Gated Inference Network: Inferencing and Learning State-Space Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

State-space models (SSMs) perform predictions by learning the underlying dynamics of observed sequence. We propose a new Kalman-based approach in both high and low dimensional observation space, which utilizes Bayesian filtering-smoothing to model system's dynamics more accurately than RNN-based SSMs and can be learned in an end-to-end manner. The designed architecture, which we call the *Gated Inference Network* (GIN), is able to integrate the uncertainty estimates and learn the dynamics of the system in both linear and non-linear cases that enables us to perform estimation and imputation tasks in both data presence and absence situations. The proposed model uses the GRU cells into its structure to complete the data flow in the classic Bayesian approach, while avoids expensive computations and potentially unstable matrix inversions. The GIN is able to deal with any time-series data and gives us a strong robustness to handle the observational noise. In the numerical experiments, we show that the GIN reduces the uncertainty of estimates and outperforms its counterparts , LSTMs, GRUs and variational approaches, in the estimation and the imputation tasks.

1 Introduction

State estimation and inference in the states in dynamical systems is one of the most interesting problems that has lots of application in signal processing and time series, where Rauch et al. work [1] is among the pioneers of research for learning the dynamic systems. In some cases, learning state space is a very complicated task due to the relatively high dimension of observations and measurements, which only provides the partial information about the states. As an instance, consider an image of a pendulum. The frames reveal the regional information, but they do not give any about the directional information. In addition, the number of the pixels in a simple image could easily exceed couple of hundreds that makes the inference much more sophisticated in this high dimensional space. Noise is another significant issue in this scenario, where it is more likely to obtain a noisy observation even we might lose some part of the crucial information due to the noise. Time series prediction and estimating the next scene, e.g, the state prediction or next observation prediction, is another substantial application that again requires the inference within the states which comes from the observations.

Classical memory networks such as LSTMs [2], GRUs [3] and simple RNNs like [4] and [5] fail to give some intuition about the uncertainties and dynamics. A bunch of approaches perform the Kalman Filtering (KF) in the latent state which usually requires a deep encoder for feature extraction. Archer et al. [6], Krishnan et al. [7], Hashempour et al. [8] and [9] belong to these group of works. However, the mentioned solution has some restrictions, they are not able to deal with the non-linearity of the system so it is necessary to employ the variants of KF, e.g., EKF, or use variational inference, like what Kingma did in [10], that increases the complexity of the model noticeably. Moreover,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

in the variational inference approaches that usually implemented in the context of variational auto encoders for dimension reduction, they do not have access to the loss directly and have to minimize its lower bound instead, which reduce the ability of learning dynamics and affect the performance of the model.

The mentioned restrictions for KF and its variants and variational models in addition the necessity of having a metric to measure the uncertainty, motivate us to introduce the GIN, an end to end structure with dynamics learning ability enjoying Bayesian properties for filtering-smoothing. The contributions of GIN are: (i) modeling high-low dimensional sequences: we show the eligibility of the GIN to handle both cases. We conduct experiments for each case and compare the GIN with the state-of-the-art approaches, e.g. Klushyn et al. [11] for the former case and Satorras et al. [12], Rangapuram et al.[13] for the latter one. (ii) Introducing non-linearity in the Kalman filtering-smoothing via GRU cells: to attain more accurate inference of observed dynamical system, we apply further non-linearity by GRU cells that increases the modeling capability of the Kalman filtering-smoothing. By conduction an ablation study of the GIN being replaced by a linear Gaussian state transition without non-linearity, we show the GIN is able for better learning state space representation with disentangled dynamics features. (iii) Noise robustness: verified by the numerical results, inferencing for highly distorted sequences is feasible with the GIN. (iv) Missing data imputation: by using Bayesian properties, the GIN decides whether to keep the previous information in the memory cell or update them by the obtained observation.

The concept of the GIN is related to KalmanNet [14] and Ruhe et al. [15] that use GRU in their structure for the state update. However, thy are only able to deal with low-dimensional state space and accordingly cannot be applied for complex sensory inputs like images. Moreover, their structure require the full, or at least partial, dynamic information, while the GIN can learn them directly in the absence of the dynamics information.

2 Related Works

LSTMs and GRUs are among the more complex RNN structures that are able to deal with both high and low dimensional observations. However, in these models, they ignore some part of the domain knowledge that is present, e.g., uncertainty. But, in our experiments we artificially add a part that stands for the uncertainty of the LSTMs and GRUs to make them comparable with our results.

To deal with complex sensory inputs, some approaches integrate a deep auto encoders into their architecture. Among these works, Embed to Control (E2C) [16] uses a deep encoder to obtain the latent observation and a variational inference about the states. However, these methods are not able to deal with missing data problem and imputation task since they do not rely on memory cells and are not recurrent. Another bunch of works like BackpropKF [17] apply CNNs for dimension-reduction and output both the uncertainty vector and latent observation. Therefore, they can perform extended variants of KF in the latent space with the known dynamics. However, these kinds of approaches cannot handle the cases without the knowledge of the dynamics of the system, while GIN provides a principled way for learning them when we are lacking the dynamics.

Toward learning state space (system identification) a bunch of works like Wang et al. [18], Ko et al. [19] and Frigola et al. [20] propose algorithms to learn GPSSMs based on maximum likelihood estimation with the iterative EM algorithm. In [20], Frigola et al. obtain sample trajectories from the smoothing distribution, then conditioned on this trajectory they conduct M step for the model's parameters. Other group of works consider EM-based variational-inference for system identification like Structured Inference Networks (SIN) [7], where it utilizes a RNN to update the state. The Kalman Variational Autoencoder (KVAE) [21] and Extended KVAE (EKVAE) [11] use the original KF equations and enjoys both filtering and smoothing. However, these variational inference based methods are not able to estimate the states directly since they are generative models of the observations. Optimizing the lower bound of loss instead of the direct loss is another issue with the variational based approaches, while it is addressed by a direct end-to-end optimization in the GIN. We compare the GIN with variational-based approaches in the experiment section.

Among the related works, the recurrent kalman network (RKN) [22] is more similar to our approach, where they design an auto-encoder based structure and apply KF equations for filtering with some simplifications in the state information. However, RKN is limited to the cases where the latent state dimension is two times bigger than the latent observation, and some rough assumption about the state covariance, where in the GIN all are released.

3 Gated Inference Network for System Identification

In the context of System Identification (SI), i.e. when we lack the dynamics, the GIN is similar to a Hammerstein-Wiener (HW) model [23] [24], in the sense that it estimates the system parameters directly from the observations, which is in the figure 1. $e(.)$ and $d(.)$ are implemented with non-linear functions, e.g. auto-encoders and MLPs for high and low dimensional observations. However, in the presence of system dynamics −usually in low dimensional observations−, $e(.)$ and $d(.)$ are identity. Due to the presence of noisy inputs, often unknown dynamics and hidden variables of the SI tasks, direct maximization of the likelihood is difficult. Hence, it is common to employ the EM algorithm to solve the problem iteratively. In the SI-based approaches, e.g. KVAE [21], EKVAE [11] and also our GIN, filtering/smoothing with a set of fixed parameters γ is conducted (E step), and then updating the set of parameters γ is performed such that the obtained likelihood is maximized (M step).

Transition block in 1 represents the dynamics of the system that allows for a valuable inference using the Gaussian state space filtering-smoothing equations. However, unlike a HW model that assumes the linearity of state space transition, we employ non-linear GRU cells in the transition block that calculate the Kalman Gain (KG) and smoothing gain (SG) in an appropriate manner by circumventing the complexity of filtering-smoothing equations, i.e matrix inversion issues. In addition to non-linear $e(.)$ and $d(.)$, GRU cells empower the whole system by applying further non-linearity to the linear Gaussian state space model (LGSSM), e.g. transition block in the conventional HW models. Numerical results indicate that linearity assumption of the state space fails to obtain the accurate dynamics for complex sensory inputs. However, by the proposed structure, having a good inference for even the complex non-linear systems with high dimensional observations is feasible. To achieve this, we assume that the state representation can be converted into Gaussian state space models (GSSMs), which are commonly used to model sequences of vectors.

In the absence of dynamics, the construction of the GIN is based on that part of filtering-smoothing computation, where we have to use unavailable knowledge, i.e., dynamics and noise matrices. The dynamics of the system $\mathbf{F}$ and $\mathbf{H}$ might not be available or hard to obtain; while the process noise and observation noise are unknown. Accordingly, we construct GIN to learn the KG and SG from data in an end to end manner, then we utilize the constructed KG and SG during inference time to obtain the filtered-smoothed states. The proposed architecture is depicted in figure 2. In the presence of dynamics, auto-encoder and *Dynamics Network* in 2 are removed, while $\mathbf{F}$ and $\mathbf{H}$ are directly used.

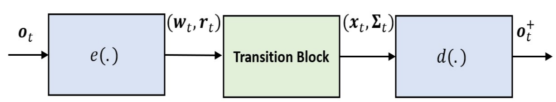


Figure 1: The GIN as a HW model for system identification. In the case of high-dimensional observation $\mathbf{o}_t$, the nonlinear functions $e(.)$ and $d(.)$ can be implemented by an encoder and decoder, respectively. While for the low-dimensional case, MLPs are utilized instead. The relation between the internal variables, $\mathbf{w}_t$ and $\mathbf{x}_t$, is simulated by the transition block.

3.1 Dynamic Parameter Learning

By defining $\gamma_t = (\mathbf{F}_t; \mathbf{H}_t)$ as parameters which explain how the posterior state $\mathbf{x}^+_{t-1}$ updates from time $t-1$ to t, $\mathbf{x}_{1:T}$ as the latent states and $\mathbf{w}_{1:T}$ as the latent noisy observations, we introduce the prediction parameterization as $p_{\gamma_t}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{w}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t; \mathbf{F}_t\mathbf{x}_{t-1}, \mathbf{Q}_t)$ and filtering parameterization as $p_{\gamma_t}(\mathbf{x}_t|\mathbf{w}_{1:t}) = \int p_{\gamma_t}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{w}_t) p_{\gamma_t}(\mathbf{x}_{t-1}|\mathbf{w}_{1:t-1}) d\mathbf{x}_{t-1} = \mathcal{N}(\mathbf{x}_t; \mathbf{x}^+_t, \mathbf{\Sigma}^+_t)$ (see A.1 for the further details and smoothing parameterization). These parameterization gives us some insight to 1-illustrate a tractable way to construct $p_\gamma(\mathbf{x}|\mathbf{w})$ and accordingly obtain the posteriors $(\mathbf{x}^+, \mathbf{\Sigma}^+)$, based on which $\mathbf{o}^+$ is constructed and 2- appropriately modeling γ and KG(SG). Due to the non-linearity of the complex systems, we have to tackle this issue by training the parameters γ_t of the model in each time step t as a function of the latent observations that already observed, up to time $t-1$. In more details, the updates in dynamics of the system at each time step t might be a function of the history of the system, which are latent observations $\mathbf{w}_{0:t-1}$. The state $\mathbf{x}^+_{t-1}$ in GSSM is a function of the latent observations $\mathbf{w}_{0:t-1}$, in other words $\mathbf{x}^+_{t-1}$ includes the required information to update the dynamics, then the joint probability distribution of the GSSM is

$$p_\gamma(\mathbf{w}, \mathbf{x}) = \prod_{t=1}^{T} p_{\gamma_t(\mathbf{x}^+_{t-1})}(\mathbf{w}_t|\mathbf{x}_t).p(\mathbf{x}_1) \prod_{t=2}^{T} p_{\gamma_t(\mathbf{x}^+_{t-1})}(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{1}$$

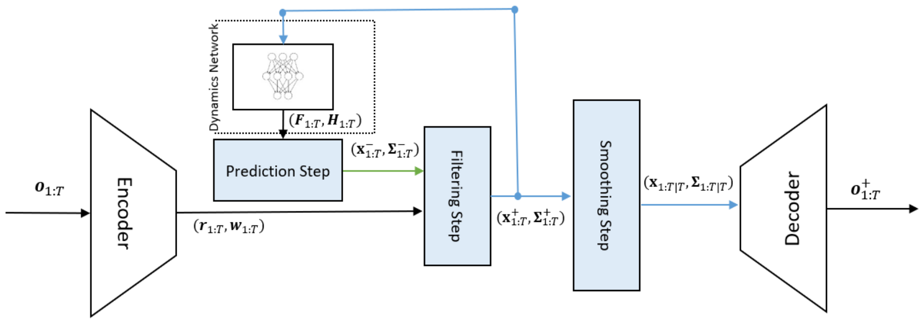


Figure 2: The high level structure of *Gated Inference Network* for high dimensional observation in the lack of dynamics, while for low dimensional cases auto-encoder is replaced by MLPs. The latent observation $\mathbf{w}_t$ and its corresponding estimate of uncertainty $\mathbf{r}_t$, i.e., observation noise, are obtained from the encoder(MLPs). In each time step, the last posterior mean $\mathbf{x}^+_{t-1}$ is fed to the *Dynamic Network* to compute $\mathbf{F}_t$ and $\mathbf{H}_t$. In the *Prediction Step* the next priors $(\mathbf{x}^-_t, \boldsymbol{\Sigma}^-_t)$ are obtained by using new dynamics and the last posteriors. In the filtering step, by using the priors $(\mathbf{x}^-_t, \boldsymbol{\Sigma}^-_t)$ and the observation $(\mathbf{w}_t, \mathbf{r}_t)$, the next posteriors $(\mathbf{x}^+_t, \boldsymbol{\Sigma}^+_t)$ are obtained. Applying smoothing operation over the obtained posteriors $(\mathbf{x}^+_t, \boldsymbol{\Sigma}^+_t)$ is feasible in the smoothing step. Finally, the decoder(MLP) is utilized to produce $\mathbf{o}^+_t$, which can be the high-low dimensional noise free estimates.

where we usually initial $\mathbf{x}_1$ as a zero mean and known covariance normal distribution. In the GIN, the dependence of γ_t on $\mathbf{x}^+_{t-1}$ is modeled by *Dynamics Network* α_t, i.e. $\gamma_t = \alpha_t(\mathbf{x}^+_{t-1})$. We learn K state transition and emission matrices $\mathbf{F}^k$ and $\mathbf{H}^k$, and combine each one with the state dependent coefficient $\alpha^k(\mathbf{x}^+_{t-1})$. A separated neural network with softmax output is utilized to learn α^k.

$$\mathbf{F}_t = \sum_{k=1}^{K} \alpha^k_t \mathbf{F}^k_t, \quad \mathbf{H}_t = \sum_{k=1}^{K} \alpha^k_t \mathbf{H}^k_t \tag{2}$$

In the case when we are aware of the dynamics of the system, e.g. Lorenz attractor and NLCT, we can directly use the dynamics instead of learning them from the data and remove auto-encoder(MLP).

3.2 Learning The Process Noise

In the filtering procedure, see A.1, the process noise in time t is obtained as

$$\mathbf{Q}_t = \boldsymbol{\Sigma}^-_t - \mathbf{F}_t \boldsymbol{\Sigma}^+_{t-1} \mathbf{F}^T_t. \tag{3}$$

where $\boldsymbol{\Sigma}^-_t$, $\mathbf{F}_t$ and $\boldsymbol{\Sigma}^+_{t-1}$ are prior state covariance, transition matrix and posterior state covariance at time t, respectively. It is shown that the process noise, $\mathbf{Q}_t$, can be written as a function of $\mathbf{F}_t$, while the derivations are rather lengthy, therefore, we refer to the appendix materials A.2. From (23), the relation of the process noise with the transition matrix indicates that $\mathbf{F}_t$ can possess the effects of $\mathbf{Q}_t$ if we learn it in an appropriate manner. $\mathbf{F}_t(\mathbf{Q}_t)$ notation means that the learned $\mathbf{F}_t$ comprises the effects of $\mathbf{Q}_t$, while for the simplicity we use $\mathbf{F}_t$ abbreviation. Therefore, it is possible to rewrite (3) as

$$\boldsymbol{\Sigma}^-_t = \mathbf{F}_t \boldsymbol{\Sigma}^+_{t-1} \mathbf{F}^T_t. \tag{4}$$

Another way to have a a meaningful inference about the process noise matrix is to obtain it from (21) as a recursive function of $\mathbf{x}^+_{t-1}$ and $\mathbf{Q}_{t-1}$. Intuitively, $\mathbf{g}$ function in (21) can be implemented by a memory cell, e.g., a GRU cell, to keep the past status of $\mathbf{Q}$,however, it increases the complexity of the model. Equivalently, one can obtain $\mathbf{Q}_t$ directly from $\mathbf{x}^+_{t-1}$ as stated in (23), where a fully connected with a positive activation function plays the rule of $\mathbf{g}$ that maps $\mathbf{x}^+_{t-1}$ to $\mathbf{Q}_t$. Numerical results corresponding to the mentioned methods for inference in the process noise can be found in A.9. Both of these solutions can be utilized when the dynamics are known, i.e. we cannot learn the effects of $\mathbf{Q}_t$ jointly with $\mathbf{F}_t$ as $\mathbf{F}_t$ is not trainable.

3.3 Filtering and Smoothing

To construct the KG and SG networks, we have to find appropriate inputs containing useful information to attain the KG and SG. In filtering-smoothing instruction, KG and SG are given by (5) and (6), respectively.

$$\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{H}_t^T.[\mathbf{H}_t \mathbf{\Sigma}_t^- \mathbf{H}_t^T + \mathbf{R}_t]^{-1} \propto (\mathbf{\Sigma}_t^-, \mathbf{R}_t) \tag{5}$$

$$\mathbf{J}_t = \mathbf{\Sigma}_t^+ \mathbf{F}_{t+1}^T.[\mathbf{F}_{t+1} \mathbf{\Sigma}_t^+ \mathbf{F}_{t+1}^T + \mathbf{Q}_{t+1}]^{-1} = \mathbf{\Sigma}_t^+ \mathbf{F}_{t+1}^T \mathbf{\Sigma}_{t+1}^- \propto \mathbf{\Sigma}_{t+1}^- \tag{6}$$

(5) is proportional to the prior covariance at time t, $\mathbf{\Sigma_t^-}$, and the observation noise matrix, $\mathbf{R_t}$, while (6) is proportional to prior covariance matrix at time $t+1$, $\mathbf{\Sigma}_{t+1}^-$. Our encoder(MLP) directly maps the observation noise matrix from the observation space, but the state covariance is a recursive function of previous states. Consequently, we consider GRU^{KG} and GRU^{SG} which are GRU networks, including mapping $[\mathbf{f}(\mathbf{\Sigma_t^-}), \mathbf{R_t}]$ and $\mathbf{f}(\mathbf{\Sigma_{t+1}^-})$ to the KG and SG, respectively. GRU^{KG} considers $\mathbf{R}_t$, a diagonal matrix with $\mathbf{r}_t$ elements in figure 2, as a part of its input to incorporate the effects of observation noise. In the case of high dimensional state space, due to the high dimension of $\mathbf{\Sigma_t^-}$ and $\mathbf{\Sigma_{t+1}^-}$, $\mathbf{f}$ is a convolutional layer with pooling to extract the valuable information of the covariance matrix that reduces its size, while for the low dimension of $\mathbf{\Sigma_t^-}$ and $\mathbf{\Sigma_{t+1}^-}$, the identity function is used for $\mathbf{f}$. Such structure does not have extreme negative effects on the performance since in the reality covariance matrix is sparse and we can code the information into smaller dimension, without loosing the general information.

Prediction Step. Similar to the model based Kalman Filter, by using dynamics of the system and linear transition, the next priors are obtained from the current posterior by

$$\mathbf{x}_{t+1}^- = \mathbf{F}_t \mathbf{x}_t^+ , \quad \mathbf{\Sigma}_{t+1}^- = \mathbf{F}_t \mathbf{\Sigma}_t^+ \mathbf{F}_t^T \tag{7}$$

where $\mathbf{F}_t$ is the learned transition matrix comprises the effects of the process noise by which it is feasible to predict state mean and the state covariance matrix.

Filtering Step. To obtain the next posteriors based on the new observation $(\mathbf{w}_t, \mathbf{r}_t)$, i.e. the output of $e(.)$ in figure 1, we have to use the obtained KG matrix from GRU^{KG} network and emission matrix $\mathbf{H}_t$ to complete updating the state mean vector and state covariance matrix. This procedure is given by

$$\mathbf{S}_t^- = \mathbf{H}_t.\mathbf{\Sigma}_t^-.\mathbf{H}_T^T + \mathbf{R}_t, \quad \mathbf{K}_t = GRU^{KG}(\mathbf{f}(\mathbf{\Sigma}_t^-), \mathbf{R}_t), \tag{8}$$

$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t.[\mathbf{w}_t - \mathbf{H}_t \mathbf{x}_t^-], \quad \mathbf{\Sigma}_t^+ = \mathbf{\Sigma}_t^- + \mathbf{K}_t.\mathbf{S}_t^-.\mathbf{K}_t^T. \tag{9}$$

In addition to avoiding the matrix inversion that arises in the computation of Kalman gain and applying non-linearity to handle more complex dynamics, the architecture of KG network, GRU^{KG}, can reduce the dimension of the input to its corresponding GRU cell, and thus reduces the total amount of parameters quadratically. Additionally, using model based Kalman Filter for observation update and positive $\mathbf{r_t}$ vector, makes this procedure trivial to guarantee that the state covariance will be symmetry and positive definite, which is not negatively affected by the proposed structure.

Smoothing Step. After obtaining filtered states $(\mathbf{x}_{1:T}^+, \mathbf{\Sigma}_{1:T}^+)$ in filtering step, we employ smoothing properties of Bayesian to get smoothed version of the states. In this stage, we use $\mathbf{J}_{1:T}$ matrices obtained from GRU^{SG} network, transition matrices $\mathbf{F}_{1:T}$ and filtered states $(\mathbf{x}_{1:T}^+, \mathbf{\Sigma}_{1:T}^+)$. The procedure in each smoothing step is given by:

$$\mathbf{x}_{t|T} = \mathbf{x}_t^+ + \mathbf{J}_t \big[\mathbf{x}_{t+1|T} - \mathbf{F}_{t+1} \mathbf{x}_t^+\big], \quad \mathbf{\Sigma}_{t|T} = \mathbf{\Sigma}_t^+ + \mathbf{J}_t \big(\mathbf{\Sigma}_{t+1|T} - \mathbf{F}_{t+1} \mathbf{\Sigma}_t^+ \mathbf{F}_{t+1}^T\big) \mathbf{J}_t^T \tag{10}$$

where the first smoothing state is set to the last filtering state, i.e. $(\mathbf{x}_{T|T}, \mathbf{\Sigma}_{T|T}) = (\mathbf{x}_T^+, \mathbf{\Sigma}_T^+)$.

3.4 Training and Loss

For the state estimation task, the output follows Gaussian distributions. On this base, we can maximise a pseudo-likelihood function $\mathcal{L}_\mathbf{s} := \log \prod_{t=1}^T p(\mathbf{s}_t|\mathbf{o}_{1:T})$, where $\mathbf{s}_t$ is the estimated state, i.e. equal to $\mathbf{o}_t^+$ in figure 2. For the image imputation task, in addition to the pseudo-likelihood for inferring the states, we add the reconstruction pseudo-likelihood for inferring images by using Bernoulli distributions as $\mathcal{L}_\mathbf{i} := \log \prod_{t=1}^T p(\mathbf{i}_t|\mathbf{o}_{1:T})$, i.e. the decoder in figure 2 maps both state $\mathbf{s}_t$ and image $\mathbf{i}_t : \mathbf{o}_t^+ = [\mathbf{i}_t, \mathbf{s}_t]$. Further details of distribution assumptions can be found in the appendix A.3. Filtering procedure, which includes *Dynamics Network*, prediction step and filtering step, can be considered as a memory network. Similarly, smoothing procedure that includes smoothing step is another memory network so that we can calculated the gradients by using (truncated) BPTT [25] to train the GRU cells inside the architecture in an end to end manner. To prevent GRU cells from potential instability, we split lengthy sequences into smaller pieces($\leq$200) and pass them separately.

Likelihood for Inferring States. Consider the ground truth sequence is defined as $\mathbf{s_{1:T}}$. We determine the log likelihood of the states as:

$$\mathcal{L}(\mathbf{s}_{1:T}) = \sum_{t=1}^{T} \log \mathcal{N}\Bigg(\mathbf{s_t}\Bigg|\text{dec}_{\text{mean}}(\mathbf{x_{t|T}}), \text{dec}_{\text{covar}}(\mathbf{\Sigma_{t|T}})\Bigg) \tag{11}$$

where the $\text{dec}_{\text{mean}}(.)$ and $\text{dec}_{\text{covar}}(.)$ determines those parts of the decoder that are used to obtain the state mean and state variance, respectively.

Likelihood for inferring images. For the imputation task, consider the ground truth as the sequence of images and their corresponding states, which are defined as $[\mathbf{s}_{1:T}, \mathbf{i}_{1:T}]$ and the dimension of $\mathbf{i}_t$ is D_o. We determine the log likelihood:

$$\mathcal{L}(\mathbf{o}^{+}_{1:T}) = \mathcal{L}(\mathbf{s}_{1:T}) + \lambda \sum_{t=1}^{T} \sum_{k=0}^{D_o} \mathbf{i}_t^{(k)} \log\big(\text{dec}_k(\mathbf{x_{t|T}})\big) + \big(1 - \mathbf{i}_t^{(k)}\big) \log(1 - \text{dec}_k(\mathbf{x_{t|T}})). \tag{12}$$

$\text{dec}_k(\mathbf{x}_t)$ defines the corresponding part of the decoder that maps the k-th pixel of $\mathbf{i}_t$ image and λ constant determines the importance of the reconstruction. The first term in RHS is obtained from (11) and we abbreviate the second term as $\mathcal{L}(\mathbf{i}_{1:T})$.

3.5 The Gated Inference Network

The mentioned structure for the filtering and smoothing enables us to introduce a new type of memory neural network, which lets us to deal with both low and high dimensional observation and state spaces. Keeping numerical stability due to the structure, computational efficiency by circumventing expensive calculations, applying non-linearity by GRU cells for handling non-linear dynamics and comparatively low memory consumption are the benefits of the new memory cell. Relying on the states learning, the GIN is capable to handle the situation when the inputs are not available, e.g. imputation task, by just ignoring filtering step and setting the posterior to the prior. The design of the GIN to obtain the KG/SG can be seen as a controlling unit that analyses how much the new/future observation affects the state that leads to a better state update.

4 Evaluation and Experiments

We divide our experiments into two parts, first the tasks in which the observation space is high dimensional like sequence of images, and second the applications that the observation is in low dimension by itself so there is no need to include encoder for dimension reduction. In the cases that the dynamics are known, the procedure is completed by removing *Dynamics Network* and auto-encoder(MLPs). The training algorithms of both cases are added in the appendix section A.10.

4.1 High Dimensional Observation with Lack of Dynamics

The internal state is divided into two parts, the first one is utilized to determine the mean with size of n, and the second to indicate the covariance with size of n^2. For this purpose, we consider two experiments including single pendulum and double pendulum where the dynamics of the latter one is more complicated and also another difficulty exists that the second link of the double pendulum may be occluded by the first link during the simulation.

4.1.1 Single Pendulum and Double Pendulum

The inputs of the encoder are the images with size of 24×24. The angular velocity is disturbed by transition noise which follows Normal distribution with $\sigma = 0.1$ as its standard deviation at each step. In the pendulum experiment, we perform the filtering-smoothing by the GIN where the observation is distorted with high observation noise. Furthermore, we compare GIN with LGSSM, where the GRU cells are omitted from the GIN structure and classic filtering-smoothing equations are used, instead.

Table 1: Double pendulum state estimation. (x_1, x_3) refers to the position of the first joint, while (x_2, x_4) is for the second joint.

Model	$SE^{Pos}_{x_1}$	$SE^{Pos}_{x_3}$	$SE^{Pos}_{x_2}$	$SE^{Pos}_{x_4}$	Log Likelihood
LSTM (units=50)	0.163	0.171	0.148	0.167	3.901 ± 0.706
LSTM (units=100)	0.154	0.147	0.134	0.152	4.053 ± 0.565
GRU (units=50)	0.189	0.183	0.179	0.177	3.886 ± 0.369
GRU (units=100)	0.164	0.156	0.162	0.145	3.976 ± 0.231
KVAE (n=2m)	0.193	0.188	0.178	0.149	3.679 ± 0.101
KVAE (n=3m)	0.171	0.159	0.151	0.162	3.801 ± 0.116
RKN (n=2m)	0.134	0.129	0.139	0.118	4.176 ± 0.294
$LGSSM_{filter}$(n=3m)	0.125	0.119	0.121	0.107	4.192 ± 0.127
$LGSSM_{smooth}$(n=3m)	0.109	0.111	0.104	0.101	4.231 ± 0.154
GIN_{filter}(n=2m)	0.115	0.109	0.119	0.109	4.224 ± 0.105
GIN_{filter}(n=3m)	0.093	0.091	0.098	0.089	4.329 ± 0.151
GIN_{smooth}(n=2m)	0.091	0.104	0.101	0.092	4.308 ± 0.123
GIN_{smooth}(n=3m)	**0.079**	**0.083**	**0.085**	**0.077**	**4.477 ± 0.168**

Table 2: Pendulum state estimation. By consider $n = 3m$, intuitively the last part of the state is dedicated to the acceleration information causing a more lieklihood. See A.8 and A.9 for more details.

Model	$SE^{Pos}_{x_1}$	$SE^{Pos}_{x_2}$	Log Likelihood
LSTM (units=25)	0.092	0.094	5.891 ± 0.151
LSTM (units=100)	0.089	0.087	5.751 ± 0.215
GRU (units=30)	0.095	0.089	5.986 ± 0.168
GRU (units=100)	0.091	0.089	5.698 ± 0.205
KVAE (n=2m)	0.104	0.095	5.786 ± 0.098
KVAE (n=3m)	0.088	0.093	5.858 ± 0.113
RKN (n=2m)	0.078	0.075	6.161 ± 0.23
$LGSSM_{filter}$	0.077	0.073	6.211 ± 0.265
$LGSSM_{smooth}$	0.071	0.069	6.242 ± 0.109
GIN_{filter}(n=2m)	0.073	0.07	6.192 ± 0.239
GIN_{filter}(n=3m)	0.067	0.066	6.315 ± 0.220
GIN_{smooth}(n=2m)	0.065	0.067	6.292 ± 0.173
GIN_{smooth}(n=3m)	**0.059**	**0.057**	**6.445 ± 0.165**

The distortion with noise changes between noiseless situation to the whole image distorted with noise such that pure noise is observed. Moreover, the cell may observe fully distorted image for consecutive time steps, which means that the noise has correlation with time. The log-likelihood and squared error (SE) of positions for single and double pendulum are given in Table 2 and 1, respectively.

We conduct another experiment to check the ability of our model for image imputation and make a comparison against the previous variational inference methods. By randomly deleting the half of images from the generated sequences, we conduct the image imputation task to our model by predicting those missing parts, while the missingness applied to train and test are not same, but random. The results are in table 3 and 4.The GIN outperforms all the other models, although the variational inference models have more complex structures in KAVE and EKVAE. The RKN uses factorised inference causing a lower memory consumption but they have some restrictive assumptions about the covariance matrix, loosing some information is inevitable in this scenario. Moreover, n just can be $2m$ in the RKN. We include the results using the MSE as well, to illustrate that our approach is also competitive in prediction accuracy.(See A.9).

4.1.2 Visual Odometry of KITTI Dataset

We also evaluate the GIN with the higher dimensional observations for the visual odometry task on the KITTI dataset [26]. This dataset consists of 11 separated image sequences with their corresponding labels. In order to extract the positional features, we use a feature extractor network proposed by Zhou et al. in [27]. The obtained features are considered as the observations of the GIN, i.e. $(\mathbf{w}, \mathbf{r})$.

Table 3: Image imputation task for the different models. Models contain boolean masks determining the available and missed images. For uninformed masks, a black image is considered as the input of the cell whenever the image is missed, which requires the model to infer the accurate dynamics for the generation. We conduct uninformed experiment as well.

Model	Log Likelihood
E2C	-95.539 ± 1.754
SIN	-101.268 ± 0.567
KVAE (informed smooth)	-14.217 ± 0.236
KVAE (unformed smooth)	-39.260 ± 5.399
EKVAE (informed smooth)	-12.897 ± 0.524
EKVAE (unformed smooth)	-29.246 ± 3.328
RKN (informed)	-12.782 ± 0.0160
RKN (uninformed)	-12.788 ± 0.0142
LGSSM(informed smooth)	-12.695 ± 0.048
GIN (informed smooth)	-12.215 ± 0.027
GIN (unformed smooth)	-12.246 ± 0.029

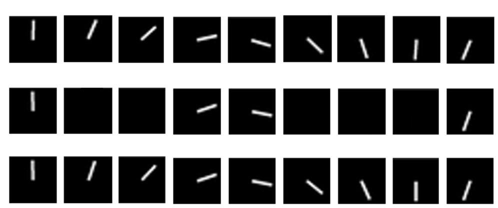


Figure 3: Pendulum image imputation. Each figure, beginning from up to down, indicates the ground truth, uninformed observation and the imputation results of the GIN(smoothed). Missingness is applied randomly for train and test. For further visualization, see A.5.

Table 4: Image imputation for double pendulum.

Model	Log Likelihood
KVAE (informed smooth)	-15.917 $\pm$ 0.294
KVAE (unformed smooth)	-38.544 $\pm$ 6.419
EKVAE (informed smooth)	-13.917 $\pm$ 0.414
EKVAE (unformed smooth)	-33.548 $\pm$ 4.516
RKN (informed)	-13.832 $\pm$ 0.023
RKN (uninformed)	-13.898 $\pm$ 0.0191
LGSSM(informed smooth)	-13.775 $\pm$ 0.013
GIN (informed smooth)	-13.284 $\pm$ 0.021
GIN (unformed smooth)	-13.351 $\pm$ 0.019

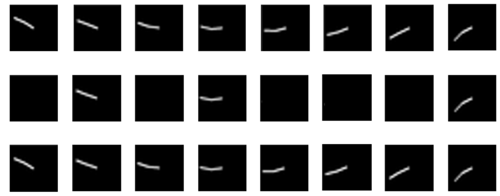


Figure 4: Double pendulum image imputation. Each figure, beginning from up to down, indicates the ground truth, uninformed observation and the imputation results of the GIN(smoothed).

Additionally, we compare the results with LSTM, GRU, DeepVO [28] and KVAE. The results are in Table 5, where the common evaluation scheme for the KITTI dataset is exploited. The GIN outperforms the LSTM and GRU, while the performance is comparable with the tailored DeepVO for this task. The results of the KVAE degrades substantially as we have to reduce the size of the latent observation, which causes an inevitable information loss, to prevent the complexity of matrix inversion in the smoothing-filtering.

4.2 Low Dimensional Observation with Presence of Dynamics

In the low dimensional observations, the encoder is replaced by a MLP to perform as a nonlinear function, while it outputs $\mathbf{w}_t$ and $\mathbf{r}_t$. In the presence of the dynamics, they are directly employed without using the *Dynamics Network*. We conduct two experiments for this case, Lorenz attractor problem and the real world dynamics NCLT dataset, where we are aware of the dynamics.

4.2.1 Lorenz Atrractor

The Lorenz system is a system of ordinary differential equations that describes a non-linear dynamic system used for atmospheric convection. Due to nonlinear dynamics of this chaotic system (see A.6), it can be a good evaluation for the GIN cell, while it is much more complicated than the linear cases. We evaluate the performance of the GIN on a trajectory of 5k length. Each point in the generated trajectories is distorted with an observation noise that follows Gaussian distribution with standard deviation $\sigma = 0.5$. The likelihood with Gaussian distribution is calculated and maximized in the training phase. The mean square error (MSE) of the test data for various number of training samples are depicted in figure 5. *Hybrid* is a graphical GNN based model [12] and DSSM [13] is a version of LGSSM using LSTM cells and is not aware of dynamics.

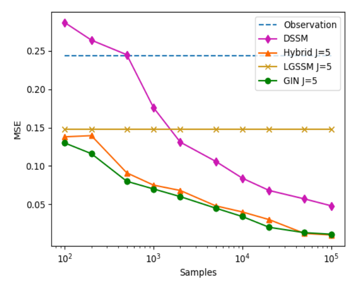
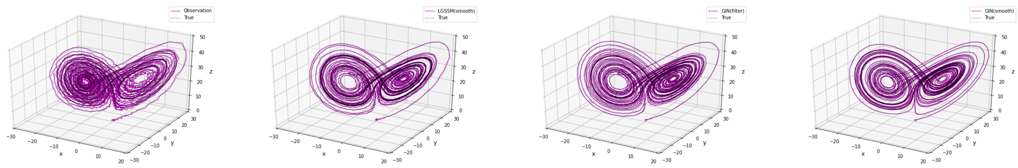


Figure 5: MSE of Lorenz attractor with respect to the training samples.

Due to the non-linearity of the dynamics of this system, LGSSM has to use linearization and then use the linearized dynamics to model the transition. The DSSM model performs better for lager amount of data (>10K) because it needs to learn the dynamics.

Table 5: Comparison of model performance on KITTI dataset. The GIN is performing better than conventional memory cells, while its performance is comparable with DeepVO, a tailored technique for the visual odometry. See 20, 21 and 22 figures in A.5 for the visualization results.

Seq	LSTM		GRU		DeepVO		KVAE		LGSSM		GIN	
	$t_{\text{rel}}(\%)$	$r_{\text{rel}}(^\circ)$	$t_{\text{rel}}(\%)$	$r_{\text{rel}}(^\circ)$	$t_{\text{rel}}(\%)$	$r_{\text{rel}}(^\circ)$	$t_{\text{rel}}(\%)$	$r_{\text{rel}}(^\circ)$	$t_{\text{rel}}(\%)$	$r_{\text{rel}}(^\circ)$	$t_{\text{rel}}(\%)$	$r_{\text{rel}}(^\circ)$
03	8.99	4.55	9.34	3.81	8.49	6.89	12.14	4.38	7.51	3.98	6.98	3.27
04	11.88	3.44	12.36	2.89	7.19	6.97	13.17	4.73	9.12	2.64	9.14	2.28
05	8.96	3.43	10.02	3.43	2.62	3.61	11.47	5.14	6.11	3.21	4.38	2.51
06	9.66	2.8	10.99	3.22	5.42	5.82	10.93	3.98	6.70	3.51	6.14	2.90
07	9.83	5.48	13.70	6.52	3.91	4.60	12.73	4.68	6.59	3.49	7.21	2.98
10	13.58	3.49	13.37	3.25	8.11	8.83	14.79	10.91	9.32	2.90	8.37	2.59
mean	10.53	3.87	11.63	3.85	5.96	6.12	12.53	5.63	7.55	3.28	7.03	2.75



(a) Observation (b) LGSSM(smooth) (c) GIN(filter) (d) GIN(smooth)

Figure 6: Inferred 5k length trajectories for Lorenz attractor.

The results of the Hybrid GNN and the GIN are similar, while the results of the GIN are slightly improved. Although, the core of both models is based on the GRU cell, this enhancement may come from the structure of the GIN that learns the observation and process noises separately. Comparison with LGSSM indicates that applying non-linearity(GRU cells) to the GIN, benefits it in non-linear dynamics. The required numbers of training samples to achieve 0.1 and 0.05 MSE for the GIN are approximately 350 and 3800 samples, respectively. On the other hand, for the hybrid GNN these numbers are a bit higher. Inferred trajectories are in figure 6.

4.2.2 Real World Dynamics: Michigan NCLT dataset

To evaluate the performance of the GIN on a real world dataset, the Michigan NCLT dataset [29] is utilized that encompasses a collection of navigation data gathered by a segwey robot moving inside of the University of Michigan's North Campus. The states in each time, $\mathbf{x}_t \in \mathbb{R}^4$, comprise the position and the velocity in each direction and the observations, $\mathbf{y}_t \in \mathbb{R}^2$, include noisy positions. The ultimate purpose is to localize the real position of the segway robot, while only the noisy GPS observations are available. We apply the GIN to find the current location of the segway robot.

In this experiment, we randomly select the session 2012-01-22 captured in a cloudy situation with the length of 6.1 Km. By sampling with 1Hz and removing the unstable GPS observations, 4280 time steps are achieved and we split the whole sequence into training, testing and validation folds with the length of 3600 (18 sequences of length $T = 200$) , 280 (1 sequence of length $T = 280$) and 400 (2 sequences of length $T = 200$), respectively. For the dynamics of the system, we consider a uniform motion pattern with a constant velocity (see A.7).

Table 6: MSE for NCLT experiment.

Model	MSE[dB]
GIN(smooth)	18.64±0.13
Hybrid GNN	20.73± 0.21
KalmanNet	22.2±0.17
DSSM	29.54±0.58
Vanilla RNN	40.21±0.52
LGSSM	24.38±0.17
Observation	25.47±0.08

The training procedure is completed by maximizing the likelihood with Gaussian distribution assumption. The mean squared error of each approach for the test set are mentioned in the table 6, where the GIN (73.12 ± 2.21 MSE) outperforms other approaches. In summary, this experiment indicates that the GIN can generalize with good performance to a real world dataset.

5 Conclusion

The GIN, an approach for representation learning in both high and low dimensional latent space that uses GRU cell in its structure, is introduced in this paper. The data flow, filtering and smoothing equations follow classic Bayesian, while, due to the usage of a GRU based KG and SG network, the computational issues are tackled resulting in an efficient model with numerical stable results. In the presence of the dynamics, the GIN directly use them, otherwise it directly learns them in an end to end manner, which makes the GIN as a HW model with strong system identification abilities. Due to the design of the GIN, an insightful representation for the uncertainty of the predictions is incorporated in this approach, while it outperforms its counterparts including LSTMs, GRUs and several generative models with variational inferences in both estimation and imputation tasks. We have deliberated the GIN in the exposure of the sequences with few movements in the pixels for imputation task, and leave the extensions of the GIN to real world complex videos to the future works.

Bibliography

[1] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[4] R. Wilson and L. Finkel, "A neural implementation of the kalman filter," *Advances in neural information processing systems*, vol. 22, pp. 2062–2070, 2009.

[5] N. Yadaiah and G. Sowmya, "Neural network based state estimation of dynamical systems," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006, pp. 1042–1049.

[6] E. Archer, I. M. Park, L. Buesing, J. Cunningham, and L. Paninski, "Black box variational inference for state space models," *arXiv preprint arXiv:1511.07367*, 2015.

[7] R. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[8] A. Ghalamzan, K. Nazari, H. Hashempour, and F. Zhong, "Deep-lfd: deep robot learning from demonstrations," *Software Impacts*, vol. 9, p. 100087, 2021.

[9] H. Hashempour, K. Nazari, F. Zhong *et al.*, "A data-set of piercing needle through deformable objects for deep learning from demonstrations," *arXiv preprint arXiv:2012.02458*, 2020.

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

[11] A. Klushyn, R. Kurle, M. Soelch, B. Cseke, and P. van der Smagt, "Latent matters: Learning deep state-space models," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[12] V. Garcia Satorras, Z. Akata, and M. Welling, "Combining generative and discriminative models for hybrid inference," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," *Advances in neural information processing systems*, vol. 31, 2018.

[14] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. van Sloun, and Y. C. Eldar, "Kalmannet: Neural network aided kalman filtering for partially known dynamics," *arXiv preprint arXiv:2107.10043*, 2021.

[15] D. Ruhe and P. Forré, "Self-supervised inference in state-space models," *arXiv preprint arXiv:2107.13349*, 2021.

[16] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *arXiv preprint arXiv:1506.07365*, 2015.

[17] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," in *Advances in neural information processing systems*, 2016, pp. 4376–4384.

[18] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 283–298, 2007.

[19] J. Ko and D. Fox, "Learning gp-bayesfilters via gaussian process latent variable models," *Autonomous Robots*, vol. 30, no. 1, pp. 3–23, 2011.

[20] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in gaussian process state-space models with particle mcmc," *Advances in neural information processing systems*, vol. 26, 2013.

[21] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," *arXiv preprint arXiv:1710.05741*, 2017.

[22] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann, "Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces," in *International Conference on Machine Learning*. PMLR, 2019, pp. 544–552.

[23] M. Schoukens and K. Tiels, "Identification of block-oriented nonlinear systems starting from linear approximations: A survey," *Automatica*, vol. 85, pp. 272–292, 2017.

[24] P. Gilabert, G. Montoro, and E. Bertran, "On the wiener and hammerstein models for power amplifier predistortion," in *2005 Asia-Pacific Microwave Conference Proceedings*, vol. 2. IEEE, 2005, pp. 4–pp.

[25] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[27] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1851–1858.

[28] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[29] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.

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

[31] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The contributions are accurately mentioned.
   (b) Did you describe the limitations of your work? [Yes] In section 5, we clarified the potential limitations and future works.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All assumptions are clearly made in the paper.
   (b) Did you include complete proofs of all theoretical results? [Yes] In the Appendix, we provide additional mathematical explanations.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] It is elaborated in the Appendix A.8.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [No] Whenever we use the existing resources, we cite to the references.

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We have not used.

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The used datasets do not have these kind of issues.

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No] It is not related to human subjects.

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [No] It is not related to human subjects.

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No] It is not related to human subjects.

A Appendix

A.1 Background

Gaussian state space models

In order to model the vectors of time series $\mathbf{w} = \mathbf{w}_{1:T} = [\mathbf{w}_1, ..., \mathbf{w}_T]$, Gaussian state space models (GSSMs) are commonly applied due to their filtering-smoothing ability. In fact, GSSMs model the first-order Markov process on the state space $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_T]$, which can also include the external control input $\mathbf{u} = [\mathbf{u}_1, ..., \mathbf{u}_T]$ by multivariate normality assumption of the state

$$p_{\gamma_t}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}), \quad p_{\gamma_t}(\mathbf{w}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{w}_t; \mathbf{H}_t\mathbf{z}_t, \mathbf{R}). \tag{13}$$

For the cases, which are not controlled via external input, $\mathbf{B}_t$ matrix is simply $\mathbf{0}$ matrix. By Defining γ_t as parameters which explain how the state state changes during the time, it contains the information of $\mathbf{F}_t, \mathbf{B}_t$ and $\mathbf{H}_t$ which are the state transition, control and emission matrices. In each step, the procedure is distorted via $\mathbf{Q}$ and $\mathbf{R}$ that are process noise and observation noise, respectively. It is common to initial the first state $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_0)$, then the joint probability distribution of the GSSM is

$$p_\gamma(\mathbf{w}, \mathbf{x}|\mathbf{u}) = p_\gamma(\mathbf{w}|\mathbf{x})p_\gamma(\mathbf{x}|\mathbf{u}) = \prod_{t=1}^{T} p_{\gamma_t}(\mathbf{w}_t|\mathbf{x}_t).p(\mathbf{x}_1)\prod_{t=2}^{T} p_{\gamma_t}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \tag{14}$$

GSSMs have substantial properties that we can utilize. Filtering and smoothing are among these properties which allow us to obtain the filtered and smoothed posterior based on the priors and observations. By applying classic Bayesian properties, we can have a strong tool to handle the missing data in the image imputation task.

Filtering and Smoothing

The idea of Kalman filter applies two iterative steps, in the former one a prediction is made by the prior state information, while in the latter one an update is done based on the obtained observation. By normality assumption of known additive process and observation noise, the filter can go through the two mentioned steps. In the prediction step, the filter uses the transition matrix $\mathbf{F}$ to estimate the next priors $(\mathbf{x}^-_{t+1}, \Sigma^-_{t+1})$ which are the estimate of the the next states without any observation.

$$\mathbf{x}^-_{t+1} = \mathbf{F}\mathbf{x}^+_t, \text{and} \quad \mathbf{\Sigma}^-_{t+1} = \mathbf{F}\mathbf{\Sigma}^+_t\mathbf{F}^T + \mathbf{Q}, \text{and} \quad \mathbf{Q} = \sigma^2_{\text{trans}}\mathbf{I} \tag{15}$$

In the presence of new observation, the Kalman filter idea goes through the second step and modifies the predicted prior based on the new observation and emission matrix $\mathbf{H}$ that results in the next posterior $(\mathbf{x}^+_{t+1}, \mathbf{\Sigma}^+_{t+1})$.

$$\mathbf{K}_{t+1} = \mathbf{\Sigma}^-_{t+1}\mathbf{H}^T\big(\mathbf{H}\mathbf{\Sigma}^-_{t+1}\mathbf{H}^T + \mathbf{R}\big)^{-1}, \text{and} \tag{16}$$

$$\mathbf{x}^+_{t+1} = \mathbf{x}^-_{t+1} + \mathbf{\Sigma}^-_{t+1}\mathbf{H}^T\big(\mathbf{H}\mathbf{\Sigma}^-_{t+1}\mathbf{H}^T + \mathbf{R}\big)^{-1}(\mathbf{w}_t - \mathbf{H}\mathbf{x}^-_{t+1}) = \mathbf{x}^-_{t+1} + \mathbf{K}_{t+1}(\mathbf{w}_t - \mathbf{H}\mathbf{x}^-_{t+1}), \tag{17}$$

$$\mathbf{\Sigma}^+_{t+1} = \mathbf{\Sigma}^-_{t+1} - \mathbf{\Sigma}^-_{t+1}\mathbf{H}^T\big(\mathbf{H}\mathbf{\Sigma}^-_{t+1}\mathbf{H}^T + \mathbf{R}\big)^{-1}\mathbf{H}\mathbf{\Sigma}^-_{t+1}. \tag{18}$$

The whole observation update procedure can be considered as a weighted mean between the the next prior, that comes from state update, and new observation, where this weighting is a function of $\mathbf{Q}$ and $\mathbf{R}$ that has uncertainty nature.

For the smoothing, the first state of smoothing is initialized with the last state of filtering $(\mathbf{x}_{T|T}, \mathbf{\Sigma}_{T|T}) = (\mathbf{x}^+_T, \mathbf{\Sigma}^+_T)$, then the smoothing procedure is given by:

$$p(\mathbf{x}_t|\mathbf{w}_{1:T}) = \mathcal{N}(\mathbf{x}_{t|T}, \mathbf{\Sigma}_{t|T}), \qquad \mathbf{J}_t = \mathbf{\Sigma}^+_t\mathbf{F}^T_{t+1}[\mathbf{F}_{t+1}\mathbf{\Sigma}^+_t\mathbf{F}^T_{t+1} + \mathbf{Q}_{t+1}]^{-1}$$

$$\mathbf{x}_{t|T} = \mathbf{x}^+_t + \mathbf{J}_t[\mathbf{x}_{t+1|T} - \mathbf{F}_{t+1}\mathbf{x}^+_t], \qquad \mathbf{\Sigma}_{t|T} = \mathbf{\Sigma}^+_t + \mathbf{J}_t\big[\mathbf{\Sigma}_{t+1|T} - [\mathbf{F}_{t+1}\mathbf{\Sigma}^+_t\mathbf{F}^T_{t+1} + \mathbf{Q}_{t+1}]\big]\mathbf{J}^T_t$$

A.2 Process Noise Matrix

As stated in 15, we can elaborate the process noise matrix at time t in more details

$$\mathbf{Q}_t = \mathbf{\Sigma}_t^- - \mathbf{F}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{F}_t^T = \mathbf{\Sigma}_t^- - \mathbf{F}_t \big[\mathbf{\Sigma}_{t-1}^- - \mathbf{K}_{t-1}[\mathbf{H}_{t-1}\mathbf{\Sigma}_{t-1}^-\mathbf{H}_{t-1}^T + \mathbf{R}_{t-1}]^{-1}\mathbf{K}_{t-1}^T\big]\mathbf{F}_t^T \tag{19}$$

combining 15 into 19 results in

$$\begin{aligned}\mathbf{Q}_t = \mathbf{\Sigma}_t^- - \mathbf{F}_t \big[&[\mathbf{F}_{t-1}\mathbf{\Sigma}_{t-2}^+\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}] \\ &-\mathbf{K}_{t-1}[\mathbf{H}_{t-1}[\mathbf{F}_{t-1}\mathbf{\Sigma}_{t-2}^+\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}]\mathbf{H}_{t-1}^T + \mathbf{R}_{t-1}]^{-1}\mathbf{K}_{t-1}^T\big]\mathbf{F}_t^T\end{aligned} \tag{20}$$

which is a function of $\mathbf{F}_t$, $\mathbf{Q}_{t-1}$, $\mathbf{F}_{t-1}$ and $\mathbf{H}_{t-1}$. In the GIN, $\mathbf{F}_t$ and $\mathbf{H}_t$ are learned by the *Dynamics Network* with the input of $\mathbf{x}_{t-1}^+$. From 17, $\mathbf{x}_{t-1}^+$ is derived as a function of both $\mathbf{F}_{t-1}$ and $\mathbf{H}_{t-1}$, meaning the learned $\mathbf{F}_t$ carries the information of both $\mathbf{H}_{t-1}$ and $\mathbf{F}_{t-1}$. Therefore, one can rewrite the equation 20 as

$$\mathbf{Q}_t = \mathbf{g}\Big(\mathbf{F}_t\big(\mathbf{x}_{t-1}^+\big), \mathbf{Q}_{t-1}\Big), \text{where} \quad \mathbf{F}_t = \textit{Dynamics Network}\Big(\mathbf{x}_{t-1}^+\big(\mathbf{H}_{t-1}, \mathbf{F}_{t-1}\big)\Big). \tag{21}$$

where $\mathbf{g}$ is a nonlinear function mapping $\mathbf{F}_t$ and $\mathbf{Q}_{t-1}$ to $\mathbf{Q}_t$. It is possible to go one step further and simplify $\mathbf{x}_{t-1}^+$ more, as it has $\mathbf{\Sigma}_{\mathbf{t-1}}^-$ term in 17, combining it with 15 results in

$$\begin{aligned}\mathbf{x}_{t-1}^+ = \mathbf{x}_{t-1}^- + &[\mathbf{F}_{t-1}\mathbf{\Sigma}_{t-2}^+\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}] \\ &\mathbf{H}_{t-1}^T\big(\mathbf{H}_{t-1}[\mathbf{F}_{t-1}\mathbf{\Sigma}_{t-2}^+\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}]\mathbf{H}_{t-1}^T + \mathbf{R}_{t-1}\big)^{-1}(\mathbf{w}_t - \mathbf{H}_{t-1}\mathbf{x}_{t-1}^-)\end{aligned} \tag{22}$$

indicating that not only $\mathbf{F}_{t-1}$ and $\mathbf{H}_{t-1}$, but also $\mathbf{Q}_{t-1}$ is included in $\mathbf{x}_{t-1}^+$, meaning that $\mathbf{Q}_t$ can be written solely as a function of $\mathbf{F}_t$ or $\mathbf{x}_{t-1}^+$.

$$\mathbf{Q}_t = \mathbf{g}\Big(\mathbf{F}_t\big(\mathbf{x}_{t-1}^+\big)\Big), \text{where} \quad \mathbf{F}_t = \textit{Dynamics Network}\Big(\mathbf{x}_{t-1}^+\big(\mathbf{H}_{t-1}, \mathbf{F}_{t-1}, \mathbf{Q}_{t-1}\big)\Big). \tag{23}$$

A.3 Output Distribution

In the case of grayscale images, consider each pixel, y_i, is one or zero with the probability of p_i or $1 - p_i$ respectively, meaning that $P(Y = y) = p^y(1-p)^{1-y}$. By re-writing the probability equation into the exponential families form

$$f_\theta(y) = h(y).exp\big(\theta.y - \psi(\theta)\big) \rightarrow e^{log(p^y(1-p)^{1-y})} = e^{y\, log(\frac{p}{1-p}) + log(1-p)} \tag{24}$$

and by choosing $\theta = log(\frac{p}{1-p})$ and $\psi(\theta) = log(1-p)$, we can obtain $p = \frac{1}{1+e^{-\theta}}$. It means that by considering θ as the last layer of the decoder and applying a softmax layer, p is obtained. Equivalently, one can calculate the deviance between real p and estimation of it, $\hat{p}$, which is given by

$$D(p, \hat{p}) = \big[p\, log(\frac{p}{\hat{p}}) + (1-p)log(\frac{1-p}{1-\hat{p}})\big] \tag{25}$$

and minimize the deviance with respect to $\hat{p}$ as we did in 12.

Similarly, consider x, $\hat{x}_\theta$ and θ as the ground truth state, estimated state and the model variables respectively, where the residual follows Gaussian distribution $x = \hat{x}_\theta + \epsilon \sim \mathcal{N}(\hat{x}_\theta, \hat{\sigma}_\theta)$, where $\hat{\sigma}_\theta$ is the estimated variance. Then, the negative log likelihood is given by 26 as we obtained it in 11.

$$-log(\mathcal{L}) \propto \frac{1}{2}log(\hat{\sigma}_\theta) + \frac{(x - \hat{x}_\theta)^2}{2\hat{\sigma}_\theta} \tag{26}$$

A.4 Noise Generation Process

In the high dimensional observation experiments, to make our results comparable with those of the RKN [22], we use the same observation noise process as they did. It makes the noise factors correlated over time by introducing a sequence of factors f_t of the same length of the data sequence. Let $f_0 \sim \mathcal{U}(0,1)$ and $f_{t+1} = \min(\max(0, f_t + r_t), 1)$ with $r_t \sim \mathcal{U}(-0.2, 0.2)$, where f_0 is the initialized factor and $\mathcal{U}$ is the uniform distribution. Then by defining two thresholds, $t_1 \sim \mathcal{U}(0, 0.25)$ and $t_2 \sim \mathcal{U}(0.75, 1)$, $f_t < t_1$ are set to 0 and $f_t > t_2$ are set to 1 and the rest are splitted linearly within the range of $[0,1]$. The t-th obtained observation is given by $\mathbf{o}_t = f_t\mathbf{i}_t + (1-f_t)\mathbf{i}_t^{pn}$, where the $\mathbf{i}_t$ is the t-th true image and $\mathbf{i}_t^{pn}$ is the t-th generated pure noise.

A.5 Visualization for The Imputation Experiment

Graphical results of informed, uninformed and noisy observations for image imputation task for both single and double pendulum experiments can be found in 7, 8, 9 and 10 figures. Inference for the trained smoothened distribution of all high dimensional experiments are in 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21 and 22 figures. The results of NCLT experiment are in 23.

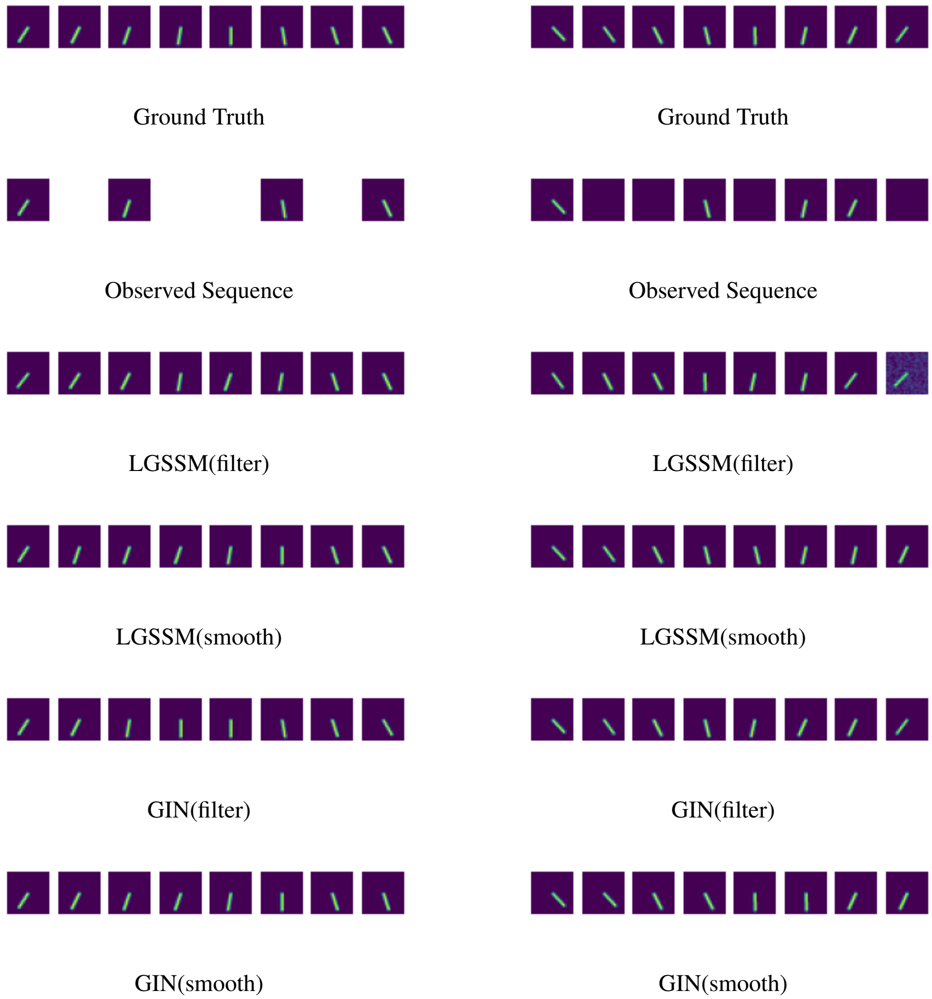


Figure 7: Informed(left column) and uninformed(right column) image imputation task for the single pendulum experiments.

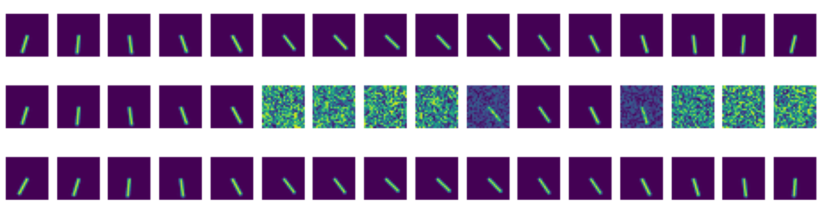


Figure 8: Image imputation task for the single pendulum experiment exposed to the noisy observations, where the generated noise has correlation with the time. Each figure, beginning from top to bottom, indicates the ground truth, noisy observation and the imputation results of the GIN.

Figure 9: Informed(left column) and uninformed(right column) image imputation task for the double pendulum experiments.

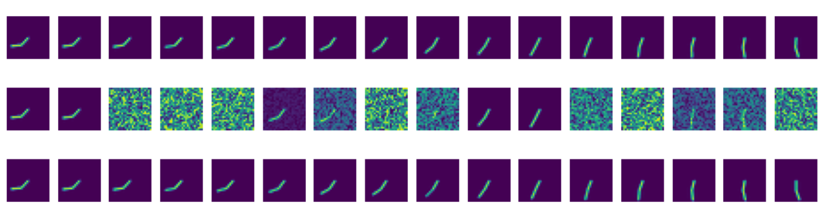


Figure 10: Image imputation task for the double pendulum experiment exposed to the noisy observations, where the generated noise has correlation with the time. Each figure, beginning from top to bottom, indicates the ground truth, noisy observation and the imputation results of the GIN.

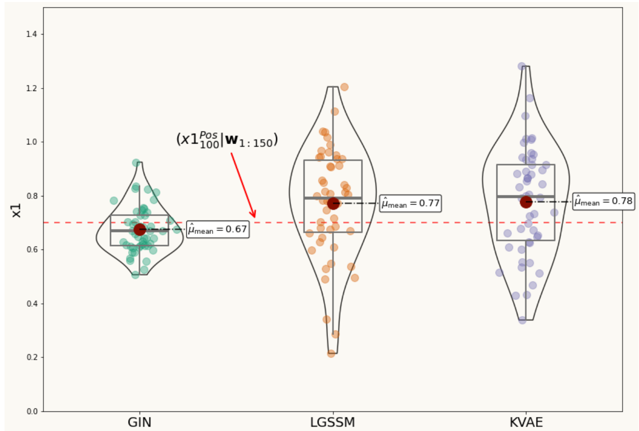


Figure 11: Inference for the single pendulum $x1$ position at 100-th time step. Generated samples from smoothened distribution, $f(x1_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1^{\text{Pos}}_{100}|\mathbf{w}_{1:150})$ is the ground truth state with distribution of $\delta(x1_{100} - 0.7)$. We calculate the sample mean and fit a distribution on the samples for further visualization and comparison purpose.

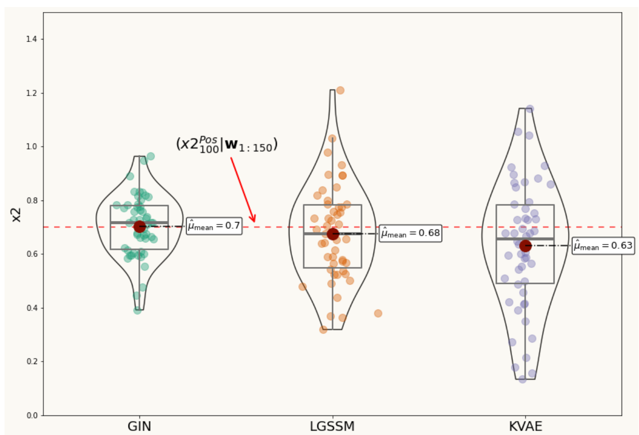
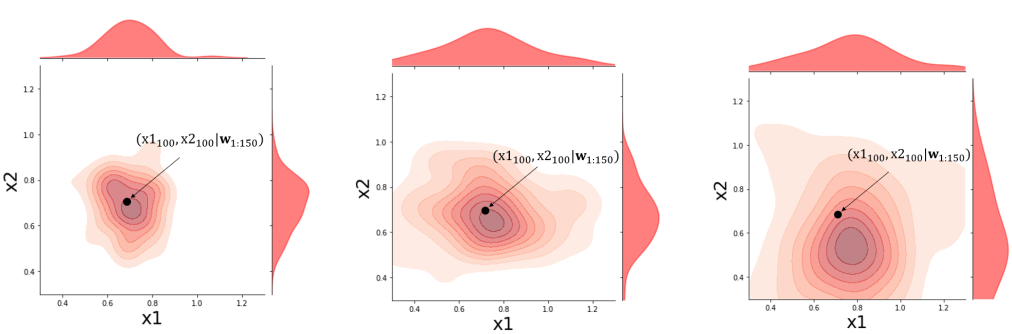


Figure 12: Inference for the single pendulum $x2$ position at 100-th time step. Generated samples from smoothened distribution, $f(x2_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively.



(a) GIN

(b) LGSSM

(c) KVAE

Figure 13: Generated samples from the trained smoothened joint distribution of the single pendulum position, $(x1, x2)$, at 100-th time step for the GIN, LGSSM and KVAE, respectively. The ground truth is shown with a black point.

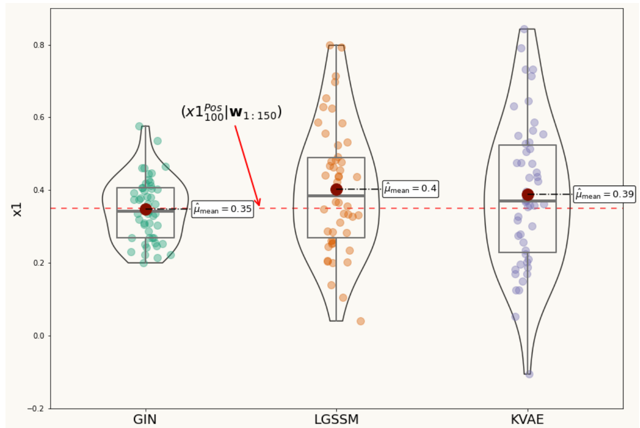


Figure 14: Inference for the double pendulum $x1$ position at 100-th time step. Generated samples from smoothened distribution, $f(x1_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1_{100}^{\text{Pos}}|\mathbf{w}_{1:150})$ is the ground truth state with distribution of $\delta(x1_{100} - 0.35)$.

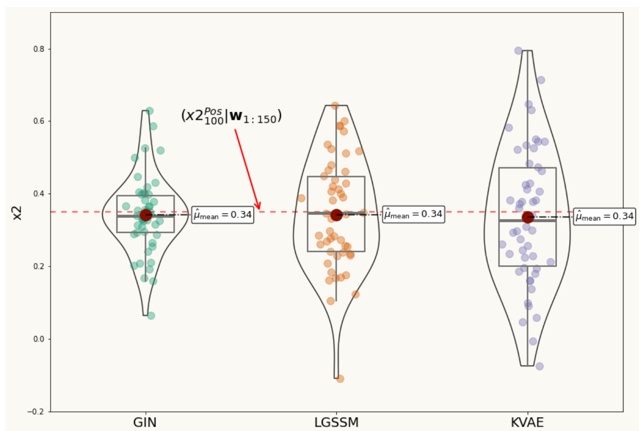
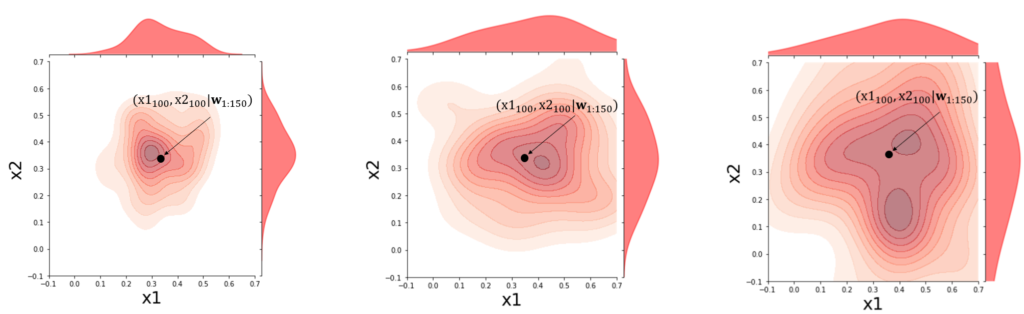


Figure 15: Inference for the double pendulum $x2$ position at 100-th time step. Generated samples from smoothened distribution, $f(x2_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x2_{100}^{\text{Pos}}|\mathbf{w}_{1:150})$ is the ground truth state with distribution of $\delta(x2_{100} - 0.35)$.



(a) GIN

(b) LGSSM

(c) KVAE

Figure 16: Generated samples from the trained smoothened joint distribution of the double pendulum first joint position, $(x1, x2)$, at 100-th time step for the GIN, LGSSM and KVAE, respectively. The ground truth is shown with a black point.

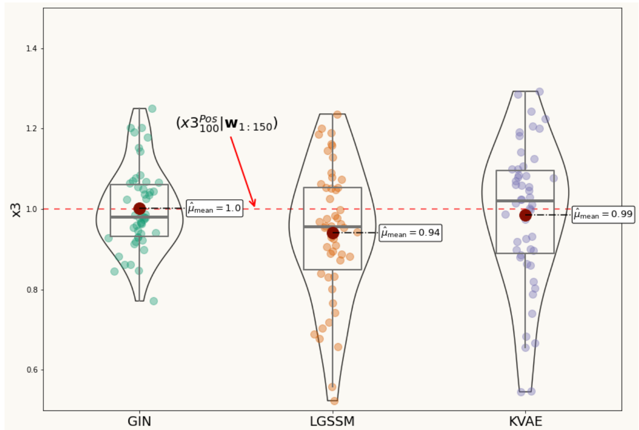


Figure 17: Inference for the double pendulum $x3$ position at 100-th time step. Generated samples from smoothened distribution, $f(x3_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x3^{\text{Pos}}_{100}|\mathbf{w}_{1:150})$ is the ground truth state with distribution of $\delta(x3_{100}-1)$.

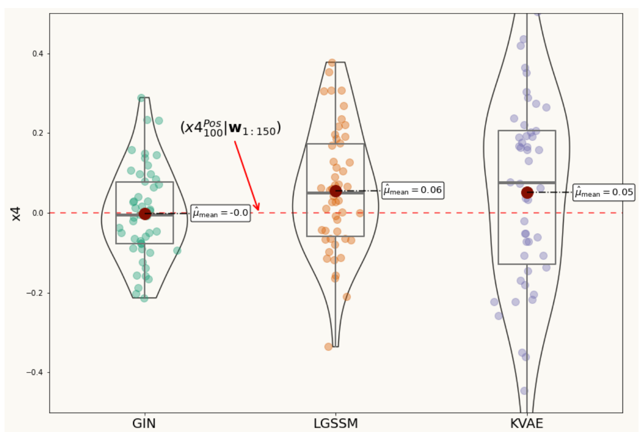
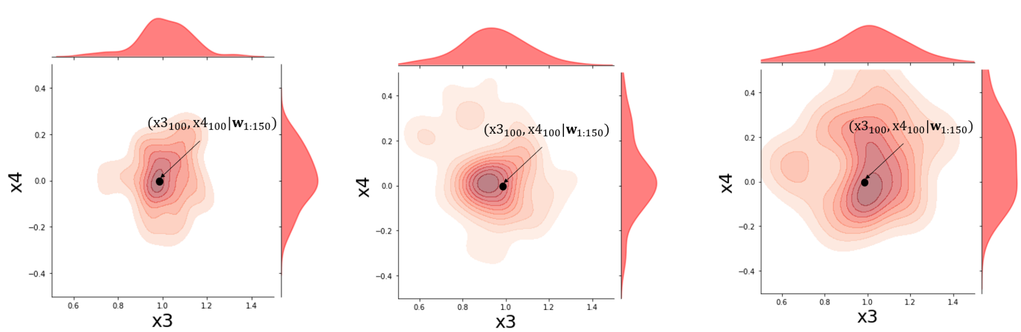


Figure 18: Inference for the double pendulum $x4$ position at 100-th time step. Generated samples from smoothened distribution, $f(x4_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively.



(a) GIN

(b) LGSSM

(c) KVAE

Figure 19: Generated samples from the trained smoothened joint distribution of the double pendulum second joint position, $(x3, x4)$, at 100-th time step for the GIN, LGSSM and KVAE, respectively. The ground truth is shown with a black point.

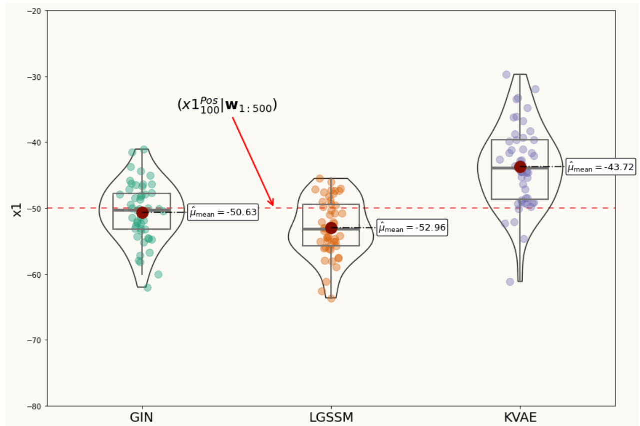


Figure 20: Inference for the visual odometry $x1$ position at 100-th time step. Generated samples from smoothened distribution, $f(x1_{100}|\mathbf{w}_{1:500})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1^{\text{Pos}}_{100}|\mathbf{w}_{1:500})$ is the ground truth state with distribution of $\delta(x1_{100} + 50)$.

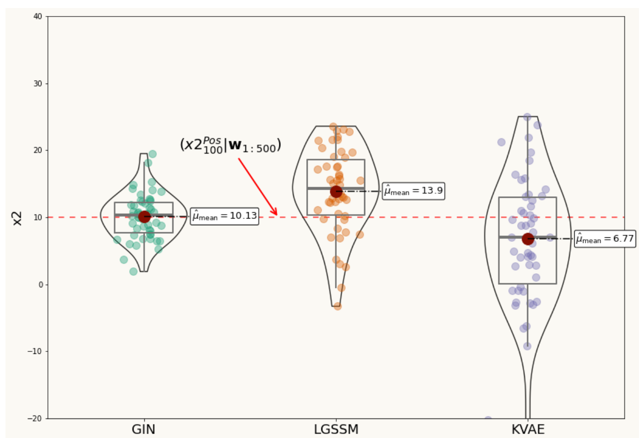
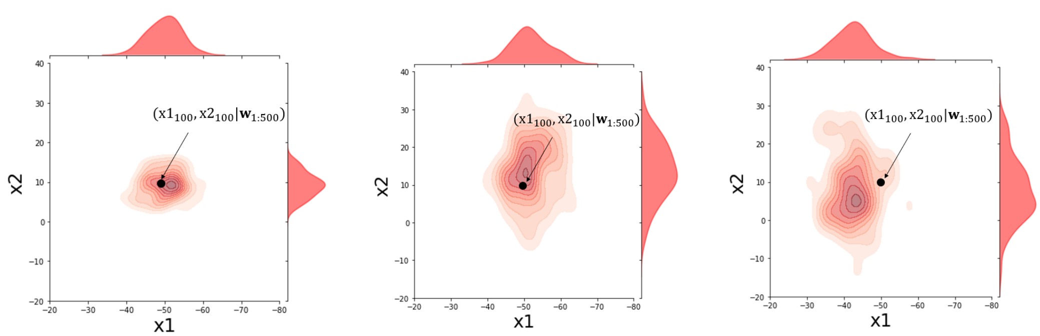


Figure 21: Inference for the visual odometry $x2$ position at 100-th time step. Generated samples from smoothened distribution, $f(x2_{100}|\mathbf{w}_{1:500})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x2^{\text{Pos}}_{100}|\mathbf{w}_{1:500})$ is the ground truth state with distribution of $\delta(x1_{100} - 10)$.



(a) GIN

(b) LGSSM

(c) KVAE

Figure 22: Generated samples from the trained smoothened joint distribution of the visual odometry joint position, $(x1, x2)$, at 100-th time step for the GIN, LGSSM and KVAE, respectively. The ground truth is shown with a black point.

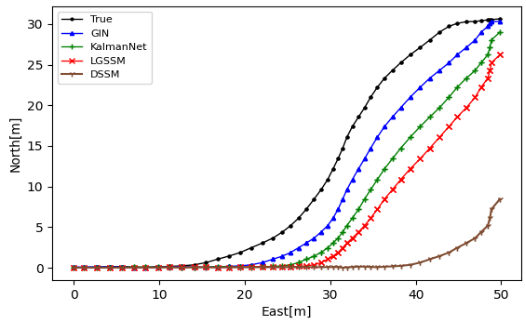


Figure 23: NCLT dataset position for the first 50 observations: ground truth positions and the generated trajectories with the GIN, LGSSM, KalmanNet and DSSM approaches are illustrated.

A.6 Lorenz Attractor Dynamics

There are three differential equations that model a Lorenz system, x the convection rate, y the horizontal temperature variation and z the vertical temperature variation.

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z \tag{27}$$

where the constant values σ, ρ and β are 10, 28 and $-\frac{8}{3}$, respectively. To construct a trajectory we use Lorenz system equations 27 with $dt = 10^{-5}$, then we sample from it with the step time of $\Delta t = 0.01$. Based on the equations of the system 27, the state is $\mathbf{s_t} = [x_t, y_t, z_t]$ and we can write the dynamics of the system as $\mathbf{A_t}$ and obtain the transition matrix $\text{Exp}[\mathbf{A_t}] = \mathbf{F_t}$. To achieve this, we use the Taylor expansion of Exp function with 5 degrees.

$$\dot{\mathbf{s}}_t = \mathbf{A_t}\mathbf{s_t} = \begin{bmatrix} -10 & 10 & 0 \\ 28 - z & -1 & 0 \\ y & 0 & -\frac{8}{3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \text{and} \quad \mathbf{F_t} = \text{Exp}[\mathbf{A_t}] = \mathbf{I} + \sum_{j=1}^{J} \frac{(\mathbf{A_t}.\Delta t)^j}{j!} \tag{28}$$

where J is the degrees of expansion and $\mathbf{I}$ is the identity matrix. For the emission matrix we use $\mathbf{H_t} = \mathbf{I}$ and for process and observation noise standard deviation, we use $\mathbf{Q_t} = \frac{1}{100}\sigma^2\mathbf{I}$ and $\mathbf{R_t} = \sigma^2\mathbf{I}$, respectively.

A.7 Movement Model Details for The NCLT Experiment

We assume that the segway robot is moved with a constant velocity, that the equations for such dynamics are given by

$$\frac{\partial p_1}{\partial t} = v_1, \quad \frac{\partial p_2}{\partial t} = v_2, \quad \frac{\partial v_1}{\partial t} = 0, \quad \frac{\partial v_2}{\partial t} = 0, \quad \mathbf{x}_t = [p_1, v_1, p_2, v_2], \quad \mathbf{y}_t = [p_1, p_2]. \tag{29}$$

By such assumptions for the motion's equations the transition, process noise distribution, emission and measurement noise distribution matrices can be obtained by

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \sigma^2 \begin{bmatrix} \Delta t & 0 & 0 & 0 \\ 0 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{R} = \lambda^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{30}$$

where $\Delta t = 1$ since the sampling frequency is 1Hz. Process and measurement variance parameters, σ and λ, are unknown that the model will learn them.

A.8 Network Structure and Parameters

In all experiments, Adam optimizer [30] has been used on NVIDIA GeForce GTX 1050 Ti. We conduct a grid search for finding the hyperparameters to rule out the possibility of the models being trained with the suboptimal hyperparameters. To find the initial learning rate, by conducting a grid search between 0.001 and 0.2 with the increment of 0.005, we select the best one among them that corresponds to the highest log-likelihood. With an initial learning rate of 0.006 and an exponential decay with rate of 0.9 every 10 epochs, we employ back propagation through time [25] to compute the gradients as we deploy GRU cells in the structure. Layer normalization technique [31] is used to stabilize the dynamics in the recurrent structure and normalize the filter response. Elu + 1 activation function, can ensure the positiveness of the diagonal elements of the process, noise and covariance matrices.

In order to prevent the model being stuck in the poor local minima, e.g. focusing on the reconstruction instead of learning the dynamics obtained by filtering-smoothing, we find it useful to use two training tricks for an end-to-end learning:

- Generating time correlated noisy sequences as consecutive observations, forces the model to learn the dynamics instead of focusing on reconstruction, e.g. figure 8 and 10.
- For the first few epochs, only learn auto-encoder(MLPs) and globally learned parameters, e.g. $\mathbf{F}^{(k)}$ and $\mathbf{H}^{(k)}$, but not *Dynamics Network* parameters $\alpha_t(\mathbf{x}_{t-1})$. All the parameters are jointly learned, afterwards. This allows the system to learn good embedding and meaningful latent vectors at first, then learns how to employ K different dynamics variables.

In the lack of dynamics, for the low dimensional observations we use $K = 5$, while for the high dimensional observations we use $K = 15$ as they need to learn more complex dynamics. In general, if the GIN is flexible enough, tuning the parameters is not difficult as the GIN is capable to learn how to prune unused elements by the *Dynamics Network*.

A.8.1 Empirical running times and parameters

We present the number of parameters of the utilized cell structures in our experiments and their corresponding empirical running times for 1 epoch in the table 7 and 8. In the first row of each model structure in the high dimensional experiments, we set the number of their parameters approximately equal to our GIN to indicate the outperformance of the GIN with the same number of the parameters, i.e. tables 1, 2 and 5, while we include the empirical running time and parameters for more complex structures as well.

Table 7: Empirical running times and parameters of high-low dimensional experiments.

Cell	Single Pend		Double Pend		KITTI	
	Param	T/E	Param	T/E	Param	T/E
LSTM (units=25)	$\sim$18k	$\sim$56s	$\sim$18k	$\sim$56s	$\sim$45k	$\sim$83s
LSTM (units=50)	$\sim$36k	$\sim$70s	$\sim$36k	$\sim$71s	$\sim$70k	$\sim$95s
LSTM (units=100)	$\sim$76k	$\sim$98s	$\sim$76k	$\sim$96s	$\sim$120k	$\sim$131s
GRU (units=30)	$\sim$18k	$\sim$61s	$\sim$18k	$\sim$62s	$\sim$42k	$\sim$79s
GRU (units=50)	$\sim$27k	$\sim$65s	$\sim$27k	$\sim$67s	$\sim$53k	$\sim$84s
GRU (units=100)	$\sim$57k	$\sim$86s	$\sim$57k	$\sim$85s	$\sim$90k	$\sim$111s
KVAE (n=40)	$\sim$25k	$\sim$95s	$\sim$25k	$\sim$97s	$\sim$62k	$\sim$141s
KVAE (n=60)	$\sim$36k	$\sim$114s	$\sim$36k	$\sim$111s	$\sim$80k	$\sim$165s
RKN (n=100)	$\sim$25k	$\sim$57s	$\sim$25k	$\sim$58s	$\sim$45k	$\sim$79s
LGSSM (n=30)	$\sim$12k	$\sim$82s	$\sim$12k	$\sim$84s	$\sim$30k	$\sim$117s
LGSSM (n=45)	$\sim$15k	$\sim$98s	$\sim$15k	$\sim$97s	$\sim$36k	$\sim$136s
GIN (n=30)	$\sim$18k	$\sim$55s	$\sim$18k	$\sim$55s	$\sim$42k	$\sim$80s
GIN (n=45)	$\sim$25k	$\sim$59s	$\sim$25k	$\sim$58s	$\sim$48k	$\sim$83s

Table 8: Low dimensional experiments.

Cell	Lorenz		NCLT	
	Param	T/E	Param	T/E
KalmanNet			$\sim$30k	$\sim$65s
GNN	$\sim$10k	$\sim$35s	$\sim$10k	$\sim$40s
RNN			$\sim$40k	$\sim$79s
DSSM	$\sim$40k	$\sim$76s	$\sim$40k	$\sim$81s
LGSSM	$\sim$6k	$\sim$20s	$\sim$6k	$\sim$22s
GIN	$\sim$9k	$\sim$28s	$\sim$9k	$\sim$31s

A.8.2 Single-Pendulum and Double-Pendulum Experiments

Data Generation.

Dataset consists of 1000 train, 100 valid and 100 test sequences with the length of 150. The sequences are distorted via generated noise, while in the informed imputation task half of the images are removed and boolean flags indicating the availability of the observations are passed to the cell instead. If the imputation task is in uninformed type, black images are considered as the observations instead of informing the cell with boolean flags.

Table 9: The structure of the encoder and decoder for single and double pendulum experiments.

Encoder	Decoder
6×6 Conv, 12, max pooling 2×2, stride 2×2	$\mathbf{o}_x^+$: fully connected: *out dim*, linear activation
LayerNormalizer()	$\mathbf{o}_\Sigma^+$: fully connected, Elu + 1 activation
4×4 Conv, 12, max pooling 2×2, stride 2×2	if *imputation task*:
LayerNormalizer()	fully connected: 144
fully connected: 40	6×6 Trns Conv, 16, stride 4×4
$\mathbf{w}$: fully connected: m, linear activation	LayerNormalizer()
$\mathbf{r}$: fully connected, Elu + 1 activation	4×4 Trns Conv, 12, stride 2×2
	LayerNormalizer()
	$\mathbf{o}_i^+$: 1×1 Trns Conv, stride 1×1, softmax

Encoder/Decoder and the Dynamics Network Architecture

To design the dynamics network, we use a MLP including 60 hidden units with Relu activation function and a softmax activation for the last layer. The state mean, with size of n, and number of the bases, with size of k, are the input and output of the dynamics, respectively. The structures of the encoder and decoder are in the table 9. In the table 9, m is the latent observation dimension that various values for this parameter are taken into account in the results. In the state estimation tasks, *out dim* is 4 and 8 for the single-pendulum and double-pendulum experiment, respectively. For the imputation task, number of the hidden units of the KG and SG network is set to 40 and 30, respectively. The convolutional layer applied over the covariance matrix has 8 filters with kernel size of 5.

A.8.3 Lorenz Attractor and NCLT Experiments

In these two experiments that we have the knowledge of the dynamics, we employ a fully connected with the observations as its input and output dimension of 3 and 2 for Lorenz attractor and NCLT experiments, respectively, to obtain the observation noise, $\mathbf{r}$. The activation function is Elu + 1. Similarly another fully connected with the posterior state as its input and output dimension of 3 and 2 for Lorenz attractor and NCLT experiments, respectively, to attain the uncertainty estimates, $\mathbf{o}_\sigma^+$. To estimate the process noise matrix, a fully connect with the posterior state as the input and Elu + 1 activation function is used. Similarly, a GRU cell that maps the posterior states to the process noise matrix with 10 hidden units can be used.

A.9 MSE Results for The State Estimation Task

The MSE results for the single and double pendulum experiments are in the table 10 and 11. In addition to 4, where $\mathbf{F}$ matrix includes the effects of the process noise, two other mentioned solutions introduced in 3.2, are included in the MSE results as well. Using GRU cell and MLP for mapping $\mathbf{x}^+$, as their input, to $\mathbf{Q}$, as their output, where the former one is shown by GRU($\mathbf{Q}$) and the latter one by MLP($\mathbf{Q}$) in the tables.

Table 10: MSE for single pendulum experiment.

Model	MSE
LSTM (units = 25, $m = 15$)	0.092±0.003
LSTM (units = 25, $m = 20$)	0.092±0.005
LSTM (units = 25, $m = 40$)	0.090±0.005
LSTM (units = 100, $m = 15$)	0.089±0.002
LSTM (units = 100, $m = 20$)	0.089±0.005
LSTM (units = 100, $m = 40$)	0.090±0.004
GRU (units = 30, $m = 15$)	0.095±0.006
GRU (units = 30, $m = 20$)	0.093±0.002
GRU (units = 30, $m = 40$)	0.094±0.005
GRU (units = 100, $m = 15$)	0.091±0.002
GRU (units = 100, $m = 20$)	0.092±0.004
GRU (units = 100, $m = 40$)	0.091±0.008

Model	$\mathbf{F}(\mathbf{Q})$	MLP($\mathbf{Q}$)	GRU($\mathbf{Q}$)
LGSSM filter($m = 15$, $n = 30$, $K = 10$)	0.089±0.009	0.088±0.005	0.088±0.006
LGSSM filter($m = 15$, $n = 30$, $K = 15$)	0.088±0.011	0.087±0.007	0.086±0.004
LGSSM filter($m = 15$, $n = 45$, $K = 10$)	0.085±0.004	0.084±0.007	0.084±0.009
LGSSM filter($m = 15$, $n = 45$, $K = 15$)	0.084±0.005	0.083±0.004	0.082±0.004
LGSSM filter($m = 20$, $n = 40$, $K = 10$)	0.084±0.009	0.082±0.014	0.082±0.011
LGSSM filter($m = 20$, $n = 40$, $K = 15$)	0.083±0.012	0.081±0.005	0.080±0.014
LGSSM filter($m = 20$, $n = 60$, $K = 10$)	0.079±0.005	0.078±0.012	0.076±0.009
LGSSM filter($m = 20$, $n = 60$, $K = 15$)	0.077±0.006	0.075±0.011	0.074±0.008
LGSSM smooth($m = 15$, $n = 30$, $K = 10$)	0.086±0.011	0.083±0.004	0.084±0.007
LGSSM smooth($m = 15$, $n = 30$, $K = 15$)	0.085±0.012	0.084±0.008	0.083±0.012
LGSSM smooth($m = 15$, $n = 45$, $K = 10$)	0.081±0.008	0.080±0.009	0.079±0.003
LGSSM smooth($m = 15$, $n = 45$, $K = 15$)	0.081±0.014	0.078±0.007	0.077±0.011
LGSSM smooth($m = 20$, $n = 40$, $K = 10$)	0.082±0.005	0.078±0.004	0.076±0.013
LGSSM smooth($m = 20$, $n = 40$, $K = 15$)	0.080±0.003	0.076±0.006	0.074±0.010
LGSSM smooth($m = 20$, $n = 60$, $K = 10$)	0.076±0.008	0.073±0.002	0.070±0.009
LGSSM smooth($m = 20$, $n = 60$, $K = 15$)	0.073±0.013	0.071±0.011	0.068±0.013
GIN filter($m = 15$, $n = 30$, $K = 10$)	0.078±0.013	0.076±0.005	0.075±0.004
GIN filter($m = 15$, $n = 30$, $K = 15$)	0.078±0.014	0.075±0.009	0.074±0.012
GIN filter($m = 15$, $n = 45$, $K = 10$)	0.074±0.010	0.073±0.008	0.072±0.009
GIN filter($m = 15$, $n = 45$, $K = 15$)	0.073±0.015	0.074±0.011	0.071±0.005
GIN filter($m = 20$, $n = 40$, $K = 10$)	0.072±0.005	0.072±0.008	0.070±0.002
GIN filter($m = 20$, $n = 40$, $K = 15$)	0.071±0.007	0.071±0.004	0.071±0.009
GIN filter($m = 20$, $n = 60$, $K = 10$)	0.067±0.009	0.066±0.005	0.065±0.006
GIN filter($m = 20$, $n = 60$, $K = 15$)	0.065±0.013	0.064±0.009	0.063±0.010
GIN smooth($m = 15$, $n = 30$, $K = 10$)	0.071±0.007	0.070±0.003	0.068±0.009
GIN smooth($m = 15$, $n = 30$, $K = 15$)	0.070±0.008	0.068±0.011	0.068±0.007
GIN smooth($m = 15$, $n = 45$, $K = 10$)	0.065±0.011	0.065±0.009	0.064±0.012
GIN smooth($m = 15$, $n = 45$, $K = 15$)	0.064±0.008	0.066±0.007	0.063±0.009
GIN smooth($m = 20$, $n = 40$, $K = 10$)	0.064±0.005	0.065±0.003	0.062±0.008
GIN smooth($m = 20$, $n = 40$, $K = 15$)	0.063±0.004	0.064±0.011	0.063±0.007
GIN smooth($m = 20$, $n = 60$, $K = 10$)	0.059±0.009	0.061±0.012	0.057±0.006
GIN smooth($m = 20$, $n = 60$, $K = 15$)	0.058±0.005	0.057±0.009	0.056±0.004

Table 11: MSE for double pendulum experiment.

Model	MSE
LSTM (units = 50, $m = 15$)	0.172±0.012
LSTM (units = 50, $m = 20$)	0.166±0.009
LSTM (units = 50, $m = 40$)	0.167±0.011
LSTM (units = 100, $m = 15$)	0.164±0.006
LSTM (units = 100, $m = 20$)	0.162±0.009
LSTM (units = 100, $m = 40$)	0.159±0.010
GRU (units = 50, $m = 15$)	0.194±0.014
GRU (units = 50, $m = 20$)	0.189±0.013
GRU (units = 50, $m = 40$)	0.188±0.015
GRU (units = 100, $m = 15$)	0.173±0.009
GRU (units = 100, $m = 20$)	0.169±0.014
GRU (units = 100, $m = 40$)	0.166±0.018

Model	$\mathbf{F}(\mathbf{Q})$	MLP($\mathbf{Q}$)	GRU($\mathbf{Q}$)
LGSSM filter($m = 15$, $n = 30$, $K = 10$)	0.154±0.013	0.159±0.021	0.153±0.009
LGSSM filter($m = 15$, $n = 30$, $K = 15$)	0.152±0.008	0.153±0.010	0.152±0.012
LGSSM filter($m = 15$, $n = 45$, $K = 10$)	0.144±0.011	0.141±0.015	0.139±0.013
LGSSM filter($m = 15$, $n = 45$, $K = 15$)	0.142±0.007	0.138±0.012	0.137±0.017
LGSSM filter($m = 20$, $n = 40$, $K = 10$)	0.144±0.012	0.137±0.009	0.138±0.013
LGSSM filter($m = 20$, $n = 40$, $K = 15$)	0.141±0.007	0.137±0.008	0.136±0.016
LGSSM filter($m = 20$, $n = 60$, $K = 10$)	0.129±0.009	0.126±0.014	0.122±0.015
LGSSM filter($m = 20$, $n = 60$, $K = 15$)	0.127±0.012	0.124±0.013	0.119±0.009
LGSSM smooth($m = 15$, $n = 30$, $K = 10$)	0.147±0.009	0.148±0.014	0.144±0.015
LGSSM smooth($m = 15$, $n = 30$, $K = 15$)	0.146±0.014	0.146±0.013	0.142±0.017
LGSSM smooth($m = 15$, $n = 45$, $K = 10$)	0.139±0.017	0.136±0.009	0.133±0.017
LGSSM smooth($m = 15$, $n = 45$, $K = 15$)	0.137±0.009	0.135±0.017	0.133±0.012
LGSSM smooth($m = 20$, $n = 40$, $K = 10$)	0.136±0.014	0.131±0.022	0.132±0.011
LGSSM smooth($m = 20$, $n = 40$, $K = 15$)	0.134±0.011	0.129±0.014	0.129±0.022
LGSSM smooth($m = 20$, $n = 60$, $K = 10$)	0.123±0.019	0.116±0.016	0.115±0.013
LGSSM smooth($m = 20$, $n = 60$, $K = 15$)	0.120±0.010	0.112±0.009	0.108±0.014
GIN filter($m = 15$, $n = 30$, $K = 10$)	0.126±0.014	0.125±0.012	0.125±0.011
GIN filter($m = 15$, $n = 30$, $K = 15$)	0.124±0.015	0.124±0.019	0.121±0.009
GIN filter($m = 15$, $n = 45$, $K = 10$)	0.115±0.011	0.114±0.015	0.110±0.017
GIN filter($m = 15$, $n = 45$, $K = 15$)	0.114±0.019	0.112±0.020	0.110±0.011
GIN filter($m = 20$, $n = 40$, $K = 10$)	0.113±0.013	0.111±0.009	0.111±0.013
GIN filter($m = 20$, $n = 40$, $K = 15$)	0.111±0.009	0.109±0.018	0.108±0.009
GIN filter($m = 20$, $n = 60$, $K = 10$)	0.099±0.018	0.094±0.017	0.095±0.021
GIN filter($m = 20$, $n = 60$, $K = 15$)	0.097±0.009	0.093±0.009	0.091±0.008
GIN smooth($m = 15$, $n = 30$, $K = 10$)	0.115±0.011	0.116±0.009	0.113±0.017
GIN smooth($m = 15$, $n = 30$, $K = 15$)	0.113±0.015	0.113±0.018	0.112±0.014
GIN smooth($m = 15$, $n = 45$, $K = 10$)	0.105±0.009	0.101±0.014	0.101±0.015
GIN smooth($m = 15$, $n = 45$, $K = 15$)	0.102±0.013	0.100±0.011	0.098±0.008
GIN smooth($m = 20$, $n = 40$, $K = 10$)	0.101±0.008	0.098±0.010	0.094±0.015
GIN smooth($m = 20$, $n = 40$, $K = 15$)	0.098±0.017	0.095±0.014	0.092±0.007
GIN smooth($m = 20$, $n = 60$, $K = 10$)	0.086±0.013	0.081±0.008	0.079±0.009
GIN smooth($m = 20$, $n = 60$, $K = 15$)	0.083±0.009	0.079±0.006	0.076±0.013

A.10 Algorithms

Algorithm High-Dimensional Observations Training

Input: Ground Truth $\mathbf{gt}_{1:T}$, Observations $\mathbf{o}_{1:T}$, last posteriors $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$, initial posterior $(\mathbf{x}^+_0, \mathbf{\Sigma}^+_0)$
$\alpha_{1:T}$ = *Dynamics Network* $(\mathbf{x}^+_{0:T-1})$
Obtain $\mathbf{F}_{1:T}(\mathbf{Q})$ and $\mathbf{H}_{1:T}$ by 2
$(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T})$ = *Prediction Step* $((\mathbf{x}^+_{0:T-1}, \mathbf{\Sigma}^+_{0:T-1}), \mathbf{F}_{1:T}(\mathbf{Q}))$
$(\mathbf{w}_{1:T}, \mathbf{r}_{1:T})$ = *encoder* $(\mathbf{o}_{1:T})$
$\mathbf{K}_{1:T}$ = GRU^{KG} (*Conv2D*($\mathbf{\Sigma}^-_{1:T}$), $\mathbf{r}_{1:T}$)
$\mathbf{J}_{1:T}$ = GRU^{SG} (*Conv2D*($\mathbf{\Sigma}^-_{1:T}$))
$(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$ = *Filtering Step* $(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T}, \mathbf{K}_{1:T}, \mathbf{w}_{1:T}, \mathbf{H}_{1:T})$
$(\mathbf{x}_{1:T|T}, \mathbf{\Sigma}_{1:T|T})$ = *Smoothing Step* $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T}, \mathbf{J}_{1:T}, \mathbf{F}_{1:T}(\mathbf{Q}))$
$\mathbf{o}^+_{1:T}$ = *decoder* $(\mathbf{x}_{1:T|T}, \mathbf{\Sigma}_{1:T|T})$
$\mathcal{L}_{1:T}$ = - Likelihood $(\mathbf{gt}_{1:T}, \mathbf{o}^+_{1:T})$
Backward Propagation ()

Algorithm Low-Dimensional Observations Training

Input: Ground Truth $\mathbf{gt}_{1:T}$, Observations $\mathbf{y}_{1:T}$, last posteriors $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$, initial posterior $(\mathbf{x}^+_0, \mathbf{\Sigma}^+_0)$
if Dynamics are not known **then**
  $\alpha_{1:T}$ = *Dynamics Network* $(\mathbf{x}^+_{0:T-1})$
  Obtain $\mathbf{F}_{1:T}(\mathbf{Q})$ and $\mathbf{H}_{1:T}$ by 2
  $(\mathbf{w}_{1:T}, \mathbf{r}_{1:T})$ = MLP $(\mathbf{y}_{1:T})$
  $(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T})$ = *Prediction Step* $(\mathbf{x}^+_{0:T-1}, \mathbf{\Sigma}^+_{0:T-1}, \mathbf{F}_{1:T}(\mathbf{Q}))$
  $\mathbf{K}_{1:T}$ = GRU^{KG} $(\mathbf{\Sigma}^-_{1:T}$, $\mathbf{r}_{1:T}$)
  $\mathbf{J}_{1:T}$ = GRU^{SG} $(\mathbf{\Sigma}^-_{1:T})$
  $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$ = *Filtering Step* $(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T}, \mathbf{K}_{1:T}, \mathbf{w}_{1:T}, \mathbf{H}_{1:T})$
  $(\mathbf{x}_{1:T|T}, \mathbf{\Sigma}_{1:T|T})$ = *Smoothing Step* $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T}, \mathbf{J}_{1:T}, \mathbf{F}_{1:T}(\mathbf{Q}))$
  $\mathbf{o}^+_{1:T}$ = MLP $(\mathbf{x}_{1:T|T}+, \mathbf{\Sigma}_{1:T|T})$
  $\mathcal{L}_{1:T}$ = - Likelihood $(\mathbf{gt}_{1:T}, \mathbf{o}^+_{1:T})$
  Backward Propagation ()
else
  Q *network* = MLP $(\mathbf{x}^+_{0:T-1})$ or GRU $(\mathbf{x}^+_{0:T-1}, \mathbf{Q}_{1:T})$
  $(\mathbf{F}_{1:T}, \mathbf{H}_{1:T})$ = Dynamics
  $\mathbf{r}_{1:T}$ = *trainable layer*$(\mathbf{y}_{1:T})$
  $\mathbf{q}_{1:T}$ = Q *network*$(\mathbf{x}^+_{0:T-1})$
  $(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T})$ = *Prediction Step* $((\mathbf{x}^+_{0:T-1}, \mathbf{\Sigma}^+_{0:T-1}), \mathbf{Q}_{1:T}, \mathbf{F}_{1:T})$
  $\mathbf{K}_{1:T}$ = GRU^{KG} $(\mathbf{\Sigma}^-_{1:T}$, $\mathbf{R}_{1:T}$)
  $\mathbf{J}_{1:T}$ = GRU^{SG} $(\mathbf{\Sigma}^-_{1:T}$)
  $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$ = *Filtering Step* $(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T}, \mathbf{K}_{1:T}, \mathbf{y}_{1:T}, \mathbf{H}_{1:T})$
  $(\mathbf{x}_{1:T|T}, \mathbf{\Sigma}_{1:T|T})$ = *Smoothing Step* $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T}, \mathbf{J}_{1:T}, \mathbf{F}_{1:T}(\mathbf{Q}))$
  $\sigma_{1:T|T}$ = *Trainable Layer* $(\mathbf{\Sigma}_{1:T|T})$
  $\mathbf{o}^+_{1:T}$ = $[\mathbf{x}_{1:T|T}, \sigma_{1:T|T}]$
  $\mathcal{L}_{1:T}$ = - Likelihood $(\mathbf{gt}_{1:T}, \mathbf{o}^+_{1:T})$
  Backward Propagation ()
end if