Continuous-Time Neural Filters for Dynamical Prediction

Albert Li, Brent Yi

I. Introduction

A. Motivation

Effective human-robot or robot-robot interactions depend on guarantees of safety and co-agent trust. Ensuring these characteristics in highly-dynamical environments necessitates accurate predictions over meaningful time horizons. For well-understood mechanical systems, this task is equivalent to developing a suitable dynamical model using, for example, Lagrangian dynamics. Techniques like model predictive control have deployed these prediction models with great success, but many dynamical systems either cannot be modeled with physics-based approaches or are governed by physical or non-physical interactions that are too complex to analytically model, motivating data-driven prediction.

To that end, deep neural networks have enjoyed significant usage for probabilistic prediction modeling. However, most neural networks are designed as discrete-time models, whereas many time series data are naturally conceived of as trajectories in continuous time. Additionally, many concepts in nonlinear control theory are developed for the analysis of continuous-time systems [1].

This paper investigates the marriage of deep neural networks and continuous-time spatiotemporal learning to synthesize expressive nonlinear stochastic prediction models. To do this, we propose a network architecture that learns a Bayesian filter jointly composed of a *dynamics network* and an *observation network*. The deep filter can observe on data, yielding a filtered latent trajectory that can be passed back through the observation model to reconstruct the observations, allowing us to compute a reconstruction loss and jointly train the models using any optimization-based approach. Our approach is evaluated on the Van der Pol (VDP) and pendulum dynamical systems to allow for accurate ground truth comparisons.

B. Related Work

For time series forecasting, recurrent neural networks (RNNs) and their variants have long been a standard choice. RNNs maintain a set of deterministic hidden states which act as a proxy (up to a nonlinear transformation) for the prediction belief and are updated by observing on sequential inputs. Examples of the application of recurrent networks include predictions on human trajectories [2], financial activity [3], and system identification [4]. Recent work such as [5] uses common recurrent architectures in conjunction with deep Kalman filters to learn probabilistic graphical models for Bayesian counterfactual inference. Similarly, particle filters have been adapted for modeling multimodal probabilistic outcomes nonparametrically with RNNs [6].

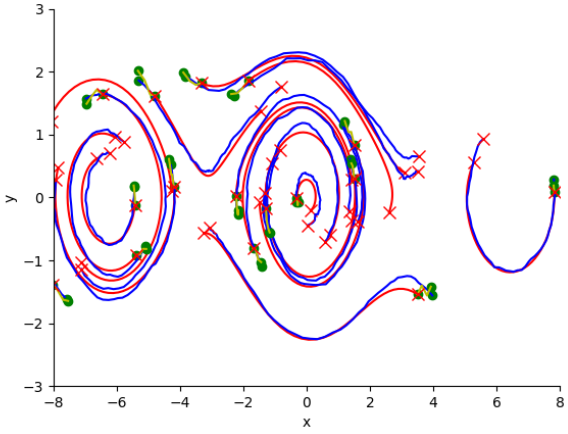

Fig. 1: An example reference vector field (blue) and the learned model (red). Bayesian filtering techniques can be effective in helping facilitate efficient and accurate dynamics learning. Maintaining a continuous model also has many benefits, like constant memory scaling, infinite resolution data imputation, and compatibility with continuous-time control theory.

A plethora of methods have emerged leveraging the expressive capacity of generative models for the specific task of learning physical state space models from highly nonlinear spatiotemporal data [7], [8], [9]. Most often, Bayesian models of these dynamical systems are latent variable models, in which the observed data are conditioned on hidden latent variables. This is a natural setting for deep dynamical prediction, since the predictor can compress the high-dimensional observed data into a low-dimensional latent dynamical model governing the evolution of the temporal sequence.

The development of neural ordinary differential equations (ODEs), which instead parameterize the derivative of a continuous function, do away with the limitations of standard discrete-time prediction models [10]. The inputs into this network are the initial latent conditions of a dynamical system and the output is the state of the system at some later time. To accomplish this, the forward pass of this network is executed using any differential equation solver and remarkably, backpropagation can also be executed via a time-reversed differential equation solver call using the adjoint sensitivity method.

The benefits of neural ODEs include constant-time memory complexity as well as the freedom for the designer to not only choose a solver suited for the specific data, but also to adjust the solver settings as necessitated by context. This means, for example, that a high step resolution can be enforced during training to learn a fine representation of

the underlying vector field while a loose tolerance can be used during runtime to improve computational speed while leveraging the accuracy of the learned parameters.

This is especially well-suited for dynamical control. Learning the vector field allows infinite resolution imputation, which could be useful in cases when the agent desires a finer prediction resolution to improve safety or performance in control (for example, if a robot enters a crowded space). Learning can also be done even when data are irregularly sampled or missing, which is difficult to handle in standard learning-based methods. Finally, for safe predictive control, formal methods like control barrier functions, which assume access to a continuous-time dynamical model, benefit greatly from a compatible expressive prediction model [11].

II. Preliminaries

A. Discrete-Time Bayesian Filtering

Consider the following discrete-time model with additive noise drawn from some arbitrary distribution:

$$\begin{aligned} z_{t+1} &= f(t, z_t) + w_t \\ y_t &= g(z_t) + v_t. \end{aligned} \tag{1}$$

The discrete-time Bayesian filtering problem consists of recursively computing a distribution over the current state given a history of observations:

$$p(z_t \mid y_{1:t}), \tag{2}$$

where the notation $y_{1:t}$ indicates a sequence of discrete observations. To recover this distribution, assume a distribution over the predicted state given only prior measurements:

$$p(z_t \mid y_{1:t-1}). \tag{3}$$

Then, an application of Bayes' rule allows the prediction to be updated with the most recent measurement:

$$p(z_t \mid y_{1:t}) = \frac{p(y_t \mid z_t)p(x_t \mid y_{1:t-1})}{\int_{z_t'} p(y_t \mid z_t')p(z_t' \mid y_{1:t-1})dz_t'}, \tag{4}$$

where the prior distribution $p(z_t \mid y_{1:t-1})$ is the most recently computed posterior. Though this computation is generally intractable, the Kalman filter allows the analytical recovery of the estimated state of a linear Gaussian system with white noise [12]. Let the dynamics be written now as

$$\begin{aligned} z_{t+1} &= Az_t + w_t, \\ y_t &= Cz_t + v_t, \end{aligned} \tag{5}$$

where $w_t \sim \mathcal{N}(0, Q_t)$ and $v_t \sim \mathcal{N}(0, R_t)$. The Kalman filter maintains a belief over the true state represented by a mean and covariance, which are sufficient to fully describe a Gaussian distribution. The analytical prediction and update equations are written

$$\begin{aligned} \mu^z_{t|t-1} &= A\mu^z_{t-1|t-1}, \\ \Sigma^z_{t|t-1} &= A_{t-1}\Sigma^z_{t-1|t-1}A^\top_{t-1} + Q_t, \\ \mu^z_{t|t} &= \mu^z_{t|t-1} + K_t(y_t - C_t\mu^z_{t|t-1}), \\ \Sigma^z_{t|t} &= \Sigma^z_{t|t-1} - K_tC_t\Sigma^z_{t|t-1}, \\ K_t &= \Sigma^z_{t|t-1}C^\top(C_t\Sigma^z_{t|t-1}C_t^\top + R_t)^{-1}, \end{aligned} \tag{6}$$

where for an arbitrary variable x, $x_{t|t-1}$ indicates the predicted value of x at time t given measurements from times $t-1$ and earlier, while $x_{t|t}$ indicates the updated value of x after taking into consideration the most recent measurement from time t. The filter is initialized with some prior belief over the initial state, $p(z_0)$.

For general nonlinear systems, the filtering procedure can at best be made approximate. Many techniques exist, but the most common one (and the one considered in this paper) is the extended Kalman filter (EKF), wherein the dynamics are linearized at each step in order to allow the use of the standard Kalman filter equations.

B. Continuous-Discrete Gaussian Bayesian Filtering

Now, consider instead the following *continuous-discrete* model with additive noise:

$$\begin{aligned} \dot{z}(t) &= f(t, z(t)) + w(t), \\ y_k &= g(z(t_k)) + v(t_k), \end{aligned} \tag{7}$$

where $w(t)$ and $v(t)$ are white noise variables drawn from some distribution and t_k indicates the time t corresponding to some discrete index k. This model features continuous dynamics with intermittent discrete measurements. For any digital system (and for data with which we seek to design models), measurements are received discretely. However, it is still possible and often desirable to maintain a continuous dynamical model, which motivates the above formulation.

Approximating the posterior as a Gaussian, we can construct the mean and covariance dynamics, which are then integrated between the times when measurements are received. When new data arrives, the same discrete update step as before is executed. The continuous prediction dynamics for this model are derived in [13] as:

$$\begin{aligned} \dot{\mu}^z(t) &= \mathbb{E}[f(t, z(t))], \\ \dot{\Sigma}^z(t) &= \mathbb{E}[(z(t) - \mu^z(t))f^\top(t, z(t))] \\ &\quad + \mathbb{E}[f(t, z(t))(z(t) - \mu^z(t))^\top] + \mathbb{E}[\Sigma^z(t)]. \end{aligned} \tag{8}$$

If the dynamics are differentiable, (8) reduces to:

$$\begin{aligned} \dot{\mu}^z(t) &= A(t)\mu^z(t), \\ \dot{\Sigma}^z(t) &= A(t)\Sigma^z(t) + \Sigma^z(t)A^\top(t) + Q(t), \end{aligned} \tag{9}$$

where $A(t)$ represents the Jacobian of $f(t, z(t))$.

C. Bayesian Smoothing

While a Bayesian filter computes the posterior of a state conditioned on past and present observations, it is possible to recover a better state estimate by leveraging *future* observations to refine previously computed estimates. This process is known as Bayesian smoothing, which recovers the maximum a posteriori (MAP) estimate of the state [13].

The discrete procedure for Kalman smoothing is as follows: first, execute the discrete Kalman filter until some final time T. During the forward pass, the predicted and updated values of the belief distribution parameters are cached. Then, a backwards pass is executed beginning at time T wherein each successively older estimate is recursively *smoothed* by

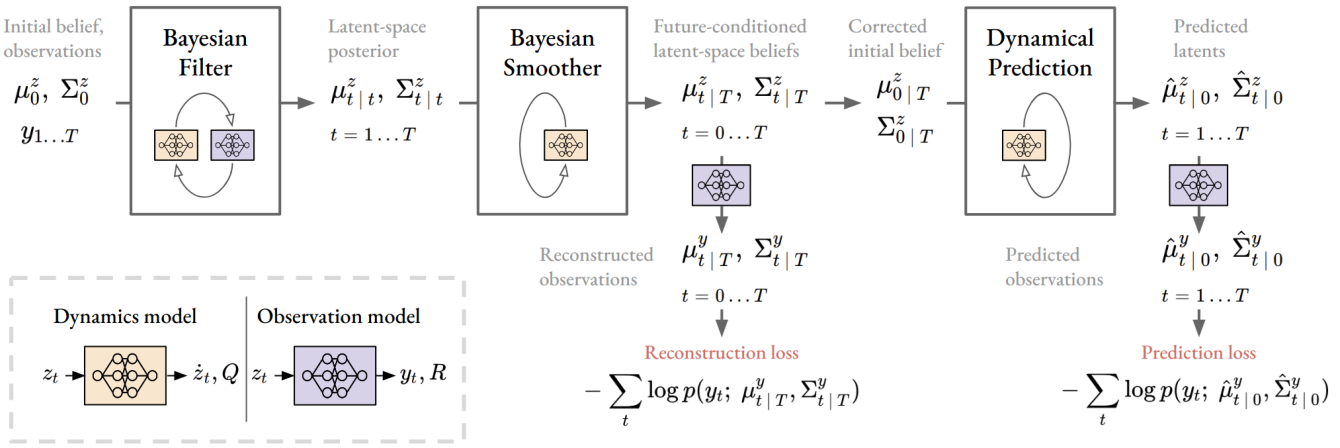

Fig. 2: The architecture of the continuous-time neural filter. Note the difference between the two terms composing the joint loss.

conditioning it on data from the future. These steps may be summarized with the following equations:

$$\begin{aligned}\mu^z_{t|T} &= \mu^z_{t|t} + K^s_t(\mu^z_{t+1|T} - \mu^z_{t+1|t}),\\ \Sigma^z_{t|T} &= \Sigma^z_{t|t} + K^s_t(\Sigma^z_{t+1|T} - \Sigma^z_{t+1|t})(K^s_t)^\top,\\ K^s_t &= \Sigma^z_{t|t}A_t^\top(\Sigma^z_{t+1|t})^{-1}.\end{aligned} \tag{10}$$

The notation $x_{t|T}$ indicates a variable conditioned on all measurements up to time T. Again, A represents the linearization of the dynamics for nonlinear systems.

Like in filtering, there is an analogous continuous-discrete Bayesian smoother derived in [13], the details of which are summarized in the remainder of this section. Because the measurements are discrete, the smoothing updates are also discrete. However, the forward pass requires the computation of an additional auxiliary continuous variable. For differentiable dynamics in the interval (t_k, t_{k+1}), we have

$$\dot{C}_k = C_k A^\top, \tag{11}$$

where $C_k(t_k) = \Sigma(t_k)$, resetting with every new measurement, and for nonlinear systems, A is Jacobian of the dynamics. Let the end of the integration interval be denoted t^-_{k+1}. Then, we cache the continuous-discrete smoothing gain

$$G_k = C_k(t^-_{k+1})\Sigma(t^-_{k+1})^{-1}, \tag{12}$$

which gives the following smoothing updates:

$$\begin{aligned}\mu^z_{t|T} &= \mu^z_{t|t} + G_k(\mu^z_{t+1|T} - \mu^z_{t+1|t}),\\ \Sigma^z_{t|T} &= \Sigma^z_{t|t} + G_k(\Sigma^z_{t+1|T} - \Sigma^z_{t+1|t})G_k^\top.\end{aligned} \tag{13}$$

D. Neural Ordinary Differential Equations

In many conventional neural network architectures such as residual networks, complex nonlinear transformations are compositions of transformations on a hidden state of the form

$$h_{k+1} = h_k + f(h_k, \theta_k), \tag{14}$$

where θ_k are the network parameters corresponding to the k^{th} hidden layer with values h_k. Recent research has shown that these operations can be interpreted as Euler discretizations with an implicit step size of 1 on some continuous transformation [14], [15], [16]. In the limit as the step size approaches 0 and the number of layers becomes infinite, we obtain a network globally parameterizing the derivative of this continuous transformation:

$$\dot{h}(t) = f(t, h(t), \theta). \tag{15}$$

This is the parameterization of choice for our dynamical models, where $h(t)$ is equivalent to our latent variable $z(t)$. Many dynamical prediction models that also rely on Bayesian inference techniques use Long-Short Term Memory (LSTM) networks to learn from discrete time-series data, which have been used to predict outcomes like future medical conditions [7]. Even in works like [10], which instead uses a neural ODE-based architecture, experiments are conducted on generative latent time-series prediction with an auto-encoding architecture where the encoder network is again a LSTM network that takes in the observed data in reverse to produce an initial latent state, which is then integrated forward in time and observed upon. The latter strategy shares some similarities with ours.

The use of LSTMs in these applications resembles the state estimation-based approach, but with non-optimal "measurement updates" in the sense that there is no connection between the dynamics model they use for propagating latent trajectories, the encoder which "estimates" latent states from observations, and the decoder which observes on the estimated latent states.

As explained above, the state estimation literature suggests that an optimal estimator should utilize both the dynamics and observation model together to iteratively maximize the amount of information recovered about the latent state from each observation. This motivates our model design in the next section.

III. Continuous-Time Neural Filters

The *continuous-time neural filter* (CTNF) is composed of two networks in contrast with the model in [10]: a continuous dynamics model $f_\theta(t, z(t))$ and an observation model $g_\phi(z)$. A sequence of time-series data $y_{1:T}$ is fed through a continuous-discrete EKF to obtain filtered belief

distributions $\mathcal{N}(\mu^z_{t|t}, \Sigma^z_{t|t}),\ t = 1, \ldots, T$. Next, an extended Kalman smoothing pass is executed, producing smoothed latent belief distributions $\mathcal{N}(\mu^z_{t|T}, \Sigma^z_{t,|T}),\ \ t = 0, \ldots, T$. These can be converted into approximate distributions over the corresponding observations made on the latent beliefs

$$\begin{aligned} \mu^y_{t|T} &= g(\mu^z_{t|T}), \\ \Sigma^y_{t|T} &= C_t \Sigma^z_{t|T} C_t^\top + R_t. \end{aligned} \tag{16}$$

A *reconstruction loss* is computed by taking the averaged negative log-likelihood of B batches of length T sequences of data given the smoothed belief distributions:

$$\mathcal{L}_R = -\frac{1}{BT} \sum_{i=1}^{B} \sum_{t=1}^{T} \log p(y_t; \mu^y_{t|T}, \Sigma^y_{t|T}). \tag{17}$$

The above procedure leverages the information gain from intermittent updates to correct the latent state estimate each step. However, we are also interested in learning an accurate dynamical model f that can be used for prediction without leveraging corresponding measurements. Therefore, we additionally use (9) to integrate $\mu_{0|T}, \Sigma_{0|T}$ up to time T to recover *predicted* distributions over the latent variable without measurement updates, $\mathcal{N}(\hat{\mu}^z_t, \hat{\Sigma}^z_t),\ \ t = 1, \ldots, T$. We can then recover distributions over the reconstructed observations as before, yielding a *prediction loss*:

$$\mathcal{L}_P = -\frac{1}{BT} \sum_{i=1}^{B} \sum_{t=1}^{T} \log p(y_t; \hat{\mu}^y_t, \hat{\Sigma}^y_t). \tag{18}$$

Finally, these two losses are mixed to produce a joint loss used for training the model:

$$\mathcal{L}_J = \alpha \mathcal{L}_R + (1 - \alpha) \mathcal{L}_P, \tag{19}$$

where $\alpha \in [0, 1]$ is a tuning parameter to adjust the weight granted to filter reconstruction versus prediction. This is nominally set to 0.5. The model architecture is summarized in Figure 2.

The model is also able to learn the process and measurement noise. For simplicity, in the trials presented in this paper, we choose to learn the elementwise logarithm of the entries of a diagonal covariance matrix. This allows the model to learn the contextual uncertainty of the environment in which the data is collected, making its deployment in the same environment more effective.

IV. Experiments

A. Setup

Model performance was evaluated on two dynamical systems: the Van der Pol (VDP) system and the pendulum. These are both highly nonlinear systems in two dimensions, allowing easy visualization for qualitative comparison. The VDP dynamics are written

$$\begin{aligned} \dot{x}_1 &= \mu \left(x_1 - \frac{1}{3} x_1^3 - x_2 \right), \\ \dot{x}_2 &= \frac{1}{\mu} x, \end{aligned} \tag{20}$$

where μ is a model parameter. The states represent planar Cartesian coordinates. The pendulum dynamics are written

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\sin x_1 - \beta x_2, \end{aligned} \tag{21}$$

where β is a parameter governing frictional effects. The states represent angular position and velocity. The training data were generated by integrating stochastic differential equations to recover 10000 noisy trajectories from each dataset, each consisting of 10 datapoints collected over a time horizon of 0.5 seconds.

The results were benchmarked against a naive discrete-time baseline model consisting of three networks: a LSTM network to propagate latent dynamics and two multilayer perceptron encoder and decoder networks for converting between latent and observation variables. These each had a single hidden layer with 64 hidden units. 5 datapoints were initially fed into the this network to initialize the latent dynamics, at which point the remaining data are used as training targets and are withheld from the network.

The dynamics neural ODE for the CTNF was a multilayer perceptron with 3 hidden layers, each with 64 hidden units and softplus activation functions. The observation network was also a multilayer perceptron with 3 hidden layers, each with 16 hidden units.

Both models were trained by minimizing the negative log-likelihood over observation reconstructions with loss mixing. The batch sizes were chosen to be 64. For the given integration interval and dimensionality of the data, this was found to be a reasonable compromise between lowering the variance and maintaining quick training iterations. A step scheduler was used for the learning rate, decaying it from an initial value of $\alpha = 0.01$ by a decay ratio of 0.975 every 10 steps down to a minimum of $\alpha = 1e-4$. The optimizer of choice was ADAM.

B. Results

We report prediction negative log-likelihoods on a hold-out validation set of 50 datapoints collected over a time horizon of 5 seconds (i.e., the loss is computed from a sequence of observations reconstructed without performing measurement updates). We allow the first 5 of these datapoints to be known by the prediction model in order to initialize it, then take the loss over the remaining 45 datapoints. The results are summarized in the following table and in Figures 4 and 3:

Model	VDP NLL	Pend. NLL
Baseline	+3.95	+3.06
CTNF	-2.13	+1.69

The CTNF model outperforms the baseline for both dynamical systems significantly. Additionally, the VDP and pendulum results for the CTNF model are reported after 10 and 20 epochs of training respectively, while the baseline model was trained for 200 epochs. We can see that though the CTNF models may not have fully converged, they still vastly outperform the baseline model with much better data efficiency.

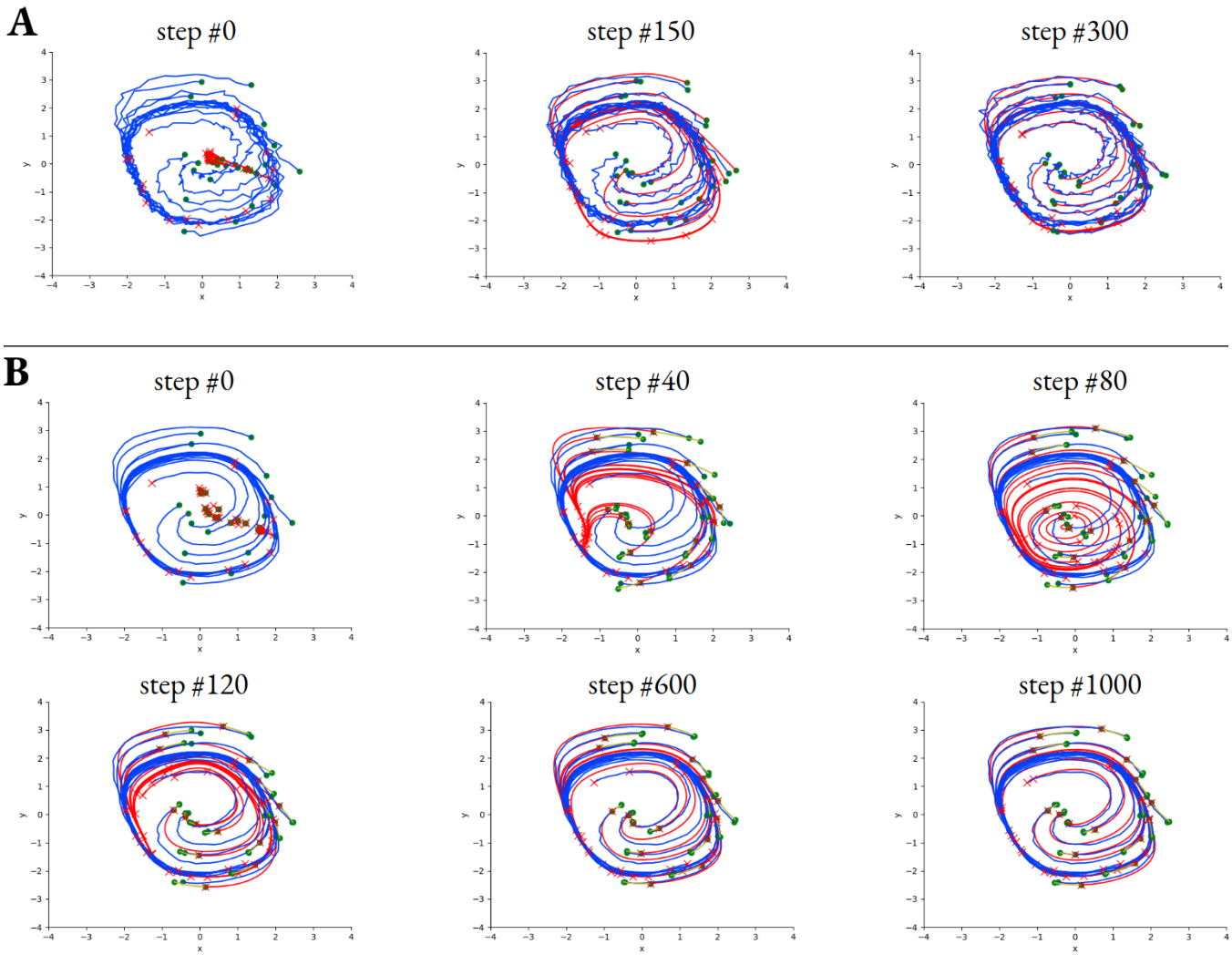

Fig. 3: Filtering versus prediction rollouts on the Van der Pol dynamics using the CTNF model. Green dots and red crosses represent the start and end of trajectories respectively. Blue lines represent ground truth trajectories while red lines represent predicted trajectories from rolling out the latent dynamics and observing upon them. **A:** Filtering runs on noisy measurements. In about 300 iterations, the filter architecture jointly defined by the dynamics and observation model is good enough to be an effective real-time EKF. **B:** Prediction runs on denoised validation trajectories. The yellow lines are the portion of the trajectory that is filtered and smoothed to initialize the latent state. At first, we observe that the filtering and smoothing steadily improves in reconstructing the initial observation. In later iterations, predictions rapidly improve since the effect of cascading errors over time is minimized.

In validation trials, we found that the CNTF is also able to extrapolate data even when it receives measurements that are farther in space and time than the data observed during training. We hypothesize that this is due to two major factors:

1) The continuous-time dynamics network allows local training on global parameters of the underlying dynamical vector field.
2) The smoothing operation effectively revises the initial condition of the latent state, so any prediction which begins at the smoothed initial condition will yield a more accurate prediction given a correct dynamics model.

In comparison, as shown in Figure 4, the baseline model seems to reasonably recover the underlying latent dynamics, but suffers from the inability to precisely recover the correct initial observation. This suggests that there is merit in the use of Bayesian smoothers in the CTNF prediction model, which has no trouble almost exactly recovering the correct initial observation, as seen in Figure 3. Additionally, the baseline model never seems to completely converge on the true trajectory, even after dozens of times more iterations of training than the CTNF.

The LSTM baseline was also observed to be prone to overfitting on the training data, and in particular, becoming overly optimistic about the uncertainty of its predictions. This is partially because the LSTM does not have a mathematically explicit way to relate the covariance of each prediction with the next, which is natural in a state estimation framework. Contrastly, this relation allows the CTNF to recover covariances more compatible with the predicted observations, yielding a superior negative log-likelihood.

It is also worth emphasizing the additional benefit that the CTNF model allows infinite resolution data imputation, since we have learned a continuous representation of the

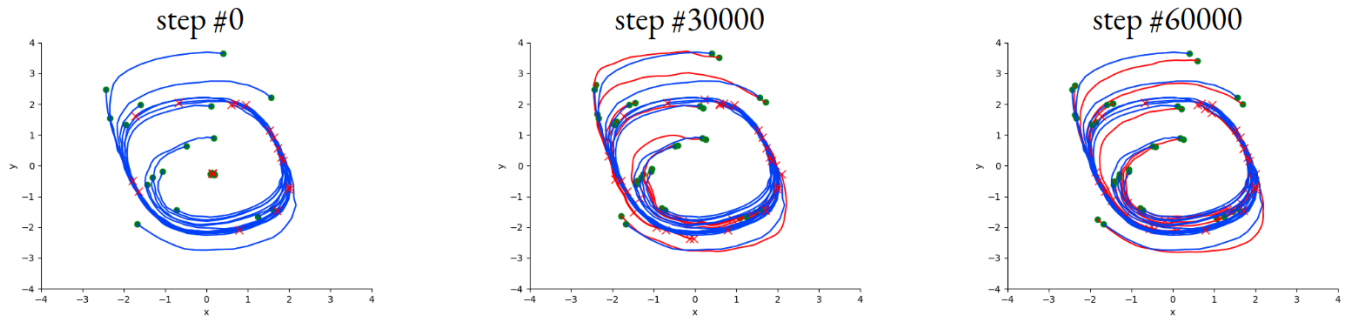

Fig. 4: Forward predictions of the baseline model. Note how it generally learns the dynamics well but has trouble precisely learning the initial condition of the trajectory, whereas the CTNF excels. The trajectories are also not precise even after tens of thousands of iterations.

latent dynamics, which means we may recover much finer or coarser predictions while the baseline model is restricted to the time discretization of the data upon which it trains.

V. Conclusion

A. Summary

This paper presents the continuous-time neural filter, a deep learning method leveraging the properties of Bayesian state estimation to effectively learn arbitrary nonlinear dynamics and observation models. In addition to demonstrating the ability of the architecture to perform filtering and smoothing well, we also show that it is capable of performing accurate dynamical prediction even when the inputs to the system are spatiotemporally distant from the data used to train the model. Additionally, the model outperforms a baseline model implemented using a LSTM, the standard option for time-series prediction.

While the method shows incredible promise, one detriment is that since no assumptions are made a priori about the parameterization of the dynamics, we may recover a model that has a latent representation that is difficult to interpret. This is because the latent states can be highly entangled, and will not necessarily be subsets of the interpretable observations. One possible way to recover partially interpretable latent models is by prescribing an expert observation model. For example, if you would like at least two latent states to represent the planar position of your system and the observation data give the planar position, then you can design the observation model to map the observations directly to the first two latent states. This may be essential for state-space control and stability analysis of learned dynamics.

B. Extensions and Future Work

While the CTNF was effective using an EKF, other approximate filtering methods can be explored, especially for data that are even more nonlinear than the systems studied here. For instance, the unscented Kalman filter [17] is another popular alternative to the EKF that is part of a family of filters known as *sigma point filters*, which have been shown to outperform the EKF in many cases at a slight additional computational cost.

One benefit of the UKF is that it avoids the costly computation of the Jacobian of the dynamics and measurement models during the prediction and update steps in training. To avoid the numerical issues associated with repeatedly computing matrix square roots, square root UKF implementations also exist for the continuous-discrete setting [18]. For even more complex or multimodal distributions, the general CTNF framework may also permit the use of continuous-discrete particle filtering [19].

We are also hoping to experiment with more complex methods of learning and modeling noise and covariance. Recent robotics research has shown that for many systems of interest, noise is not simply independent of state, but *heteroscedastic*, or dependent on the state of or inputs into a system. Taking these types of noise into account has been shown to yield better filtering performance, which suggests such models would also be useful to investigate for improving CTNF models [20].

Finally, we plan to apply CTNFs to dynamical prediction of non-physics-based systems, such as the prediction of human trajectories or the actions of other agents in unstructured environments. We hope to demonstrate that CTNFs can learn effective predictions over few training examples while still retaining high expressiveness such that they can be smoothly integrated into control systems while allowing an avenue for formally analyzing important properties of closed-loop systems such as safety and stability.

References

[1] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002, the book can be consulted by contacting: PH-AID: Wallet, Lionel. [Online]. Available: https://cds.cern.ch/record/1173048

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[3] X. Zhang, X. Liang, A. Zhiyuli, S. Zhang, R. Xu, and B. Wu, "AT-LSTM: An attention-based LSTM model for financial time series prediction," *IOP Conference Series: Materials Science and Engineering*, vol. 569, p. 052037, aug 2019.

[4] Y. Wang, "A new concept using lstm neural networks for dynamic system identification," in *2017 American Control Conference (ACC)*, 2017, pp. 5324–5329.

[5] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep kalman filters," 2015.

[6] X. Ma, P. Karkus, D. Hsu, and W. S. Lee, "Particle filter recurrent neural networks," *CoRR*, vol. abs/1905.12885, 2019.

[7] R. G. Krishnan, U. Shalit, and D. A. Sontag, "Structured inference networks for nonlinear state space models," *ArXiv*, vol. abs/1609.09869, 2016.

[8] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3601–3610.

[9] M. Karl, M. Sölch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," *ArXiv*, vol. abs/1605.06432, 2016.
[10] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *CoRR*, 2018.
[11] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 6 2019, pp. 3420–3431.
[12] R. E. Kalman, "A new approach to linear filtering and prediction problems"," *Journal of Basic Engineering*, 1960.
[13] S. Särkkä and J. Sarmavuori, "Gaussian filtering and smoothing for continuous-discrete dynamic systems," *Signal Process.*, vol. 93, pp. 500–510, 2013.
[14] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 7 2018, pp. 3282–3291.
[15] L. Ruthotto and E. Haber, "Deep neural networks motivated by partial differential equations," *Journal of Mathematical Imaging and Vision*, 04 2018.
[16] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, 05 2017.
[17] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.
[18] S. Sarkka, "On unscented kalman filtering for state estimation of continuous-time nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1631–1641, 2007.
[19] B. Ng, A. Pfeffer, and R. Dearden, "Continuous time particle filtering," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, ser. IJCAI'05. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, p. 1360–1365.
[20] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard, "Most likely heteroscedastic gaussian process regression," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 393–400. [Online]. Available: https://doi-org.stanford.idm.oclc.org/10.1145/1273496.1273546