CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Learning Dynamical Systems using Deep Generative Models

Master's Thesis in Computer Science and Engineering

Adrian Lundberg

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis 2021

Learning Dynamical Systems using Deep Generative Models

Adrian Lundberg

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Learning Dynamical Systems using Deep Generative Models
Adrian Lundberg

Supervisor: Lars Hammarstrand, Electrical Engineering
Advisor: Shuangshuang Chen, Volvo Cars
Examiner: Lars Hammarstrand, Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2021

Learning Dynamical Systems using Deep Generative Models
Adrian Lundberg
Department of Electrical Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Humans have a great ability to understand the dynamics of objects and are able to predict how the interactions of these objects will evolve over time. A probabilistic model that is able to learn this, has a variety of appealing applications, from object tracking to robotic planning. While learning such dynamics directly from high-dimensional data, like video frames, has shown to be challenging, recent advances in deep learning might provide the tools needed to learn the temporal dependencies present in the data.

In this thesis, we study deep generative models such as variational autoencoders, generative adversarial networks and normalizing flow models to see how well they are able to learn the dynamics of linear and nonlinear dynamical systems. The models are compared quantitatively as well as qualitatively by visually comparing the models' abilities to reconstruct and predict data. Additionally, models are evaluated from the perspective of representation learning, where the representation a model has learned is given an interpretation and a metric score. Our results show that learning dynamical systems using deep generative models is a challenging task, but that combining elements of these different models can be helpful. The results also show that a model's learned representation can be useful for explaining the model's prediction ability.

Acknowledgments

First of all, I would like to thank Volvo Cars for providing me the opportunity to conduct this thesis. I'm especially thankful for the support, enthusiasm and advice given by my advisor Shuangshuang Chen. I would also like to express my gratitude to Lars Hammarstrand, my supervisor at Chalmers, for guidance and assistance as well as help with formalities during the thesis.

Adrian Lundberg, Gothenburg, February 2021

Contents

List of Figures

List of Tables

1
Introduction

The world is filled with physical phenomena which evolve over time, ranging from an apple simply falling to the ground, to intricate collisions of molecules in a gas. How ever complex these phenomena might be, our understanding and beliefs of such systems can be formalized by setting up a model. These models are referred to as dynamical systems and by *learning* them, the rules governing their evolution over time are identified. This gives us a complete description of the system, which let us reason about its dynamics and predict how it will behave over time.

A common way to represent dynamical systems is by introducing a state transition and emission model, respectively describing how the system's internal state evolves and how observations are generated from that state. Among the traditional ways to specify a dynamical system's transition and emission model, the Kalman filter is a popular choice because of its simplicity and applicability. Furthermore, the extended Kalman filter is able to deal with nonlinearities in both the transition and emission model. However, specifying the transition and emission models a priori is not always possible, and for these tasks, working with Kalman filters is laborious and requires a significant amount of domain knowledge.

Deep generative models, including generative adversarial networks, variational autoencoders, normalizing flow models and autoregressive models, have in recent years seen remarkable advances. Whether the problem domain is images, text or speech, generative models have enabled a scalable way of modeling the data by combining progress in deep neural networks and stochastic optimization methods. Generative models, in contrast to discriminative models, can learn from unlabeled data which makes it particularly interesting since there are large amounts of unlabeled data compared to labeled one.

This thesis will investigate how deep generative models can be used to learn dynamical systems. Using generative models, we aim to learn directly from unlabeled observations of these dynamical systems without unnecessary modeling assumptions. While this thesis will work with simulated dynamical systems, the unsupervised learning approach translates well into dealing with the real-world counterpart where data labels are unavailable.

1.1 Problem Formulation

Dynamical systems are mathematical representations of physical phenomena that evolves over time, which are used in financial and weather forecasting, medical diagnosis, vehicle control and much more. Measurements of these systems are typically corrupted by noise due to various imperfections in the measurement process, and the form of this noise and at which level it's present are sources of uncertainty in the model. Another source of uncertainty is the structure of the model since there can be many unobservable variables present. Such variables are often referred to as hidden or latent variables and the introduction of them in probabilistic models provides us with the powerful tool to reason about dynamical systems.

1.1.1 State Space Models

The state space model (SSM) is a model that makes use of these latent variables to describe dynamical phenomena. The model is fully described by two stochastic processes, an unobserved state process, modeling transition from the previous state $\mathbf{x}_{t-1}$ to the current state $\mathbf{x}_t$, and an observed process, modeling how the current measurement $\mathbf{y}_t$ is generated by the current hidden state $\mathbf{x}_t$. These two processes are known as the transition and emission model, respectively, and can be represented using the following density functions

$$\begin{aligned}\mathbf{x}_t &\sim p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boldsymbol{\theta}, \mathbf{u}_t, V_t)\\ \mathbf{y}_t &\sim p(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta}, \mathbf{u}_t, W_t)\end{aligned}$$

Where $\mathbf{u}_t$ is possible control input and $\boldsymbol{\theta}$ is a vector of unknown parameters that the state space model depends on. The process and measurement noise of the system are represented by V_t and W_t, respectively. The core idea is that the true state is not observed, hence the *latent* state. We can only *infer* it from the observations and thus a natural question arises: what is the state given the observations?

1.1.2 Posterior Inference

The answer to our question is the posterior distribution $p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$ and can be expressed using the Bayes' rule

$$p(\mathbf{x}_{0:T}, \boldsymbol{\theta}|\mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T}|\mathbf{x}_{0:T})p(\mathbf{x}_{0:T}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:T})} \tag{1.1}$$

However, instead of calculating this distribution at all time steps, which is computationally very inefficient, we can instead consider the following three marginal distributions:

- *Filtering distributions* computed using the *Bayesian filter* are the marginal distributions of the current state $\mathbf{x}_t$ given the current and previous measurements $\mathbf{y}_{1:t}$.

$$p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}_{1:t}) \qquad\qquad t = 1, \ldots, T. \tag{1.2a}$$

- *Prediction distributions* computed using the *prediction step* of the *Bayesian filter* are the marginal distributions of the future state $\mathbf{x}_{t+n}$, n steps ahead of time.

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t+n}|\mathbf{y}_{1:t}) \qquad \begin{aligned} t &= 1, \ldots, T. \\ n &= 1, 2, \ldots. \end{aligned} \tag{1.2b}$$

- *Smoothing distributions* computed using the *Bayesian smoother* are the marginal distributions of the state $\mathbf{x}_t$ given a certain interval $\mathbf{y}_{1:T}$ of measurements with $T > t$.

$$p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}_{1:T}) \qquad t = 1, \ldots, T. \tag{1.2c}$$

These marginal distributions, though more efficient to compute than the full posterior, still don't have closed form solutions when the models contain nonlinear terms or are perturbed by non-Gaussian noise. This is the case for many real world processes, requiring us to instead learn approximate distributions of these posteriors.

The thesis will deal with the problem of identifying suitable approximations of such posteriors and in doing so *learns* the transition and emission model of the dynamical system's state space model. This learning process takes a parametric approach where we aim to find the best fitting parameters $\boldsymbol{\theta}$ of our generative model p_θ.

1.2 Objective

The overall objective of this thesis is to investigate how well deep generative models are suited for learning dynamical systems and this can be subdivided into a few concrete research questions:

- What constitutes a suitable evaluation metric in the context of learning dynamical systems?
- Which are the current state-of-the-art approaches to learning dynamical systems?
- How do these different approaches and their choices/assumptions affect the learning of a system?

1.3 Limitations

Since this thesis aims to provide a thorough overview of different approaches as well as consider dynamical systems in general rather than trying to find the best solution for a specific problem, there is one natural limitation to make: use of simulation data instead of real-world measurements. Not only would this give us the ability to with ease investigate and evaluate a wide range of dynamical systems since the ground truth is available, but also would time be saved from not having to deal with cleaning and pre-processing of real-world data. This shifts the focus of the thesis such that theoretical and academic aspects of the problem will be highlighted, rather than immediate practical applications.

1.4 Thesis Outline

Chapter 2 introduces the families of deep generative models being considered in this thesis.
Chapter 3 describes the creation of the data sets as well as the chosen models to be evaluated using these data sets. The choice of these models are motivated with some background and related work.
Chapter 4 discusses evaluation of generative models in general, and in particular how our chosen models will be evaluated.
Chapter 5 presents the results for the different models trained using the different data sets.
Chapter 6 draws conclusions of the results and suggests some future work.

1.5 Contributions

Our main contribution is the comparison of several deep generative models in the context learning of dynamical systems. These models are from different types of families and are chosen to be broadly representable of the current state-of-the-art in those respective families. By evaluating these models with data sets of different properties, we are able to provide the reader with insight to better understand their strengths and weaknesses and what to consider when planning to use deep generative models in a similar setting.

2
Background

2.1 Variational Autoencoders

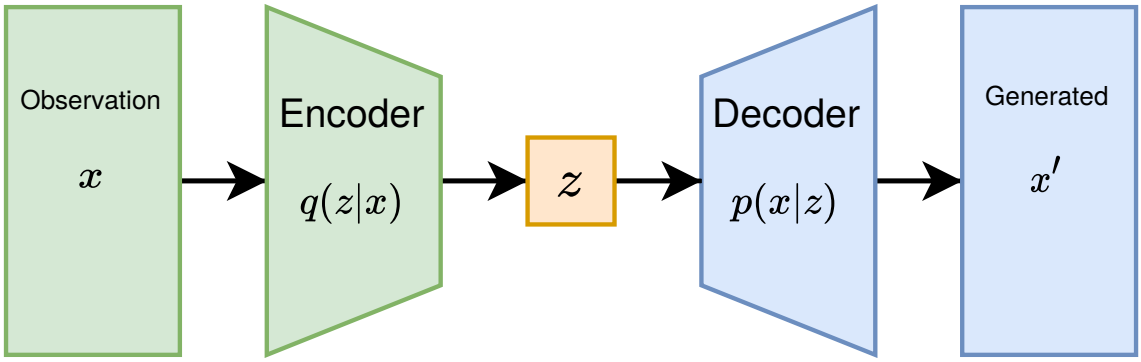

Figure 2.1: Illustration of a variational autoencoder

2.1.1 Encoder and Decoder

One of the main components in the variational autoencoder (VAE) is the *encoder* $q_\phi(\mathbf{z}|\mathbf{x})$, also referred to in the literature as the *inference model* or *recognition model*. The inference model can be parameterized by a neural network and in that case the *variational parameters* ϕ include the weights and biases of this network. As visualized by fig. 2.1, the encoder transforms the high-dimensional input $\mathbf{x}$ to a lower-dimensional output $\mathbf{z}$, often denoted the latent space. The other main component of the VAE is the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ which instead tries to reconstruct the observations $\mathbf{x}$ from the latent space $\mathbf{z}$.

2.1.2 Variational Inference

As the encoder formulates a parameterized distribution $q_\phi(\mathbf{z}|\mathbf{x})$, its associated *variational parameters* ϕ, should be picked in such a way that the distance from this approximate posterior to the true (but intractable) posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is as small as possible. This distance can be measured using the Kullback-Leibler divergence ($\mathbb{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})\right]$), and minimizing the distance by inferring the optimal variational parameters is the heart of variational inference. The learning objective associated with this training procedure is the so called *evidence lower bound* (ELBO),

also sometimes referred to as the *variational lower bound*. It's often derived using Jensen's equality or by the definition of KL divergence, but deriving it in the following alternative way highlights the tightness of the actual bound:

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]}_{\text{ELBO}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]}_{\mathbb{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})\right]}
\end{aligned} \tag{2.1}$$

Due to the non-negativity of KL divergence, the second term in the RHS of eq. (2.1), $\mathbb{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})\right]$, will be at least 0 and thus the ELBO is a lower bound of the evidence $\log p_\theta(\mathbf{x})$.

2.2 Autoregressive Models

Using the chain rule of probability, autoregressive models factorize the joint distribution as:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_1, x_2, ..., x_{i-1}) \tag{2.2}$$

Like VAEs, autoregressive models are explicit density estimators, but in addition they allow for exact inference which simplifies evaluation of this generative model. To sample from an autoregressive model we need to sample x_1, then $x_2|x_1$, and so on until we have sampled $x_n|x_1, x_2, ..., x_{n-1}$. Due to this sequential nature, sampling can be a quite computationally inefficient process, limiting its usefulness in real-time applications.

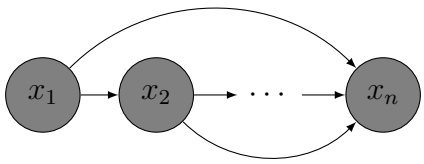

Figure 2.2: Graphical model for autoregressive models

2.3 Normalizing Flow Models

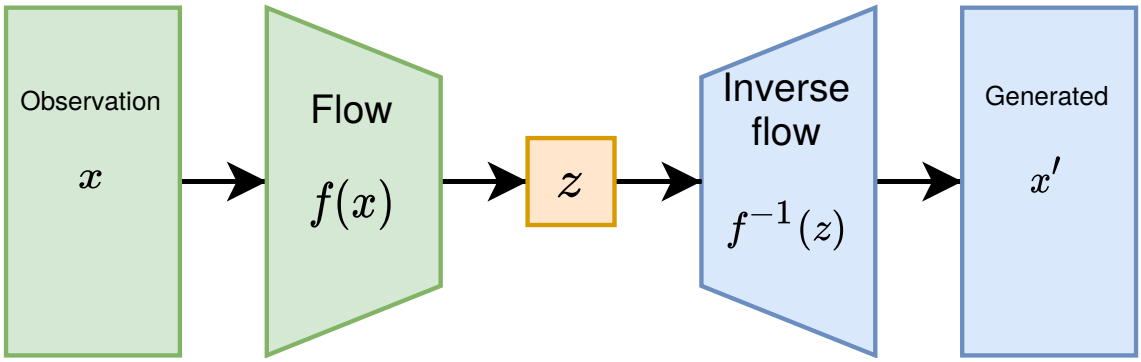

Figure 2.3: Illustration of a normalizing flow model

In the framework of normalizing flow models, we start with a simple distribution and then apply a sequence of functions, transforming this simple distribution into a more complex one. The functions applied to the original distribution are invertible and thus the density of the resulting complex distribution can be calculated by transforming back to the simple distribution and keeping track of the determinants of the Jacobians of each transformation, according the change of variables formula.

2.3.1 Change of Variables Formula

Let $\mathbf{z}$ be a random variable and p_Z its known probability density function. Let f be an invertible function and $f(\mathbf{z}) = \mathbf{x}$. Then using the change of variables formula, the density of the random variable $\mathbf{x}$ can be calculated as:

$$\begin{aligned} p_X(\mathbf{x}) &= p_Z(\mathbf{z}) \left| \det \frac{d\mathbf{z}}{d\mathbf{x}} \right| \\ &= p_Z(f^{-1}(\mathbf{x})) \left| \det \frac{df^{-1}}{d\mathbf{x}} \right| \end{aligned} \tag{2.3}$$

2.3.2 Learning

We can construct a complicated nonlinear invertible function f by composing multiple invertible functions f_i, using the fact that the composition of invertible functions is itself invertible. By applying such a sequence of K composed functions to our simple base distribution p_Z, we aim to arrive at a complex distribution $p_X(\mathbf{x})$ that better resembles the input data.

$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0) \tag{2.4}$$

We then arrive at the log-likelihood as:

$$\log p_{\mathrm{X}}(\mathbf{x}) = \log p_{\mathrm{Z}_{\boldsymbol{K}}}(\mathbf{z}_K) \tag{2.5}$$

$$= \log p_{\mathrm{Z}_{\mathbf{0}}}(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right| \tag{2.6}$$

In order to allow efficient computation, the function f_i needs to be both easily invertible and have a Jacobian determinant that is easy to compute.

2.4 GAN

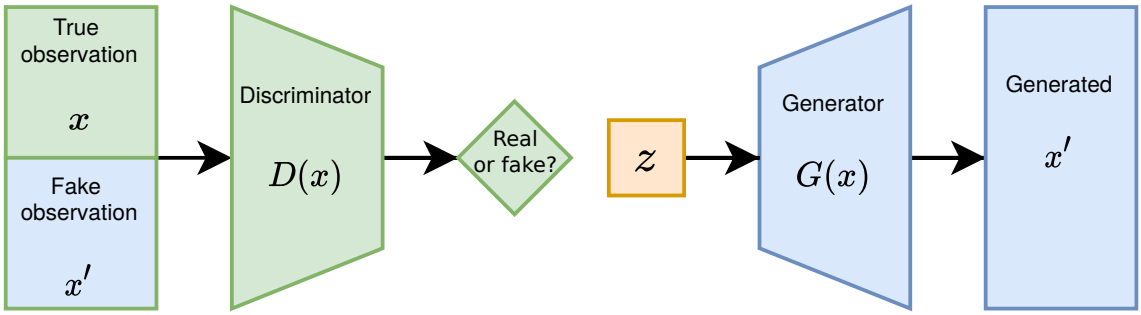

Figure 2.4: Illustration of a generative adversarial network

Another popular family of generative models are the generative adversarial networks (GANs). They differ from the models we have discussed so far, such as the VAEs, autoregressive models and normalizing flow models, since they are not trained using maximum likelihood.

A GAN consists of two main components, a discriminator D and a generator G. The discriminator is optimized to distinguish between real and fake data. The fake data comes from the generator and this component is in contrast optimized to make these generative samples as seemingly real as possible. The generator aims to minimize the chance of generated samples being labeled as fake by the discriminator, that is to minimize $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$. Simultaneously, the discriminator aims to predict real data with high probability, that is to maximize $\mathbb{E}_{x\sim p_{\text{real data}}}[\log D(x)]$. Together, G and D are playing a minmax game described as:

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x\sim p_{\text{real data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.7}$$

3
Method

3.1 Data sets

This section introduces the dynamical systems which the selected models in section 3.2 will be learning. These systems are chosen in regard to their dynamical properties; in particular we investigate nonlinear and linear dynamical systems.

3.1.1 Pendulum

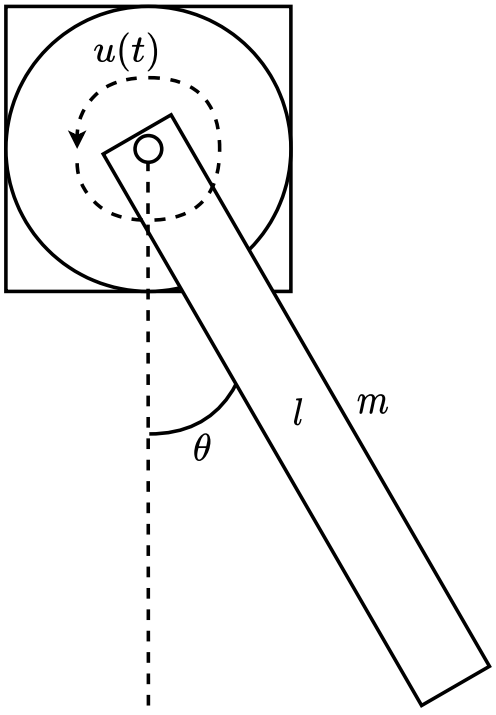

Figure 3.1: Illustration of the pendulum system

As an example of a nonlinear dynamical system we consider a pendulum system. The system consists of a bar of length l and mass m, (with no extra mass attached to its tip). The pendulum is actuated by a control input u each time step through applying a torque to the pendulum. The system is illustrated in fig. 3.1 and its nonlinear model is given by

$$\frac{m}{3}l^2\ddot{\theta} + \frac{m}{2}gl\sin(\theta) + u = 0 \tag{3.1}$$

Rearranging yields

$$\ddot{\theta} = -(\frac{3g}{2l}\sin(\theta) + \frac{3u}{ml^2}) \tag{3.2}$$

3.1.1.1 Ground Truth Generation

Using the Forward Euler method and adding transition noise w, we arrive at the following computational scheme:

$$\dot{\theta}^{n+1} = \dot{\theta}^n - (\frac{3g}{2l}\sin(\theta) + \frac{3u}{ml^2})\Delta t + w^0 \qquad w^0 \sim \mathcal{N}(0, Q_1) \tag{3.3}$$

$$\theta^{n+1} = \theta^n + \dot{\theta}^{n+1}\Delta t + w^1 \qquad w^1 \sim \mathcal{N}(0, Q_2) \tag{3.4}$$

An illustration of this computational scheme is found in fig. 3.2.

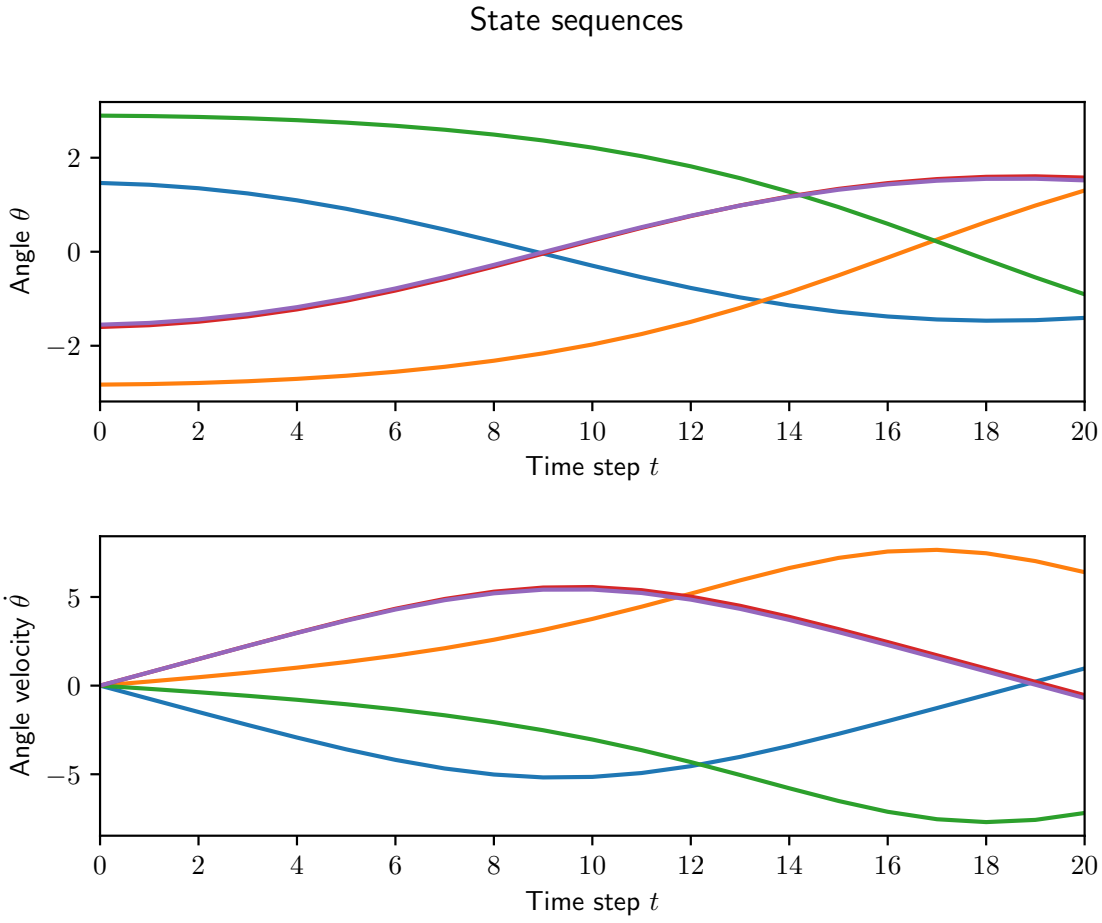

Figure 3.2: Example of generated pendulum state sequences consisting of ground truth values for angle and angle velocity at each time step.

3.1.1.2 Observation Generation

We extract the observations as

$$\{\cos(\theta) + v, \sin(\theta) + v\} \qquad v \sim \mathcal{N}(0, R) \tag{3.5}$$

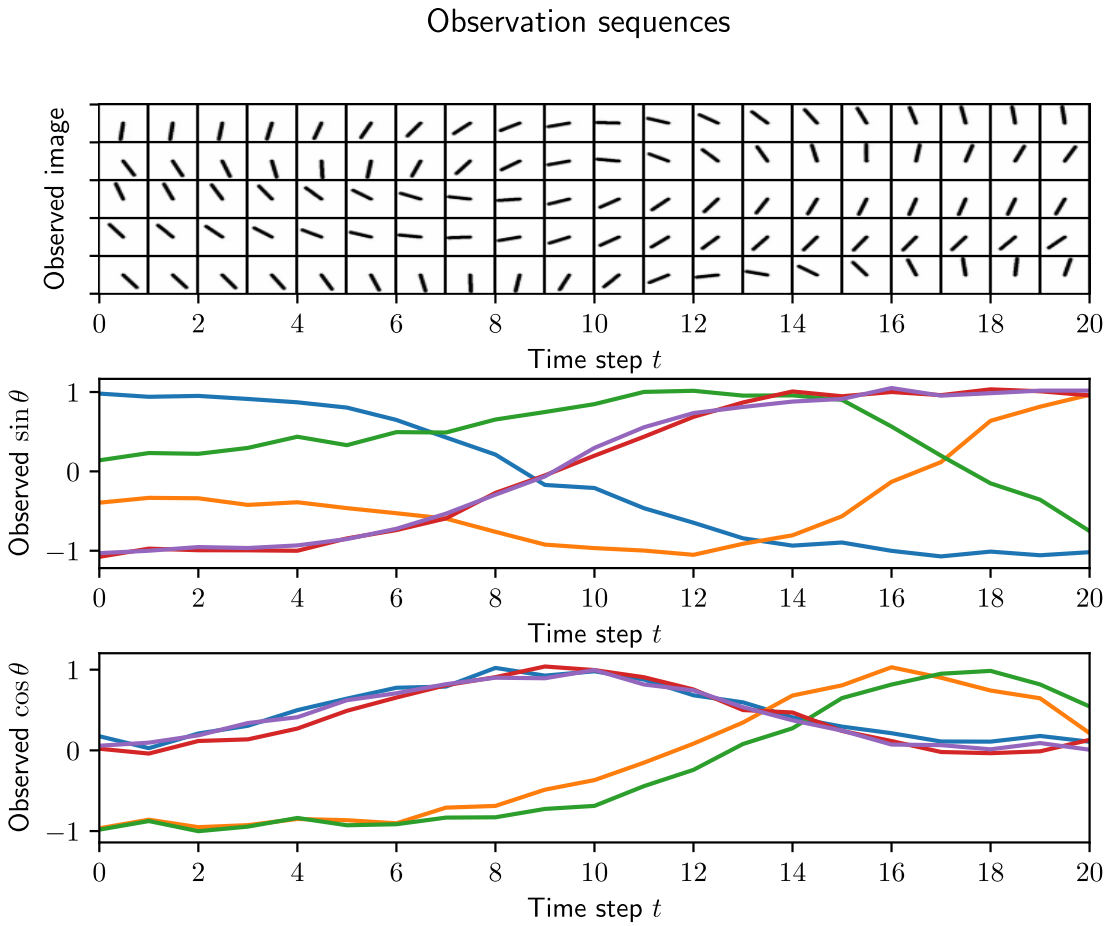

Figure 3.3: Example of generated pendulum observation sequences consisting of both rendered image sequences and lower dimensional states perturbed by noise.

Training is done directly on the sequences of these observations $\in R^2$ or on sequences of observations $\in R^{32\times32}$, obtained by generating images based on the ground truth states using rendering functionality of OpenAI-Gym[3]. The two types of observation sequences are visualized in fig. 3.3.

3.1.2 Spring-Mass System

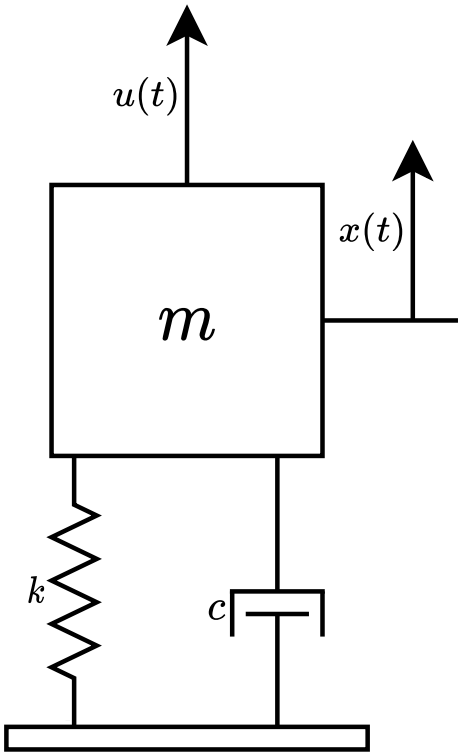

Figure 3.4: Illustration of Spring-mass system

As an example of a linear dynamical system we consider a damped spring-mass system. The system consists of a mass m attached to a linear spring with constant k and a damper with constant c. The displacement of the mass varies in a linear manner over time.

The system is illustrated in fig. 3.4 and its differential equation for the damped spring-mass system is given by

$$m\ddot{x} + c\dot{x} + kx - u = 0 \tag{3.6}$$

Rearranging yields

$$\ddot{x} = \frac{-c\dot{x} - kx + u}{m} \tag{3.7}$$

3.1.2.1 Ground Truth Generation

Using the Forward Euler method and adding transition noise w, we arrive at the following computational scheme:

$$\dot{x}^{n+1} = \dot{x}^n + \frac{-c\dot{x}^n - kx + u}{m}\Delta t + w^0 \qquad w^0 \sim \mathcal{N}(0, Q_1) \tag{3.8}$$

$$x^{n+1} = x^n + \dot{x}^{n+1}\Delta t + w^1 \qquad w^1 \sim \mathcal{N}(0, Q_2) \tag{3.9}$$

The generated ground truth sequences are illustrated in fig. 3.5.

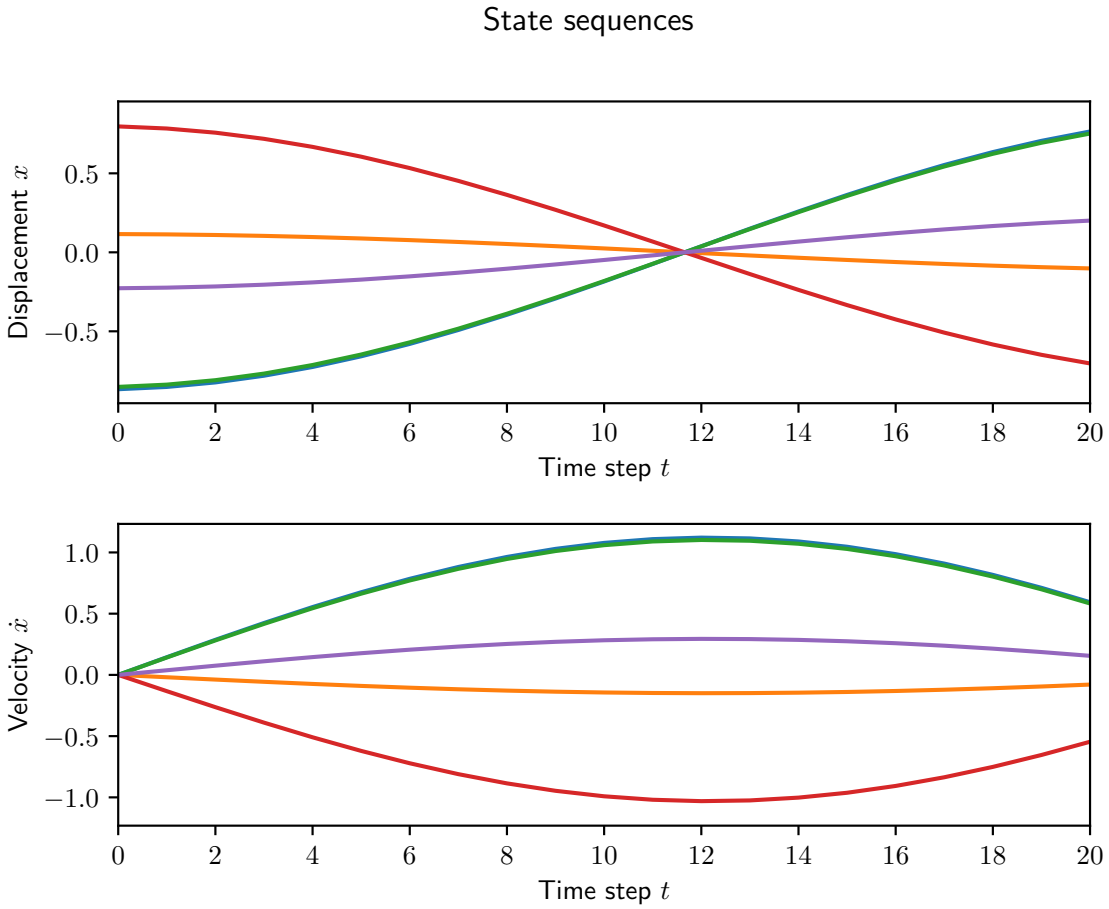

Figure 3.5: Example of generated spring-mass state sequences consisting of ground truth values for displacement and velocity at each time step.

3.1.2.2 Observation Generation

We extract the observations either as

$$\{x + v\} \qquad\qquad v \sim \mathcal{N}(0, R), \tag{3.10}$$

or with the help of OpenAI-Gym as in the pendulum case. The observation sequences for the spring-mass system are illustrated in fig. 3.6.

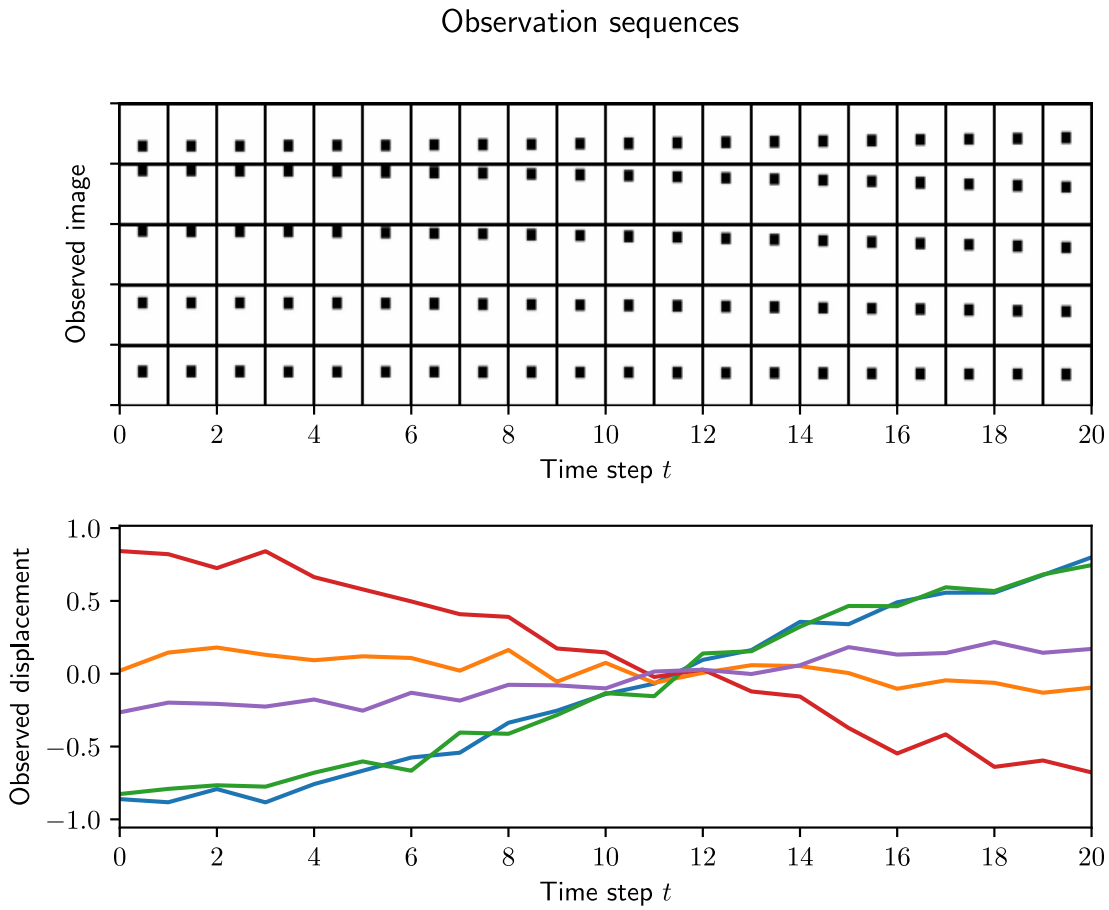

Figure 3.6: Example of generated spring-mass observation sequences consisting of both rendered image sequences and lower dimensional states perturbed by noise.

3.2 Models

In order to evaluate the dynamical systems described in section 3.1 we have chosen some models broadly representative of the current state of the art in deep generative modeling. These models belong to different families of generative models and include elements of variational autoencoders, generative adversarial networks, and normalizing flow models with and without autoregressive properties.

Every model uses some sort of encoder-decoder structure, thus being able to transform our high-dimensional observations into a lower dimensional latent representation, and then back again to observation space. The decoding of our latent space back to the high-dimensional observation space is a process referred to as reconstruction. Besides reconstruction, the models are also capable of generating future frames, based on observing some initial frames, i.e. performing prediction.

3.2.1 VAE with Locally Linear Transitions (VAELLT)

Using state space models in the context of variational autoencoders is an approach originally proposed by [30] and subsequently used by [14] and [9] with promising results. The idea is to introduce matrices to describe the transition and emission model using linear transformations.

In this model we further investigate *Deep Variational Bayes Filters* [14]. They provide a standard variational autoencoder architecture but put special emphasis on the transition dynamics in the latent space.

3.2.1.1 Recognition Model

A key element in this model is to let the transition model become the driving force for shaping the latent space. This is achieved by preventing the recognition model from directly drawing the latent state $\mathbf{z}_t$, and instead let it infer an intermediate variable $\mathbf{w}_t$ later used to describe the transition dynamics of the latent space.

$$q(\mathbf{w}_t|\mathbf{z}_t, \mathbf{x}_{t+1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{w}_t; \mu, \sigma)$$

Their recognition model is realized as a neural network, taking $\mathbf{z}_t$, $\mathbf{x}_{t+1}$ and $\mathbf{u}_t$ as input and outputs the distribution parameters μ and σ of $\mathbf{w}_t$. Here $\mathbf{u}_t$ is a possible control input. However, in this thesis, we consider the case without any control inputs, i.e. $\mathbf{u}_t = \mathbf{0}$.

3.2.1.2 Transition Model

We introduce matrices $\mathbf{A}_t, \mathbf{B}_t$, and $\mathbf{C}_t$ to express the transition in the latent space as:

$$\mathbf{z}_{t+1} = \mathbf{A}_t\mathbf{z}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{C}_t\mathbf{w}_t \qquad t = 1, \ldots, T.$$

where $\mathbf{w}_t$ is sampled from the recognition model, or from the prior in absence of input data.

The parameters of these matrices are estimated with the help of a third neural network $\boldsymbol{\alpha_t} = f_\psi(\mathbf{z}_t, \mathbf{u}_t)$ and a set of tunable base matrices $\{\mathbf{A}^{(i)}, \mathbf{B}^{(i)}, \mathbf{C}^{(i)}\}$:

$$\mathbf{A}_t = \sum_{i=1}^{M} \alpha_t^{(i)} \mathbf{A}^{(i)}, \quad \mathbf{B}_t = \sum_{i=1}^{M} \alpha_t^{(i)} \mathbf{B}^{(i)}, \quad \mathbf{C}_t = \sum_{i=1}^{M} \alpha_t^{(i)} \mathbf{C}^{(i)}.$$

where the weight vector $\boldsymbol{\alpha_t}$ is shared between the matrices.

3.2.1.3 Emission Model

The emission model constructs observations $\mathbf{x}_t$ from latent state variables $\mathbf{z}_t$ and is implemented using a neural network.

$$p(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}\left(\mathbf{x}_t; \mu(\mathbf{z}_t), \sigma\right)$$

3.2.1.4 Training and Testing

The training task consists of learning the parameters of the matrices in the transition model, as well as the weights of the neural network in the emission model. This training is done using variational inference and the ELBO is defined as:

$$\mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})\right] - \mathbb{KL}\left[\log q_\phi(\mathbf{w}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})||p(\mathbf{w}_{1:T})\right]$$

3.2.1.4.1 Prediction. When performing prediction, i.e. sampling without observation input, $\mathbf{w}_{1:T}$ is sampled from the prior $p(\mathbf{w}_{1:T})$ instead of the recognition model q_ϕ, whereas the matrices of the transition model are readily available in this mode. The prior employed is an isotropic Gaussian, i.e. $p(\mathbf{w}_{1:T}) \sim \mathcal{N}\left(\mu, \sigma^2 I\right)$.

3.2.2 VAELLT and Planar Normalizing Flow (VAELLT-PNF)

A key challenge with variational autoencoders is choosing an approximate posterior distribution that is simple and tractable, yet sufficiently expressive to resemble the true posterior distribution. The derivation of the ELBO in eq. (2.1) tells us that the bound is tight when $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$, that is, when our approximate posterior distribution matches the true. That these two matches is unlikely to happen in practice since most existing work employ simple approximate posteriors, such as Gaussian or Category distributions.

A way to address the issue was proposed in *Variational Inference with Normalizing Flows* [26] which with the help of normalizing flows aims to achieve a more flexible approximate posterior distribution. One of the normalizing flows proposed in this paper was the planar flow, which in this model will be used in conjunction with the VAELLT model described section 3.2.1.

3.2.2.1 Invertible Linear-time Transformations

The planar flows belong to a family of transformations of the form

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\top \mathbf{z} + b) \tag{3.11}$$

For these types of transformations, the logdet-Jacobian term can be calculated quickly:

$$\psi(\mathbf{z}) = h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{w}$$
$$\left|\det \frac{\partial f}{\partial \mathbf{z}}\right| = \left|\det(\mathrm{I} + \mathbf{u}\psi(\mathbf{z})^\top)\right| = \left|1 + \mathbf{u}^\top \psi(\mathbf{z})\right|$$

From eq. (2.5) we conclude that the density $q_K(\mathbf{z})$ obtained by successively transforming a random variable $\mathbf{z}_0$ with distribution q_0 through K transformations f_k is:

$$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}) - \sum_{k=1}^{K} \left|1 + \mathbf{u}_k^\top \psi(\mathbf{z}_{k-1})\right|$$
$$\mathbf{z}_K = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z}_0)$$

3.2.2.2 Recognition Model

If we now parameterize the approximate posterior distribution with a flow length K, $q_\phi(\mathbf{z}|\mathbf{x}) := q_K(\mathbf{z}_k)$, the evidence lower bound of eq. (2.1) can be written as an expectation over the initial distribution $q_0(\mathbf{z})$:

$$\begin{aligned}
\mathrm{ELBO} &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})\right] \\
&= \mathbb{E}_{q_0(z_0)}\left[\log q_K(\mathbf{z}_K) - \log p_\theta(\mathbf{x}, \mathbf{z}_K)\right] \\
&= \mathbb{E}_{q_0(z_0)}\left[\log q_K(\mathbf{z}_K) - \log p_\theta(\mathbf{x}|\mathbf{z}_K) - \log p_\theta(\mathbf{z}_K)\right] \\
&= \mathbb{E}_{q_0(z_0)}\left[\log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \left|1 + \mathbf{u}_k^\top \psi(\mathbf{z}_{k-1})\right| - \log p_\theta(\mathbf{x}|\mathbf{z}_K) - \log p_\theta(\mathbf{z}_K)\right]
\end{aligned} \tag{3.12}$$

We make use of the same principles as in the VAELLT model in section 3.2.1 with the difference that at each time step we apply the normalizing flow to $\mathbf{w}_t \sim q_\phi$, the output of the recognition model, and modify the evidence lower bound to the one derived in eq. (3.12).

3.2.2.3 Transition and Emission Model

We make use of the same transition and emission model as in VAE-LLT model described in section 3.2.1.

3.2.3 VAE and Autoregressive Normalizing Flow (VAE-ARNF)

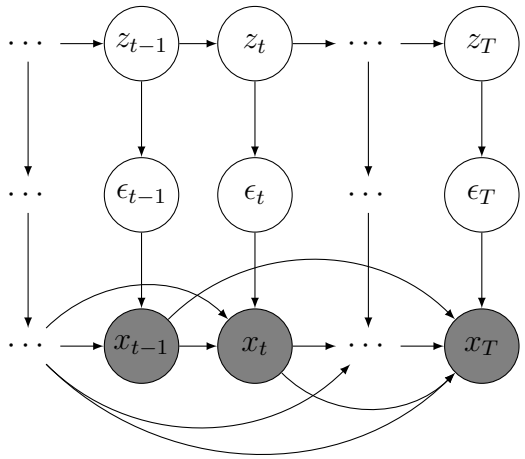

Figure 3.7: Graphical model for the autoregressive normalizing flow model

This model uses the implementation from *Improving sequential latent variable models with autoregressive flows* [19]. Their approach consists of a standard convolutional encoder-decoder architecture, but normalizing flows are incorporated to improve dynamics modeling. The flows considered are called *affine autoregressive flows* and are applied across time steps within the sequences.

3.2.3.1 Affine Autoregressive Flows

While [19] makes use of these flows in a new context, the affine autoregressive normalizing flows and their properties have been highlighted in the literature previously [15, 21]. As shown by [15], sampling from an autoregressive Gaussian model is a transformation constituting a normalizing flow. Using the reparameterization trick of [16, 27], we see that the sampling procedure of a Gaussian variable $\mathbf{x}_t \sim p_\theta(\mathbf{x}_t|\mathbf{x}_{1:t-1})$ is a transformation from a noise vector $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ to a corresponding vector $\mathbf{x}_t$:

$$\mathbf{x}_t = \mu_t(\mathbf{x}_{1:t-1}) + \sigma_t(\mathbf{x}_{1:t-1}) \odot \boldsymbol{\epsilon}_t \tag{3.13}$$

As long as $\sigma_t > 0$, this transformation is a bijection and can be inverted:

$$\boldsymbol{\epsilon}_t = \frac{\mathbf{x}_t - \mu_t(\mathbf{x}_{1:t-1})}{\sigma_t(\mathbf{x}_{1:t-1})} \tag{3.14}$$

Using the change of variables formula described in section 2.3.1, we express log-likelihood of the model as:

$$\log p_\theta(\mathbf{x}_{1:T}) = \log p_\theta(\boldsymbol{\epsilon}_{1:T}) - \log |\det \frac{\partial \mathbf{x}_{1:T}}{\partial \boldsymbol{\epsilon}_{1:T}}| \tag{3.15}$$

The Jacobian of the autoregressive transformation is triangular and thus the determinant is simply the product of the diagonal terms:

$$\log|\det\frac{\partial \mathbf{x}_{1:T}}{\partial \boldsymbol{\epsilon}_{1:T}}| = \sum_{t=1}^{T}\sum_{i}\log\sigma_t(\mathbf{x}_{1:t-1}) \tag{3.16}$$

3.2.3.2 Recognition Model

The recognition model is a convolutional encoder based on the DC-GAN structure [23]. The encoded data is then sent to a LSTM-layer [11] followed by fully connected layers to output the mean and log-variance of the approximate posterior distribution.

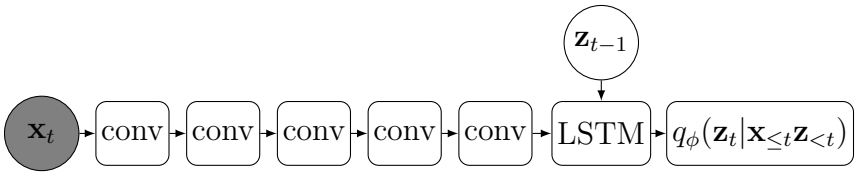

Figure 3.8: Graphical model for the recognition model.

3.2.3.3 Transition Model

In this model, the temporal dependencies in the observation sequence are learned using recurrent LSTM layers. These LSTM layers are found both in the approximate posterior and in the prior, which are both learned during training.

3.2.3.4 Emission Model

The emission model is a convolutional decoder and has the inverse architectural structure of the recognition model, as visualized by fig. 3.9. To generate frames, $\mathbf{z}_t$ is sampled from the approximate posterior distribution (or prior if predicting) and passed as input to a sequence of 4 transposed convolutional layers, which outputs the shift and scale parameters $\mu_\theta(\mathbf{x}_{<t})$ and $\sigma_\theta(\mathbf{x}_{<t})$. These parameters can then be used to synthesize frames using eq. (3.13).

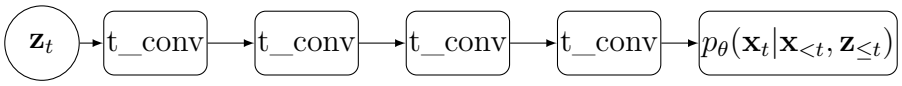

Figure 3.9: Graphical model for the emission model.

3.2.3.5 Training and Testing

The model is trained using variational inference where the evidence lower bound is defined as:

$$\mathcal{L} = \sum_{t=1}^{T}\mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\left[\log p_\theta(\boldsymbol{\epsilon}_t|\boldsymbol{\epsilon}_{<t},\mathbf{z}_{<t}) - \log\frac{q(\mathbf{z}_t|\mathbf{x}_{<t},\mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{x}_{<t},\mathbf{z}_{<t})} - \log\left|\det\frac{\partial \mathbf{x}_t}{\partial \boldsymbol{\epsilon}_t}\right|\right]$$

where $\boldsymbol{\epsilon}_t$ is calculated using the inverse transform in eq. (3.14) and $\log \left|\det \frac{\partial \mathbf{x}_t}{\partial \boldsymbol{\epsilon}_t}\right|$ is calculated according to eq. (3.16).

3.2.3.5.1 Prediction. In absence of input data, the latent prior is used instead of the approximate posterior distribution. Since the prior consists of a LSTM network, where the transition dynamics has been learned, consistency between training and testing performance is ensured. At each time step, we feed the latent variable of the previous time step, $\mathbf{z}_{t-1}$, to the prior to generate $\mathbf{z}_t$. The latent variable $\mathbf{z}_t$ is in turn given as input to the convolutional decoder network to generate the predicted $\hat{\mathbf{x}}_t$.

3.2.4 VAE-GAN

Stochastic Adversarial Video Prediction [18] introduces a method for combining variational autoencoders and generative adversarial networks for prediction of videos. While their encoder is a standard convolutional network, the GAN-architecture mixes ideas from various prior work. The generative network of their model is greatly influenced by the *convolutional dynamic neural advection* (CDNA) introduced in [8], and the discriminator is a feed-forward convolutional network based on SNGAN [20].

3.2.4.1 Recognition Model

The recognition model $q_\phi(\mathbf{z}_t|\mathbf{x}_{t:t+1})$ consists of a feed-forward convolutional network that at every time step encodes image pairs $\mathbf{x}_{t:t+1}$, to a latent variable $\mathbf{z}_t$. This deep encoder is parameterized as a conditionally Gaussian distribution $\mathcal{N}\left(\mu_{\mathbf{z}_t}, \sigma^2_{\mathbf{z}_t}\right)$.

3.2.4.2 Transition Model

Temporal correlations of the latent variables are learned with the help of recurrent LSTM layers. This LSTM network is architecturally a part of the generator network, and thus the transition modeling remains consistent whether the latent variables are sampled from the recognition model $q_\phi(\mathbf{z}_t|\mathbf{x}_{t:t+1})$ or the prior $p(\mathbf{z}_t)$, which is a fixed unit Gaussian $\mathcal{N}(0, 1)$.

3.2.4.3 Emission Model

The emission model consists of a convolutional decoder that takes as input the latent space vector and a previous observation image of the sequence, and outputs the convolutional kernels used to transform the input frame to the predicted consequent one. This generator specifies a distribution $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{0:t-1})$, and parameterized by a fixed-variance Laplacian distribution.

3.2.4.4 Training and Testing

The learning objective is derived as a combination of losses based on the four main components: the encoder E, the generator G, and the two discriminators D and D^{VAE}, which are respectively used depending on if the sequence to be discriminated was generated using the prior $p(\mathbf{z}_t)$ or the approximate posterior distribution $q_\phi(\mathbf{z}_t|\mathbf{x}_{t:t+1})$. These losses are then minimized according to a min-max game:

$$G^*, E^* = \arg\min_{G,E} \max_{D,D^{\text{VAE}}} \mathcal{L}_{\text{VAE}}(G,E) + \mathcal{L}_{\text{GAN}}(G,D) + \mathcal{L}^{\text{VAE}}_{\text{GAN}}(G,E,D^{\text{VAE}})$$

where

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{\mathbf{x}_{0:T},\mathbf{z}_t\sim E(\mathbf{x}_{t:t+1})|_{t=0}^{T=1}}\left[\sum_{t=1}^{T}\|\mathbf{x}_t - G(\mathbf{x}_0,\mathbf{z}_{0:t-1})\|_1\right] + \mathbb{E}_{\mathbf{x}_{0:T}}\left[\sum_{t=1}^{T}\mathbb{KL}\left[E(\mathbf{x}_{t-1:t})||p(\mathbf{z}_{t-1})\right]\right]$$

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{\mathbf{x}_{1:T}}\left[\log D(\mathbf{x}_{0:T-1})\right] + \mathbb{E}_{\mathbf{x}_{1:T},\mathbf{z}_t\sim p(\mathbf{z}_t)|_{t=0}^{T-1}}\left[\log(1 - D(G(\mathbf{x}_0,\mathbf{z}_{0:T-1})))\right]$$

$$\mathcal{L}^{\text{VAE}}_{\text{GAN}} = \mathbb{E}_{\mathbf{x}_{1:T}}\left[\log D^{\text{VAE}}(\mathbf{x}_{0:T-1})\right] + \mathbb{E}_{\mathbf{x}_{1:T},\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x}_{t:t+1})|_{t=0}^{T-1}}\left[\log(1 - D^{\text{VAE}}(G(\mathbf{x}_0,\mathbf{z}_{0:T-1})))\right]$$

3.2.4.4.1 Prediction. During testing, a sequence of frames is predicted by repeated one-step-ahead predictions by feeding predicted frames back to the generator together with latent variables sampled from the prior. For every such prediction, the generator is free to choose if it should use the transformed version of the previous frame (using the predicted convolutional kernels), copy pixels from the previous frame, or construct the image from scratch.

4
Evaluation

The evaluation of generative models is a challenging task due to the different learning objectives involved. Even in the case of this thesis – where the training of every model is based on variational inference and its associated ELBO learning objective – direct comparison of the learning objective loss is still problematic due to the various additional elements the models include (such as Normalizing Flow and GAN components). While standard quantitative metrics such as PSNR and SSIM [29] do exist, it has been noted [22] that these metrics are not always in correspondence with qualitative scores given by humans. More importantly, such metrics are measurements of individual frame quality, and do not try to quantify the quality of transition from frame to frame, i.e. how well the dynamics have been learned. One way to reason about the learned dynamics is by looking at the model's generative ability and the learned representation, both of which we will use to evaluate our models.

4.1 Generative Ability

The generative ability of our models are visually examined through their ability to reconstruct and predict. The reconstruction step is a single time step operation and thus requires no understanding of the temporal dependencies of the sequence, but can be regarded as a prerequisite for prediction and is therefore included.

4.1.1 Reconstruction

Reconstruction is the process of reconstructing a given input image but exactly how this done varies for different deep generative models. For instance, in the case of variational autoencoders, the input image is mapped to the distribution parameters of the latent space using the encoder network, and then reconstructed using the decoder network. For the normalizing flow models, the input image is mapped to parameters of the transformations used to transform the simple distribution to the more complex one that aims to resemble the input data. For GANs, the input image is never given directly to the generative part of the model, the generator, and thus we don't have access to any direct reconstructions. However, in our case, where all models include an encoder-decoder structure, the reconstruction process is straight forward and easily compared between the models.

4.1.2 Prediction

In order to perform prediction we need a number of frames that the models observe to understand the context to base the predictions on. Borrowing the notation of [1], we refer to those frames as *context frames*. In other words, predictions are conditioned on a set of c context frames $\mathbf{x}_0, \ldots, \mathbf{x}_{c-1}$ with the goal to generate new frames $\mathbf{x}_{c:T}$ through sampling from $p(\mathbf{x}_{c:T}|\mathbf{x}_{0:c-1})$. For our models, an additional latent variable $\mathbf{z}_t$ is sampled from a prior and is used at this step, hence $p(\mathbf{x}_{c:T}|\mathbf{x}_{0:c-1}, \mathbf{z}_t)$.

4.2 Learned Representations

It's natural to wonder what the learned latent variables $\mathbf{z}_t$ in our models represent. Is it possible to give an interpretation of this latent space, something that we can relate to the actual ground truth state? For instance, in the context of pendulum dynamics, do the respective learned latent variables z_i represent the angle θ and angular velocity $\dot{\theta}$ of the dynamical system? If it does, such a representation could be described as *disentangled*. While there is no exact definition of disentangled representations we use the one provided by [2]: "a representation where a change in one dimension corresponds to a change in one factor of variation, while being relatively invariant to changes in other factors". The variables of our ground truth state are here referred to as *factors of variation* because they determine the varieties in the observed data.

4.2.1 Quantifying Disentangled Representations

Using the notation common in the field of representation learning, we now denote the learned latent space as the *code*, and its variables c_i, whereas the generative factors (the ground truth variables) are referred to as z_i. We wish that our learned latent space represents the generative factors and a natural, ideal such representation would be a one-to-one mapping between the codes c_i and the generative factors z_i. If the learned code is of higher dimension than that of the generative factors, it is expected that these extra code variables are irrelevant and not predictive about the generative factors. Thus, with the help of regressors we can quantify how well the learned representation is predictive of the factors that gave rise to the observations used for learning that representation.

Such a quantification of learned representations is put into a framework in *A framework for the quantitative evaluation of disentangled representations* [7] through the following steps:

1. Train model M on a synthetic data set with generative factors $\mathbf{z}$
2. Retrieve code $\boldsymbol{c}$ for each sample $\boldsymbol{x}$ in the data set ($\boldsymbol{c} = M(\boldsymbol{x})$)
3. Train regressor f to predict $\boldsymbol{z}$ given $\boldsymbol{c}$ ($\hat{\boldsymbol{z}} = f(\boldsymbol{c})$)
4. Quantify f's deviation from the ideal mapping and the prediction error.

The framework explicitly defines and quantify three criteria of disentangled representations that are implicit in prior works [5, 2, 17, 4, 10], namely *disentanglement*, *completeness* and *informativeness*.

4.2.1.1 Disentanglement

Disentanglement measures the degree to which a representation factorizes or *disentangles* the underlying factors of variation, with each code variable c_i capturing at most one generative factor z_i.

$$
\begin{aligned}
D_i &= (1 - H_K(P_i)), && \text{Disentanglement score} \\
H_K(P_i) &= -\sum_{k=0}^{K-1} P_{ik} \log_K P_{ik}, && \text{Entropy of } P_i \\
P_{ij} &= R_{ij} / \sum_{k=0}^{K-1} R_{ik}, && \text{Probability of } c_i \text{ being important for predicting } z_j
\end{aligned}
$$

If a code variable c_i is important for predicting a single generative factor, the score will be 1. If a code variable c_i is equally important for predicting all generative factors, the score will be 0.

4.2.1.2 Completeness

In contrast, completeness measures the degree to which each underlying factor z_i is captured by a single code variable c_i.

$$C_j = (1 - H_D(\tilde{P}_{\cdot j})) \qquad \text{Completeness score} \tag{4.1}$$

$$H_D(\tilde{P}_{\cdot j}) = -\sum_{d=0}^{D-1} \tilde{P}_{dj} \log_D(\tilde{P}_{dj}) \qquad \text{Entropy of } \tilde{P}_{\cdot j} \tag{4.2}$$

If just a single code variable c_i contributes to the prediction of z_j, the score will be 1. If all code variables equally contribute to the prediction of z_j, the score will be 0.

4.2.1.3 Informativeness

Informativeness measures the amount of information that a representation captures about the underlying factors of variation. The informativeness of a code variable c_i about a generative factor z_j is given by the prediction error E of the regression.

$$
\begin{aligned}
E(z_j, \hat{z}_j), & && \text{Informativeness score} \\
\hat{z}_j = f_j(\boldsymbol{c}), & && \text{Predicted generative factor by regressor } f_j
\end{aligned}
$$

5

Experimental Results

In this chapter, the models described in section 3.2 are evaluated using the data sets described in section 3.1. Reconstruction and prediction ability as well as the learned representations are evaluated both quantitatively and qualitatively using the metrics described in chapter 4.

5.1 Reconstruction Ability

Reconstruction is the process of reconstructing an input image given the latent space encoded from that image. This ability is compared in fig. 5.1 and we can see that the models are able to reconstruct the images well for both the pendulum and spring-mass data set.

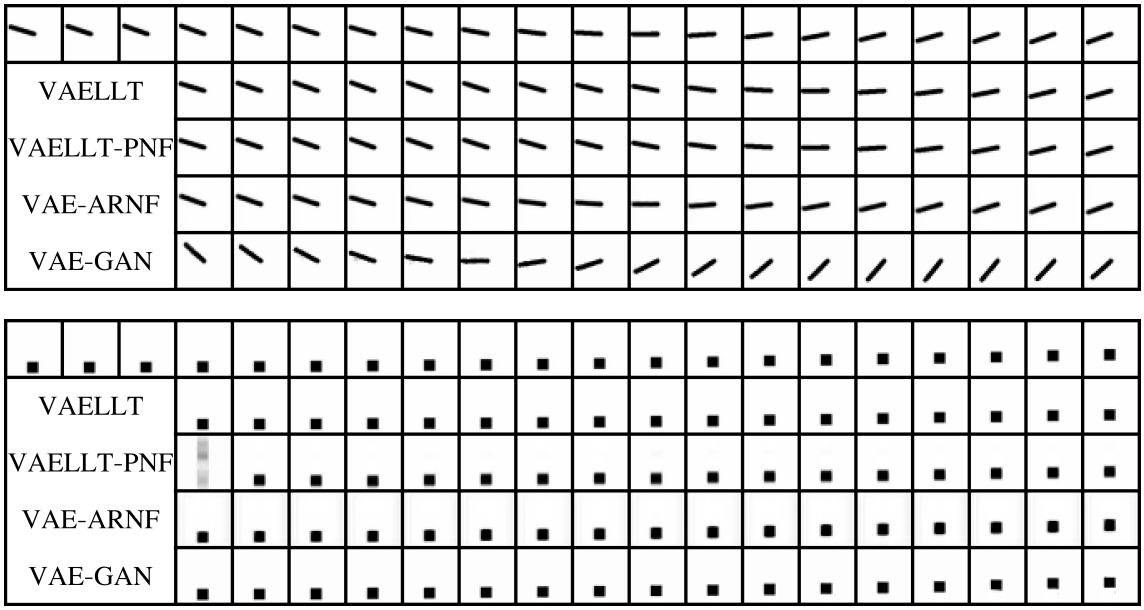

Figure 5.1: Reconstruction ability. Each respective top row is a test sequence of the data sets and the rows below it display a model's ability to reconstruct this sequence.

5.2 Prediction Ability

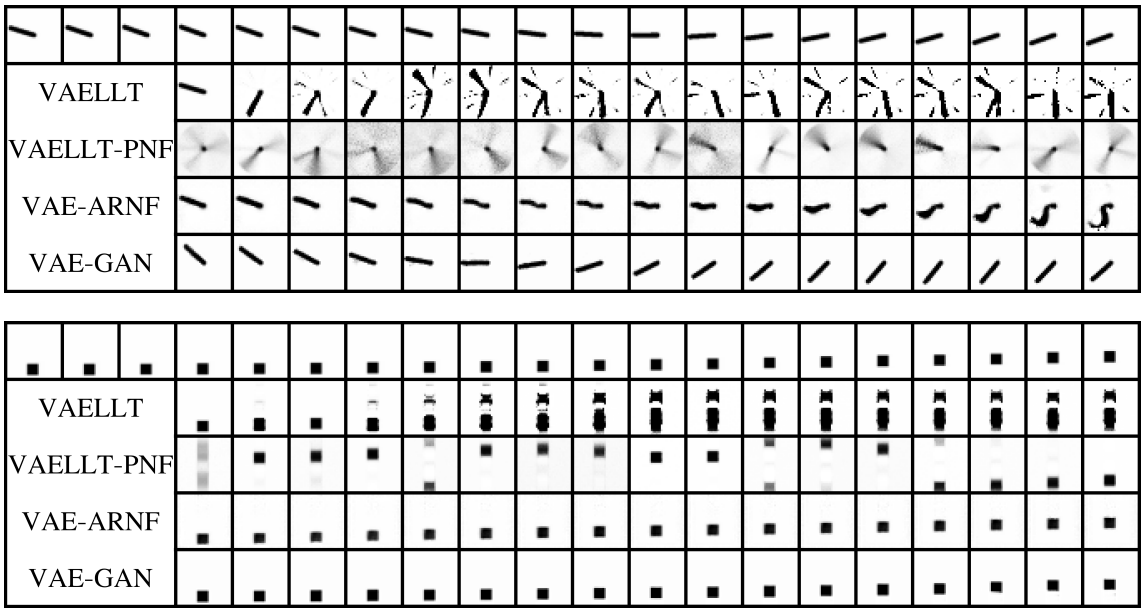

Figure 5.2: Prediction ability. The first three frames of the top row is the context frames and the subsequent are the ground truth frames which they try to predict based on only observing these context frames.

Figure 5.2 visualizes the prediction ability using 3 context frames (described in section 4.1.2). We can observe that only VAE-GAN is able to predict a reasonable position of the pendulum based on the initial observed context frames. In the spring-mass case however, VAE-ARNF seems to be able to make reasonable predictions as well, which is also supported by its lower prediction MSE in table 5.1 compared to other models.

Table 5.1: Prediction MSE of pendulum and spring-mass data set for different models

	Pendulum	Spring-mass
VAELLT	111.852	39.4738
VAELLT-PNF	44.8339	45.1883
VAE-ARNF	299.382	26.1941
VAE-GAN	6.69549	1.44907

5.3 Learned Representations

Extending the state. As done in similar prior work [30, 14], the models for the pendulum data set were trained using a latent space of three variables. In order to highlight and provide interpretations to the learned representation $c \in \mathbb{R}^3$, we extend the ground truth state dimensionality from $\mathbb{R}^2$ to $\mathbb{R}^3$ by calculating the cosine and sine of the angle θ. In other words, our ground truth state used in this comparison is $\{z_0 = \sin(\theta), z_1 = \cos(\theta), z_2 = \dot{\theta}\}$. Using this extended state, we calculate the learned representation metrics described in section 4.2. The models for the spring-mass data set were trained using a latent space $\in \mathbb{R}^2$, and the comparison in that case is done using the expected $\{z_0 = x, z_1 = \dot{x}\}$.

5.3.1 Pendulum System

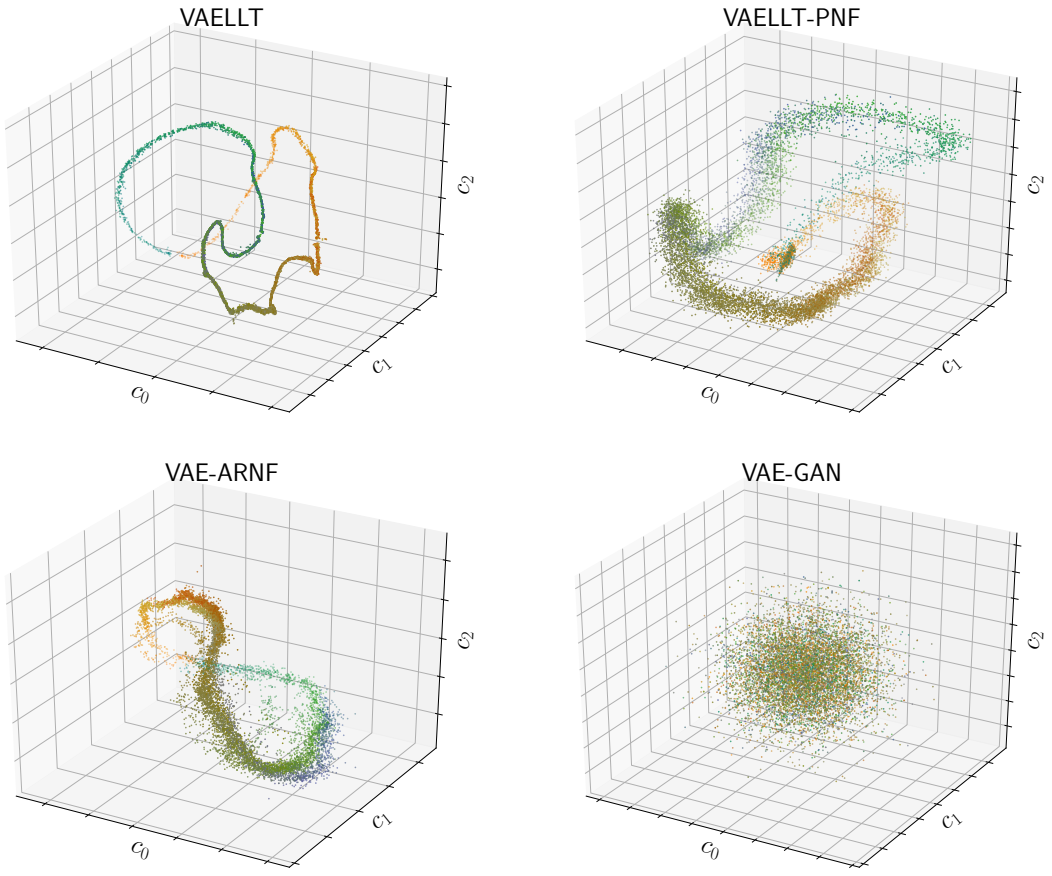

Figure 5.3: Learned representations for the four models trained on the pendulum data set.

Figure 5.3 displays the learned latent space for the four models trained using the pendulum data set. We can observe that the latent space in the VAE-GAN model has been completely ignored and remains unstructured. This has previously been

Table 5.2: Disentanglement, completeness and informativeness score using the pendulum data set with the lasso regressor.

	c_0	c_1	c_2	Disentanglement score
VAELLT	0.237	0.110	0.144	0.141
VAELLT-PNF	0.096	0.566	0.124	0.256
VAE-ARNF	0.073	0.125	0.091	0.084
VAE-GAN	1.000	1.000	NaN	NaN
	z_0	z_1	z_2	Completeness score
VAELLT	0.028	0.118	0.046	0.064
VAELLT-PNF	0.376	0.179	0.114	0.223
VAE-ARNF	0.071	0.096	0.039	0.069
VAE-GAN	NaN	NaN	0.868	NaN
	z_0	z_1	z_2	Informativeness score
VAELLT	0.426	0.410	0.999	0.611
VAELLT-PNF	0.534	0.606	0.995	0.712
VAE-ARNF	0.243	0.209	0.993	0.482
VAE-GAN	1.000	1.000	1.000	1.000

reported in the literature as consequence of a strong decoder [28, 25], which the VAE-GAN model arguably has. If the decoder is sufficiently powerful, it will be able to reconstruct the data without relying on the latent space, and thus the latent space $\mathbf{z}$ remains ignored. When this happens, the KL divergence term of the ELBO learning objective simply enforces the posterior approximation distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to become the prior $p(\mathbf{z})$, which is what we can observe has occurred for the VAE-GAN model in the figure.

Figure 5.3 also illustrates how the other models have learned some sort of interpretable representations. In this case we can see that the learned representation relates c_0 and c_1 in a circular manner, and a rough interpretation could be that they represent the sine and cosine of the ground truth angle θ.

Table 5.2 details the representation learning scores and we can from this table quantitatively conclude that the VAE-GAN model has failed to learn a disentangled representation since both disentanglement and completeness score is NaN. Furthermore, its informativeness score tells us that trying to predict the ground truth variables from the learned code resulted in a 100% prediction error. It's difficult to draw any general conclusions about the models' disentanglement and completeness scores, but the informativeness scores paint a clearer picture: the learned representations are much worse at predicting the ground truth angular velocity z_2 compared to the sine and cosine angle z_0 and z_1.

5.3.2 Spring-Mass System

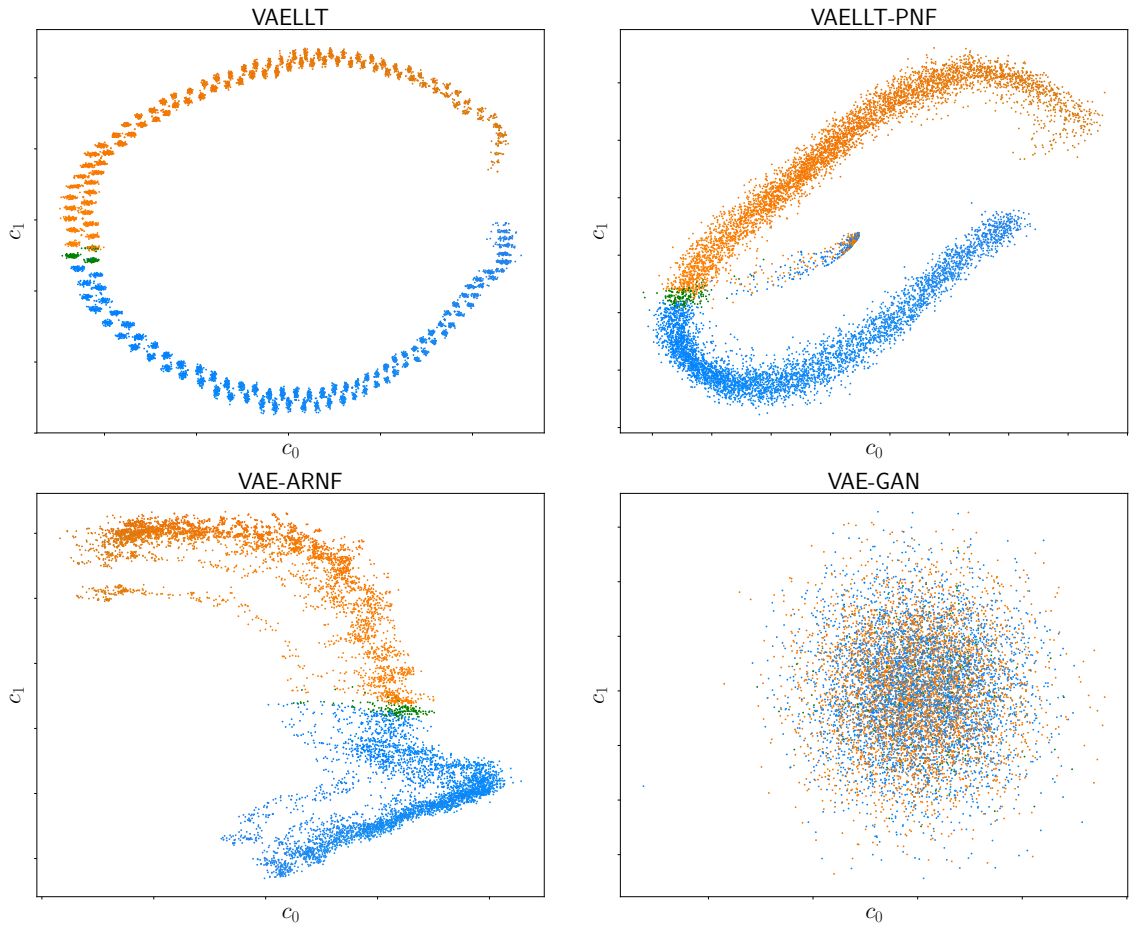

Figure 5.4: Learned representations for the four models trained on the spring-mass data set.

In fig. 5.4 we can note that once again the latent space of the VAE-GAN model is ignored, which is also reflected in the representation learning scores of table 5.3. We can see that VAELLT learns a representation where one latent code reaches its absolute maximum as the other latent code reaches its absolute minimum, in similar fashion to how a spring-mass system's ground truth velocity reaches its absolute maximum as the displacement reaches its equilibrium position. While it's natural assume that the ground truth has been "correctly" learned based on this observation, the prediction ability in fig. 5.2 clearly shows that VAELLT lacks an understanding of the velocity through the irregular changes of displacement between time steps.

The missing notion of velocity is also apparent in table 5.1 where the prediction error of z_2 for the VAELLT model remains high. In contrast, the learned latent space of VAE-ARNF is harder to interpret, but is able to provide a (comparably) low prediction error in both latent space (informativeness score of table 5.3) and in observation space (table 5.1).

Table 5.3: Disentanglement, completeness and informativeness score using the spring-mass data set with the Lasso and Forest regressor, respectively.

Lasso	c_0	c_1	Dis.	Forest	c_0	c_1	Dis.
VAELLT	0.017	0.000	0.020	VAELLT	0.024	0.002	0.007
VAELLT-PNF	0.001	0.000	0.001	VAELLT-PNF	0.020	0.002	0.007
VAE-ARNF	0.297	0.000	0.310	VAE-ARNF	0.040	0.001	0.005
VAE-GAN	NaN	NaN	NaN	VAE-GAN	0.000	0.000	0.000
Lasso	z_0	z_1	Com.	Forest	z_0	z_1	Com.
VAELLT	0.291	0.386	0.338	VAELLT	0.171	0.318	0.245
VAELLT-PNF	0.145	0.178	0.161	VAELLT-PNF	0.116	0.245	0.181
VAE-ARNF	0.436	0.036	0.236	VAE-ARNF	0.399	0.562	0.480
VAE-GAN	NaN	NaN	NaN	VAE-GAN	0.000	0.000	0.000
Lasso	z_0	z_1	Inf.	Forest	z_0	z_1	Inf.
VAELLT	0.516	0.569	0.542	VAELLT	0.022	0.320	0.171
VAELLT-PNF	0.578	0.547	0.563	VAELLT-PNF	0.144	0.303	0.224
VAE-ARNF	0.344	0.449	0.397	VAE-ARNF	0.024	0.160	0.092
VAE-GAN	1.000	1.000	1.000	VAE-GAN	0.801	0.767	0.784

6
Conclusion and Discussion

Given the poor results in the prediction task for many of the models we can conclude that learning dynamical systems directly in this unsupervised fashion remains a challenging task. For the models VAELLT, VAELLT-PNF and VAE-ARNF, investigation of the learned latent spaces gave some insight to the models' understanding of the dynamical systems. For instance, VAE-ARNF's better comprehension of the ground truth velocity in the spring-mass system, allowed for better prediction compared to other models. However, the VAE-GAN was able to produce very good predictions and this accomplishment cannot be ascribed the model's learned latent state, since it remained completely ignored. Exactly what lies behind VAE-GAN's excellent performance is hard to say, but for models where the latent space is used and shaped during training, it's reasonable to conclude that learning a notion of velocity (and time derivatives in general) is crucial to make accurate predictions.

The difficulty for generative models to learn notions of time derivatives has been hinted by prior work; both [18] and [30] use two images per time step as input to their variational autoencoders. By stacking two images together in this way, the latent space will now encode any information of the transition between the two adjacent frames, e.g. the velocity of the pendulum is directly observed rather than having to be figured out, which our results have shown to be quite difficult.

There are however several things regarding the usage of normalizing flows which could be interesting to investigate in future work. For instance, the autoregressive normalizing flows of VAE-ARNF belong to the family of affine flows, but in the recent years there has been work in non-affine flows [12, 13, 6], which might add additional flexibility needed to learn the pendulum system better. While it's disappointing that the added planar flow of VAELLT-PNF didn't improve the learning of the dynamical systems, it's quite expected since the transition model remained the same. An interesting idea for future work would be to draw inspiration from [24] and instead employ these normalizing flows in latent space such that correlation of latent variables is encouraged more explicitly.

Bibliography

[1] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction, 2018.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug 2013.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. cite arxiv:1606.01540.

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[5] Guillaume Desjardins, A Courville, and Yoshua Bengio. Disentangling factors of variation via generative entangling. arxiv. *Machine Learning*, 2012.

[6] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7509–7520, 2019.

[7] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[8] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

[9] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning, 2017.

[10] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*, 2018.

[13] Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. *arXiv preprint arXiv:1905.02325*, 2019.

[14] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data, 2016.

[15] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow, 2016.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.

[18] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction, 2018.

[19] Joseph Marino, Lei Chen, Jiawei He, and Stephan Mandt. Improving sequential latent variable models with autoregressive flows. In Cheng Zhang, Francisco Ruiz, Thang Bui, Adji Bousso Dieng, and Dawen Liang, editors, *Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference*, volume 118 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 08 Dec 2020.

[20] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[21] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation, 2017.

[22] Nikolay Ponomarenko, Lina Jin, Oleg Ieremeiev, Vladimir Lukin, Karen Egiazarian, Jaakko Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, et al. Image database tid2013: Peculiarities, results and perspectives. *Signal Processing: Image Communication*, 30:57–77, 2015.

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[24] Hooshmand Shokri Razaghi and Liam Paninski. Filtering normalizing flows. In *Bayesian Deep Learning Workshop at NeurIPS*, 2019.

[25] Sahand Rezaei-Shoshtari, David Meger, and Inna Sharf. Learning the latent space of robot dynamics for cutting interaction inference. *arXiv preprint arXiv:2007.11167*, 2020.

[26] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

[27] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014.

[28] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music, 2019.

[29] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[30] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.