ODE\textsuperscript{2}-VAE: Deep generative second order ODEs
with Bayesian neural networks

Çağatay Yıldız\textsuperscript{1}, Markus Heinonen\textsuperscript{1,2}, Harri Lähdesmäki\textsuperscript{1}
Department of Computer Science
Aalto University, Finland, FI-00076
\{cagatay.yildiz, markus.o.heinonen, harri.lahdesmaki\}@aalto.fi

Abstract

We present Ordinary Differential Equation Variational Auto-Encoder (ODE\textsuperscript{2}-VAE),
a latent second order ODE model for high-dimensional sequential data. Lever-
aging the advances in deep generative models, ODE\textsuperscript{2}-VAE can simultaneously
learn the embedding of high dimensional trajectories and infer arbitrarily complex
continuous-time latent dynamics. Our model explicitly decomposes the latent
space into momentum and position components and solves a second order ODE
system, which is in contrast to recurrent neural network (RNN) based time series
models and recently proposed black-box ODE techniques. In order to account for
uncertainty, we propose probabilistic latent ODE dynamics parameterized by deep
Bayesian neural networks. We demonstrate our approach on motion capture, image
rotation and bouncing balls datasets. We achieve state-of-the-art performance in
long term motion prediction and imputation tasks.

1 Introduction

Representation learning has always been one of the most prominent problems in machine learning. Lever-
aging the advances in deep learning, variational auto-encoders (VAEs) have recently been
applied to several challenging datasets to extract meaningful representations. Various extensions to
vanilla VAE have achieved state-of-the-art performance in hierarchical organization of latent spaces,
disentanglement and semi-supervised learning (Tschannen et al., 2018).

VAE based techniques usually assume a static data, in which each data item is associated with a
single latent code. Hence, auto-encoder models for sequential data have been overlooked. More
recently, there have been attempts to use recurrent neural network (RNN) encoders and decoders for
tasks such as representation learning, classification and forecasting (Srivastava et al., 2015; Lotter
et al., 2016; Hsu et al., 2017; Li and Mandt, 2018). Other than neural ordinary differential equations
(ODEs) (Chen et al., 2018b) and Gaussian process prior VAEs (GPPVAE) (Casale et al., 2018),
aforementioned methods operate in discrete-time, which is in contrast to most of the real-world
datasets, and fail to produce plausible long-term forecasts (Karl et al., 2016).

In this paper, we propose ODE\textsuperscript{2}-VAEs that extend VAEs for sequential data with a latent space
governed by a continuous-time probabilistic ODE. We propose a powerful second order ODE that
allows modelling the latent dynamic ODE state decomposed as position and momentum. To handle
uncertainty in dynamics and avoid overfitting, we parameterise our latent continuous-time dynamics
with deep Bayesian neural networks and optimize the model using variational inference. We show
state-of-the-art performance in learning, reproducing and forecasting high-dimensional sequential
systems, such as image sequences. An implementation of our experiments and generated video
sequences are provided at https://github.com/cagatayyildiz/ODE2VAE.

2 Probabilistic second-order ODEs

We tackle the problem of learning low-rank latent representations of possibly high-dimensional sequential data trajectories. We assume data sequences $x_{0:N} := (x_0, x_1, \ldots, x_N)$ with individual frames $x_k \in \mathbb{R}^D$ observed at time points $t_0, \ldots, t_N$. We will present the methodology for a single data sequence $x_{0:N}$ for notational simplicity, but it is straightforward to extend our method to multiple sequences. The observations are often at discrete spacings, such as individual images in a video sequence, but our model also generalizes to irregular sampling.

We assume that there exists an underlying generative low-dimensional continuous-time dynamical system, which we aim to uncover. Our goal is to learn latent representations $z_t \in \mathbb{R}^d$ of the sequence dynamics with $d \ll D$, and reconstruct observations $x_t \in \mathbb{R}^L$ for missing frame imputation and forecasting the system past observed time $t_N$.

2.1 Ordinary differential equations

In discrete-time sequential systems the state sequence $z_0, z_1, \ldots$ is indexed by a discrete variable $k \in \mathbb{Z}$, and the state progression is governed by a transition function on the change $\Delta z_k = z_k - z_{k-1}$. Examples of such models are auto-regressive models, Markov chains, recurrent models and neural network layers.

In contrast, continuous-time sequential systems model the state function $z_t : \mathcal{T} \rightarrow \mathbb{R}^d$ of a continuous, real-valued time variable $t \in \mathcal{T} = \mathbb{R}$. The state evolution is governed by a first-order time derivative

$$\dot{z}_t := \frac{dz_t}{dt} = h(z_t),$$

that drives the system state forward in infinitesimal steps over time. The differential $h : \mathbb{R}^d \rightarrow \mathbb{R}^d$ induces a differential field that covers the input space. Given an initial location vector $z_0 \in \mathbb{R}^d$, the system then follows an ordinary differential equation (ODE) model with state solutions

$$z_T = z_0 + \int_0^T h(z_t) dt. \tag{2}$$

The state solutions are in practise computed by solving this initial value problem with efficient numerical solvers, such as Runge-Kutta (Schober et al., 2019). Recently several works have proposed learning ODE systems $h$ parametrised as neural networks (Chen et al., 2018b) or as Gaussian processes (Heinonen et al., 2018).

2.2 Bayesian second-order ODEs

First-order ODEs are incapable of modelling high-order dynamics\footnote{Time-dependent differential functions $f(x, t)$ can indirectly approximate higher-order dynamics.}, such as acceleration or the motion of a pendulum. Furthermore, ODEs are deterministic systems unable to account for uncertainties in the dynamics. We tackle both issues by introducing Bayesian neural second-order ODEs

$$\ddot{z}_t := \frac{d^2 z_t}{dt^2} = f_W(z_t, \dot{z}_t), \tag{3}$$

which can be reduced to an equivalent system of two coupled first-order ODEs

$$\begin{cases} \ddot{s}_t = \dot{v}_t \\ \dot{v}_t = f_W(s_t, v_t), \end{cases} \quad \begin{bmatrix} s_T \\ v_T \end{bmatrix} = \begin{bmatrix} s_0 \\ v_0 \end{bmatrix} + \int_0^T \begin{bmatrix} v_t \\ f_W(s_t, v_t) \end{bmatrix} dt, \tag{4}$$

where (with a slight abuse of notation) the state tuple $z_t = (s_t, \dot{v}_t)$ decomposes into the state position $s_t$, which follows the state velocity (momentum) $v_t$. The velocity or evolution of change is governed by a neural network $f_W(s_t, v_t)$ with a collection of weight parameters $\mathcal{W} = \{W_\ell\}_{\ell=1}^L$ over its $L$ layers and the bias terms. We assume a prior $p(\mathcal{W})$ on the weights resulting in a Bayesian neural network (BNN). Each weight sample, in turn, results in a deterministic ODE trajectory (see Fig. 1).

The BNN acceleration field $f_W : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ depends on both state and velocity. For instance, in a pendulum system the acceleration $\ddot{z}$ depends on both its current location and velocity. The system is now driven forward from starting position $s_0$ and velocity $v_0$, with the BNN determining only how the velocity $v_t$ evolves.
With auto-encoders, we aim to learn latent representations \( z \) which results in the log densities over time. Variational inference that minimizes the Kullback-Leibler divergence, or equivalently maximizes the evidence lower bound (ELBO), results in efficient inference. In variational auto-encoders (VAE) an amortized variational approximation optimized by maximizing the marginal log likelihood but that generally involves intractable integrals. In variational auto-encoders (VAE) an amortized variational approximation \( q_\phi(z|x) \approx p_\theta(z|x) \) with parameters \( \phi \) is used instead (Jordan et al., 1999; Kingma and Welling, 2013; Rezende et al., 2014). Variational inference that minimizes the Kullback-Leibler divergence, or equivalently maximizes the evidence lower bound (ELBO), results in efficient inference.

\[ \frac{\partial \log q(z_t|W)}{\partial t} = -\text{Tr} \left( \frac{\partial f_W(z_t)}{\partial z_t} \right) dt = -\text{Tr} \left( \begin{bmatrix} \frac{\partial v_t}{\partial s_t} \\ \frac{\partial v_t}{\partial v_t} \end{bmatrix} \right) = -\text{Tr} \left( \begin{bmatrix} \frac{\partial f_W(s_t, v_t)}{\partial s_t} \\ \frac{\partial f_W(s_t, v_t)}{\partial v_t} \end{bmatrix} \right), \]

which results in the log densities over time,

\[ \log q(z_T|W) = \log q(z_0|W) - \int_0^T \text{Tr} \left( \frac{\partial f_W(s_t, v_t)}{\partial v_t} \right) dt. \]

3 ODE\(^2\)VAE model

In this section we propose a novel dynamic VAE formalism for sequential data by introducing a second order Bayesian neural ODE model in the latent space to model the data dynamics. We start by reviewing the standard VAE models and then extend it to our ODE\(^2\)VAE model.

With auto-encoders, we aim to learn latent representations \( z \in \mathbb{R}^d \) for complex observations \( x \in \mathbb{R}^D \) parameterised by \( \theta \), where often \( d \ll D \). The posterior \( p_\theta(z|x) \propto p_\theta(x|z)p(z) \) is proportional to the prior \( p(z) \) of the latent variable and the decoding likelihood \( p_\theta(x|z) \). Parameters \( \theta \) could be optimized by maximizing the marginal log likelihood but that generally involves intractable integrals. In variational auto-encoders (VAE) an amortized variational approximation \( q_\phi(z|x) \approx p_\theta(z|x) \) with parameters \( \phi \) is used instead (Jordan et al., 1999; Kingma and Welling, 2013; Rezende et al., 2014). Variational inference that minimizes the Kullback-Leibler divergence, or equivalently maximizes the evidence lower bound (ELBO), results in efficient inference.

3.1 Dynamic model

Building upon the ideas from black-box ODEs and variational auto-encoders, we propose to infer continuous-time latent position and velocity trajectories that live in a much lower dimensional space but still match the data well (see Fig. 2 for illustration). For this, consider a generative model that consists of three components: (i) a distribution for the initial position \( p(s_0) \) and velocity \( p(v_0) \) in the latent space, (ii) true (unknown) dynamics defined by an acceleration field, and (iii) a decoding likelihood \( p(x_t|s_t) \).

The generative model is given in Eqs. 7-11. Note that the decoding likelihood is defined only from the position variable. Velocity thus serves as an auxiliary variable, driving the position forward.

\[ s_0 \sim p(s_0) \]
\[ v_0 \sim p(v_0) \]
\[ s_t = s_0 + \int_0^t v_\tau d\tau \]
\[ v_t = v_0 + \int_0^t f_{\text{true}}(s_\tau, v_\tau) d\tau \]
\[ x_t \sim p(x_t|s_t) \quad i \in [0, N] \]
where $\mu_s$, $\sigma_s$, $\mu_v$, $\sigma_v$ are encoding neural networks. The encoder for the initial position depends solely on the first item in the data sequence $x_0$, whereas the encoder for the initial velocity depends on multiple data points $x_{0:m}$, where $m \leq N$ is the amortized inference length. We use neural network encoders and decoders whose architectures depend on the application (see the supplementary document for details). The variational approximation for the latent dynamics $q_{\text{ode}}(z_{1:N} | x_{0:N}, z_0, W)$ is defined implicitly via the instantaneous change of variable for the second order ODEs shown in Eq. 5. The initial density is given by the encoder $q_{\text{enc}}(z_0 | x_0)$, and density for later points can be solved by numerical integration using Eq. 6. Note that we treat the entire latent trajectory evaluated at observed time points, $Z \equiv z_{0:N}$, as a latent variable, and the latent trajectory samples $z_{1:N}$ are solved conditioned on the ODE initial values $z_0$ and BNN parameter values $W$. Finally, evidence lower bound (ELBO) becomes as follows (for brevity we define $X \equiv x_{0:N}$):

$$
\log p(X) \geq - \underbrace{\text{ELBO}}_{\text{VAE loss}}
$$

$$
= - \mathbb{E}_{q(W,Z|X)} \left[ \log \frac{q(W)q(Z|W,X)}{p(W)p(Z)} \right] + \mathbb{E}_{q(W,Z|X)} \left[ \log p(X|W,Z) \right]
$$

$$
= - \underbrace{\text{ODE regularization}}_{\text{VAE loss}}
$$

$$
= - \text{KL}[q(W)||p(W)] + \mathbb{E}_{q_{\text{enc}}(z_0|X)} \left[ - \log \frac{q_{\text{enc}}(z_0|X)}{p(z_0)} \right] + \log p(x_0|z_0)
$$

Figure 2: A schematic illustration of ODE$^2$VAE model. Position encoder $(\mu_s, \sigma_s)$ maps the first item $x_0$ of a high-dimensional data sequence into a distribution of the initial position $s_0$ in a latent space. Velocity encoder $(\mu_v, \sigma_v)$ maps the first $m$ high-dimensional data items $x_{0:m}$ into a distribution of the initial velocity $v_0$ in a latent space. Probabilistic latent dynamics are implemented by a second order ODE model $f_V$ parameterised by a Bayesian deep neural network ($\mathcal{W}$). Data points in the original data domain are reconstructed by a decoder.
We now examine each term in Eq. 16. The first term is the BNN weight penalty, which helps avoiding overfitting. The second term is the standard VAE bound, meaning that VAE is retrieved for sequences of length 1. The only (but major) difference between the second and the third terms is that the expectation is computed with respect to the variational distribution induced by the second order ODE. Finally, we optimize the Monte Carlo estimate of Eq. 16 with respect to variational posterior \( \{m, s\} \), encoder and decoder parameters, and also make use of reparameterization trick to tackle uncertainties in both the initial latent states and in the acceleration dynamics (Kingma and Welling, 2013).

### 3.3 Penalized variational loss function

A well-known pitfall of VAE models is that optimizing the ELBO objective does not necessarily result in accurate inference (Alemi et al., 2017). Several recipes have already been proposed to counteract the imbalance between the KL term and reconstruction likelihood (Zhao et al., 2017; Higgins et al., 2017). In this work, we borrow the ideas from Higgins et al. (2017) and weight the KL term resulting from the BNN with a constant factor \( \beta \). We choose to fix \( \beta \) to the ratio between the latent space dimensionality and number of weight parameters, \( \beta = \frac{q}{|W|} \), in order to counter-balance the penalties on latent variables \( W \) and \( z_i \).

Our variational model utilizes encoders only for obtaining the initial latent distribution. In cases of long input sequences, dynamic loss term can easily dominate VAE loss, which may cause the encoders to underfit. The underfitting may also occur in small data regimes or when the distribution of initial data points differs from data distribution. In order to tackle this, we propose to minimize the distance between the encoder distribution and the distribution induced by the ODE flow (Eqs. 12 and 6). At the end, we have an alternative, penalized target function, which we call ODE\(^2\)VAE-KL:

\[
L_{\text{ODE}^2\text{VAE}} = -\beta \text{KL}[q(W) || p(W)] + \mathbb{E}_{q(W,Z|X)} \left[ -\log \frac{q(Z|W,X)}{p(Z)} + \log p(X|W,Z) \right] + \gamma \mathbb{E}_{q(W)} \left[ \text{KL}[q_{\text{ode}}(Z|X) || q_{\text{enc}}(Z|W,X)] \right].
\]

We choose the constant \( \gamma \) by cross-validation. In practice, we found out that an annealing scheme in which \( \gamma \) is gradually increased helps optimization, which is also used in (Karl et al., 2016; Rezende and Mohamed, 2015).

### 3.4 Related work

Despite the recent VAE and GAN breakthroughs, little attention has been paid to deep generative architectures for sequential data. Existing VAE-based sequential models rely heavily on RNN encoders and decoders (Chung et al., 2015; Serban et al., 2017), with very few interest in stochastic models (Fraccaro et al., 2016). Some research has been carried out to approximate latent dynamics by LSTMs (Lotter et al., 2016; Hsu et al., 2017; Li and Mandt, 2018), which results in observations to be included in latent transition process. Consequently, the inferred latent space and dynamics do not fully reflect the observed phenomena and usually fail to produce decent long term predictions (Karl et al., 2016). In addition, RNNs are shown to be incapable of accurately modeling nonuniformly sampled sequences (Chen et al., 2018b), despite the recent efforts that incorporate time information in RNN architectures (Li et al., 2017; Xiao et al., 2018).

Recently, neural ODEs introduced learning ODE systems with neural network architectures, and proposed it for the VAE latent space as well for simple cases (Chen et al., 2018b). In Gaussian process prior VAE, a GP prior is placed in the latent space over a sequential index (Casale et al., 2018). To the best of our knowledge, there is no work connecting second order ODEs and Bayesian neural networks with VAE models.

\[2\] Although our variational approximation model assumes deterministic second-order dynamics, the underlying true model may also have more complex or stochastic dynamics.
Table 1: Comparison of VAE-based models

<table>
<thead>
<tr>
<th>Method</th>
<th>Higher order</th>
<th>Continuous-time dynamics</th>
<th>Stochastic state</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Kingma and Welling (2013)</td>
</tr>
<tr>
<td>VRNN</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Chung et al. (2015)</td>
</tr>
<tr>
<td>SRNN</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>Fraccaro et al. (2016)</td>
</tr>
<tr>
<td>GPPVAE</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>Casale et al. (2018)</td>
</tr>
<tr>
<td>DSAE</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>Li and Mandt (2018)</td>
</tr>
<tr>
<td>Neural ODE</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>Chen et al. (2018b)</td>
</tr>
<tr>
<td>ODE2 VAE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>current work</td>
</tr>
</tbody>
</table>

* GPPVAE uses a latent GP prior but only a discrete case was demonstrated in Casale et al. (2018).

4 Experiments

We illustrate the performance of our model on three different datasets: human motion capture (see the acknowledgements), rotating MNIST (Casale et al., 2018) and bouncing balls (Sutskever et al., 2009). Our goal is twofold: First, given a walking or bouncing balls sequence, we aim to predict the future sensor readings and frames. Second, we would like to interpolate an unseen rotation angle from a sequence of rotating digits. The competing techniques are specified in each section. For all methods, we have directly applied the public implementations provided by the authors. Also, we have tried several values for the hyper-parameters with the same rigor and we report the best results. To numerically compare the models, we sample 50 predictions per test sequence and report the mean and standard deviation of the mean squared error (MSE) over future frames. We include the mean MSE of mean predictions (instead of trajectory samples) in the supplementary.

We implement our model in Tensorflow (Abadi et al., 2016). Encoder, differential function and the decoder parameters are jointly optimized with Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001. We use Tensorflow’s own odeint fixed function, which implements fourth order Runge-Kutta method, for solving the ODE systems on a time grid that is five times denser than the observed time points. Neural network hyperparameters, chosen by cross-validation, are detailed in the supplementary material. We also include ablation studies with deterministic NNs and first order dynamics in the appendix.

4.1 CMU walking data

To demonstrate that our model can capture arbitrary dynamics from noisy observations, we experiment on two datasets extracted from CMU motion capture library. First, we use the dataset in Heinonen et al. (2018), which consists of 43 walking sequences of several subjects, each of which is fitted separately. The first two-third of each sequence is reserved for training and validation, and the rest is used for testing. Second dataset consists of 23 walking sequences of subject 35 (Gan et al., 2015), which is partitioned into 16 training, 3 validation and 4 test sequences. We followed the preprocessing described in Wang et al. (2008), after which we were left with 50 dimensional joint angle measurements.

We compare our ODE2 VAE against a GP-based state space model GPDM (Wang et al., 2008), a dynamic model with latent GP interpolation VGPLVM (Damianou et al., 2011), two black-box ODE solvers npODE (Heinonen et al., 2018) and neural ODEs (Chen et al., 2018b), as well as an RNN-based deep generative model DTSBN-S (Gan et al., 2015). In test mode, we input the first three frames and the models predict future observations. GPDM and VGPLVM are not applied to the second dataset since GPDM optimizes its latent space for input trajectories and hence does not allow simulating dynamics from any random point, and VGPLVM implementation does not support multiple input sequences.

The results are presented in Table 2. First, we reproduce the results in Heinonen et al. (2018) by obtaining the same ranking among GPDM, VGPLVM and npODE. Next, we see that DTSBN-S is not able to predict the distant future accurately, which is a well-known problem with RNNs. As expected, all models attain smaller test errors on the second, bigger dataset. We observe that neural ODE usually perfectly fits the training data but failed to extrapolate on the first dataset. This overfitting problem...
Table 2: Average MSE on future frames

<table>
<thead>
<tr>
<th>Model</th>
<th>Test error</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPDM</td>
<td>126.46 ± 34 N/A</td>
<td>Wang et al. (2008)</td>
</tr>
<tr>
<td>VGPLVM</td>
<td>142.18 ± 1.92 N/A</td>
<td>Damianou et al. (2011)</td>
</tr>
<tr>
<td>DTSBN-S</td>
<td>80.21 ± 0.04 34.86 ± 0.02</td>
<td>Gan et al. (2015)</td>
</tr>
<tr>
<td>npODE</td>
<td>45.74 22.96</td>
<td>Heinonen et al. (2018)</td>
</tr>
<tr>
<td>NEURAL ODE</td>
<td>87.23 ± 0.02 22.49 ± 0.88</td>
<td>Chen et al. (2018b)</td>
</tr>
<tr>
<td>ODE^2VAE</td>
<td>93.07 ± 0.72 10.66 ± 1.4</td>
<td>current work</td>
</tr>
<tr>
<td>ODE^2VAE-KL</td>
<td>15.99 ± 4.16 8.09 ± 1.95</td>
<td>current work</td>
</tr>
</tbody>
</table>

is not surprising considering the fact that only ODE initial value distribution is penalized. On the contrary, our ODE^2VAE regularizes its entire latent trajectory and also samples from the acceleration field, both of which help tackling overfitting problem. We demonstrate latent state trajectory samples and reconstructions from our model in the supplementary.

4.2 Rotating MNIST

Next, we contrast our ODE^2VAE against recently proposed Gaussian process prior VAE (GPPVAE) (Casale et al., 2018), which replaces the commonly iid Gaussian prior with a GP and thus performs latent regression. We repeat the experiment in Casale et al. (2018) by constructing a dataset by rotating the images of handwritten “3” digits. We consider the same number of rotation angles (16), training and validation sequences (360&40), and leave the same rotation angle out for testing (see the first row of Figure 4b for the test angle). In addition, four rotation angles are randomly removed from each rotation sequence to introduce non-uniform sequences and missing data (an example training sequence is visualized in the first row of Figure 4a).

Test errors on the unseen rotation angle are given in Table 3. During test time, GPPVAE encodes and decodes the images from the test angle, and the reconstruction error is reported. On the other hand, ODE^2VAE only encodes the first image in a given sequence, performs latent ODE integration starting from the encoded point, and decodes at given time points - without seeing the test image even in test mode. In that sense, our model is capable of generating images with arbitrary rotation angles. Also note that both models make use of the angle/time information in training and test mode. An example input sequence with missing values and corresponding reconstructions are illustrated in Figure 4a, where we see that ODE^2VAE nicely fills in the gaps. Also, Figure 4b demonstrates our model is capable of accurately learning and rotating different handwriting styles.

Table 3: Average prediction errors on test angle

<table>
<thead>
<tr>
<th>MODEL</th>
<th>TEST ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPPVAE-DIS</td>
<td>0.0309 ± 0.00002</td>
</tr>
<tr>
<td>GPPVAE-JOINT</td>
<td>0.0288 ± 0.00005</td>
</tr>
<tr>
<td>ODE^2VAE</td>
<td>0.0194 ± 0.00006</td>
</tr>
<tr>
<td>ODE^2VAE-KL</td>
<td><strong>0.0188 ± 0.0003</strong></td>
</tr>
</tbody>
</table>

4.3 Bouncing balls

As a third showcase, we test our model on bouncing balls dataset, a standard benchmark used in generative temporal modeling literature (Gan et al., 2015; Hsieh et al., 2018; Lotter et al., 2015). The dataset consists of video frames of three balls bouncing within a rectangular box and also colliding with each other. The exact locations of the balls as well as physical interaction rules are to be inferred from the observed sequences. We make no prior assumption on visual aspects such as ball count, mass, shape or on the underlying physical dynamics.
Figure 4: Panel (a) shows a training sequence with missing values (first row) and its reconstruction (second row). First row in panel (b) demonstrates test angles from different sequences, i.e., handwriting styles, and below are model predictions.

We have generated a training set of 10000 sequences of length 20 frames and a test set of 500 sequences using the implementation provided with Sutskever et al. (2009). Each frame is 32x32x1 and pixel values vary between 0 and 1. We compare our method against DTSBN-S (Gan et al., 2015) and decompositional disentangled predictive auto-encoder (DDPAE) (Hsieh et al., 2018), both of which conduct experiments on the same dataset. In test mode, first three frames of an input sequence are given as input and per pixel MSE on the following 10 frames are computed. We believe that measuring longer forecast errors is more informative about the inference of physical phenomena than reporting one-step-ahead prediction error, which is predominantly used in current literature (Gan et al., 2015; Lotter et al., 2015).

Predictive errors and example reconstructions are visualized in Figures 3 and 5. The RNN-based DTSBN-S nicely extrapolates a few frames but quickly loses track of ball locations and the error escalates. DDPAE achieves a much smaller error over time; however, we empirically observed that the reconstructed images are usually imperfect (here, generated balls are bigger than the originals), and also the model sometimes fails to simulate ball collisions as in Figure 5. Our ODE\(^2\)VAE generates long and accurate forecasts and significantly improves the current state-of-the-art by almost halving the error. We empirically found out that a CNN encoder that takes channel-stacked frames as input yields smaller prediction error than an RNN encoder. We leave the investigation of better encoder architectures as an interesting future work.

5 Discussion

We have presented an extension to VAEs for continuous-time dynamic modelling. We decompose the latent space into position and velocity components, and introduce a powerful neural second order differential equation system. As shown empirically, our variational inference framework results in Bayesian neural network that helps tackling overfitting problem. We achieve state-of-the-art performance in long-term forecasting and imputation of high-dimensional image sequences.

There are several directions in which our work can be extended. Considering divergences different than KL would lead to Wasserstein auto-encoder formulations (Tolstikhin et al., 2017). The latent ODE flow can be replaced by stochastic flow, which would result in an even more robust model. Proposed second order flow can also be combined with generative adversarial networks to produce real-looking videos.

Figure 5: An example test sequence from bouncing ball experiment. Top row is the original sequence. Each model takes the first three frames as input and predicts the further frames.
Acknowledgements.

The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. The calculations presented above were performed using computer resources within the Aalto University School of Science Science-IT project. This work has been supported by the Academy of Finland grants no. 311584 and 313271.

References


