Latent GP-ODEs with Informative Priors

Ilze Amanda Auzina
University of Amsterdam
ilze.amanda.auzina@gmail.com

Çağatay Yıldız
University of Tübingen
cagatay.yildiz@uni-tuebingen.de

Efstratios Gavves
University of Amsterdam
egavves@uva.nl

Abstract

For many complex systems the parametric form of the differential equation might be unknown or infeasible to determine. Earlier works have explored to model the unknown ODE system with a Gaussian Process model, however, the application has been limited to a low dimensional data setting. We propose a novel framework by combining a generative and a Bayesian nonparametric model. Our model learns a physically meaningful latent representation (position, momentum) and solves in the latent space an ODE system. The use of GP allows us to account for uncertainty as well as to extend our work with informative priors. We demonstrate our framework on an image rotation dataset. The method demonstrates its ability to learn dynamics from high dimensional data and we obtain state-of-the-art performance compared to earlier GP-based ODEs models on dynamic forecasting.

1 Introduction

One of the main properties of the physical world is that it evolves over time. This is often characterised in the form of an ordinary differential equation (ODE). An autonomous ODE can be defined as:

$$\dot{\mathbf{x}}_t := \frac{d\mathbf{x}_t}{dt} = \mathbf{f}(\mathbf{x}_t) \tag{1}$$

with the evolution over time is specified as:

$$\mathbf{x}_T = \mathbf{x}_0 + \int_0^T \mathbf{f}(\mathbf{x}_t)dt \tag{2}$$

where the state of the system is captured by vector $\mathbf{x}_t \in \mathbb{R}^D$ and it evolves in time $t \in \mathbb{R}_+$ from an initial state $\mathbf{x}_0$ to a time point T following its time derivative $\mathbf{f}(\mathbf{x}_t)$. In our work $\mathbf{f}(\cdot)$ denotes a differential equation which could be first or second order. Traditionally the dynamics are defined by expert knowledge of the system given prior observations and the parameters are found by numerical optimization [3]. However, for many problems the functional (parametric) form of $\mathbf{f}$ might be unknown. Hence, in recent years there has been growing amount of research that tries to learn the underlying continuous-time dynamics from data [5, 19].

An increasingly popular research direction is to model the unknown ODE with a Gaussian Process (GP) prior [10, 20, 25]:

A causal view on dynamical systems workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

$$\mathbf{f}(\mathbf{x}) \sim GP(\mathbf{0}, K(\mathbf{x}, \mathbf{x}')) \tag{3}$$

where we assume zero-mean GPs for simplicity and the covariance matrix $K(\mathbf{x}, \mathbf{x}')$ fully specifies the GP. In the remainder of this paper, we refer to ODE systems with GP priors as *GP-ODEs*.

The key property of GPs that make them applicable for learning unknown non-parametric ODE functions is that they encode functions where similar states $\mathbf{x}, \mathbf{x}'$ induce similar differentials $\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')$. In particular, this similarity is captured by the kernel function $K(\mathbf{x}, \mathbf{x}')$, which also permits prior knowledge embedding into the model.

In order to perform the integration in (2), the differential function $\mathbf{f}$ must be continuous, whereas GPs are stochastic processes. This was first noticed by Heinonen et al. [10], who address the issue by discarding the GP uncertainty in order to obtain a continuous function. Recently, Wilson et al. [23] demonstrated that function-space sampling of GP posteriors is possible thanks to the Matheron's rule, which describes a decoupled sampling from the posterior by decomposing it into a prior and an update term (for further details see Appendix A). Later, Hegde et al. [8], Yıldız et al. [25] showcased that this framework allows for sampling ODE instantiations, or vector fields $\mathbf{f}$, to simulate GP-ODE systems.

Despite achieving state-of-the-art results, GP-ODE models are demonstrated on *(i)* low-dimensional data settings, where the input is either some low dimension observations or simulation data of the actual ODE system, and *(ii)* without the use of prior knowledge of the system, which is an integral part of GP modeling. To the best of our knowledge, Solin et al. [20] is the only work in this line of research that considers a high-dimensional data problem by employing variational auto-encoders (VAEs) alongside a GP-ODE system. Yet, they do not train the feature encoder and dynamical model in an end-to-end fashion, leading to sub-optimal performance. This is in stark contrast with *static* VAE methods, which are proven to extract meaningful latent representations from high-dimensional data, specifically due to joint training.

Similarly, this line of research lacks a principled investigation of meaningful priors induced on GP-ODE systems, with an exception of [25] that modifies the right-hand-side of the ODE equation (2) to incorporate domain specific knowledge. Since the neural network based ODE models have been shown to benefit from inductive biases, such as second-order dynamics formulations [24], we believe GP-ODEs can also be improved by stronger priors.

In this work, we propose VAE-GP-ODE, a probabilistic dynamic model that extends previous work by learning dynamics from high-dimensional data with a structured GP prior. In particular, we propose a VAE to embed the high-dimensional input into a latent space. Subsequently, we model the latent space continuous-time dynamics with a GP. The use of a GP allows us to specify structural priors over the model dynamics, hence, we are able to constrain our dynamics given prior knowledge of the system. Our model is trained end-to-end using variational inference. We showcase our model's ability to learn, reproduce and forecast given a high-dimensional sequential input (image sequences).

2 Background

In this section, we review the standard VAE and Sparse Gaussian Process models.

2.1 Variational Auto-Encoders

The main task of an auto-encoder is to compress the input data $\mathbf{x} \in \mathbb{R}^D$ into a much lower dimensional $d \ll D$ latent variable $\mathbf{z} \in \mathbb{R}^d$ without loss of task-relevant information. Due to the non-linear neural network likelihood mapping, $p_\theta(\mathbf{x}|\mathbf{z})$, the posterior in VAEs becomes intractable and hence is approximated by an amortized variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$, where ϕ and θ are the parameters of the *encoders* and *decoders* [13, 18]. In a VAE setting the model parameters ϕ are estimated via variational inference, where the optimization task is to maximize the evidence lower bound (ELBO) (4) or equivalently minimise the Kullback-Leibler divergence between the approximate and true probability distribution.

$$\log p(\mathbf{x}) \geq \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z})]}_{\text{ELBO}} \tag{4}$$

2.2 Sparse Gaussian Processes (SGPs)

GPs (3) define priors over functions. In this work, we consider priors on vector-valued functions $\mathbf{f} : \mathbb{R}^D \mapsto \mathbb{R}^D$, which implies matrix-valued kernel functions $K(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{D\times D}$. While GPs allow for handling uncertainties, their computational complexity grows cubically with the number of input points. To tackle this, GPs are sparsified [15] by the introduction of inducing variables $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_J)^T \in R^{J\times D}$ and inducing locations $\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_J)^T \in R^{J\times D}$ such that $\mathbf{u}_m = \mathbf{f}(\mathbf{m}_j)$. In turn, the function values $\mathbf{f}(\cdot)$ is evaluated conditioned on the inducing variables:

$$p(\mathbf{U}) = N(\mathbf{U}|\mathbf{0}, \mathbf{K_{MM}}) \tag{5}$$

$$p(\mathbf{f}|\mathbf{U}) = N(\mathbf{f}|\mathbf{A}\text{vec}(\mathbf{U}), \mathbf{K_{XX}} - \mathbf{A}\mathbf{K_{MM}}\mathbf{A^T}) \tag{6}$$

where $\mathbf{K}$ is the covariance matrix for the given input and $\mathbf{A} = \mathbf{K_{XM}}\mathbf{K_{MM}^{-1}}$. We have chosen to follow the SGP framework as it allows us to utilise mini batching during training and hence use stochastic gradient descent as an optimization algorithm.

3 Methods

We focus on problems where the goal is to learn an unknown ODE from a high-dimensional sequential input given prior knowledge of the system. In the following sections we first describe the generative and nonperametric Bayesian components of our model. Second, we define our optimisation objective following variational inference principles.

3.1 VAE-GP-ODE Model

In our work we follow the generative model specification as first introduced by Chen et al. [5]. In particular, our generative model involves latent variable(s) that correspond to the initial value(s) of the ODE system as well as a differential function to be inferred. Inspired by Yildiz et al. [24], we contrast first- and second-order ODE models, for which the corresponding latent variables are (a) the initial position, or (b) the initial position and initial velocity. As such the generative model can be expressed as follows (specifying for option (b), for option (a) ignore the velocity specification):

$$\mathbf{s}_0 \sim p(\mathbf{s}_0) \tag{7}$$

$$\mathbf{v}_0 \sim p(\mathbf{v}_0) \tag{8}$$

$$\mathbf{s}_t = \mathbf{s}_0 + \int_0^t \mathbf{v}_\tau d\tau \tag{9}$$

$$\mathbf{v}_t = \mathbf{v}_0 + \int_0^t \mathbf{f}(\mathbf{s}_\tau, \mathbf{v}_\tau) d\tau \tag{10}$$

$$\mathbf{x}_i \sim p(\mathbf{x}_i|\mathbf{s}_i) \quad i \in [0, N] \tag{11}$$

where the state position $\mathbf{s}_t$ and state velocity $\mathbf{v}_t$ together form the latent state vector $\mathbf{z}_t = (\mathbf{s}_t, \mathbf{v}_t)$ and N is the total number of steps in a trajectory. As it can be noted in Eq.11 the likelihood $p(\mathbf{x}|\mathbf{s})$ only depends on the state position while velocity acts as an auxiliary variable driving the dynamics.

GPs as Nonparametric ODE model In our work we consider both first and second-order ODEs. Second-order ODEs might be a favourable modelling choice as they are able to model higher order dynamics as well as they can act as an useful inductive bias leading to major performance improvements [7]. The first-order ODE follows Eq.1 while the second-order ODE has the following form:

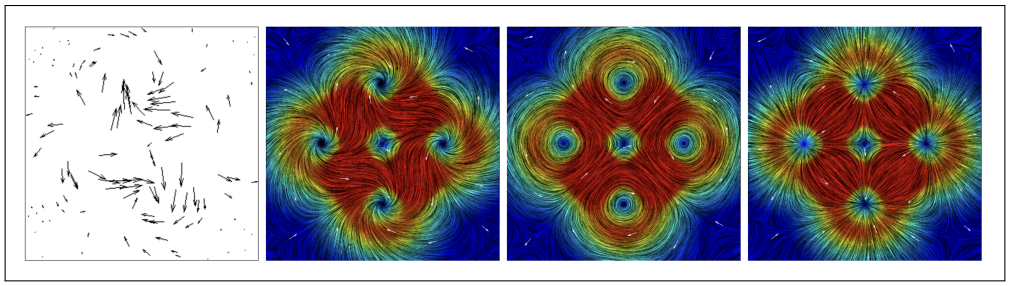

Figure 1: Learning a vector field. From lelt to right: samples, learned field, divergence- and curl-free components (kindly taken from [9]).

$$\ddot{\mathbf{x}}_t := \frac{d^2\mathbf{x}_t}{dt^2} = \mathbf{f}(\mathbf{x}_t, \dot{\mathbf{x}}_t) \tag{12}$$

In the case of a second-order ODE we decouple the ODE into two first-order ODEs:

$$\begin{cases} \dot{\mathbf{s}}_t = \mathbf{v}_t \\ \dot{\mathbf{v}}_t = \mathbf{f}(\mathbf{s}_t, \mathbf{v}_t) \end{cases} \quad \begin{bmatrix} \mathbf{s}_T \\ \mathbf{v}_T \end{bmatrix} = \begin{bmatrix} \mathbf{s}_0 \\ \mathbf{v}_0 \end{bmatrix} + \int_0^T \begin{bmatrix} \mathbf{v}_t \\ \mathbf{f}(\mathbf{s}_t, \mathbf{v}_t) \end{bmatrix} dt \tag{13}$$

where the derivative of position with respect to time $\dot{\mathbf{s}}_t$ is velocity and the derivative of velocity with respect to time $\dot{\mathbf{v}}_t$ is acceleration. We model $\mathbf{f}(\cdot)$ with a GP prior Eq.3. As the task at hand is to learn dynamics from high-dimensional input we follow the sparse variational inference framework for GPs using inducing variables [21].

Discussion on informative priors for GPs The differential function $\mathbf{f}(\cdot) : \mathbb{R}^D \mapsto \mathbb{R}^D$ defines a vector field. Therefore, we model the vector field as a vector-valued GP [1]:

$$\mathbf{f} \sim GP(\mathbf{0}, \mathbf{K}) \tag{14}$$

where $\mathbf{K}$ is an operator valued kernel, meaning that each entry is a matrix valued kernel $K(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{D\times D}$ while the kernel matrix over data points $\mathbf{X} \in \mathbb{R}^{N\times D}$ becomes $\mathbf{K} = (K(\mathbf{x_i}, \mathbf{x_j}))_{i,j=1}^N \in \mathbb{R}^{ND\times ND}$. As a baseline implementation we consider the identity decomposable kernel with a radial basis function (rbf) scalar matrix:

$$K_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \text{exp}\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right) \tag{15}$$

where l and σ^2 are the kernel parameters, called lengthscale and variance, respectively. After which we consider more complex operator valued kernels, such as, divergence-free kernel that assumes stronger structural bias on the vector field:

$$K_{\text{df}}(\mathbf{x}, \mathbf{x}') = \frac{\sigma_f^2}{l^2} e^{-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2l^2}} \left(\left(\frac{\mathbf{x} - \mathbf{x}'}{l}\right)\left(\frac{\mathbf{x} - \mathbf{x}'}{l}\right)^T + \left((D-1) - \frac{||\mathbf{x} - \mathbf{x}'||^2}{l^2}\right) \cdot \mathbf{I}\right) \tag{16}$$

where D is the number of input dimensions (for more details see the Appendix C). The resulting vector field does not have any sources or sinks and therefore can model real world fluid flow problems. In principle, one can construct a covariance function that captures the known physical characteristic of the underlying vector field (Fig. 1). The benefit of such an approach is that it reduces search space of the *true* model, as well as makes the predictions aligned with the known physical properties of a given system.

3.2 Variational Inference

The joint probability model Given the generative model (7-11), the model unknowns are the initial values $\mathbf{z}_0$, differential function $\mathbf{f}$, inducing inputs $\mathbf{Z}$ and inducing outputs $\mathbf{U}$. Overall, the full joint distribution becomes:

$$p(\mathbf{x}_{0:N}, \mathbf{f}, \mathbf{z}_0, \mathbf{U}, \mathbf{Z}) = \prod_{i=0}^{N} p(\mathbf{x}_i|\mathbf{f}, \mathbf{z}_0)p(\mathbf{z}_0)p(\mathbf{f}|\mathbf{U})p(\mathbf{U})p(\mathbf{Z}) \tag{17}$$

Due to non-linear likelihood mapping, the exact likelihood becomes intractable. Similar to previous work [5, 24], we resort to variational inference that involves amortized inference for the initial values and the mean-field inference framework for inducing variables [11].

Variational distributions Relying on the independence between initial values and dynamics we propose the following posterior factorization:

$$q(\mathbf{f}, \mathbf{U}, \mathbf{z}_0|\mathbf{x}_{0:N}) = q_{\text{enc}}(\mathbf{z}_0|\mathbf{x}_{0:N})q(\mathbf{f}, \mathbf{U}) \tag{18}$$

$$= q_{\text{enc}}(\mathbf{z}_0|\mathbf{x}_{0:N})p(\mathbf{f}|\mathbf{U})q(\mathbf{U}), \tag{19}$$

where the middle term was defined in (6). For initial values $\mathbf{z}_0$, we propose a *position encoder* for the first-order method while second-order formulation requires one *position encoder* and one *velocity encoder*. For the *position encoder*, the encoder takes as input only the first frame of the input sequence, whereas for the *velocity encoder* we consider the first n frames, where $n \ll N$ with N being the total sequence length. The variational encoding distribution is defined as

$$q_{\text{enc}}(\mathbf{z}_0|\mathbf{x}_{0:N}) = q_{\text{enc}}\left(\begin{pmatrix} s_0 \\ \mathbf{v}_0 \end{pmatrix} \middle| \mathbf{x}_{0:N}\right) = N\left(\begin{pmatrix} \mu_s(\mathbf{x}_0) \\ \mu_v(\mathbf{x}_{0:n}) \end{pmatrix}, \begin{pmatrix} \text{diag}(\sigma_s(\mathbf{x}_0)) & \mathbf{0} \\ \mathbf{0} & \text{diag}(\sigma_v(\mathbf{x}_{0:n})) \end{pmatrix}\right) \tag{20}$$

where $\mu_s, \mu_v, \sigma_s, \sigma_v$ are encoding neural networks. The variational approximation of the vector field is defined as

$$q(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{U})q(\mathbf{U})d\mathbf{U} \tag{21}$$

where $q(\mathbf{U})$ is a factorised Gaussian of the inducing variables across state dimensions

$$q(\mathbf{U}) = \prod_{d=1}^{D} N(\mathbf{u}_d|\mathbf{m}_d, \mathbf{Q}_d) \tag{22}$$

with $\mathbf{u}_d \in \mathbb{R}^J$, $\mathbf{m}_d \in \mathbb{R}^J$ and $\mathbf{Q}_d \in \mathbb{R}^{J\times J}$. We finally note that the inducing locations $\mathbf{Z}$ are treated as kernel hyperparameters [11].

Evidence lower bound Our goal is to learn the vector field $\mathbf{f}$, the inducing points $\mathbf{U}$, as well as the parameters for the latent encoding $\mathbf{z}$. Hence, the evidence lower bound (ELBO) becomes as follows:

$$\log p(X) = \log \int\int\int p(X|\mathbf{z}_0, \mathbf{f})p(\mathbf{z}_0)p(\mathbf{f}|\mathbf{U})p(\mathbf{U})d\mathbf{z}_0 d\mathbf{f} d\mathbf{U} \tag{23}$$

$$= \log \int\int\int \frac{q(\mathbf{f}, \mathbf{U}, \mathbf{z}_0|X)}{q(\mathbf{f}, \mathbf{U}, \mathbf{z}_0|X)} p(X|\mathbf{z}_0, \mathbf{f})p(\mathbf{z}_0)p(\mathbf{f}|\mathbf{U})p(\mathbf{U})d\mathbf{z}_0 d\mathbf{f} d\mathbf{U} \tag{24}$$

$$\geq \underbrace{\mathbb{E}_{q(\mathbf{f},\mathbf{U},\mathbf{z_0}|X)}\left[\log p(X|\mathbf{z_0}, \mathbf{f})\right]}_{L_x} + \underbrace{\mathbb{E}_{q_{\text{enc}}(\mathbf{z}_0|X)}\left[\log \frac{p(\mathbf{z}_0)}{q_{\text{enc}}(\mathbf{z_0}|X)}\right]}_{L_z} + \underbrace{\mathbb{E}_{q(\mathbf{f},\mathbf{U})}\left[\log \frac{p(\mathbf{f}|\mathbf{U})p(\mathbf{U})}{p(\mathbf{f}|\mathbf{U})q(\mathbf{U})}\right]}_{L_u} \tag{25}$$

where L_x is the likelihood term, L_z is the regularization KL term and L_u inducing KL term (for details see Appendix B). The expected likelihood term L_x decomposes over time:

$$L_x = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{f},\mathbf{U},\mathbf{z}_0|\mathbf{x}_{0:N})} \log p(\mathbf{x}_i|\mathbf{f},\mathbf{z}_0). \tag{26}$$

We compute the likelihood $p(\mathbf{x}_i|\mathbf{f},\mathbf{z}_0)$ over the ODE state solution $\mathbf{z}_i = \mathbf{z}_0 + \int_0^{t_i} \mathbf{f}(\mathbf{z}(\tau))d\tau$, which later is mapped to the data space via the decoder. Since the expectation is not available in closed form, we resort to Monte Carlo approximation, where we sample realizations of the vector field $\mathbf{f}^s \sim q(\mathbf{f}^s)$ and the encoder distributions $\mathbf{z}_0^s \sim q_{\text{enc}}(\mathbf{z}_0^s|\mathbf{x}_{0:m})$. Consequently, the likelihood term can be approximated as

$$L_x \approx \frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{N} \log p(\mathbf{x}_i|\mathbf{f}^s,\mathbf{z}_0^s) = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{N} \log p(\mathbf{x}_i|\text{dec}(\mathbf{z}_i^s)) \tag{27}$$

4 Experiments

We showcase the performance of our model on rotating MNIST dataset [4]. The first goal of our model is to showcase that *GP-ODE* type models are suitable for large-scale data, meaning that we are able to successfully predict future frames given the initial frame $\mathbf{x}_1$ for 1^{st} order ODEs or initial five frames $\mathbf{x}_{1:5}$ for 2^{nd} order ODE. The second goal is to constrain the dynamics given an informative prior and thereby allowing to extrapolate to time points outside the training domain.

The model was implemented in PyTorch. The encoder, Gaussian process, and decoder parameters were all jointly optimized using the Adam optimizer [12] with a learning rate of 0.001. For the numerical solver, we used the torchdiffeq package [6]. An implementation of our experiments is provided at `https://github.com/IlzeAmandaA/VAE-GP-ODE`.

Dataset For the rotating MNIST design, we follow the experimental set-up as introduced by Casale et al. [4]. The dataset is created by rotating images of handwritten "3" digits. In total, we consider 16 rotation angles, 360 training sequences, and 40 test sequences. Each sequence starts with a random initial angle (see Appendix D fig. 5). We set the latent dimensionality of the VAE to 6 and keep it fixed across both first- and second-order ODE. We use a Bernoulli distribution for the reconstruction likelihood as in [24]. We initialize the hyperparameters of the GP model to 2.0 (lengthscale) and 1.0 (variance), respectively.

Qualitative Analysis To analyse the performance of the model we demonstrate model's predicition in data space as in fig. 2. In addition, we visualize the latent trajectories $\mathbf{z}_{1:T}$ (only position component $\mathbf{s}_{1:T}$ for second-order dynamics). For this, we map the six dimensional latent space into a $2D$ space using principle component analysis (PCA). For example, fig. 3 shows the embedding, where each color corresponds to a latent trajectory associated with a distinct data trajectory. For all latent space plots, we double the integration time, i.e., rollouts with $T = 32$ time points.

Experimental Design We investigate 3 types of design choices: (i) pretrained VAE latent space, (ii) 1^{st} versus 2^{nd} order ODE system, and (iii) informative prior (df-kernel).

4.1 Pretrained VAE Latent Space

To contrast our work to Solin et al. [20] we repeat their training set-up where the dynamics are learned in a latent space that is induced by a separately trained VAE on the dataset. The results obtained confirm our presupposition that training end-to-end is advantageous over decoupled training. When the training is decoupled then the embeddings obtained are not constrained by the dynamics of the observed process. Hence, the model is unsuccessful in learning the dynamics, see fig 2 'Pretrained VAE'. The qualitative observation matches the learned latent space where the predictions collapse to the Gaussian prior (see fig. 3 'Pretrained VAE').

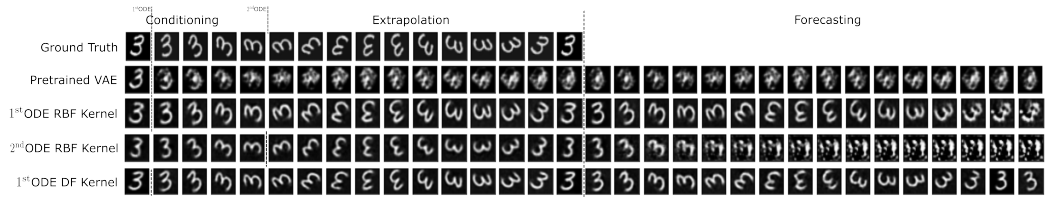

Figure 2: Reconstructed test sequences. Top row: ground truth. Second row: Pretrained VAE (model according to [20]). Bottom 3 rows: VAE-GP-ODE with varying order differential equation (1st or 2nd order), and without (RBF kernel) or with (DF kernel) informative prior.
Conditioning: input for the encoder (1st ODE only $\mathbf{x}_1$, 2nd ODE $\mathbf{x}_{1:5}$ frames); Extrapolation: model's prediction within the training data sequence length ($T_{max} = 16$); Forecasting: model's prediction outside the training data sequence length ($T > 16$).

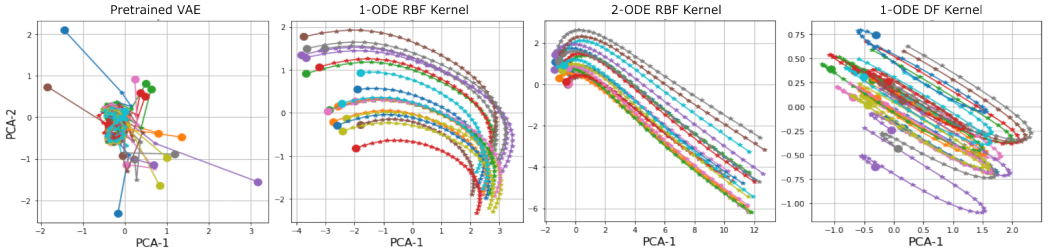

Figure 3: Learned latent space. From left: Pretrained VAE [20]; 1st order ODE with RBF kernel; 2nd order ODE with RBF; and 1st order ODE with DF kernel.
Each color corresponds to a latent trajectory associated with a distinct data sample. A circle indicates the start of the trajectory and the subsequent stars a subsequent time point up until T=32.

4.2 First versus Second Order ODE

There are no major performance differences between the 1st and 2nd order ODE systems as shown in figure 2. Both models successfully perform an extrapolation up to the time period $T = 16$, confirming model's flexibility to learn dynamics for both 1st and 2nd order ODE formulations. With respect to mean square error (MSE) both implementation choices achieve similar performance (see table 1). Important to note that neither of the models is able to forecast to time points outside the training regime (see fig. 2, section Forecasting) with the 1st ODE model capturing half a rotation and the 2nd ODE model only the first few angles as $T > 16$. This observation comes in line with the learned latent vector field as seen in figure 3. The learned latent space has not captured the rotational dynamics. Therefore, when the trajectory is forecasted beyond the training time points, the model is unable to *repeat* the rotation perfectly.

Table 1: MSE on test sequences (RBF kernel)

VAE-GP-ODE (RBF kernel)	MSE (std)
1st ODE	0.047 ($\pm$0.13)
2nd ODE	0.041 ($\pm$0.12)

4.3 Informative Prior

In order to bridge the gap between the unobserved true and the inferred dynamics, we set the kernel of the GP-ODE model to have divergence-free (DF) properties. We only investigate the 1st ODE model since the DF kernel input and output dimensions must match. The obtained results show that adding an informative prior to the GP-ODE model is beneficial (see fig. 2). The DF kernel specification constraints the latent space, therefore, the resulting model is able to better forecast outside the training regime $T > 16$. This is confirmed by investigating the latent space which now forms a loop with an increased prediction horizon, in contrast to the results obtained by the RBF kernel where the latent space does not capture the full rotation (see fig. 3). This is of importance, as the training data contains only a single rotation, so there is no information in the data itself that after $T > 16$ the rotation should continue, but by specifying an informative prior (DF kernel) the model has learned exactly this. As a

consequence, the model is able to forecast to time points far in the future, for example see figure 4 where the model is able to forecast even for $10 * T(T = 16)$ time points.

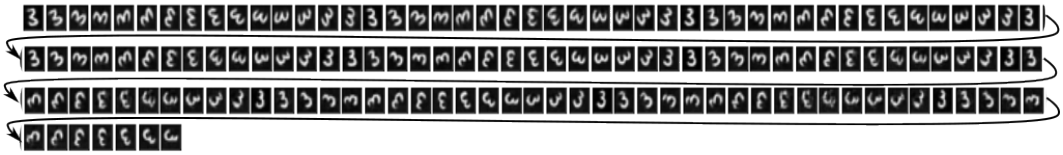

Figure 4: Forecasting to $10 \times T(T = 16)$ future frames with 1st ODE DF kernel model.

5 Discussion

In the present work, we introduce end-to-end *GP-ODE* models for continuous-time dynamic modeling for high-dimensional data. We propose a generative model for latent feature extraction in combination with a nonparametric Bayesian model for learning the differential equation. As shown in the results our framework obtains state-of-the-art extrapolation performance on high-dimensional image sequences, as well as is able to forecast to time points well beyond the training regime when an informative prior is specified. To our knowledge, this is the first work that successfully applies GP-ODE type model for learning dynamics from high dimensional data with physics inspired informative priors in an end-to-end fashion.

In contrast to earlier works [7, 24] we do not observe a performance improvement with an explicit second order bias. We conjecture that the benefits of 2nd ODE systems might be dataset-dependent. More concretely, a second order model for rotating MNIST dataset corresponds to learning dynamics in a polar coordinate system where the latent states correspond to the angular position and angular velocity. Therefore, a 1st ODE model might learn the dynamics in some other coordinate space, e.g., Euclidean space. Such an interpretation would also explain the differences in latent trajectories as seen in figure 3: For the 1st ODE system, the state of the system is *some* coordinate in the latent plane, hence, the velocity changes are less identifiable. For the 2nd ODE system, the visualized state is the angular position, which follows an almost linear trend matching the constant rotation speed of the data trajectories.

Moreover, even though the model is successful in constraining the dynamics given an informative prior, the obtained performance is sensitive to the hyperparameter initialization of the GP kernel. Even for a slightly different initialization (variance 0.7, lengthscale kept unchanged) the model learns to a certain extent a different latent state, resulting in forecasting with varying quality (see Appendix D fig. 6, fig. 7). This could be partly accounted by the PCA transformation to 2d space, however it also indicates that the current method is lacking some additional constraint as multiple alternative latent space representations are seen as equally likely by the model. Similarly, if the dataset is constrainted to only have a fixed initial angle, the model does not learn the rotation in the latent space and is therefore unable to forecast (see Appendix D, fig. 8, fig.9). In our future work we are planning to address this by adjusting the model design.

Additional future research direction could be to look into alternative informative priors. For example, for the given dataset perhaps a positive-curl kernel would bring even further performance improvements. Similarly, a combination of curl- and divergence-free kernels could also be of interest. On the more theoretical side it could be investigated whether the approximation methods used for efficient sampling from the posterior do not detrimentally affect the assumed physical constraints of the kernel.

References

[1] M. A. Alvarez, L. Rosasco, N. D. Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

[2] R. Brault, M. Heinonen, and F. Buc. Random fourier features for operator-valued kernels. In *Asian Conference on Machine Learning*, pages 110–125. PMLR, 2016.

[3] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[4] F. P. Casale, A. Dalca, L. Saglietti, J. Listgarten, and N. Fusi. Gaussian process prior variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

[5] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[6] R. T. Q. Chen. torchdiffeq, 2018. URL `https://github.com/rtqichen/torchdiffeq`.

[7] N. Gruver, M. Finzi, S. Stanton, and A. G. Wilson. Deconstructing the inductive biases of hamiltonian neural networks. *arXiv preprint arXiv:2202.04836*, 2022.

[8] P. Hegde, Ç. Yıldız, H. Lähdesmäki, S. Kaski, and M. Heinonen. Bayesian inference of odes with gaussian processes. *arXiv preprint arXiv:2106.10905*, 2021.

[9] M. Heinonen. Learning with spectral kernels. University Lecture, 2017.

[10] M. Heinonen, C. Yildiz, H. Mannerström, J. Intosalmi, and H. Lähdesmäki. Learning unknown ode models with gaussian processes. In *International Conference on Machine Learning*, pages 1959–1968. PMLR, 2018.

[11] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

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

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[14] L. Milenska. *An Impact of Divergence-Free Magnetic Field Interpolation Using a Solenoidal Gaussian Process Kernel*. University of California, Santa Cruz, 2018.

[15] J. Quinonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[16] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[17] C. E. Rassmussen and C. K. Williams. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[18] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[19] Y. Rubanova, R. T. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[20] A. Solin, E. Tamir, and P. Verma. Scalable inference in sdes by direct matching of the fokker–planck–kolmogorov equation. *Advances in Neural Information Processing Systems*, 34:417–429, 2021.

[21] M. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

[22] N. Wahlström. *Modeling of magnetic fields and extended objects for localization applications*. PhD thesis, Linköping University Electronic Press, 2015.

[23] J. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. Deisenroth. Efficiently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pages 10292–10302. PMLR, 2020.

[24] C. Yildiz, M. Heinonen, and H. Lahdesmaki. Ode2vae: Deep generative second order odes with bayesian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Ç. Yıldız, M. Kandemir, and B. Rakitsch. Learning interacting dynamical systems with latent gaussian process odes. *arXiv preprint arXiv:2205.11894*, 2022.

Appendix A Decoupled Sampling

Matheron's Rule describes that given two jointly Gaussian random variables $\mathbf{a}$ and $\mathbf{b}$, the random variable $\mathbf{a}$ conditional on $\mathbf{b} = \beta$ is equal in distribution to

$$(\mathbf{a}|\mathbf{b} = \beta) \overset{\mathrm{d}}{=} \mathbf{a} + \mathrm{Cov}(\mathbf{a}, \mathbf{b})\mathrm{Cov}(\mathbf{b}, \mathbf{b})^{-1}(\beta - \mathbf{b}). \tag{28}$$

For a GP defined as $f \sim GP(0, k)$ with a marginal $f_m = f(Z)$ and the process conditioned on $f_m = \mathbf{u}$ the above is as follows (Corollary 2 Wilson et al. [23]),

$$\underbrace{(f|\mathbf{u})(\cdot)}_{\text{posterior}} \overset{\mathrm{d}}{=} \underbrace{f(\cdot)}_{prior} + \underbrace{k(\cdot, Z)K_{m,m}^{-1}(\mathbf{u} - f_m)}_{\text{update}}. \tag{29}$$

Consequently, sampling from the posterior can be decoupled in a prior and an updated term. Furthermore, Wilson et al. [23] propose to use different bases for the prior and update terms. In particular, they propose to use Fourier basis functions for the prior term and canonical basis for the update term (as summarised by Hegde et al. [8])

$$\underbrace{f(\mathbf{x})|\mathbf{u}}_{\text{posterior}} \approx \underbrace{\sum_{i=1}^{F} w_i \phi_i(\mathbf{x})}_{\text{prior}} + \underbrace{\sum_{j=1}^{M} \nu_j K(\mathbf{x}, \mathbf{z}_j)}_{\text{udpate}}, \tag{30}$$

where F is the number of Fourier bases $\phi_i(\cdot)$ used with $w_i \sim N(0, 1)$ [16], and $K(\cdot, \mathbf{z}_j)$ is the kernel function with $\nu = K(\mathbf{Z}, \mathbf{Z})^{-1}(\mathbf{u} - \mathbf{\Phi}\mathbf{w}), \mathbf{\Phi} = \phi(\mathbf{Z}) \in \mathbb{R}^{M \times F}, \mathbf{w} \in \mathbb{R}^F$. For the experiment presented we compute the features maps for the RBF kernel and the divergence-free kernel as described in the following 2 paragaraphs.

RBF Kernel We compute the feature maps for the RBF kernel as $\phi_i(\mathbf{x}) = \sqrt{\frac{\sigma_f^2}{F}}\cos(\mathbf{x}^T\omega_i + b)$ where ω_i is sampled proportional to the spectral density of the rbf kernel $\omega_i \sim N(\mathbf{0}, \mathbf{\Lambda}^{-1})$, $\mathbf{\Lambda} = \mathrm{diag}(l_1^2, l_2^2, \ldots, l_D^2)$ is a diagonal matrix of the lengthscale parameters of the kernel and σ_f^2 is the signal variance parameter.

Divergence-Free Kernel For the divergence-free kernel we compute the feature maps as specified by Brault et al. [2]

$$\phi_i(\mathbf{x}) = \sqrt{\frac{\sigma_f^2}{F}} \begin{pmatrix} \cos(\langle x, \omega_i \rangle + b)B(\omega_i) \\ \sin(\langle x, \omega_i \rangle + b)B(\omega_i) \end{pmatrix}, \tag{31}$$

where $B(\omega) = I||\omega|| - \frac{\omega\omega^T}{||\omega||_2}$ and $\omega \sim N(\mathbf{0}, \mathbf{\Lambda}^{-1})$.

Appendix B Detailed Derivations

ELBO With the above variational model specification we can derive the following evidence lower bound

$$\log p(X) = \log \int\int\int p(X|\mathbf{z}_0, \mathbf{f}) p(\mathbf{z}_0) p(\mathbf{f}|\mathbf{U}) p(\mathbf{U}) d\mathbf{z}_0 d\mathbf{f} d\mathbf{U} \tag{32}$$

$$= \log \int\int\int \frac{q(\mathbf{f}, \mathbf{U}, \mathbf{z}_0|X)}{q(\mathbf{f}, \mathbf{U}, \mathbf{z}_0|X)} p(X|\mathbf{z}_0, \mathbf{f}) p(\mathbf{z}_0) p(\mathbf{f}|\mathbf{U}) p(\mathbf{U}) d\mathbf{z}_0 d\mathbf{f} d\mathbf{U} \tag{33}$$

$$\geq \mathbb{E}_{q(\mathbf{f},\mathbf{U},\mathbf{z}_0|X)} \log \left[\frac{p(X|\mathbf{z}_0,\mathbf{f}) p(\mathbf{z}_0) p(\mathbf{f}|\mathbf{U}) p(\mathbf{U})}{q(\mathbf{z}_0|X) q(\mathbf{f},\mathbf{U})} \right] \tag{34}$$

$$\geq \mathbb{E}_{q(\mathbf{f},\mathbf{U},\mathbf{z}_0|X)} \left[\log p(X|\mathbf{z}_0, \mathbf{f}) + \log \frac{p(\mathbf{z}_0)}{q(\mathbf{z}_0|X)} + \log \frac{p(\mathbf{f}|\mathbf{U}) p(\mathbf{U})}{p(\mathbf{f}|\mathbf{U}) q(\mathbf{U})} \right] \tag{35}$$

$$\geq \underbrace{\mathbb{E}_{q(\mathbf{f},\mathbf{U},\mathbf{z}_0|X)} \left[\log p(X|\mathbf{z}_0, \mathbf{f}) \right]}_{L_x} + \underbrace{\mathbb{E}_{q(\mathbf{z}_0|X)} \left[\log \frac{p(\mathbf{z}_0)}{q(\mathbf{z}_0|X)} \right]}_{L_z} + \underbrace{\mathbb{E}_{q(\mathbf{f},\mathbf{U})} \left[\log \frac{p(\mathbf{f}|\mathbf{U}) p(\mathbf{U})}{p(\mathbf{f}|\mathbf{U}) q(\mathbf{U})} \right]}_{L_u} \tag{36}$$

Hence the ELBO decomposes as

$$L = L_x + L_z + L_u \tag{37}$$

Regularization Term The regularization term L_z corresponds to KL divergence between the encoding distribution and the prior on the latent space.

$$L_z = \mathbb{E}_{q(\mathbf{z}_0|X)} \log \frac{p(\mathbf{z}_0)}{q(\mathbf{z}_0|X)} \tag{38}$$

$$= -KL \left[q_{\text{enc}}(\mathbf{z}_0|\mathbf{x}_{0:N}) || p(\mathbf{z}_0) \right] \tag{39}$$

Inducing KL The L_u corresponds to the KL divergence between the variational posterior and the prior distribution of inducing values. This term can be derived analytically as the KL between two multivariate Gaussians.

$$L_u = \mathbb{E}_{q(\mathbf{f},\mathbf{U})} \log \frac{p(\mathbf{f}|\mathbf{U}) p(\mathbf{U})}{p(\mathbf{f}|\mathbf{U}) q(\mathbf{U})} \tag{40}$$

$$= -KL \left[q(\mathbf{U}) || p(\mathbf{U}) \right] \tag{41}$$

Appendix C Kernel Specifications

Divergence-free kernel In the same way as Gaussian distributions are closed under linear transformations, Gaussian Processes are closed under linear operators [17]. An example of such linear operator could be differentiation or integration (for further details see [22]). Using this property we can construct new covariance functions. One of such functions is a divergence-free covariance function.

In particular, a vector field $\mathbf{g}(\mathbf{u})$ is known to be solenoidal if it can written as the curl of another vector field $\mathbf{f}(\mathbf{u})$

$$\mathbf{g}(\mathbf{u}) = \nabla \times \mathbf{f}(\mathbf{u}) \tag{42}$$

We model the vector field $\mathbf{f}(\mathbf{u})$ with a GP, where each component $f_i(\mathbf{u})$ is a GP with a squared exponential covariance function (RBF kernel)

$$f(\mathbf{u}) \sim GP(\mathbf{0}, k_{\text{rbf}}(\mathbf{u}, \mathbf{u}')) \tag{43}$$

The resulting covariance function of $\mathbf{g}(\mathbf{u})$ is given by

$$K(x,x') = \frac{\sigma_f^2}{l^2} e^{-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2l^2}} \left(\left(\frac{\mathbf{x}-\mathbf{x}'}{l} \right) \left(\frac{\mathbf{x}-\mathbf{x}'}{l} \right)^T + \left((n-1) - \frac{||\mathbf{x}-\mathbf{x}'||^2}{l^2} \right) \cdot \mathbf{I} \right) \tag{44}$$

where n is the dimension of the physical space. For a complete derivation see Wahlström [22] Appendix A. Important note on the above kernel is that it is only divergence-free analytically, in fact [14] has shown that the numerical divergence is non-zero.

Appendix D Supplementary Results DF-Kernel

Training Data For all experiments we used rotating MNIST dataset with random initial angle rotation. All sequences were still kept the same length $T = 16$ as the dataset follows a circular rotation.

Figure 5: Example dataset (random inital angle)

Different Initialization Hyperparameters initialized to 0.7 variance and 2.0 lengthscale. Model trained for 20'000 epochs.

Figure 6: Extrapolation to T=32 with DF-kernel (first-order ODE)

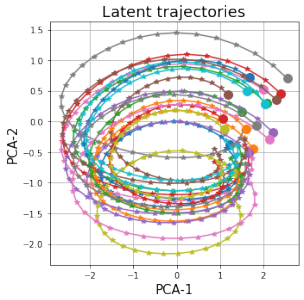

Figure 7: Latent space (first-order ODE) with DF-kernel

Fixed initial angle If the dataset is constrained to have all sequences with the same initial angle then the model is unsuccessful to learn the rotation in the latent space. Training settings: variance 1.0, lengthscale 2.0, df-kernel, and 5000 training epochs.

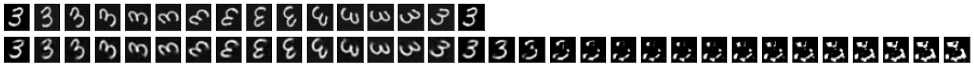

Figure 8: Extrapolation to T=32 with DF-kernel (first-order ODE) with fixed angle training data

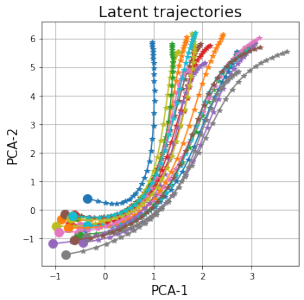

Figure 9: Latent space (first-order ODE) with DF-kernel with fixed angle training data