 2022.

[60] K. Fukami, T. Nakamura, and K. Fukagata. Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. *Physics of Fluids*, 32(9):095110, 2020.

[61] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):041101, 2018.

[62] Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5):e0197704, 2018.

[63] S. Pawar, S. E. Ahmed, O. San, and A. Rasheed. Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Physics of Fluids*, 32(3):036602, 2020.

[64] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[65] J. Weiss. A Tutorial on the Proper Orthogonal Decomposition. In *AIAA Aviation 2019 forum*, page 3333, 2019.

[66] M. Lesjak. Prediction of Chaotic Systems with Physics - Enhanced Machine Learning Models (Master thesis). 2020.

[67] H. G. Matthies and M. Meyer. Nonlinear Galerkin methods for the model reduction of nonlinear dynamical systems. *Computers & structures*, 81(12):1277–1286, 2003.

[68] K. Hornik, M. Stinchcombe, and H. White. Multilayer Feedforward Networks are Universal Approximators. *Neural networks*, 2(5):359–366, 1989.

[69] M. Lukoševičius. A Practical Guide to Applying Echo State Networks. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 659–686. Springer, 2012.

[70] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[71] Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. Chapter 10 - Convolutional neural networks. In Andrea M. and Sandra V., editors, *Machine Learning*, pages 173–191. Academic Press, 2020.

[72] N. Platt, L. Sirovich, and N. Fitzmaurice. An investigation of chaotic Kolmogorov flows. *Physics of Fluids A: Fluid Dynamics*, 3(4):681–696, 1991.

[73] G. J. Chandler and R. R. Kerswell. Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow. *Journal of Fluid Mechanics*, 722:554–595, 2013.

[74] D. Kelshaw. kolsol 1.0.1. https://pypi.org/project/kolsol/, 2022.

[75] C. Canuto, M. Yousuff Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics*. Springer Berlin, Heidelberg, 1988.

[76] M. D. Hartl. Lyapunov exponents in constrained and unconstrained ordinary differential equations. *arXiv preprint physics/0303077*, 2003.

[77] J. L. Kaplan and J. A. Yorke. Chaotic behavior of multidimensional difference equations. In *Functional Differential Equations and Approximation of Fixed Points: Proceedings, Bonn, July 1978*, pages 204–227. Springer, 2006.

[78] A. Racca and L. Magri. Robust optimization and validation of echo state networks for learning chaotic dynamics. *Neural Networks*, 142:252–268, 2021.

[79] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu. Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237:10–26, 2012.

[80] S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

[81] A.-K. Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

[82] H. Montanelli and Y. Nakatsukasa. Fourth-order time-stepping for stiff pdes on the sphere. *SIAM Journal on Scientific Computing*, 40(1):A421–A451, 2018.

[83] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[84] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[85] J. R. Sashank, K. Satyen, and K. Sanjiv. On the convergence of adam and beyond. In *International conference on learning representations*, volume 5, 2018.

[86] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[87] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[88] T. S. Lundgren. Linearly Forces Isotropic Turbulence. Technical report, MINNESOTA UNIV MINNEAPOLIS, 2003.

[89] P. I. Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[90] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in neural information processing systems*, 25, 2012.

[91] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[92] A. Griffith, A. Pomerance, and D. J. Gauthier. Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12), 2019.

[93] D. R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of global optimization*, 21:345–383, 2001.

[94] C. Rasmussen and C. Williams. In *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[95] J. Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference: Novosibirsk, July 1–7, 1974*, pages 400–404. Springer, 1975.

[96] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13:455–492, 1998.

[97] G. Benettin, L. Galgani, A. Giorgilli, and J. Strelcyn. Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them. Part 1: Theory. *Meccanica*, 15:9–20, 1980.

A Bayesian optimisation

The performance of the echo state network was highly dependent on the prescribed values for the spectral radius ρ, input scaling σ_{in}, Tikhonov parameter β and the leaky integration rate α [69] as outlined in section 3.2.1. Thus, tuning of the hyperparameters was required to obtain good performance for the pure data-driven model and the hybrid models. In this work, Bayesian optimisation [89, 90, 91] is used for hyperparameter tuning. It was shown for computational costly functions and a low number of evaluation points in the optimisation that hyperparameter tuning through Bayesian optimisation is able to outperform other approaches [92, 93]. Here, the principles of Bayesian optimisation are discussed following the tutorials of Brochu et al. [91] and Frazier [89] and the book of Rasmussen and Williams [94].

The primary aim of Bayesian optimisation is to globally optimize (in this section maximizing is considered) a black-box (objective) function $g_{\text{bb}}(\zeta)$ where $\zeta \in \mathbb{R}^{N_\zeta}$ is the search space bounded by the hyperparameters specifying the black-box function [89]. Here, $g_{\text{bb}}(\zeta)$ is a computational expensive, continuous function with no available information about its structure (such as convex/concave) [89]. Furthermore, due to the large computational cost per evaluation, the amount of sampling points in the optimisation is typically low [89].

Bayes theory

Bayesian optimisation relies on the statistical foundation of Bayes theory, which is briefly outlined here. Bayes theory states that the probability P of observing a model M given certain data O is given by [91]

$$P(M|O) \propto P(O|M)P(M). \tag{A.1}$$

Here, $P(M|O)$ is the posterior distribution, which states the probability of M given O. Furthermore, $P(M)$ is the prior distribution providing initial knowledge on the model and $P(O|M)$ is the likelihood stating the probability of data O given the model M [89, 91, 94]. Assume that (limited) initial data $g_{\text{bb}}(\zeta_{\text{in}})$ is available at points ζ_{in} (subscript "in" for initial), then the Bayes rules can be written as [91]

$$P(g_{\text{bb}}|\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}) \propto P(\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}|g_{\text{bb}})P(g_{\text{bb}}), \tag{A.2}$$

where the initial data is collected in $\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})} = \{g_{\text{bb}}(\zeta_{\text{in}}), \zeta_{\text{in}}\}$.

Gaussian Process regression

Gaussian process regression (GPR) is a common methodology to predict the function value at new points [89, 91, 94]. The primary assumptions is that the objective function $g_{\text{bb}}(\zeta)$ can be written as a Gaussian process (GP) such that [89, 91, 94]

$$g_{\text{bb}}(\zeta) \sim \mathcal{GP}(\mu_0(\zeta), \Sigma_0(\zeta, \zeta)), \tag{A.3}$$

where $\mu_0(\zeta)$ is a mean function and $\Sigma_0(\zeta, \zeta)$ is the covariance function/kernel. Common kernels are the square exponential and the Matern kernels. Here, a Gaussian process is an infinite set of random variables, such that each subset of these random variables is described by multi-variate Gaussian. In other words, a Gaussian process is an infinite collection of functions, where for every point in the space ζ, it returns the mean and variance over all these functions at that point [89, 91, 94].

Since $g_{\text{bb}}(\zeta)$ is assumed to be described as a Gaussian process and given the aforementioned definitions, the prior of the initial data $\{\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}, \boldsymbol{\zeta_{\text{in}}}\}$ can be described using a multi-variate Gaussian [91, 94]

$$\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})} \sim \mathcal{N}(\mu_0(\boldsymbol{\zeta_{\text{in}}}), \Sigma_0(\boldsymbol{\zeta_{\text{in}}}, \boldsymbol{\zeta_{\text{in}}})). \tag{A.4}$$

However, using the known data, it is not yet possible to predict the value at a new point ζ_+ in the hyperparameter space. Assume, for now, that the new point ζ_+ is given. Then, since the objective function is assumed to be described by a Gaussian process and the new point is a subset of the full space, also the prior [89, 91, 94]

$$g_{\text{bb}}(\zeta_+) \sim \mathcal{N}(\mu_0(\zeta_+), \Sigma_0(\zeta_+, \zeta_+)) \tag{A.5}$$

is a multivariate Gaussian [89, 91, 94]. Thus, the initial data set $\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}$ and the new point $g_{\text{bb}}(\zeta_+)$ are described as multi-variate Gaussian and their joint (normal) distribution is [89, 91, 94]

$$\begin{bmatrix} \boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})} \\ g_{\text{bb}}(\zeta_+) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_0(\boldsymbol{\zeta_{\text{in}}}) \\ \mu_0(\zeta_+) \end{bmatrix}, \begin{bmatrix} \Sigma_0(\boldsymbol{\zeta_{\text{in}}}, \boldsymbol{\zeta_{\text{in}}}) & \Sigma_0(\zeta_+, \boldsymbol{\zeta_{\text{in}}}) \\ \Sigma_0(\boldsymbol{\zeta_{\text{in}}}, \zeta_+) & \Sigma_0(\zeta_+, \zeta_+) \end{bmatrix} \right). \tag{A.6}$$

Then following Frazier [89] and Rasmussen and Williams [94], the posterior distribution $g_{\text{bb}}(\zeta_+)|[\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}, \zeta_+]$ for a new point ζ_+ based on the priors and the initial data is written as

$$\begin{aligned} g_{\text{bb}}(\zeta_+)|[\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}, \zeta_+] &\sim \mathcal{N}\left(\mu_n(\zeta_+), \Sigma_n(\zeta_+)\right) \\ \mu_n(\zeta_+) &= \Sigma_0(\boldsymbol{\zeta_{\text{in}}}, \zeta_+)\Sigma_0(\boldsymbol{\zeta_{\text{in}}}, \boldsymbol{\zeta_{\text{in}}})^{-1}\left(\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})} - \mu_0(\boldsymbol{\zeta_{\text{in}}})\right) + \mu_0(\zeta_+) \\ \Sigma_n(\zeta_+) &= \Sigma_0(\zeta_+, \zeta_+) - \Sigma_0(\boldsymbol{\zeta_{\text{in}}}, \zeta_+)\Sigma_0(\boldsymbol{\zeta_{\text{in}}}, \boldsymbol{\zeta_{\text{in}}})^{-1}\Sigma_0(\zeta_+, \boldsymbol{\zeta_{\text{in}}}), \end{aligned} \tag{A.7}$$

which is also a multi-variate normal distribution.

Acquisition function: Expected improvement

In Bayesian optimisation, the location of the next point ζ_+ in the search space is decided through an acquisition function [89, 91]. A popular choice is the Expected Improvement (EI) as introduced by Močkus [95].

For the Expected Improvement, consider the initial data $\boldsymbol{g}_{\text{bb}}(\boldsymbol{\zeta_{\text{in}}})$ at evaluation points $\boldsymbol{\zeta_{\text{in}}}$. Since Bayesian optimisation attempts to find a maximum for the objective function $g_{\text{bb}}(\zeta)$, interest is in a new point ζ_+ for which a new maximum value is found such that $g_{\text{bb}}(\zeta_+) > \boldsymbol{g}_{\text{bb}}^{\text{max}}(\boldsymbol{\zeta_{\text{in}}})$, where the superscript "max" indicates that it is the current maximum in the initial data set [89]. Now, for a given ζ_+, $g_{\text{bb}}(\zeta_+)$ can be larger or smaller than $\boldsymbol{g}_{\text{bb}}^{\text{max}}(\boldsymbol{\zeta_{\text{in}}})$. If $g_{\text{bb}}(\zeta_+)$ is larger, this means that it becomes the new best value. However for smaller $g_{\text{bb}}(\zeta_+)$ the maximum value in the initial data set $\boldsymbol{g}_{\text{bb}}(\boldsymbol{\zeta_{\text{in}}})$ remains [89]. Following the notation of Jones et al. [96], this improvement is written as $\max(g_{\text{bb}}(\zeta_+) - \boldsymbol{g}_{\text{bb}}^{\text{max}}(\boldsymbol{\zeta_{\text{in}}}), 0)$.

Now, even though ζ_+ is not know beforehand, the expectation value of the improvement (Expected Improvement) can be computed under the posterior distribution (Eq. A.7) as a function of ζ with [89, 96]

$$\text{EI}(\zeta) = \mathbb{E}[\max(\tilde{g}_{\text{bb}}(\zeta) - \boldsymbol{g}_{\text{bb}}^{\text{max}}(\boldsymbol{\zeta_{\text{in}}}), 0)]. \tag{A.8}$$

Here, $\tilde{g}_{\text{bb}}(\zeta_+)$ is a multi-variate normal distribution with mean function $\mu_n(\zeta_+)$ and covariance kernel $\Sigma_n(\zeta_+)$ taken from the posterior distribution (Eq. A.7) [96]. Now, by using Eq. A.8, the expected value of the improvement of the posterior compared to maximum of the initial data set $\boldsymbol{g}_{\text{bb}}(\boldsymbol{\zeta_{\text{in}}})$ is computed. Hence, the maximum of the expected value of interest and the corresponding ζ_+ is used by the objective function to evaluate the output at that point [89, 96].

Algoritm of Bayesian optimisation

The acquisition function and GPR form the foundation of the Bayesian optimisation algorithm [89, 91]. In the first step before the actual optimisation, N_{rdm} points in ζ are evaluated using the objective function $g_{\text{bb}}(\zeta)$, whose values are stored as initial data $\{g_{\text{bb}}(\zeta_{\text{in}}), \zeta_{\text{in}}\}$. Then, using the initial points ζ_{in}, the prior distribution of $\boldsymbol{g_{\text{bb}}(\zeta_{\text{in}})}$ is constructed [89, 91].

Using the initial data points and the prior, the posterior distribution is constructed using all currently available information. Then, using the acquisition function (for example the Expected Improvement, Eq. A.8), the next evaluation point ζ_+ is determined using the posterior distribution [89, 96]. Next, at ζ_+, the output of the black-box function $g_{\text{bb}}(\zeta_+)$ is computed. Finally, the newly obtained evaluation of $g_{\text{bb}}(\zeta_+)$ is added to the known initial data set and the prior and posterior distributions are updated. This optimisation process is repeated for a desired N_{bo} times [89, 91]. The outline of the algorithm is also depicted in the flowchart of Fig. A.1.

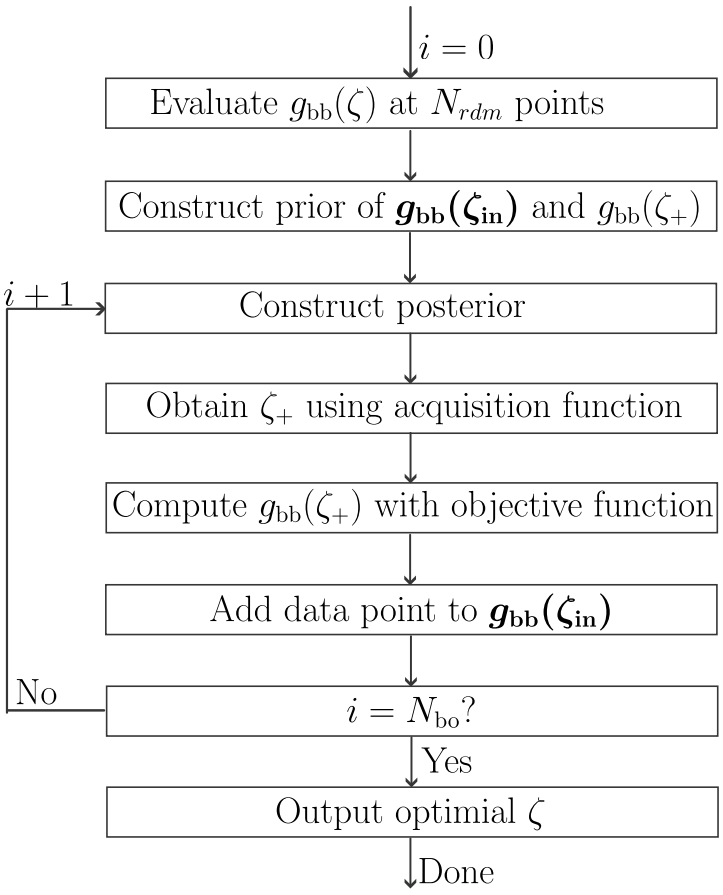


Figure A.1: Schematic flowchart showing the general procedure of the Bayesian optimisation.

B Lyapunov exponent computations

The Lyapunov exponent provides an indication of the average rate of separation of trajectories with slightly different initial conditions separate over time for chaotic systems [76], which were computed using the numerical code of Racca et al. [57].

Consider Eq. 2.12, but for simplicity using a single nonlinear operator $\mathfrak{h}$ such that the state of the chaotic system is described as

$$\dot{\boldsymbol{y}}(\boldsymbol{x},t) = \mathfrak{h}(\boldsymbol{y}(\boldsymbol{x},t)). \tag{B.9}$$

In order to compute the Lyapunov exponents, for both test case, the base line solution $\boldsymbol{y}(\boldsymbol{x},t)$ was integrated in time together with 15 additional simulations with slightly randomly perturbed initial conditions $\hat{\boldsymbol{y}}(\boldsymbol{x},t)$, obtained through [57]

$$\hat{\boldsymbol{y}}_i(\boldsymbol{x},t) = \boldsymbol{y}(\boldsymbol{x},t) + \epsilon_{\text{Lyap}} X_{\text{rand}}, \tag{B.10}$$

where $\epsilon_{\text{Lyap}} \ll 1$ is a small perturbation factor, X_{rand} was a random normalized matrix and $i = 1 \dots 15$ gives the index of the perturbed states. Then, the primary idea was to divide the full time integration into small intervals [97]. At the end of each integration interval of all 16 initial conditions, the evolution of the perturbation $\delta\boldsymbol{y}_i(\boldsymbol{x},t)$ was obtained by subtracting the base line solution from each perturbed state [97] using

$$\delta\boldsymbol{y}_i(\boldsymbol{x},t) = \hat{\boldsymbol{y}}_i(\boldsymbol{x},t) - \boldsymbol{y}(\boldsymbol{x},t). \tag{B.11}$$

Then, following Benettin et al. [97], at the end of each integration interval all the perturbation $\delta\boldsymbol{y}(\boldsymbol{x},t)$ were orthonormalized through a Gram-Schmidt like procedure [57]. Doing so, the first of the 15 perturbation was normalized as [57, 97]

$$\delta\boldsymbol{y}_{\text{norm},1}(\boldsymbol{x},t) = \frac{\delta\boldsymbol{y}_1(\boldsymbol{x},t)}{||\delta\boldsymbol{y}_1(\boldsymbol{x},t)||}. \tag{B.12}$$

Then for all other $i = 2 \dots 15$ perturbations, the orthogonalizing occurred by subtracting the projections of $\delta\boldsymbol{y}_i(\boldsymbol{x},t)$ onto the perturbations $\delta\boldsymbol{y}_{\text{norm},j}(\boldsymbol{x},t)$ for $j < i$ using [57, 97]

$$\delta\boldsymbol{y'}_i(\boldsymbol{x},t) = \delta\boldsymbol{y}_i(\boldsymbol{x},t) - \sum_{j=1}^{i-1} (\delta\boldsymbol{y}_i(\boldsymbol{x},t), \delta\boldsymbol{y}_{\text{norm},j}(\boldsymbol{x},t)) \delta\boldsymbol{y}_{\text{norm},j}(\boldsymbol{x},t) \tag{B.13}$$

Here, $\delta \boldsymbol{y'}_i(\boldsymbol{x}, t)$ is the orthogonalized i-th perturbation. Hence, the components of $\delta \boldsymbol{y}_i(\boldsymbol{x}, t)$ in the direction of all $\delta \boldsymbol{y}_{\text{norm},j}(\boldsymbol{x}, t)$ with $j < i$ were removed and thus all perturbations were orthogonal to each other. Then, the orthogonalized perturbation was also normalized such that [57, 97]

$$\delta \boldsymbol{y}_{\text{norm},i}(\boldsymbol{x}, t) = \frac{\delta \boldsymbol{y'}_i(\boldsymbol{x}, t)}{||\delta \boldsymbol{y'}_i(\boldsymbol{x}, t)||} \tag{B.14}$$

to obtain the orthonormalization of $\delta \boldsymbol{y}_i(\boldsymbol{x}, t)$. This orthonormalized basis for each perturbation was then used to perturb the initial conditions for the next integration interval [57, 97] and replaced X_{rand} in Eq. B.10 after the first time interval. Finally, after the orthonormalization at the end of each integration interval, the Lyapunov exponent for each perturbed state $\lambda^t_{\text{Lyap},i}$ over that interval was computed as [57]

$$\lambda^t_{\text{Lyap},i} = \frac{1}{T_{orth}} \ln \left[\frac{||\delta \boldsymbol{y'}_i(\boldsymbol{x}, t)||}{\epsilon_{\text{Lyap}}} \right], \tag{B.15}$$

with T_{orth} the number of time steps in an integration interval and the superscript t indicated the instance in time. Furthermore ϵ_{Lyap} equals the norm of the perturbation after the previous orthonormalization [57, 97]. Thus, the current perturbation is scaled by the perturbation after the previous orthonormalization and gives the exponential growth of the perturbations in that interval. Then, over the full integration time, the Lyapunov exponent $\lambda^t_{\text{Lyap},i}$ for each perturbed state was computed at each orthonormalization. At the end, the final Lyapunov exponents λ_{Lyap} were computed as the average over all orthonormalization [57, 97].

C Hyperparameters of deep learning and Hybrid-A reduced order models

In this appendix, the hyperparameters as obtained from the Grid search-Bayesian optimisation are presented for the pure deep learning and the Hybrid-A reduced order models. First, for each latent space the hyperparameters of the $Re = 20$ test case are shown, followed by the $Re = 34$ test case.

$Re = 20$

Table C1: Tables showing for $Re = 20$ the hyperparameters for each reservoir size N_{res} for the leaking rate α, Tikhonov factor β, input scaling σ_{in} and the spectral radius ρ. Here, the left table is for the pure deep learning model and the right table for the Hybrid-A model. Furthermore, the top row is for latent space size $N_{\text{lat}} = 16$, the middle row for $N_{\text{lat}} = 48$ and the bottom row for $N_{\text{lat}} = 80$.

N_{res}	α	β	σ_{in}	ρ
500	0.10	1.0	3.63	2.14
1000	0.10	1.0	3.63	2.14
1500	0.10	10^{-6}	3.63	2.14
2000	0.10	0.1	3.63	2.14
3000	0.18	0.0001	0.82	1.58
4000	0.18	0.0001	0.82	1.58
5000	0.18	0.0001	0.82	1.58

N_{res}	α	β	σ_{in}	ρ
500	0.10	1.0	3.63	2.14
1000	0.10	1.0	3.63	2.14
1500	0.10	1.0	3.63	2.14
2000	0.18	0.001	0.82	1.58
3000	0.10	0.0001	3.63	2.14
4000	0.10	0.1	3.63	2.14
5000	0.10	0.1	5.0	3.10

N_{res}	α	β	σ_{in}	ρ
500	0.22	1.0	0.37	0.22
1000	0.10	0.1	0.38	0.57
1500	0.10	0.1	3.63	2.14
2000	0.10	1.0	3.63	2.14
3000	0.10	1.0	3.81	1.85
4000	0.10	10^{-6}	3.62	2.14
5000	0.10	0.1	5.00	1.65

N_{res}	α	β	σ_{in}	ρ
500	0.18	1.0	0.82	1.58
1000	0.10	1.0	3.63	2.14
1500	0.12	1.0	4.40	1.10
2000	0.18	0.1	0.82	1.58
3000	0.18	10^{-6}	0.82	1.58
4000	0.10	10^{-6}	3.63	2.14
5000	0.18	1.0	0.82	1.58

N_{res}	α	β	σ_{in}	ρ
500	0.11	0.01	0.1	0.1
1000	0.49	0.1	0.1	0.85
1500	0.18	1.0	0.82	1.58
2000	0.10	0.1	0.1	0.61
3000	0.10	0.1	0.1	0.65
4000	0.18	10^{-5}	0.82	1.58
5000	0.10	1.0	3.63	2.14

N_{res}	α	β	σ_{in}	ρ
500	0.82	0.0001	1.07	0.79
1000	0.82	1.0	1.07	0.79
1500	0.46	0.0001	1.79	1.01
2000	0.18	0.0001	0.82	1.58
3000	0.10	1.0	0.1	0.1
4000	0.18	1.0	0.82	1.58
5000	0.10	1.0	0.61	0.10

$Re = 34$

Table C2: Tables showing for $Re = 20$ the hyperparameters for each reservoir size N_{res} for the leaking rate α, Tikhonov factor β, input scaling σ_{in} and the spectral radius ρ. Here, the left table is for the pure deep learning model and the right table for the Hybrid-A model. Furthermore, the top row is for latent space size $N_{\text{lat}} = 64$, the middle row for $N_{\text{lat}} = 192$ and the bottom row for $N_{\text{lat}} = 320$.

N_{res}	α	β	σ_{in}	ρ
500	0.21	0.01	0.01	4.27
1000	1.00	0.0001	0.01	5.00
1500	0.17	0.01	0.01	3.56
2000	0.26	0.01	0.01	0.01
3000	0.33	0.0001	0.01	5.00
4000	1.00	0.1	0.01	5.00
5000	0.1	0.001	0.01	1.74

N_{res}	α	β	σ_{in}	ρ
500	0.65	1.0	2.04	2.50
1000	1.00	1.0	3.04	3.02
1500	0.20	1.0	1.13	2.34
2000	0.17	1.0	1.38	1.01
3000	0.20	1.0	1.44	1.51
4000	0.18	1.0	0.74	1.52
5000	0.10	1.0	0.85	1.20

N_{res}	α	β	σ_{in}	ρ
500	0.1	0.1	0.01	3.28
1000	0.1	0.001	0.01	0.23
1500	0.14	0.01	0.01	5.00
2000	0.41	0.001	0.01	0.01
3000	0.60	0.01	0.01	0.01
4000	0.64	0.01	0.01	0.01
5000	0.95	0.01	0.01	0.01

N_{res}	α	β	σ_{in}	ρ
500	0.98	1.0	2.22	2.44
1000	0.67	1.0	1.93	2.41
1500	0.72	1.0	2.10	2.70
2000	1.00	0.1	0.01	2.61
3000	0.27	0.1	0.01	0.01
4000	0.59	0.1	0.01	0.01
5000	0.16	1.0	0.46	0.98

N_{res}	α	β	σ_{in}	ρ
500	0.10	10^{-5}	0.01	5.00
1000	0.13	0.0001	0.01	5.00
1500	0.30	0.001	0.01	0.05
2000	0.10	0.001	0.001	0.76
3000	0.17	0.0001	0.01	0.01
4000	1.00	0.0001	0.01	0.01
5000	1.00	0.0001	0.01	5.00

N_{res}	α	β	σ_{in}	ρ
500	0.82	1.0	0.01	5.00
1000	0.10	1.0	0.01	4.01
1500	1.00	1.0	0.01	5.00
2000	1.00	1.0	0.01	5.00
3000	1.00	1.0	0.01	2.58
4000	1.00	1.0	0.01	4.75
5000	1.00	1.0	0.01	5.00

D RMSE of mean flow at $Re = 20$ including $N_{\mathbf{res}} = 500$

In section 6.2, the long-term statistics of the knowledge-based, deep learning and the Hybrid-A models were presented. It was mentioned that for the $\langle u \rangle$ and $\langle v \rangle$ components of the flow field, the result for the deep learning model at $N_{\text{res}} = 500$ and $N_{\text{lat}} = 80$ was much larger compared to the other experiments. For that reason, the results including this data point are presented in Fig. D.2.

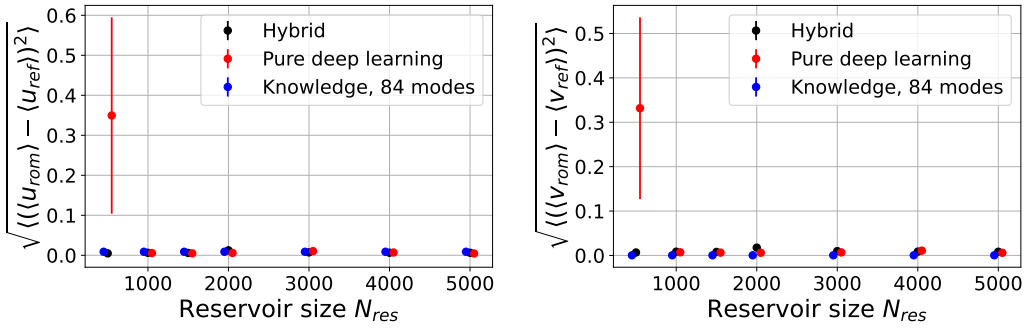
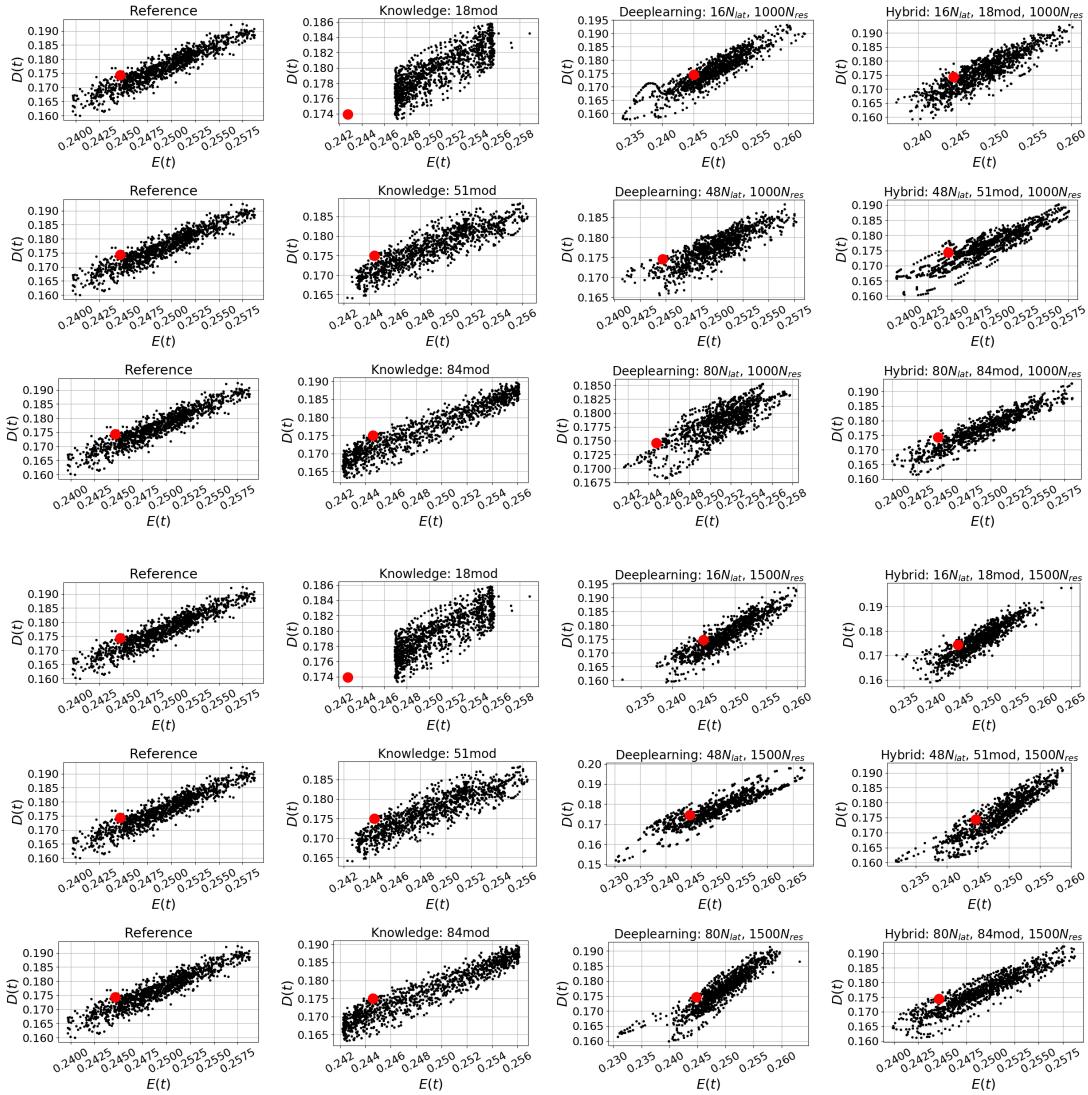
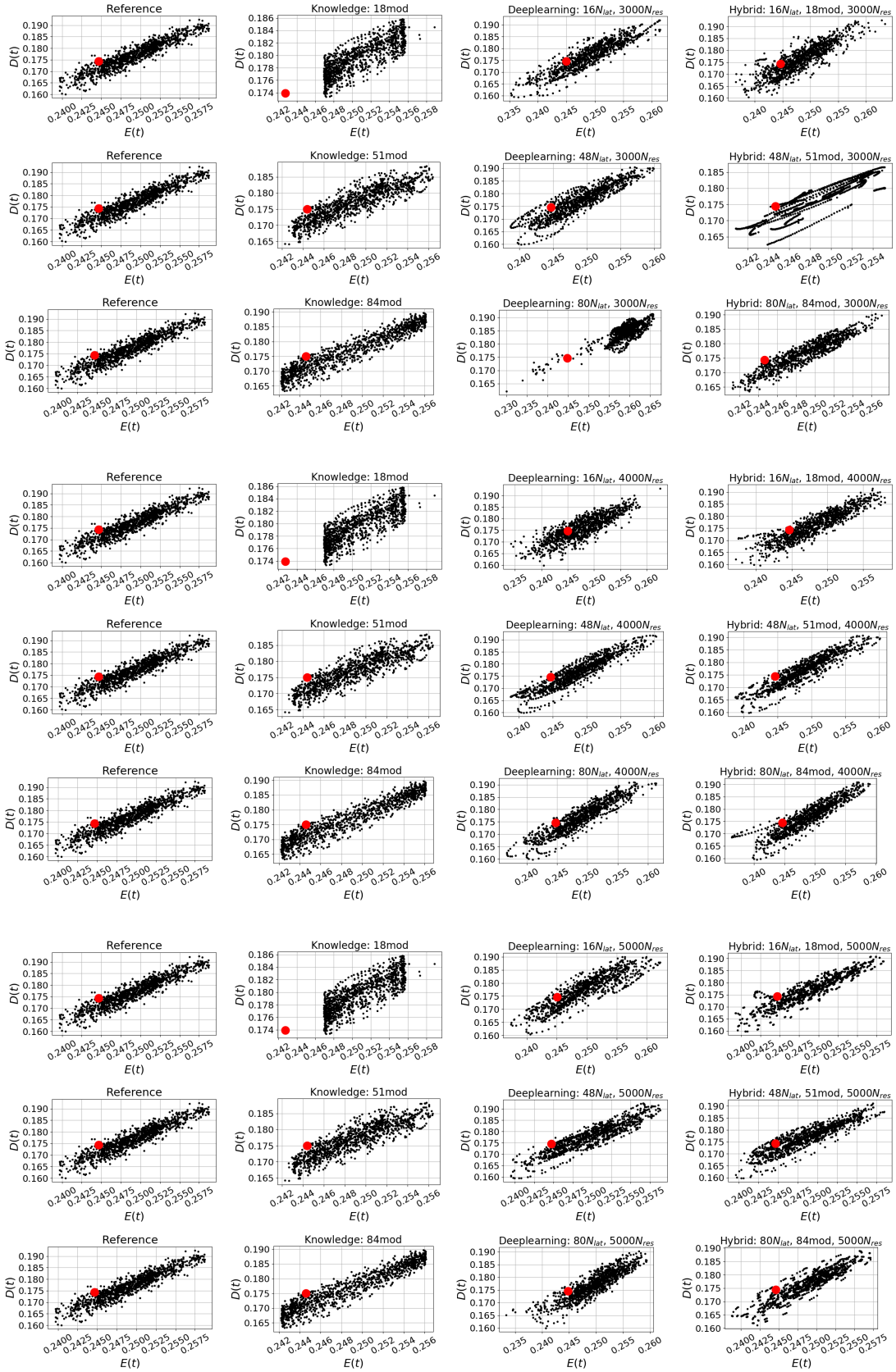


Figure D.2: The root mean square error of $\langle v \rangle$ and $\langle v \rangle$ of the flow at $Re = 20$ for $N_{\text{lat}} = 80$ including the data point for $N_{\text{res}} = 500$. Here, the left plot is for $\langle u \rangle$ and the right plot for $\langle v \rangle$. The knowledge-based and deep learning results are displaced slightly in the horizontal direction for clarity.

E $D(t)$ against $E(t)$, $Re = 20$

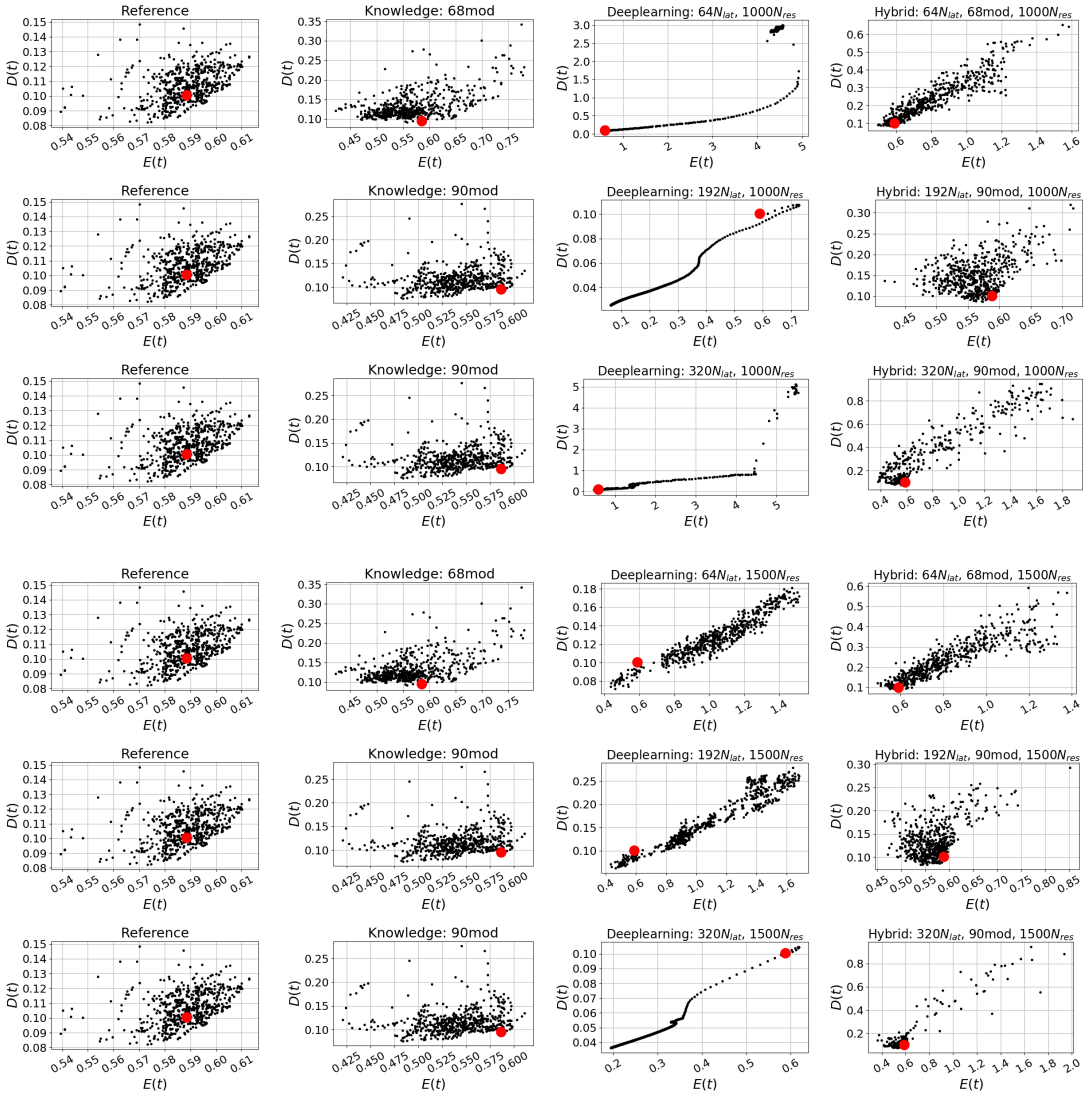
In this appendix, the results are shown for the phase space trajectories of the dissipation $D(t)$ and the kinetic energy energy $E(t)$ for $N_{\text{res}} = [1000, 1500, 3000, 4000, 5000]$ for $Re = 20$. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 16$, second row $N_{\text{lat}} = 48$ and the third row $N_{\text{lat}} = 80$, which is repeated every 3 rows. Furthermore, the model settings are also shown in the title of each plot.

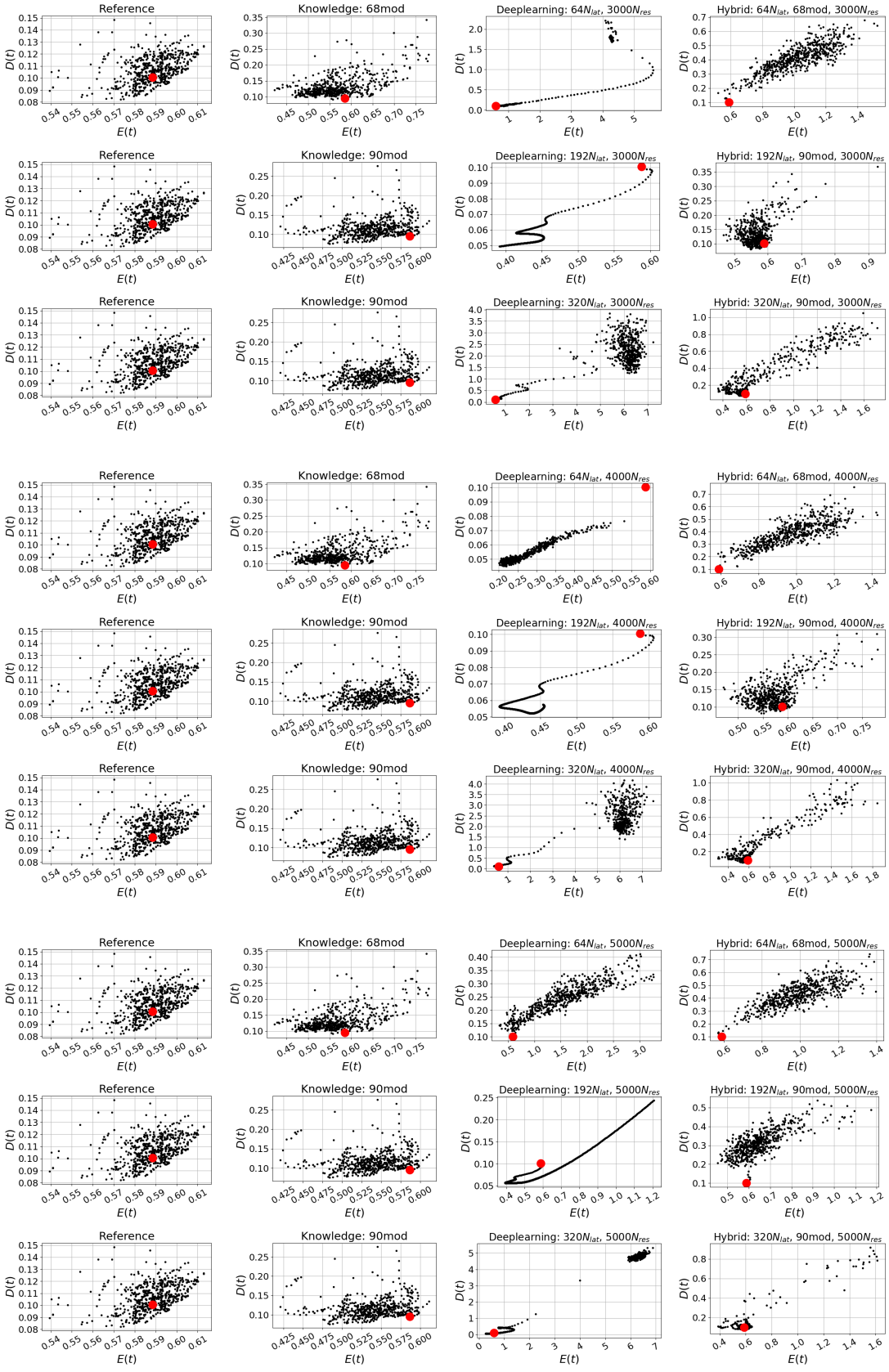



Reference
Knowledge: 18mod
Deeplearning: 16N_{lat}, 3000N_{res}
Hybrid: 16N_{lat}, 18mod, 3000N_{res}
Reference
Knowledge: 51mod
Deeplearning: 48N_{lat}, 3000N_{res}
Hybrid: 48N_{lat}, 51mod, 3000N_{res}
Reference
Knowledge: 84mod
Deeplearning: 80N_{lat}, 3000N_{res}
Hybrid: 80N_{lat}, 84mod, 3000N_{res}
Reference
Knowledge: 18mod
Deeplearning: 16N_{lat}, 4000N_{res}
Hybrid: 16N_{lat}, 18mod, 4000N_{res}
Reference
Knowledge: 51mod
Deeplearning: 48N_{lat}, 4000N_{res}
Hybrid: 48N_{lat}, 51mod, 4000N_{res}
Reference
Knowledge: 84mod
Deeplearning: 80N_{lat}, 4000N_{res}
Hybrid: 80N_{lat}, 84mod, 4000N_{res}
Reference
Knowledge: 18mod
Deeplearning: 16N_{lat}, 5000N_{res}
Hybrid: 16N_{lat}, 18mod, 5000N_{res}
Reference
Knowledge: 51mod
Deeplearning: 48N_{lat}, 5000N_{res}
Hybrid: 48N_{lat}, 51mod, 5000N_{res}
Reference
Knowledge: 84mod
Deeplearning: 80N_{lat}, 5000N_{res}
Hybrid: 80N_{lat}, 84mod, 5000N_{res}
$D(t)$
$E(t)$

F $D(t)$ against $E(t)$, $Re = 34$

Appendix showing the results for the phase space trajectories of the dissipation $D(t)$ and the kinetic energy energy $E(t)$ for $N_{\text{res}} = [1000, 1500, 3000, 4000, 5000]$ for $Re = 34$. Here, the first column represents the reference solution, the second column the knowledge-based, the third column the deep learning and finally the fourth column the the Hybrid-A model predictions. Furthermore, the first row represents $N_{\text{lat}} = 16$, second row $N_{\text{lat}} = 48$ and the third row $N_{\text{lat}} = 80$, which is repeated every 3 rows. Furthermore, the model settings are also shown in the title of each plot.




Reference
Knowledge: 68mod
Deeplearning: 64N_{lat}, 3000N_{res}
Hybrid: 64N_{lat}, 68mod, 3000N_{res}
Reference
Knowledge: 90mod
Deeplearning: 192N_{lat}, 3000N_{res}
Hybrid: 192N_{lat}, 90mod, 3000N_{res}
Reference
Knowledge: 90mod
Deeplearning: 320N_{lat}, 3000N_{res}
Hybrid: 320N_{lat}, 90mod, 3000N_{res}
Reference
Knowledge: 68mod
Deeplearning: 64N_{lat}, 4000N_{res}
Hybrid: 64N_{lat}, 68mod, 4000N_{res}
Reference
Knowledge: 90mod
Deeplearning: 192N_{lat}, 4000N_{res}
Hybrid: 192N_{lat}, 90mod, 4000N_{res}
Reference
Knowledge: 90mod
Deeplearning: 320N_{lat}, 4000N_{res}
Hybrid: 320N_{lat}, 90mod, 4000N_{res}
Reference
Knowledge: 68mod
Deeplearning: 64N_{lat}, 5000N_{res}
Hybrid: 64N_{lat}, 68mod, 5000N_{res}
Reference
Knowledge: 90mod
Deeplearning: 192N_{lat}, 5000N_{res}
Hybrid: 192N_{lat}, 90mod, 5000N_{res}
Reference
Knowledge: 90mod
Deeplearning: 320N_{lat}, 5000N_{res}
Hybrid: 320N_{lat}, 90mod, 5000N_{res}
$D(t)$
$E(t)$

G Architecture feed-forward neural network

In this appendix, the architecture of the feed-forward neural network of the Hybrid-FFNN is presented in Fig. G.3. Here, "Dense" represent a dense layer and "Dec" are the decoders of the multi-scale autoencoder. Furthermore, around 100,000 were non-trainable because in the training process it was not desired to change the parameters of the decoders.

Layer (type)	Output Shape	Param #	Connected to
input_1 (Input)	[(None, 640)]	0	[]
dense (Dense)	(None, 300)	192300	['input_1[0][0]']
dense_1 (Dense)	(None, 300)	90300	['dense[0][0]']
dense_2 (Dense)	(None, 300)	90300	['dense_1[0][0]']
dense_3 (Dense)	(None, 320)	96320	['dense_2[0][0]']
dense_4 (Dense)	(None, 320)	102720	['dense_3[0][0]']
reshape (Reshape)	(None, 8, 8, 5)	0	['dense_4[0][0]']
Dec_0 (Sequential)	(50, 66, 66, 2)	10988	['reshape[0][0]']
Dec_1 (Sequential)	(50, 66, 66, 2)	30380	['reshape[0][0]']
Dec_2 (Sequential)	(50, 66, 66, 2)	59468	['reshape[0][0]']
add (Add)	(None, 66, 66, 2)	0	['Dec_0[0][0]', 'Dec_1[0][0]', 'Dec_2[0][0]']

Total params: 672,776
Trainable params: 571,940
Non-trainable params: 100,836

Figure G.3: Image showing the Tensorflow architecture of the feed-forward neural network of the Hybrid-FFNN model.