B.Comp. Dissertation

Physics-Informed Machine Learning: Variational Autoencoders for Disentangled Latent Representations in Dynamical Systems

by

Lian Guo Yang
e0735382@u.nus.edu

Final Report Submitted for the Final Year Project (FYP)

Bachelor of Computing

Department of Computer Science

School of Computing

National University of Singapore

2025

B.Comp. Dissertation

Physics-Informed Machine Learning: Variational Autoencoders for Disentangled Latent Representations in Dynamical Systems

by

Lian Guo Yang
e0735382@u.nus.edu

Final Report Submitted for the Final Year Project (FYP)

Bachelor of Computing

Department of Computer Science

School of Computing

National University of Singapore

2025

Project No:
H0811680

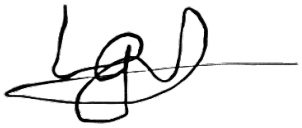
Advisor:
A/P Stéphane Bressan
Dr. Han Liang Wee, Eric

Deliverables:
Report: 1 Volume

Declaration

I hereby declare that this final report is my original work and it has been written by myself in its entirety. I have duly acknowledged all the sources of information which have been used in the report.

This final report has also not been submitted for any degree in any university previously.



Lian Guo Yang

e0735382@u.nus.edu

4th July 2025

Abstract

Understanding dynamical systems is essential in fields such as physics, engineering, and biology, where complex processes evolve over time. This project explores a physics-informed machine learning approach to discover a compact set of state variables that capture the system's essential degrees of freedom. Using Variational Autoencoders (VAEs), we develop a two-step framework that combines hyperparameter tuning with latent variable selection techniques to identify compact, disentangled representations in a low-dimensional latent space that may correspond to key state variables. Inspired by the nested autoencoder proposed by Chen et al. (2021), we introduce additional techniques to improve disentanglement and identify the most informative latent dimensions for accurate reconstruction. Our experiments demonstrate the model's potential for discovering candidate state variables and achieving improved disentanglement, resulting in promising reconstruction accuracy across multiple dynamical systems datasets.

Subject Descriptors:

I.2.6 Learning

G.3 Probability and Statistics

I.5.1 Models – Neural Nets

Keywords:

Physics-informed machine learning, Dynamical systems, Variational Autoencoder, Latent representations, Disentanglement

Implementation Software and Hardware:

NVIDIA A100-40 GPU

Ubuntu 22.04.5 LTS (Jammy Jellyfish)

SoC Compute Cluster

Acknowledgments

I would like to express my sincere gratitude to my supervisor, A/P Stéphane Bressan, for his early supervision, invaluable guidance, support, and encouragement throughout this project. His insightful advice and unwavering support have been instrumental in shaping both the foundation and direction of my work. I am truly grateful for the opportunity to have benefited from his mentorship.

I am also deeply thankful to Dr. Han Liang Wee, Eric, who later took over the supervision of this project. His support and guidance during the later stages of the project have been instrumental in bringing the work to completion. His expertise and thoughtful insights greatly improved the quality and focus of the project.

I am deeply appreciative of my advisors, Dawen Wu, Khoo Zi-Yu, and Pratik Karmakar, for their invaluable contributions to this project. Their constructive feedback and willingness to share their expertise have been essential in refining my approach. The provision of existing code and datasets greatly facilitated my progress, allowing me to focus on advancing the research. I am incredibly fortunate to have received their mentorship, which has significantly enriched both my understanding and research experience.

Finally, I would like to thank my peers and colleagues for their helpful discussions and feedback, which have contributed meaningfully to the development of this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Dynamical systems, which evolve over time, are fundamental to understanding complex behaviours across various domains, from physics and biology to economics and engineering (Brin & Stuck, 2002). The study of dynamical systems originated in the late 19th century from curiosity about the stability and development of the solar system, leading to the development of a field with diverse applications (Brin & Stuck, 2002). These systems exhibit unique properties, such as stability, periodicity, and chaotic behaviour, that can be analysed by studying how the system's state evolves over time. As these dynamical systems grow in complexity, the need for efficient modelling techniques becomes increasingly critical.

For complex dynamical systems, such as Earth's climate, deriving low-dimensional models is essential for reducing computational costs and improving interpretability (Chavelli et al., 2024; Qian et al., 2020). However, accurately modelling these systems remains challenging due to high computational requirements and inherent uncertainties in capturing multiscale interactions (Karniadakis et al., 2021). Recent advancements in machine learning have provided powerful tools to tackle these challenges, particularly by extracting compact state variables from high-dimensional data that can be used to predict future system behaviour and, ideally, represent meaningful physical quantities. However, extracted variables often exhibit entanglement and redundancy, limiting their utility in analysis and interpretation. Disentanglement enhances interpretability and reduces redundancy, enabling clearer analysis. Effective modelling thus requires selecting the most relevant, independent state variables and minimising unnecessary redundancy and entanglement to enable a more structured understanding of system dynamics.

1.2 Objectives

Motivated by the work of Chen et al. (2021), where a nested autoencoder was used to identify state variables in dynamical systems, and Chavelli et al. (2024), where Variational Autoencoders (VAEs) were employed to regularise the latent space toward a prior distribution to encourage disentanglement, this report proposes a framework that utilises, VAEs after hyperparameter tuning with latent variable selection, to determine a compact set of latent variables that may correspond to the system's underlying degrees of freedom.

In this report, techniques for reducing the latent dimensionality of VAEs are investigated, and different hyperparameter configurations are systematically evaluated to determine their impact on model performance and representation quality. The experimental design aims to improve interpretability, efficiency, and scalability. This work aims to contribute to the development of automated and robust techniques for uncovering essential structure in dynamical systems. We hypothesise that combining hyperparameter tuning with latent variable selection improves the approximation of the degrees of freedom in dynamical systems. The selected latent variables are expected to preserve reconstruction quality while reducing redundancy.

1.3 Overview

In Chapter 2, we introduce foundational background, focusing on the work of Chen et al. (2021) on nested autoencoders for identifying state variables and degrees of freedom, as well as an introduction to the VAEs used in our approach. Chapter 3 reviews existing work and concepts proposed by other researchers in this field, establishing the context for our study. Chapter 4 outlines our proposed framework to advance these approaches. In Chapter 5, we detail our experimental setup and present the results, demonstrating the effectiveness of our methods. We conclude with future research directions and provide closing remarks in Chapter 6.

Chapter 2

Background

2.1 Dynamical Systems

Dynamical systems study how systems evolve over time, with the aim of understanding their long-term behaviour. The modern theory of dynamical systems was developed in the late 19th century, prompted by questions about the stability and evolution of the solar system (Brin & Stuck, 2002). Formally, a dynamical system consists of a space X, representing all possible states of the system, and a map $f : X \to X$ that governs how the system transitions between states over time.

2.2 Autoencoders

Autoencoders are unsupervised neural networks designed to learn efficient and compressed representations of data (Hinton & Salakhutdinov, 2006). They consist of two main components, an encoder that compresses the input into a lower-dimensional representation known as the latent space, and a decoder that reconstructs the input from this compressed representation to produce an output.

As illustrated in Figure 2.1, the encoder maps the high-dimensional input X to a compact latent representation. The decoder then reconstructs the original input from this latent code, yielding the output X'.

Autoencoders are particularly valuable for tasks such as dimensionality reduction, anomaly detection, and feature extraction for downstream applications. In seminal work, Hinton and Salakhutdinov (2006) demonstrated that deep autoencoders can be used to learn compact, non-linear representations that outperform traditional linear techniques.

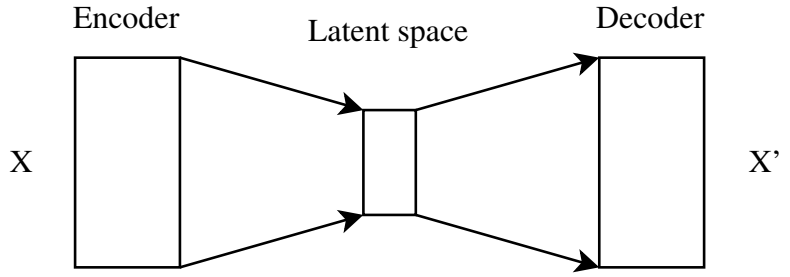


Figure 2.1: Autoencoder Architecture

2.3 Variational Autoencoders (VAEs)

This project employs Variational Autoencoders (VAEs), introduced by Kingma and Welling (2019). VAEs are generative models that enhance traditional autoencoders by incorporating probabilistic latent variables, thus enabling both a structured latent space and uncertainty quantification. This probabilistic framework provides insights into the stability and predictability of the system's dynamics, making VAEs a more suitable choice for capturing complex system behaviours than traditional autoencoders, which lacked this structured and interpretable latent space.

VAEs have the potential to learn disentangled representations in the latent space, where each latent representation is independent. This is particularly useful in the context of dynamical systems because identifying independent state variables simplifies analysis and improves interpretability (Kingma & Welling, 2022). VAEs leverage the regularising effect of the Kullback-Leibler Divergence (KLD) term in its loss function, which constrains the latent distribution to follow the prior distribution and promotes the learning of disentangled latent representations (Kingma & Welling, 2022).

2.3.1 Evidence Lower Bound (ELBO) and Loss Function

The VAE framework optimises an Evidence Lower Bound (ELBO) loss function, composed of two terms, the reconstruction loss and the Kullback-Leibler Divergence (KLD) regularisation term. This objective balances reconstruction accuracy with regularisation by reconstructing the input data while ensuring that the latent space follows a prior distribution, typically a standard Gaussian. The KLD term structures the latent space, promoting smooth sampling and effective interpolation (Kingma & Welling, 2019).

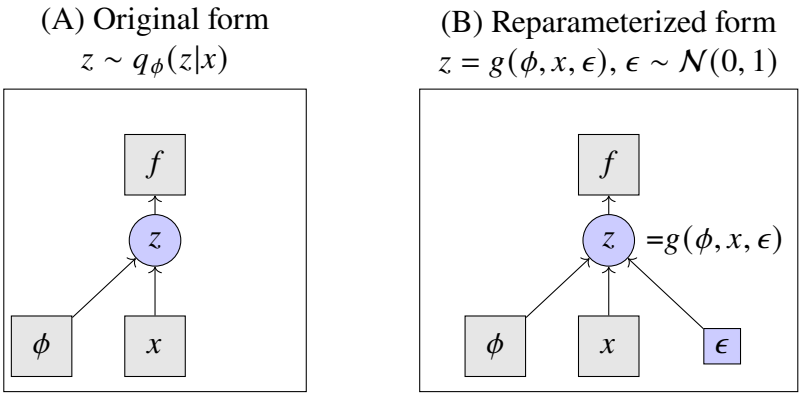


Figure 2.2: Illustration of the Reparameterisation Trick. (A) shows the original form, where direct sampling makes it non-differentiable. (B) shows the reparameterised form, where randomness separated through ϵ, allowing gradients to flow for optimisation.

2.3.2 Reparameterisation Trick in VAE

In the standard VAE framework, latent variables are sampled from a Gaussian distribution $N(\mu, \sigma)$. However, direct sampling introduces randomness, which prevents gradients from propagating through the network during backpropagation, making it challenging to train the model with gradient-based methods such as stochastic gradient descent (SGD) (Kingma & Welling, 2022). To address this, the reparameterisation trick is applied, as illustrated in Figure 2.2. By re-expressing the latent variable as $z = \mu + \sigma \cdot \epsilon$, where $\epsilon \sim N(0, 1)$, the sampling process becomes differentiable, enabling gradients to flow through μ and σ, allowing effective optimisation of model parameters.

2.3.3 Extensions and Latent Space Optimisation in VAEs

Burgess et al. (2018) introduced the β-VAE, which uses a weighting hyperparameter β to control the strength of KLD regularisation, thereby enhancing disentanglement and improving the interpretability of the latent variables, while preserving the model's ability to reconstruct the input. A representation is disentangled when each latent variable corresponds to a distinct and interpretable generative factor, such as rotation, scale, or identity (Burgess et al., 2018).

This refinement, shown in Equation 2.1, improves the model's convergence, enabling it to isolate key factors of variation within the data.

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta \cdot D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})) \tag{2.1}$$

More recently, efforts to optimise the number of latent dimensions have gained traction. Bonheme and Grzes (2022) introduced the Finding the Optimal Number of Dimensions from Unsupervised Estimation (FONDUE) algorithm, a binary search approach for finding the optimal dimension threshold, though this method requires retraining the model multiple times. These advancements reflect ongoing efforts to maximise the interpretability and efficiency of VAEs by capturing the optimal degrees of freedom through disentanglement and latent space optimisation.

Chapter 3

Related Works

3.1 Discovering State Variables

Our approach is inspired by the work of Chen et al. (2021), which introduced a nested autoencoder framework with an outer autoencoder to process images and an inner one to capture essential physical dimensions. Although this model effectively reconstructs inputs, identifying the minimum set of independent variables to fully describe dynamical systems remains a challenge. Earlier work by Ljung (1999) applied linear regression models to discover simple patterns in such systems. Schmid and Sesterhenn (2008) later developed Dynamic Mode Decomposition (DMD), a method similar to Principal Component Analysis (PCA), that isolates complex data patterns while retaining temporal information and is particularly useful in fluid dynamics. Motivated by these foundations, the Sparse Identification of Nonlinear Dynamics with Control (SINDYc) framework introduced sparse regression to capture non-linear relationships in diverse systems (Brunton et al., 2016). Recently, Schmidt and Lipson (2009) advanced this area further by using symbolic regression to derive polynomial expressions, helping to pinpoint essential variables in complex systems. These past efforts highlight the complexity of identifying optimal state variables, motivating our approach to address this challenge with efficiency and precision.

3.2 Intrinsic Dimension Estimators

Intrinsic dimension refers to the minimum number of independent variables needed to describe the state of a dynamical system. Intrinsic dimension estimators are commonly classified into projective methods, topological nearest-neighbour methods, and graph-based methods (Campadelli et al., 2015).

3.2.1 Projective Method

Projective methods project high-dimensional data onto a lower-dimensional subspace while retaining the variance of the original data. Principal Component Analysis by Jolliffe (2002) offers a relatively simple and computationally efficient approach to dimensionality estimation, but may underperform when capturing non-linear structures.

3.2.2 Graph-Based Method

Graph-based methods build a graph over the data using distance functions. Using Euclidean distance forms a k-nearest neighbours graph, while approximating geodesic distances results in a minimum spanning tree. Intrinsic dimension is then estimated from statistics computed on the graph structure (Chavelli et al., 2023).

3.2.3 Topological Nearest-Neighbour Method

Topological methods estimate intrinsic dimension based on geometric relationships, such as distances and angles among local neighbours, to capture the underlying data topology. Notable examples include the k-Nearest Neighbours estimator by Costa and Hero (2007), Correlation Integral by Grassberger and Procaccia (1983), Manifold-Adaptive Dimension Estimation Algorithm by Farahmand et al. (2007), maximum likelihood-based estimators by Levina and Bickel (2004), Poisson mixture model by Haro et al. (2008), Minimum Neighbour Distance-Maximum Likelihood estimators by Rozza et al. (2012), Two-Nearest Neighbour estimator by Facco et al. (2017), method of moments estimator by Amsaleg et al. (2018), and U-statistic-based estimator by Hein and Audibert (2005).

The model proposed by Chen et al. (2021) used an intrinsic dimension estimator to determine the minimal number of latent variables to accurately reconstruct dynamical systems. After experimenting with several intrinsic dimension estimation algorithms, including MiND_ML, MiND_KL, Hein, and CD, Chen et al. (2021) selected the Levina and Bickel (2004) algorithm, a topological nearest-neighbour-based method, as it was found to be the most accurate in estimating the intrinsic dimension (Chen et al., 2021).

However, recent work by Chavelli et al. (2023) has shown that the Levina and Bickel (2004) algorithm used in the Chen et al. (2021) model is not the most effective method. Instead, maximum likelihood-based estimator by Haro et al. (2008) and Correlation Integral by Grassberger and Procaccia (1983) outperform it in terms of accuracy. Although

intrinsic dimension estimators can effectively analyse dynamical systems, they are not directly connected to the continuous nature of dynamical systems, resulting in non-integer estimates of the intrinsic dimension, which may be unsuitable for dynamical systems where the degrees of freedom are expected to be an integer (Chavelli et al., 2023).

3.3 Autoencoders in Dynamical Systems Modelling

Autoencoders are neural networks designed to learn compact representations of a dataset. They consist of an encoder that maps data to a lower-dimensional latent space and a decoder that reconstructs the original data from this latent representation. By training to minimise reconstruction error, autoencoders can extract essential features of the dataset.

Dynamical systems can be effectively modelled through mathematical frameworks, and neural networks offer a powerful means of formalising these models through appropriate architectures. Chen et al. (2021) proposed a nested autoencoder architecture, where the outer autoencoder captures essential features of the input data, and the inner autoencoder further compresses these features into a set of meaningful state variables. In this structure, latent variables within the autoencoder encapsulate important characteristics of the data, while the state variables represent those relevant latent features that capture the underlying dynamics of the system.

3.3.1 Nested Autoencoder Architecture

Figure A.5 shows the architecture of the model proposed by Chen et al. (2021). They used a two-step training approach to model dynamical systems effectively. First, input frames (A), $X_t \in \mathbb{R}^M$, are handled by an encoder to produce a latent representation (B).

The intrinsic dimension refers to the minimum number of independent variables needed to describe a dynamical system (Chen et al., 2021). To estimate the number of state variables, we used an intrinsic dimension estimator based on geometric manifold learning algorithms, specifically the Levina–Bickel algorithm, which is a maximum likelihood-based method (Levina & Bickel, 2004). They set the dimension of the inner autoencoder with the estimated number of state variables. The latent representation (B) is then transformed into neural state variables (C), representing the system intrinsic dynamics. If the inner autoencoder successfully learns the latent representation (B), the neural state variables (C) will closely match the latent representation (B). These neural state variables

are passed through a decoder to reconstruct the latent state (D). This reconstruction yields predicted output frames (E) that capture the system's dynamic behaviour.

While this method successfully extracts state variables, it lacks a probabilistic formulation of the latent space and does not produce a disentangled representation. In this setup, each input is mapped deterministically to a vector in the latent space, which does not capture uncertainty and variability in the data. These limitations motivate the transition from an unstructured generative model to a structured probabilistic generative model.

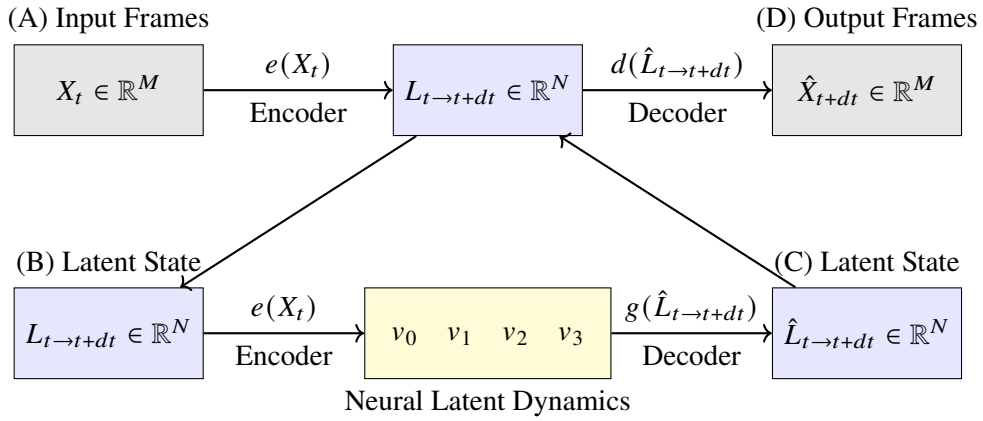


Figure 3.1: Overview of the neural network model proposed by Chen et al. (2021)

3.4 Hypothesis

We hypothesise that applying a two-step framework that can be applied on VAEs, first by fine-tuning the hyperparameters of VAEs, followed by latent variable selection using techniques that use Principal Component Analysis and Shapley values, to produce disentangled latent representations. These representations are expected to approximate the underlying degrees of freedom in dynamical systems. We anticipate that the selected latent variables from the latent space of the VAE will have low reconstruction error while reducing redundancy compared to unregularised latent spaces.

Collectively, these works highlight the importance of learning interpretable state representations for dynamical systems. However, many existing works rely on deterministic models or fixed latent dimensionality. Our work aims to address these gaps by applying VAEs with KLD weight tuning and latent variable selection, producing structured, disentangled representations that better reflect the true degrees of freedom in the system.

Chapter 4

Methodology

We adopt a two-step optimisation strategy to improve the training process of the Variational Autoencoders (VAEs). The first step involves tuning the Kullback–Leibler Divergence (KLD) weight, β, to balance reconstruction accuracy and latent space regularisation. The second step performs latent variable selection to identify the most informative dimensions, thereby enhancing model efficiency while preserving reconstruction quality.

4.1 Model Architecture

The VAE model consists of an encoder and a decoder, both built with convolutional, fully connected, and transposed convolutional layers. Together, these components enable feature extraction, dimensionality reduction, and upscaling for image reconstruction.

4.1.1 Encoder Network

The encoder network begins with a series of convolutional layers designed to progressively reduce the spatial dimensions of the input image while increasing the depth of feature channels. These layers are followed by fully connected layers that further condense the dimensionality, resulting in a compact latent representation.

Convolutional Layers: The encoder comprises of six convolutional layers, starting with an input depth of 3 channels and progressively increasing to 512. Each layer uses a 3×3 kernel, a stride of 2, and padding of 1. After each convolution, `BatchNorm2d` is applied to stabilise training, followed by a `LeakyReLU` activation with a negative slope of 0.01 to introduce non-linearity while preserving small gradients for negative values.

After the convolutional layers, the output is flattened and fed into a sequence of fully connected layers to further reduce the dimensionality, transforming the feature map into a

compact latent representation.

Fully Connected Layers: The encoder's fully connected layers include three linear transformations that reduce the dimensionality from 1536 to the desired latent space dimensions. A sigmoid activation function is applied at the final stage to compress the latent representation.

The encoder's output layer includes two additional fully connected layers, denoted as μ and σ, which represent the mean and variance of the latent space distribution.

4.1.2 Decoder Network

The decoder network reconstructs the image from the compact latent representation. It starts with fully connected layers to reshape the latent vector, followed by transposed convolutional layers for progressively upscaling the spatial dimensions to match the original input size.

Fully Connected Layers: The decoder's fully connected layers expand the latent space dimensions back up to 1536, to prepare for the spatial structure needed by the transposed convolutional layers.

Transposed Convolutional Layers: Five transposed convolutional layers are used, each with a kernel size of 3×3, a stride of 2, padding of 1, and an output padding of 1, allowing gradual upscaling of the feature map to its original size. Each transposed convolutional layer is followed by `BatchNorm2d` and `LeakyReLU`, mirroring the encoder structure to maintain consistency in activation and normalisation.

The final layer is a transposed convolutional layer designed to produce the output image with the desired number of channels (3 for RGB images), using a 2×2 kernel, stride of 2, and padding of 1, followed by a `Tanh` activation function to normalise the pixel values.

4.1.3 Rationale and Table of Model Architecture

This model uses convolutional layers with `BatchNorm2d` and `LeakyReLU` to extract features and encode input images into a compact latent space. It then uses transposed convolutional layers, also with batch normalisation and activation, to decode the latent representation and generate high-quality reconstructions. Table 4.1 and 4.2 shows the layers and architecture of the encoder and decoder of the model respectively.

Layer	Type	Channels (Input, Output)
Convolutional Layers		
1	Conv2D	(3, 32)
2	Conv2D	(32, 64)
3	Conv2D	(64, 128)
4	Conv2D	(128, 256)
5	Conv2D	(256, 512)
6	Conv2D	(512, 512)
Fully Connected Layers		
1	Linear	(1536, 512)
2	Linear	(512, 256)
3	Linear	(256, 32)
4	Sigmoid	-
Output Layers		
1	Linear	(32, x)
2	Linear	(32, x)

Table 4.1: Architectural Overview of the Encoder Networks. The encoder utilises convolutional layers followed by fully connected layers to produce a compact latent representation. Here, x denotes the number of latent dimensions chosen for the VAE.

Layer	Type	Channels (Input, Output)
Fully Connected Layers		
1	Linear	(x, 32)
2	Linear	(32, 256)
3	Linear	(256, 512)
4	Linear	(512, 1536)
5	Sigmoid	-
Transposed Convolutional Layers		
1	ConvTrans2D	(512, 512)
2	ConvTrans2D	(512, 256)
3	ConvTrans2D	(256, 128)
4	ConvTrans2D	(128, 64)
5	ConvTrans2D	(64, 32)
Output Layer		
1	ConvTrans2D	(32, 3)

Table 4.2: Architectural Overview of the Decoder Networks. The decoder reconstructs the image using fully connected and transposed convolutional layers. Here, x denotes the number of latent dimensions chosen for the VAE.

4.1.4 Input and Output Representation

Each input to the model consists of three consecutive frames at time steps $t, t+1, t+2$, horizontally stacked into a single image. The model is trained to reconstruct the next sequence of frames at $t+1, t+2, t+3$, also stacked in the same format. This setup encourages the VAE to learn temporal dynamics by observing the short-term evolution of the system and predicting its near-future behaviour.

In the work by Chen et al. (2021), horizontally stacked four consecutive images were used to form the input. In our case, horizontally stacked three consecutive images were used to form the input. This was hypothesised to sufficiently describe the full state of the dynamical system, enabling temporal dependencies to be learned by the model. The intuition is that the first frame represents the positions of the system, the second frame captures the momentum and derivative of the position (i.e., velocity), and the third frame shows the derivative of the velocity, which corresponds to the acceleration of the dynamical system. This configuration enables the model to capture temporal dynamics and understand the underlying system evolution across time.

Figure 4.1 illustrates the structure of the VAE used in this study. The input to the encoder consists of three horizontally stacked frames at $t, t+1, t+2$ which are compressed into a latent space. The decoder then reconstructs the next three frames at $t+1, t+2, t+3$, preserving the format and temporal structure of the input. This design enables the model to capture and predict the change of dynamical system over time.

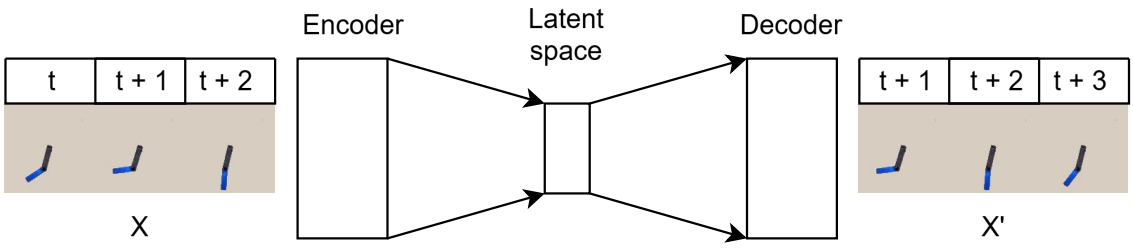


Figure 4.1: Variational Autoencoder Architecture

4.2 Loss Function

The total loss function used in this work follows the β-VAE framework and consists of two key components, the reconstruction loss and the Kullback-Leibler Divergence (KLD)

loss (Burgess et al., 2018). The reconstruction loss encourages the decoder to accurately reproduce the input data, while the KLD weight regularises the latent space by aligning the approximate posterior with a standard normal prior.

Equation 4.1 shows the full loss formulation, where β is the weighting coefficient that balances reconstruction fidelity against regularisation. The reconstruction loss is typically measured as the mean squared error between the ground truth and the reconstructed output.

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Loss}} - \underbrace{\beta \cdot D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z}))}_{\text{KLD Loss}} \tag{4.1}$$

Equation 4.2 shows the general form of the KLD loss, which quantifies the divergence between the learned latent distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{z})$. Under the assumption that $Q(x) = \mathcal{N}(\mu, \sigma^2)$ and $P(x) = \mathcal{N}(0, 1)$, the KLD simplifies to the expression shown in Equation 4.3.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{4.2}$$

$$\text{KLD Loss} = \frac{1}{2} \sum_{i=1}^{D} \left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2 \right), \quad \text{where } P \sim \mathcal{N}(0, 1) \tag{4.3}$$

By varying the β value, we can control the degree of regularisation imposed on the latent space. Higher values of β encourage the model to learn a more compact and disentangled latent space, while lower values of β prioritise reconstruction quality. A balanced value of β is critical for achieving disentanglement without compromising reconstruction quality.

4.2.1 Choice of Reconstruction Loss

In VAEs, the choice of reconstruction loss depends on the nature of the data. Given that our dataset comprises continuous valued inputs such as RGB images, we modelled the decoder as a Gaussian distribution, consistent with the original VAE formulation (Kingma & Welling, 2022). As a result, we used Mean Squared Error (MSE) as it is more appropriate than Binary Cross Entropy (BCE) used in the original paper. BCE is used when the decoder models a Bernoulli distribution, such as in datasets like the Modified National Institute of Standards and Technology (MNIST) database (Kingma & Welling, 2022).

4.3 KLD Weight Tuning

To balance the trade-off between reconstruction accuracy and KLD regularisation, we adjusted the value of the β parameter, which controls the strength to the KLD weight. The objective was to regulate the latent space without compromising reconstruction quality. We began by testing a wide range of β values, to understand how it affects the model's performance. Based on the analysis, we progressively narrowed down the range using grid search to fine-tune β and optimise model performance.

This tuning process aimed to identify a configuration that maintains high reconstruction quality while enforcing regularisation through the KLD weight. By balancing these objectives, we aimed to enhance the model's ability to generate smooth, meaningful latent representations without overfitting to the training data.

4.4 Latent Variable Selection

The aim of latent variable selection was to determine the minimum number of latent dimensions required to effectively represent the dataset, thus simplifying the model while preserving reconstruction accuracy. This process followed an iterative approach, beginning with the assessment of individual latent dimensions and progressively advancing to subset selection and model reduction.

4.4.1 Exhaustive Latent Variable Selection

This initial stage aimed to identify the minimum set of latent variables required for the dataset. The process was carried out in three stages, evaluating individual latent variables, exploring subsets of variables, and reducing the model using the selected dimensions.

While our initial approach focused on assessing the individual contribution of each latent variable, it quickly became clear that this was not sufficient. Many latent variables are dependent, and their influence on reconstruction cannot be properly understood in isolation. To address this, we extended our approach by exhaustively searching through all possible subsets of latent variables. This allowed us to capture the pairwise effects of variable combinations and better identify redundancies or dependencies across dimensions. For each possible number of latent variables, we selected the subset that achieved the lowest reconstruction error. By systematically exploring all combinations, we found compact

subsets that offered the best trade-off between reconstruction and model simplicity.

4.4.1.1 Mean Zeroing (`mean-zero`)

In the initial phase, we assessed each latent dimension individually by setting its mean to zero and observing the effect on the reconstruction loss using the test data. If zeroing a dimension's mean led to a 10% or greater increase in reconstruction loss, the dimension was deemed essential and retained in the model. While straightforward, this approach evaluates the contribution of each variable independently and does not account for interactions between variables.

4.4.1.2 Model Reduction (`model-reduct`)

To further streamline the model for retraining and simplicity, we constructed a new, reduced model with fewer latent dimensions, incorporating only the selected variables identified in the previous steps. In this method, we created a smaller architecture and transfer the trained weights to the new model from the original model, dropping the unused latent variables. By transferring the trained weights to this reduced model, we minimised the effect of unselected latent variables particularly the noise introduced by the reparameterisation trick. This ensured that only the necessary latent variables contributed to the reconstruction. This final step simplified the model, enhancing both efficiency and interpretability. We recorded only the lowest test reconstruction loss based on the number of latent variables retained.

4.4.2 Selection Using Various Techniques

Exhaustive selection of latent variables is not efficient in our case, because we are attempting to reduce the model from a relatively large number of latent dimensions. Evaluating all possible subsets becomes computationally expensive and infeasible. To address this, we explored additional selection techniques that provide more scalable alternatives while still being able to identify important latent variables.

4.4.2.1 Shapley Values

To better understand the individual contribution and interactions among the latent variables, Shapley values, a cooperative game theoretic method that assigns importance scores to each latent variable based on its marginal impact across all possible subsets of

variables is used (Ghorbani & Zou, 2019).

The Shapley value ϕ_i for each latent variable i is computed as:

$$\phi_i = C \sum_{S \subseteq D \setminus \{i\}} \frac{V(S \cup \{i\}) - V(S)}{\binom{|D|-1}{|S|}}, \tag{4.4}$$

ϕ_i denotes the Shapley value of the latent variable i, representing its average contribution to the model performance. C is a normalising constant and it is typically $1/n!$. D is the set of all latent variables, S is any subset of D that does not contain i, and $V(S)$ is defined as the model's reconstruction loss when using a subset S of the latent variables.

Shapley value can be approximated by estimating the difference in reconstruction loss when a latent variable is included or excluded from randomly sampled subsets. These Shapley values help capture both the individual contribution of a latent variable but also capture their interaction effects with other latent variables. Latent variables with low Shapley values can be considered uninformative, as their inclusion in the model does not lead to a significant reduction in reconstruction loss. Conversely, those latent variables with high Shapley values contribute significantly to the model and are crucial to produce high quality reconstructions.

Motivated by the principle of Occam's Razor, which favours simpler explanations, we adopt a heuristic grounded in the Minimum Description Length (MDL) formalism Wu (2020). Specifically, when searching for the optimal subset of latent variables, we adopt various simple heuristics to guide selection. One such heuristic is defined as the product of the test reconstruction error and $\log_2(k + 1)$, where k denotes the number of selected latent variables. This formulation provides an intuitive and computationally efficient approximation of the total description length, balancing model complexity against predictive performance.

Forward Selection (`forward-select`): Forward selection is a greedy subset selection algorithm that incrementally builds a subset of variables by progressively adding those that most improve a given objective function, the heuristic function in this case, as described by Guyon and Elisseeff (2003). Each iteration grows the subset by evaluating changes in the objective function after including a candidate variable. In the present implementation, this procedure starts from a binary mask of all zeros, representing an initially empty set of latent variables, and iteratively adds one variable at a time based on which inclusion

yields the lowest heuristic. This aligns with the general structure of forward selection described in the literature (Guyon & Elisseeff, 2003), where the model is constructed from the bottom up, beginning with no variables and stopping once no further improvement is observed.

Backward Elimination (`backward-elim`): Backward elimination is a greedy subset selection algorithm that incrementally decreases the size of the subset by progressively eliminating the least promising variables that least affect the model's performance, given the objective function, the heuristic function in this case, as described by Guyon and Elisseeff (2003). Each iteration shrinks the subset by evaluating changes in the objective function after excluding a candidate variable. In the present implementation, the procedure is initialised from a binary mask of all ones, representing the full set of latent variables, and one variable is removed at a time based on which inclusion yields the lowest heuristic. This follows with the general structure of backward elimination described in the literature (Guyon & Elisseeff, 2003), where the model is constructed from the top down, beginning with all variables and stopping once no further improvement is observed.

Greedy Selection (`greedy-select`): Greedy selection is implemented and begins with a randomly initialised subset of latent variables and iteratively explores neighbouring configurations by flipping one latent variable at a time. At each step, the candidate subset with the lowest heuristic value is compared against the current best (parent) subset. If it yields a better score, it replaces the parent and the process continues; otherwise, the search terminates. Unlike strictly additive or subtractive methods, this approach allows both inclusion and exclusion of latent variables at each step, enabling traversal of the search space in both directions. The method is inspired by the principles of greedy search strategies described in Guyon and Elisseeff (2003).

Genetic Algorithm (`genetic-algo`): A generational genetic algorithm is adopted, where a population of candidate latent masks is initialized randomly. In each generation, two parent solutions are selected from the population and combined using crossover to produce a new offspring. Mutation is then applied to introduce small random changes. This process is repeated until a complete new generation is formed. The current population is then fully replaced with the newly generated offspring. This procedure continues until a

stopping criterion is met, such as a fixed number of generations. This approach enables a diverse and global search over the combinatorial space of latent variable subsets (Oh et al., 2004).

4.4.2.2 Principal Component Analysis (PCA) (`pca`)

To reduce the dimensionality of the latent space, we applied Principal Component Analysis (PCA) to the means of the latent variables. In our case, we performed PCA without centering the data. This is because the latent variables are already regularised to follow a standard normal distribution, $N(0, 1)$, due to the Kullback-Leibler Divergence (KLD) term in the VAE loss function. Centering the data could shift the structure of the trained latent variables and potentially distort the structure of the latent space. Thus, to preserve the prior distribution of the latent variables, we avoided centering the data during PCA.

Following the uncentered PCA formulation (Shlens, 2014), we directly applied Singular Value Decomposition (SVD) to the mean of the latent variable matrix and extract the principal components (PCs), to reconstruct our outputs.

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top} \tag{4.5}$$

Here, $\mathbf{X}$ is the matrix of the mean of the latent variables, $\mathbf{\Sigma}$ contains the singular values, and $\mathbf{V}$ corresponds to the principal components.

After computing PCA on the latent space, we retained the top k principal components that cumulatively explain at least 95% of the total variance. This effectively reduced the latent dimensionality from the original x dimensions that is used during model training to a more compact k dimensions without significant loss of information.

To evaluate the impact of explained variance on reconstruction quality, we retained increasing numbers of principal components corresponding to cumulative variance thresholds ranging from 25% to 100%. As shown in Table 4.3, reconstruction accuracy improves with more components, but diminishing returns are observed beyond 95% variance. Notably, using only 5 latent variables to retain 95% variance achieves suboptimal test loss compared to the full 256-dimensional latent space.

Table 4.3: Test performance by number of latent variables required to reach target variance thresholds

Explained Variance	**Num of LV**	**Test Loss**	**Reconstruction Error**	**Test KLD Loss**	**KLD Weight (β)**
25%	1.0	0.007904	0.007868	90.048	10^{-7}
50%	2.0	0.006176	0.006139	92.159	10^{-7}
75%	3.8	0.003568	0.003530	94.706	10^{-7}
95%	5.0	0.000348	0.000309	95.759	10^{-7}
100%	256.0	0.000276	0.000238	96.054	10^{-7}

(a) 25% (b) 50% (c) 75% (d) 95% (e) 100%

Figure 4.2: Decoded outputs as more latent variables are retained (increasing explained variance).

In summary, our methodology employed a structured VAE architecture trained on temporally stacked inputs and guided by a carefully tuned KLD weight. Post-training, we applied latent variable selection techniques to identify the most informative dimensions. This two-step framework sets the foundation for the experiments presented in the next chapter, where we evaluated the effectiveness of our approach on dynamical systems datasets.

Chapter 5

Experimental Setup and Results

5.1 Experimental Setup

This experiment evaluates the model's ability to identify the degrees of freedom in three distinct dynamical systems, the double pendulum and air dancer dataset from Chen et al. (2021) and a double pendulum dataset from Dr. Zi-Yu Khoo (National University of Singapore). The framework presented here offers potential for extension to other datasets and modelling tasks in future work, providing a basis for broader validation and refinement. To account for the stochastic nature of some parts of the experiment, each configuration involving randomness is executed three times, and the average performance is reported to reduce the impact of random variability.

1. **Dataset:** The experiments were conducted on three datasets: a double pendulum dataset provided by Dr. Khoo, and the double pendulum dataset and the air dancer dataset from Chen et al. (2021). These datasets consist of time-series images sequences that capture the physical motion of the double pendulum and the air dancer, respectively. Each sequence of images represents the dynamic behaviour of these systems.

(a) Double Pendulum (Dr. Khoo)

(b) Double Pendulum (Chen et al.)

(c) Air Dancer (Chen et al.)

2. **Model Configuration:** We used a standard VAE architecture, featuring a convolutional and fully connected layers in both the encoder and decoder, as described in the methodology section, Table 4.1 and Table 4.2. The latent space was initially set to 10 dimensions to examine potential entanglements in the representations, and then expanded to 256 dimensions to study the model's behaviour with a larger latent space. To address this, KLD weight tuning and latent variable selection were applied to optimise the number of active latent variables and the regularisation strength.

3. **Training Setup:** The model was trained using PyTorch Lightning version 2.4.0. The KLD weight, β, was dynamically adjusted to observe its effect on test loss. Each experiment was run over 400 epochs, with final results selected based on the model's state at the last epoch.

   Throughout training, we logged critical metrics including test loss, reconstruction loss, KLD loss, and the number of latent variables to analyse the trade-offs and assess the model's efficiency in learning meaningful representations. We trained a total of four different model variants: `double-pendulum-z-10`, `double-pendulum-z-256`, `double-pendulum-c-256`, and `air-dancer-c-256`. The first part of each name indicates the dataset used, where `z` refers to the dataset provided by Dr. Khoo, and `c` refers to the dataset from Chen et al. (2021). The trailing number, 10 or 256, denotes the number of latent variables used in the respective model.

   The dataset is split into 70% for training, 15% for validation, and 15% for testing. Following the description in Chen et al. (2021), this includes 56,729 triplets of

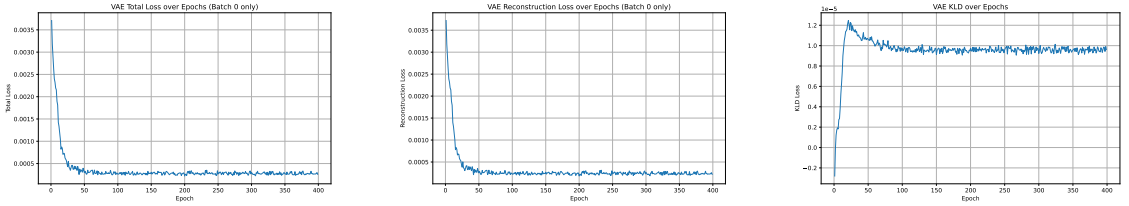
consecutive frames from the Air Dancer system and 16,502 frames from the Double Pendulum system.

5.1.1 Training Curves

To evaluate the convergence behaviour of the model during training, we tracked the total loss, reconstruction loss, and KLD loss over each epoch. As shown in Figure 5.2, the total loss rapidly decreases during the initial 100 epochs before gradually stabilising. This sharp decline is likely due to the model learning to reconstruct the static background early, which dominates the pixel-wise loss. Beyond epoch 200, the loss flattens, indicating convergence.

The reconstruction loss follows a similar trend to the total loss, decreasing steadily as the model learns to reconstruct the input data. The KLD loss, however, initially increases during the early epochs and begins to decrease after around 30 epochs. This behaviour suggests that in the early stages, the model prioritises capturing the underlying structure of the data to reduce reconstruction loss. As training progresses, the model shifts focus toward aligning the latent variables with the prior distribution. The decrease in KLD loss indicates improved disentanglement, as the model encourages independence across latent dimensions by regularising them toward the prior.

These curves demonstrate that the model was able to converge after approximately 200 epochs, balancing reconstruction quality and prior alignment.



(a) Training total loss over epochs.

(b) Training reconstruction loss over epochs.

(c) Training KLD loss over epochs.

Figure 5.2: Training loss components over epochs.

5.1.2 Challenges in Training VAEs

In this section, we discuss the limitations encountered when training our `double-pendulum-c-256` and `air-dancer-c-256` models. As a sanity check, we first

validated our VAE architecture on the Modified National Institute of Standards and Technology (MNIST) dataset, as detailed in the appendix (§ A). The model performed well on this simpler, non-dynamical dataset, confirming that the architecture itself is functional.

For `double-pendulum-c-256`, one major limitation was the dataset size, which is approximately four times smaller than that of `double-pendulum-z-256`. This limited data likely contributed to the model's failure to converge during training.

The `air-dancer-c-256` dataset poses additional challenges. While the double pendulum dataset has a known ground truth latent dimension of four, the air dancer system is estimated to require a latent space of around seven to eight dimensions, as suggested by Chen et al. (2021). However, due to time constraints and limited experience working with VAEs on more complex datasets, we used the same model architecture and latent dimensionality as for `double-pendulum-z-256`. This underparameterisation likely constrained the model's ability to represent the full complexity of the dynamics of the air dancer's dynamics, leading to suboptimal performance.

Moreover, recent work such as Xiao et al. (2024) highlights that VAEs can exhibit double descent behaviour, where increasing the dataset size initially improves test performance, then worsens due to overfitting, and eventually improves again when the model becomes sufficiently expressive to generalise the dataset. Our setup likely lies in the underfitting regime because the dataset is relatively small and the model is not large enough.

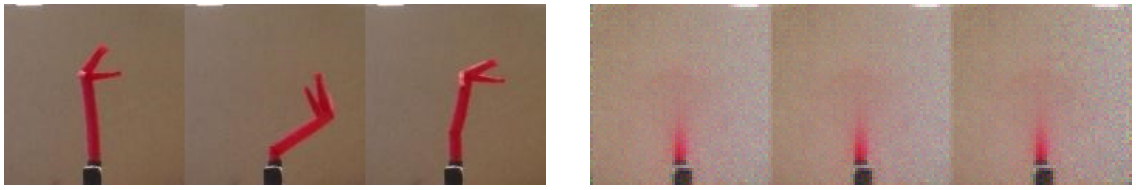
This is further evidenced by the reconstruction results shown in Figure 5.4. The model fails to reconstruct the detailed parts of the air dancer, although it manages to capture the background and the base of the air dancer. Most of the highly output frames appear highly blurred and lack distinctive features. Similar challenges in reconstructing complex systems are also visible shown in the work by Xiao et al. (2024), as shown in Figure 5.5. The left column shows ground truth images, the middle shows reconstructions from a VAE trained using standard training on a limited dataset, and the right shows reconstructions using DMaaPx (Diffusion Model as an approximation of the data distribution, a method that generates more samples using a diffusion model), supporting the argument that VAEs are generally difficult to train on high-dimensional chaotic dynamics.

Computational constraints also prevented us from significantly increasing the model size or training time. Further extensions could explore richer model architectures or larger datasets, potentially augmented through techniques such as noise injection or diffusion based synthetic sampling introduced by Xiao et al. (2024). These steps may

enable VAEs to better capture the underlying structure of `double-pendulum-c-256` and `air-dancer-c-256`.

(a) Expected output (b) Failed reconstruction output

Figure 5.3: Output comparison `double-pendulum-c-256`



(a) Expected output (b) Failed reconstruction output

Figure 5.4: Output comparison `air-dancer-c-256`

Figure 5.5: Comparison of reconstruction quality across different training methods

5.1.3 KLD Weight Tuning

We began by analysing the effect of KLD weight tuning on the test loss using the `double-pendulum-z-256` dataset. This analysis aims to determine the optimal regularisation strength that balances reconstruction and regularisation.

Figure 5.6 shows the test loss plotted against KLD weights, β. The results indicate the test loss remains consistently low for KLD weights, β in the range 10^{-10} to 10^{-7}, indicating stable and effective model performance. Although a slight fluctuation is observed within

this range, it does not significantly affect the reconstruction error, as confirmed in Figure 5.6. Beyond 10^{-6}, test loss increases sharply, indicating that excessive regularisation begins to dominate the loss function, leading to convergence issues and poor reconstructions.

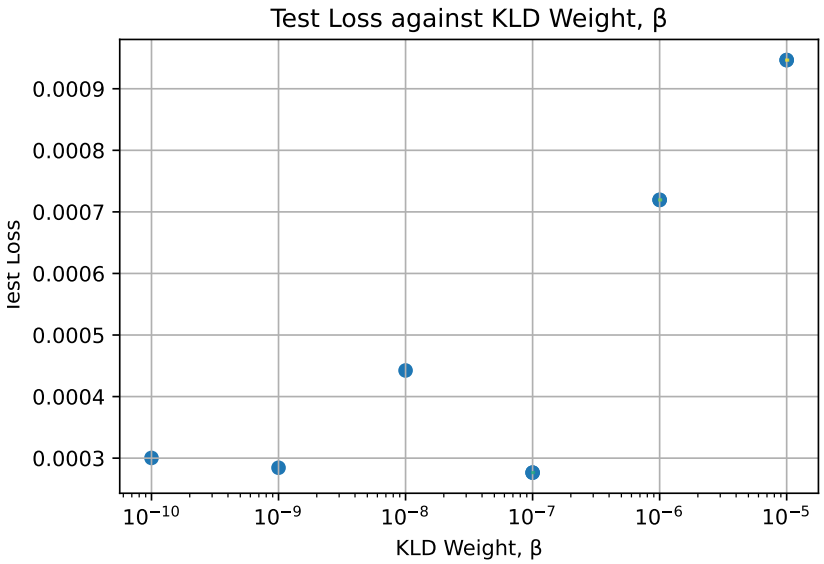


Figure 5.6: `double-pendulum-z-256`: Test loss plotted against different KLD weight (β) values.

Figure 5.7 shows the KLD loss plotted against KLD weights, β. To confirm that the changes in test loss are not solely due to fluctuations in the KLD term, we analysed the reconstruction error independently, as shown in Figure 5.7. The reconstruction error remains low and relatively stable for KLD weight, β, up to 10^{-7}, reinforcing the observation that the model performance is not compromised in this range. However, at higher KLD weight, β, the reconstruction error increases demonstrating the model's failure to capture key features of the input.

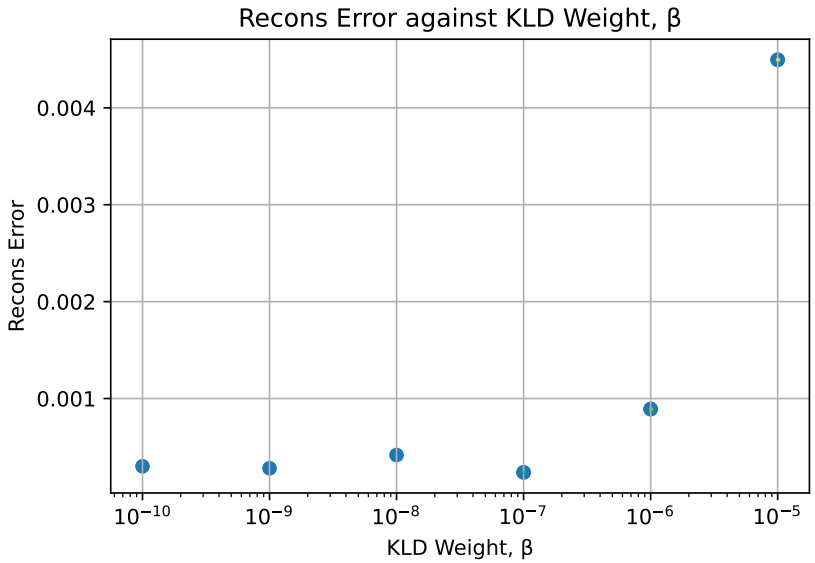


Figure 5.7: `double-pendulum-z-256`: Reconstruction error plotted against different KLD weight (β) values.

Finally, Figure 5.8 presents the KLD loss plotted against different values of the KLD weight, β. The figure shows that as KLD weight, β, increases, the KLD loss decreases. This is expected, as higher values of KLD weight, β, apply stronger regularisation, encouraging the latent space to more closely follow the prior distribution, which is a standard Gaussian in this case. Negative KLD loss for KLD weight, β, 10^{-6} and 10^{-5} indicates a numerical instability in training, suggesting that the model does not converge and fails to learn a meaningful latent representation. This result highlights the importance of carefully tuning KLD weight, β, to strike a balance between reconstruction accuracy and latent space regularity, ultimately promoting meaningful disentanglement.

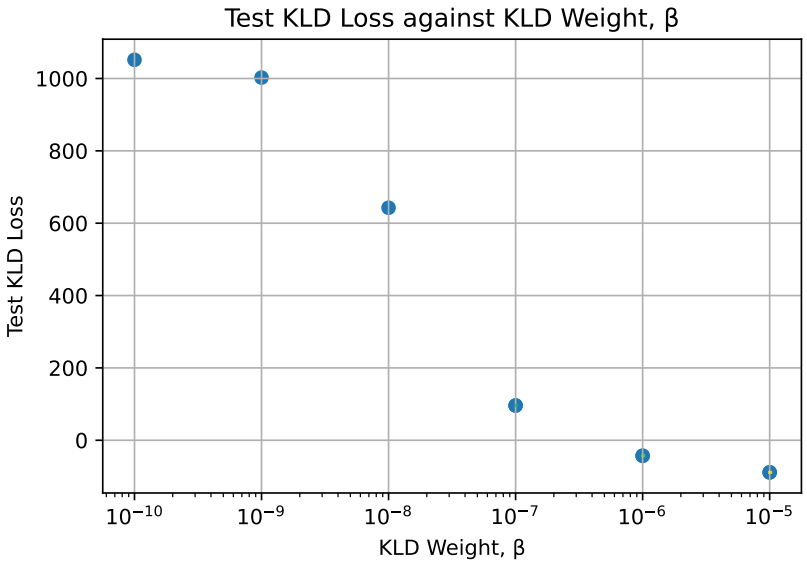


Figure 5.8: `double-pendulum-z-256`: Test KLD loss plotted against different KLD weight (β) values.

Figure 5.9 presents a visual comparison of the input sequence, ground truth, and model output for the `double-pendulum-z-256` dataset. The results shown correspond to the sixth test sample and illustrate the model's reconstruction performance.

(a) Input Sequence (b) Ground Truth (c) Model Output

Figure 5.9: Input, ground truth, and reconstruction of `double-pendulum-z-256`

5.1.3.1 Results for `double-pendulum-z-10`

The following results for `double-pendulum-z-10` follow similar trends as observed in `double-pendulum-z-256` . One notable observation is that initially, the KLD loss in Figure 5.12 for the `double-pendulum-z-10` case is approximately 25 times smaller than in Figure 5.8 for the `double-pendulum-z-256` case. This is expected, as models with more latent variables naturally accumulate a larger KLD loss due to the increased number of latent variables being regularised.

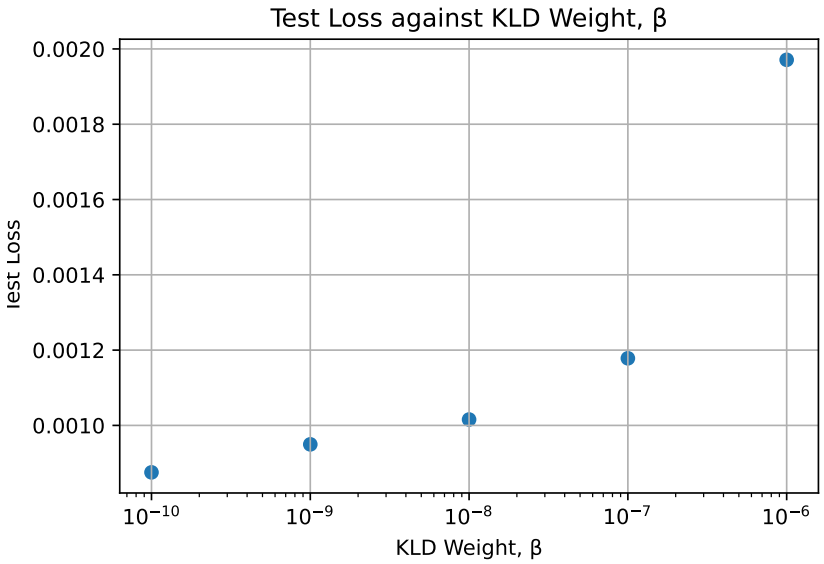


Figure 5.10: `double-pendulum-z-10`: Test loss plotted against different KLD weight (β) values.

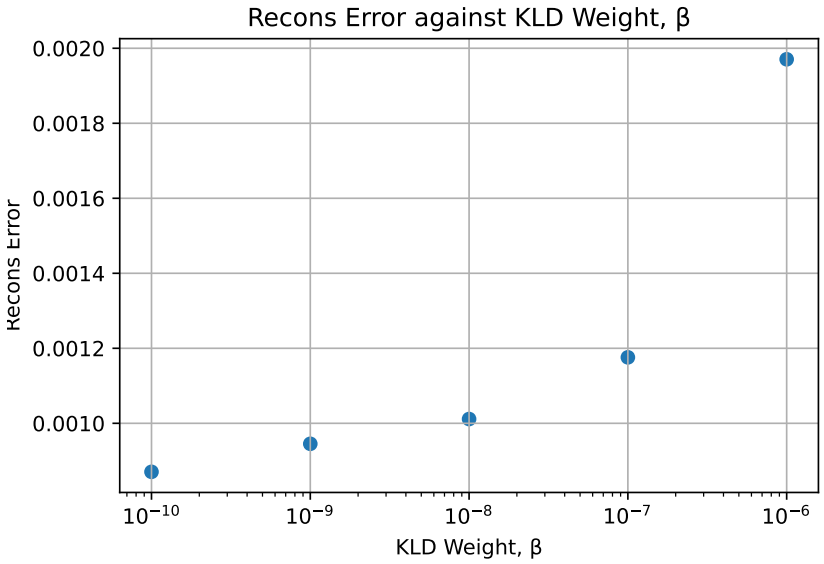


Figure 5.11: `double-pendulum-z-10`: Reconstruction error plotted against different KLD weight (β) values.

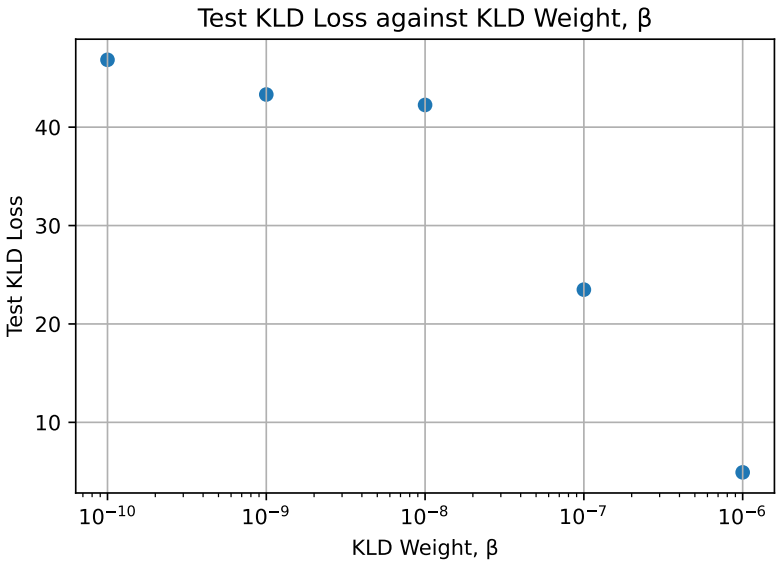


Figure 5.12: `double-pendulum-z-10`: KLD loss plotted against different KLD weight (β) values.

5.2 Latent Variable Selection

In this section, we present the results and analysis of the trained `double-pendulum-z-10` and `double-pendulum-z-256` performance across various evaluation metrics. We analyse the impact of tuning the KLD weight, β, selecting latent variables, and adjusting the model architecture. Observations from testing the model on `double-pendulum-z-10` and `double-pendulum-z-256` are provided, focusing on reconstruction accuracy, generalisation capabilities, and latent space structure. These results offer insights into the effectiveness of our optimisation approach and the model's ability to capture essential dynamics in complex systems.

5.2.1 `mean-zero` and `model-reduct`

In this section, we present the test loss as a function of the number of latent variables using the `mean-zero` (see § 4.4.1.1) and `model-reduct` (see § 4.4.1.2) methods. While we initially evaluated the importance of each latent variable independently, this approach does not account for interactions between variables. In practice, many latent dimensions contribute to reconstruction only when considered together. As a result, focusing solely on individual contributions can be misleading and may underestimate the importance of

variables that are only useful in combination. To overcome this limitation, we extended our analysis to explore subsets of latent variables, enabling us to capture joint effects and better identify which combinations preserve reconstruction quality.

The results are based on the `double-pendulum-z-10` dataset with a KLD weight, β of 1×10^{-7}. As shown in Figure 5.13, the blue (`mean-zero`) and orange (`model-reduct`) markers largely overlap across all values, indicating highly consistent results between the two approaches. Both trends suggest that retaining just two latent variables is sufficient to preserve performance, with minimal increase in test loss. This implies that the remaining eight latent dimensions are largely redundant and do not contribute significantly to reconstruction quality.

Our approach to selecting latent variables effectively reduced the number of variables needed for accurate reconstruction. Initially, all 10 latent variables were employed, but through pruning each latent dimension iteratively, we successfully decreased this number to 2. This reduction process led to a more interpretable latent space, enabling the model to maintain high reconstruction accuracy while using fewer latent variables.

As expected, reconstruction error decreased as the number of latent variables increased, aligning with the understanding that additional degrees of freedom enable the model to better capture the underlying data distribution. This selective pruning allows for a more compact yet effective representation in the latent space, balancing model complexity with reconstruction fidelity.

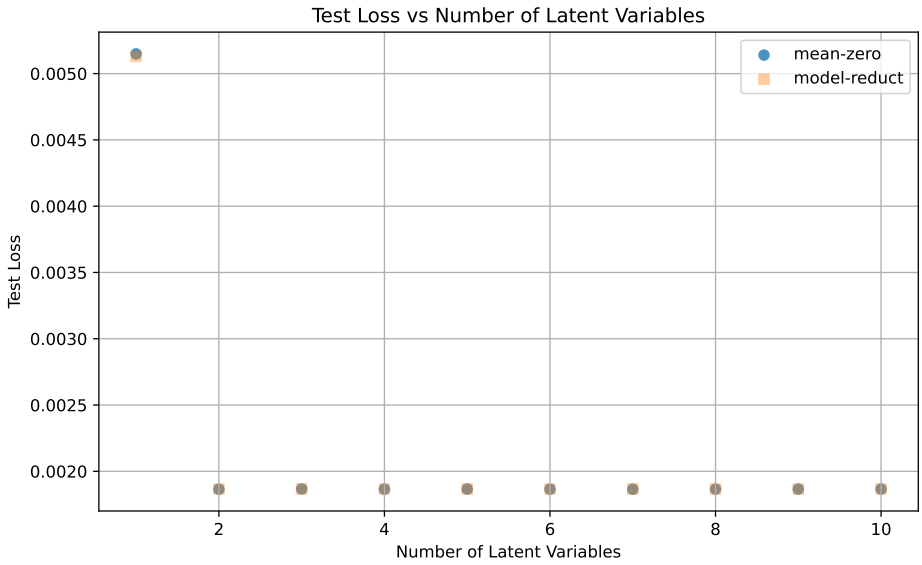


Figure 5.13: `double-pendulum-z-10`: Test loss vs. number of latent variables for the `mean-zero` and `model-reduct` methods.

5.2.1.1 Comparison of Reconstruction Error

We present a comparison between the `mean-zero` and `model-reduct` methods for the `double-pendulum-z-10` dataset. Figures 5.13 and 5.14 show the test loss across different numbers of latent variables, using a KLD weight, β of 1×10^{-7}. This setting provides a clearer analysis of both methods.

The results indicate that both `mean-zero` and `model-reduct` produce nearly identical test losses across all latent dimensions, with differences remaining below 0.35%. This suggests that either method can be used to estimate latent variable importance, with minimal impact on overall performance.

The main purpose of this comparison is to evaluate the reconstruction differences caused by each method. The only distinction lies in how unused latent variables are handled. `mean-zero` retains the original architecture but sets the means of selected dimensions to zero, while `model-reduct` removes the unselected variables entirely. Even when noise is set to zero, variations in matrix dimensions can lead to minor differences in sampling from the reparameterisation trick. To ensure consistency, we use `model-reduct` as the baseline for comparison.

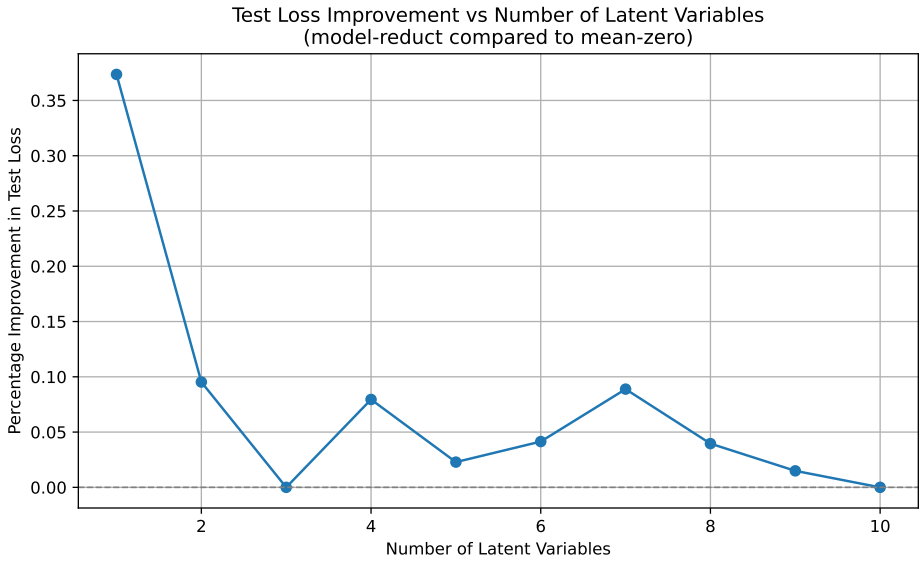


Figure 5.14: `double-pendulum-z-10`: Percentage difference in test loss vs. number of latent variables for the `mean-zero` and `model-reduct` methods.

However, both `mean-zero` and `model-reduct` are computationally expensive, even for a latent dimensionality of 10, requiring approximately 10 hours to complete. This exhaustive evaluation of the 2^{10} possible subsets is computationally prohibitive and does not scale well. To mitigate this inefficiency, we proceed to explore alternative latent variable selection methods that offer improved time complexity and scalability.

5.2.2 Shapley Values and Subset Selection Methods

Shapley values were computed to quantify each latent variable's average marginal contribution to the reconstruction loss across all possible subsets. By capturing both individual and interaction effects, they provide a foundational understanding of latent variable importance. This information helped inform and justify the subset selection process by highlighting which variables were consistently influential.

The subset selection process was driven by a heuristic inspired by the Minimum Description Length (MDL) principle. This heuristic is defined as the product of the training reconstruction loss and $\log_2(k+1)$, where k is the number of selected latent variables (Wu, 2020). Training loss was used rather than test loss to compute this heuristic in order to prevent training data leakage and ensure that the selection process does not overfit to the test set. This heuristic provides an efficient approximation of the trade-off

between reconstruction accuracy and model complexity.

To mitigate the risk of getting stuck in local minima, we adopted a replacement rule in all methods. A parent mask is only replaced if the best child subset achieves a heuristic value that is at least 10% lower than that of the parent mask. The results of the baseline method, which uses all latent variables, alongside those from the selection strategies applied to `double-pendulum-z-10` and `double-pendulum-z-256` with a fixed $\beta = 10^{-7}$, are summarised in Table 5.1 and Table 5.2.

`forward-select` starts from an empty mask and incrementally adds latent variables that yield the greatest improvement in the heuristic. It performs well on `double-pendulum-z-10`, selecting only 2 latent variables while achieving near-baseline performance. However, its strictly additive and greedy nature limits the ability to revise earlier decisions, making it prone to suboptimal selections in high-dimensional settings like `double-pendulum-z-256`.

`backward-elim` begins with the full set and iteratively removes the least informative variables. This top-down strategy is effective for eliminating clearly redundant dimensions, as shown in `double-pendulum-z-10`. However, its strictly subtractive process is conservative and does not revisit earlier removals, often resulting in larger-than-necessary subsets. In `double-pendulum-z-256`, it retained 247 out of 256 variables, underscoring its limited aggressiveness in pruning.

`greedy-select` explores the local neighbourhood of a randomly chosen subset by toggling one variable at a time, accepting new configurations if they improve the heuristic. While its bidirectional flexibility allows for both additions and removals in a single run, it frequently converges to suboptimal local minima, especially in high-dimensional settings. As seen in `double-pendulum-z-256`, it selected over 129 latent variables yet yielded poorer reconstruction performance, suggesting difficulty in escaping suboptimal configurations.

`genetic-algo` simulates evolutionary dynamics by iteratively evolving a population of binary masks through crossover and mutation. Although designed for global search and capable of capturing complex interactions, its performance is highly sensitive to parameter tuning and population diversity. In both `double-pendulum-z-10` and `double-pendulum-z-256`, it consistently selected large subsets with high test loss, indicating issues such as slow convergence and ineffective balancing between exploration and exploitation.

These results highlight the advantages and limitations of different subset selection strategies, aligning with the findings of Guyon and Elisseeff (2003). As they observed, greedy methods that build or reduce subsets step by step can be effective, but they often make early decisions that are hard to reverse. In our case, `forward-select` builds compact subsets but struggles in high-dimensional settings because it cannot revise earlier additions. Coversely, `backward-elim` starts with the full set and removes variables, which works well in small latent spaces but tends to keep too many variables when the initial set is large, as shown in Table 5.2. The poor performance of `greedy-select` and `genetic-algo` on `double-pendulum-z-256` supports the idea that simple heuristic searches can easily get stuck and fail to find better solutions without more powerful exploration strategies. Overall, our findings suggest that a carefully selected subset of latent variables may be sufficient to represent the full latent space.

Table 5.1: Summary of Latent Variable Selection Results on `double-pendulum-z-10`

Method	**Num of LV**	**Test Loss**	**Reconstruction Error**	**Test KLD Loss**
Baseline	10.0	0.001856	0.001852	11.358
`forward-select`	2.0	0.001867	0.001865	13.955
`backward-elim`	4.0	0.001866	0.001865	13.286
`greedy-select`	2.0	0.001867	0.001865	13.955
`genetic-algo`	3.7	0.004042	0.004041	8.838

Table 5.2: Summary of Latent Variable Selection Results on `double-pendulum-z-256`

Method	**Num of LV**	**Test Loss**	**Reconstruction Error**	**Test KLD Loss**
Baseline	256.0	0.000276	0.000238	96.054
`forward-select`	17.0	0.004444	0.004424	50.139
`backward-elim`	247.0	0.000289	0.000250	99.111
`greedy-select`	129.7	0.005225	0.005200	62.615
`genetic-algo`	127.3	0.006067	0.006051	41.410

5.2.3 Principal Component Analysis (PCA)

To evaluate the effectiveness of the proposed framework, we apply Principal Component Analysis (PCA) to a Variational Autoencoder (VAE) trained on `double-pendulum-z-256`. A key component in optimising the model's performance involved careful tuning of the KLD weight, β.

5.2.3.1 PCA on Latent Variables

To assess whether the learned latent space could be further compressed without significantly degrading the quality of reconstruction, we applied PCA to the latent variables. PCA enables dimensionality reduction by projecting the data onto orthogonal axes that capture the maximum variance. This analysis allows us to evaluate whether a compact set of latent variables is sufficient to retain key information, thereby providing insight into the degree of disentanglement achieved by the model.

As shown in Figure 5.15, the reconstruction error decreases significantly as the number of principal components retained increases from 1 to 6. This aligns with our hypothesis that additional latent variables contribute to better model performance. The sharp decline from 3 to 6 latent variables suggests that the remaining components still carry critical information. The initial spike at 1 to 2 latent variable indicates that the model is unable to represent the system dynamics adequately at such low dimensionality particularly under low KLD Weight, β. Beyond six components, the reconstruction error plateaus, implying that additional latent variables does not improve the model and indicating that the top k principal components captured the underlying relationship.

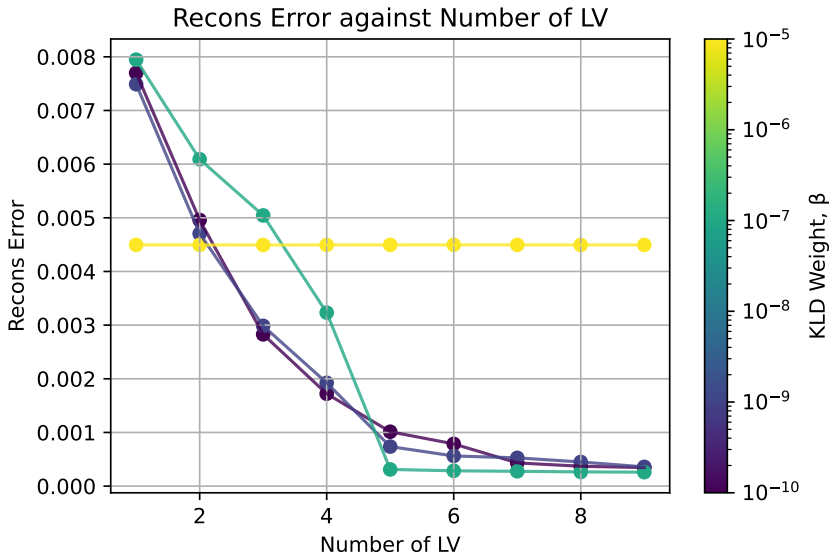


Figure 5.15: `double-pendulum-z-256`: Test Reconstruction Loss vs. Number of Latent Variables. Each curve corresponds to a different KLD weight β, as indicated by the colour bar on the right.

Figure 5.16 presents the reconstruction error across various KLD weights, β, intial (blue) and after applying PCA (red) to the latent space. PCA was applied to retain only the top components that explain 95% of the variance in the latent space.

At low KLD weights, β, (e.g., 10^{-9}), we observe a significant increase in reconstruction error after PCA, indicating that information is distributed across many entangled latent dimensions. As a result, compression via PCA leads to substantial loss. As KLD weights, β, increases to intermediate values (e.g., 10^{-7}), the reconstruction error after PCA closely matches the uncompressed baseline, suggesting better disentanglement and that the model is learning a more compact and structured latent representation. Notably, for KLD weights, β (e.g. 10^{-7}), reducing the 256-dimensional latent space to just 5 principal components was sufficient to retain the full variance, highlighting that a compact subset of latent variables can effectively preserve the essential information required for reconstruction.

As the KLD weight, β, continues to increase, the gap between the blue and red points narrows, indicating a more disentangled and compressible latent space. However, at higher KLD weights, β (e.g., 10^{-5}), both versions exhibit elevated reconstruction error due to poor convergence, and PCA provides no tangible benefit. This deterioration may caused by excessive regularisation, which hinders the model's ability to capture meaningful features regardless of dimensionality reduction.

Figure 5.16: `double-pendulum-z-256`: Comparison of Test Reconstruction Error vs. KLD Weight, β, Initial (blue) and `pca` (red)

These findings collectively demonstrate that PCA serves as a valuable post hoc tool for evaluating the degree of disentanglement in the learned latent space. When a well-balanced KLD weight, β, is used, most of the meaningful variation in the data can be captured by a small number of principal components. Based on our experiments, KLD weight, $\beta = 10^{-7}$ consistently yielded the best trade-off between reconstruction and regularisation.

5.2.3.2 Discussion of PCA on Latent Spaces

When the KLD weight, β, is small, VAEs perform like standard autoencoders. Without regularisation, the latent space becomes highly entangled. Each latent variable mixes different directions of variation, and there is no clear structure for PCA to extract. Since PCA assumes that the data lies along a small number of linear directions of high variance (Jolliffe, 2002), it performs poorly in this case because too much information is spread across the latent space. As a result, applying PCA to an unregularised latent space leads to large losses in reconstruction quality. This behaviour aligns with the findings of Chen et al. (2021), where an intrinsic dimension estimator by Levina and Bickel (2004) was used, which is based on local geometry rather than global variance, to better capture the structure of an entangled autoencoder latent space.

However, when KLD weight, β is set to a more balanced value, the VAE begins to disentangle the latent space. The model puts more emphasis on regularisation and aligns the latent variables with distinct features, making the space more structured and compressible. In this case, PCA becomes more effective because most of the important variation is captured in a few principal components.

Interestingly, Rolinek et al. (2019) showed that this behaviour emerges even though VAEs are not explicitly designed to align with principal directions. They argued that the diagonal Gaussian assumption in the encoder, combined with the randomness from sampling, naturally pushes the latent space toward orthogonality. This encourages the VAE to distribute meaningful information along nearly orthogonal directions. When the VAE is properly regularised, applying PCA to the latent space works well because the model has already structured the space in a way that PCA can readily interpret.

In summary, PCA is a useful tool for evaluating disentanglement, but only when the latent space is well regularised. When there is no structure in the latent code, PCA fails because the underlying representation is not yet meaningful.

5.2.4 Comparison of Latent Variable Selection Techniques

In this section, we compare various techniques used for selecting latent variables from a trained Variational Autoencoder (VAE). The goal is to identify the most informative subset of latent dimensions while maintaining high reconstruction fidelity.

To assess the performance of different latent variable selection strategies, we compare a range of methods applied to the `double-pendulum-z-10` and `double-pendulum-z-256` datasets. These include both individual variable importance estimators and subset selection techniques, covering baseline approaches such as `mean-zero` and `pca`, as well as structured selection methods like `forward-select`, `backward-elim`, `greedy-select`, and `genetic-algo`.

For `double-pendulum-z-10` with KLD weight $\beta = 1\times10^{-7}$, we include all methods in the comparison, including the exhaustive `model-reduct`. However, due to the prohibitive cost of testing and the impracticality of traversing the entire search space, we omit `model-reduct` from the `double-pendulum-z-256` experiments. The summary tables below report the number of selected latent variables, the reconstruction error, and the test KLD loss for each method.

Based on Table 5.3, we observe that `model-reduct`, `forward-select`, and `pca` achieve the best performance, balancing low reconstruction error with a small number of latent variables. `model-reduct` has the lowest reconstruction error, which makes sense as it exhaustively evaluates all possible latent variable subsets. However, this method is extremely computationally expensive—testing just one configuration takes around 10 hours, making it impractical for larger models or multiple evaluations.

On the other hand, `forward-select` offers a much more efficient alternative. It provides near-identical performance to `model-reduct` while reducing runtime to about 10 minutes per run. This suggests that forward selection is a strong practical compromise between performance and efficiency. Meanwhile, `pca` trades off a slight increase in reconstruction error for major computational savings, completing in under 5 minutes. Its simplicity and speed make it a valuable tool, especially when model interpretability or fast iteration is needed.

From Table 5.4, `pca` is again shown to perform remarkably well, with low reconstruction error achieved using just 5 latent variables. This finding aligns with the results presented by Chen et al. (2021), where the intrinsic dimension of the double pendulum system was

estimated to be 4.71 ± 0.03, close to the ground truth of 4. While intrinsic dimension estimators offer useful insights, they often yield non integer values, which can be undesirable in dynamical systems where the ground truth is an integer. In contract, PCA provides a more interpretable and integer based estimate, making it a more practical tool for such dynamical systems.

Although `backward-elim` achieves the lowest reconstruction error, it retains 247 out of 256 latent variables, offering minimal reduction in model complexity. This highlights a key limitation of backward elimination in high-dimensional spaces as it tends to make only minor pruning decisions. In contrast, `pca` aggressively compresses the latent space with negligible performance loss and completes in a fraction of the time. This makes it a particularly effective method when rapid evaluation and substantial dimensionality reduction are needed.

Overall, `pca` stands out as a fast and effective heuristic for identifying compact representations, especially in settings where interpretability or computational constraints make exhaustive subset search infeasible.

Table 5.3: Comparison of Latent Variable Selection Methods on `double-pendulum-z-10`

Method	**Num of LV**	**Test Loss**	**Reconstruction Error**	**Test KLD Loss**
Baseline	10.00	0.001856	0.001852	11.358
`model-reduct`	2.00	0.001867	0.001865	13.955
`forward-select`	2.00	0.001867	0.001865	13.955
`backward-elim`	4.00	0.001866	0.001865	13.286
`greedy-select`	2.00	0.001867	0.001865	13.955
`genetic-algo`	3.70	0.004042	0.004041	8.838
`pca`	2.05	0.002048	0.002044	11.355

Table 5.4: Comparison of Latent Variable Selection Methods on `double-pendulum-z-256`

Method	**Num of LV**	**Test Loss**	**Reconstruction Error**	**Test KLD Loss**
Baseline	256.00	0.000276	0.000238	96.054
`forward-select`	17.00	0.004444	0.004424	50.139
`backward-elim`	247.00	0.000289	0.000250	99.111
`greedy-select`	129.67	0.005225	0.005200	62.615
`genetic-algo`	127.33	0.006067	0.006051	41.410
`pca`	5.00	0.000348	0.000309	95.759

Chapter 6

Conclusion

This project presents an enhanced approach for identifying latent variables in dynamical systems using Variational Autoencoders (VAEs), with a particular focus on KLD weight tuning and latent variable selection. These optimisation strategies reduce model complexity while preserving low reconstruction error. Through this process, a compact set of latent variables was identified that serve as strong candidates for the system's underlying state variables, promoting disentanglement and interpretability.

Our experimental results show that this two-step optimisation framework enables high-quality reconstructions using significantly fewer latent variables than the original 10 or 256, demonstrating that much of the information can be captured in a more compact, structured form. This improved balance between regularisation and fidelity supports the discovery of meaningful latent structure in complex dynamical systems.

While the proposed framework demonstrates strong potential in identifying compact latent representations, our experiments also highlight the challenges of training VAEs. In particular, the poor performance on the `double-pendulum-c-256` and `air-dancer-c-256` datasets reflects well-documented training instabilities in VAEs, especially when the model capacity and dataset size are limited. These difficulties are consistent with recent findings by Xiao et al. (2024), who observe that VAEs can exhibit double descent behaviour and require sufficiently expressive architectures to generalise effectively.

Nonetheless, VAEs remain a principled and theoretically grounded approach for uncovering disentangled latent space in complex systems. VAEs can incorporate probabilistic reasoning and encourage a structured regularised latent space, makes them well suited for state variable discovery in dynamical system. While convergence can be challenging under constrained conditions, careful tuning and thoughtful architecture can yield interpretable and compact latent representations. If the goal is to achieve disentangled and interpretable

latent representations, VAEs are an appropriate choice. However, if the objective is solely on high quality reconstruction, alternative generative models such as Generative Adversarial Networks (GANs) may be more effective, as they typically produce sharper outputs without emphasising disentanglement.

Looking ahead, we hope this work provides practical insights into improving the robustness and adaptability of VAEs. The techniques explored here may also be extended to other generative models. Ultimately, this approach contributes to the broader effort of state variable discovery, offering a scalable and principled tool for analysing and modelling high-dimensional dynamical systems.

6.1 Future Work

To further enhance the performance and interpretability of our VAE model, several directions can be pursued in future work. These efforts could broaden the applicability of our framework and provide deeper insights into latent variable modelling in VAEs.

6.1.1 Exploring Higher-Dimensional Datasets

Our findings suggest that challenges with training VAEs become more pronounced when modelling high-dimensional dynamical systems, particularly under limited data or model capacity. Future work could explore the use of richer model architectures and larger datasets to mitigate these issues. In particular, data augmentation strategies such as noise injection or synthetic sampling using diffusion models (Xiao et al., 2024) may enhance generalisation and improve reconstruction fidelity. These extensions would support the development of more robust and scalable representation learning frameworks for high-dimensional, chaotic systems.

To evaluate the model's scalability and robustness, future work may involve testing on datasets with higher-dimensional features. Specifically, datasets proposed by Chen et al. (2021), including the Rigid Double Pendulum, Elastic Double Pendulum, and Swing Stick datasets. These datasets possess higher latent dimensions, providing an opportunity to examine more complex relationships within the data and evaluate the VAE's ability to manage increased complexity in latent representations. Additionally, varied backgrounds, such as red and green stripes can be introduced, to further assess the model's resilience. For instance, the double pendulum dataset can be projected into a higher-dimensional

space to verify whether the required latent dimensions remain consistent.

6.1.2 Comparing Latent Variable Selection Across Architectures

While this project focused on latent variable selection in VAEs, future work could explore applying similar techniques to other models in the same family, such as standard Autoencoders or GANs. It would be valuable to investigate how methods like Shapley values, forward selection, or PCA perform under different training objectives and latent space structures. Comparing across architectures could help clarify whether these approaches generalise well and offer insights into how latent variables behave in models that do not rely on regularisation through the KLD term.

An open question is whether intrinsic dimension estimators, which usually give non-integer values, can be adjusted or combined with some kind of structure or prior like the latent space of VAEs to give more meaningful integer results. This would be especially useful for dynamical systems, where the true number of degrees of freedom is often known and should be an integer. One possible direction could be to explore some intrinsic dimension estimators introduced by Chen et al. (2021). It could also be interesting to combine these estimators with model-based approaches like PCA or Shapley guided subset selection to make the results more stable and aligned with what we expect. This might help in finding a more reliable way to estimate the minimal number of latent variables across different model types.

Inspired by the nested autoencoder architecture proposed by Chen et al. (2021), future work could explore implementing a nested VAE structure, where one VAE encodes local dynamics and another captures high level representations. In this setup, the inner VAE could replace the intrinsic dimension estimator by learning to disentangle the most informative latent variables directly through reconstruction, rather than relying on external estimation. This setup may be particularly useful for very high dimensional systems, as it allows for progressive compression and may improve both convergence and latent space structure.

6.1.3 Enhanced Disentanglement Analysis

To gain deeper insights into how the model separates meaningful features, future work may involve conducting a more fine-grained analysis of each latent variable's contribution.

This includes evaluating the role of individual latent variables in capturing specific aspects of the system, such as motion or spatial configurations. Such analysis would enhance interpretability, allowing for a more comprehensive understanding of the model's ability to effectively distinguish distinct features of a system.

A promising direction is to explore additional latent variable selection techniques beyond those already implemented—such as genetic algorithms, forward–backward selection, or hybrid methods. These approaches may better capture interactions between variables and help identify subsets that jointly contribute to reconstruction or downstream tasks more effectively.

One promising direction is the application of symbolic regression to the latent space. Symbolic regression can be used to extract interpretable mathematical expressions that relate latent variables to known physical quantities, offering a principled way to identify which dimensions encode meaningful dynamics (Cranmer et al., 2020). This approach could help verify whether the learned latent variables correspond to the true underlying state variables of the dynamical system. Symbolic regression has been applied in prior work to discover interpretable representations and to reverse engineer physical laws from data (Cranmer et al., 2020). Incorporating such techniques would significantly improve the interpretability of the model and aid in evaluating the effectiveness of disentanglement.

Bibliography

Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M., Kawarabayashi, K.-i., & Nett, M. (2018). "Extreme-value-theoretic estimation of local intrinsic dimensionality". *Data Mining and Knowledge Discovery*, *32*, 1–38.

Bonheme, L., & Grzes, M. (2022). "Fondue: An algorithm to find the optimal dimensionality of the latent representations of variational autoencoders". https://arxiv.org/abs/2209.12806

Brin, M., & Stuck, G. (2002). "Introduction to dynamical systems". Cambridge university press.

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). "Sparse identification of nonlinear dynamics with control (sindyc)". (Tech. rep.). Cornell University Library, arXiv.org.

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., & Lerchner, A. (2018). "Understanding disentangling in β-vae". https://arxiv.org/abs/1804.03599

Campadelli, P., Casiraghi, E., Ceruti, C., & Rozza, A. (2015). "Intrinsic dimension estimation: Relevant techniques and a benchmark framework". *Mathematical Problems in Engineering*, *2015*, 1–21.

Chavelli, F., Khoo, Z.-Y., Wu, D., Low, J. S. C., & Bressan, S. (2024). "Physics-informed discovery of state variables in second-order and hamiltonian systems". https://arxiv.org/abs/2408.11691

Chavelli, F., Zi-Yu, K., Low, J. S. C., & Bressan, S. (2023). "Assessing the effectiveness of intrinsic dimension estimators for uncovering the phase space dimensionality of dynamical systems from state observations: A comparative analysis". In C. Strauss, T. Amagasa, G. Kotsis, A. M. Tjoa, & I. Khalil (Eds.), *Database and expert systems applications* (pp. 259–265). Springer Nature Switzerland.

Chen, B., Huang, K., Raghupathi, S., Chandratreya, I., Du, Q., & Lipson, H. (2021). "Discovering state variables hidden in experimental data". https://arxiv.org/abs/2112.10755

Costa, J., & Hero, A. (2007, December). "Determining intrinsic dimension and entropy of high-dimensional shape spaces".

Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., & Ho, S. (2020). "Discovering symbolic models from deep learning with inductive biases". https://arxiv.org/abs/2006.11287

Facco, E., d'Errico, M., Rodriguez, A., & Laio, A. (2017). "Estimating the intrinsic dimension of datasets by a minimal neighborhood information". *Scientific Reports*, *7*(1).

Farahmand, A.-m., Szepesvári, C., & Audibert, J.-Y. (2007). "Manifold-adaptive dimension estimation". *ACM International Conference Proceeding Series*, *227*, 265–272.

Ghorbani, A., & Zou, J. (2019). "Data shapley: Equitable valuation of data for machine learning". https://arxiv.org/abs/1904.02868

Grassberger, P., & Procaccia, I. (1983). "Measuring the strangeness of strange attractors". *Physica D: Nonlinear Phenomena*, *9*(1), 189–208.

Guyon, I., & Elisseeff, A. (2003). "An introduction to variable and feature selection". *J. Mach. Learn. Res.*, *3*(null), 1157–1182.

Haro, G., Randall, G., & Sapiro, G. (2008). "Translated poisson mixture model for stratification learning (preprint)". *International Journal of Computer Vision*, *80*, 358–374.

Hein, M., & Audibert, J.-Y. (2005). "Intrinsic dimensionality estimation of submanifolds in rd". *Proceedings of the 22nd International Conference on Machine Learning*, 289–296.

Hinton, G. E., & Salakhutdinov, R. R. (2006). "Reducing the dimensionality of data with neural networks". *Science*, *313*(5786), 504–507. Retrieved April 3, 2025, from http://www.jstor.org/stable/3846811

Jolliffe, I. T. (2002). "Principal component analysis and factor analysis". In *Principal component analysis* (pp. 150–166). Springer.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). "Physics-informed machine learning". *Nature Reviews Physics*, *3*(6), 422–440.

Kingma, D. P., & Welling, M. (2019). "An introduction to variational autoencoders". *Foundations and Trends® in Machine Learning*, *12*(4), 307–392.

Kingma, D. P., & Welling, M. (2022). "Auto-encoding variational bayes". https://arxiv.org/abs/1312.6114

Levina, E., & Bickel, P. (2004). “Maximum likelihood estimation of intrinsic dimension”. *Advances in neural information processing systems*, *17*.

Ljung, L. (1999). “System identification: Theory for the user”. (2nd;2nd;). Prentice Hall.

Oh, I.-S., Lee, J.-S., & Moon, B.-R. (2004). “Hybrid genetic algorithms for feature selection”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(11), 1424–1437.

Qian, E., Kramer, B., Peherstorfer, B., & Willcox, K. (2020). “Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems”. *Physica D: Nonlinear Phenomena*, *406*, 132401.

Rolinek, M., Zietlow, D., & Martius, G. (2019). “Variational autoencoders pursue pca directions (by accident)”. https://arxiv.org/abs/1812.06775

Rozza, A., Lombardi, G., Ceruti, C., Casiraghi, E., & Campadelli, P. (2012). “Novel high intrinsic dimensionality estimators”. *Machine Learning*, *89*.

Schmid, P., & Sesterhenn, J. (2008). “Dynamic mode decomposition of numerical and experimental data”. *Journal of Fluid Mechanics*, *656*.

Schmidt, M., & Lipson, H. (2009). “Distilling free-form natural laws from experimental data”. *Science*, *324*(5923), 81–85. Retrieved November 2, 2024, from http://www.jstor.org/stable/20493639

Shlens, J. (2014). “A tutorial on principal component analysis”. https://arxiv.org/abs/1404.1100

Wu, T. (2020). “Intelligence, physics and information – the tradeoff between accuracy and simplicity in machine learning”. https://arxiv.org/abs/2001.03780

Xiao, T. Z., Zenn, J., & Bamler, R. (2024). “A note on generalization in variational autoencoders: How effective is synthetic data & overparameterization?” https://arxiv.org/abs/2310.19653

Appendix

A Evaluation of VAE Models on MNIST

To evaluate the effectiveness of the proposed convolutional VAE architecture, we compare it with a standard fully connected VAE baseline. The objective of this comparison is to determine whether the convolutional VAE yields improved performance and more meaningful latent representations.

Both models were trained on the Modified National Institute of Standards and Technology (MNIST) dataset. A sample input image is shown in Figure A.1.

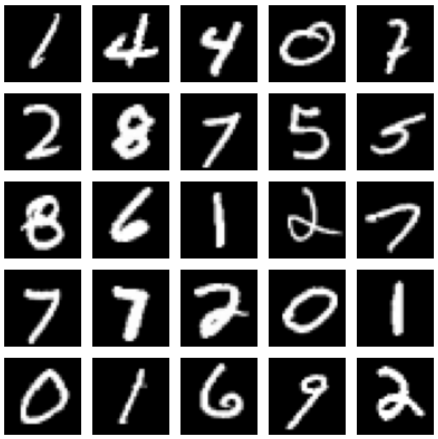


Figure A.1: Example Input from the MNIST Dataset

For this evaluation, we trained three variants of the VAE. The first model is a generic baseline VAE that consists entirely of fully connected layers, with a loss function combining Binary Cross-Entropy (BCE) and Kullback-Leibler Divergence (KLD). The second model is the proposed convolutional VAE which has both convolutional and fully connected layers and employs BCE and KLD in its loss function. The third model has the same implementation as the second but uses Mean Squared Error (MSE) instead of BCE.

To assess each model's generative ability, we visualise the reconstruction generated by varying the two inputs fed to the decoder z_0 and z_1, with a scaling factor of 3. The corresponding visualisations are shown in Figures A.2A.3A.4

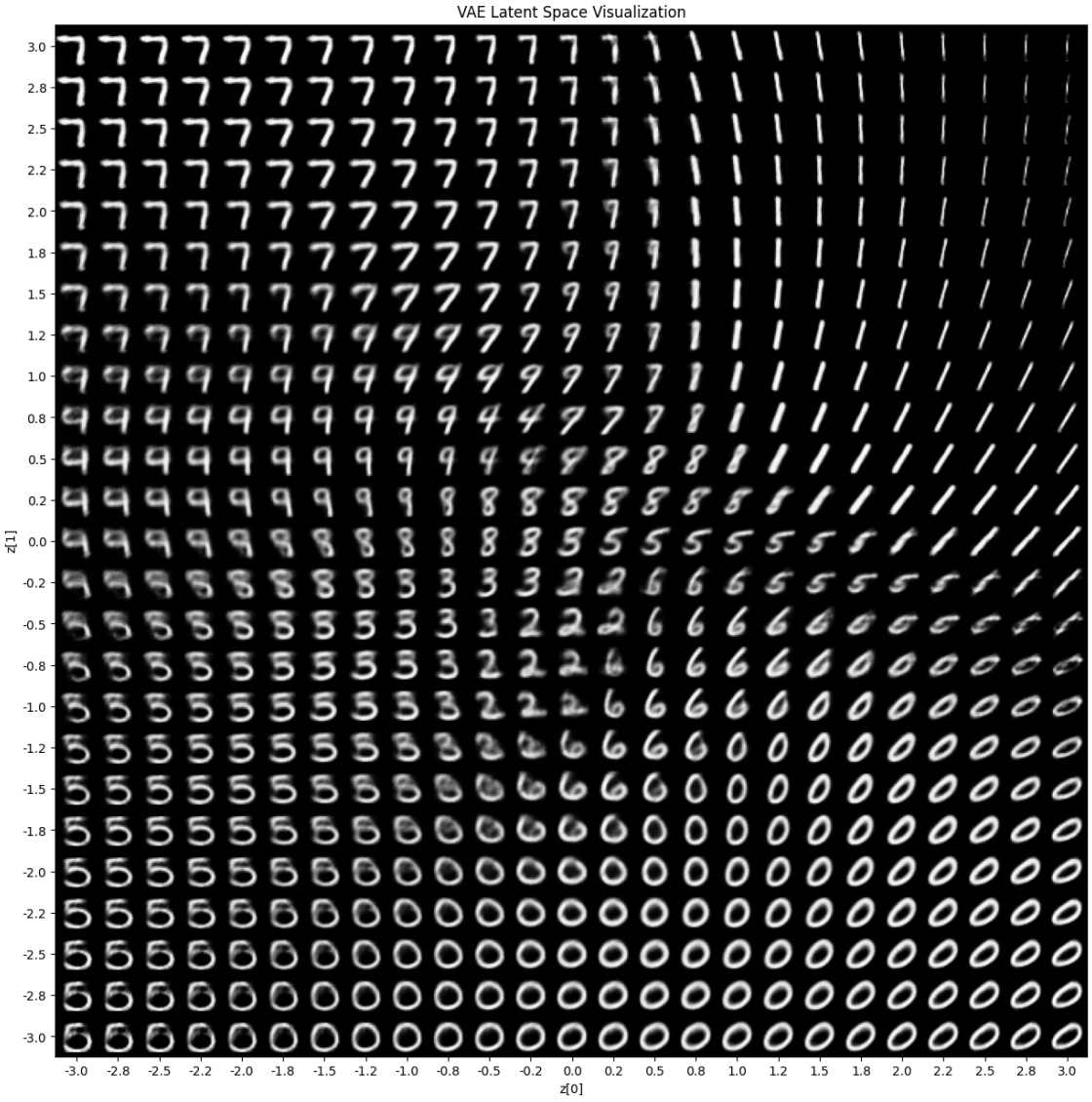


Figure A.2: Generic Model

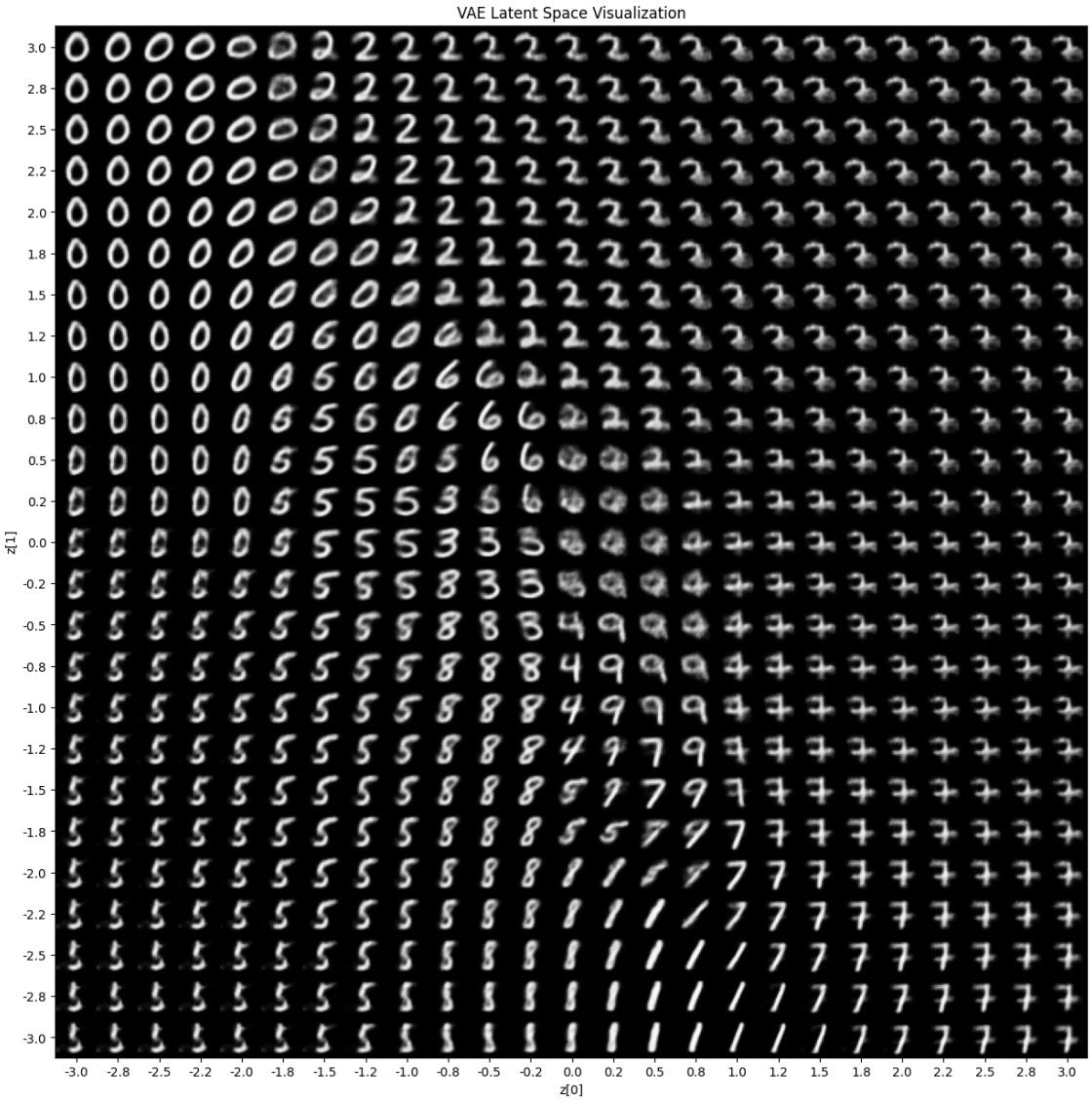


Figure A.3: Proposed Model with BCE Loss

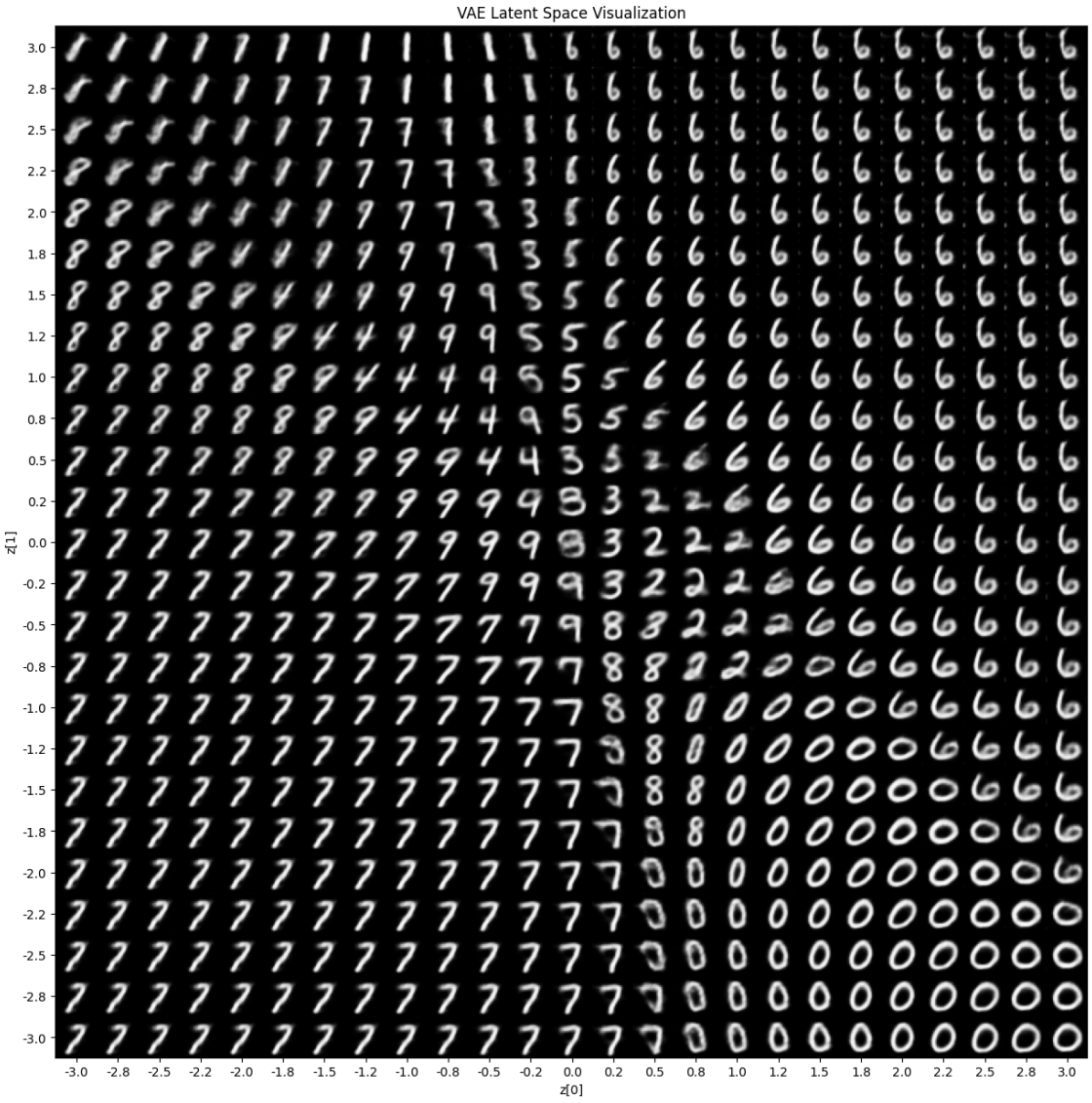


Figure A.4: Proposed Model with MSE Loss

A.1 Quantitative Evaluation

Model	Test Loss	Reconstruction Loss (BCE)	KLD Loss
Generic Model	13569.56	12884.67	6.8489
Proposed Model with BCE	12856.21	12843.6748	12.534
Proposed Model with MSE	13305.52	13290.95	14.5723

Table A.1: BCE Loss Table

A.2 Discussion

Models trained with BCE consistently achieved slightly lower test losses compared to those trained with MSE. This result is expected, as the MNIST dataset consists of grayscale images with pixel values in the range between 0 and 1, making BCE a more appropriate loss function.

Given that our data set comprises continuous valued inputs such as RGB images, we model the decoder as a Gaussian distribution, in line with the formulation in the original VAE framework Kingma and Welling, 2022. As a result, we used Mean Squared Error (MSE) as the reconstruction loss is more appropriate than Binary Cross Entropy (BCE). BCE is more generally used when the decoder models a Bernoulli distribution, such as MNIST and MSE is more typically used when the decoder models a Gaussian distribution.

Despite differences in test loss, the reconstruction losses remain similar across models, suggesting that all models have effectively learned the underlying relationship of the MNSIT dataset. Visualisation of $z[0]$ and $z[1]$ shows that the learned latent spaces produce smooth transitions between the digits and that the latent variables do not appear to perfectly follow a standard normal prior despite the use of KLD loss in the loss function. This is because the KLD regularisation may not be sufficiently strong to fully restrict the posterior.

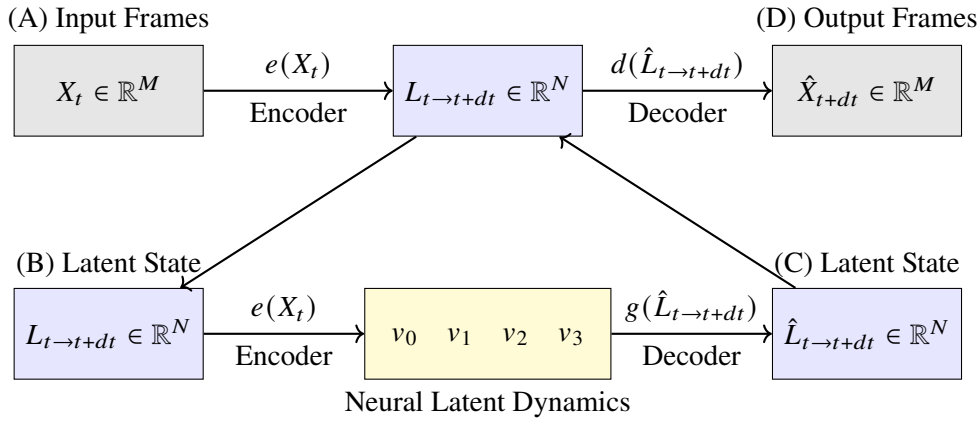


Figure A.5: Overview of the neural network model proposed by Chen et al. (2021)