GENERATIVE ODE MODELING WITH KNOWN UNKNOWNS

Ori Linial
Faculty of Electrical Engineering
Technion Institute of Technology
Haifa, Israel
linial04@campus.technion.ac.il

Danny Eyten
Faculty of Medicine
Technion Institute of Technology
Haifa, Israel
Pediatric critical care unit
Rambam Medical Center
Haifa, Israel
danny.eytan@technion.ac.il

Uri Shalit
Faculty of Industrial Engineering and Management
Technion Institute of Technology
Haifa, Israel
urishalit@technion.ac.il

ABSTRACT

In several crucial applications, domain knowledge is encoded by a system of ordinary differential equations (ODE). A motivating example is intensive care unit patients: The dynamics of some vital physiological variables such as heart rate, blood pressure and arterial compliance can be approximately described by a known system of ODEs. Typically, some of the ODE variables are directly observed while some are unobserved, and in addition many other variables are observed but not modeled by the ODE, for example body temperature. Importantly, the unobserved ODE variables are “known-unknowns”: We know they exist and their functional dynamics, but cannot measure them directly, nor do we know the function tying them to all observed measurements. Estimating these known-unknowns is often highly valuable to physicians. Under this scenario we wish to: (i) learn the static parameters of the ODE generating each observed time-series (ii) infer the dynamic sequence of all ODE variables including the known-unknowns, and (iii) extrapolate the future of the ODE variables and the observations of the time-series. We address this task with a variational autoencoder incorporating the known ODE function, called GOKU-net for Generative ODE modeling with Known Unknowns.

1 INTRODUCTION

Many scientific fields use the language of ordinary differential equations to describe important phenomena. These include microbiology, ecology, medicine, epidemiology and finance, to name but a few. Typically, an ODE model of the form $\frac{dz(t)}{dt} = f_{\theta_f}(z(t))$ is derived from first principles and mechanistic understanding, where $z(t)$ are time-varying variables and θ_f are static parameters, or degrees of freedom, of the ODE model f. Once a model f_{θ_f} is specified, the values $\hat{\theta}^i_f$ and possibly $\hat{z}(t)$ are found that best fit an observed dataset. These estimated values are often of great interest: in ecology these might correspond to the carrying capacity of a species, whereas in medicine they might represent the cardiovascular dynamics of a patient in the course of critical illness. Predictions based on extrapolating the estimated models into the future are also widely used, for example predicting how a patient’s state will evolve or respond to specific interventions.

Usually the assumption is that the dynamic variables $z(t)$ are directly observed, possibly with some independent noise. At most, an assumption is made that the observations, which we denote hereafter as $x(t)$, are a known, fixed mapping of the unobserved $z(t)$. This assumption however is not always realistic: in the case of critically-ill patients for example, while some physiological variables are directly observed such as arterial blood pressure, others that are key determinants of the dynamical system such as cardiac contractility (the heart's ability to squeeze blood), stroke volume, or systemic vascular resistance are not only unobserved but also have a non-trivial mapping to the observed variables. Moreover, estimating the trajectory of these variables is of great clinical importance both diagnostically and in tailoring treatments aimed at their modification.

This work addresses the scenario where, on the one hand we have the mechanistic understanding needed to define the dynamic variables z and a corresponding ODE model f_{θ_f}, but on the other hand we cannot assume that we have a good model for how the variables z tie in to the observations x. In such a scenario, z and θ_f take on the role of *known-unknowns*: variables with a concrete meaning, which we do not know and wish to infer from data. We propose a variational autoencoder framework called GOKU-net, standing for Generative ODE Known-Unknown net. This is a VAE architecture with the known differential equation f at its heart, and with an added component that allows us on the one hand to effectively use standard VAE conditional-Gaussian parameterizations, yet still obtain estimates of the known-unknown quantities which correspond to their natural physical range.

2 Task definition

We consider a setting where we are given N observed trajectories $X^i = (x_0^i, ..., x_{T-1}^i)$, $i = 1, \ldots N$, each describing a time evolving phenomena observed at times $t = 0, \ldots T-1$. We assume each of these time sequences was generated by a noisy unknown emission process g from underlying latent trajectories $Z^i = (z_0^i, ..., z_{T-1}^i)$. The dynamics of the latent variables Z^i are governed by an ODE with known functional form f and unknown static parameters θ_f^i. Note that the latent trajectories share the same functional form but have different ODE parameters across the samples $i = 1, \ldots, N$:

$$\frac{dz^i(t)}{dt} = f_{\theta_f^i}(z^i(t)) \tag{1}$$

$$x_t^i = g(z^i(t)) + \varepsilon_t^i, \quad \varepsilon_t^i \sim \mathcal{N}(0, \sigma_x I). \tag{2}$$

Given a training set $\{X^i\}_{i=1}^N$, and a new test sequence $X' = \left(x'_0, \ldots, x'_{T-1}\right)$, our task is three-fold: (i) Estimate the static parameters θ_f for X', (ii) Estimate the latent states $z'_0, \ldots, z'_{T-1}$ corresponding to the times of the observations X', and (iii) Extrapolate z'_t and x'_t for a set of future times $t > T-1$.

Consider the pixel-pendulum experiment we report in Section 4.1: The latent state parameter Z is the pendulum's angle and angular velocity; and the ODE system f is the classic pendulum equation, see Eq. (7). We take the parameter θ_f to be a single number, the pendulum's length. Finally, we assume our observations X are frames in a video of the pendulum, as shown in Fig. 2b. That means the emission function g is the function that takes as input the angle of the pendulum and generates a 28×28 pixel image. The image is always scaled so that the length cannot be inferred from a single image. The task here is, given a previously unseen video, to infer the pendulum's length, the sequence of angles and velocities, and to extrapolate the video into the future of the sequence.

3 Model and method

Given N observed trajectories $X^i = (x_0^i, ..., x_{T-1}^i)$, $i = 1, \ldots N$, our main idea is based on explicit reconstruction of the latent trajectory Z, and the corresponding ODE parameters θ_f, in a variational autoencoder approach (Rezende et al., 2014; Kingma & Welling, 2013). As usual, that implies learning both an inference function (encoder) and an emission function (decoder). The inference function takes an observed sequence X^i as input, and has two components: The first infers the ODE parameters $\hat{\theta}_f^i$, and the second infers the initial $t = 0$ latent state $\hat{z}_0^i$. We next use the known ODE functional form f, the inferred ODE parameters $\hat{\theta}_f^i$ and the inferred initial state $\hat{z}_0^i$ to obtain an estimated trajectory $\hat{Z}^i$ by a numerical ODE solver. We then use $\hat{Z}^i$ as input to a *learned* emission function $\hat{g}$, obtaining a reconstructed signal $\hat{X}^i$. We estimate the log-likelihood of the reconstructed signal, and use stochastic backpropagation through the ODE solver in order to update the parameters of the inference network and emission model. Extrapolating the latent trajectory using the ODE solver lets us make estimates of X^i arbitrarily far forwards or backwards in time. Figure 1 illustrates the proposed model. Implementation details in the appendix.

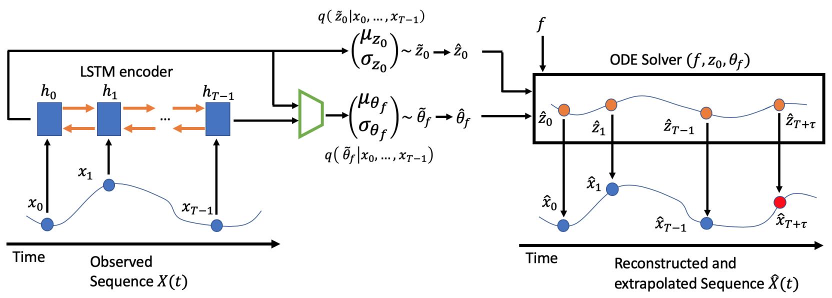

$q(\tilde{z}_0|x_0,\dots,x_{T-1})$
$\begin{pmatrix}\mu_{z_0}\\ \sigma_{z_0}\end{pmatrix} \sim \tilde{z}_0 \rightarrow \hat{z}_0$
$\begin{pmatrix}\mu_{\theta_f}\\ \sigma_{\theta_f}\end{pmatrix} \sim \tilde{\theta}_f \rightarrow \hat{\theta}_f$
$q(\tilde{\theta}_f|x_0,\dots,x_{T-1})$
LSTM encoder
h_0
h_1
h_{T-1}
...
x_0
x_1
x_{T-1}
Time
Observed Sequence $X(t)$
f
ODE Solver (f, z_0, θ_f)
$\hat{z}_0$
$\hat{z}_1$
$\hat{z}_{T-1}$
$\hat{z}_{T+\tau}$
$\hat{x}_0$
$\hat{x}_1$
$\hat{x}_{T-1}$
$\hat{x}_{T+\tau}$
Time
Reconstructed and extrapolated Sequence $\hat{X}(t)$

Figure 1: **GOKU-net model.** An observed signal $X(t)$ is taken as input by a bi-directional LSTM to produce $\hat{z}_0$ and $\hat{\theta}_f$. The ODE solver uses these values together with the given ODE function f to produce the latent signal $\hat{Z}$. Then reconstruct $\hat{X}$ using an emission network $\hat{g}$. The ODE solver can integrate $\hat{z}_t$ arbitrarily far forward in time, enabling the extrapolation of X for any $t = T + \tau$.

3.1 Grounding loss for under-identified systems

In some of our experiments we found that while reconstruction loss on X was small, the inferred latent Z and θ_f were far from the ground truth. This can happen because of under-identification: different sets of θ_f, z_0 can give rise to the same $\hat{X}$ by way of different emission models. We overcome this limitation by assuming we have a sparse set of observations from the latent space Z grounding our signal (Vani et al., 2017). This might be justified in some cases by assuming that observing the true latent Z is possible but difficult or expensive, thus not performed regularly. As we will see in Section 4 below, even as little as 1% of the latent data is often enough to ground the latent signals and parameters. We used these observations only during training, assuming such observations are not available to our model during test time, see appendix for more details.

4 Experiments

In this section we analyze how GOKU-net can be used for observed signal extrapolation and ODE parameter identification in two domains: an OpenAI Gym video simulator of a pendulum (Brockman et al., 2016; Greydanus et al., 2019); and a model of the cardiovascular system based on Zenker et al. (2007). We added the classic Lotka-Volterra system (Lotka, 1910) with added non-linear emission function in the appendix. In each case we train the model on a set of sequences with varying ODE parameters (θ_f) and initial conditions (z_0), and test on unseen sequences with parameters and initial conditions sampled from the same distribution as the train. We evaluate GOKU-net, and relevant baselines across three grounding conditions: no access to grounding latent states Z (denoted 0%), and access to randomly sampled 1% and 5% of the latent Z. The grounding observations are available only during training, except for the direct-inference (DI) baseline which must use them at test time too. In the appendix we give the full details of the baselines and architectures used for each dataset.

4.1 Single Pendulum From Pixels

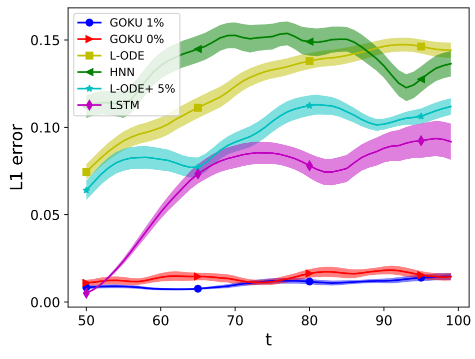
Our second task is a model of a friction-less pendulum from an observed sequence of frames. The pendulum's ODE has a single parameter which is the pendulum's length l, and the ODE state is $z_t = (\theta(t), \omega(t))$ (angle and angular velocity). Data set details in the appendix.

4.1.1 Evaluation and results

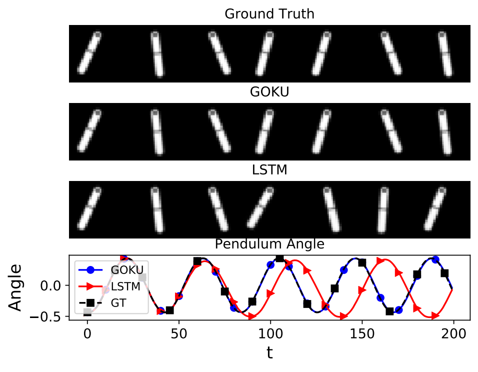
We present a comparison between our method and the baselines on the pixel-pendulum data set. Fig. 2a shows how the extrapolation error of the observed signals evolves over time, starting from time $t = 50$. In Fig. 2b we demonstrate how GOKU-net extrapolates on a single, randomly selected, signal when compared to the best performing baseline, the LSTM. The latent signals identification error and the ODE parameters identification error for different grounding mask rates in terms of L_1 error is shown in Table 4 in the appendix.

In terms of identification, GOKU-net performs much better than the baselines. We see that even with no latent samples (0%) GOKU-net identifies the parameter θ_f quite well, though it still does not identify Z in this case. When performing extrapolation, we see in Figure 2a that GOKU-net extrapolates much better than all baselines, even with no samples from the latent Z. This includes HNN (Greydanus et al., 2019) which has difficulty with the fact that the ODE parameter is not constant. GOKU-net also outperforms the L-ODE+ baseline, where we aid Latent ODE by giving it access to some latent space information.

We also experimented with data generated by and ODE system that is different from the one given as input to GOKU-net (named unknown unknowns). We did this by adding friction to the observed data, while GOKU is still given with the friction-less ODE. More details in the appendix.



(a)



(b)

Figure 2: Pixel pendulum results. (a) shows the mean extrapolation error for observations X over time steps after end of input sequence. Percentages in legend are percent grounding observation in training. (b) shows a comparison of the pendulum's predicted dynamics for GOKU-net that is trained with mask rate = 1%. The above 3 figures are examples of each method's predicted frames every 30 time steps and the below figure is the predicted pendulum's angle per time step.

4.2 Cardiovascular System

Our last experiment uses the cardiovascular system (CVS) model suggested by Zenker et al. (2007). This ODE system is more involved than the ones above. The system is a simplified mechanistic model of the cardiovascular system. In this system we wish to find the ODE parameters: $\theta_{CVS} = (I_{external}, R_{TPR_{Mod}})$ since these are known-unknowns that describe recognized clinical conditions: $I_{external} < 0$ tells us that a patient is currently losing blood, and $R_{TPR_{Mod}} > 0$ tells us that their total peripheral resistance is getting lower, which is a condition of distributive shock as can be seen in sepsis for example. Both conditions can lead to an observed drop in blood-pressure. Discerning the relative contribution of each of the two to such a drop is very important clinically as often the underlying causes are not immediately clear and the choice of correct treatment relies on their accurate estimation. The ODE state is $z_t = (SV(t), P_a(t), P_v(t), S(t))$. The observed state is the patient's vital signs and defined to be: $x_t = (P_a(t), P_v(t), f_{HR}(t))$. Note that some of the observed variables are the same as some of the latent variables, though with added noise as we explain now. More details, and data set details in the appendix.

4.2.1 Evaluation and results

In addition to identification and extrapolation, we attempt to classify each X^i series according to the sign of the inferred $I_{external}$ and $R_{TPR_{Mod}}$. These correspond to one of four possible clinical conditions: (1) Healthy (both non-negative), (2) Hemorrhagic shock ($I_{external} < 0$, $R_{TPR_{Mod}} \geq 0$), (3) Distributive shock ($I_{external} \geq 0$, $R_{TPR_{Mod}} < 0$) and (4) Combined shock ($I_{external} < 0$, $R_{TPR_{Mod}} < 0$). We compare this with the following: cluster the observations using K-means with $K = 4$, and assign each cluster to the most common true clinical condition.

Table 1 shows results on all the above tasks. We see in Table 1 that without any access to the latent space, GOKU-net successfully classifies which of the four clinical conditions the signal corresponds to, and extrapolates much better than the LSTM baseline. In this scenario, using Z as we do for the

Method	$I_{external}$	$R_{TPR_{Mod}}$	Class.	X extrap.
GOKU-0%	24 ± 2	$3 \pm .0$	0%	29 ± 1
K-Means	n/a	n/a	13%	n/a
LSTM	n/a	n/a	n/a	90 ± 2
DI-5%	$1 \pm .0$	$0 \pm .0$	0%	11 ± 10
GOKU-5%	26 ± 4	$3 \pm .0$	0	26 ± 1

Table 1: CVS parameter identification and extrapolation error ($\times 10^{-3}$), and classification error of clinical conditions. See text for K-means method. L-ODE and L-ODE+ failed and not shown.

grounding error is not realistic: measuring stroke-volume $SV(t)$ is a very hard task, and measuring $S(t)$ is impossible because it has no measurable meaning (it is a control signal). Nonetheless we add the grounding baselines for comparison, noting that DI with mask rate lower than 5% failed completely in reconstructing X.

5 Acknowledgments

We wish to thank Neta Ravid for her help with the CVS model, and Guy Tennenholtz and Hagai Rossman for their useful comments on the manuscript.

References

Mauricio Alvarez, David Luengo, and Neil D Lawrence. Latent force models. In *Artificial Intelligence and Statistics*, pp. 9–16, 2009.

David Barber and Yali Wang. Gaussian processes for bayesian estimation in ordinary differential equations. In *International conference on machine learning*, pp. 1485–1493, 2014.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Li-Fang Cheng, Bianca Dumitrascu, Michael Zhang, Corey Chivers, Michael Draugelis, Kai Li, and Barbara E Engelhardt. Patient-specific effects of medication using latent force models with gaussian processes. *arXiv preprint arXiv:1906.00226*, 2019.

Frank Dondelinger, Dirk Husmeier, Simon Rogers, and Maurizio Filippone. ODE parameter inference using adaptive gradient matching with gaussian processes. In *Artificial intelligence and statistics*, pp. 216–228, 2013.

Javier González, Ivan Vujačić, and Ernst Wit. Reproducing kernel hilbert space based estimation of systems of ordinary differential equations. *Pattern Recognition Letters*, 45:26–32, 2014.

Nico S Gorbach, Stefan Bauer, and Joachim M Buhmann. Scalable variational inference for dynamical systems. In *Advances in Neural Information Processing Systems*, pp. 4806–4815, 2017.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pp. 15353–15363, 2019.

Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ODE models with Gaussian processes. In *International Conference on Machine Learning*, pp. 1959–1968, 2018.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007.

Simon J Julier and Jeffrey K Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pp. 182–193. International Society for Optics and Photonics, 1997.

Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5 (2):102–119, 1960.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Thirty-first aaai conference on artificial intelligence*, 2017.

Alfred J Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 1910.

Đorđe Miladinović, Muhammad Waleed Gondal, Bernhard Schölkopf, Joachim M Buhmann, and Stefan Bauer. Disentangled state space representations. *arXiv preprint arXiv:1906.03255*, 2019.

Said Ouala, Duong Nguyen, Lucas Drumetz, Bertrand Chapron, Ananda Pascual, Fabrice Collard, Lucile Gaultier, and Ronan Fablet. Learning latent dynamics for partially-observed chaotic systems. *arXiv preprint arXiv:1907.02452*, 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Hossein Soleimani, Adarsh Subbaswamy, and Suchi Saria. Treatment-response models for counterfactual reasoning with continuous-time, continuous-valued interventions. In *33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017*. AUAI Press Corvallis, 2017.

Ankit Vani, Yacine Jernite, and David Sontag. Grounded recurrent neural networks. *arXiv preprint arXiv:1705.08557*, 2017.

Eric A Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pp. 153–158. Ieee, 2000.

Philippe Wenk, Alkis Gotovos, Stefan Bauer, Nico Gorbach, Andreas Krause, and Joachim M Buhmann. Fast Gaussian process based gradient matching for parameter identification in systems of nonlinear ODEs. *arXiv preprint arXiv:1804.04378*, 2018.

Philippe Wenk, Gabriele Abbati, Stefan Bauer, Michael A Osborne, Andreas Krause, and Bernhard Schölkopf. Odin: ODE-informed regression for parameter and state inference in time-continuous dynamical systems. *arXiv preprint arXiv:1902.06278*, 2019.

Sven Zenker, Jonathan Rubin, and Gilles Clermont. From inverse problems in mathematical physiology to quantitative differential diagnoses. *PLoS computational biology*, 3(11), 2007.

A Related work

We divide existing work into several categories. First, work on parameter identification in dynamical systems which assumes both the ODE function f and the emission function g are known. Second, work on latent state sequence modeling. This work does not assume any known dynamics or emission model. Finally, there is recent work tying together machine learning models and physical models in a task-specific way. We summed up the differences between some of these methods and ours in Table 2.

Methods for parameter identification ODE parameter identification has been the subject of many decades of research across many scientific communities. For example, classic work on state-space models, including methods such as the Kalman filter (Kalman et al., 1960) and its non-linear extensions (Jazwinski, 2007; Julier & Uhlmann, 1997; Wan & Van Der Merwe, 2000) can learn the parameters of a dynamic system from observations; however, they are limited to the case where the emission function g is known. Moreover, they usually perform inference on each sequence separately. Many machine learning methods have been proposed for this task, for example using reproducing kernel Hilbert space methods (González et al., 2014) and Gaussian Processes (Dondelinger et al., 2013; Barber & Wang, 2014; Gorbach et al., 2017), Fast Gaussian Process Based Gradient Matching (FGPGP, Wenk et al. (2018)) and recent follow up work (Wenk et al., 2019). In general these methods assume that the given signal is a the latent signal with independent additive noise.

Sequence modeling Methods for extrapolation of a given signal, assuming there is some unknown but arbitrary latent sequence have been proposed in LSTM Graves (2013), Deep Markov Models (Krishnan et al., 2017), Latent-ODE (Chen et al., 2018), NbedDyn (Ouala et al., 2019) the Disentangled State Space Model (DSSM, Miladinović et al. (2019)), and using Gaussian Processeses (Heinonen et al., 2018), among many others. These methods do not infer the ODE parameters as they do not learn any intrinsically meaningful latent space. They also do not exploit the prior information embedded in the mechanistic knowledge underlying the derivation of the ODE system f. Of the above methods, DSSM has been shown to learn a latent space which might under the right circumstances correspond to meaningful parameters, but that is not guaranteed, nor is it the goal of the method. In the healthcare regime, Cheng et al. (2019) proposed a method for learning a sequence which includes a dynamic system in the form of a latent force model (Alvarez et al., 2009); this approach builds on learning to fit general basis functions to describe the observed dynamics, and does not take as input an ODE system derived from prior mechanistic understanding.

Machine learning with mechanistic components Closer in spirit to our work is the work by Greydanus et al. (2019) on Hamiltonian neural networks. In their model the latent space can be interpreted in the form of learning a conserved physical quantity (Hamiltonian). Although related to our work, we note that not all ODE systems could be easily written as a Hamiltonian system, and not all of them have easily identified conserved quantities. Specifically, the systems we are interested in and that motivate our research do not usually have a Hamiltonian representation. In the field of learning for healthcare, Soleimani et al. (2017) show how a specific ODE model, the linear time-invariant impulse-response model, can be used in conjunction with latent-space models to estimate how a patient's measurements would react to interventions. This model brings together mechanistic modeling in terms of response to impulse treatments, along with data-driven modeling using Gaussian processes. It does not learn emission functions from Z, and focuses on the specific ODE model relevant to the task they address.

B Model

B.1 Generative model

Using the relationships between latent and observed variables given in Eqs. equation 1 and equation 2, we define a generative model over the set of ODE parameters θ_f, the latent states Z, and the observations X. Note that while we assume the true ODE function f is given to us, we estimate the emission by a learned function $\hat{g}$.

An important issue we must address is that in standard VAEs the prior distributions of the latent vectors z_0 and θ_f are set to be a zero-mean unit-variance Gaussian. However, in our case the latent space corresponds to specific variables with physical constraints: for example, the variable for blood

Method	ODE function	Emission function	θ_f and Z identification	X extrapolation
LSTM (Graves, 2013)	not required	learned	✗	✓
Latent-ODE (Chen et al., 2018)	not required	learned	✗	✓
HNN (Greydanus et al., 2019)	can be used	learned	✗	✓
DSSM (Miladinović et al., 2019)	not required	learned	✗	✓
NbedDyn (Ouala et al., 2019)	not required	partially given	✗	✓
ODIN (Wenk et al., 2019)	required	given	✓	✗
UKF (Wan & Van Der Merwe, 2000)	required	given	✓	✓
GOKU-net	required	learned	✓	✓

Table 2: Related Work: for each method we indicate whether it requires the ODE function f as input; whether it assumes the emission function g is known or can it learn it; and whether it allows identification of the "known-unknown" static parameters θ_f and dynamic variables Z; and whether it allows for extrapolating the observed signal X.

volume has a limited set of realistic values. We overcome this by defining arbitrary latent vectors $\tilde{z}_0$ and $\tilde{\theta}_f$ with standard Gaussian priors, and add deterministic transformations h_z and h_{θ_f} such that:

$$\tilde{z}_0 \sim \mathcal{N}(0, I), \quad z_0 = h_z(\tilde{z}_0), \tag{3}$$

$$\tilde{\theta}_f \sim \mathcal{N}(0, I), \quad \theta_f = h_\theta(\tilde{\theta}_f). \tag{4}$$

We then have z_t, $t > 0$ and X generated following Eqs. equation 1 and equation 2. With the above generative model, we have the following factorized joint distribution over latent and observed variables:

$$p(X, Z, \theta_f, \tilde{z}_0, \tilde{\theta}_f) =$$
$$p(\tilde{z}_0)p(\theta_f)p(z_0|\tilde{z}_0)p(\theta_f|\tilde{\theta}_f) \prod_{t=1}^{T-1} p(x_t|z_t)p(z_t|z_{t-1}, \theta_f).$$

This follows due to the conditional independence: for $t' \neq t$: $x_t \perp\!\!\!\perp (z_{t'}, \theta_f, \tilde{\theta}_f, \tilde{z}_0)|z_t$ and for $t' \neq t-1$: $z_t \perp\!\!\!\perp (X, \tilde{\theta}_f, \tilde{z}_0, z_{t'})|z_{t-1}, \theta_f$. The probabilities $p(z_0|\tilde{z}_0)$ and $p(\theta_f|\tilde{\theta}_f)$ are deterministic, meaning they are Dirac functions with the peak defined by Eqs. (3) and (4). The transition distribution $p(z_t|z_{t-1}, \theta_f)$ is also a Dirac function with the peak defined by Eq. (1). Finally, the emission distribution $p(x_t|z_t)$ is defined by Eq. (2).

B.2 Inference

We define the following joint posterior distribution over the unobserved random variables Z, θ_f, $\tilde{z}_0$ and $\tilde{\theta}_f$, conditioned on a sequence of observations X:

$$q(Z, \theta_f, \tilde{z}_0, \tilde{\theta}_f|X) =$$
$$q(\tilde{z}_0|X)q(\tilde{\theta}_f|X)q(z_0|\tilde{z}_0)q(\theta_f|\tilde{\theta}_f) \prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f).$$

The inference network conditionals $q(z_t|z_{t-1}, \theta_f)$, $q(z_0|\tilde{z}_0)$ and $q(\theta_f|\tilde{\theta}_f)$ are deterministic and mirror the generative model as defined in Eqs. (1), (3) and (4), respectively. For the posterior probabilities $q(\tilde{z}_0|X)$ and $q(\tilde{\theta}_f|X)$ we use conditional normal distributions as follows:

$$q(\tilde{z}_0|X) = \mathcal{N}(\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}), \quad [\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}] = \phi^{enc}_{\tilde{z}_0}(X),$$
$$q(\tilde{\theta}_f|X) = \mathcal{N}(\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f}), \quad [\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f}] = \phi^{enc}_{\tilde{\theta}_f}(X),$$

where $\phi^{enc}_{\tilde{z}_0}$ and $\phi^{enc}_{\tilde{\theta}_f}$ are learned neural networks.

B.3 Objective

We define our objective function using the evidence lower-bound (ELBO) variational objective proposed by Kingma & Welling (2013); Rezende et al. (2014):

$$\mathcal{L}(X) = \mathbb{E}_{q(Z,\theta_f,\tilde{\theta}_f,\tilde{z}_0|X)}\left[\log p(X|Z,\theta_f,\tilde{z}_0,\tilde{\theta}_f)\right] - KL\left[q(Z,\theta_f,\tilde{z}_0,\tilde{\theta}_f|X)||p(Z,\theta_f,\tilde{z}_0,\tilde{\theta}_f)\right]. \tag{5}$$

Since for all $t' \neq t$ we have $x_t \perp\!\!\!\perp (x_{t'}, z_{t'}, \theta_f, \tilde{z}_0, \tilde{\theta}_f)|z_t$, the first term of equation 5 decomposes as:

$$\mathbb{E}_{q(Z,\theta_f,\tilde{\theta}_f,\tilde{z}_0|X)}\left[\log p(X|Z,\theta_f,\tilde{z}_0,\tilde{\theta}_f)\right] = \sum_{t=0}^{T-1}\mathbb{E}_{q(z_t|X)}\left[\log p(x_t|z_t)\right].$$

The KL term decomposes into the following sum of KL terms:

$$KL\left[q(Z,\theta_f,\tilde{z}_0,\tilde{\theta}_f|X)||p(Z,\theta_f,\tilde{z}_0,\tilde{\theta}_f)\right] = KL\left[q(\tilde{\theta}_f|X)||p(\tilde{\theta}_f)\right] + KL\left[q(\tilde{z}_0|X)||p(\tilde{z}_0)\right].$$

The full derivation of the objective and the above decompositions are given in Appendix B.6.

B.4 Grounding Loss

As discussed in Section 3.1 we added sparse observations of the latent signal to ground the latent signal to its correct values. To this end, we added the following term to the objective function, with an adjustable hyper parameter:

$$\mathcal{L}_{ground} = \sum_{t=0}^{T-1} M(t)\cdot\|\hat{z}_t - z_t^{\text{observed}}\|_2^2 \tag{6}$$

where $M(t) \in \{0,1\}$ indicates for which time points the latent variables are observed. $\hat{z}_t$ is the latent vector predicted by the model and $z_t^{observed}$ is the observed samples of the latent vectors.

B.5 Implementation

We model $[\hat{g}, h_z, h_{\theta_f}, \phi_{\tilde{z}_0}^{enc}, \phi_{\tilde{\theta}_f}^{enc}]$ as neural networks: $\hat{g}$, h_z and h_{θ_f} as fully connected neural networks; $\phi_{\tilde{z}_0}^{enc}$ as an RNN which going over the observed X backwards in time to predict z_0; and $\phi_{\tilde{\theta}_f}^{enc}$ as a bi-directional LSTM (Huang et al., 2015) with fully connected networks from X into $\tilde{\theta}_f$. We use bi-directional LSTM for θ_f identification since θ_f is time invariant.

In order to perform stochastic backpropagation, we must calculate the gradient through the ODE defined by f and θ_f. We do this by using the Latent-ODE implementation (Chen et al., 2018) for the ODE solver with the adjoint method, set to use the Runge-Kutta-4 numerical integration method.

B.6 Objective function

Derivation of the likelihood term in the objective (left term in Eq. (5)):

$$\begin{aligned}\mathbb{E}_{q(Z,\theta_f,\tilde{\theta}_f,\tilde{z}_0|X)}\left[\log p(X|Z,\theta_f,\tilde{z}_0,\tilde{\theta}_f)\right] &= \mathbb{E}_{q(Z,\theta_f,\tilde{\theta}_f,\tilde{z}_0|X)}\left[\prod_{t=0}^{T-1}\log p(x_i|z_i)\right]\\ &= \sum_{t=0}^{T-1}\mathbb{E}_{q(Z,\theta_f,\tilde{\theta}_f,\tilde{z}_0|X)}\left[\log p(x_i|z_i)\right]\\ &= \sum_{t=0}^{T-1}\mathbb{E}_{q(z_i|x_i)}\left[\log p(x_i|z_i)\right].\end{aligned}$$

This follows since for $t' \neq t$: $x_t \perp\!\!\!\perp (z_{t'}, \theta_f, \tilde{\theta}_f, \tilde{z}_0)|z_t$.

Before decomposing the KL term, we note that the conditionals $p(z_0|\tilde{z}_0)$ and $p(\theta_f|\tilde{\theta}_f)$ are deterministic, meaning they are Dirac functions with the peak defined by Eqs. (3) and (4). The transition distribution $p(z_t|z_{t-1}, \theta_f)$ is also a Dirac function with the peak defined by Eq. (1) as stated in Section 3.

Therefore, The KL term from Eq. (5) can be written as:

$$KL\left[q(Z, \theta_f, \tilde{z}_0, \tilde{\theta}_f|X)||p(Z, \theta_f, \tilde{z}_0, \tilde{\theta}_f)\right] =$$
$$\int_Z \int_{\theta_f} \int_{\tilde{z}_0} \int_{\tilde{\theta}_f} q(\tilde{z}_0|X)q(\tilde{\theta}_f|X)q(z_0|\tilde{z}_0)q(\theta_f|\tilde{\theta}_f) \prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f)\cdot$$
$$\cdot \log\left[\frac{p(\tilde{z}_0)p(\tilde{\theta}_f)p(z_0|\tilde{z}_0)p(\theta_f|\tilde{\theta}_f)\prod_{t=1}^{T-1} p(z_t|z_{t-1}, \theta_f)}{q(\tilde{z}_0|X)q(\tilde{\theta}_f|X)q(z_0|\tilde{z}_0)q(\theta_f|\tilde{\theta}_f)\prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f)}\right].$$

The KL term we got, decomposes into the sum of 3 terms:

(i) The first term:

$$\int_Z \int_{\theta_f} \int_{\tilde{z}_0} \int_{\tilde{\theta}_f} q(\tilde{z}_0|X)q(\tilde{\theta}_f|X)q(z_0|\tilde{z}_0)q(\theta_f|\tilde{\theta}_f) \prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f) \log\left[\frac{p(\tilde{z}_0)\cancel{p(z_0|\tilde{z}_0)}}{q(\tilde{z}_0|X)\cancel{q(z_0|\tilde{z}_0)}}\right] =$$
$$\int_{\tilde{\theta}_f} q(\tilde{\theta}_f|X) \log\left[\frac{p(\tilde{z}_0)}{q(\tilde{z}_0|X)}\right] \int_Z \int_{\theta_f} \int_{\tilde{z}_0} q(\tilde{z}_0|X)q(z_0|\tilde{z}_0)q(\theta_f|\tilde{\theta}_f) \prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f) =$$
$$KL\left(q(\tilde{z}_0|X)||p(\tilde{z}_0)\right),$$

where $p(z_0|\tilde{z}_0) = q(z_0|\tilde{z}_0)$ by construction since both are determined exactly by Eq. (3).

(ii) In the same way, we get:

$$\int_Z \int_{\theta_f} \int_{\tilde{z}_0} \int_{\tilde{\theta}_f} q(\tilde{z}_0|X)q(\tilde{\theta}_f|X)q(z_0|\tilde{z}_0)q(\theta_f|\tilde{\theta}_f) \prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f) \log\left[\frac{p(\tilde{\theta}_f)p(\theta_f|\tilde{\theta}_f)}{q(\tilde{\theta}_f|X)q(\theta_f|\tilde{\theta}_f)}\right] =$$
$$KL\left(q(\tilde{\theta}_f|X)||p(\tilde{\theta}_f)\right),$$

where $p(\theta_f|\tilde{\theta}_f) = q(\theta_f|\tilde{\theta}_f)$ by construction since both are determined exactly by Eq. (4).

(iii) The last term:

$$\int_Z \int_{\theta_f} \int_{\tilde{z}_0} \int_{\tilde{\theta}_f} q(\tilde{z}_0|X)q(\tilde{\theta}_f|X)q(z_0|\tilde{z}_0)q(\theta_f|\tilde{\theta}_f) \prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f) \log\left[\frac{\prod_{t=1}^{T-1} p(z_t|z_{t-1}, \theta_f)}{\prod_{t=1}^{T-1} q(z_t|z_{t-1}, \theta_f)}\right] = 0,$$

where $p(z_t|z_{t-1}, \theta_f) = q(z_t|z_{t-1}, \theta_f)$ by construction since both are determined exactly by the ODE system f. Thus the logarithmic term equals 0.

C ALGORITHMS

The algorithms below give the training procedure for a single iteration and a batch of size 1. The extension to larger batch sizes is straightforward. The notations for X^i, Z^i and M^i are time sequences of length T for the observed signal, the observed latent signal and the grounding mask indicator (Eq. equation 6) respectively.

At inference time, we are given the signals X^i, Z^i, M^i of length T, but extrapolate to time $T + \tau$. Meaning, the ODE-solver for-loop is from time $t = 1$ to $t = T + \tau$, and the resulting sequences $\hat{X}^i$ and $\hat{Z}^i$ are of length $T + \tau$.

Algorithm 1 gives the training procedure for GOKU-net.

Algorithm 2 gives the training procedure for the Direct Identification (DI) baseline described in Section 4. In it, we first learn the parameters $\hat{\theta}_f^i$ and the initial state of the ODE $\hat{z}_0$ of every signal in the train and test sets. We then evaluate $\hat{Z}^i$ for the train set, and use these to learn the emission function $\hat{g}$, using the given train set signals X^i. In some cases, learning $\hat{z}_0^i$ was too difficult for the baseline so we tried a different approach. We used the learned parameters $\hat{\theta}_f^i$, the given ODE function f and the first observed latent vector $z_{t'}^i$ (meaning the mask $M^i(t') = 1$ and $M^i(t < t') = 0$), and used the ODE solver to calculate $\hat{z}_0^i$ backwards in time.

To address the unknown-unknowns task, we tested two algorithms: Algorithms 3 and 4. These algorithms are very similar to Algorithm 1, with the changes highlighted in blue. The main changes are that these methods also include an abstract part to the latent vectors which model the *Unknown Unknowns* part of the ODE. Algorithm 3 models the abstract part as a different trajectory with a different ODE which is modeled as a neural network such that $\hat{Z}^i = Z^{i,ODE} + Z^{i,abs}$. Algorithm 4 models the abstract part inside the ODE function itself. I.e., the ODE is changed to be: $\frac{dz_t}{dt} = f_{ODE}(z_t, \theta_f) + f_{abs}(z_t)$.

Algorithm Algorithm 5, describing the Latent-ODE+ (L-ODE+) baseline, follows the algorithm in Chen et al. (2018) while also grounding a set number of latent space dimensions to the observed one Z. We denoted the first D dimensions of the latent sequence of T time steps as $\hat{Z}_D^i \in \mathbb{R}^{T\times D}$, where D is the ODE dimension. The changes with respect toe Chen et al. (2018) are colored in blue.

Algorithm 1 GOKU-net

```
Input:
    1. sequence $X^i = (x_0, ..., x_{T-1})$
    2. ODE function $f$
    3. observed latent sequence $Z^i$ with mask $M^i$
    4. ODE solver
    5. hyper-parameters $\lambda_1$ and $\lambda_2$
initialize the neural nets $\phi_{\tilde{z}_0}^{enc}$, $\phi_{\tilde{\theta}_f}^{enc}$, $h_z$, $h_\theta$ and $\hat{g}$.
$[\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}] = \phi_{\tilde{z}_0}^{enc}(X^i)$,   $\tilde{z}_0 \sim \mathcal{N}(\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0})$,   $\hat{z}_0 = h_z(\tilde{z}_0)$
$[\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f}] = \phi_{\tilde{\theta}_f}^{enc}(X^i)$,   $\tilde{\theta}_f \sim \mathcal{N}(\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f})$,   $\hat{\theta}_f = h_z(\tilde{\theta}_f)$
for $t = 1, ..., T-1$ do
  $\hat{z}_t = ODEsolver(f, \hat{\theta}_f, \hat{z}_{t-1})$
end for
$\hat{X}^i = \hat{g}(\hat{Z}^i)$
ll_loss = $\mathcal{L}_{likelihood}(X^i, \hat{X}^i)$;   {see first term in Eq. (5)}
kl_loss = $\mathcal{L}_{kl}(\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}, \mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f})$;   {see second term in Eq. (5)}
grouding_loss = $\mathcal{L}_{ground}(Z^i, M^i, \hat{Z}^i)$;   {see Eq. (6)}
loss = ll_loss + $\lambda_1$ kl_loss + $\lambda_2$ grouding_loss
backpropagate(loss)
```

Algorithm 2 Direct Identification (DI)

```
Input:
    1. ODE function $f$
    2. ODE solver
    3. train and test sets of observed signals $X^i$, $Z^i$ and $M^i$.
for train and test sets do
  Initialize $\hat{\theta}_f, \hat{z}_0$
  for $t = 1, ..., T-1$ do
    $\hat{z}_t = ODEsolver(f, \hat{\theta}_f, \hat{z}_{t-1})$
  end for
  $loss = ||Z^i - \hat{Z}^i||_2$
  $\hat{\theta}_f := \hat{\theta}_f + \lambda \frac{\partial loss}{\partial \hat{\theta}_f}$        {backpropagate loss through the ODE solver}
  $\hat{z}_0 := \hat{z}_0 + \lambda \frac{\partial loss}{\partial \hat{z}_0}$
end for
for train set do
  $\hat{X}^i = \hat{g}(\hat{Z}^i)$
  $generative\_loss = ||X - \hat{X}^i||_2$
  backpropogate($generative\_loss$)
end for
```

Algorithm 3 GOKU with Unknown Unknowns - Z version

```
Input:
    1. sequence $X^i = (x_0, ..., x_{T-1})$
    2. ODE function $f$
    3. observed latent sequence $Z^i$ with mask $M^i$
    4. ODE solver
initialize the neural nets $\boxed{f_{abs}}$, $\phi^{enc}_{\tilde{z}_0}$, $\phi^{enc}_{\tilde{\theta}_f}$, $h_z$, $h_\theta$ and $\hat{g}$.
$[\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}] = \phi^{enc}_{\tilde{z}_0}(X^i)$,    $\tilde{z}_0 \sim \mathcal{N}(\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0})$,    $z_0 = h^{ODE}_z(\tilde{z}_0)$
$[\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f}] = \phi^{enc}_{\tilde{\theta}_f}(X^i)$,    $\tilde{\theta}_f \sim \mathcal{N}(\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f})$,    $\hat{\theta}_f = h^{ODE}_z(\tilde{\theta}_f)$
$\boxed{z^{abs}_0 = \phi_{abs}(z_0 || \hat{\theta}_f)}$
for $t = 1, ..., T-1$ do
  $z^{ODE}_t = ODEsolver(f, \hat{\theta}_f, z_{t-1})$
  $\boxed{z^{abs}_t = ODEsolver(f_{abs}, z^{abs}_{t-1})}$
end for
$\boxed{\hat{Z}^i = Z^{i,ODE} + Z^{i,abs}}$
$\hat{X}^i = \hat{g}(\hat{Z}^i)$
$loss = \mathcal{L}_{likelihood}(X^i, \hat{X}^i) + \lambda_1 \mathcal{L}_{kl}(\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}, \mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f}) + \lambda_2 \mathcal{L}_{ground}(Z^i, M^i, \hat{Z}^i)$        {see Eqs. (5) and (6)}
backpropagate($loss$)
```

Algorithm 4 GOKU with Unknown Unknowns - f version

```
Input:
    1. sequence $X^i = (x_0, ..., x_{T-1})$
    2. ODE function $f$
    3. observed latent sequence $Z^i$ with mask $M^i$
    4. ODE solver
initialize the neural nets $\boxed{f_{abs}}$, $\phi^{enc}_{\tilde{z}_0}$, $\phi^{enc}_{\tilde{\theta}_f}$, $h_z$, $h_\theta$ and $\hat{g}$.
$[\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}] = \phi^{enc}_{\tilde{z}_0}(X^i)$,    $\tilde{z}_0 \sim \mathcal{N}(\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0})$,    $z_0 = h^{ODE}_z(\tilde{z}_0)$
$[\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f}] = \phi^{enc}_{\tilde{\theta}_f}(X^i)$,    $\tilde{\theta}_f \sim \mathcal{N}(\mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f})$,    $\hat{\theta}_f = h^{ODE}_z(\tilde{\theta}_f)$
for $t = 1, ..., T-1$ do
  $z^{ODE}_t = ODEsolver(f + \boxed{f_{abs}}, \hat{\theta}_f, z_{t-1})$
end for
$\hat{X}^i = \hat{g}(\hat{Z}^i)$
$loss = \mathcal{L}_{likelihood}(X^i, \hat{X}^i) + \lambda_1 \mathcal{L}_{kl}(\mu_{\tilde{z}_0}, \sigma_{\tilde{z}_0}, \mu_{\tilde{\theta}_f}, \sigma_{\tilde{\theta}_f}) + \lambda_2 \mathcal{L}_{ground}(Z^i, M^i, \hat{Z}^i)$        {see Eqs. (5) and (6)}
backpropagate($loss$)
```

Algorithm 5 Grounded Latent ODE (L-ODE+)

```
**Input:**
        1. sequence $X^i = (x_0, ..., x_{T-1})$
        2. observed latent sequence $Z^i$ with mask $M^i$
        3. ODE solver
        4. true ODE dim $D$
initialize the neural nets $f_{abs}$, $\phi^{enc}$ and $\hat{g}$.
$[\mu_{z_0}, \sigma_{z_0}] = \phi^{enc}(X^i)$,        $\hat{z}_0 \sim \mathcal{N}(\mu_{z_0}, \sigma_{z_0})$
**for** $t = 1, ..., T-1$ **do**
    $\hat{z}_t^{ODE} = ODEsolver(f_{abs}, \hat{z}_{t-1})$
**end for**
$\hat{X}^i = \hat{g}(\hat{Z}^i)$
*ll_loss* = $\mathcal{L}_{likelihood}(X^i, \hat{X}^i)$
*kl_loss* = $\mathcal{L}_{kl}(\mu_{z_0}, \sigma_{z_0})$
$\hat{Z}^i_D = (\{\hat{z}_0\}_{d=0}^{D-1}, ..., \{\hat{z}_{T-1}\}_{d=0}^{D-1})$        {see text for explanation}
*grounding_loss* = $\mathcal{L}_{ground}(Z^i, M^i, \hat{Z}^i_D)$        {see Eq. (6)}
*loss* = *ll_loss* + $\lambda_1$ *kl_loss* + $\lambda_2$ *grouding_loss*
backpropagate(*loss*)
```

D EXPERIMENTS

In this section we provide details for the experiments section. We start by describing the baselines we used, and then provide dataset details for three domains: the two described in the text (single pendulum and cardiovascular system) and the Lotka-Volterra domain. Each dataset was randomly divided into train and test sets (90%, 10%).

D.1 BASELINES

Direct identification (DI) Separately for each sparse sequence Z^{observed} we infer θ_f and z_0 by setting the loss function to be Eq. (6), and using gradient decent through an ODE solver; details in the appendix. Since this method does not use the observations X, we assume, *only for the DI baseline*, that sparse Z observations are also available for the test set. In order to obtain estimates for the observations X under this baseline, we construct a training set where the instances are the $(\hat{Z}, \hat{\theta_f})$ inferred for each training sequence, and the labels are the corresponding observations X. We then learn a function $\hat{g}$ predicting X from $(\hat{Z}, \hat{\theta_f})$.

Data-driven For data-driven baselines with no input from mechanistic models we use (i) LSTM (Graves, 2013) and (ii) Latent-ODE (Chen et al., 2018), originally called Neural-ODE; we denote it L-ODE. These methods can only be used to extrapolate the given signal X for future time steps, since their latent space has no meaningful interpretation.

Data-driven with grounding Since in some of our experiments we assume sparse access to true latent Z in training, we created an extension of L-ODE (Chen et al., 2018) to this scenario. Assume $z(t) \in \mathbb{R}^k$. We designate k of the latent dimensions of the L-ODE model and ground them to the available Z^{observed} with the same loss term equation 6 as the DI baseline and GOKU-net. We call this baseline L-ODE+.

D.2 LOTKA-VOLTERRA

This domain was not described in the text and is brought here. The classic predator-prey model known as the Lotka-Volterra equations (LV), initially introduced by Lotka (1910) can be described mathematically as the following ODE:

$$\begin{aligned}\dot{z}_{\text{prey}}(t) &= \alpha z_{\text{prey}}(t) - \beta z_{\text{prey}}(t) z_{\text{predator}}(t)\\ \dot{z}_{\text{predator}}(t) &= -\gamma z_{\text{predator}}(t) + \delta z_{\text{prey}}(t) z_{\text{predator}}(t).\end{aligned}$$

The ODE parameters we wish to infer are therefore $\theta_{LV} = (\alpha, \beta, \gamma, \delta)$, and the ODE state we wish to infer is $z_t = (z_{\text{prey}}(t), z_{\text{predator}}(t))$.

Data Set We generated 10,000 sequences each of 100 time points with time step of $\Delta t = 0.05$. The four ODE parameters were uniformly sampled from $\theta_{LV} \sim U[1, 2]^4$, and initial ODE conditions uniformly sampled from $z_0 \sim U[1.5, 3]^2$. We then generated a sequence of 4-dimensional observations x_t using a non-linear, deterministic, and time-independent emission function g. The function g was created by randomly setting the weights of a neural network with 10-units hidden layer and ReLU activation function (the network's weights were initialized using `PyTorch`'s default weight initialization). The observations were additionally corrupted with white Gaussian noise with standard deviation of $\sigma_x = 0.01$. Our test set had sequences of length 400 each, where the first 100 time points were the test input, and the additional 300 time points were used only for evaluating the signals' extrapolation.

Evaluation and results The LV ODE parameters have some invariance. The stationary point of the LV ODE is known to be $(\frac{\alpha}{\beta}, \frac{\gamma}{\delta})$. We evaluate the error in estimating the stationary point over the test set:

$$E_{\theta_f}^{LV} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \sqrt{\left(\frac{\alpha_i}{\beta_i} - \frac{\hat{\alpha}_i}{\hat{\beta}_i}\right)^2 + \left(\frac{\gamma_i}{\delta_i} - \frac{\hat{\gamma}_i}{\hat{\delta}_i}\right)^2}.$$

We present the results in Table 3, and the X extrapolation results in Fig. 3. We see that in terms of identification, GOKU-net very clearly outperforms the baselines. However, it requires at least a small number of Z samples, and without them identification performance suffers. In terms of extrapolation GOKU-net also outperforms the baselines by a large margin. The data-driven approaches LSTM and L-ODE deteriorate quickly as the extrapolation goes further in time. The direct identification method completely failed in learning the mapping from Z to X: even though it had the correct neural network structure for g, it failed to learn good network weights. GOKU-net can use its access to the underlying ODE structure to extrapolates much more gracefully. This is true even when it has no access to the latent Z.

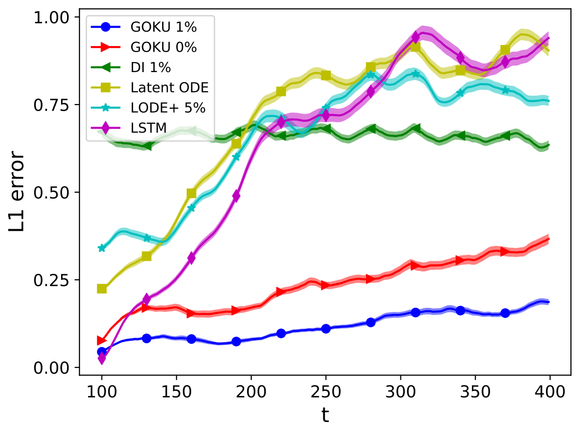


Figure 3: LV: mean extrapolation error for observations X over time steps after end of input sequence. Percentages in legend are percent grounding observation in training.

Method	5%	1%	0%
GOKU-net	$0.044 \pm .001$	$0.047 \pm .001$	$3.287 \pm .025$
DI	$0.006 \pm .001$	$0.192 \pm .006$	n/a

(a) θ_f identification using $E^{LV}_{\theta_f}$

Method	5%	1%	0%
GOKU-net	$0.068 \pm .001$	$0.077 \pm .001$	$5.158 \pm .046$
DI	$0.033 \pm .001$	$1.609 \pm .020$	n/a
L-ODE+	$0.548 \pm .008$	$0.769 \pm .008$	n/a

(b) Z identification

Method	5%	1%	0%
GOKU-net	$0.097 \pm .003$	$0.094 \pm .003$	$0.187 \pm .005$
DI	$0.119 \pm .002$	$0.644 \pm .004$	n/a
LSTM	n/a	n/a	$0.485 \pm .006$
L-ODE	n/a	n/a	$0.567 \pm .004$
L-ODE+	$0.541 \pm .005$	$0.552 \pm .005$	n/a

(c) X extrapolation

Table 3: Mean error across test samples of the Lotka-Volterra experiment, L_1 for Z and X, and $E^{LV}_{\theta_f}$ for θ_f . 5%, 1% and 0% indicate how many of the Z latent states were observed. DI is direct identification, L-ODE denotes Latent ODE (Chen et al., 2018), and L-ODE+ is the grounded version described in D.1. n/a are cases where the noted method is not applicable for the input. Methods not presented cannot perform the given task.

Algorithms implementation details In GOKU we first use an MLP with ReLU activation and 200 hidden units which results in a vector of dimension 64. For $\phi^{enc}_{\tilde{z}_0}$ we used an RNN with hidden dimension of 64 followed by a linear transformation to $\mu_{\tilde{z}_0}$ and another linear transformation to $\sigma_{\tilde{z}_0}$, both with dimension of 64. $\phi^{enc}_{\tilde{\theta}_f}$ is very similar to $\phi^{enc}_{\tilde{z}_0}$ with the small change of using a bi-directional LSTM instead of RNN. For the h function we used an MLP, with 200 hidden units and

ReLU activation. The final layer followed by a Softplus activation so that the θ_{LV} and z_0 would be physically feasible. The emission function we used is an MLP with ReLU activation and 200 hidden units. In Latent ODE, We used the same layers for the input to RNN network as GOKU, followed by an RNN which then transforms linearly to μ_{z_0} and σ_{z_0} with dimension 4. The ODE function f_{abs} is modeled as a neural network of sizes $4 \to 200 \to 200 \to 4$ with ReLU activation. The emission function is the same as GOKU. In LSTM, We used an LSTM with 4 layers and a hidden size of 128, followed by the same emiision function as GOKU. In DI we used the same emission function as in GOKU.

D.3 Single Pendulum From Pixels

Our second task is a model of a friction-less pendulum from an observed sequence of frames. We describe pendulums as non-linear oscillators obeying the following ODE:

$$\frac{d\theta(t)}{dt} = \omega(t), \qquad \frac{d\omega(t)}{dt} = -\frac{g}{l}\sin\theta(t). \tag{7}$$

We set the gravitational constant g to 10. The ODE has a single parameter which is the pendulum's length l, and the ODE state is $z_t = (\theta(t), \omega(t))$. This task is more challenging than the Lotka-Volterra one above, because of the complex emission function we used, as we explain next.

Data Set We followed Greydanus et al. (2019) and used the `Pendulum-v0` environment from OpenAI Gym (Brockman et al., 2016). For training we simulated 500 sequences of 50 time points, with time steps of $\Delta t = 0.05$. We generated the data as in Greydanus et al. (2019), with one important change: the ODE parameter l was uniformly sampled, $l \sim U[1, 2]$ instead of being constant, making the task much harder. As in Greydanus et al. (2019), we pre-processed the observed data such that each frame is of size 28×28. Each test set sequence is 100 time steps long, where the first 50 time steps are given as input, and the following 50 were used only for evaluating the signals extrapolation.

Evaluation and results In addition to the results given in Section 4.1, We give here the identification of $\hat{\theta}_f^i$ and $\hat{Z}$. Table 4 shows the identification error for both tasks.

Algorithms implementation details For all algorithms we used an input-to-rnn network and emission function exactly as suggested in Greydanus et al. (2019), composed of four fully-connected layers with ReLU activations and residual connections. The output of the input-to-rnn net dimension is 32.

In GOKU, The RNN and LSTM are implemented as in the LV experiment with output of dimension 16, followed by a linear transformation to μ_{z_0} and σ_{z_0} of dimension 16 as well. The h functions are implemented as in the LV, except that h_z output is linear without the softplus activation. In Latent ODE, The networks are implemented as in the LV experiment with the small change that the latent dimension is 16. In LSTM, We used an LSTM with 4 layers and a hidden size of 16, followed by the same emission function as GOKU. In HNN, We used the code provided by Greydanus et al. (2019). The only change made is that l is uniformly sampled instead of being constant.

Method	5%	1%	0%
GOKU-net	0.021 ± .005	0.028 ± .008	0.096 ± .009
DI	0.077 ± .029	0.511 ± .044	n/a

(a) θ_f identification

Method	5%	1%	0%
GOKU-net	0.072 ± .005	0.241 ± .016	2.417 ± .134
DI	0.092 ± .023	0.742 ± 0.076	n/a
L-ODE+	0.276 ± 0.013	0.840 ± 0.047	n/a

(b) Z identification

Table 4: Pixel-pendulum mean L_1 error across test samples with standard error of the mean of the pixel-pendulum experiment. Details as in Table 3. X extrapolation given in Figure 2a

D.4 Pixel Pendulum with Unknown Unknowns

In this experiment we aimed to show how GOKU can be modified to handle unknown unknowns in the ODE: We are given an ODE system that only partially describes the system that created the data. Specifically in this scenario, the pixel-pendulum data is created with a friction model:

$$\frac{d\theta(t)}{dt} = \omega(t), \quad \frac{d\omega(t)}{dt} = -\frac{g}{l}\sin\theta(t) - \frac{b}{m}\omega(t),$$

and we are only given with the friction-less ODE system in Eq. (7). We testes two models: the first one models the abstract latent trajectory $Z^{i,abs}$ and then evaluates $\hat{Z} = Z^{i,ODE} + Z^{i,abs}$ (Algorithm 3) and the second one models the time derivatives of the unknown part, making the ODE functional form as:

$$\frac{dz_t}{dt} = f_{ODE}(z_t, \theta_f) + f_{abs}(z_t),$$

where f_{abs} is modeled as a neural network (Algorithm 4).

Data set We created this data set in the same way as in the friction-less pixel-pendulum experiment. Here we set $l \sim U[1,2]$ as in the non-friction experiment, and we set in addition $m = 1$, $b = 0.7$. We tested this task with grounding mask rate = 5%.

Algorithm implementation details For GOKU-UU-Z (Algorithm 3) we modeled the ϕ_{abs} net as a fully connected network with 200 hidden units and ReLU activation. The network is fed with a 32-dimension concatenated vector of $\tilde{z}_0$ and $\tilde{\theta}_f$, and outputs a vector of size 2. The network's output is fed to an ODE solver with f_{abs} as the ODE, where f_{abs} is modeled as a fully connected network with $2 \rightarrow 200 \rightarrow 200 \rightarrow 2$ layers and ReLU activations. We then calculate the combined latent signal by:

$$\hat{Z}^i = Z^{i,ODE} + Z^{i,abs},$$

where $Z^{i,ODE}$ is the result of the known ODE, and $Z^{i,abs}$ is the result of the added unknown-unknowns model. For GOKU-UU-f (Algorithm 4), we only added a neural network that models Algorithm 4, which is implemented as a fully connected network with $2 \rightarrow 200 \rightarrow 200 \rightarrow 2$ layers and ReLU activations.

Results In Fig. 4 we show the X extrapolation error for future times for the two algorithms we test here: GOKU-UU-f and GOKU-UU-z (see appendix algorithms section for more details), and compare them to the baselines. To demonstrate the extrapolation of X, we randomly selected one test sample and show the pendulum's predicted angle for future times. In Fig. 5 we compare GOKU-UU-f and GOKU-UU-z and observe how GOKU-UU-f had near perfect results, where GOKU-UU-z could not learn the friction model. In Fig. 6 we compare GOKU-UU-f (which was much better than GOKU-UU-z) to the baselines for the same sample. In Fig. 7 we demonstrate that the f_{abs} of GOKU-UU-f learned only the friction part. We also tested if GOKU-UU methods could provide θ_f and Z identification, we summed the results in Table 5.

These results show that using GOKU-net with the unknown-unknowns-f modification can successfully identify the ODE parameters and variables, and extrapolate the observed signal, although it does not observe the full ODE functional form.

Method	θ_f identification	Z identification
GOKU-UU-f	0.057 ± .004	0.019 ± .002
GOKU-UU-z	0.140 ± .016	0.601 ± .041
GOKU	0.047 ± .005	0.359 ± .022
L-ODE+	n/a	0.101 ± .010

Table 5: θ_f and Z identification for the pixel pendulum with friction task

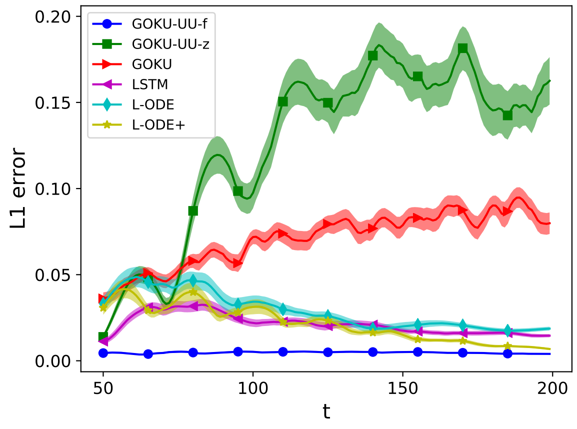


Figure 4: Pixel pendulum with friction - mean extrapolation error for observations X over time steps after end of input sequence.

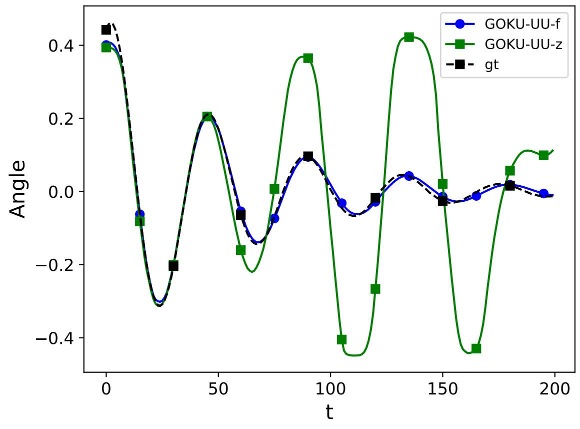


Figure 5: Pixel pendulum with friction predicted angle example. Comparing the two GOKU with unknown unknowns methods (GOKU-UU-f and GOKU-UU-z)

D.5 Cardiovascular System

Our last experiment uses the cardiovascular system (CVS) model suggested by Zenker et al. (2007) as stated in the Section 4.2. This ODE system is more involved than the ones above. The system is a simplified mechanistic model of the cardiovascular system: It is a multi-compartment model comprising the heart, the venous and the arterial subsystems together with a reflex loop component representing the nervous system control of blood pressure. Although it is far from comprehensive, this model can capture the prototypical behaviour of the cardiovascular system and its responses to pathological insults such as internal bleeding or septic shock that manifests as a reduction in peripheral vascular resistance. We implemented here a slightly modified version of the original Zenker ODE, using the following system:

$$\begin{aligned}\frac{dSV(t)}{dt} &= I_{external}\\ \frac{dP_a(t)}{dt} &= \frac{1}{C_a}\left(\frac{P_a(t)-P_v(t)}{R_{TPR}(S)} - SV\cdot f_{HR}(S)\right)\end{aligned}$$

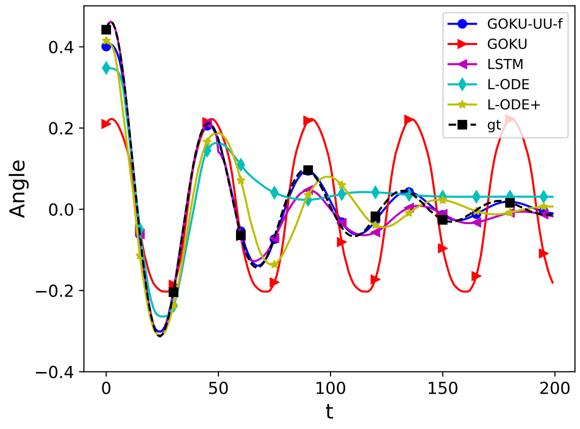


Figure 6: Pixel pendulum with friction predicted angle example. Comparing GOKU-UU-f (which was much better than GOKU-UU-z) with the baselines.

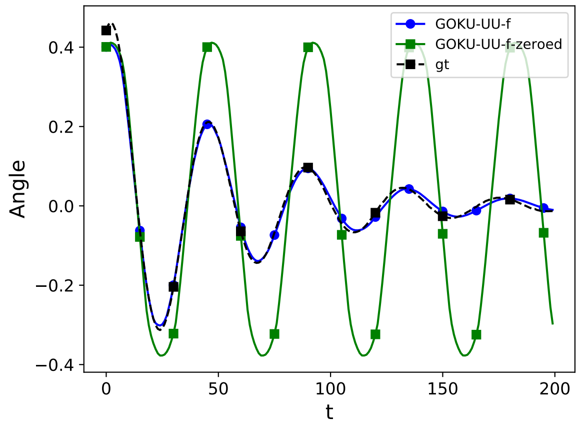


Figure 7: Pixel pendulum with friction predicted angle example. Here we demonstrate that zeroing the f_abs part of GOKU-UU-f, results in a friction-less signal.

$$
\begin{aligned}
\frac{dP_v(t)}{dt} &= \frac{1}{C_v}\left(-C_a\frac{dP_a(t)}{dt} + I_{external}\right) \\
\frac{dS(t)}{dt} &= \frac{1}{\tau_{Baro}}\left(1 - \frac{1}{1 + e^{-k_{width}(P_a(t) - P_{a_{set}})}} - S\right),
\end{aligned}
$$

where $R_{TPR}(S) = S(t)(R_{TPR_{Max}} - R_{TPR_{Min}}) + R_{TPR_{Min}} + R_{TPR_{Mod}}$, and $f_{HR}(S) = S(t)(f_{HR_{Max}} - f_{HR_{Min}}) + f_{HR_{Min}}$. In this model the variables have a directly interpretable mechanistic meaning: SV, Pa, Pv, S are respectively cardiac stroke volume (the amount of blood ejected by the heart), arterial blood pressure, venous blood pressure and autonomic barorelfex tone (the reflex responsible for adapting to perturbations in blood pressure, keeping homeostasis).

In this system we wish to find the ODE parameters: $\theta_{CVS} = (I_{external}, R_{TPR_{Mod}})$ since these are known-unknowns that describe recognized clinical conditions: $I_{external} < 0$ tells us that a patient is currently losing blood, and $R_{TPR_{Mod}} > 0$ tells us that their total peripheral resistance is getting lower, which is a condition of distributive shock as can be seen in sepsis for example. Both conditions can lead to an observed drop in blood-pressure. Discerning the relative contribution of each of the two to such a drop is very important clinically as often the underlying causes are not immediately clear and the choice of correct treatment relies on their accurate estimation. For clarity, in this example we consider the rest of the model parameters as known, setting them to the values stated in Zenker et al. (2007). The ODE state is $z_t = (SV(t), P_a(t), P_v(t), S(t))$. The observed state is the patient's vital signs and defined to be: $x_t = (P_a(t), P_v(t), f_{HR}(t))$. Note that some of the observed variables are the same as some of the latent variables, though with added noise as we explain now.

Data Set We simulated 1000 sequences of length 400, with time steps of $\Delta_t = 1$. The parameter $I_{external}$ was randomly sampled to be either -2 or 0, and the parameter $R_{TPR_{Mod}}$ was randomly sampled to be either 0.5 or 0. Initial ODE states uniformly sampled from $SV(0) \sim U[90, 100]$, $P_a(0) \sim U[75, 85]$, $P_v \sim [3, 7]$ and $S \sim [0.15, 0.25]$, where the intervals were set to the values given in Zenker et al. (2007). The observations were additionally corrupted with white Gaussian noise with standard deviation of $\sigma_x = 5$ for P_a, $\sigma_x = 0.5$ for P_v and $\sigma_x = 0.05$ for f_{HR} (standard deviation matches scale of the observed signal).

Algorithms implementation details For all algorithms we used an input-to-rnn network of 2 fully connected layers with ReLU activation and 64 hidden units, and output with dimension of 64.

In GOKU, The RNN and LSTM are implemented as in the LV experiment with output of dimension 64, followed by a linear transformation to μ_{z_0} and σ_{z_0} of dimension 64 as well. The h functions are implemented as in the LV, except that their output has a sigmoid activation layer, to bound them to a physically feasible solution. The emission function is a takes P_a and P_v from the latent trajectories, and a fully connected $4 \rightarrow 200 \rightarrow 1$ network with ReLU activation layer to compute f_{HR}. In Latent ODE the emission function is a fully connected $4 \rightarrow 200 \rightarrow 3$ network with ReLU activation layer. In LSTM, we used the same network as in the LV experiment.