Latent Ordinary Differential Equations for Irregularly-Sampled Time Series

Yulia Rubanova [1] Ricky T. Q. Chen [1] David Duvenaud [1]

Abstract

Time series with non-uniform intervals occur in many applications, and are difficult to model using standard recurrent neural networks (RNNs). In this paper, we generalize the hidden states of an RNN by parameterizing continuous-time dynamics of the latent state using a differential equation, which we call ODE-RNN. Furthermore, we refine the recently-proposed Latent Ordinary Differential Equation (Latent ODE) model by allowing this continuous-time model to condition on instantaneous observations. ODE-RNN and Latent ODE can naturally handle arbitrary time gaps between observations. Finally, we augment the Latent ODE model with a learned Poisson Process to enhance its prediction-like properties. We show experimentally that ODE-based models outperform RNN-based counterparts on irregularly-sampled data.

1. Introduction

Time series with irregular time intervals are difficult to model with recurrent neural networks, even though they have became increasingly popular for modeling sequential data. These models treat observations as a sequence of independent tokens and don't account for variable gaps between observations. A common approach is to discretize the timeline into equal non-overlapping intervals (Lipton et al., 2016; Marlin et al., 2012) and use heuristics such as taking the average of multiple observations in a given interval, or introducing a binary mask to indicate that no observations were made within the interval. Unfortunately, preprocessing the observations this way leads to loss of information about the density of events, which itself can be informative (e.g. "missing not at random"). A different approach is to continuously update the hidden state when there are no observations, usually modeled by a learned exponential decay(Che et al., 2018; Cao et al., 2018; Rajkomar et al., 2018; Mei & Eisner, 2017).

In this paper, we generalize RNNs with exponential decay to continuously changing hidden states between observations.

[1]University of Toronto, Vector Institute.

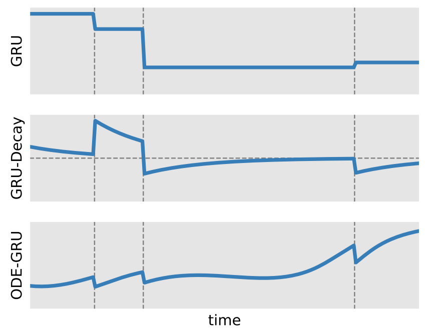


Figure 1: Comparing hidden states dymamics of different time-series models. *Top:* A standard GRU-RNN has constant or undefined hidden states between observations. *Middle:* The GRU-Decay model has states which exponentially decay towards zero. *Bottom:* Our ODE-GRU model has continuous-time hidden-state dynamics specified by a neural network.

This allows us to model time series without discretizing the timeline while also making predictions regarding future states when no observations are made. From an ordinary differential equations (ODE) point-of-view, we give the continuous model the ability to correct itself based on instantaneous interventions. The resulting model is a natural combination of ODE and RNN that has a notion of data missing-ness and can adapt to irregular time intervals.

We compare several RNN variants and hybrid ODE-RNNs in both one-step prediction and encoder-decoder settings, and find that ODE-RNNs can perform better when the data is sparse. We further investigate whether it is beneficial to model the density of events using Poisson Process likelihood along with the continuous hidden state.

2. Background

2.1. RNN with Exponential Decay

One natural way to handle missing data is incorporate the time gaps $\Delta t = t_i - t_{i-1}$ between the observed points into the update function for the hidden state. Many works have designed recurrent architectures that exponentially decay the hidden state towards zero when no observations are made. (Che et al., 2018; Cao et al., 2018; Rajkomar et al., 2018; Mozer et al., 2017). Thus, updates to the hidden state at the next observation time becomes

$$h_i = \text{RNNCell}(h_{i-1} \cdot \exp\{-\tau \Delta t\}, x_i) \quad (1)$$

where τ is a (possibly learned) decay rate parameter.

2.2. Neural Ordinary Differential Equations

Neural ODEs (Chen et al., 2018) model a time series using a continuous hidden state $h(t)$, where $h(t)$ is a solution of the following differential equation:

$$\begin{cases} \frac{dh(t)}{dt} = f(h(t), t) \\ h(t_0) = h_0 \end{cases} \quad (2)$$

The function f specifies the dynamics of the hidden state and is parameterized by a neural network. The ODE is solved using the call to a numerical ODE solver

$$h_0, \ldots, h_N = \text{ODESolve}(f, h_0, (t_0, \ldots, t_N)) \quad (3)$$

where $(t_0, \ldots, t_N)$ are discrete time points where $h(t)$ is evaluated, though note that the neural ODE models $h(t)$ as a continuous function over time.

Chen et al. (2018) proposed using this neural ODE as part of a variational autoencoder framework (Kingma & Welling, 2013) for learning sequence to sequence problems. In the generative/decoder model, h_0 is passed through an ODE solver to produce estimates of the target sequence. However, because $h(t)$ depends only on the initial state h_0 (3), $h(t)$ cannot be changed based on multiple observations. To get around this problem, Chen et al. (2018) resorted to a standard RNN for the recognition/encoder model.

3. Method

3.1. An ODE-RNN Hybrid

Following the discussion in (Mozer et al., 2017), we first note that an RNN with exponential decayed hidden state implicitly defines a continuous hidden state that follows the following ordinary differential equation:

$$\frac{dh(t)}{dt} = -\tau h \quad (4)$$

the solution of which is the pre-update term $h_0 \cdot \exp\{-\tau \Delta t\}$ in (1). This differential equation is time-invariant, assumes all dimensions decay with the same rate, and explicitly assumes that the stationary point (i.e. the zero valued state) is special.

Instead of continuously decaying the hidden state, we can model the hidden state using a neural ODE. This gives the model more flexibility in optimizing its behavior in the absence of observations. Intuitively, exponential decay can only model the loss of information whereas a learned dynamics can arbitrarily adapt to the lack of observations, which can be informative in itself.

Additionally, from the point-of-view of using an ODE to represent time series, we allow the hidden state to instantaneously change in a discontinuous manner when conditioned on discrete interventions. This can be done by using standard RNN update cells.

This hybrid algorithm is summarized in Algorithm 1. We use an ODE to perform the transition between the hidden states $h'_i = \text{ODESolve}(f, h_{i-1}, (t_{i-1}, t_i))$ and then update the hidden state using a standard RNN update $h_i = \text{RNNCell}(h'_i, x_i)$. Thus, the model provides a general form of transition function that does not make any *a priori* assumptions about the dynamics, and defines a continuous hidden state in-between the observation times.

Algorithm 1 ODE-RNN

```
**Input:** Data points $\{x_i\}_{i=1..N}$
$h_0$ = **0**
**for** i in 1..N **do**
    $h'_i = \text{ODESolve}(f, h_{i-1}, (t_{i-1}, t_i))$
    $h_i = \text{RNNCell}(h'_i, x_i)$
**end for**
$o_i = \text{OutputNN}(h_i)$ for all $i = 1..N$
**Return:** $\{o_i\}_{i=1..N}; h_N$
```

3.2. Autoregressive Modeling with the ODE-RNN

Consider a series of observations $\{x_i\}_{i=0,\ldots,N}$ and the corresponding time stamps $\{t_i\}_{i=0,\ldots,N}$ on interval $t \in [0, T]$. An autoregressive model makes a one-step-ahead prediction conditioned on the history of observations. This can be viewed as decomposing the joint density as a product of conditionals, ie. $p(x) = \prod_i p(x_i|x_{i-1} \ldots x_0)$. An immediate result of Algorithm 1 is the application of ODE-RNN to modeling the conditional distributions $p(x_i|x_{i-1} \ldots x_0)$., which we parameterize using the outputs of an ODE-RNN $\{o_i\}_{i=1..N}$.

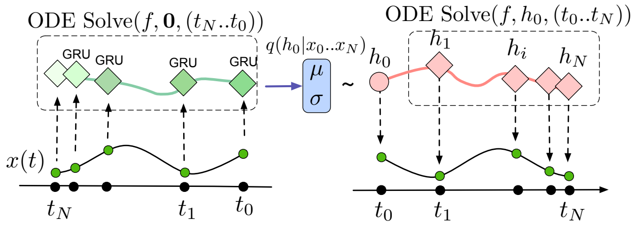


Figure 2: A Latent ODE model that uses an ODE-GRU as encoder and an ODE as decoder.

3.3. Sequence to Sequence Learning with Latent ODEs

As autoregressive models are only trained on one-step-ahead predictions, they don't have the ability to represent long-term dependencies and tend to memorize short-term effects instead of global trends. This is often sufficient for densely sampled data, but can be suboptimal when observations are sparse. This motivates encoder-decoder style architectures where a variable-length sequence is encoded into a fixed-dimensional embedding, which is then decoded into a second variable-length sequence (Sutskever et al., 2014).

Chen et al. (2018) proposed a variational autoencoder where the generative model is based on solving an ODE. However, they model the variational posterior using outputs from a RNN encoder. The necessity of using RNN encoder comes from the fact that (3) does not provide the ability to update the latent state conditioned on observations. Building on this encoder-decoder architecture, we enhance the encoder to have a continuously changing hidden state via the ODE-RNN architecture. This results in an fully ODE-based sequence-to-sequence model which we refer to as Latent ODE and is illustrated in Figure 2. The parameters of the latent distribution μ and σ are taken as the final hidden state of an ODE-RNN. We train Latent ODE models using a standard objective for latent variable models, the evidence lower bound (ELBO):

$$\begin{aligned}\mathbb{E}_{h_0 \sim q(h_0 | x_{t_0}, \ldots, x_{t_N})} \left[\log p(x_{t_0}, \ldots, x_{t_N}))\right] \\ - \mathcal{D}_{\text{KL}}[q(h_0 | x_{t_0}, \ldots, x_{t_N}) || p(h_0)]\end{aligned} \tag{5}$$

The stochastic latent state in Latent ODE provides a number of desirable properties. The latent state y_0 represents a summary of a discrete number of the encoded trajectory while maintaining uncertainty in the underlying continuous trajectory. Posterior inference allows to sample multiple possible continuous trajectories conditioned on the observations.

3.4. Poisson process likelihoods

The fact that a measurement was made at a particular time is often informative about the state of the system (Che et al., 2018). In the ODE framework, we can use the continuous latent state to parameterize the intensity of events using *inhomogeneous* Poisson point processes (Palm, 1943), which has the following log-likelihood.

$$\log p(t_0, \ldots, t_N | \lambda(t)) = \sum_{i=1}^{N} \log \lambda(t_i) - \int_{t_0}^{t_N} \lambda(t)\, dt \tag{6}$$

Where the rate $\lambda(t)$ is often approximated through discrete steps (Mei & Eisner, 2017). However, this approach makes the precise evaluation of $\int_{t_{\text{start}}}^{t_{\text{end}}} \lambda(t) dt$ a hard task. Using an ODE framework, we can evaluate both the latent trajectory and the rate $\lambda(t)$ as a continuous function.

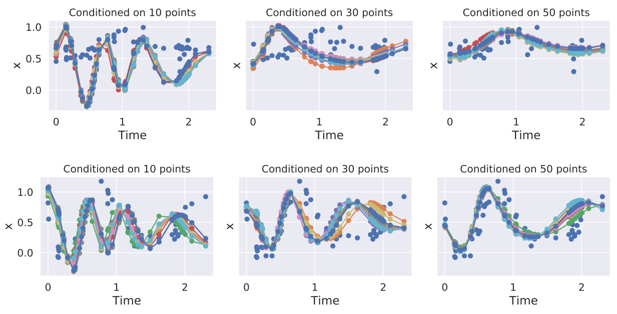


Figure 3: Predictions from Latent ODE when conditioned on a small subset of points. Though the model was only trained with exactly 30 observations, its behavior in the sparse-data setting correctly recovers the overall sinusoidal behavior of the dataset and correctly recovers the full time series when more observations are provided.

We augment the Latent ODE framework with a Poisson Process over the observation times, where we parameterize $\lambda(t)$ as a function of $h(t)$. The joint generative model is specified as $p(h_0)p(t_0, \ldots, t_N | h_0) \prod_{i=0}^{N} p(x_i | h_0)$, where the distributions are specified below.

$$p(h_0) = \text{Normal}\big(h_0; 0, I\big) \tag{7}$$

$$\{h(t_i)\} \leftarrow \text{ODESolve}\big(f, h_0, (t_0, \ldots, t_N)\big) \tag{8}$$

$$p(t_0 ... t_N | h_0) = \text{PoissonProcess}\big(t_1 ... t_N; \lambda(h(t))\big) \tag{9}$$

$$p(x_i | h_0) = \text{Normal}\big(x_i; \mu(h(t_i)), \sigma(h(t_i))\big) \tag{10}$$

We can use a single call to the ODE solver (8) to get both the Poisson intensity and observed values. Computing $\int \lambda(t) dt$ is also done as part of the ODE solving.

3.5. Different Observation Times & Sparse Features

With irregular time intervals, the time stamps can be different for every training example, and therefore are hard to batch. We solve this issue by using the union of time stamps from all the time series in the batch and solve an ODE over the union of time points. This does not significantly increase the time complexity of the ODE solver, as the adaptive time stepping in ODE solvers is independent of the evaluation time intervals. We use masks to ensure we only compute the loss on observed data, which may include masking both time values and features, similar to Che et al. (2018).

4. Experiments

The ODE-RNN model can be used with any hidden state update formula for the RNNCell function in Algorithm 1. Throughout our experiments, we use the Gated Recurrent Unit (GRU) (Cho et al., 2014) update equation.

Table 1: Test Mean Squared Error (MSE) on the MuJoCo dataset. "–" indicates an incomplete experiment.

		Interpolation (% Observed Pts.)				Extrapolation (% Observed Pts.)			
	Model	10%	20%	30%	50%	10%	20%	30%	50%
Autoreg	RNN Δt	$2.45{\cdot}10^{-2}$	$1.71{\cdot}10^{-2}$	$1.25{\cdot}10^{-2}$	$0.79{\cdot}10^{-2}$	$\mathbf{7.26{\cdot}10^{-2}}$	$\mathbf{6.79{\cdot}10^{-2}}$	$\mathbf{6.59{\cdot}10^{-2}}$	$3.06{\cdot}10^{-1}$
	RNN GRU-D	–	$1.37{\cdot}10^{-2}$	$1.10{\cdot}10^{-2}$	–	$8.86{\cdot}10^{-3}$	–	–	–
	ODE-GRU (Ours)	$\mathbf{1.65{\cdot}10^{-2}}$	$\mathbf{1.21{\cdot}10^{-2}}$	$\mathbf{9.86{\cdot}10^{-3}}$	$\mathbf{6.65{\cdot}10^{-4}}$	$1.35{\cdot}10^{-1}$	$3.20{\cdot}10^{-1}$	$1.55{\cdot}10^{-1}$	$\mathbf{2.65{\cdot}10^{-1}}$
Enc-Dec	RNN-VAE	$6.51{\cdot}10^{-2}$	$6.41{\cdot}10^{-2}$	$6.31{\cdot}10^{-2}$	$6.10{\cdot}10^{-2}$	$2.38{\cdot}10^{-2}$	$2.14{\cdot}10^{-2}$	$2.02{\cdot}10^{-2}$	$1.78{\cdot}10^{-2}$
	Latent ODE (RNN enc.)	$2.48{\cdot}10^{-2}$	$5.78{\cdot}10^{-3}$	$2.77{\cdot}10^{-2}$	$4.47{\cdot}10^{-3}$	$1.66{\cdot}10^{-2}$	$1.65{\cdot}10^{-2}$	$1.48{\cdot}10^{-2}$	$1.38{\cdot}10^{-2}$
	Latent ODE (Ours)	$\mathbf{3.60{\cdot}10^{-3}}$	$\mathbf{2.95{\cdot}10^{-3}}$	$\mathbf{3.00{\cdot}10^{-3}}$	$\mathbf{2.85{\cdot}10^{-3}}$	$\mathbf{1.44{\cdot}10^{-2}}$	$\mathbf{1.40{\cdot}10^{-2}}$	$\mathbf{1.18{\cdot}10^{-2}}$	$\mathbf{1.26{\cdot}10^{-2}}$

4.1. Baselines

We compare our ODE-based models to the RNN-based equivalents with different transition functions. In the class of autoregressive models, we compare ODE-GRU to the standard RNN. Among the encoder-decoder models, we compare Latent ODE to the RNN variant where both the encoder and decoder are recurrent neural nets. Where applicable, we compare against RNNs with exponential decay.

4.2. MuJoCo: Physics Simulation

We created a physical simulation using "Hopper" model from Deepmind Control Suite (Tassa et al., 2018). We randomly sample the initial conditions of the hopper so that the hopper is located above the ground with random initial velocities. During the simulation the hopper rotates and falls on the ground. We generated 10,000 sequences of 100 time points, and randomly split the data into 80% for training and 20% for test.

We construct interpolation and extrapolation tasks, where during training we always subsample a small percentage of the points, to test the behavior of models under sparse observations. For evaluation, we compute the mean squared error (MSE) on the full time series without subsampling. Therefore, the models must learn to impute missing data in addition to the task they are explicitly trained on.

Our ODE-GRU model outperforms autoregressive RNNs on interpolation but have worse performance for extrapolation. However, we don't expect autoregressive models to extrapolate very well as they were only trained for one-step-ahead prediction. On the other hand, encoder-decoder architectures have much better extrapolation performance. We find that Latent ODEs can easily outperform standard RNN-VAEs on both interpolation and extrapolation.

4.3. Physionet

We evaluate our approach on the PhysioNet Challenge 2012 (Silva et al., 2012) dataset of 8000 patients from Intensive Care Unit (ICU). This dataset covers the first 48 hours after the patient's admission to ICU. There are 37 measurement types in total. Different patients take different tests during their stay in the hosital. As a result, the measurements are sparse and irregularly sampled, and the observation times for different measurements are not aligned. Most works use a very coarse discretization of aggregated measurements per hour (Che et al., 2018) but for our experiments, we use a finer grid of six minutes. Tables 2 and 3 show that ODE-enhanced models perform better than RNN baselines.

As the measurements are extremely sparse, we regularize the model by fitting the rate of events in Physionet dataset using a Poisson Process jointly trained with the Latent ODE model.

As shown in table 3, modelling the intensity rate of events together with improves the mean squared error on both interpolation and extrapolation tasks. Examples of fitted Poisson rate are shown in the supplement. The learned rates vary for different measurement types and, for some measurements, even reflect sharp changes in the event density.

Table 2: Test MSE on PhysioNet. Autoregressive models.

Model	Interpolation ($\cdot 10^{-3}$)
RNN Δt	8.38
RNN (imputed input)	10.04
RNN (hidden state decay)	8.45
RNN GRU-D	7.53
ODE-GRU (Ours)	**5.01**

Table 3: Test MSE on PhysioNet. Encoder-decoder models.

Model	Interpolation	Extrapolation
RNN-VAE	$9.35{\cdot}10^{-3}$	$7.19{\cdot}10^{-3}$
Latent ODE (RNN enc.)	$8.12{\cdot}10^{-3}$	$6.32{\cdot}10^{-3}$
Latent ODE	$8.75{\cdot}10^{-3}$	$6.33{\cdot}10^{-3}$
Latent ODE + Poisson	$\mathbf{7.90{\cdot}10^{-3}}$	$\mathbf{6.32{\cdot}10^{-3}}$

References

Cao, Wei, Wang, Dong, Li, Jian, Zhou, Hao, Li, Lei, and et al. Brits: Bidirectional recurrent imputation for time series, May 2018. URL https://arxiv.org/abs/1805.10572.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, 8(1):6085, 2018. ISSN 2045-2322. doi: 10.1038/s41598-018-24271-9.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. Curran Associates, Inc., 2018.

Cho, Kyunghyun, Merrienboer, v., Bart, and Yoshua. On the properties of neural machine translation: Encoder-decoder approaches, Oct 2014. URL https://arxiv.org/abs/1409.1259.

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

Lipton, Z. C., Kale, D., and Wetzel, R. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In Doshi-Velez, F., Fackler, J., Kale, D., Wallace, B., and Wiens, J. (eds.), *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pp. 253–270, Children's Hospital LA, Los Angeles, CA, USA, 18–19 Aug 2016. PMLR.

Marlin, B. M., Kale, D. C., Khemani, R. G., and Wetzel, R. C. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2Nd ACM SIGHIT International Health Informatics Symposium*, IHI '12, pp. 389–398, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0781-9. doi: 10.1145/2110363.2110408.

Mei, H. and Eisner, J. M. The neural hawkes process: A neurally self-modulating multivariate point process. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6754–6764. Curran Associates, Inc., 2017.

Mozer, C., M., Kazakov, Denis, Lindsey, and V., R. Discrete event, continuous time rnns, Oct 2017. URL https://arxiv.org/abs/1710.04110.

Palm, C. Intensitätsschwankungen im fernsprechverker. *Ericsson Technics*, 1943.

Rajkomar, A., Oren, E., Chen, K., M. Dai, A., Hajaj, N., J. Liu, P., Liu, X., Sun, M., Sundberg, P., Yee, H., Zhang, K., Duggan, G., Flores, G., Hardt, M., Irvine, J., Le, Q., Litsch, K., Marcus, J., Mossin, A., and Dean, J. Scalable and accurate deep learning for electronic health records. *npj Digital Medicine*, 1, 01 2018. doi: 10.1038/s41746-018-0029-1.

Silva, I., Moody, G., Scott, D. J., Celi, L. A., and Mark, R. G. Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012. *Computing in cardiology*, 39:245–248, 2012. ISSN 2325-8861. URL https://www.ncbi.nlm.nih.gov/pubmed/24678516https://www.ncbi.nlm.nih.gov/pmc/PMC3965265/.

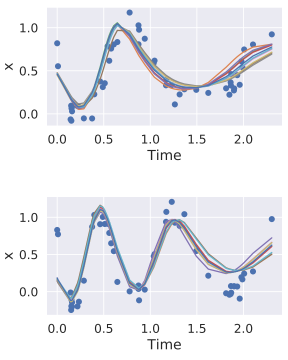
Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind Control Suite. *arXiv e-prints*, art. arXiv:1801.00690, Jan 2018.

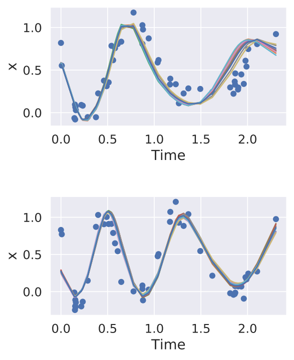
Supplementary tables and figures

Table 4: Mean squared error on the toy dataset

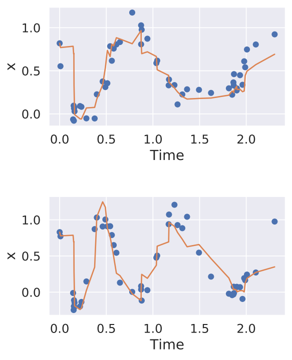
		Interpolation				Extrapolation			
	% observed points	10	20	30	50	10	20	30	50
Autoreg	RNN Δt	0.06081	0.04680	0.05822	0.04116	0.06172	0.06115	0.06891	0.05617
	RNN (imputed input)	0.08558	0.06043	0.03922	0.04116	**0.06095**	0.07212	**0.06541**	**0.05049**
	RNN (hidden state decay)	1.65891	0.05344	0.04974	0.03275	0.06172	0.06115	0.06891	0.05617
	RNN GRU-D	2.35628	0.05997	0.04832	0.04116	**0.06095**	0.07212	**0.06541**	**0.05049**
	ODE-GRU	**0.05150**	**0.03211**	**0.02643**	**0.01666**	0.06592	**0.04774**	0.10940	0.08000
VAE	RNN-VAE	0.07352	0.07346	0.07323	0.07304	0.20107	0.03710	0.07281	0.02871
	Latent ODE (RNN enc)	**0.06860**	0.06764	**0.02754**	0.05721	**0.04920**	0.04807	**0.01788**	0.02703
	Latent ODE (ODE enc)	0.07133	**0.03144**	0.05354	**0.01717**	0.05313	**0.04427**	0.03572	**0.01388**



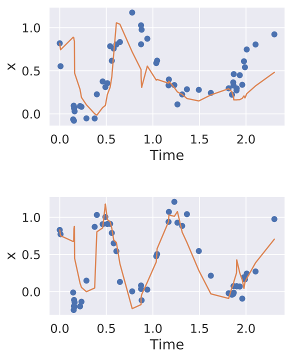
(a) ODE-VAE (ODE enc)



(b) ODE-VAE (RNN enc)

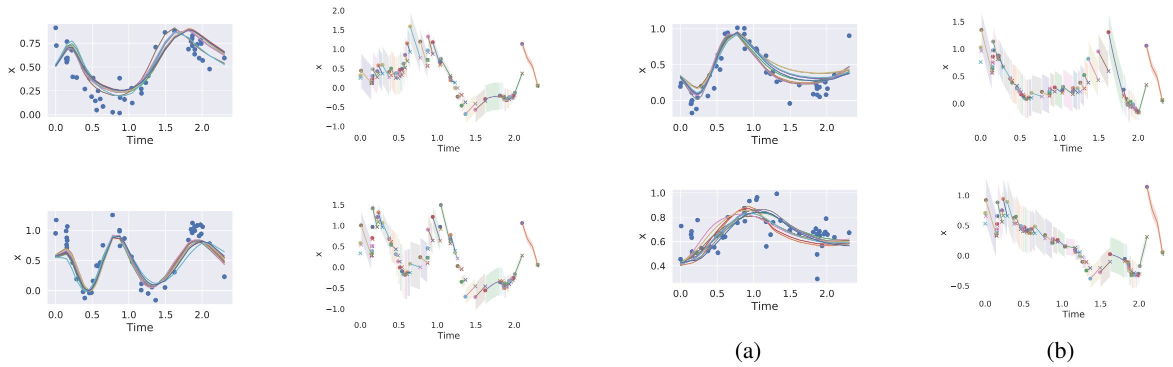


(c) ODE-GRU



(d) Standard RNN

Figure 4: Reconstructions on toy dataset



(a) (b)

Figure 5: **(a)** Reconstructions on toy dataset from ODE-VAE. Observations are shown as points. Lines are reconstructions for different samples of y_0 in ODE-VAE model. **(b)** Corresponding latent state in the recognition model (first dimension). The recognition model encodes the data backwards in time (from right to left). The lines show latent ODE path in-between the encoded data points. The discontinuities between the paths show the update of the latent state using the observation at that time point. The end of the end of the ODE path from the previous observation is shown as a small circle. The updated state (and the start a new ODE path) is shown as cross. The shaded area shows the predicted standard deviation for the initial latent state y_0. Notice that the right-most point is similar for all four trajectories – only one data point was encoded, which does not contain much information about the trajectory. As the encoding progresses (from right to left), the latent updated generally become smaller.

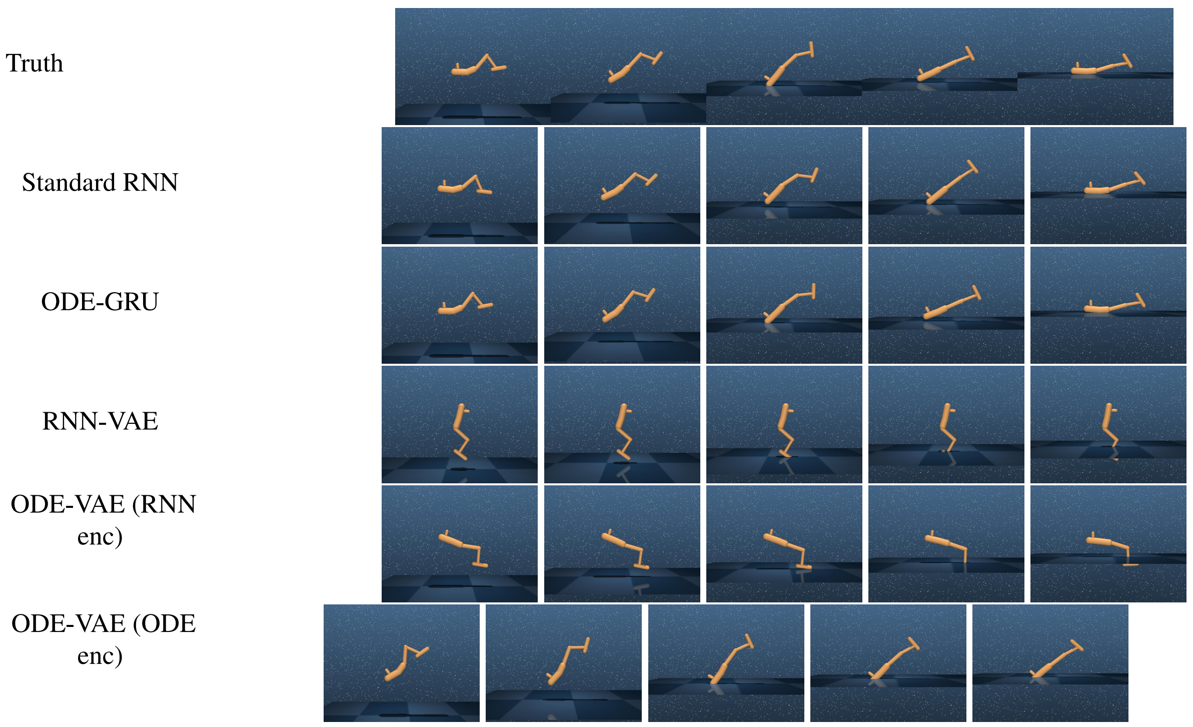


Figure 6: Reconstructed trajectories on Mujoco dataset

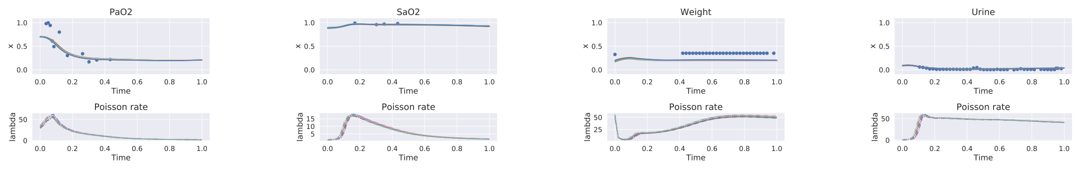


Figure 7: Examples of learned poisson rate for different features in Physionet.