Technische Universität Braunschweig

Studienarbeit No. 848

A Comparison of Reduced-Order Models for Wing Buffet Predictions

Anton Schreiber

Examiner: Dr.-Ing. André Bauknecht
Institute of Fluid Mechanics
Head of Institute: Prof. Dr.-Ing. David E. Rival
TU Braunschweig

Supervisor: Dr.-Ing Andre Weiner (TU Braunschweig)

Publication: December 2023

Affidavit

I, Anton Schreiber, declare that I have authored this thesis independently, that I have not used other than the declared sources and resources, and that I have explicitly marked all material that has been quoted either literally or by content from the used sources.

Abstract

The primary aim of this study is to train and evaluate a set of Reduced-Order Models (ROMs) for predicting upper-wing pressure distributions on a civil aircraft configuration. Using wind tunnel data recorded for the Airbus XRF-1 research configuration in the European Transonic Windtunnel (ETW), the ROMs integrate dimensionality reduction and time-evolution components. For dimensionality reduction, both Singular Value Decomposition (SVD) and a convolutional Variational Autoencoder (CNN-VAE) neural network are employed to reduce/encode Instationary Pressure Sensitive Paint (IPSP) data from a transonic buffet flow condition. Fully-Connected (FC) and Long Short-Term Memory (LSTM) neural networks are then applied to predict the evolution of the latent representation, which is subsequently reconstructed/decoded back to its original high-dimensional state.

SVD and CNN-VAE are trained on data from five flow conditions and tested on two unseen conditions. When comparing the power spectra of the reconstructed and experimental data, SVD demonstrates marginally better performance. Subsequently, FC and LSTM models are applied for the forward evolution of the reduced/latent representation for one flow condition, resulting in four evaluated ROMs. Among them, the CNN-VAE-LSTM model excels by accurately capturing buffet dynamics, encompassing both transient features and steady-state oscillations. SVD-based models tend to struggle with transient buffet behavior, as they predominantly learn steady-state dynamics. Additionally, the CNN-VAE-LSTM model was employed for an end-to-end training approach. The results suggest that it faces difficulties maintaining dynamics in predictions, demanding further investigation and optimization.

Contents

Nomenclature

Latin Symbols

$a(\cdot)$	activation function
$\mathbf{a}(t)$	mode coefficient
$\hat{\mathbf{a}}(f)$	Fast-Fourier-transformed signal
$\mathbf{A}$	mode coefficients
$\tilde{\mathbf{A}}$	truncated mode coefficients/ reduced data matrix
b	bias term
$\mathbf{b}$	bias vector
c	chord, $[m]$
c_p	pressure coefficient
$\bar{c}$	mean chord, $[m]$
$\mathbf{c}$	cell state
$\tilde{\mathbf{c}}$	candidate cell state
C	channel dimension,
$\mathbf{C}$	covariance matrix
$d(\cdot)$	decoder network
$d^*(\cdot)$	optimal decoder network
$e(\cdot)$	encoder network
$e^*(\cdot)$	optimal encoder network
f	frequency, $[1/s]$
$f_\theta(\cdot)$	model
$\mathbf{f}$	forget gate
$FFT(\cdot)$	Fast Fourier Transform
$g(\cdot)$	representation of advanced optimization techniques
$\mathbf{h}$	hidden state
H	height dimension
$\mathbf{i}$	input gate
$\mathbf{I}$	identity matrix
$L(\cdot)$	loss function
m	state dimension
$m(\cdot)$	transformation through neural network layer
Ma	Mach number
n	snapshot (time) dimension
$N(\cdot)$	normal distribution
N_f	number of features
N_{neu}	number of neurons
$\mathbf{o}$	output gate

r	truncation rank/ bottleneck layer size
R^2	Coefficient of Determination (R-squared)
Re	Reynolds number
s	stride
$sigm(\cdot)$	sigmoid activation function
std	standard deviation
St	Strouhal number
$tanh(\cdot)$	hyperbolic tangent activation function
$\mathbf{u}$	left singular vector
U	fluid velocity, $[m/s]$
$\mathbf{U}$	left singular vectors
$\tilde{\mathbf{U}}$	truncated left singular vectors
var	variance
$\mathbf{v}$	right singular vector
$\mathbf{V}$	right singular vectors
$\tilde{\mathbf{V}}$	truncated right singular vectors
w	weight
W	width dimension
$\mathbf{W}$	weight matrix
x	feature
$\mathbf{x}$	feature vector
$\mathbf{X}$	data matrix
$\tilde{\mathbf{X}}$	approximated data matrix
$\bar{\mathbf{X}}$	data matrix mean
y	label
$\mathbf{y}$	label vector
$\hat{\mathbf{y}}$	model prediction
z	weighted sum (activation)
$\mathbf{z}$	vector of weighted sums (activations)

Greek Symbols

α	angle of attack, $[°]$
χ	sampled vector
ζ	random variable
η	spanwise position
θ	model parameters
θ^*	optimal model parameters
λ	learning rate
ν	kinematic viscosity, $[m^2/s]$
ω	reduced frequency
ϕ	sweep angle, $[°]$
σ	singular value
$\mathbf{\Sigma}$	singular values
$\tilde{\mathbf{\Sigma}}$	truncated singular values
τ	dimensionless time

Indices

0	initial; starting
θ	model parameters
∞	undisturbed external flow; incoming flow
f	forget
i, j, o	(tensor) indexing
in	input
k	kernel (filter)
l	layer
loc	local
n	step
$normal$	normal component
out	output
p	pressure
$reconstr.$	reconstructed
t	timestep

Acronyms

AE	Autoencoder
ARMA	AutoRegressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
CNN	Convolutional Neural Network
DL	Deep Learning
DMD	Dynamic Mode Decomposition
E2E	End-to-End
ETW	European Transonic Windtunnel
FC	Fully-Connected
GP	Galerkin Projection
GPU	Graphics Processing Unit
GPT	Generative Pre-Trained Transformer
IPSP	Instationary Pressure-Sensitive Paint
KL	Kulback-Leibler
LCO	Limit Cycle Oscillations
LSTM	Long Short-Term Memory
ML	Machine Learning
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NODE	Neural Ordinary Differential Equation
PCA	Principal Component Analysis
POD	Proper Orthogonal Decomposition
PSD	Power Spectral Density
RNN	Recurrent Neural Network
ROM	Reduced Order Model(ing)
SVD	Singular Value Decomposition
TCAN	Temporal Convolutional Attention-based Network
TFN	Through Flow Nacelle

$UHBR$	Ultra High Bypass Ratio
VAE	Variational Autoencoder
VAR	Vector Autoregression
XRF	eXternal Research Forum

Chapter 1

Introduction

1.1 Motivation

For decades, scientists and engineers have been striving to comprehend the complex, multi-scale dynamics of aerodynamic configurations [4]. However, certain phenomena are still not fully understood due to their highly nonlinear nature. One such phenomenon is the transonic buffet, involving an interaction of the shock-wave with a boundary layer separation that can induce critical structural vibration in aircraft (known as buffeting), thus posing significant challenges for engineers [8]. Estimating buffet conditions within the flight envelope is crucial for designing safer and more efficient aircraft.

Over the last few years, Machine Learning (ML) algorithms have increasingly been utilized in fluid mechanics applications due to their ability to extract informative features from data. This is particularly beneficial for the wide range of spatial and temporal features of fluid flows, which require more meaningful representations. The modular and agile modeling framework of ML can be adapted to address a broad spectrum of fluid mechanics problems, paving the way for new applications of dimensionality reduction and Reduced-Order Modeling (ROM) in general [4, 10].

The research unit FOR 2895 [19] conducted Instationary Pressure-Sensitive Paint (IPSP) measurements on the Airbus XRF-1 research configuration. These measurements were carried out within the European Transonic Windtunnel (ETW) and provided time-resolved pressure distributions over the upper wing during transonic buffet conditions. However, the measurements come with considerable cost and do not provide well-defined outlines for buffet conditions. Therefore, the use of ROMs becomes valuable to reduce the significant cost linked to further high-fidelity investigations. With these models, intermediate flow conditions can be explored and and an estimate of buffet onset, in terms of both angle of attack and Mach number, can be formulated.

1.2 State of the Art

The auto-regressive prediction of sequential data has a wide range of applications. To forecast time series such as airline passengers or sales, statistical models like ARMA (Auto-regressive Moving Average), ARIMA (Auto-regressive Integrated Moving Average), and VAR (Vector Autoregression) are often used [25, 39, 40]. ARMA models use moving average components, in which the current value linearly depends on the time series mean, a current error term, and past error terms. ARIMA models incorporate differencing, which is an approach to handling non-stationary series by transforming them into stationary ones. VAR models consider the interdependencies between multiple variables to forecast multiple time series variables simultaneously. An often used library in Python to implement such models is the *statsmodels* library.

Statistical models encounter significant limitations when confronted with datasets containing numerous features, intricate nonlinear relationships, and a data volume surpassing a certain threshold. Deep learning demonstrates its effectiveness in addressing complex data in these scenarios, proving its capability to handle such challenges with ease [25]. Van den Oord et al. [23] proposed the audio generation model WaveNet, which is used for text-to-speech, speech enhancement, and more. The auto-regressive benchmark model consists of stacked layers of dilated causal convolutions, i.e. by introducing holes within the convolutional kernel, it is possible to achieve a larger receptive field while preserving a compact kernel size.

In the field of language modeling, ChatGPT is probably the most popular application, currently based on GPT-4 [24], a Generative Pre-trained Transformer (GPT) model from OpenAI. An additional model in this field is the Temporal Convolutional Attention-based Network (TCAN) [13]. TCAN was able to outperform prior state-of-the-art models based on Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) architectures and also the previous ChatGPT model GPT-2 [29], while having 10 times fewer model parameters. Pravallika et al. [28] used a Long Short-Term Memory (LSTM) [14] architecture to predict the sea surface temperature of the Indian Ocean. In light of the work's results, the authors consider employing a convolutional LSTM as a more sophisticated model for future work. For the solar power forecast of a photovoltaic plant in Konya, Turkey, Tosun et al. [38] trained three models, namely a wide LSTM, a deep LSTM, and a WaveNet model, on six months of power generation data from a plant at a resolution of 15 minutes. The researchers could show that these architectures outperform classical methods like regression but also fully-connected neural networks.

In the field of fluid mechanics, models often have to deal with high-dimensional, spatiotemporal data, obtained from experiments or numerical simulations. Based on the latter, Maulik et al. [21] investigated time series learning of latent-space dynamics for an advection-dominated problem given by the viscous Burgers equation. The problems were projected onto a latent space utilizing Singular Value Decomposition (SVD) where time evolution is then learned by Neural Ordinary Differential Equation (NODEs) and LSTM networks. The study showed that both methods could provide stable 400 timestep predictions in contrast to the Galerkin Projection (GP) method. Eivazi et al. [10] proposed a nonlinear reduced-order modeling approach, combining an autoencoder (AE) with an LSTM network, to handle complex unsteady fluid flows. In this method, the AE serves as a substitute for SVD. For future state prediction, an initial sequence was fed through the encoder and utilized as input for the LSTM network, enabling auto-regressive prediction within the latent space. Subsequently, the predicted time steps were encoded and mapped back to the original space. The AE and LSTM were trained independently and compared against the widely recognized Dynamic Mode Decomposition (DMD) method and a newly developed ROM incorporating POD (Proper Orthogonal Decomposition) modes and the LSTM network. Notably, the findings demonstrated that the autoencoder-LSTM model achieved superior performance, particularly in more complex test scenarios, outperforming the other considered methods. In a related study by Solera-Rico et al. [33], the authors introduced a novel approach combining a β-variational-autoencoder (β-VAE) and a transformer model for numerical fluid flow modeling. The performance of the AE was compared to POD, revealing the superior performance of the AE, particularly in complex test cases. Additionally, the prediction accuracy of the transformer network was compared to that of an LSTM architecture. Interestingly, the authors found that only the transformer network adequately captured and learned the correlations among the temporal coefficients, highlighting its superiority in capturing temporal dependencies.

Based on experimental data of the Airbus XRF-1 model, two related follow-up studies [44, 45] dealt with the auto-regressive prediction of buffet pressure fields. Zahn et al. [44] explored the application of an LSTM-based ROM, which was trained on a specific flow condition and fixed sensor position, to predict buffet pressure characteristics at various sensor positions and angles of attack. The findings of the study indicated that LSTMs successfully captured the buffet frequency content. Although a slight shift in the amplitude between the predictions and the experimental data was visible, there is an overall precise agreement. In a related investigation by Zahn et al. [45], they employed a convolutional-VAE-LSTM (CNN-VAE) on IPSP data. The convolutional VAE was used for reducing the dimensionality of the flow field data, while the LSTM was employed to predict the temporal evolution of the pressure distributions. Utilizing an initial sequence of 32 encoded timesteps, the proposed model achieved stable predictions up to timestep 200, accurately capturing the buffet flow physics.

1.3 Approach

The objective of this study is to implement, train, and compare a set of ROMs for the prediction of upper-wing pressure distributions based on IPSP data recorded at buffet conditions. To achieve this, the data must undergo a process of dimensionality reduction to lower-dimensional states, followed by a forward evolution in time and subsequent reconstruction to its original state. ROMs are employed incorporating the dimensionality reduction techniques SVD and CNN-VAE, along with time-evolution frameworks involving FC and LSTM neural networks. The temporal evolution of the reduced state follows an auto-regressive approach, in which predictions are used to make successive predictions, with a focus on achieving stability for long-term forecasts. While the traditional sequential training approach is followed for most ROMs, an end-to-end training strategy is applied for the best-performing model.

The mentioned objectives are structured within the framework of the present work. After introducing the buffet phenomenon in chapter 2, chapter 3 derives the mathematical fundamentals of the employed ROM methodologies and metrics. This includes the presentation of SVD and CNN-VAE as dimensionality reduction techniques, an introduction to Deep Learning (DL) with neural networks, and an overview of FC, CNN, and LSTM neural networks. Additionally, this chapter includes a section dedicated to metrics that serve as the foundation for the optimization and evaluation of the ROMs.

Focusing on model optimization, the subsequent chapter 4 provides details about the employed training methodologies. It offers comprehensive insights into the frameworks adopted, the range of parameters utilized, and the parameter studies conducted, aimed toward enhancing model performance. Chapter 5 involves applying all developed ROMs to previously unseen data, assessing their capability for making stable long-term predictions while accurately reproducing buffet flow physics. A detailed examination of various metrics supports the evaluations. Finally, the study's findings are summarized and discussed in Chapter 6, providing implications for future work.

Chapter 2

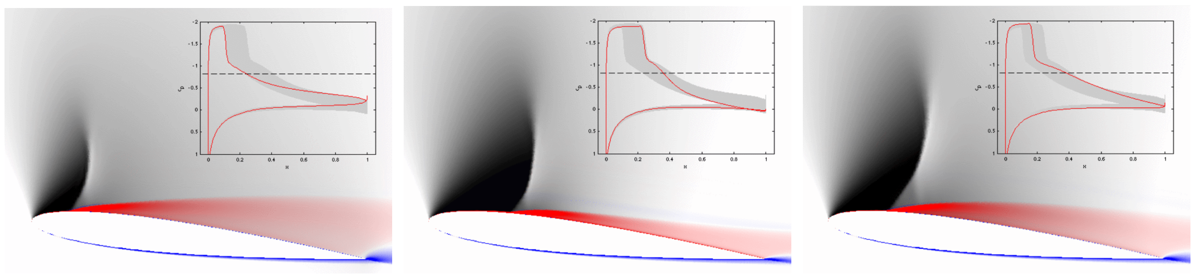
Three-Dimensional Transonic Buffet

The transonic buffet represents an interaction between shock-wave oscillation and boundary layer separation on aircraft wings. It is a complex phenomenon that occurs during edge-of-the-envelope transonic flight conditions. Despite advances in experimental and numerical investigations, it is still an unresolved flow phenomenon due to a lack of fundamental understanding. This section aims to briefly elaborate on the current state of knowledge regarding this phenomenon.

2.1 Physical Mechanism

The buffet occurs at transonic flight conditions which are essentially characterized by high subsonic cruise speeds and therefore in the operating range of modern aircraft. At these conditions, the flow around the airfoil locally exceeds the speed of sound and forms supersonic flow regions. At the downstream border of these regions - the shock wave - the flow decelerates to subsonic speeds. With increasing Mach number (Ma) and/ or angle of attack (α), the shock intensifies while moving aft over the airfoil, eventually inducing a massive boundary layer separation at the shock that generally goes along with an aerodynamic resonance [6]. In this case, the flow can become highly unsteady, introducing oscillations of the flow variables and load coefficients, affecting aerodynamic performance. These variations indicate buffet [8]. To describe buffet in the frequency domain, the Strouhal number $St = fc/U$, with frequency f, chord c, and flow speed U, is commonly employed as it characterizes oscillating flows.

In fig. 2.1, the movement of the shock wave and its influence on the shock-induced separation can be observed for a generic airfoil following 2D numerical simulations [20]. It becomes evident that, while the intensity of the shock remains almost constant, the separation layer as well as the supersonic flow region strongly vary in shape and size over time.



(a) **(b)** **(c)**

Figure 2.1: Airfoil buffet for three consecutive timesteps, taken from [20]

2.2 Three-Dimensional Behavior

While the causes of shock-separation interactions are well understood, understanding the 3D dynamics of buffet phenomena presents a significant challenge. However, recent research progress has led to the development of techniques to tackle more complex flow fields. As highlighted in various studies, there are fundamental flow physical differences between 2D airfoils, exhibiting harmonic shock motion at a specific frequency, and 3D swept wings, exhibiting more complex broadband dynamics. Consequently, the research focus has shifted towards the more practical aspects of three-dimensional swept wings [11, 36, 37].

According to [7, 16, 26, 35], the general 3D buffet behavior on a classical swept wing is elaborated in the following. In fig. 2.2, the schematic shock-separation-interaction is illustrated for the NASA (National Aeronautics and Space Administration) Common Research Model. Depicted are the time-averaged locations of the shock at different angles of attack with $Ma = 0.85$, taken from [35]. Starting from the inboard leading edge, the shock foot runs aft in spanwise direction until it converges with a second shock foot at $\eta \approx 0.25$ that originates at approximately $60\%c$ in the wing root. This characteristic shock appearance is typically denoted as lambda-shock.

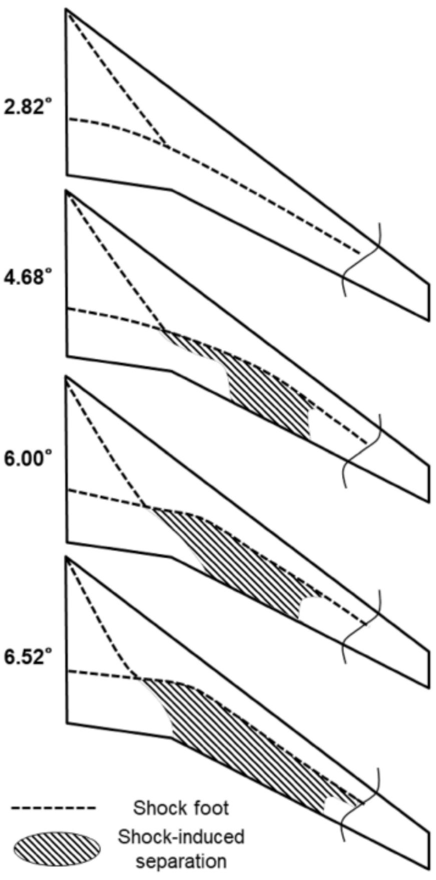


Figure 2.2: Schematic shock and separation behavior on a 3D swept wing, taken from [35]

As depicted in the illustration, the region of particular interest for buffet investigations is located between the previously mentioned point of convergence and the vicinity of the wingtip. This area is particularly significant as most of the shock movement and shock-induced separation phenomena occur there. As the angle of attack increases, there is typically a corresponding decrease in pressure on the upper-wing side. Starting with smaller angles of attack, pressure and shock begin to oscillate on the main wing at buffet conditions. Small areas of shock-induced separation and trailing edge separation occur. This behavior is a 2D buffet instability and occurs at a constant frequency, independent of the sweep angle [26]. Shock-induced separation becomes highly possible when the shock normal Mach number[1] is > 1.2 [35]. When further increasing the angle of attack, the separated flow regions grow large over the main wing with large spatial amplitude shock oscillations.

Besides the 2D buffet instability, a second dominant behavior can be observed, denoted as buffet cells. For swept wings, pressure and shock oscillations propagate in spanwise direction toward the wingtip as cellular patterns at a frequency proportional to the sweep angle. The frequency of the outboard convection increases with the sweep angle which explains why the buffet on conventional aircraft wings exhibits a higher frequency than the 2D buffet under the same flow conditions [26]. Furthermore, the spanwise shock front oscillation goes along with periodic boundary layer separation. Multiple studies suggest that buffet cells are linked to the generation of pressure perturbations in the lambda-shock region [16, 27].

[1] $Ma_{loc,normal} = Ma_{loc} \times \cos\phi$, with local Mach number Ma_{loc} and sweep angle ϕ

2.3 Buffeting and its effects

While buffet is a purely aerodynamic phenomenon, the aircraft's performance, stability, and safety are influenced by the aeroelastic structural response called buffeting. The large fluctuations in flow field characteristics can result in critical vibrations due to the similarity between the reduced frequency[2] of the buffet induced oscillation of the wing and its low-frequency structural modes. This similarity can induce Limit Cycle Oscillations (LCOs), which adversely affect the handling quality and fatigue life of the aircraft [11] or lead to destructive outcomes. Moreover, buffeting limits the aircraft flight envelope due to design standards demanding a margin of 30% on the cruise lift coefficient [8].

[2]The reduced frequency ω is a fundamental aeroelastic parameter used to describe the link between airfoil oscillation frequency and flow velocity: $\omega = fc/2U$ with frequency f, chord c and flow velocity U

Chapter 3

Reduced Order Modeling

In fluid dynamics, high-fidelity data of flow fields can be computed with CFD simulations, or gathered by carrying out experiments - both at significant cost and/ or time expenditure. Therefore, ROMs are utilized to represent a high-dimensional system in terms of a low-dimensional one. By balancing accuracy and efficiency, these lower-order models are still able to capture nonlinearities sufficiently while reducing the (computational) costs significantly. In the present work, hybrid ROMs are employed by combining dimensionality reduction techniques (3.1) with deep learning-based prediction models (3.2). Additionally, different metrics are applied for the training, optimization, and evaluation of such models (3.3). In the following, the fundamentals of the separate ROM components and different metrics will be derived.

3.1 Dimensionality Reduction

Dimensionality reduction for flow modeling is based on extracting dominant patterns and flow feature relationships. If these prerequisites are fulfilled, techniques like SVD or CNN-VAE neural networks are utilized to encode information by projecting the full-order space onto a reduced (or latent) space, where the dynamics are described compactly and efficiently. When working with supervised learning algorithms in DL, identifying a lower-dimensional representation of immensely large datasets can be viewed as a pre-processing step [4]. The subsequent sections will provide detailed explanations of how dimensionality reduction can be accomplished using SVD and CNN-VAEs to preprocess data prior to its utilization in time-evolution models, that evolve the reduced state in time.

3.1.1 Singular Value Decomposition (SVD)

The SVD is a widely used modal decomposition technique in linear algebra. It involves factorizing a given matrix into three separate matrices. The SVD provides a powerful tool for analyzing and understanding the structure, properties, and relationships within a given matrix. The derivations in this section are based on [1, 3, 34, 41]

Let $\mathbf{X}$ be any $\mathbb{R}^{m \times n}$ data matrix

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_n} \\ | & | & & | \end{bmatrix}. \tag{3.1}$$

In the context of this work, it consists of time-series data, where each column represents a specific timestep commonly referred to as "snapshot". Each snapshot has a state dimension m, which is much larger than the number of snapshots n for many systems. The data matrix can be decomposed as:

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T, \tag{3.2}$$

where

$\mathbf{U} \in \mathbb{R}^{m \times m}$	– left singular vectors
$\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$	– singular values
$\mathbf{V}^T \in \mathbb{R}^{n \times n}$	– right singular vectors

U and **V** are both orthonormal matrices, which means that, for the full SVD, the transpose is identical to the inverse of the matrix, and the product of the matrix with itself in either the transposed or inversed form leads to the identity matrix **I**:

$$\mathbf{U}^T = \mathbf{U}^{-1} \tag{3.3}$$

$$\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I} \tag{3.4}$$

The columns of **U** and **V** hold the left and right singular vectors, respectively, in order from most to least important concerning variance. These singular vectors correspond to the orthogonal modes into which the data matrix is decomposed and are also the eigenvectors of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$, respectively. Depending on the area of application, the singular vectors are called Proper Orthogonal Decomposition (POD) modes or principal components in the Principal Component Analysis (PCA).

The matrix $\boldsymbol{\Sigma}$ is a diagonal matrix with the singular values positioned on the main diagonal in descending order: $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n > 0$, considering that $m \geq n$. This particular arrangement ensures that $\boldsymbol{\Sigma}$ can have a maximum of n non-zero singular values, allowing for a more compact but exact representation of **X**, a concept known as the Economy SVD. To explain the amount of variance that is explained by a single mode, a connection can be drawn between $\boldsymbol{\Sigma}$ and the covariance matrix **C** of the data matrix:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T, \tag{3.5}$$

and with equation 3.2

$$\mathbf{C} = \frac{1}{n-1}\mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T. \tag{3.6}$$

Equation 3.6 provides a connection between the singular values and variances in **C**, scaled by the number of snapshots n, enabling the quantification of the variance explained by each mode. When expressing the relative variance of a single mode compared to the overall variance in the dataset $\sigma_i^2 / \sum_j \sigma_j^2$, this leads to values in the range $[0, 1]$. Each of the $\sigma_i \mathbf{u_i} \mathbf{v_i}^\mathbf{T}$ is a $(m \times n)$ matrix and contributes to the reconstruction of the data matrix based on the value of σ_i. Specifically, the first mode $\sigma_1 \mathbf{u}_1 \mathbf{v}_\mathbf{1}^\mathbf{T}$ contributes the most to variance reconstruction as σ_1 represents the largest singular value.

While σ_i represents the importance of each mode, the left singular vectors $\mathbf{u}_i$ capture the spatially variant but time-invariant aspect of the mode. On the other hand, the right singular vectors $\mathbf{v}_i$ capture the spatially invariant but time-variant part of the mode. Therefore, $\mathbf{u}_i$ represents the spatial pattern, while $\mathbf{v}_i$ indicates the extent to which this spatial pattern is dominant over time. This temporal variation is contained within the matrix of mode coefficients **A**:

$$\mathbf{A} = \boldsymbol{\Sigma}\mathbf{V}^T. \tag{3.7}$$

For dimensionality reduction, the goal is to reduce and reconstruct a data matrix with fewer memory requirements and without losing the inherent variance of the data. This optimization problem can be stated as:

$$\underset{\tilde{\mathbf{X}}, s.t. rank(\tilde{\mathbf{X}})=r}{\arg\min} \quad \|\mathbf{X} - \tilde{\mathbf{X}}\|_F = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T, \tag{3.8}$$

where

$\tilde{\mathbf{U}} \in \mathbb{R}^{m\times r}$	– truncated left singular vectors
$\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{r\times r}$	– truncated singular values
$\tilde{\mathbf{V}}^T \in \mathbb{R}^{r\times r}$	– truncated right singular vectors
$\|\cdot\|_F$	– Frobenius norm

Since most singular vectors $\mathbf{u_i}$ do not contribute much to the variance, they are discarded. Based on different metrics or algorithms such as optimal hard threshold, the so-called truncation rank r can be identified and leads to the approximation of the data matrix $\tilde{\mathbf{X}}$.

In this study, the SVD is computed based on a subset of the data, and its performance is evaluated using a separate, unseen subset. These data subsets have a consistent number of state dimensions m, while they can vary in the number of snapshots n. Different datasets are projected onto the latent space through the precomputed SVD to effectively train and apply time-evolution models. This projection is followed by a reconstruction step that restores the dataset to its original, full-dimensional space. Hence, for any given dataset $\mathbf{X}$, reduction is achieved by performing a matrix multiplication with the transpose of the previously computed left singular vectors denoted as $\tilde{\mathbf{U}}$ (as derived from equation 3.2):

$$\tilde{\mathbf{U}}^T\mathbf{X} = \tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T = \tilde{\mathbf{A}} \tag{3.9}$$

This operation results in the reduced dataset $\tilde{\mathbf{A}} \in \mathbb{R}^{r\times n}$ that resides in the latent space. To reconstruct $\tilde{\mathbf{A}}$ to its original shape $\mathbb{R}^{m\times n}$, the left singular vectors are multiplied again to the left side of equation 3.9:

$$\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{X} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T = \tilde{\mathbf{U}}\tilde{\mathbf{A}} = \mathbf{X}_{reconstr.} \tag{3.10}$$

3.1.2 Convolutional Variational Autoencoder (CNN-VAE)

AEs are neural networks that can be used for dimensionality reduction purposes. They employ multiple non-linear transformations to compress high-dimensional data into a lower-dimensional latent space. VAEs are a specific type of AE that incorporates a probabilistic approach, allowing for more robust data compression in the latent space.

Standard Autoencoder

There are two key components in the architecture of an AE: the encoder network denoted as e, and the decoder network denoted as d. The role of the encoder network is to map the initial high-dimensional data $\mathbf{x}$ from $\mathbb{R}^m$ to the lower-dimensional latent space $\mathbb{R}^r$ using $e(\mathbf{x})$, where $m > r$. In the context of neural networks, the latent space is commonly referred to as the bottleneck layer, characterized by r neurons. This layer creates a bottleneck for the flow of information during the encoding process. Unlike SVD, which linearly decomposes data, AEs can utilize nonlinear activation functions to develop nonlinear embeddings. This characteristic enables AEs to generate more compact and expressive representations of the data with equal latent space dimension [4]. The decoder reverses this process by decompressing the data back to the original $\mathbb{R}^m$ space with $d(e(\mathbf{x}))$ [31]. The information flow of a standard AE is illustrated in fig. 3.1.

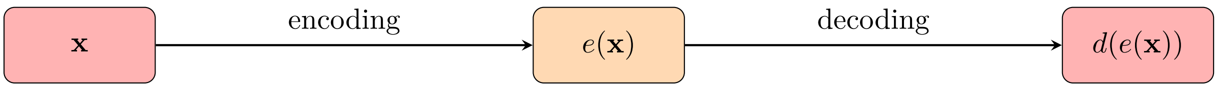


Figure 3.1: Standard autoencoder information flow, based on [31]

The objective of the optimization problem is to find an encoder and decoder pair (e_θ, d_θ) that minimizes the information loss during compression and maximizes the accuracy of the reconstruction, which can be written as [31]:

$$(e_\theta, d_\theta) = \underset{\theta}{\arg\min}\, \epsilon(\mathbf{x}, d(e(\mathbf{x}))) \tag{3.11}$$

where

θ	– autoencoder parameters (weights)
$\epsilon(\mathbf{x}, d(e(\mathbf{x})))$	– reconstruction error between input $\mathbf{x}$ and encoded-decoded $\mathbf{x}$

Typically, the chosen metric to measure the reconstruction error is the Mean-Squarred-Error (MSE), introduced in section 3.3. The optimization process is iterative and conducted through backpropagation and gradient descent (as derived in section 3.2.1), like for most neural networks. In each iteration, the input data $\mathbf{x}$ is reduced and reconstructed by the AE and compared against the original $\mathbf{x}$ to compute a gradient for the model parameters by nonlinear optimization methods, such as backpropagation [4]. However, the search for solving the optimization problem in equation 3.11 is achieved with gradient descent, which is the algorithm that updates the model parameters according to the computed gradient. That way, one solution (e_θ, d_θ) can be obtained [31].

The main degrees of freedom AE architectures can have are the bottleneck layer size, which corresponds to the dimension of the latent space, and the depth of the network, which refers to the number of transformation layers - in this case, convolutional layers (refer to section 3.2.2) - between the input and output. Convolutional layers apply a set of learnable kernels (or filters) to an input, typically images. By sliding these filters across the input, patterns, and features can be extracted in a step-by-step manner. The bottleneck layer size and the latent space dimension collectively determine the level of data compression and the quality of compression achieved. While AEs have the ability to reconstruct data without loss when provided with a large number of transformation layers for compression and decompression, this trade-off often leads to a loss of regularity in the compressed data, i.e. the usability of the data in the latent space becomes limited. In such scenarios, generating new data in the latent space and passing it through the decoder network may lead to meaningless results due to the overfitting of the decoder network. Overfitting occurs when the decoder network becomes too specialized for the training data, resulting in the generation of meaningless or unrealistic outputs. This happens since there is no regularisation in the optimization problem in equation 3.11, enforcing the decoder network to maintain structure in the latent space [31].

Regularisation with Variational Autoencoders

Unlike standard AEs, VAEs address this issue of overfitting by introducing a probabilistic framework. For training VAEs, the encoding-decoding process is adjusted so that an encoded input $\mathbf{x}$ is no longer represented as a deterministic vector in the latent space but as a probability distribution over the latent space $N(\mu_\mathbf{x}, std_\mathbf{x})$ with a mean vector and a standard deviation (std) vector. During decoding, a vector χ is sampled from this latent space distribution $\chi \sim N(\mu_\mathbf{x}, std_\mathbf{x})$, and the reconstruction error is backpropagated through the network. This process is very similar to standard AEs, with one exception: the random sampling node blocks the backpropagation and needs to be adjusted. Here, the so-called reparametrization trick is used that introduces a random variable ζ representing a standard normal distribution $N(0,1)$ with zero mean and a standard deviation of 1 so that the sampled vector can be expressed as $\chi = \mu_\mathbf{x} + std_\mathbf{x}\zeta$ [31]. How information is fed through the VAE is illustrated in fig. 3.2:

$$\begin{tikzcd}\boxed{\mathbf{x}} \arrow[r, "\text{encoding}"] & \boxed{N(\mu_\mathbf{x}, \sigma_\mathbf{x})} \arrow[r, "\text{sampling}"] & \boxed{\chi = \mu_\mathbf{x} + std_\mathbf{x}\zeta} \arrow[r, "\text{decoding}"] & \boxed{d(\chi)}\end{tikzcd}$$

Figure 3.2: Variational autoencoder information flow, based on [31]

Therefore, the optimization problem is extended by a regularisation term, namely the Kullback-Leibler (KL) divergence. The KL divergence measures the difference in information represented by two distributions [9] and by optimizing for low values it ensures that the probability distributions returned by the encoder are close to standard normal distributions. Accordingly, the optimization problem changes to:

$$(e_\theta, d_\theta) = \arg\min_\theta \epsilon(\mathbf{x}, d(\chi)) + \mathrm{KL}(N(\mu_\mathbf{x}, std_\mathbf{x}), N(0,1)), \tag{3.12}$$

and ensures a compromise between a performant and regularised encoding-decoding process.

3.2 Deep Learning (DL)

DL is a subset of machine learning that focuses on algorithms capable of learning data representations with multiple layers of abstraction, enabling informed decision-making based on these learned features. DL involves combining various non-linear components, which represent algorithms, into a computational model designed to handle specific types of data [18]. As stated in the universal approximation theorem proposed by Hornik et al. [15], a sufficiently large and complex DL model has the capacity to approximate any function. This characteristic makes DL models extensively utilized due to their effectiveness in understanding the intricate features of high-dimensional data [30].

This study focuses on the application of supervised learning algorithms, in which labeled datasets are used, providing examples of inputs along with corresponding desired model outputs [4]. Following a derivation of the key concepts and training methods for neural networks, two specific network architectures are presented: the CNN and the LSTM neural network.

3.2.1 Neural Networks

The traditional representation of artificial neural networks, as illustrated in Figure 3.3, takes inspiration from the structural arrangement of biological neuronal networks and their mathematical representation. In the brain, such networks comprise hierarchical layers of neurons (nerve cells) responsible for processing visual stimuli and other types of information. Neural networks consist of an input layer that receives data (features) and an output layer that generates predictions. In the context of supervised learning, the fundamental process involves comparing these predictions to the ground truth (labels), which forms the basis of the learning process [2, 3, 4].

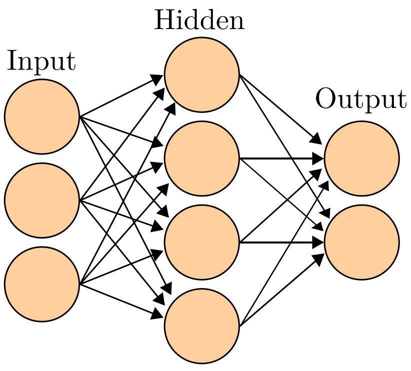


Figure 3.3: Artificial neural network by Colin M.L. Burnett, CC BY 4.0

Following the terminology introduced in [2] and [41], each neuron in the input layer of the network corresponds to a feature x_i within the feature vector $\mathbf{x} = [x_1, x_2, ..., x_{N_f}]$, where N_f represents the number of features. The connections between neurons of one layer and those of the subsequent layer indicate the flow of information. When every neuron in one layer is linked to all neurons in the following layer, these layers are referred to as fully connected (FC). These connections are parameterized as weights w_{ji}, representing their strength. The weighted sum (or activation) z_j^l of the j-th neuron within a hidden layer l can be computed as a linear combination of all N_{neu}^{l-1} neurons (features) of the preceding layer:

$$z_j^l\left(\mathbf{x}^{l-1}\right) = \sum_{i=1}^{N_{neu}^{l-1}} w_{ji}^{l-1} x_i^{l-1} + b_j^{l-1}. \tag{3.13}$$

Hence, the contribution of the i-th neuron in the preceding layer is the sum of its weighted feature value $w_{ji}^{l-1} x_i^{l-1}$ and a bias term b_j^{l-1}. In vector-matrix notation, the weighted sum of all hidden layer l neurons is given as:

$$\mathbf{z}_l = \mathbf{W}_{l-1}^T \mathbf{x}_{l-1} + \mathbf{b}_{l-1}, \tag{3.14}$$

where

$\mathbf{z}_l \in \mathbb{R}^{N_{neu}^l \times 1}$	– vector of activations
$\mathbf{W}_{l-1} \in \mathbb{R}^{N_{neu}^{l-1} \times N_{neu}^l}$	– weight matrix
$\mathbf{x}_{l-1} \in \mathbb{R}^{N_{neu}^{l-1} \times 1}$	– feature vector
$\mathbf{b}_{l-1} \in \mathbb{R}^{N_{neu}^{l-1} \times 1}$	– bias vector

Each value z_j^l is subsequently passed through a nonlinear activation function $a(z_j^l)$, without which the network's outputs would be only a linear combination of its inputs, limiting its capacity for approximation. Moreover, the purpose of the bias term becomes clear when considering the application of activation functions. It represents the shift of the activation function, similar to how a linear function can be shifted, which enables more complex computations within each neuron. The choice of activation function depends on the underlying dataset and the particular problem being addressed. One frequently employed activation function in deep learning is the Rectified Linear Unit (ReLU). It returns 0 for input values less than or equal to 0 and keeps the input unchanged for values greater than 0. There are also other popular activation functions, such as the hyperbolic tangent (*tanh*), which outputs values in the range between -1 and 1, and the sigmoid activation function *sigm*, which outputs values in the range between 0 and 1 [41].

Applying activation functions to all the activations within the current layer yields the inputs (the feature vector) of the subsequent layer:

$$\mathbf{x}_l = a_l(\mathbf{z}_l) = a_l(\mathbf{W}_{l-1}^T \mathbf{x} l - 1 + \mathbf{b}_{l-1}). \tag{3.15}$$

At a higher level, the transformations of a layer l can be summarized as $m_l(\mathbf{x_l})$. This allows for the representation of the network's approximation $f_\theta(\mathbf{x})$ as:

$$f_\theta(\mathbf{x}) = m_{N_L} \circ m_{N_{L-1}} \circ ... \circ m_0(\mathbf{x}), \tag{3.16}$$

with the network's parameters (weights) θ, the input feature vector $\mathbf{x}$ and the number of layers N_L. To make a prediction, neural networks perform a forward pass, during which a specific feature vector $\mathbf{x}_i$ traverses through all transformation blocks of the network, resulting in the model prediction $\hat{\mathbf{y}}_i$:

$$\hat{\mathbf{y}}_i = f_\theta(\mathbf{x}_i). \tag{3.17}$$

In a supervised learning context, the optimization task revolves around finding a set of parameters θ that minimizes a specific error (or loss) function. A commonly employed loss function in regression problems is the Mean Squared Error (MSE), also known as $L2$ loss [30], which will be used for the model optimizations in the present study. Formulated with a general loss function L, this results in the general optimization problem:

$$\theta^* = \arg\min_{\theta} \mathrm{L}(\theta), \tag{3.18}$$

where θ^* refers to the combination of model weights yielding the minimal loss.

As previously mentioned in section 3.1.2, optimization methods such as backpropagation are employed to determine the model weights that minimize the error between the predictions and the true labels [4]. Consequently, the gradient of the loss function $dL/d\theta$ is computed and used to modify the model weights using optimizer algorithms such as gradient descent. This involves updating the current weights θ_n with a portion λ of the gradient to yield the next set of weights θ_{n+1}:

$$\theta_{n+1} = \theta_n - \lambda \frac{d\mathrm{L}}{d\theta}. \tag{3.19}$$

The parameter λ is commonly known as the learning rate. However, in practical scenarios, more advanced optimizer algorithms are frequently employed, particularly for complex and deep networks. One such algorithm is ADAM (short for adaptive moment estimation), which calculates distinct learning rates for individual parameters by making use of a concept called momentum [5]. Additionally, learning rate scheduling techniques are utilized to dynamically adjust the learning rate during training. Using the initial learning rate λ_0 and a function representation denoted as g for these advanced techniques, equation 3.19 can be redefined as follows:

$$\theta_{n+1} = \theta_n - g\left(\lambda_0, \frac{d\mathrm{L}}{d\theta}\right). \tag{3.20}$$

3.2.2 Convolutional Neural Network (CNN)

CNNs are a powerful category of neural networks that employ convolutional kernels (or filters) in place of traditional activation functions [4]. They start to outperform generic FC networks when the data is structured in a way that neighboring points exhibit significant correlation, resulting in patterns and structures. This type of data can typically be found in images, but also in time series. Fundamentally, convolutional layers are the primary components of CNNs, followed by other layers such as pooling layers, FC layers, and normalization layers [45].

Mathematically, convolution is a function that operates on two inputs: an image and a filter. The central aim of CNN optimization is to train the convolutional kernels to identify specific patterns within the input image and highlight these patterns in the resulting output image [30]. Input data matrices in CNNs are typically three-dimensional, characterized by an input channel dimension C_{in}[1], a height dimension H, and a width dimension W: $(C_{in} \times H \times W)$. Any point in the data matrix can be accessed by the indices (i, j, o). Based on the terminology introduced in [12, 41], the convolution can be expressed as:

$$\mathbf{z}_{i,j,o} = \sum_{l=0}^{C_{in}-1} \sum_{m=0}^{H_k-1} \sum_{n=0}^{W_k-1} x_{l,i\times s+m,j\times s+n} \mathbf{W}_{o,l,m,n} + \mathbf{b}_o. \tag{3.21}$$

Similar to the previous section, x denotes an input at a specific index, and $\mathbf{b}$ represents a bias term. Within the matrix $\mathbf{W}$, the filter's weights are stored, which define what kind of patterns to extract from the input.

[1]Most images are represented in RGB format, which means that they are composed of three channels: red, green, and blue. These channels are combined to create the full spectrum of colors within the image.

The convolution operation revolves around the concept of sliding a filter across the input data and performing element-wise multiplications between the filter weights and the corresponding values of the input. The summation of these operations results in a value within the output image. The parameter s defines the stride, which determines the number of steps by which the filter shifts while performing a convolution. This parameter offers a means to further control the reduction of the output size during the convolution operation. The entire set of filters within a convolutional layer can be represented by a matrix of dimension $(C_{in} \times H_k \times W_k \times C_{out})$, where H_k and W_k correspond to the filter's height and width, and C_{out} to the number of filters applied to the input [30]. This approach enables an increase in the abstraction of the spatial patterns present in an image, even as its resolution may be reduced.

In this study, convolutional layers are utilized for dimensionality reduction within the CNN-VAE architecture, as elaborated in section 3.1.2. Through the application of multiple convolutional layers, the CNN-VAE model is capable of learning an encoded and abstract representation of the input in the bottleneck layer. This is then followed by deconvolutions, representing the inverse operation, to decode the input and restore it to its original size.

3.2.3 Long Short-Term Memory (LSTM)

LSTM networks are a subset of the class of RNNs, which are specialized for tasks involving sequential data input, allowing them to utilize their internal state (memory) to process input sequences and maintain information about them [22]. However, a significant issue with conventional RNNs is the vanishing gradient problem. This issue revolves around RNNs struggling to capture long-term dependencies, as the backpropagated gradients either explode or vanish after a certain number of timesteps [18].

A solution to this problem is the LSTM architecture, first introduced by Hochreiter and Schmidhuber in 1997 [14]. Unlike traditional RNNs, the LSTM is designed to manage both short-term and long-term dependencies effectively. In contrast to the single neural network layer employed by conventional RNNs, the LSTM architecture, illustrated in fig. 3.4, consists of four interconnected layers, represented as rectangular boxes [22].

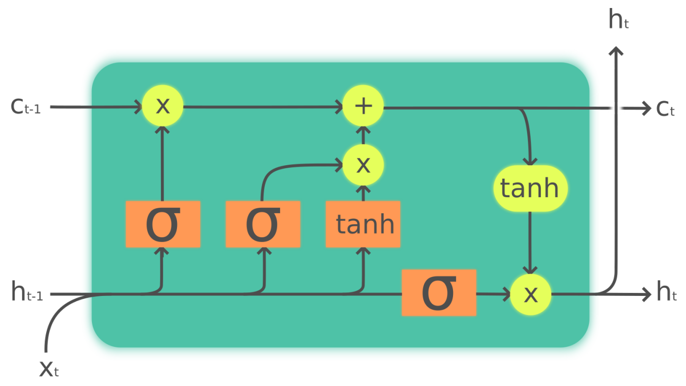


Figure 3.4: The LSTM Cell by Guillaume Chevalier, CC BY 4.0

The core idea of the LSTM algorithm is the cell state $\mathbf{c}$, which can be altered by structures called gates to store and forget information about past inputs [4]. The LSTM cell comprises several crucial components, each serving a specific purpose in processing sequential data. Following the nomenclature introduced in [22, 45], the forget gate $\mathbf{f}_t$ is the first of these components. It determines which portion of the previous cell state $\mathbf{c}_{t-1}$ to retain, based on the previous hidden state $\mathbf{h}_{t-1}$ and the current feature vector $\mathbf{x}_t$:

$$\mathbf{f}_t = sigm\left(\mathbf{W}_f \mathbf{x}_t + \mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right), \tag{3.22}$$

where $\mathbf{W}_f$ is the weight matrix of the forget layer and $\mathbf{b}_f$ is the bias term. The sigmoid activation function *sigm* is then applied to the weighted sum.

The input gate $\mathbf{i}_t$ is the second component of the LSTM cell, determining the extent to which new information is used to update the previous cell state $\mathbf{c}_{t-1}$, based on the same inputs as the forget gate:

$$\mathbf{i}_t = sigm\left(\mathbf{W}_i n \mathbf{x}_t + \mathbf{W}_i n \mathbf{h}_{t-1} + \mathbf{b}_i n\right). \tag{3.23}$$

The LSTM cell also introduces a candidate cell state vector $\tilde{\mathbf{c}}_t$ that processes the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ by means of a *tanh* activation function:

$$\tilde{\mathbf{c}}_t = tanh\left(\mathbf{W}_h \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}_h\right). \tag{3.24}$$

The new cell state $\mathbf{c}_t$ is then computed by multiplying the previous state $\mathbf{c}_t$ with the forget gate vector $\mathbf{f}_t$ and by adding the product of the input gate vector $\mathbf{i}_t$ and the new cell state $\tilde{\mathbf{c}}_t$:

$$\mathbf{c}_t = \mathbf{f}_t \mathbf{c}_{t-1} + \mathbf{i}_t \tilde{\mathbf{c}}_t \tag{3.25}$$

Finally, the output gate $\mathbf{o}_t$ determines what part of the new cell state $\mathbf{c}_t$ to output. It is computed in a similar manner as the previous gates:

$$\mathbf{o}_t = sigm\left(\mathbf{W}_o ut \mathbf{x}_t + \mathbf{W}_o ut \mathbf{h}_{t-1} + \mathbf{b}_o ut\right). \tag{3.26}$$

The updated cell state $\mathbf{c}_t$ is squashed using a *tanh* activation function and then multiplied by the output gate vector to create the new hidden state $\mathbf{h}_t$, which also serves as the output of the current time step:

$$\mathbf{h}_t = \mathbf{o}_t tanh(\mathbf{c}_t). \tag{3.27}$$

3.3 Metrics

Error Metrics play an essential role in the training and testing of ROMs. They provide a means to assess a model's performance and quantify the discrepancies between model predictions and actual labels. Given the complexity of the spatiotemporal data in this study, there are multiple possibilities to analyze errors.

3.3.1 Mean Squared Error (MSE)

As mentioned earlier in section 3.2.1, the MSE (or L2) error serves as the primary evaluation metric in this work. Following the terminology of section 3.2, the MSE for all N pairs of feature-label data $\mathbf{x}_i$ and $\mathbf{y}_i$ can generally be computed as follows [41]:

$$\text{MSE}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - f_\theta(\mathbf{x}_i))^2, \tag{3.28}$$

where θ refers to the model's parameters f.

According to the definition in section 3.1.1, a data matrix $\mathbf{X}$ is two-dimensional, with m state dimensions and n snapshots, indexed by (i, j). Consequently, the MSE of a reconstructed data matrix can be expressed as:

$$\text{MSE} = \frac{1}{m \times n}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\mathbf{X}^{(i,j)} - \mathbf{X}^{(i,j)}_{reconstr.}\right)^2, \tag{3.29}$$

yielding a scalar value.

For evaluating the reconstruction performance of SVD and CNN-VAE, two distinct categories of MSE are employed: temporal and spatial MSE. The temporal MSE, quantifying errors for each timestep, is computed column-wise:

$$\mathbf{MSE}[j] = \frac{1}{m} \sum_{i=1}^{m} \left(\mathbf{X}^{(i,j)} - \mathbf{X}^{(i,j)}_{\text{reconstr.}} \right)^2, \quad \text{for } j = 1, 2, \ldots, n. \tag{3.30}$$

Similarly the spatial MSE reveals information about the spatial distribution of the error, averaged over time:

$$\mathbf{MSE}[i] = \frac{1}{n} \sum_{j=1}^{n} \left(\mathbf{X}^{(i,j)} - \mathbf{X}^{(i,j)}_{\text{reconstr.}} \right)^2, \quad \text{for } i = 1, 2, \ldots, m. \tag{3.31}$$

When visualizing spatial errors, it is useful to reshape the spatial MSE vector into a matrix with dimensions $(H \times W)$. In the case of CNN-VAE, the data matrices are already three-dimensional with $(H \times W \times n)$, resulting in an MSE matrix with two dimensions. The computations were adapted accordingly.

Additionally, the hybrid model architecture results in two distinct spaces that can be used for error assessment: the latent space and the original full space. The concept of "latent loss" refers to an error metric calculated between the reduced snapshot and the reduced snapshot predicted by the models evolving the reduced state in time. Furthermore, the term "full space loss" denotes the computed error between the original snapshot and the reconstructed snapshot, with or without the time evolution of the reduced snapshot taken into account.

3.3.2 Coefficient of Determination (R^2)

To assess the extent to which the predictions capture the variance present in the input data, the coefficient of determination R^2 is employed. This statistical metric quantifies the proportion of the variance in the input data that is accounted for by the predictions of the model. This mathematical relationship can be expressed as follows [17]:

$$\begin{aligned} R^2 &= 1 - \frac{\frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\mathbf{X}^{(i,j)} - \mathbf{X}^{(i,j)}_{reconstr.} \right)^2}{\frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\mathbf{X}^{(i,j)} - \bar{\mathbf{X}} \right)^2} \\ R^2 &= 1 - \frac{MSE}{var\left(\mathbf{X}^{(i,j)}\right)}, \end{aligned} \tag{3.32}$$

where $Var\left(\mathbf{X}^{(i,j)}\right)$ represents the variance of the data matrix, which is the squared deviation of each data point from the mean $\bar{\mathbf{X}}$.

3.3.3 Power Spectral Density (PSD)

To assess the accuracy of the models in capturing the buffet characteristics, the distribution of energy in the frequency domain is compared using the Power Spectral Density (PSD). To achieve this, both the original and reconstructed data matrices are decomposed through SVD. As outlined in section 3.1.1, the matrix of mode coefficients $\mathbf{A}$ captures the temporal variations, with each row corresponding to the time variation of the i-th POD mode, which can be denoted as $\mathbf{a}_i(t)$.

Using the Fast Fourier Transform (FFT) algorithm, which is commonly employed to decompose a discretized signal into its sine and cosine components, the mode coefficients can be analyzed.

This involves transforming the temporal behavior of the mode coefficients $\mathbf{a}_i(t)$ into the frequency domain: [4]:

$$\hat{\mathbf{a}}_i(f) = FFT(\mathbf{a}_i(t)), \tag{3.33}$$

where $\hat{\mathbf{a}}_i(f)$ represents how the i-th mode is present at each frequency f. Finally, to assess the power of each mode, the PSD can be computed as the normalized squared magnitude of the FFT [4]:

$$\mathbf{PSD}_i = \frac{|\hat{\mathbf{a}}_i(f)|}{n}, \tag{3.34}$$

where n is once again the number of snapshots that make up the mode.

Chapter 4

Model Optimization

This chapter outlines the methodologies employed in the present study. Beginning with a brief overview of the source and nature of the underlying data, the training processes of the dimensionality reduction techniques and time-evolution models are explained. The chapter concludes with the presentation of an end-to-end training approach. The aim of the following sections is to provide insights into the frameworks employed, the range of parameters utilized, and the conducted parameter studies aimed to optimize model performance.

4.1 Data Source and Preparation

Training, testing, and validation of the ROMs were conducted using surface pressure data obtained from wind tunnel test campaigns carried out in the ETW. Throughout these campaigns, the Airbus provided industrial-standard research test case XRF-1 has been employed for buffet investigations. The campaigns included two different configurations of the model, one with Ultra High Bypass Ratio (UHBR) Through Flow Nacelles (TFNs) and one with the clean wing. For this work, only the c_p-data obtained with the nacelle configuration was used. Wind tunnel tests were carried out by the ETW and the German Aerospace Center (DLR) using IPSP [43]. The IPSP measurements enable time-resolved buffet investigation of the upper wing through 2kHz sampling [19, 45]. In this work, the temporal scale was normalized in terms of the mean chord and the free-stream airspeed, resulting in a dimensionless time parameter represented as $\tau = tU_\infty/\bar{c}$.

During the measurements, the free-stream Mach number and Reynolds number were varied between $Ma_\infty = [0.84,\ 0.87,\ 0.9]$ and $Re = [3.3\text{ Mio.}, 6.7\text{ Mio.}, 12.9\text{ Mio.}, 25\text{ Mio.}]$. The angle of attack was varied additionally for each of the flow conditions [19, 45]. For simplification, the present work focuses on the $Ma_\infty = 0.84$ flow condition with $Re = 12.9$ Mio. considering seven different angles of attack $\alpha = [1.5°, 3.0°, 3.5°, 4.0°, 4.5°, 5.0°, 6.0°]$.

As a first post-processing step, non-physical c_p values were clipped by applying a weight mask that defines values of $c_p \geq 1.5$ as 1 and values of $c_p \leq -1.5$ as 0 [45]. The experimental data also exhibited shock reflections that were reduced in a second post-processing step via SVD, as shown in fig. 4.1. For that, the area around the reflections was masked. Then, the SVD was computed for a data matrix constructed from the mask. The undesired modes were identified, the masked area was reconstructed without the mentioned modes and the post-processed area was added back into the dataset. Furthermore, a small area on the wing edges was masked and set to uniform values as non-physical c_p outliers are especially likely in these areas due to the strong curvature of the wing surface.

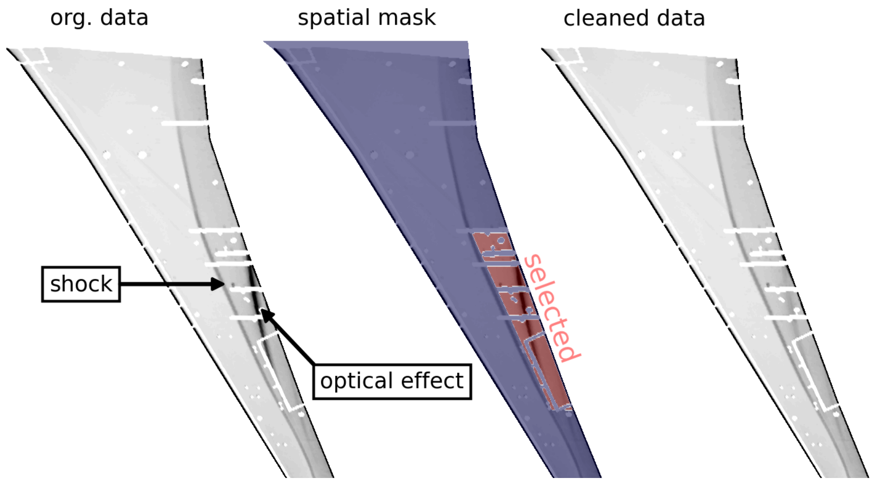


Figure 4.1: Spatial mask with SVD to eliminate shock reflections from the dataset ($Ma_\infty = 0.9, Re = 25\text{Mio.}, \alpha = 2.50°$)

4.1.1 Interpolation

As a first step after receiving the post-processed dataset, its resolution was downscaled according to the approach by ZAHN et al. [45] who conducted work on a comparable dataset at higher Re. Interpolating the snapshots from an array of (465×159) to (256×128) data points enables faster processing of the data since the length and width of the array are both powers of two $(2^8 \times 2^7)$.

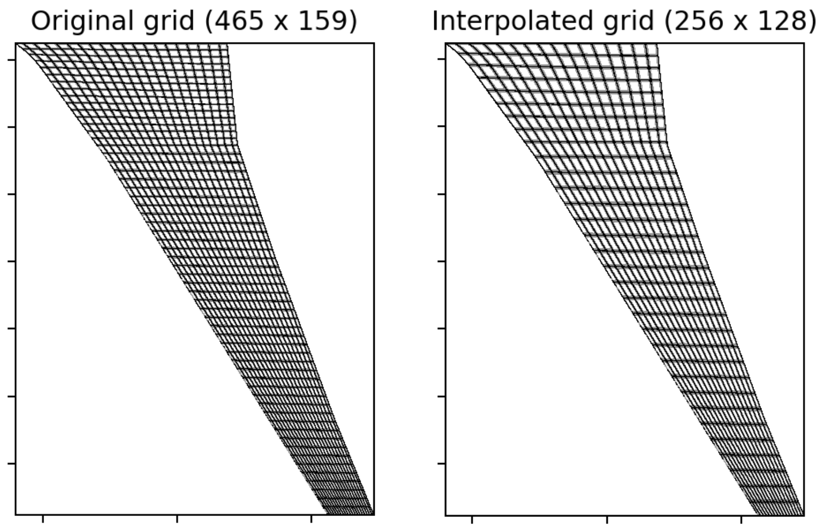


Figure 4.2: Comparison of the original and interpolated grid. Note that these are not the actual data grids. The displayed grids were coarsened by a factor of 8 and only for scale comparison.

In fig. 4.2 a comparison of the original and the downscaled point grids is shown. It has to be noted that the displayed grids were coarsened by a factor of eight to enable a sensible visual comparison. In fig. 4.3 a snapshot is visualized in the original and the downscaled resolution, respectively. As expected, the interpolated snapshot is slightly blurred compared to the original one. Downscaling means losing information - which is not necessarily purely a negative effect. The interpolation acts like a filter and could also aid in suppressing noise artifacts in the experimental data that the unsupervised learning techniques may try to reconstruct. Apart from that, no differences can be identified and there are no changes in the physical meaning of the data.

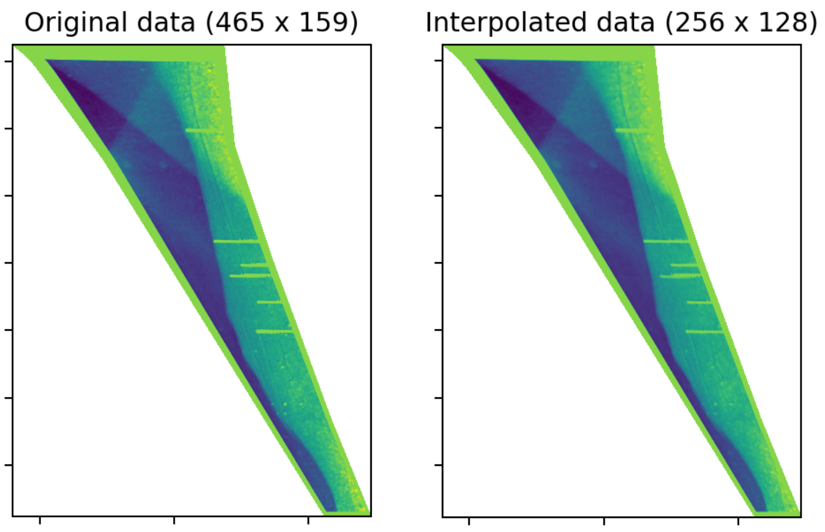


Figure 4.3: Comparison of original and interpolated c_p snapshot ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 5.00°, \tau = 53.69$)

4.2 Dimensionality Reduction

After interpolating and normalizing the dataset to [-1, 1], dimensionality reduction was accomplished through SVD and a CNN-VAE neural network separately. Both were trained on five of the angles of attack and tested on the remaining two which are $\alpha = [3.0°, 5.0°]$. To avoid extrapolation during testing, the training dataset contains the lowest and highest angles of attack measured. In section 4.2.1, the results obtained from training and testing the SVD are presented, followed by the results of the CNN-VAE in section 4.2.2.

4.2.1 SVD

SVD was performed with the Python library *flowtorch* [42] and its dedicated SVD function. The following elaboration serves the purpose of analyzing the SVD's performance in dealing with this type of data and finding a suitable truncation rank.

SVD Computation

As previously mentioned, the modal decomposition of the flow is based on five of the seven flow conditions. These data subsets with 500 timesteps each were concatenated to a data tensor from which its temporal mean was subtracted in order to center it and therefore have a zero-mean component in the time axis. Since the SVD is based on data matrices with only one state dimension, the height and width dimensions were flattened. The SVD computation then results in 2500 left singular vectors (or POD modes). In fig. 4.4, the individual and cumulative contributions of the mentioned modes to the sum of all singular values are illustrated. Due to the significant drop in individual contributions after the first few modes, the y-axis of the upper figure is logarithmic. The dashed line in the figure represents the *flowtorch.analysis.SVD* attribute *opt_rank*, which corresponds to the optimal truncation rank determined through optimal hard thresholding. For this specific dataset, the *opt_rank* was calculated to be 323. The lower illustration in fig. 4.4 indicates that the first 1994 singular values yield a cumulative contribution of 90.02% to the sum of all singular values.

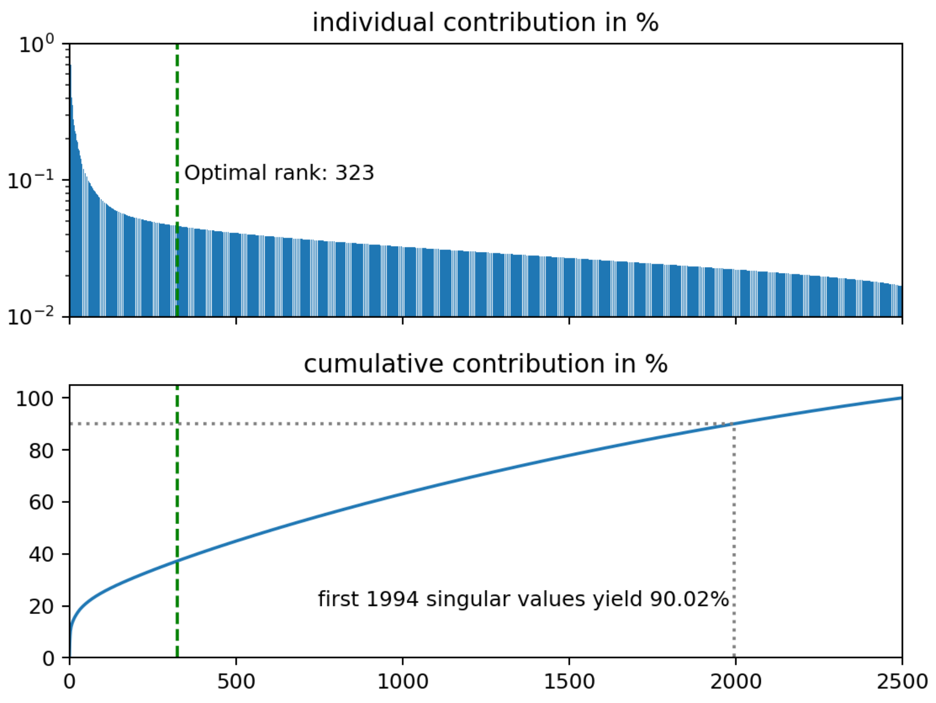


Figure 4.4: Singular value contributions

Since the actual performance of the modal decomposition can't be assessed on the training data, the two aforementioned test datasets were reduced and reconstructed with the left singular vectors of the SVD, as derived in section 3.1.1. The results are presented in the following section.

Truncation Rank Study

In the first step, the test datasets were reduced and reconstructed for ranks $1-400$, followed by computing the MSE and variance reconstruction in the form of R^2 (refer to section 3.3.2). The rank-dependent metrics are illustrated in fig. 4.5. As already verified for the training data, the first few ranks contribute to the majority of information reconstruction which can be recognized by the steep drop in MSE and steep increase in variance reconstruction, respectively. After the first 300 POD modes are considered, the R^2 value converges towards ≈ 0.985 for the first test case and ≈ 0.99 for the second one. This convergence indicates that the snapshot reconstruction is in very good agreement with the original data. As the rank increases further, the reconstruction improvement becomes marginal. This phenomenon stems from the fact that both test datasets were not included in the SVD computation, resulting in missing modal information for these datasets.

To analyze potential temporal dependencies in the reconstruction error, the temporal MSE distribution (refer to 3.3.1) was computed for selected ranks between 1 and 1000 in Figure 4.6. Both datasets exhibited similar behavior, so only the results for the first test case are shown. The figure reveals that there is no apparent correlation between time and MSE, with fluctuations appearing random. Moreover, the figure suggests that beyond a certain rank, further information gained from additional ranks does not significantly contribute to improving the quality of reconstruction.

In fig. 4.7, the spatial MSE distributions (refer to 3.3.1) for ranks $[3, 30, 300]$ are presented for both test datasets. It can be observed from the figure that the most significant deviation from the ground truth is concentrated in the shock foot area between the medium span and wing tip. This deviation is likely due to the intense pressure dynamics (i.e. variance) that occur in that specific region. With increasing rank, a desirable decrease in MSE can be observed.

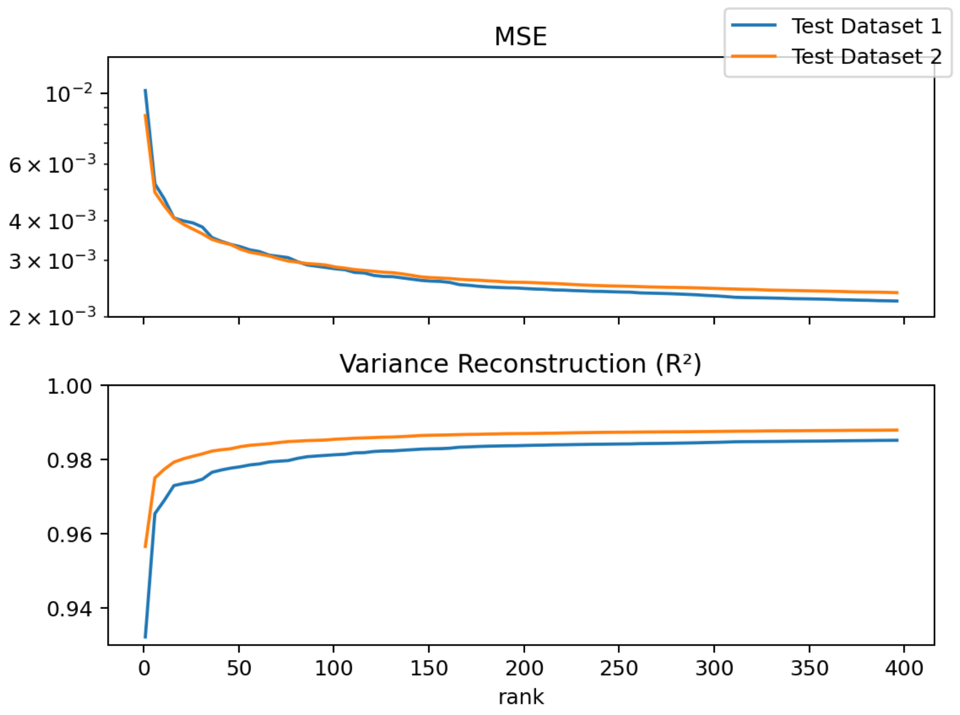


Figure 4.5: MSE and variance reconstruction with increasing rank

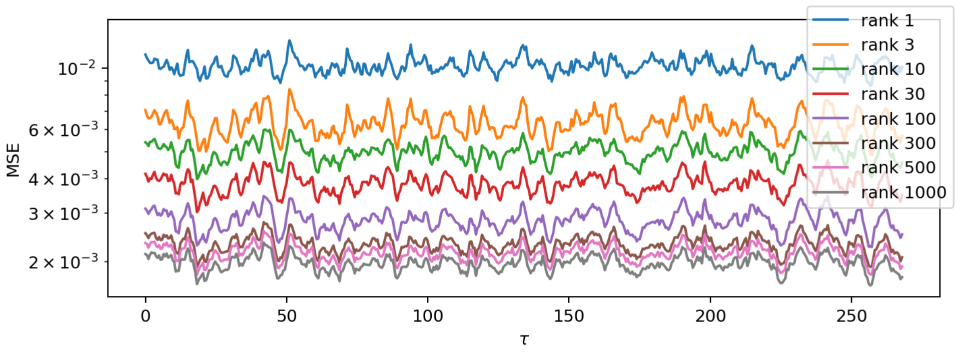


Figure 4.6: Temporal MSE distributions with increasing rank for test dataset 1

For a side-by-side comparison of the actual c_p-snapshots, fig. 4.8 depicts the reconstructed flow fields with ranks 30 and 300, alongside the ground truth, for both test datasets. The reconstructed pressure distributions capture the characteristic shock pattern, as indicated by the ground truth, with only minor deviations. However, the shock appears less detailed and slightly misaligned, especially in the reconstructions with only 30 modes. This observation is further supported by the presence of larger error regions, as illustrated in Figure 4.7. The reconstruction with rank 300 reflects more details, particularly in the areas of interest. Beyond rank 300, the physical accuracy does not increase much further, only more noise will be added by the additional POD modes, indicating that there is more data needed. Based on the discussions in this section, rank 300 has been selected as the truncation rank, i.e. only the first 300 modes are used to span the reduced space.

4.2.2 CNN-VAE

All neural networks, including the CNN-VAE model, were trained using the *PyTorch* framework. In the following sections, the training and evaluation of various CNN-VAE model architectures

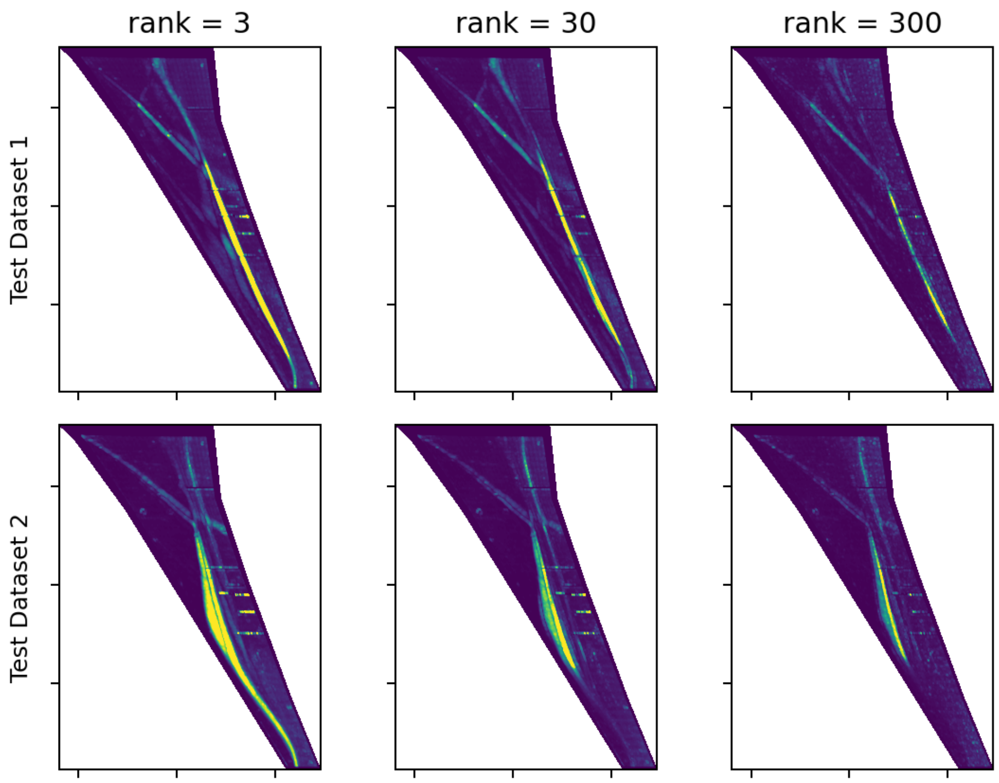


Figure 4.7: Comparison of spatial MSE distributions for reconstructed c_p-snapshot using SVD ranks 3, 30, and 300 ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha_1 = 3.00°, \alpha_2 = 5.00°$)

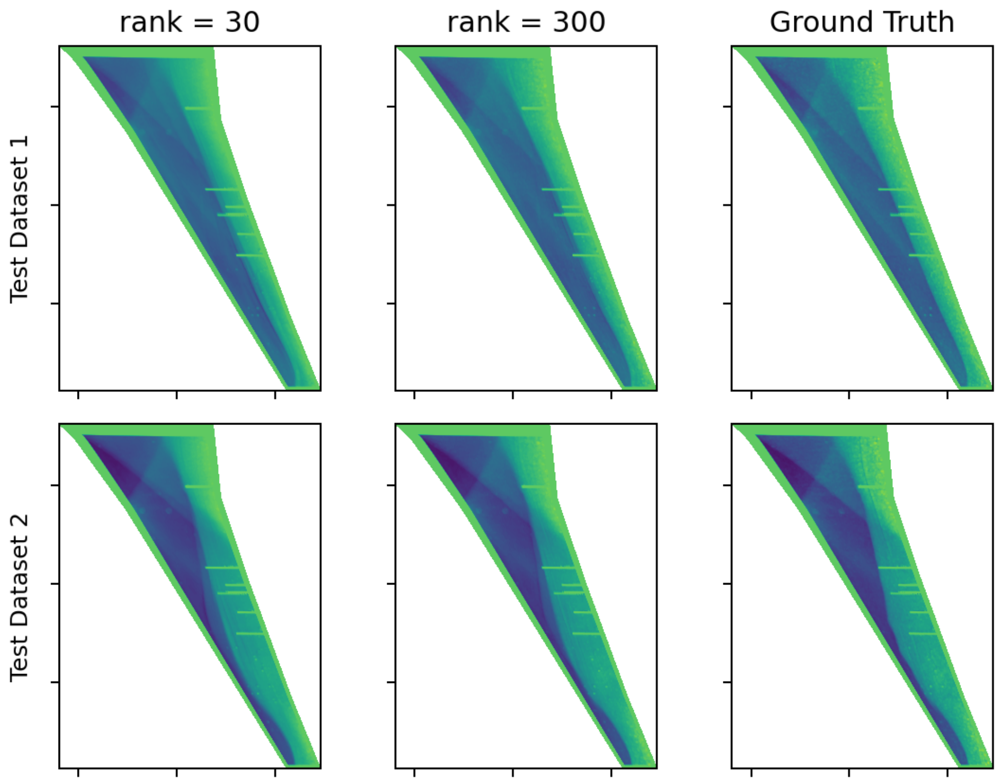


Figure 4.8: Comparison of c_p-snapshot with reconstructions using SVD ranks 30 and 300 ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha_1 = 3.00°, \alpha_2 = 5.00°, \tau = 53.69$)

are presented, following a similar approach as for the SVD for comparability. The objective of this section is to identify the most suitable CNN-VAE model, i.e. the optimal bottleneck layer (or latent) size.

CNN-VAE Training

The CNN-VAE model architecture used in this work was adopted from the previous study by Zahn et al. [45], already mentioned in 1.2. The model comprises five convolutional levels in both the encoder and decoder, transforming the input size from $(1 \times 256 \times 128)$ to $(512 \times 32 \times 16)$ and back, where the first dimension represents the number of channels, followed by the height and width. The encoder is followed by a regularization framework, from which a latent vector is sampled during a forward pass and then passed to the bottleneck layer for further processing.

During an initial parameter study, layer normalization layers were tested as an alternative to batch normalization layers, and they were found to enhance both convergence speed and model quality. Additionally, the *ReduceLROnPlateau* learning rate scheduler was employed, in combination with the *Adam* optimizer with an initial learning rate of 10^{-4}. The *MSELoss* loss function was used to compute the error between labels and predictions. In the aforementioned work by Zahn et al., the bottleneck layer size was set to 256. To account for the modifications made to the model and training routine, as well as the differences in the underlying datasets, a bottleneck layer size study was conducted, similar to the truncation rank study in the previous chapter. In this study, seven different CNN-VAE models were trained with varying bottleneck layer sizes of (8, 16, 32, 64, 128, 256, 512) neurons. Each model was trained for 500 epochs with a batch size of 32. To address uncertainties during the training process, each training was repeated 10 times, resulting in a total of 70 trainings.

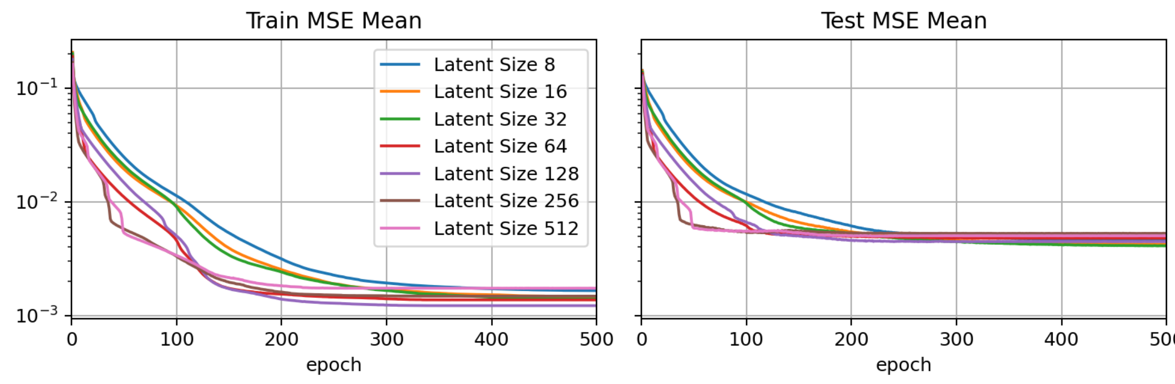


Figure 4.9: Training and test MSE mean of 10 training iterations for different latent sizes of the CNN-VAE model

Similar to the SVD approach, the dataset was split into five flow conditions for training and two for testing. Additionally, during training, 10% of each flow condition in the training set was utilized for validation. This results in a dataset comprising 2250 snapshots for training, 250 for validation, and 1000 for testing.

In fig. 4.9, the average train and test losses of each model are displayed. Notably, the loss ranges of all model configurations are similar and they tend to converge toward similar values, with the training loss varying from 1×10^{-3} to 2×10^{-3} and the test loss from 4×10^{-3} to 5×10^{-3}. While there is no clear correlation between the bottleneck layer size and the final loss, it is noticeable that models with smaller bottleneck layers generally require more epochs to converge during training.

Additionally, fig. 4.10 provides information about median, quartiles, and potential outliers in

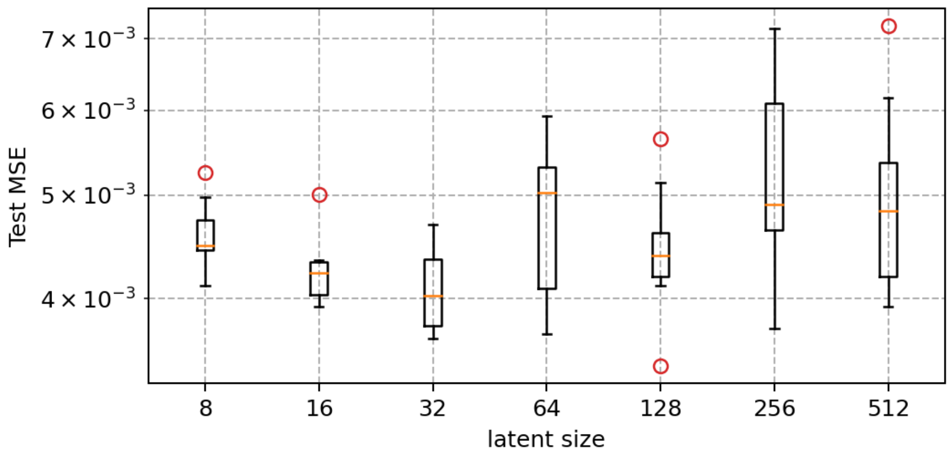


Figure 4.10: Test MSE tendency and spread for different latent sizes of the CNN-VAE model

terms of test loss for all training iterations of each model. The figure does not reveal a clear connection between the number of bottleneck neurons and a general trend of the test loss. However, it does indicate that the test loss range increases with the model complexity, and training with smaller models tends to yield more comparable results than with larger models which is due to the lower number of parameters. While no optimal bottleneck size is evident from fig. 4.10, it is worth noting that among all training iterations, the models with 32 and 128 bottleneck neurons achieved the lowest test losses.

Bottleneck Size Study

In the next step, the model with the lowest test loss among all training iterations was selected for each architecture. Similar to the SVD evaluation in section 4.2.1, the different CNN-VAE models were employed to reconstruct two test datasets. In fig. 4.11, the MSE and R^2 values were computed for each latent size. While there's no clear pattern in the behavior of the variables, the overall MSE and R^2 value ranges are comparable to those in fig. 4.5 from the SVD analysis. It's worth mentioning that the models exhibit greater accuracy differences for both test cases, with deviations in R^2 values reaching up to 2%.

The temporal MSE distribution of the CNN-VAE reconstruction exhibits similar qualitative behavior as observed in fig. 4.6, indicating the absence of temporal dependencies. In fig. 4.12, the spatial MSE distributions for the two selected latent sizes, 32 and 128, are displayed. The CNN-VAE model with 256 bottleneck neurons will be used as a reference for comparison to Zahn et al. [45]. Much like the SVD reconstruction, the spatial distribution of MSE is concentrated in the shock front regions. The reconstruction involving 256 bottleneck neurons presents more evident artifacts. Nonetheless, all models exhibit nearly identical performance, with minimal variations in MSE and slight differences in the contours of regions dominated by errors.

When comparing the c_p-snapshot reconstruction of the best-performing models with the ground truth, there are only marginal differences visible. They exhibit an overall good agreement with the ground truth as the shock patterns are predicted accurately, and there is a desirable reduction in noise compared to the experimental data, similar to the SVD reconstruction in fig. 4.8. Since the reconstructed snapshots do not yield additional insights into performance, the model choice is based on fig. 4.11, where MSE and R^2 for various bottleneck layer sizes are depicted. Overall, the CNN-VAE model employing 128 bottleneck neurons yields the most favorable outcomes across both test cases and is therefore selected.

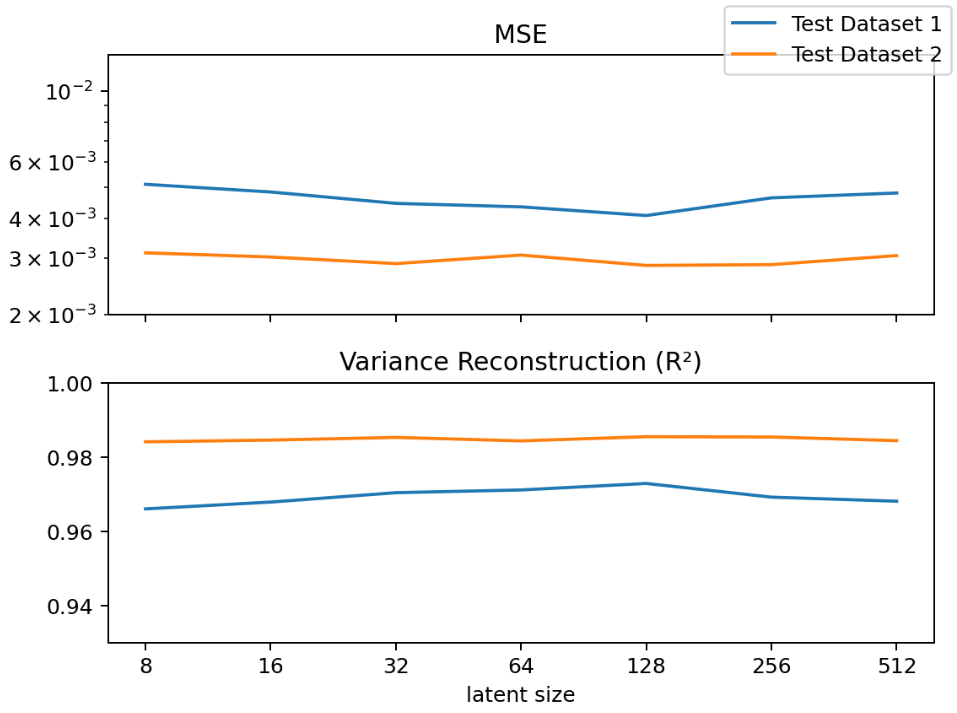


Figure 4.11: MSE and R^2 with increasing latent size

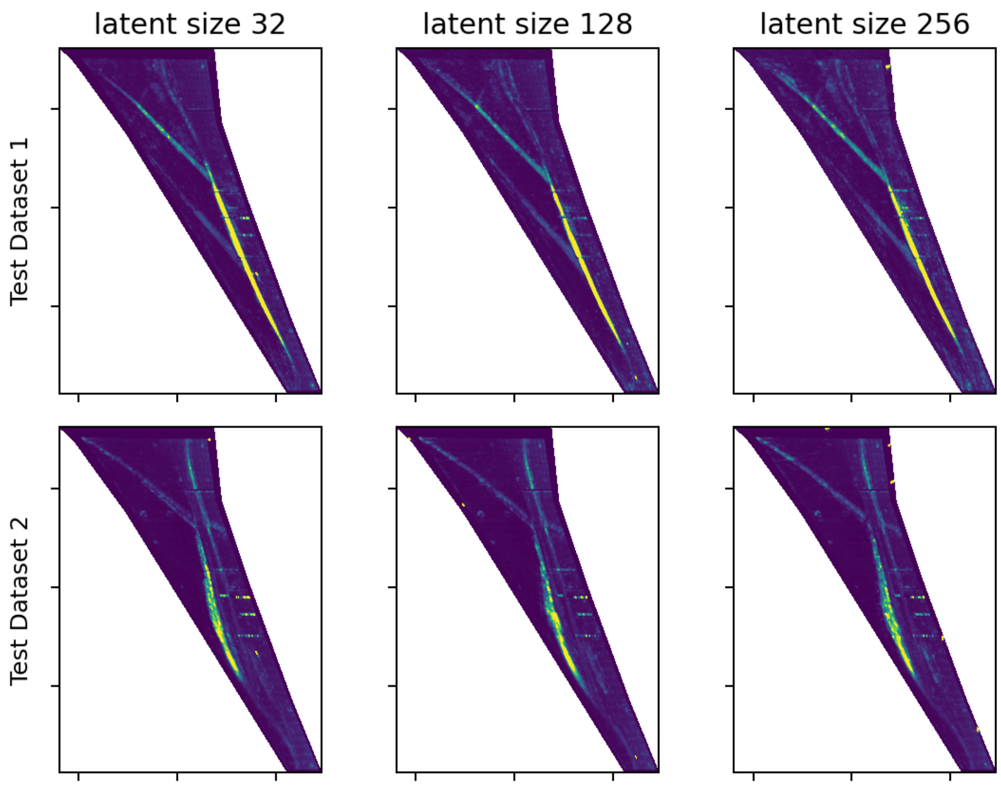


Figure 4.12: Comparison of spatial MSE distributions for reconstructed c_p-snapshot using 32, 128, and 256 bottleneck neurons ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha_1 = 3.00°, \alpha_2 = 5.00°$)

4.3 Time Evolution in Reduced State

The employment of SVD and the CNN-VAE to project the c_p snapshot data onto a low-dimensional space represented the initial step in the ROM development. For predicting time evolution, a second set of models was trained utilizing the reduced/encoded state of the $\alpha = 4.00°$ flow condition. In this case, the training incorporated the first 80% of the dataset (400 timesteps), preserving the remaining 20% (100 timesteps) for monitoring and testing. As the baseline model, a generic FC network was employed, as derived in 3.2.1. Following this, a recurrent network architecture was implemented with the LSTM model, detailed in 3.2.3. To embed the sequential time-series data into the training routine, a *DataWindow* class was implemented. Given an input sequence length and a prediction horizon, the class computes a rolling window mechanism for generating pairs of input-target sequences over a given dataset. In this work, the term "targets" refers to a sequence of labels in a data window that is compared against the prediction sequence for optimization. All input-target pairs are stored in a *TensorDataset* and then fed into a *Dataloader*. The subsequent sections will provide an overview of the model architectures, as well as the training and optimization process.

4.3.1 FC Models

Given the different compression sizes resulting from SVD and CNN-VAE, 300 and 128 dimensions, respectively, the input and output dimensions of the FC models vary. Unlike previous methods, the order of c_p-snapshots in a sequence is relevant for capturing temporal relationships. To address this, the sequences of mode coefficients were aligned to form a feature vector, which was then fed into the FC model's input layer. Given the models are designed for single-step forecasts, the number of output neurons matches the size of the reduced/latent space.

During training, the CNN-VAE methodology was adapted and adjusted to integrate auto-regressive prediction steps into the optimization process. To enhance GPU (Graphics Processing Unit) utilization and training efficiency, the batch size was increased to 64. The remaining design parameters within the training architecture are the sequence length, the number of hidden layer neurons, the number of hidden layers, and the learning rate. Following an initial parameter exploration to identify suitable parameter ranges, several potential parameter combinations were subjected to training. After a sufficient convergence was achieved with the best-performing architecture, the prediction horizon was increased during training to enhance long-term accuracy stability. Instead of computing the loss for the next optimizer step based on a single prediction, it is computed considering a series of auto-regressive single-step predictions. The subsequent two sections will provide brief overviews of the study findings for both FC models.

SVD-FC Model

The SVD-FC model combines data reduction and reconstruction using the left singular vectors from the SVD with auto-regressive prediction of mode coefficients through an FC network. For this model, a configuration of 5 hidden layers, each containing 256 neurons, was adopted as it yielded the best training results. Sequence lengths were tested in the range of $[8, 48]$, and it was found that a sequence length of 32 provides the best predictive performance. Since all time steps are organized into a feature vector, the input layer of the SVD-FC network consists of 9600 neurons, while the output layer contains 300 neurons.

In fig. 4.13, the average train and test losses for 5 training iterations are displayed. The reduced number of training iterations in comparison to the CNN-VAE models is due to FC models having fewer trainable parameters, making them more likely to yield similar outcomes. Notably, the final train and test losses differ by an order of magnitude. This difference arises since the test loss stops decreasing beyond 1000 epochs and starts to increase again. While this suggests potential overfitting, the slight increase in test loss is outweighed by the model's continued learning of

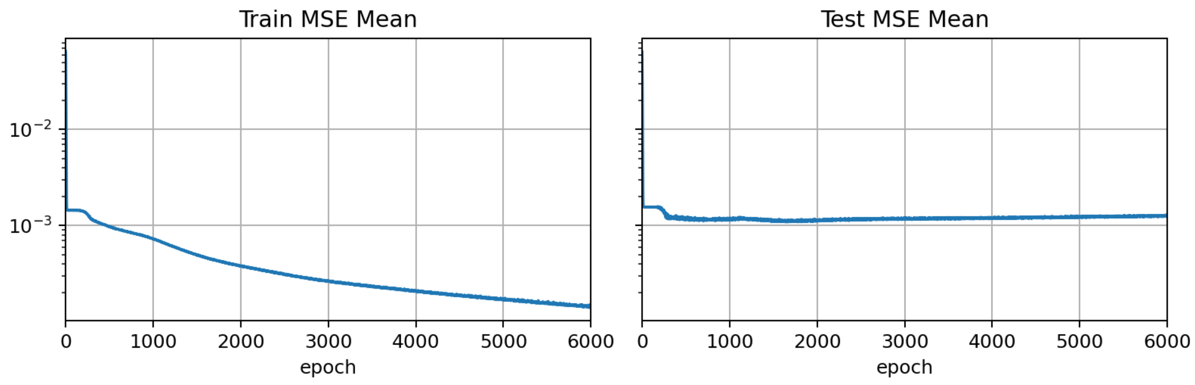


Figure 4.13: Training and test MSE mean of 5 training iterations of the SVD-FC model

temporal relationships from the training data. In the context of the present work, accurate temporal dynamics are more important than precisely matching the test data.

When training with a prediction horizon greater than 1, the extended training period positively impacts model stability when evaluated on test data, effectively acting as a regularization measure. An optimal balance between stability and overfitting was achieved around 6000 epochs of training, using a prediction horizon of 3 and a learning rate of $1 * 10^{-4}$. Higher prediction horizons would lead to a variance loss in the predictions, as the metrics used for optimization are evaluated on more and more timesteps and the gradients smear.

CNN-VAE-FC Model

Similar to the previous model, the CNN-VAE-FC model is a combination of snapshot encoding/decoding using the pre-trained CNN-VAE, and an FC network that forecasts the temporal evolution within the latent space. Given the latent space comprises only 128 features, an architecture with 128 hidden layer neurons across 5 hidden layers was employed. As the optimal input sequence length was also determined to be 32, the input layer consists of 4096 neurons, with 128 neurons in the output layer. An exploration of the remaining parameters identified a learning rate of $2 * 10^{-4}$ and a prediction horizon of 4 as suitable for the model's performance.

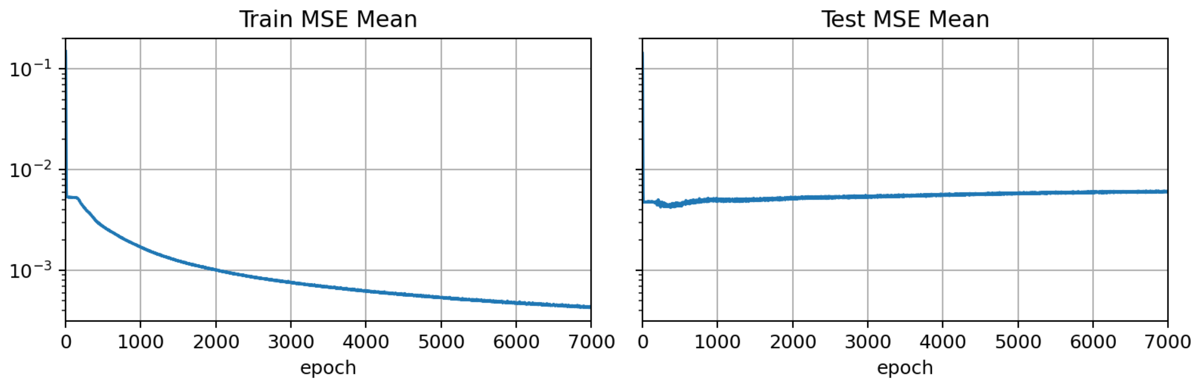


Figure 4.14: Training and test MSE mean of 5 training iterations of the CNN-VAE-FC model

In fig. 4.14, the train and test MSE mean over 5 training iterations, each involving 7000 training epochs, is depicted. Qualitatively, a similar trend is observed as seen in fig. 4.13. Notably, the model is unable to achieve as low train and test loss values as the SVD-FC model. The average final test loss corresponds to $8 * 10^{-3}$. This trade-off is acceptable given the model's extraction of temporal dynamics from the latent representation.

4.3.2 LSTM Models

A similar strategy was adopted for the training and optimization of the LSTM models. In contrast to FC networks, the input sequence is processed in a step-by-step (recurrent) manner rather than as a complete feature vector. A stacked LSTM architecture was employed, followed by an FC layer for making predictions. Throughout training, a batch size of 64 was employed, with an input sequence length of 32 timesteps. Within the model architecture, the free parameters include the number of stacked LSTMs and the number of neurons within each of the LSTM gate networks. Furthermore, a parameter study was carried out to obtain suitable learning rates and prediction horizons. In the following two sections, the findings of the optimization process are briefly presented.

SVD-LSTM Model

In the SVD-LSTM model, an architecture comprising 2 stacked LSTMs, each containing 256 features in the hidden state. The parameter study on the remaining parameters yielded a learning rate of $8 * 10^{-5}$ with a prediction horizon of 3. Fig 4.15 displays the average train and test MSE for 5 training iterations on the final configuration. A trend similar to the previous models is observable. However, in this case, the difference between train and test loss exceeds an order of magnitude. Nevertheless, the test MSE tends to converge toward a maximum loss after approximately 2000 epochs. Since the LSTM model exhibited significantly faster learning, the training process was stopped after 5000 epochs, during which a sufficient convergence was achieved.

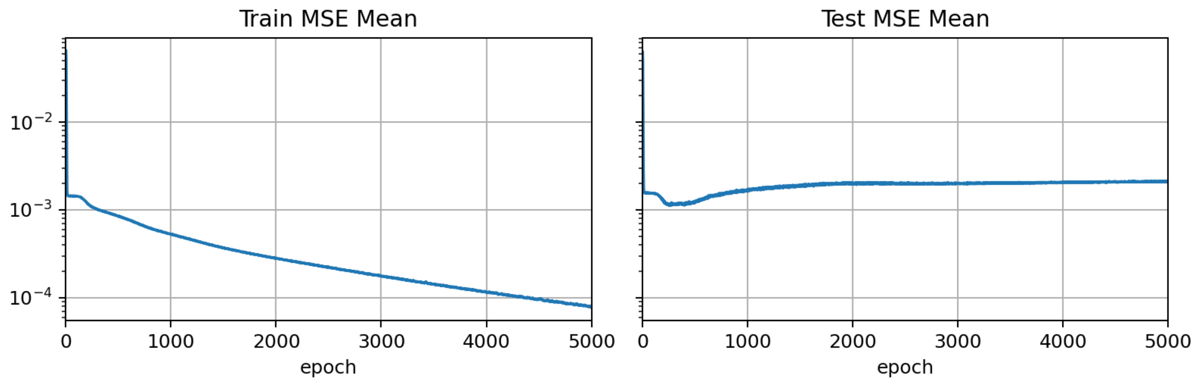


Figure 4.15: Training and test MSE mean of 5 training iterations of the SVD-LSTM model

CNN-VAE-LSTM Model

For the CNN-VAE-LSTM model, a configuration featuring 2 stacked LSTMs was implemented, each incorporating 128 hidden state features. Fig 4.16 depicts the training outcomes over 5 iterations of the final configuration with a learning rate of $8 * 10^{-5}$, a prediction horizon of 3, and a training duration of 5000 epochs. Compared to the preceding model, the average train and test losses are larger by an order of magnitude. This trend was also observed for the SVD-FC and CNN-VAE-FC models, but it is more distinct in the case of the LSTM models.

4.4 End-to-End Training

As the CNN-VAE-LSTM model stood out among the remaining ROMs, it was employed for end-to-end (E2E) training in which both ROM components are trained simultaneously by merging the separate models. E2E training is a common approach for deep neural networks as it enables bypassing all intermediate model stages requiring careful design, instead utilizing the layered

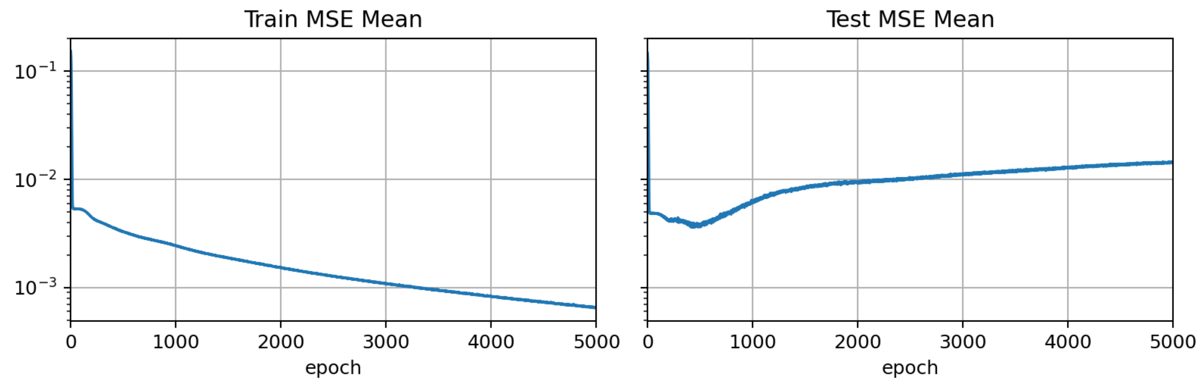


Figure 4.16: Training and test MSE mean of 5 training iterations of the CNN-VAE-LSTM model

architecture of neural networks to treat the model as a closed system [32]. For that, an LSTM was implemented between the encoder and decoder architectures.

In the first step, the the sequence length 32 is appended to the initial input size yielding the new input size $(1 \times 256 \times 128 \times 32)$. Within the *forward* function of the model, the input sequence is encoded to (64×32), where 64 represents the number of bottleneck neurons. The latent size was decreased to 64 as the end-to-end model was trained and evaluated on a single flow condition. The encoded sequence is subsequently processed by the stacked LSTM model with 2 layers, each containing 64 features in the hidden state. With a prediction horizon of 3 during training, the LSTM produces 3 recurrent single-step predictions, which are then decoded back to the full space and compared to the targets for optimization.

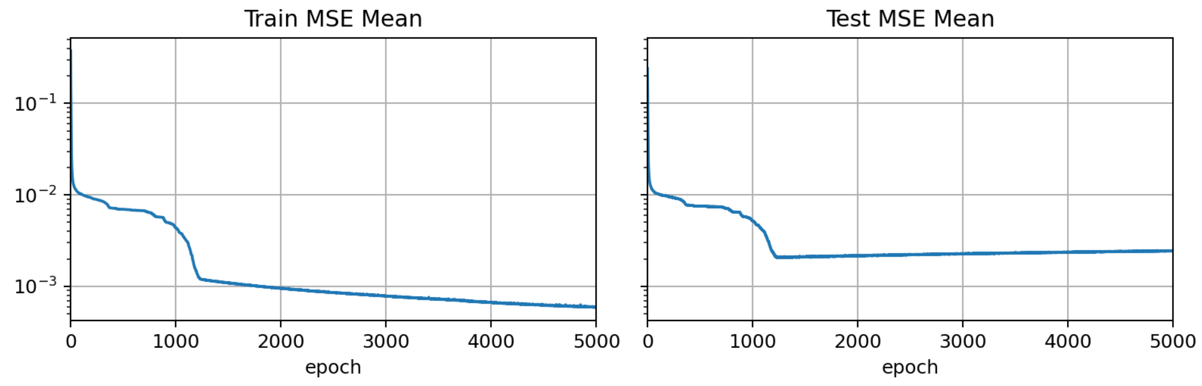


Figure 4.17: Training and test MSE of the E2E model

The previously implemented *DataWindow* class was adapted to handle the different input shapes. As each timestep consists of (256×128) spatial points and each window of data, therefore, consumes a lot of memory, the batch size was set to 24 which yields $\approx 100\%$ GPU utilization. In fig. 4.17, the train and test MSE is plotted for 5000 epochs of training with a learning rate of $1 * 10^{-4}$. As the E2E training required significant temporal resources, only one training was carried out. After a few epochs, the model seems to be stuck at an MSE of around $1 * 10^{-2}$ but manages to accelerate the learning process again until 1200 epochs, where a sharp kink with subsequent flattening of the loss curve can be observed. Compared to the sequentially trained model, the final train losses are similar, while the test loss of the E2E model is significantly lower.

Chapter 5

Results

To assess the performance of the sequentially trained ROM components, they are applied to predict buffet pressure distributions not included in the training at $Ma_\infty = 0.84$ and $Re = 12.9\text{Mio.}$. For SVD and CNN-VAE, the test flow conditions consist of the angles of attack $[3.0°, 5.0°]$ since these models were trained on the remaining angles of attack $\alpha = [1.5°, 3.5°, 4.0°, 4.5°, 6.0°]$ for this specific Reynolds/Mach number setting. Subsequently, the FC and LSTM models are evaluated using the $\alpha = 4.00°$ flow condition. Finally, the performance of the end-to-end trained CNN-VAE-LSTM is compared against its sequentially trained counterpart.

5.1 Dimensionality Reduction

The central purpose of applying SVD and CNN-VAE is to map complex high-dimensional spatiotemporal data to a lower-dimensional reduced/latent space. The primary goal of implementing these methods in the present work is to maintain the essential physical characteristics of the buffet while achieving dimensionality reduction. In the context of the forward evolution of the compressed representation, the term mode coefficients, as introduced in section 3.1.1, is also used to describe the temporal evolution of the latent dimensions generated by the CNN-VAE model. After analyzing the SVD based on the test cases, a similar evaluation is conducted for the CNN-VAE model.

5.1.1 SVD

As detailed in section 4.2.1, the parameter analysis identified rank 300 as a favorable balance between dimensionality reduction and variance explanation with regard to the buffet characteristic. Thus, both test datasets were reduced and reconstructed using the first 300 POD modes.

In fig. 5.1, an experimental c_p-snapshot of the $\alpha = 3.00°$ flow condition is compared to the reconstructed snapshot obtained using the truncated left singular vectors from the SVD. The relatively low MSE indicates a favorable agreement, with minor deviations observed primarily at the shock front. Notably, the SVD-reconstructed snapshot exhibits reduced noise compared to the original experimental data, particularly downstream of the shock. This suggests that the truncation process effectively filters out higher-frequency noise modes, which could potentially originate from measurement errors or vortices forming in the wake of the shock wave, causing pressure fluctuations. A similar pattern is observed for the $\alpha = 5.00°$ flow condition, as depicted in fig. 5.2. While deviations seem slightly larger in this case, they can be traced back to the larger variance of the second test case. Therefore, the associated MSE values are naturally higher. Nonetheless, it is important to note that based on R^2 values shown in fig. 4.5, the reconstruction for the second test case actually explains more variance.

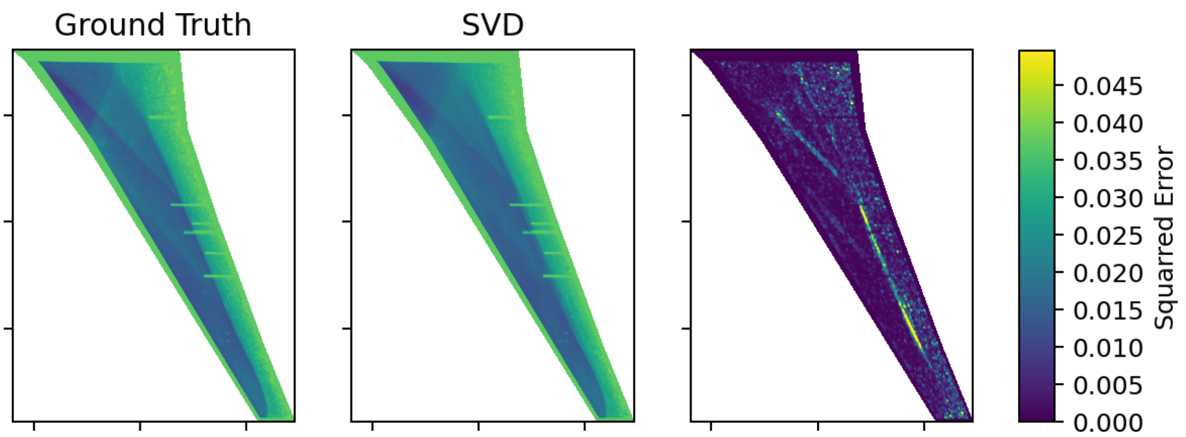


Figure 5.1: Comparison of original and SVD reconstructed c_p-snapshot ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 3.00°, \tau = 53.69$)

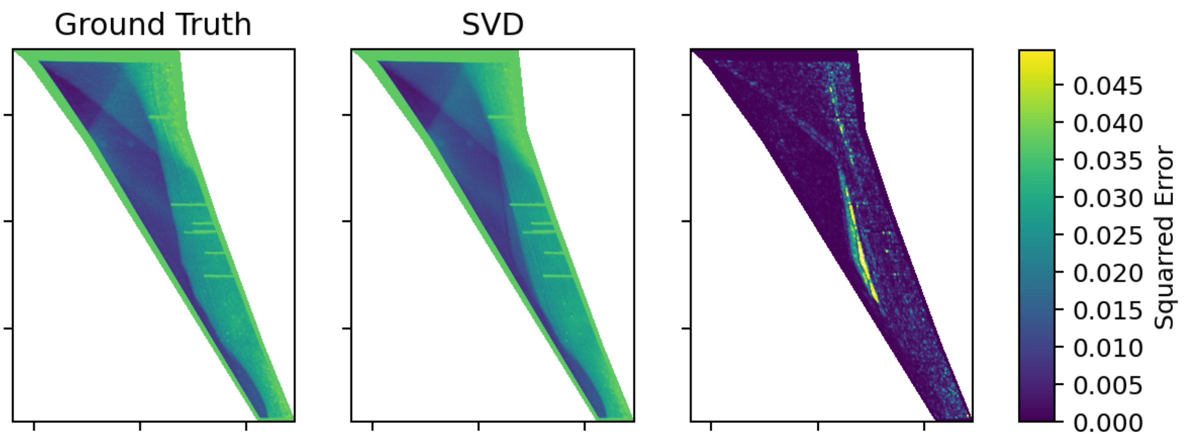


Figure 5.2: Comparison of original and SVD reconstructed c_p-snapshot ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 5.00°, \tau = 53.69$)

To assess how well buffet characteristics are captured, the PSDs of selected mode coefficients are compared. This involves a modal analysis by means of a POD, i.e. computing an SVD for both experimental and reconstructed data. The mode coefficients are then transformed into the frequency domain using FFT, from which the PSDs are derived. In the last step, the frequency is replaced with the Strouhal number St (refer to section 2.1) using mean chord length and flow velocity.

Fig. 5.3 displays the power spectra of the initial six mode coefficients for the $\alpha = 3.00°$ flow condition. While this subset of mode coefficients may not reflect the entire dataset, it significantly influences the buffet cycle as it represents the most dominant modes. It is also important to note that modes with higher indices can exhibit considerable noise due to the experimental nature of the data, making their reconstruction less desirable. The logarithmic representation of the Strouhal number (St) effectively captures a broad frequency range, with particular emphasis on lower frequencies that align with structural eigenfrequencies. In the context of the underlying flow condition, a Strouhal number of 10^{-2} corresponds to a frequency of approximately $11Hz$. According to the Nyquist-Shannon theorem, the maximum frequency that can be accurately acquired from the mode coefficients is half the sample rate, thus $1000Hz$.

The y-axis is logarithmically scaled to enhance the visibility of both small and large values.

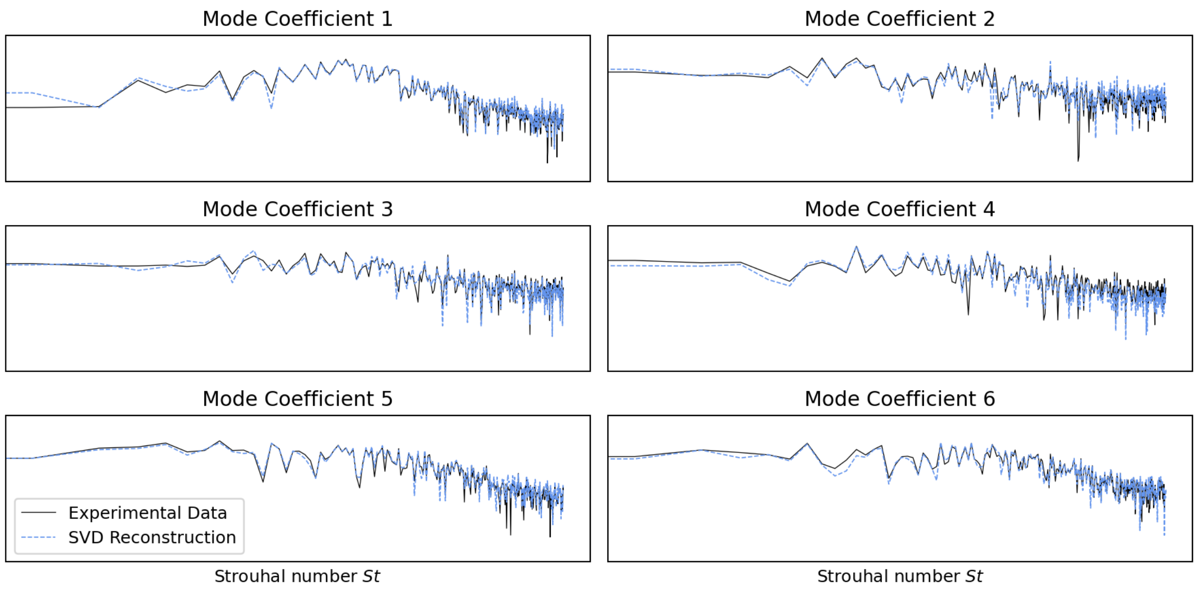


Figure 5.3: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the SVD reconstructed dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 3.00°$)

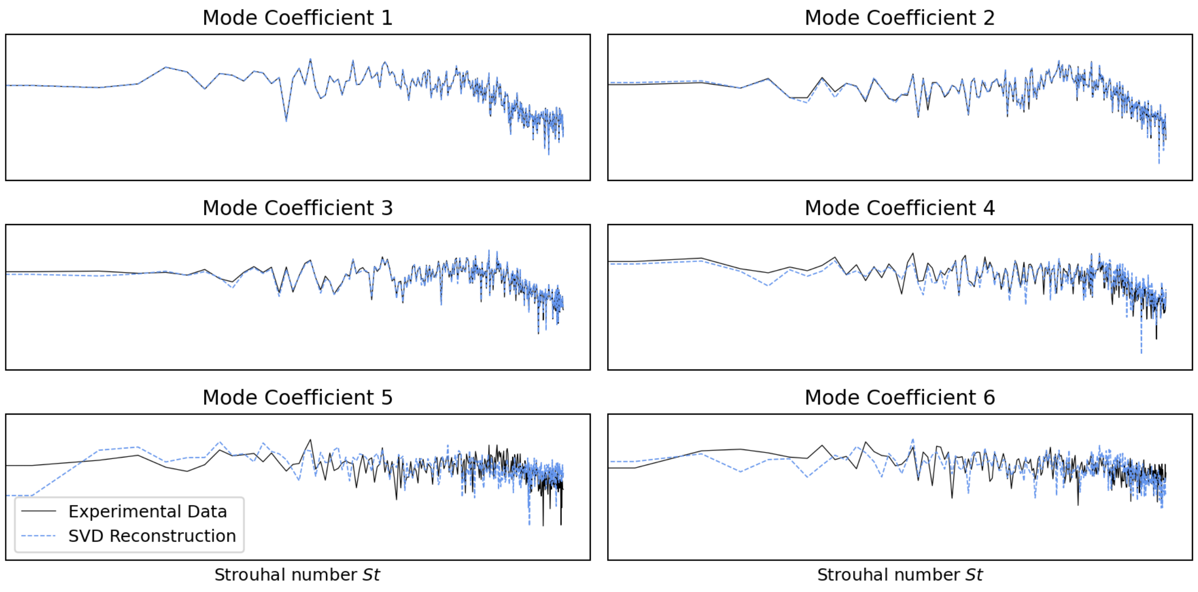


Figure 5.4: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the SVD reconstructed dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 5.00°$)

Therefore, deviations at higher y-value regions are more significant than deviations in the lower y-axis range. Overall, the reconstructed mode coefficients match the true coefficients, particularly in the lower frequency ranges. Deviations become more apparent at higher frequencies. The same pattern holds for the $\alpha = 5.00°$ test case, depicted in fig. 5.4. Notably, the agreement of mode coefficients is even closer compared to the first test case, as previously indicated by the larger R-squared value associated with the second test case (as detailed in 4.2.1). These results indicate that the SVD is a viable method for constructing ROMs as it effectively reproduces the physical behavior, maintains a limited amount of noise, and significantly reduces computational resources.

5.1.2 CNN-VAE

Similar to the previous section, the CNN-VAE model was employed to reconstruct the two test datasets. In this case, the dimension of the latent space is determined by the size of the bottleneck layer within the CNN-VAE model, which consists of 128 neurons. Notably, this is approximately half the size of the reduced space spanned by the modes of the SVD.

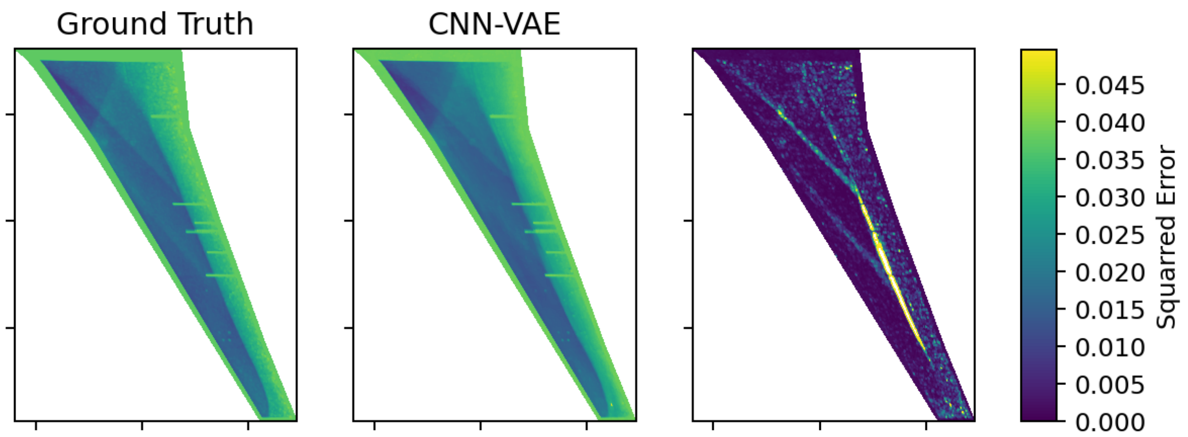


Figure 5.5: Comparison of original and CNN-VAE predicted c_p-snapshot ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 3.00°, \tau = 53.69$)

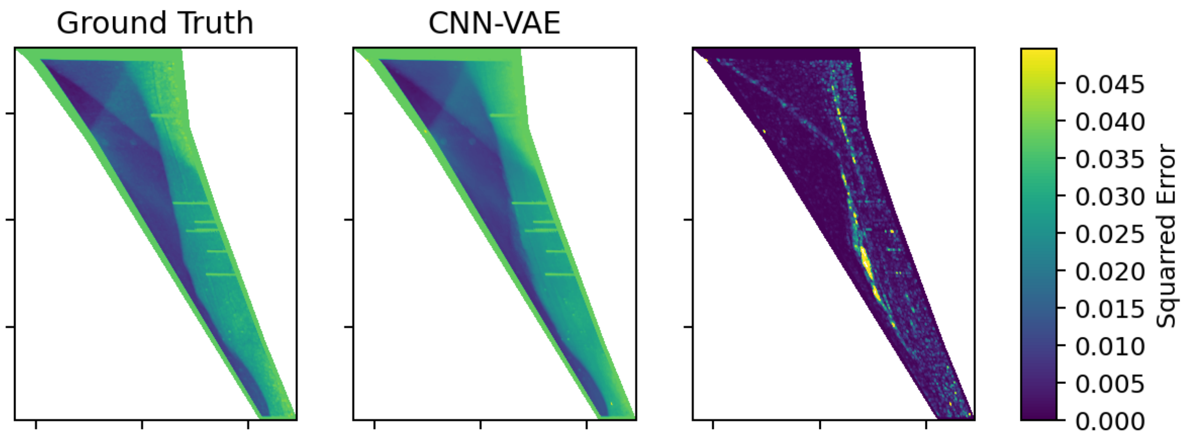


Figure 5.6: Comparison of original and CNN-VAE predicted c_p-snapshot ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 5.00°, \tau = 53.69$)

In fig. 5.5 and fig. 5.6, a comparison between the original and predicted c_p-snapshot for the two test flow conditions is depicted. In general, there is a good agreement between the ground truth and the prediction. Similar to the SVD approach, the shock fronts are slightly smeared in the regions of high variance and exhibit minor displacements compared to the original snapshots. The CNN-VAE model also effectively reduces the presence of noise in the experimental data. However, when compared to the SVD, the overall reconstruction performance of the CNN-VAE model appears to be marginally lower.

This observation made on individual c_p-snapshot is further supported by comparing the power spectra of the complete test datasets and with their corresponding CNN-VAE reconstructions in fig. 5.7 and fig. 5.8. While the overall trend of the true mode coefficients is reproduced accurately, the deviations from the experimental data are slightly larger. Specifically, for the $\alpha = 3.00°$ test case (depicted in fig. 5.7), the coefficient of the first and most dominant mode does not accurately match both the lower and higher frequency contents. However, the agreement for the $\alpha = 5.00°$ flow condition is satisfactory.

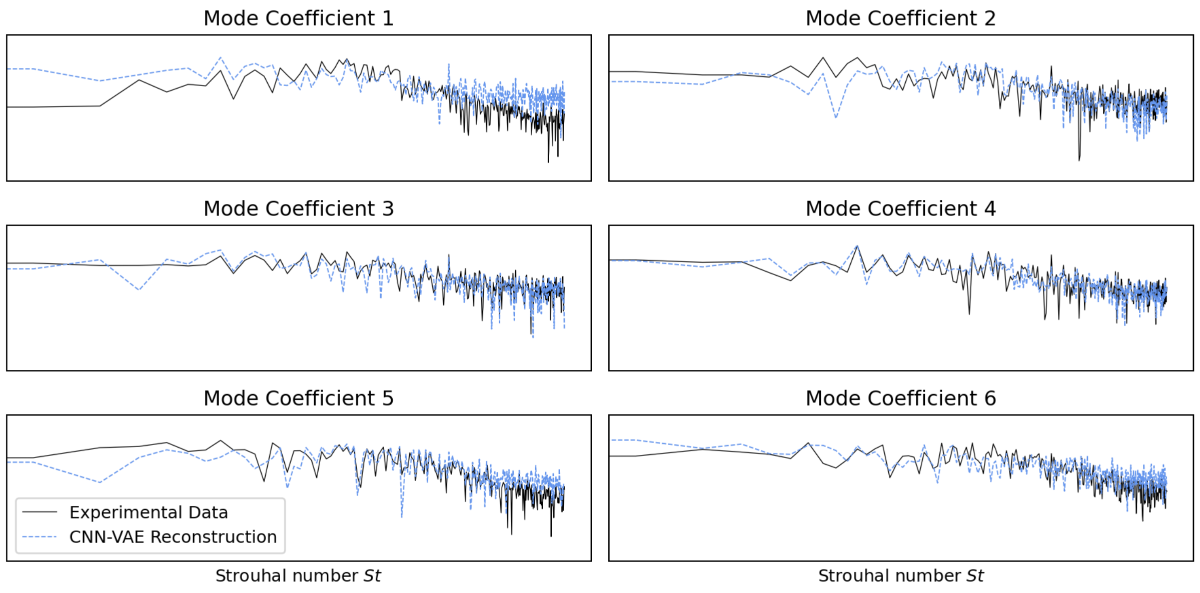


Figure 5.7: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the CNN-VAE reconstructed dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 3.00°$)

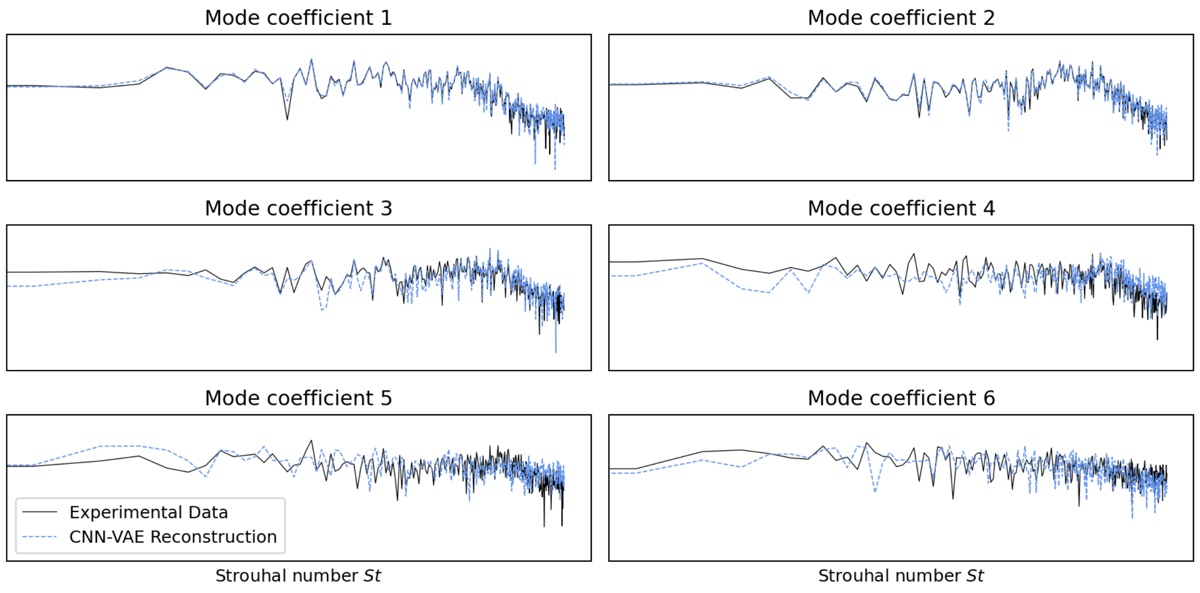


Figure 5.8: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the CNN-VAE reconstructed dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 5.00°$)

In comparison to the previous study conducted by Zahn et al. [45], the CNN-VAE model used in this work doesn't achieve the same level of precision in matching each mode coefficient as the model in the mentioned study. However, the performance of the current model is still sufficient. The differences in performance could be attributed to several factors, one of them being the underlying dataset. In the earlier study, the training involved just two flow conditions, with double the number of snapshots (1000) each, and one additional condition for testing. This setup might have enabled the model to better generalize due to the more compact dataset.

In contrast, the current study employed a broader spectrum of flow conditions, with fewer snapshots for each. This increased diversity might have impacted the model's ability to generalize to unseen flow conditions. Nevertheless, while the present model might exhibit a slightly lower accuracy on test data, it holds the potential for predicting a broader spectrum of flow conditions. Additionally, when considering validation data, the current model exhibits better performance compared to the previous one, indicating that the adjustments made to the model architecture and training process had a positive impact after all.

5.2 Reduced-Order Models

In this section, the focus shifts to the evaluation of the time-evolution models. The main goals here are the accurate reproduction of the reduced/latent space dynamics and the effective representation of buffet characteristics when reconstructed/decoded back to the full-dimensional space. As mentioned earlier, the FC and LSTM models were trained on 80% of the data from the $\alpha = 4.00°$ flow condition, and their performance is assessed in the following sections using the remaining 20% of the data.

5.2.1 SVD-FC

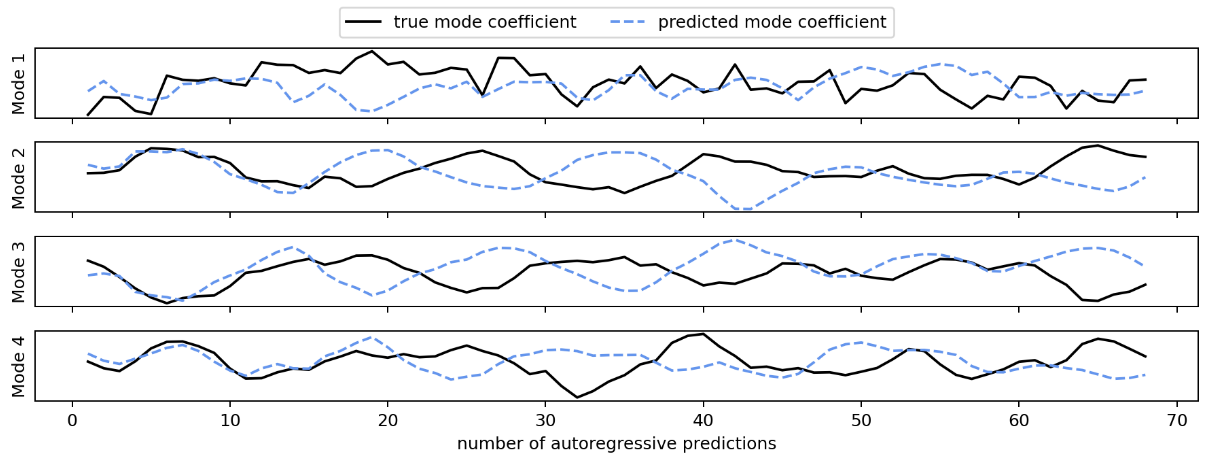


Figure 5.9: Comparison of true and SVD-FC predicted mode coefficents ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

The SVD-FC model was applied to the test case in an auto-regressive single-step prediction mode, using an initial input sequence of 32 timesteps. In fig. 5.9, the first four mode coefficients are plotted against the SVD-FC predictions. It is evident that, apart from the first mode coefficient, the model is able to accurately predict the remaining mode coefficients up to a prediction horizon of about 16 timesteps. At this point, the first half of the input sequence consists of true labels and the second half is predictions. Notably, the first mode coefficient does not align well with the predictions, indicating that the model struggles to capture its dynamics effectively. It is important to note that among the four coefficients, the first mode seems to exhibit the most frequent changes over time, which the model is not able to reproduce. Given that SVD-based mode coefficients are ordered in terms of descending importance, the quality of the first mode coefficient's representation significantly affects the ability to reproduce the full space buffet dynamics. Although the true dynamics, observable through the interaction of opposing maxima and minima, were not predicted accurately in this specific data window, the model still manages to learn similar dynamics, as the predicted mode coefficients also interact with each other.

Fig 5.10 illustrates a comparison of the FC model's loss, denoted reduced space loss, and the loss

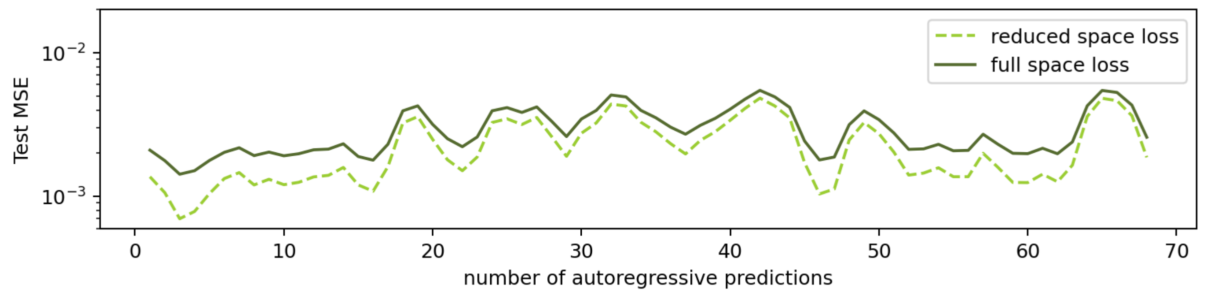


Figure 5.10: Comparison of latent loss and full space loss over 68 auto-regressive predictions of SVD-FC model ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

between the SVD reconstructed prediction and the corresponding original c_p-snapshot, referred to as full space loss. Throughout the entire prediction horizon, the reduced space loss remains lower than the full space loss. This indicates that the ROM approach is suitable for capturing information in a lower-dimensional space and therefore provides efficiency gains, resulting in faster computations and reduced memory requirements. Moreover, the close matching of both loss curves suggests that the SVD is preserving the important information when reconstructing back to the full space. Furthermore, both losses follow an upward trend over time, confirming the missing long-term stability of the SVD-FC model, as previously observed in fig. 5.9.

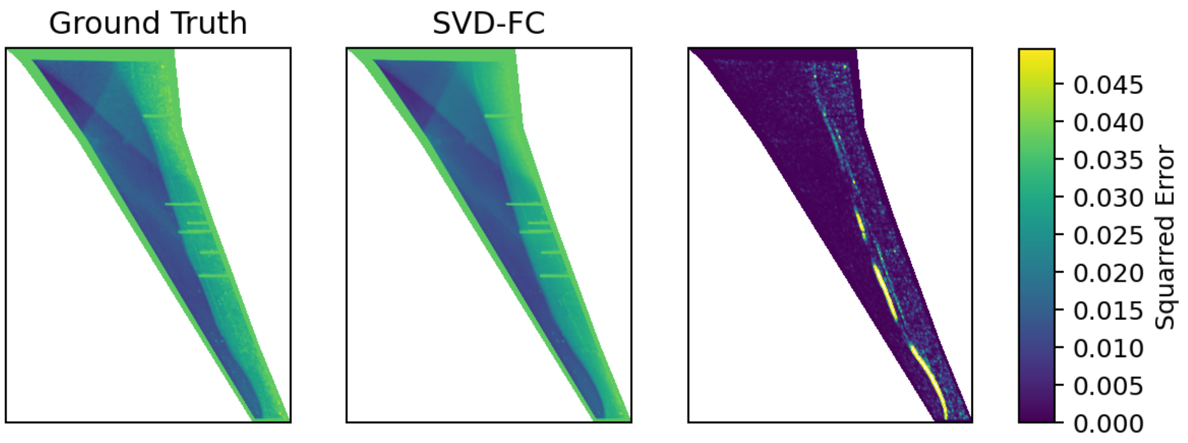


Figure 5.11: Comparison of original and SVD-FC predicted c_p-snapshot ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°, \tau = 258.25$ (50 timesteps ahead))

Fig. 5.11 represents a comparison between a true c_p-snapshot and the predicted snapshot using the SVD-FC model. The prediction is generated using an input sequence of 32 timesteps and a prediction horizon of 50 timesteps. As evident in fig. 5.11, the predicted c_p-snapshot is generally in good agreement with the ground truth and displays characteristic features. However, it is important to note that while snapshot comparisons are informative, they are limited in their ability to draw conclusions without considering the temporal context. Therefore, in order to gain more insights into the test case dynamics, the power spectra of the SVD-FC predicted dataset are compared to the corresponding experimental data in fig. 5.12. The initial sequence of 32 timesteps was ignored, leaving the remaining 68 timesteps for comparison.

In this context, the power spectra of the first six mode coefficients provide valuable information about the buffet dynamics at a higher level. The analysis reveals that the predicted dynamics exhibit considerable deviations from the ground truth for most mode coefficients. The power of the low-frequency content tends to be overestimated while the power of the mid- and high-frequency content tends to be underestimated. This behavior suggests that the SVD-FC model excels in capturing the steady-state and more persistent buffet components while struggling to capture transient behavior, characterized by rapid variations.

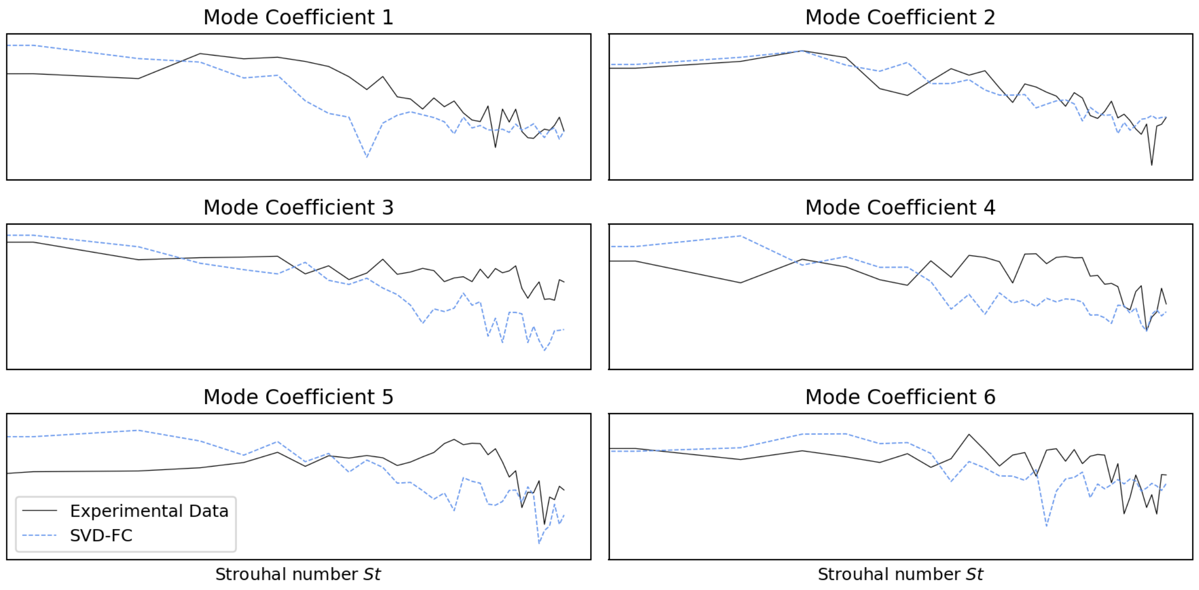


Figure 5.12: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the SVD-FC predicted dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

As part of the optimization process, the reconstructed test dataset was animated and compared against the ground truth to confirm the conclusions drawn from the PSDs. The animation was created by treating individual timesteps as frames and concatenating them to a Graphics Interchange Format (GIF) image. While the general behavior of the shock wave appeared physically accurate in terms of interaction, there were noticeable differences in the dynamics. The reconstructed dynamics appeared smeared and less pronounced compared to the ground truth. This observation matches the previous PSD findings in fig. 5.12, where overpowered lower frequencies and underpowered higher frequencies were identified. Ultimately, the lack of transient features and dominance of steady-state dynamics is reflected in the suppressed propagation of buffet cells toward the wing tip.

5.2.2 CNN-VAE-FC

Similar to the previous section, the CNN-VAE-FC model was employed for the test case using an auto-regressive single-step prediction approach. The mode coefficients, representing the latent dimensions of the CNN-VAE model, were predicted and compared to the true coefficients in fig. 5.13. In contrast to the SVD-FC model, each mode coefficient's temporal evolution is equally important in the case of the CNN-VAE-FC model.

Examining the figure, it becomes evident that the predictions only match the true coefficients for the first one or two predictions. In this aspect, the SVD-FC model seems to outperform the CNN-VAE-FC model. Furthermore, it is worth noting that the CNN-VAE-FC model encountered challenges in achieving as low train and test losses as the SVD-FC model, as elaborated in section 4.3.1, indicating that the relationships within the fewer latent dimensions might be more complex, resulting in a more challenging training process.

In fig. 5.14, the reduced space loss and the full space loss over 68 auto-regressive predictions using the CNN-VAE-FC model are plotted. Similar to the SVD-FC case, the loss curve shapes

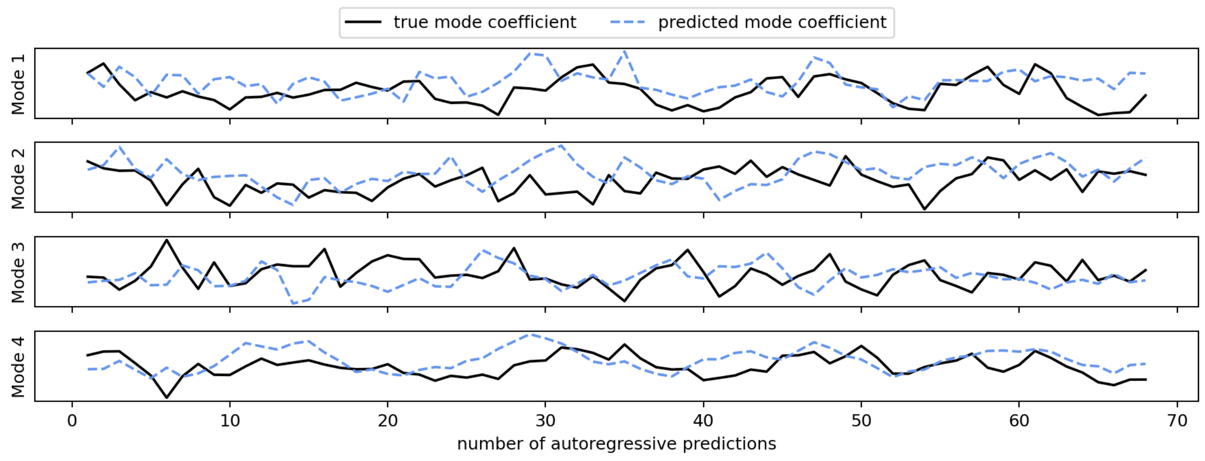


Figure 5.13: Comparison of true and CNN-VAE-FC predicted mode coefficents ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

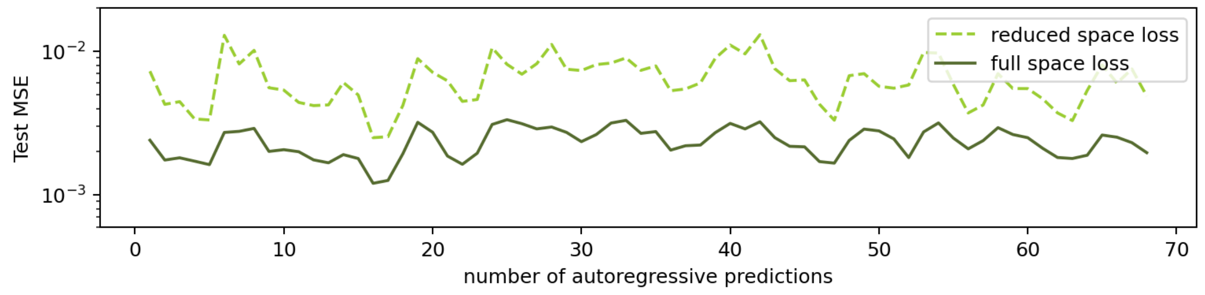


Figure 5.14: Comparison of latent loss and full space loss over 68 auto-regressive predictions of CNN-VAE-FC model ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°, \tau = 266.30$)

closely match, indicating that the CNN-VAE's decoding process effectively preserves most of the information. However, in contrast to fig. 5.10, the reduced space loss is consistently larger in the case of the CNN-VAE-FC model. This suggests that the model's predictions do not match the test data as well as the SVD-FC model. Nevertheless, the full space loss after the decoding process is similar, implying that both models provide equally good representations of the test data in terms of predicting individual c_p-snapshots.

Moving from matching predictions to accurately reflecting buffet characteristics, a comparison of the PSDs of the first six mode coefficients is depicted in fig. 5.15. The mode coefficients of the CNN-VAE-FC predicted dataset demonstrate a significant improvement in comparison to the previous model. The trend of each mode coefficient is matched more accurately, and the overestimation of lower frequencies is less pronounced. Nevertheless, the figure also indicates that some of the transient features may be overpowered.

Similarly, the temporal evolution of the CNN-VAE-FC reconstruction was visualized. Although the dynamics of the shock are more accurately captured, including transient behavior, they appear less physically coherent. In the context of buffet cell propagation, a standing wave forms initially causing a flickering effect in the pressure distribution across the wing. This behavior persists for 15-20 timesteps before the dynamics stabilize and converge towards the ground truth. Although the CNN-VAE-FC model improves upon the accurate balance of buffet dynamics compared to the SVD-FC model, the predicted dynamics still appear less smooth and coherent.

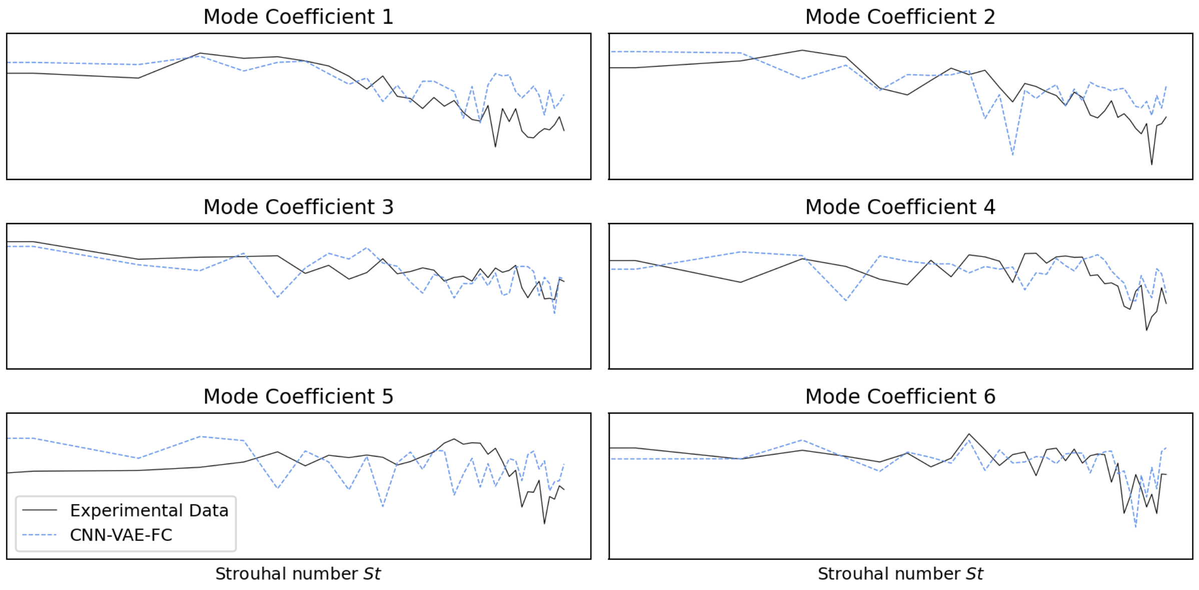


Figure 5.15: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the CNN-VAE-FC predicted dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 5.00°$)

5.2.3 SVD-LSTM

During the optimization process, the SVD-LSTM model achieved the lowest training loss while maintaining a moderate test error. In fig. 5.16, the initial mode coefficients of the reduced dataset are plotted against 68 auto-regressive SVD-LSTM predictions. The corresponding losses for each prediction and the full space loss after reconstruction are depicted in fig. 5.17.

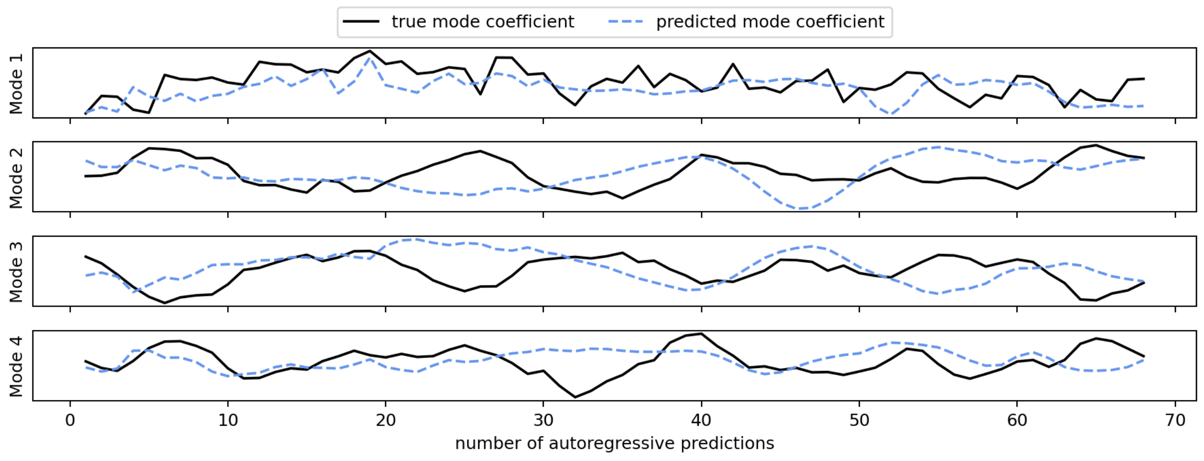


Figure 5.16: Comparison of true and SVD-LSTM predicted mode coefficents ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

The first mode coefficient is predicted with reasonable accuracy. For the remaining three mode coefficients shown in the figure, the agreement is less accurate. While the first few timesteps are well matched, the predicted wavelengths of the mode coefficient dynamics become too large afterward. Therefore, it is expected that the corresponding POD modes in the full space are not represented accurately, as their temporal behavior is not matched. Nevertheless, the LSTM was able to learn the inherent dynamics of the training data, evident in the interaction of the coefficients.

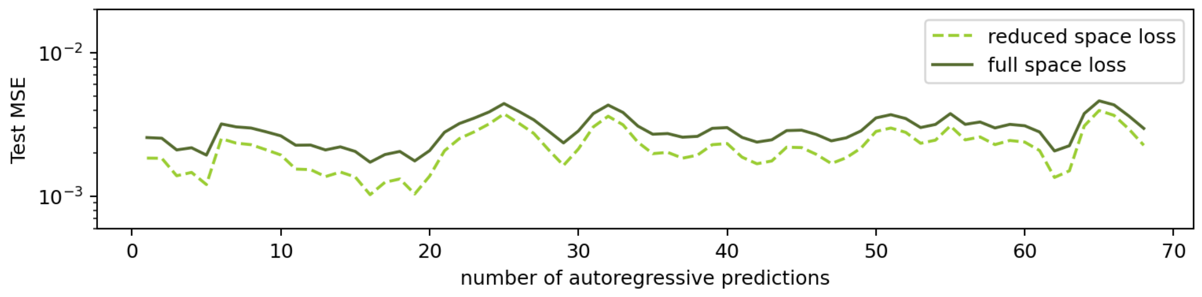


Figure 5.17: Comparison of latent loss and full space loss over 68 auto-regressive predictions of SVD-LSTM model ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

In terms of the power spectra, as shown in fig. 5.18, the SVD-LSTM model manages to capture the majority of buffet dynamics quite accurately. However, it becomes evident in for of the mode coefficients that the predicted spectra tend to be unbalanced, with stronger low-frequency content and less dominant mid- and high-frequency content. This behavior is similar to what was observed in the SVD-FC model, although less pronounced. The unbalanced frequency contents are also noticeable in the animated reconstruction, in which slow-moving features appear to be slightly more dominant.

The initial sequence of outward propagating buffet cells is captured accurately, which is a positive outcome. However, as predictions extend further into the future, they seem to converge towards more steady-state dynamics. The general behavior of the shock wave appears smeared and not as precise as in the ground truth data. It is worth noting that this behavior is consistent with what was observed in the SVD-FC model, suggesting that the limitations of capturing transient behavior might be attributed to the SVD-based dimensionality reduction approach.

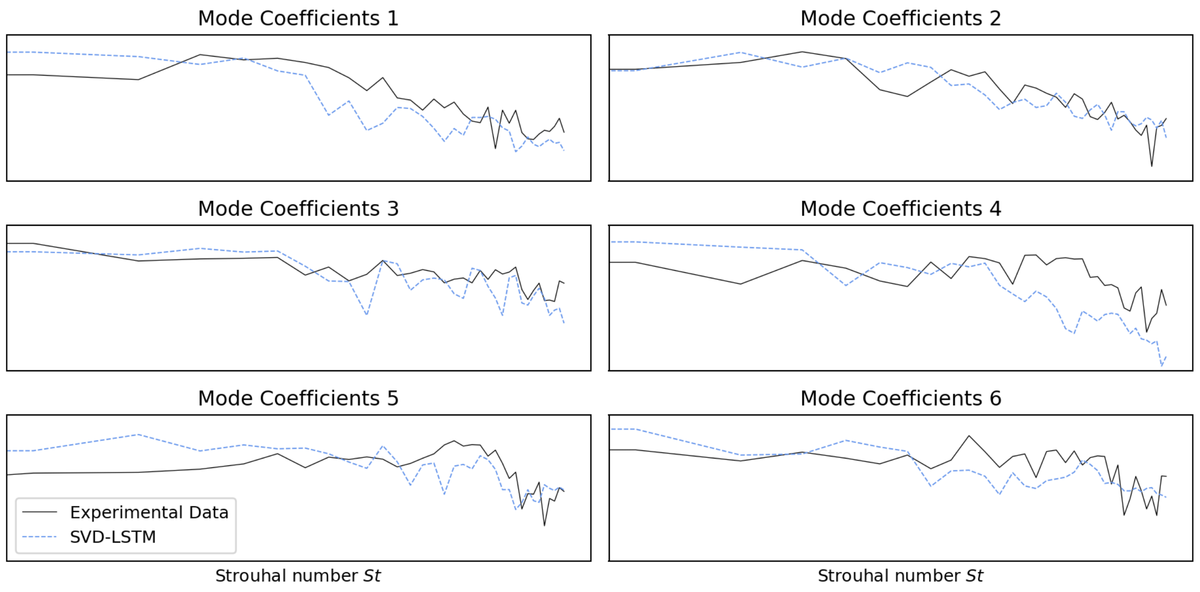


Figure 5.18: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the SVD-LSTM predicted dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

5.2.4 CNN-VAE-LSTM

Following the previous evaluation process, the CNN-VAE-LSTM model is employed in a recurrent single-step prediction mode on the test data. Fig. 5.19 illustrates predictions and ground truth for the initial four latent dimensions/mode coefficients. In comparison to the CNN-VAE-FC model, it is evident that the LSTM model provides a closer match to the dynamics of the true coefficients, although not an exact reproduction of the true values is achieved.

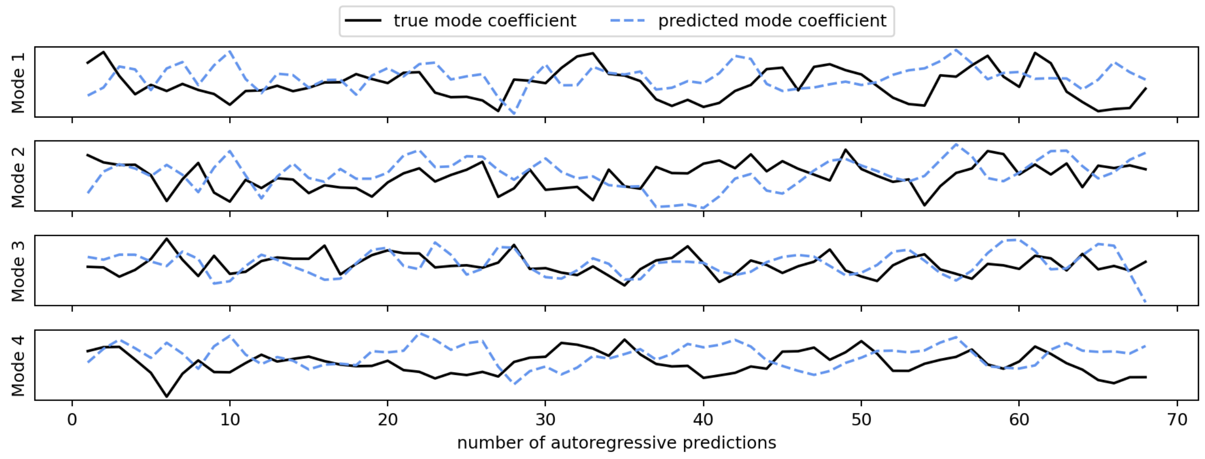


Figure 5.19: Comparison of true and CNN-VAE-LSTM predicted mode coefficents ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

In fig. 5.20, reduced space loss and full space loss are compared. Notably, a pattern similar to that observed in the corresponding fig. 5.14 of the CNN-VAE-FC model is visible. Altogether, this implies that the reduced space loss is consistently larger for the time-evolution models trained with the encoded data from the CNN-VAE. The models trained with the mode coefficients obtained from SVD exhibit significantly lower test losses, which are also lower than the final full space loss. However, these findings shouldn't be interpreted as a direct indication of more accurate buffet dynamics reproduction. This could suggest that the intricate temporal relationships within the latent space, generated by the CNN-VAEs, are more challenging to learn. Nevertheless, the CNN-VAE-based ROMs still outperform the SVD-based ones. The lower error observed in the SVD-ROMs might stem from their enhanced representation of the more persistent buffet components. Therefore, the deviations from the ground truth appear lower, although the dynamics are not reproduced accurately. It's worth noting that the reconstruction of predicted mode coefficients potentially also introduces some smearing of the shock wave oscillations, although the $\alpha = 4.00°$ flow condition was used for training the SVD. In contrast, the CNN-VAE ROMs exhibit more pronounced dynamics, resulting in greater deviations from the ground truth, despite better dynamics reproduction.

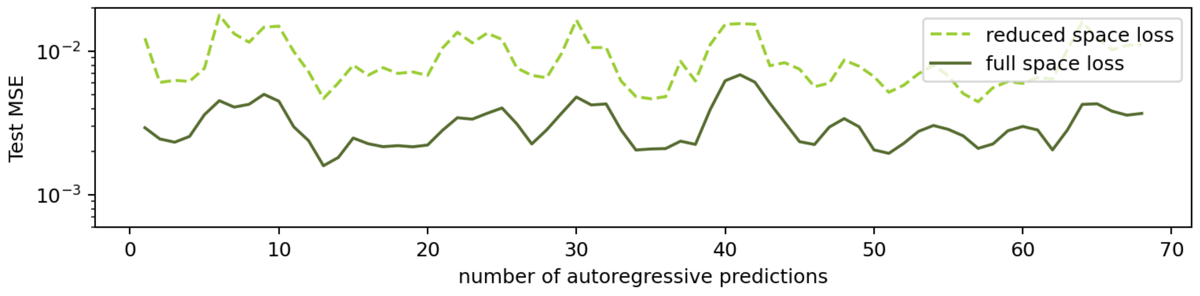


Figure 5.20: Comparison of latent loss and full space loss over 68 auto-regressive predictions of CNN-VAE-LSTM model ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

This observation is reinforced by fig. 5.21, where the PSDs of the first six mode coefficients of the test dataset and the CNN-VAE-LSTM predicted test dataset are displayed. Evidently, there is a high level of agreement with the experimental data, indicating the model's capability to capture both steady-state patterns and transient buffet dynamics effectively.

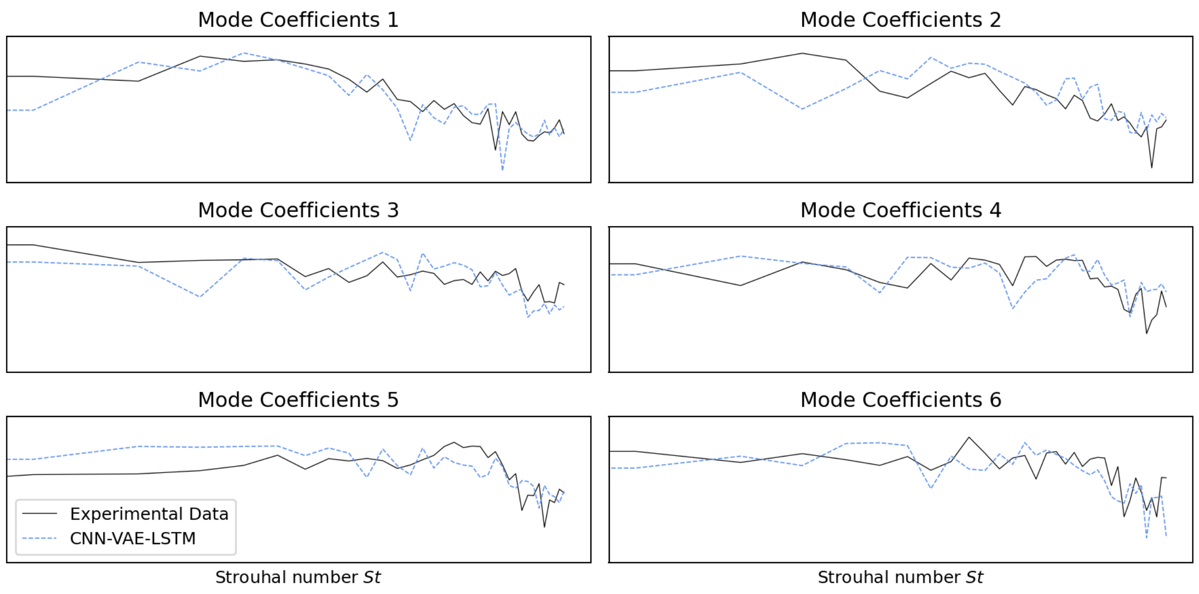


Figure 5.21: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the CNN-VAE-LSTM predicted dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

This expected behavior is further confirmed by creating an animation from the predictions. Here, the predicted evolution closely aligns with the experimental data, accurately reproducing both the low-frequency and transient buffet features. Among all the models assessed previously, the CNN-VAE-LSTM model stands out for its superior performance in capturing buffet characteristics. It not only replicates the general underlying dynamics, as seen in previous models to a certain degree, but it also demonstrates an impressive agreement between the auto-regressively predicted time series and the actual temporal evolution over 68 timesteps. Therefore the 3D buffet phenomenon is fully captured.

5.3 End-To-End CNN-VAE-LSTM

The CNN-VAE-LSTM model, which demonstrated superior performance among all evaluated ROMs, was selected for end-to-end training. In this section, the end-to-end trained model (E2E) is assessed and contrasted with the sequentially trained CNN-VAE-LSTM model, which will be referred to as the sequential model. Following the previous methodology, the E2E model was utilized to reconstruct the test case involving the $\alpha = 4.00°$ flow condition.

A comparison of the full space losses between the sequential and E2E model is illustrated in fig. 5.22. Notably, both fall into a similar value range, with the E2E model's loss showing slightly less variation and a slightly higher average. In fig. 5.23, the predicted PSDs of both models and the ground truth are depicted. The E2E-predicted spectra appear smoother in comparison. While they do show agreement with the ground truth, this indicates that there are almost no dynamics apparent as there is no diversity in the predictions.

Clearly, when visualizing the E2E predictions through animations, the dynamics are well-matched in the initial few timesteps but are vanishing from there in a fast decaying manner. Roughly after eight predictions, the model outputs similar pressure distributions for the remaining window with no apparent dynamics. Although an implementation error cannot be ruled out,

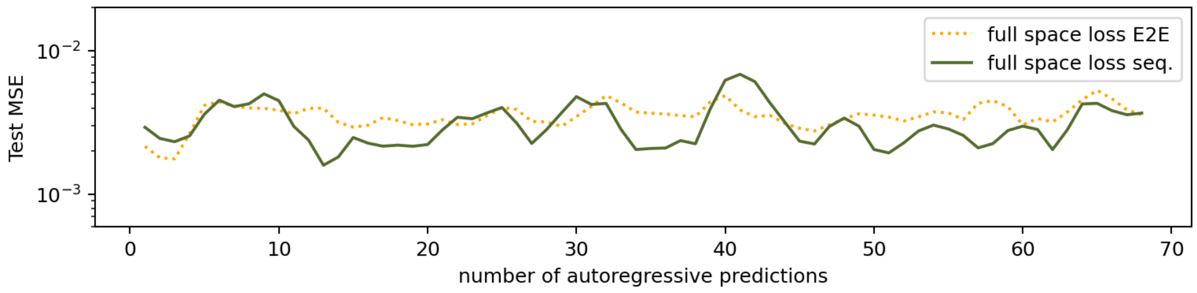


Figure 5.22: Comparison of E2E and sequential CNN-VAE-LSTM full space losses ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

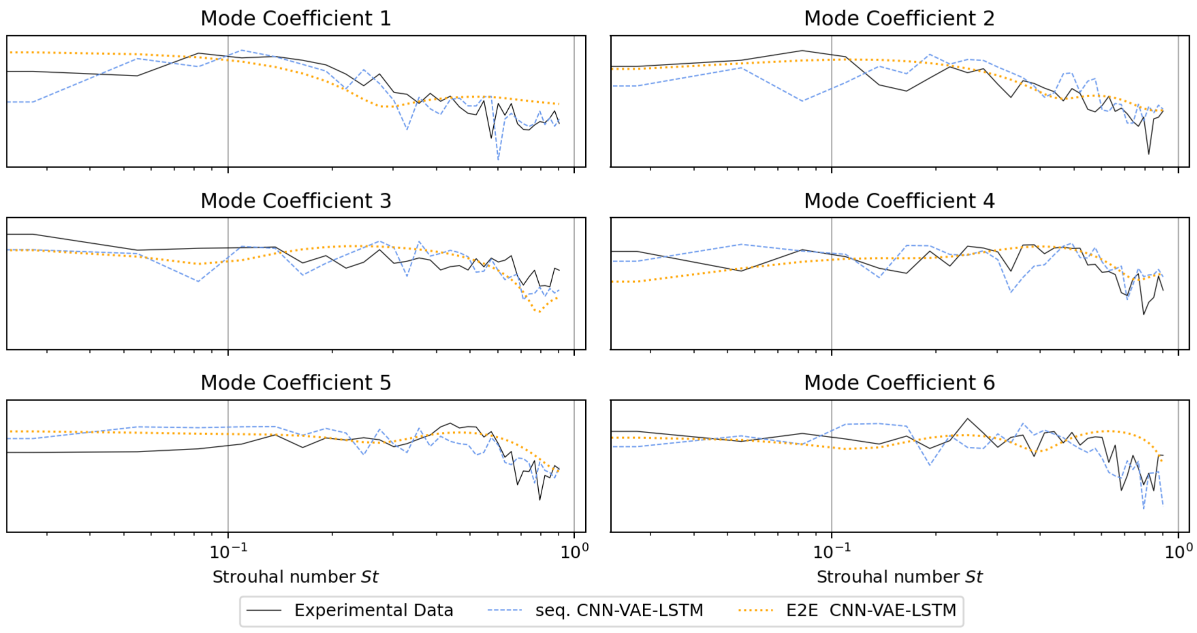


Figure 5.23: Comparative Power Spectra of the first six POD Modes. Visualizing the agreement between experimental data and the E2E predicted dataset ($Ma_\infty = 0.84, Re = 12.9\text{Mio.}, \alpha = 4.00°$)

there could be other factors responsible for this behavior. The train and test loss curves examined earlier exhibited an irregular behavior as a sharp flattening of the loss curve was observed after a certain number of epochs. Although reaching relatively low loss values, the model might struggle to maintain its internal memory beyond the extent to which it was trained, which in this case was a prediction horizon of 3.

Given the unsatisfactory results obtained with the E2E model, further investigation and optimization is required. It is important to note that optimizing an E2E model can be significantly more complex due to the simultaneous training of all its components, demanding more computational resources and expertise. Additionally, it is worth noticing that the CNN-VAE models tend to converge faster compared to the LSTMs used in this study, with the latter requiring approximately ten times the number of epochs to achieve desirable performance. Verifying if better results can be achieved after an extensive parameter is an essential step. Nevertheless, it must be determined if the tremendous time expenditure outweighs the potential benefits.

Chapter 6

Conclusion

In the present study, the training and comparison of a set of ROMs has been carried out for predicting pressure distributions based on IPSP data acquired during a wind tunnel campaign at the ETW. The campaign involved measurements of the Airbus XRF-1 configuration under transonic buffet flow conditions, characterized by strong variations in aerodynamic forces and potentially critical structural vibrations (buffeting).

The ROMs developed in this study combine dimensionality reduction techniques, specifically SVD and CNN-VAE, with FC and LSTM neural networks to predict the temporal evolution of reduced/encoded spatiotemporal data. The data was obtained from a flow condition with $Ma_\infty = 0.84$ and $Re = 12.9$Mio., for which seven angles of attack were recorded. SVD and CNN-VAE were trained using data from five angles of attack and then coupled with FC and LSTM networks, individually trained on a single angle of attack, to predict buffet dynamics.

The evaluation of SVD and CNN-VAE reconstruction performance was carried out on two unseen test flow conditions, involving a comparison of measured c_p-snapshots with their corresponding reconstructions. Additionally, the ability of SVD and CNN-VAE to capture the buffet characteristics was assessed by decomposing results and ground truth by means of a POD. An FFT of the first six mode coefficients was computed to analyze their power spectra. The evaluation indicated that SVD performs marginally better when employed on the test cases. Nevertheless, both techniques were able to capture the inherent variance accurately with $97\% \leq R^2 \leq 99\%$.

After training the FC and LSTM neural networks for auto-regressive predictions, the sequentially trained ROMs were applied to predict buffet dynamics for the unseen part of the test case. The evaluation involved assessing the reproduction of temporal relationships of the reduced state and examining the relationship between the reduced space loss and the full space loss. Additionally, the ROMs were employed to predict the test case using auto-regressive single-step predictions to discuss the PSDs. The study suggested that the SVD ROMs tend to provide more accurate predictions of reduced space dynamics. This was indicated by predictions closely matching the true mode coefficients, as well as a reduced space loss that was about an order of magnitude lower than that of CNN-VAE ROMs. Notably, the full space loss across all models fell within a similar range. However, these findings shouldn't be interpreted as a direct indication of more accurate buffet dynamics reproduction. A possible interpretation is that the complex temporal relationships within the latent space, generated by the CNN-VAE, are more challenging to learn. In contrast, the temporal coefficients derived from SVD inherently combined both slower-changing, more persistent components and rapidly changing short-term components.

When analyzing the PSDs of SVD-FC and SVD-LSTM, it becomes apparent that these models tend to overestimate lower-frequency contents while underestimating higher-frequency contents.

This suggests that they predominantly learned steady-state dynamics from data. Visualizing the predicted datasets of the SVD ROMs as animations reveals minimal transient behavior, causing the characteristic shock wave movement to appear blurred or smeared. In contrast to the SVD-FC model, the SVD-LSTM model manages to capture the outward propagation of buffet cells to some extent, indicating a limited ability to represent transient behavior.

The ROMs employing CNN-VAE for encoding and decoding provide a more precise representation of shock wave oscillations, evident from their more balanced PSDs. Among all ROMs, the CNN-VAE-LSTM stands out as it aligns closely with the experimental data's PSDs. When viewed through animations, it becomes evident that the CNN-VAE-LSTM model not only replicates the general underlying dynamics, as seen in previous models to a certain degree, but it also demonstrates an impressive agreement between the auto-regressively predicted time series and the actual temporal evolution over 68 timesteps. Hence, the CNN-VAE-LSTM model emerges as the most effective ROM in terms of capturing and representing buffet dynamics, both in terms of transient features and steady-state oscillations.

For that reason, the CNN-VAE-LSTM model was employed in an end-to-end training approach in which CNN-VAE and LSTM are trained concurrently to investigate if such an approach could further improve performance. The results, however, showed a decaying dynamics representation and potential stability issues in the E2E model, particularly over extended prediction horizons. While the integrated training methodology could improve overall performance, further investigations and optimizations are crucial to achieve the desired predictive accuracy.

In terms of future work, there are several potential approaches that can enhance the capabilities and insights provided by the developed ROMs. A first step could be applying the time-evolution models for multiple flow conditions to investigate the robustness of different architectures when dealing with a more comprehensive dataset. Additionally, more sophisticated model architectures such as time-convolution FC/LSTM networks and transformer networks could be explored. Graph Neural Networks could also be investigated, to learn a graph-based representation of data and potentially provide complex relationships between different aerodynamic features. In terms of data augmentation and interpolation, employing Generative Adversarial Networks could help generate new c_p-snapshots to increase the size and diversity of the dataset, or to interpolate between Mach numbers and angles of attack. As the overall objective is to create an estimate of the buffet onset, integrating techniques like Monte Carlo Dropout could provide the models with the ability to estimate prediction uncertainty, which is particularly important when forecasting buffet onset conditions. Although further exploration of methods on the present dataset could aid in understanding buffet, the ultimate challenge is to transfer the learned dynamics to different wing geometries and flow conditions.

Bibliography

[1] Bagheri, R.: *Understanding Singular Value Decomposition and its Application in Data Science.* URL: https://towardsdatascience.com/understanding-singular-value-decomposition-and-its-application-in-data-science-388a54be95d (visited on 06/26/2023).

[2] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics).* Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[3] Brunton, S. L. and Kutz, J. N.: *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge University Press, 2019. DOI: 10.1017/9781108380690.

[4] Brunton, S. L., Noack, B. R., and Koumoutsakos, P.: "Machine Learning for Fluid Mechanics". In: *Annual Review of Fluid Mechanics* 52(1) (2020), pp. 477–508. DOI: 10.1146/annurev-fluid-010719-060214.

[5] Bushaev, V.: *Adam — latest trends in deep learning optimization.* URL: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c (visited on 08/19/2023).

[6] Crouch, J. D., Garbaruk, A., Magidov, D., et al.: "Origin of transonic buffet on aerofoils". In: *Journal of Fluid Mechanics* 628 (2009), pp. 357–369. DOI: 10.1017/S0022112009006673.

[7] Dandois, J., Molton, P., Lepage, A., et al.: "Buffet Characterization and Control for Turbulent Wings". In: *Aerospace Lab*(6) (June 2013), p. 1–17. URL: https://hal.science/hal-01184458.

[8] Dandois, J.: "Experimental study of transonic buffet phenomenon on a 3D swept wing". In: *Physics of Fluids* 28(1) (Jan. 2016). 016101. ISSN: 1070-6631. DOI: 10.1063/1.4937426.

[9] Dhinakaran, A.: *Understanding KL Divergence.* URL: https://towardsdatascience.com/understanding-kl-divergence-f3ddc8dff254 (visited on 07/01/2023).

[10] Eivazi, H., Veisi, H., Naderi, M. H., et al.: "Deep neural networks for nonlinear model order reduction of unsteady flows". In: *Physics of Fluids* 32(10) (Oct. 2020). ISSN: 1070-6631. DOI: 10.1063/5.0020526.

[11] Giannelis, N. F., Vio, G. A., and Levinski, O.: "A review of recent developments in the understanding of transonic shock buffet". In: *Progress in Aerospace Sciences* 92 (2017), pp. 39–84. ISSN: 0376-0421. DOI: https://doi.org/10.1016/j.paerosci.2017.05.004.

[12] Goodfellow, I. J., Bengio, Y., and Courville, A.: *Deep Learning.* http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[13] Hao, H., Wang, Y., Xia, Y., et al.: *Temporal Convolutional Attention-based Network For Sequence Modeling.* 2020. arXiv: 2002.12530 [cs.CL].

[14] Hochreiter, S. and Schmidhuber, J.: "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[15] Hornik, K., Stinchcombe, M., and White, H.: "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2(5) (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

[16] Iovnovich, M. and Raveh, D.: "Numerical Study of Shock Buffet on Three-Dimensional Wings". In: *AIAA Journal* 53 (Feb. 2015), pp. 449–463. DOI: 10.2514/1.J053201.

[17] Kumar, A.: *Data Analytics: Mean Squared Error or R-Squared – Which one to use?* URL: https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/#What_is_R-Squared (visited on 08/20/2023).

[18] LeCun, Y., Bengio, Y., and Hinton, G.: "Deep learning". In: *nature* 521(7553) (2015), p. 436.

[19] Lutz, T., Kleinert, J., Waldmann, A., et al.: "Research Initiative for Numerical and Experimental Studies on High-Speed Stall of Civil Aircraft". In: *Journal of Aircraft* 60(3) (2023), pp. 623–636. DOI: 10.2514/1.C036829.

[20] Martin, D. Q.: *Shock Buffet und aerodynamische Resonanz.* URL: https://www.dlr.de/ae/desktopdefault.aspx/tabid-9619/16552_read-36584/ (visited on 06/26/2023).

[21] Maulik, R., Mohan, A., Lusch, B., et al.: "Time-series learning of latent-space dynamics for reduced-order model closure". In: *Physica D: Nonlinear Phenomena* 405 (2020), p. 132368. ISSN: 0167-2789. DOI: https://doi.org/10.1016/j.physd.2020.132368.

[22] Olah, C.: *Understanding LSTM Networks.* URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 08/19/2023).

[23] Oord, A. van den, Dieleman, S., Zen, H., et al.: "WaveNet: A Generative Model for Raw Audio". In: *CoRR* abs/1609.03499 (2016). arXiv: 1609.03499.

[24] OpenAI: *GPT-4 Technical Report.* 2023. arXiv: 2303.08774 [cs.CL].

[25] Peixerio, M.: *Time Series Forecasting in Python.* Manning, 2022.

[26] Plante, F.: "Towards Understanding Stall Cells and Transonic Buffet Cells". AAI30161490. PhD thesis. 2020. ISBN: 9798358404144.

[27] Plante, F., Dandois, J., Sartor, F., et al.: "Study of Three-Dimensional Transonic Buffet on Swept Wings". In: *35th AIAA Applied Aerodynamics Conference.* DOI: 10.2514/6.2017-3903.

[28] Pravallika, M. S., Vasavi, S., and Vighneshwar, S. P.: "Prediction of temperature anomaly in Indian Ocean based on autoregressive long short-term memory neural network". In: *Neural Computing and Applications* 34 (2022). URL: https://doi.org/10.1007/s00521-021-06878-8.

[29] Radford, A., Wu, J., Child, R., et al.: "Language Models are Unsupervised Multitask Learners". In: (2019).

[30] Raff, E.: *Inside Deep Learning: Math, Algorithms, Models.* Manning, 2022. URL: https://books.google.de/books?id=s8hhzgEACAAJ.

[31] Rocca, J.: *Understanding Variational Autoencoders (VAEs).* URL: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 (visited on 06/30/2023).

[32] Roza, F.: *End-to-end learning, the (almost) every purpose ML method.* URL: https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4 (visited on 08/31/2023).

[33] Solera-Rico, A., Vila, C. S., Gómez, M. A., et al.: *β-Variational autoencoders and transformers for reduced-order modelling of fluid flows.* 2023.

[34] Strang, G.: *Introduction to Linear Algebra.* Fourth. Wellesley, MA: Wellesley-Cambridge Press, 2009. ISBN: 9780980232714 0980232716 9780980232721 0980232724 9788175968110 8175968117.

[35] Sugioka, Y., Koike, S., Nakakita, K., et al.: "Experimental analysis of transonic buffet on a 3D swept wing using fast-response pressure-sensitive paint". In: *Experiments in Fluids* 59 (May 2018). DOI: 10.1007/s00348-018-2565-5.

[36] Sugioka, Y., Kouchi, T., and Koike, S.: "Experimental comparison of shock buffet on unswept and 10-deg swept wings". In: *Experiments in Fluids* 63 (Aug. 2022). DOI: 10.1007/s00348-022-03482-x.

[37] Timme, S. and Thormann, R.: “Towards Three-Dimensional Global Stability Analysis of Transonic Shock Buffet”. In: June 2016. DOI: 10.2514/6.2016-3848.

[38] Tosun, N., Sert, E., Ayaz, E., et al.: “Solar Power Generation Analysis and Forecasting Real-World Data Using LSTM and Autoregressive CNN”. In: *2020 International Conference on Smart Energy Systems and Technologies (SEST)*. 2020, pp. 1–6. DOI: 10.1109/SEST48500.2020.9203124.

[39] Towards Data Science, E. H.: *How To Forecast Time-Series Using Autoregression*. 2017. URL: https://towardsdatascience.com/how-to-forecast-time-series-using-autoregression-1d45db71683 (visited on 06/06/2023).

[40] Turing: *Guide to Autoregressive Models*. 2023. URL: https://www.turing.com/kb/guide-to-autoregressive-models (visited on 06/06/2023).

[41] Weiner, A.: *Machine learning in computational fluid dynamics*. https://github.com/AndreWeiner/ml-cfd-lecture. 2021.

[42] Weiner, A. and Semaan, R.: “flowTorch - a Python library for analysis and reduced-order modeling of fluid flows”. In: *Journal of Open Source Software* 6 (Dec. 2021), p. 3860. DOI: 10.21105/joss.03860.

[43] Yorita, D., Klein, C., Henne, U., et al.: “Successful Application of Cryogenic Pressure Sensitive Paint Technique at ETW”. In: *2018 AIAA Aerospace Sciences Meeting*. DOI: 10.2514/6.2018-1136.

[44] Zahn, R. and Breitsamter, C.: “Prediction of transonic wing buffet pressure based on deep learning”. In: *CEAS Aeronautical Journal* 14 (2023), pp. 155–169.

[45] Zahn, R., Weiner, A., and Breitsamter, C.: “Prediction of wing buffet pressure loads using a convolutional and recurrent neural network framework”. In: *CEAS Aeronautical Journal* (2023).

List of Figures