M.S. THESIS

Learning and inferencing state-space models through GRU cells and Bayesian principles

Hamidreza Hashempoor

February 2023

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SEOUL NATIONAL UNIVERSITY

공학석사학위논문

베이지안 원리와 GRU셀을 활용한 상태 공간 모델의 러닝과 추론

하미드레자

2023년 2월

서울대학교 대학원
전기 컴퓨터 공학부
하미드레자

Learning and inferencing state-space models through GRU cells and Bayesian principles

Advisor: Wan Choi
이 논문을 공학석사 학위논문으로 제출함

Seoul National University Graduate School
서울대학교 대학원

Faculty of Electrical and Computer Engineering
전기 컴퓨터 공학부

Hamidreza Hashempoor

December 2022

위 원 장: Byonghyo Shim
부 위 원 장: Wan Choi
위 원: Kyunghan Lee

Learning and inferencing state-space models through GRU cells and Bayesian principles

Advisor: Wan Choi

Master thesis

Seoul National University Graduate School

Faculty of Electrical and Computer Engineering

Hamidreza Hashempoor

December 2022

Committee Chair: Byonghyo Shim
Advisor: Wan Choi
Committee Member: Kyunghan Lee

Abstract

State-space models (SSMs) perform predictions by learning the underlying dynamics of observed sequence. We start with a throughout literature review on Gaussian Process (GP) models and time series models based on GPs. Then, we elaborate more on the Gaussian Process State-Space Model (GP-SSM): a Bayesian nonparametric generalisation of discrete time nonlinear state-space models. We provide a formulation of the GP-SSM that offers different insight into its properties.

Then, we propose a new SSM 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 complicated dynamics of the system that enables us to perform estimation and imputation tasks in both data presence and absence. The proposed model uses the GRU cells into its structure to complete the data flow, 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.

Finally, in the numerical experiments, we show that the GIN reduces the uncertainty of estimates and outperforms its counterparts , LSTMs, GRUs and variational approaches. Several SOTA approaches are taken into account for the sake of comparison in order to show the out-performance of the proposed algorithm.

keywords: Gaussian Process, Time Seriese

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Time Series Modeling

Time series data are a number of measurements that are taken over time. For instance, a time series dataset may be constructed by measuring and recording power generated by a solar panel, or by monitoring the essential signs of a patient under control. The ubiquity of time series data and their presence in the variety of applications makes its analysis indispensable for all contexts as disparate as the social sciences, biology, engineering or econometrics, etc. Time series are prone to indicate high correlations resulted from the temporal structure in the data. Thus, it is not amazing that various methods with separated purposes have been developed over time for time series analysis. In this thesis we mainly focus on a model-based time series approach for the sake of analysis. Models can be taken into account as mathematical constructions that often provide a view about how the data is generated and measured. It is worth noting that the models are useful to make predictions about the future and they can help to better understand what happened when the data was being generated and recorded. In the field of control theory, the process of inferring and understanding models of time series using data is called system identification [1]. However, in the fields of statistics and machine learning it is often called as estimation, fitting, inference or learning of time series [2]. Creating

meaningful models given the available data is among the most important practical problems. Otherwise, in the case of meaningless models, any reasoning and prediction or design based on the data may be problematic and contains multiple drawbacks. Due to the restricted available data for our analysis both in quality and quantity, models are usually not well-defined and suitable. It is therefore necessary to ask a crucial question: are we restricted to make just one model or could we create a bunch of models that were all well-fitted given the available data? If we take the uncertainty into account, we could move from the notion of having a specific model to that of keeping various collection of models and mixing them to construct decisions of any sort. This is one of the rudimentary ideas in Bayesian inference. In this thesis, we develop methods of Bayesian inference applicable for inferring dynamical systems using models, which are based on Gaussian processes. However we work with very general models that can be applied in a variety of scenarios, our mindset is that of the system identification field. In other words, we try to learn models typically found in engineering problems where a comparatively limited amount of noisy sensors measurements give us an ambiguous vision about the system's dynamics.

1.2 Bayesian inference for time series models

1.2.1 Bayesian methods

For learning a model from a time series or a sequence of data, we do not have the chance of gathering infinite amount of noise free data and unrestricted computational power. In practice, we are facing with limited noise datasets affected by noise which causes the uncertainty about what the most suitable model is fit to the present data. In Bayesian inference, probabilities are behaved as a solution to demonstrate the subjective uncertainty of the rational agent performing inference [3]. This uncertainty is shown as a probability distribution over the model, where the data is given

$$p(\mathcal{M}|\mathcal{D}). \tag{1.1}$$

Here, the model is considered in the functional form perspective or the value of parameters that the model might have. This is different from the well known and common approaches of time series analysis, where a single model is considered then the system parameters are found, usually by optimising a cost function such as the likelihood maximisation [4]. As the optimisation is completed, the resulting model is taken into account as the best and most fitted candidate of the system and is used for any further inference and application. However, in a Bayesian method it is already known that multiple set of models can be fit with the data [5]. For the parametric model scenarios, instead of obtaining a single estimation for the "best" value of the θ parameters, Bayesian inference provides a posterior probability distribution over the mentioned parameters $p(\theta|\mathcal{D})$. This distribution enables us to realize the fact that multiple sets of values for the parameter might also be acceptable given the observed data $\mathcal{D}$. Thus, we can interpret the posterior $p(\theta|\mathcal{D})$ as our level of acceptance and degree of belief about the value of the parameter θ. By such choice, predictions or decisions, that are conducted based on the posterior, are made by computing expectations over the posterior. In other words, but not in a formal definition, one can think of predictions as being an average of different values that their weights are defined by the posterior distribution. Also, predictions can be made with uncertainty and error level that shows both the stochastic behaviour of system and our own belief about what the correct model is and what accurate predictions are. For instance, consider a parametric model of a discrete-time stochastic dynamical system with a continuous state defined by $\mathbf{x}_t$. The state transition density then is defined as

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \theta). \tag{1.2}$$

As mentioned, Bayesian learning constructs a posterior over the parameters θ given the data $p(\theta|\mathcal{D})$. For the sake of state prediction, we can integrate over the posterior in

order to average over all acceptable values of the parameter after having seen the data

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathcal{D}) = \int p(\mathbf{x}_{t+1}|\mathbf{x}_t, \theta)p(\theta|\mathcal{D})d\theta \tag{1.3}$$

In this case, the predictions are generated by considering all acceptable values of θ, not only one of them or the "best one".

Bayesian inference consider a prior $p(\theta)$ over the parameters. The prior is a probability distribution shows how we are confident about the object to be inferred before seeing the data. Although, this requirement for a subjective distribution without seeing data is often criticised, but, the prior is an opportunity to formalise many of the assumptions that in other methods may not be explicit or less obvious. [6] points out that "you cannot do inference without making assumptions". These assumptions are very clearly specified in the Bayesian approach. In the context of learning dynamical systems, [7] derive maximum likelihood and least squares methods from Bayesian inference. Although maximum likelihood and Bayesian inference are related in their use of probability, but they are fundamentally separated. Bayesian methods are also common in "big data" analysis when the complexity of the system that generated the data is large compare to the amount of available data. In this scenario, the models of large capacity are needed to captured the whole details appropriately. Moreover, there also will be uncertainty about the system that constructs the data.

1.3 Nonparametric Models

In (1.3), we use this fact

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \theta, \mathcal{D}) = p(\mathbf{x}_{t+1}|\mathbf{x}_t, \theta) \tag{1.4}$$

that works properly for the parametric models: predictions are conditionally independent of the observed data $\mathcal{D}$ given the parameters. In other words, the data effects are already considered into the parameter θ and the subsequent prediction does not use

the original dataset. This assumption is very convenient, but it is has its own drawbacks. Choosing a model from a specific parametric class limits its flexibility. Alternatively, the usage of nonparametric models alleviates this issue. In those models, the data is not distilled to a restricted and finite set of parameters. In fact, nonparametric models can be shown to have an infinite-dimensional parameter space [8]. This allows the model to identify more complex details as the size of the dataset $\mathcal{D}$ grows.

One could think a model as an information channel from past data to future predictions [9]. In this perspective, a parametric model constructs a bottleneck in the information channel and predictions are made relying only on the learnt parameters. However, nonparametric models are memory-based since they are required to "realize" the full dataset for the sake of predictions. This can be presented as nonparametric models having a number of parameters that progressively grows with the size of the dataset. Bayesian nonparametric models employ two aspects presented so far. They allow Bayesian inference to be conducted on objects of infinite dimensions.

1.4 Contributions

The main contributions in this thesis are as follow

(i) modeling high-low dimensional sequences: we show the eligibility of the GIN to infer both cases, where we conduct three experiments of high dimensional non-linear dynamics observation for single-double pendulum and visual odometry task and two experiments of low dimensional chaotic observation for Lorenz attractor and real world NCLT dataset. For each case, we compare the GIN with the SOTA approaches like variational EM methods, e.g. [10] and [11], and Gaussian Process-SSMs (GPSSMs), e.g. [12] and [13].

(ii) Ability of learning the dynamics(in the lack of them) and utilizing available dynamics(in the presence of them) by introducing GRU cells in the Kalman filtering-smoothing: to attain more accurate inference of observed dynamical

system, we apply GRU cells that increases the modeling capability of the Kalman filtering-smoothing. By conduction an ablation study of the GIN being replaced by a linearized Gaussian state transition without non-linearity, we show the GIN is able for better learning state space representation with disentangled dynamics features.

(iii) Direct optimization: We show that the posterior and smoothing inference distribution of the state-space model is tractable while dynamics and parameters are estimated by neural networks. Despite variational approaches, this allows us to use recursive Bayesian updates for direct likelihood maximization.

(iv) Noise robustness: verified by the numerical results, inferencing for highly distorted sequences is feasible with the GIN.

(v) 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. Experimental results show the out-performance of the GIN over the SOTA studies in the imputation task.

Chapter 2

Background and Literature Review

In this chapter we are providing a self-contained review of previous work on time series modelling with Gaussian processes. Afterwards, we provide a review of related works and include a comparison table, in which we make a comparison between our approach against the SOTA.

2.1 Introduction

Time series modelling or learning dynamical systems with their parameters, also known as system identification, tries to create a model based on measured signals. This model can be integrated into multiple applications, for instance to predict future behaviour of the system or explain the underlying structure in the data or to reconstruct the original time series from the distorted noisy version. Modeling dynamical systems is ubiquitous in nature, engineering and the social sciences. For an example, we may gather data about how the number of individuals of multiple species in an ecosystem changes with the passage of time; we could monitor and save data from sensors in an airplane; or we could monitor the evolution of share price in a stock trading market. In all of the abovementioned cases, learning from time series can represent both intuition and the ability to make predictions. We assume that there are potentially two kinds of gathered

signals. In the system identification context, those signals are named inputs and outputs. Inputs are the signals that have external influences to the system and outputs are signals that rely on the current features of the system. We will denote the input vector at time t by $\mathbf{u}_t$ and the output vector by $\mathbf{w}_t$. In this review we are mainly focus on two common and well-known families of dynamical system models [1]. First, auto-regressive (AR) models that directly model the next output of a system from a number of previous inputs and outputs

$$\mathbf{w}_t = f(\mathbf{w}_{t-1}, \mathbf{w}_{t-2}, \ldots, \mathbf{w}_{t-\tau}, \mathbf{u}_t, \mathbf{u}_{t-1}, \ldots, \mathbf{u}_{t-\tau}) + \delta_t \tag{2.1}$$

where δ_t represents random noise that is independent and identically distributed (i.i.d) over time.

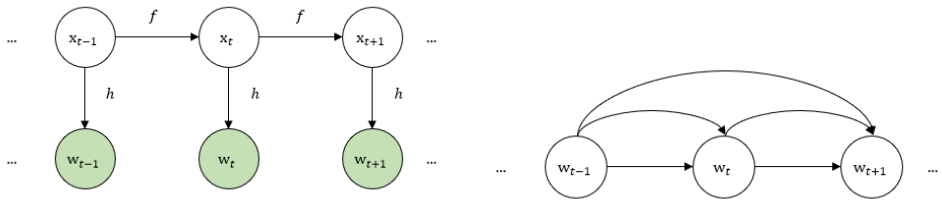
The other famous and common class of dynamical system models is named state-space models (SSM). In the SSMs latent (unobserved) variables are introduces that are called states $\mathbf{x}_t$. All the history of the system is summarised in the state at a given time and the state is enough to make predictions about its future. A SSM is usually defined by the state transition function f and the measurement function h

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{q}_t \tag{2.2}$$

$$\mathbf{w}_t = h(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{r}_t \tag{2.3}$$

where $\mathbf{q}_t$ and $\mathbf{r}_t$ are additive noises known as the process noise and measurement noise, respectively.

From this moment on, in the interest of notation simplicity, we avoid explicitly conditioning on observed inputs $\mathbf{u}_t$. When available, inputs can always be considered as arguments to the functions that are learnt. Figure 2.1b represents the graphical model of an auto-regressive model and figure 2.1a is the graphical model of a state-space model.



(a) SSM Graphical Model (b) AR Graphical Model

Figure 2.1: Graphical models of SSMs and ARs

2.2 Gaussian Process

Gaussian processes (GPs) are one of stochastic processes that have shown successful performance to conduct direct inference over the functions spaces [14]. This contradicts with models of functions, which are constructed based on the parameterised class of functions and a prior over the parameters. In dynamical systems models context, Gaussian processes can be used as priors over the functions that indicates the system dynamics or a map from latent states to measurements. In the following, we include a brief explanation of Gaussian processes and their application for the statistical regression practical problems. A Gaussian process is defined as a set of random variables, any combination of which have a joint Gaussian distribution

$$p(\mathbf{f}_i, \mathbf{f}_j, \mathbf{f}_j, \dots) = \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}_i) \\ m(\mathbf{x}_j) \\ m(\mathbf{x}_k) \\ \vdots \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}_i, \mathbf{x}_i) & k(\mathbf{x}_i, \mathbf{x}_j) & k(\mathbf{x}_i, \mathbf{x}_k) & \dots \\ k(\mathbf{x}_j, \mathbf{x}_i) & k(\mathbf{x}_j, \mathbf{x}_j) & k(\mathbf{x}_j, \mathbf{x}_k) & \dots \\ k(\mathbf{x}_k, \mathbf{x}_i) & k(\mathbf{x}_k, \mathbf{x}_j) & k(\mathbf{x}_k, \mathbf{x}_k) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \right) \tag{2.4}$$

We denote $\mathbf{f}_i$ as the value of a function at a particular input location $f(\mathbf{x}_i)$. To show that a function follows a Gaussian process, we write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.5}$$

where $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean and covariance functions respectively.

2.2.1 Gaussian Process For Regression

The regression problem is the most common and maybe the easiest promblem in machine learning in which one can utilize Gaussian processes. The task consists of learning from a dataset with the pairs of data in the form of input-output $(\mathbf{x}_i, \mathbf{w}_i)$ for N data pairs and the outputs are real-valued. After learning, the prediction of the value of the output $\mathbf{w}_j$ at any new test input $\mathbf{x}_j$ is feasible. Regression, in other words, means to learn the function mapping inputs to outputs: $\mathbf{w}_j = f(\mathbf{x}_j)$. We indicate how to perform Bayesian inference in the space of functions with the help of Gaussian processes. When we want to perform Bayesian inference on a model which is parametric, we put a prior on the parameter $p(\theta)$ and evaluate a posterior distribution over the parameter given the data, by combining the prior with the likelihood function $p(\mathbf{w}|\theta)$

$$p(\theta|\mathbf{w}) = \frac{p(\mathbf{w}|\theta)p(\theta)}{p(\mathbf{w})} \tag{2.6}$$

where $\mathbf{w}$ is called the evidence or the marginal likelihood and depends on the prior and the likelihood

$$p(\mathbf{w}) = \int p(\mathbf{w}, \theta)d\theta = \int p(\mathbf{w}|\theta)p(\theta)d\theta \tag{2.7}$$

In regression, we are aiming at finding the function, which maps inputs to outputs. A parametric approach to Bayesian regression includes inferring a family of functions that are parameterised by a set of parameters. Then, considering a prior on the mentioned parameters and performing inference. Although, we could find a less limited and very powerful method for inference on functions by directly specifying a prior over an infinite-dimensional space of functions. This contradicts with considering a prior over a set of parameters that implicitly define a distribution over functions.

Gaussian process is a very useful prior over functions. In a Gaussian process, once we gather a collection of points $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ at which we want to evaluate a function, the prior distribution over the values of the function at those locations,

$\mathbf{f} = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)\}$, has a Gaussian distribution

$$p(\mathbf{f}|\mathbf{x}) = \mathcal{N}(\mathbf{m}(\mathbf{x}), \mathbf{K}(\mathbf{x})) \tag{2.8}$$

where $\mathbf{m}(\mathbf{x})$ and $\mathbf{K}(\mathbf{x})$ are the mean vector and covariance matrix. This has resulted from the fact that the marginal of a Gaussian process has Gaussian distribution. Thus, when we only deal with the Gaussian process computations including the prior are based on Gaussian distributions. Bayes' theorem can be employed in the conventional way to obtain a posterior over the latent function in the location of interest points $\mathbf{x}$, where observations $\mathbf{w} = \{\mathbf{w}_1, \ldots, \mathbf{w}_N\}$ are available

$$p(\mathbf{f}|\mathbf{w}, \mathbf{x}) = \frac{p(\mathbf{w}|\mathbf{f})p(\mathbf{f}|\mathbf{x})}{p(\mathbf{w}|\mathbf{x})} = \frac{p(\mathbf{w}|\mathbf{f})\mathcal{N}(\mathbf{f}|\mathbf{m}(\mathbf{x}), \mathbf{K}(\mathbf{x}))}{p(\mathbf{w}|\mathbf{x})} \tag{2.9}$$

Because the denominator is a constant for specific given dataset, we note the proportionality notation here as

$$p(\mathbf{f}|\mathbf{w}, \mathbf{x}) \propto p(\mathbf{w}|\mathbf{f})\mathcal{N}(\mathbf{f}|\mathbf{m}(\mathbf{x}), \mathbf{K}(\mathbf{x})) \tag{2.10}$$

In the cases where the likelihood has the form $p(\mathbf{w}|\mathbf{f}) = \mathcal{N}(\mathbf{w}|\mathbf{f}, \mathbf{R})$, the posterior has an analytical solution as

$$p(\mathbf{f}|\mathbf{w}, \mathbf{x}) = \mathcal{N}\Big(\mathbf{f}|\mathbf{K}(\mathbf{x})\big(\mathbf{K}(\mathbf{x})+\mathbf{R}\big)^{-1}\big(\mathbf{w}-\mathbf{m}(\mathbf{x})\big), \mathbf{K}(\mathbf{x})-\mathbf{K}(\mathbf{x})\big(\mathbf{K}(\mathbf{x})+\mathbf{R}\big)^{-1}\mathbf{K}(\mathbf{x})\Big) \tag{2.11}$$

In the case of Gaussian process regression, by considering additive Gaussian noise, the posterior has the mentioned form. However, for random distribution selection of the likelihood functions, the posterior will not necessarily be Gaussian. The distribution $p(\mathbf{f}|\mathbf{w}, \mathbf{x})$ corresponds to the posterior over the latent function $\mathbf{f}(\mathbf{x})$ in the location of the interest points, $\mathbf{x}$. This may be insightful in itself, but we are also willing to find the value of $\mathbf{f}(\mathbf{x})$ at different location than the input space. In other words, we are willing to predict the distribution of $\mathbf{f}_* = \mathbf{f}(\mathbf{x}_*)$ at a new location $\mathbf{x}_*$

$$p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{w}, \mathbf{x}). \tag{2.12}$$

By marginalising out $\mathbf{f}$, we have

$$p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{w}, \mathbf{x}) = \int p(\mathbf{f}_*, \mathbf{f}|\mathbf{x}_*, \mathbf{w}, \mathbf{x})d\mathbf{f} = \int p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{f}, \mathbf{x})p(\mathbf{f}|\mathbf{w}, \mathbf{x})d\mathbf{f}. \quad (2.13)$$

The first term in the second integral is always a Gaussian because of the Gaussian process properties that consider prior for linking all possible values of $\mathbf{f}$ and $\mathbf{f}_*$ with a joint normal distribution [14]. The second term, $p(\mathbf{f}|\mathbf{w}, \mathbf{x})$, is the posterior of $\mathbf{f}$. When the likelihood is $p(\mathbf{w}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \mathbf{R})$, both the posterior and predictive distributions are Gaussian. The marginal likelihood maximisation with respect to the introduced mean and covariance functions, prepare a practical way to conduct Bayesian model selection [14].

A potential representation of Gaussian processes in a graphical model includes plotting a solid bar between the jointly Gaussian distributed variables [14]. The random variables that are in touch with the solid bar are included to the same Gaussian process and have dependency on each other where this dependency is defined with the covariance matrix. There are unlimited amount of variables in the Gaussian process but we only plot a limited set of variables. Figure 2.2 illustrates Gaussian process model for the regression with a dataset including three inputs and three outputs and where predictions are to be made at a given test point $\mathbf{x}_*$.

Another interpretation of this notation includes considering a function $f(.)$ that has $\mathcal{GP}$ distribution. All variables $\mathbf{f}$ are conditionally independent of each other given that function. This modeling is illustrated in figure 2.3.

It is worth noting that models involving the solid black bar should not be considered as a conventional undirected graphical model in probabilistic graphical models. The thick bar notation demonstrates in a simple way that variables share the same $\mathcal{GP}$ distribution. For general covariance functions, drawing figure 2.2 with the conventional directed graphical models would result in a large number of edges, which is depicted in figure 2.4.

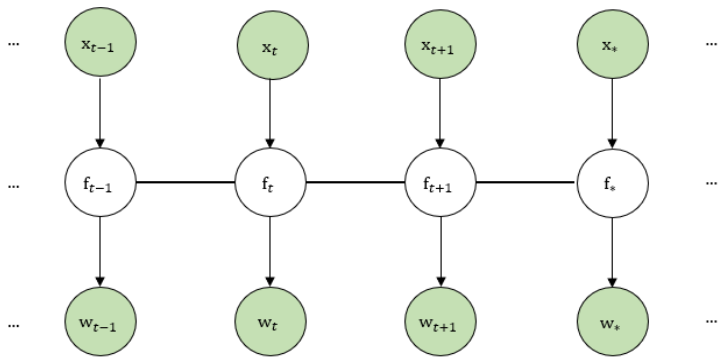


Figure 2.2: Graphical model of Gaussian process regression.

2.2.2 Linear 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

$$p_{\gamma_t}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}), \quad p_{\gamma_t}(\mathbf{w}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{w}_t; \mathbf{H}_t\mathbf{z}_t, \mathbf{R}). \tag{2.14}$$

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

$$p_\gamma(\mathbf{w}, \mathbf{x}|\mathbf{u}) = p_\gamma(\mathbf{w}|\mathbf{x})p_\gamma(\mathbf{x}|\mathbf{u}) = \prod_{t=1}^{T} p_{\gamma_t}(\mathbf{w}_t|\mathbf{x}_t).p(\mathbf{x}_1) \prod_{t=2}^{T} p_{\gamma_t}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \tag{2.15}$$

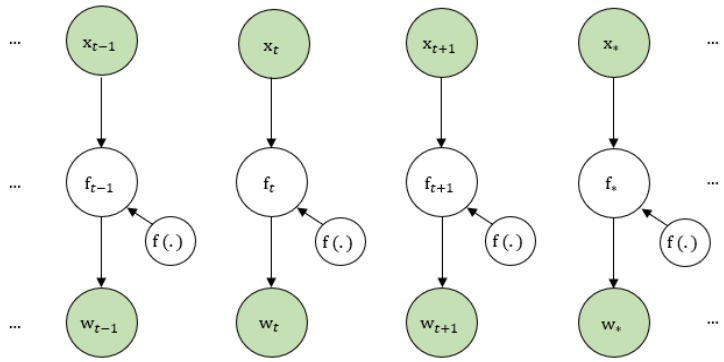


Figure 2.3: Graphical model of Gaussian process regression explicitly representing the latent function.

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.

2.2.3 Filtering and Smoothing Parameterization for LGSSMs

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.

$$\mathbf{x}_{t+1}^{-} = \mathbf{F}\mathbf{x}_{t}^{+}, \text{and} \quad \mathbf{\Sigma}_{t+1}^{-} = \mathbf{F}\mathbf{\Sigma}_{t}^{+}\mathbf{F}^{T} + \mathbf{Q}, \text{and} \quad \mathbf{Q} = \sigma_{\text{trans}}^{2}\mathbf{I} \tag{2.16}$$

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 $(\mathbf{x}_{t+1}^{+}, \mathbf{\Sigma}_{t+1}^{+})$.

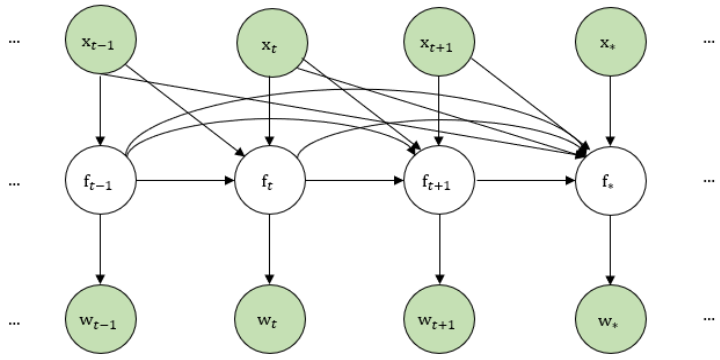


Figure 2.4: Graphical model of Gaussian process regression using only directed edges.

$$\mathbf{K}_{t+1} = \mathbf{\Sigma}^-_{t+1}\mathbf{H}^T\left(\mathbf{H}\mathbf{\Sigma}^-_{t+1}\mathbf{H}^T + \mathbf{R}\right)^{-1}, \text{and} \tag{2.17}$$

$$\mathbf{x}^+_{t+1} = \mathbf{x}^-_{t+1} + \mathbf{\Sigma}^-_{t+1}\mathbf{H}^T\left(\mathbf{H}\mathbf{\Sigma}^-_{t+1}\mathbf{H}^T + \mathbf{R}\right)^{-1}(\mathbf{w}_t - \mathbf{H}\mathbf{x}^-_{t+1}) = \mathbf{x}^-_{t+1} + \mathbf{K}_{t+1}(\mathbf{w}_t - \mathbf{H}\mathbf{x}^-_{t+1}), \tag{2.18}$$

$$\mathbf{\Sigma}^+_{t+1} = \mathbf{\Sigma}^-_{t+1} - \mathbf{\Sigma}^-_{t+1}\mathbf{H}^T\left(\mathbf{H}\mathbf{\Sigma}^-_{t+1}\mathbf{H}^T + \mathbf{R}\right)^{-1}\mathbf{H}\mathbf{\Sigma}^-_{t+1}. \tag{2.19}$$

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.

We derive smoothing parameterization, where the key idea is to use Markov property, which states that $\mathbf{x}_t$ is independent of future observations $\mathbf{w}_{t+1:T}$ as long as $\mathbf{x}_{t+1}$ is known. However, we are not aware of $\mathbf{x}_{t+1}$, but there is a distribution over it. So by conditioning on $\mathbf{x}_{t+1}$ and then marginalizing out it is possible to obtain $\mathbf{x}_t$ conditioned on $\mathbf{w}_{1:T}$.

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{w}_{1:T}) &= \int p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:T}) p(\mathbf{x}_{t+1}|\mathbf{w}_{1:T}) d\mathbf{x}_{t+1} \\ &= \int p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:t}, \cancel{\mathbf{w}_{t+1:T}}) p(\mathbf{x}_{t+1}|\mathbf{w}_{1:T}) d\mathbf{x}_{t+1} \end{aligned} \tag{2.20}$$

By using induction and and smoothed distribution for $t+1$:

$$p(\mathbf{x}_{t+1}|\mathbf{w}_{1:T}) = \mathcal{N}(\mathbf{x}_{t+1|T}, \boldsymbol{\Sigma}_{t+1|T}) \tag{2.21}$$

we calculate the filtered two-slice distribution as follows:

$$.p(\mathbf{x}_t, \mathbf{x}_{t+1}|\mathbf{w}_{1:t}) = \mathcal{N}\Bigg(\begin{pmatrix} \mathbf{x}_t^+ \\ \mathbf{x}_{t+1}^- \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_t^+ & \boldsymbol{\Sigma}_t^+ \mathbf{F}_{t+1}^T \\ \mathbf{F}_{t+1}\boldsymbol{\Sigma}_t^+ & \boldsymbol{\Sigma}_{t+1}^- \end{pmatrix} \Bigg) \tag{2.22}$$

by using Gaussian conditioning we have:

$$p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:t}) = \mathcal{N}(\mathbf{x}_t^+ + \mathbf{J}_t\big(\mathbf{x}_{t+1} - \mathbf{F}_{t+1}\mathbf{x}_t^+\big), \boldsymbol{\Sigma}_t^+ - \mathbf{J}_t\boldsymbol{\Sigma}_{t+1}^-\mathbf{J}_t^T) \tag{2.23}$$

where $\mathbf{J}_t = \boldsymbol{\Sigma}_t^+\mathbf{F}_{t+1}[\boldsymbol{\Sigma}_{t+1}^-]^{-1}$. We calculate the smoothed distribution for t using the rules of iterated expectation and covariance:

$$\begin{aligned} \mathbf{x}_{t|T} &= \mathbb{E}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:T}] \ |\mathbf{w}_{1:T}\big] \\ &= \mathbb{E}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:t}] \ |\mathbf{w}_{1:T}\big] \\ &= \mathbb{E}\big[\mathbf{x}_t^+ + \mathbf{J}_t(\mathbf{x}_{t+1} - \mathbf{F}_{t+1}\mathbf{x}_t^+) \ |\mathbf{w}_{1:T}\big] \\ &= \mathbf{x}_t^+ + \mathbf{J}_t(\mathbf{x}_{t+1|T} - \mathbf{F}_{t+1}\mathbf{x}_t^+) \end{aligned} \tag{2.24}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{t|T} &= \mathrm{cov}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:T}] \ |\mathbf{w}_{1:T}\big] + \mathbb{E}\big[\mathrm{cov}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:T}] \ |\mathbf{w}_{1:T}\big] \\ &= \mathrm{cov}\big[\mathbb{E}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:t}] \ |\mathbf{w}_{1:T}\big] + \mathbb{E}\big[\mathrm{cov}[\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{w}_{1:t}] \ |\mathbf{w}_{1:T}\big] \\ &= \mathrm{cov}\big[\mathbf{x}_t^+ + \mathbf{J}_t(\mathbf{x}_{t+1} - \mathbf{F}_{t+1}\mathbf{x}_t^+) \ |\mathbf{w}_{1:T}\big] + \mathbb{E}\big[\boldsymbol{\Sigma}_t^+ - \mathbf{J}_t\boldsymbol{\Sigma}_{t+1}^-\mathbf{J}_t^T \ |\mathbf{w}_{1:T}\big] \\ &= \mathbf{J}_t\mathrm{cov}\big[\mathbf{x}_{t+1} - \mathbf{F}_{t+1}\mathbf{x}_t^+ \ |\mathbf{w}_{1:T}\big]\mathbf{J}_t^T + \boldsymbol{\Sigma}_t^+ - \mathbf{J}_t\boldsymbol{\Sigma}_{t+1}^-\mathbf{J}_t^T \\ &= \mathbf{J}_t\boldsymbol{\Sigma}_{t+1|T}\mathbf{J}_t^T + \boldsymbol{\Sigma}_t^+ - \mathbf{J}_t\boldsymbol{\Sigma}_{t+1}^-\mathbf{J}_t^T \\ &= \boldsymbol{\Sigma}_t^+ + \mathbf{J}_t\big(\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{t+1}^-\big)\mathbf{J}_t^T. \end{aligned} \tag{2.25}$$

2.3 Related Works

LSTMs and GRUs are RNN structures with the ability of dealing with high-low dimensional data, however, in these models they ignore uncertainty estimation. To deal with complex sensory inputs, some approaches integrate a deep auto encoders into their architecture. Among these works, Embed to Control (E2C) [15] 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 [16] and RKN [17] apply CNNs for dimension-reduction and output both the uncertainty vector and latent observation, where they move away from variational inference and borrow Bayesian properties for the inference. However, these methods cannot handle the cases with the available knowledge of the dynamics and impose restrictive assumptions over covariance matrices, while the GIN provides a principled way for using the available partial dynamics information and release any assumption over covariance. Toward learning state space (system identification) a bunch of works like [18], [19] and [20] propose algorithms to learn GPSSMs based on maximum likelihood estimation with the iterative EM algorithm, where 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). [20] 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 like Structured Inference Networks (SIN) [21], where it utilizes a RNN to update the state. Kalman Variational Autoencoder (KVAE) [11] and Extended KVAE (EKVAE) [10] use the original KF equations and apply both filtering and smoothing. However, these EM-based variational inference methods are not able to estimate the states directly because of optimizing the lower bound of likelihood. Extra complexity is another issue with these approaches, while they are addressed by the proposed structure and direct end-to-end optimization in the GIN. We compare the GIN

with these approaches in the experiment section and provide a detailed discussion of recent related work in subsection 2.3.1.

2.3.1 Qualitative comparison of the GIN to recent related work

In table 2.1, we make a comparison to show whether algorithms are able to handle high and low dimensional observations, learn dynamics, use available-partial dynamics, estimate state appropriately, provide model's uncertainty estimates handling noisy data, handle missing data and perform direct optimization. Classic LGSSMs, e.g. EKF and UKF, work based on the linearization of the transition and emission equations and apply classic Bayesian updates over the linearized system with respect to the states. In other words, transition and emission matrices in the classic LGSSMs are not data-deriven nor trainable. Despite classic LGSSMs, in the GIN we use a data-driven based network to learn dynamics.

Table 2.1: Learning the dynamics in LGSSM is shown with $\times/\checkmark$ because general LGSSMs, e.g. UKF and EKF, are not able to learn the dynamics. However, in our setting and parameterization we use a data driven-based network for obtaining transition and emission matrices to make LGSSMs comparable with the GIN for high dimensional observation experiments.

Model	high-d	low-d	learn dynamics	use dynamics	state est	uncertainty	missing-noise	dir opt
LSTM [22]	✓	✓	✓	✓	✓	×	✓	✓
GRU [23]	✓	✓	✓	✓	✓	×	✓	✓
P2T [24]	✓	✓	✓	×	✓	×	×	✓
E2C [15]	✓	✓	✓	×	×	✓	×	×
BB-VI [25]	×	✓	✓	×	×	✓	✓	×
SIN [21]	✓	✓	✓	×	×	✓	✓	×
DVBF [26]	✓	✓	✓	×	×	✓	✓	×
VSMC [27]	✓	✓	✓	×	×	✓	✓	×
DSA [28]	✓	✓	✓	×	×	✓	×	×
KVAE [11]	×	✓	✓	×	×	✓	✓	×
EKVAE [10]	×	✓	✓	×	×	✓	✓	×
DSSM [13]	×	✓	✓	×	✓	✓	×	✓
HybridGNN [12]	×	✓	×	✓	✓	×	✓	✓
KalmanNet [29]	×	✓	×	✓	✓	×	✓	✓
SSI [30]	×	✓	×	✓	✓	✓	✓	✓
LGSSM	×	✓	$\times/\checkmark$	✓	✓	✓	✓	✓
GIN	✓	✓	✓	✓	✓	✓	✓	✓

Chapter 3

Gated Inference Network

3.1 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 [31] [32], in the sense that it estimates the system parameters directly from the observations, which is in the figure 3.1. $e(.)$ and $d(.)$ are implemented with non-linear functions, e.g. auto encoders-MLPs. Transition block in figure 3.1 represents the dynamics of the system that allows for the 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 computational complexity, i.e matrix inversion issues. GRU cells empower the whole system by applying non-linearity to the linearized Gaussian state space models (LGSSMs). Numerical results indicate that 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 the state fits into Gaussian state space models (GSSMs), which are commonly used to model sequences of vectors.

In most cases, the dynamics of the system might not be available or hard to obtain; while the process noise and observation noise are unknown (our first three experiments). Accordingly, we construct GIN to learn unknown variables from data in an end to end fashion, then we utilize the constructed KG and SG during inference time to obtain the filtered-smoothed states. The proposed architecture is depicted in figure 3.2. In the presence of dynamics (our last two experiments), auto-encoder and *Dynamics Network* in figure 3.2 are removed. The architecture for operating high and low dimensional observations are depicted separately with further details in figures 3.3 and 3.4.

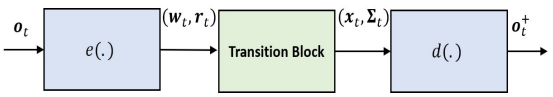


Figure 3.1: The GIN as a HW model for system identification. In the case of high-dimensional observation $\mathbf{o}_t$, the nonlinear functions $e(.)$ and $d(.)$ can be implemented by an encoder and decoder, respectively. While for the low-dimensional case, MLPs are utilized instead. The relation between the internal variables, $\mathbf{w}_t$ and $\mathbf{x}_t$, is simulated by the transition block.

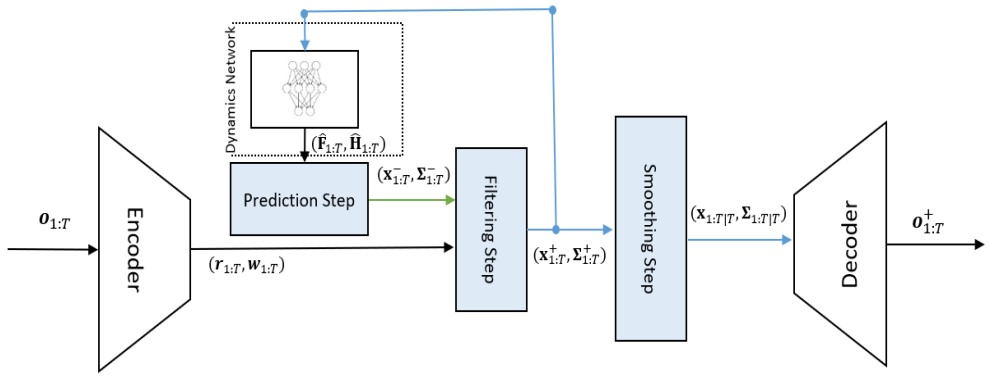


Figure 3.2: The high level structure of the GIN for high dimensional observation in the lack of dynamics, while for low dimensional cases auto-encoder is replaced by MLPs and dynamics are directly used. The latent observation $\mathbf{w}_t$ and its uncertainty $\mathbf{r}_t$, i.e., observation noise, 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 $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{H}}_t$. In the *Prediction Step* the next priors $(\mathbf{x}^-_t, \mathbf{\Sigma}^-_t)$ are obtained by using new dynamics and the last posteriors. In the filtering step, by using the priors $(\mathbf{x}^-_t, \mathbf{\Sigma}^-_t)$ and the observation $(\mathbf{w}_t, \mathbf{r}_t)$, the next posteriors $(\mathbf{x}^+_t, \mathbf{\Sigma}^+_t)$ are obtained. Applying smoothing operation over the obtained posteriors $(\mathbf{x}^+_t, \mathbf{\Sigma}^+_t)$ 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.

3.2 Parameterization

In this section we show the parameterization of the inference model. Given original noisy measurements $\mathbf{o}_{1:T}$ and latent noisy observations $\mathbf{w}_{1:T}$, we want to find good estimate of the latent states $\mathbf{x}_{1:T}$. To achieve this, we want to infer the marginal distributions $p(\mathbf{x}_t|\mathbf{w}_{1:t})$ for the online inference approach or filtering; and $p(\mathbf{x}_t|\mathbf{w}_{1:T})$ for the full inference approach or smoothing.

By defining $\gamma_t = (\mathbf{F}_t, \mathbf{H}_t, \mathbf{Q}_t, \mathbf{R}_t)$ as the model parameters at time t, where $(\mathbf{Q}_t, \mathbf{R}_t)$ are process and observation noise and $(\mathbf{F}_t, \mathbf{H}_t)$ explain the state transition and observation emission, we introduce an advantageous prediction parameterization as $p_{\gamma_t}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{w}_{1:t-1}) = \mathcal{N}(\mathbf{F}_t\mathbf{x}_{t-1}, \mathbf{Q}_t)$, where $\mathbf{x}_{t-1} \sim \mathcal{N}(\mathbf{x}^+_{t-1}, \mathbf{\Sigma}^+_{t-1})$.

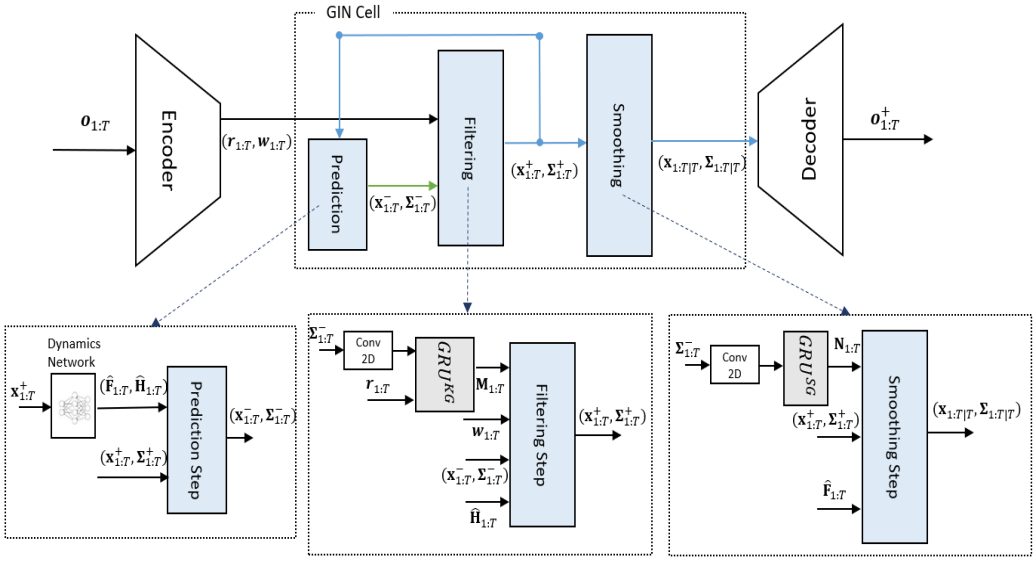


Figure 3.3: Proposed architecture for operating high dimensional observations in the lack of dynamics.

Then, $p_{\gamma_t}(\mathbf{x}_t|\mathbf{w}_{1:t-1}) = \mathcal{N}(\mathbf{F}_t\mathbf{x}_{t-1}^+, \mathbf{F}_t\boldsymbol{\Sigma}_{t-1}^+\mathbf{F}_t^T + \mathbf{Q}_t) = \mathcal{N}(\mathbf{x}_t^-, \boldsymbol{\Sigma}_t^-)$ is obtained by marginalizing out $\mathbf{x}_{t-1}$ and the Gaussianity of $p_{\gamma_t}(\mathbf{x}_t|\mathbf{w}_{1:t-1})$ results from the Gaussianity of prediction parameterization. By having $p_{\gamma_t}(\mathbf{x}_t|\mathbf{w}_{1:t-1})$ and observing $\mathbf{w}_t$, filtering parameterization is introduced as:

$$p_{\gamma_t}(\mathbf{x}_t|\mathbf{w}_{1:t}) = \mathcal{N}\Big(\mathbf{x}_t^- + \mathbf{K}_t[\mathbf{w}_t - \mathbf{H}_t\mathbf{x}_t^-],\ \ \boldsymbol{\Sigma}_t^- - \mathbf{K}_t[\mathbf{H}_t\boldsymbol{\Sigma}_t^-\mathbf{H}_t^T + \mathbf{R}_t]\mathbf{K}_t^T\Big) = \mathcal{N}\big(\mathbf{x}_t^+, \boldsymbol{\Sigma}_t^+\big) \tag{3.1}$$

where $\mathbf{K}_t$ is KG. After observing all latent observations $\mathbf{w}_{1:T}$, one can do backward induction and propagate to the previous states using the chain rule. This procedure, known as smoothing, can be parameterized as:

$$p_{\gamma_t}(\mathbf{x}_t|\mathbf{w}_{1:T}) = \mathcal{N}\Big(\mathbf{x}_t^+ + \mathbf{J}_t[\mathbf{x}_{t+1|T} - \mathbf{F}_{t+1}\mathbf{x}_t^+],\ \ \boldsymbol{\Sigma}_t^+ + \mathbf{J}_t\big[\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{t+1}^-\big]\mathbf{J}_t^T\Big) \tag{3.2}$$

where $\mathbf{J}_t$ is SG and we use short handed notation $\mathcal{N}(\mathbf{x}_{t|T}, \boldsymbol{\Sigma}_{t|T})$ instead of (3.2). Full derivation of filtering-smoothing parameterization is in appendex 2.2.3. These

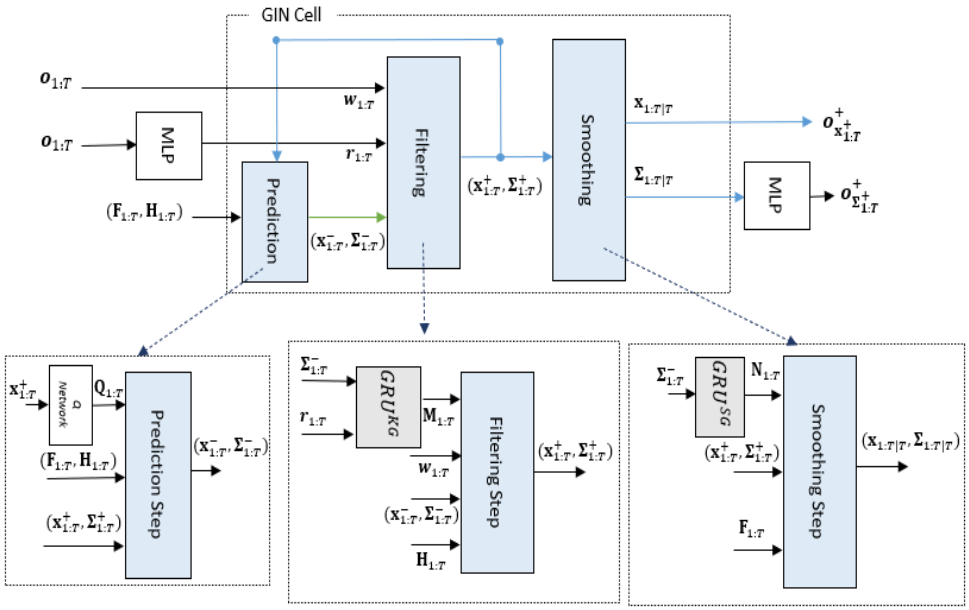


Figure 3.4: Proposed architecture for operating low dimensional observations in the presence of dynamics.

parameterizations give us some insight to 1-illustrate a tractable way to construct $p_\gamma(\mathbf{x}|\mathbf{w})$ and accordingly obtain the posterior and smoothened states, based on which $\mathbf{o}^+$ is constructed and 2- appropriately modeling γ and KG(SG) with neural networks.

To construct the KG and SG networks, we have to find appropriate inputs containing related information to attain the KG and SG. In (3.1) and (3.2), KG and SG are given by (3.3) and (3.4), respectively.

$$\mathbf{K}_t = \mathbf{\Sigma}_t^- \mathbf{H}_t^T . \left[\mathbf{H}_t \mathbf{\Sigma}_t^- \mathbf{H}_t^T + \mathbf{R}_t\right]^{-1} \propto (\mathbf{\Sigma}_t^-, \mathbf{R}_t) \tag{3.3}$$

$$\mathbf{J}_t = \mathbf{\Sigma}_t^+ \mathbf{F}_{t+1}^T . \left[\mathbf{F}_{t+1} \mathbf{\Sigma}_t^+ \mathbf{F}_{t+1}^T + \mathbf{Q}_{t+1}\right]^{-1} = \mathbf{\Sigma}_t^+ \mathbf{F}_{t+1}^T \mathbf{\Sigma}_{t+1}^- \propto \mathbf{\Sigma}_{t+1}^- \tag{3.4}$$

(3.3) is proportional to the prior covariance at time t, $\mathbf{\Sigma_t^-}$, and the observation noise matrix, $\mathbf{R_t}$, while (3.4) 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 networks including GRU that map $[\mathbf{f}(\mathbf{\Sigma_t^-}), \mathbf{R_t}]$ 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 3.2, 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. Inference in the latent state space can now be implemented by learning the process noise, prediction step, filtering step, smoothing step and learning dynamics.

3.3 Learning The Process Noise.

In the filtering procedure, the process noise in time t is obtained as

$$\mathbf{Q}_t = \mathbf{\Sigma}_t^- - \mathbf{F}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{F}_t^T. \tag{3.5}$$

where $\mathbf{\Sigma}_t^-$, $\mathbf{F}_t$ and $\mathbf{\Sigma}_{t-1}^+$ are prior state covariance, transition matrix and posterior state covariance at time t. We can elaborate the process noise matrix at time t in more details

$$\mathbf{Q}_t = \mathbf{\Sigma}_t^- - \mathbf{F}_t \mathbf{\Sigma}_{t-1}^+ \mathbf{F}_t^T = \mathbf{\Sigma}_t^- - \mathbf{F}_t \big[\mathbf{\Sigma}_{t-1}^- - \mathbf{K}_{t-1}[\mathbf{H}_{t-1}\mathbf{\Sigma}_{t-1}^- \mathbf{H}_{t-1}^T + \mathbf{R}_{t-1}]^{-1}\mathbf{K}_{t-1}^T\big]\mathbf{F}_t^T \tag{3.6}$$

combining (2.16) into (3.6) results in

$$\begin{aligned}\mathbf{Q}_t = \mathbf{\Sigma}_t^- &- \mathbf{F}_t\big[[\mathbf{F}_{t-1}\mathbf{\Sigma}_{t-2}^+\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}] \\ &-\mathbf{K}_{t-1}[\mathbf{H}_{t-1}[\mathbf{F}_{t-1}\mathbf{\Sigma}_{t-2}^+\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}]\mathbf{H}_{t-1}^T + \mathbf{R}_{t-1}]^{-1}\mathbf{K}_{t-1}^T\big]\mathbf{F}_t^T\end{aligned} \tag{3.7}$$

which is a function of $\mathbf{F}_t$, $\mathbf{Q}_{t-1}$, $\mathbf{F}_{t-1}$ and $\mathbf{H}_{t-1}$. In the GIN, $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{H}}_t$ are learned by the *Dynamics Network* with the input of $\mathbf{x}_{t-1}^{+}$. From (2.18), $\mathbf{x}_{t-1}^{+}$ is derived as a function of both $\mathbf{F}_{t-1}$ and $\mathbf{H}_{t-1}$, meaning the learned $\hat{\mathbf{F}}_t$ carries the information of both $\mathbf{H}_{t-1}$ and $\mathbf{F}_{t-1}$. Therefore, one can rewrite the equation (3.7) as

$$\mathbf{Q}_t = \mathbf{g}\Big(\hat{\mathbf{F}}_t\big(\mathbf{x}_{t-1}^{+}\big), \mathbf{Q}_{t-1}\Big), \text{where} \quad \hat{\mathbf{F}}_t = \textit{Dynamics Network}\Big(\mathbf{x}_{t-1}^{+}\big(\mathbf{H}_{t-1}, \mathbf{F}_{t-1}\big)\Big). \tag{3.8}$$

where $\mathbf{g}$ is a nonlinear function mapping $\mathbf{x}_{t-1}^{+}$ 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 $\boldsymbol{\Sigma}_{\mathbf{t-1}}^{-}$ term in (2.18), combining it with (2.16) results in

$$\begin{aligned}\mathbf{x}_{t-1}^{+} = \mathbf{x}_{t-1}^{-} + [\mathbf{F}_{t-1}\boldsymbol{\Sigma}_{t-2}^{+}\mathbf{F}_{t-1}^{T} + \mathbf{Q}_{t-1}] \\ \mathbf{H}_{t-1}^{T}\big(\mathbf{H}_{t-1}[\mathbf{F}_{t-1}\boldsymbol{\Sigma}_{t-2}^{+}\mathbf{F}_{t-1}^{T} + \mathbf{Q}_{t-1}]\mathbf{H}_{t-1}^{T} + \mathbf{R}_{t-1}\big)^{-1}(\mathbf{w}_t - \mathbf{H}_{t-1}\mathbf{x}_{t-1}^{-})\end{aligned} \tag{3.9}$$

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 $\mathbf{x}_{t-1}^{+}$.

$$\mathbf{Q}_t = \mathbf{g}\Big(\hat{\mathbf{F}}_t\big(\mathbf{x}_{t-1}^{+}\big)\Big), \text{where} \quad \hat{\mathbf{F}}_t = \textit{Dynamics Network}\Big(\mathbf{x}_{t-1}^{+}\big(\mathbf{H}_{t-1}, \mathbf{F}_{t-1}, \mathbf{Q}_{t-1}\big)\Big). \tag{3.10}$$

We call $\mathbf{g}$ as *Q Network*, where $\mathbf{g}$ can be modeled by a MLP (3.10) or a recurrent network (3.8), based on the mentioned explanations. In figure 3.4, it is shown how the *Q Network* is integrated into the whole model structure.

Thus, from (3.10), 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. $\hat{\mathbf{F}}_t(\mathbf{Q}_t)$ notation means that the learned transition matrix $\hat{\mathbf{F}}_t$ comprises the effects of $\mathbf{Q}_t$, while for the simplicity we use $\hat{\mathbf{F}}_t$ abbreviation. Therefore, it is possible to rewrite (3.5) as

$$\boldsymbol{\Sigma}_t^{-} = \hat{\mathbf{F}}_t\boldsymbol{\Sigma}_{t-1}^{+}\hat{\mathbf{F}}_t^{T}. \tag{3.11}$$

Another way to have a a meaningful inference about the process noise matrix is to obtain it from (3.8) as a recursive function of $\mathbf{x}_{t-1}^{+}$ and $\mathbf{Q}_{t-1}$. Intuitively, $\mathbf{g}$ function in (3.8) that we call it *Q Network*, 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 (3.10), 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 experiment section. 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.

3.4 Prediction Step

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

$$\mathbf{x}_t^{-} = \hat{\mathbf{F}}_t \mathbf{x}_{t-1}^{+}\,, \qquad \mathbf{\Sigma}_t^{-} = \hat{\mathbf{F}}_t \mathbf{\Sigma}_{t-1}^{+} \hat{\mathbf{F}}_t^{T} \tag{3.12}$$

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

3.5 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 3.1, we have to use the obtained KG matrix from GRU^{KG} network and learned emission matrix $\hat{\mathbf{H}}_t$ to complete updating the state mean vector and state covariance matrix. This procedure is given by

$$\mathbf{S}_t^- = \hat{\mathbf{H}}_t.\boldsymbol{\Sigma}_t^-.\hat{\mathbf{H}}_T^T + \mathbf{R}_t, \quad \mathbf{K}_t = \boldsymbol{\Sigma}_t^-\hat{\mathbf{H}}_t^T\mathbf{M}_t\mathbf{M}_t^T, \quad \mathbf{M}_t = GRU^{KG}(\mathbf{f}(\boldsymbol{\Sigma}_t^-), \mathbf{R}_t) \tag{3.13}$$

$$\mathbf{x}_t^+ = \mathbf{x}_t^- + \mathbf{K}_t.[\mathbf{w}_t - \hat{\mathbf{H}}_t\mathbf{x}_t^-], \quad \boldsymbol{\Sigma}_t^+ = \boldsymbol{\Sigma}_t^- + \mathbf{K}_t.\mathbf{S}_t^-.\mathbf{K}_t^T. \tag{3.14}$$

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, positive $\mathbf{r_t}$ vector and Cholesky factor consideration, $\mathbf{M}_t\mathbf{M}_t^T$ in (3.13), makes this procedure trivial to guarantee the positive definiteness of the resulted covariance matrices.

3.6 Smoothing Step

After obtaining filtered states $(\mathbf{x}_{1:T}^+, \boldsymbol{\Sigma}_{1:T}^+)$ 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, learned transition matrices $\hat{\mathbf{F}}_{1:T}$ and filtered states $(\mathbf{x}_{1:T}^+, \boldsymbol{\Sigma}_{1:T}^+)$. The procedure in each smoothing step is given by:

$$\mathbf{J}_t = \boldsymbol{\Sigma}_t^+\hat{\mathbf{F}}_{t+1}^T\mathbf{N}_t\mathbf{N}_t^T, \quad \mathbf{N}_t = GRU^{SG}\big(\mathbf{f}(\boldsymbol{\Sigma}_{t+1}^-)\big) \tag{3.15}$$

$$\mathbf{x}_{t|T} = \mathbf{x}_t^+ + \mathbf{J}_t\big[\mathbf{x}_{t+1|T} - \hat{\mathbf{F}}_{t+1}\mathbf{x}_t^+\big], \quad \boldsymbol{\Sigma}_{t|T} = \boldsymbol{\Sigma}_t^+ + \mathbf{J}_t\big(\boldsymbol{\Sigma}_{t+1|T} - \hat{\mathbf{F}}_{t+1}\boldsymbol{\Sigma}_t^+\hat{\mathbf{F}}_{t+1}^T\big)\mathbf{J}_t^T \tag{3.16}$$

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

3.7 Learning Dynamics

We can model the dynamics in each time step t as a function of the latent observations $\mathbf{w}_{1:t-1}$. In more details, the updates of dynamics depends on the history of the system,

which are latent observations up to current time. However, conditioning on the noisy observations can distort the procedure of learning the dynamics. Instead, we use the state $\mathbf{x}_{t-1}^{+}$ in GSSM that includes the history of the system with considerable lower noise distortion to increase system's noise robustness, where we generate time correlated noise in our experiments to show this robustness. In other words, original transition and emission equations, $\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \mathbf{q}_t$ and $\mathbf{w}_t = h(\mathbf{x}_t) + \mathbf{r}_t$, are modeled as $\mathbf{x}_t = \hat{\mathbf{F}}_t(\mathbf{x}_{t-1}^{+})\mathbf{x}_{t-1} + \mathbf{q}_t$ and $\mathbf{w}_t = \hat{\mathbf{H}}_t(\mathbf{x}_{t-1}^{+})\mathbf{x}_t + \mathbf{r}_t$. We learn K state transition and emission matrices $\hat{\mathbf{F}}^k$ and $\hat{\mathbf{H}}^k$, and combine each one with the state dependent coefficient $\alpha^k(\mathbf{x}_{t-1}^{+})$.

$$\hat{\mathbf{F}}_t(\mathbf{x}_{t-1}^{+}) = \sum_{k=1}^{K} \alpha_t^k(\mathbf{x}_{t-1}^{+})\hat{\mathbf{F}}_t^k, \quad \hat{\mathbf{H}}_t(\mathbf{x}_{t-1}^{+}) = \sum_{k=1}^{K} \alpha_t^k(\mathbf{x}_{t-1}^{+})\hat{\mathbf{H}}_t^k \tag{3.17}$$

where a separated neural network with softmax output is utilized to learn $\alpha^k(\mathbf{x}_{t-1}^{+})$ that we call it *Dynamics Network*. This formulation enables us to follow Bayesian methodology. Despite classic LGSSMs that are not able to learn the dynamics, e.g. EKF and UKF, the trainable dynamics in the GIN are function of the states. For the notation simplicity, we have used $(\hat{\mathbf{F}}_t, \hat{\mathbf{H}}_t)$ instead of $(\hat{\mathbf{F}}_t(\mathbf{x}_{t-1}^{+}), \hat{\mathbf{H}}_t(\mathbf{x}_{t-1}^{+}))$ in the paper. Intuitively, each of k sets of $\hat{\mathbf{F}}^k, \hat{\mathbf{H}}^k$ models different dynamics, that will dominate when the corresponding element of the dynamics network is high. Additionally, this formulation help us to introduce a loss term which tries to increase the distance of each pair of $\hat{\mathbf{F}}^k, \hat{\mathbf{H}}^k$ set. We found this as a potential solution to prevent the model goes through the mode collapse and prevent converging to poor local minima during training dynamics.

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:

1- Generating time correlated noisy sequences as consecutive observations, forces the model to learn the dynamics instead of focusing on reconstruction.

2- 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.

To prevent the model being stuck into mode collapse, we provided two solutions:

1- By introducing k sets of $\mathbf{F}^k, \mathbf{H}^k$, where each set of $\mathbf{F}^k, \mathbf{H}^k$ models different dynamics, we introduce a loss term with a small constant factor which tries to increase the distance of each pair of $\mathbf{F}^k, \mathbf{H}^k$ set. Intuitively, the presence of different dynamics can easily modify the states in each update. We found this method as a potential solution to prevent the model go through the mode collapse.

2- Considering the negative distance of consecutive pairs of states as additional loss term with a small constant factor (the distance can be considered as euclidean difference of mean or KL of two consecutive states). Intuitively, this solution is forcing the states to not have overlap with each other and impose them to change in each update step.

3.8 Fitting

For the state estimation task, by implementing $p(\mathbf{w}_{1:T}|\mathbf{o}_{1:T})$, $p(\mathbf{x}_t|\mathbf{w}_{1:T})$ and $p(\mathbf{s}_t|\mathbf{x}_t)$ with encoder, smoothing parameterization and decoder, we maximise the log-likelihood of output $p(\mathbf{s}_t|\mathbf{o}_{1:T}) = \int p(\mathbf{s}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{w}_{1:T})p(\mathbf{w}_{1:T}|\mathbf{o}_{1:T})d\mathbf{x}_t d\mathbf{w}_{1:T}$,where $\mathbf{s}_t$ is the estimated state, i.e. equal to $\mathbf{o}_t^+$ in figure 3.2. For the image imputation task, in addition to the state likelihood, we add the reconstruction pseudo-likelihood for inferring images by using Bernoulli distributions as $p(\mathbf{i}_t|\mathbf{x}_t)$, i.e. the decoder in figure 3.2 maps both state $\mathbf{s}_t$ and image $\mathbf{i}_t$: $\mathbf{o}_t^+ = [\mathbf{i}_t, \mathbf{s}_t]$. Here we provide further details about the procedure of obtaining output dtribution

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

$$f_\theta(y) = h(y).exp\big(\theta.y - \psi(\theta)\big) \rightarrow e^{log(p^y(1-p)^{1-y})} = e^{y\, log(\frac{p}{1-p}) + log(1-p)} \tag{3.18}$$

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

$$D(p, \hat{p}) = \big[p\, log(\frac{p}{\hat{p}}) + (1-p) log(\frac{1-p}{1-\hat{p}})\big] \tag{3.19}$$

and minimize the deviance with respect to $\hat{p}$ as we did in (3.22).

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 (3.20) as we obtained it in (3.21).

$$-log(\mathcal{L}) \propto \frac{1}{2} log(\hat{\sigma}_\theta) + \frac{(x - \hat{x}_\theta)^2}{2\hat{\sigma}_\theta} \tag{3.20}$$

After training phase, forecasting desired number of time steps is applicable by plugging the new value $\mathbf{x}_t = \hat{\mathbf{F}}_t \mathbf{x}_{t-1}$ recursively in the model, and so on. 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 [33] 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:

$$\mathcal{L}(\mathbf{s}_{1:T}) = \sum_{t=1}^{T} \log \mathcal{N}\left(\mathbf{s_t} \middle| \text{dec}_{\text{mean}}(\mathbf{x_{t|T}}), \text{dec}_{\text{covar}}(\mathbf{\Sigma_{t|T}})\right) \tag{3.21}$$

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:

$$\mathcal{L}(\mathbf{o}^{+}_{1:T}) = \mathcal{L}(\mathbf{s}_{1:T}) + \lambda \sum_{t=1}^{T} \sum_{k=0}^{D_o} \mathbf{i}_t^{(k)} \log\big(\text{dec}_k(\mathbf{x_{t|T}})\big) + \big(1 - \mathbf{i}_t^{(k)}\big) \log(1 - \text{dec}_k(\mathbf{x_{t|T}})). \tag{3.22}$$

$\text{dec}_k(\mathbf{x}_t)$ 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 (3.21) and we abbreviate the second term as $\mathcal{L}(\mathbf{i}_{1:T})$.

Chapter 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. The training algorithms of both cases are added here:

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 $\hat{\mathbf{F}}_{1:T}$ and $\hat{\mathbf{H}}_{1:T}$ by (3.17)

$(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T})$ = *Prediction Step* $((\mathbf{x}^+_{0:T-1}, \mathbf{\Sigma}^+_{0:T-1}), \hat{\mathbf{F}}_{1:T})$

$(\mathbf{w}_{1:T}, \mathbf{r}_{1:T})$ = *encoder* $(\mathbf{o}_{1:T})$

$\mathbf{K}_{1:T}$ = $\mathbf{\Sigma}^-_{1:T}\hat{\mathbf{H}}_{1:T}\ \mathbf{M}_{1:T}\mathbf{M}^T_{1:T}$, $\quad \mathbf{M}_{1:T} = GRU^{KG}$(*Conv2D*($\mathbf{\Sigma}^-_{1:T}$), $\mathbf{r}_{1:T}$)

$\mathbf{J}_{1:T}$ = $\mathbf{\Sigma}^+_{1:T}\hat{\mathbf{F}}^T_{1:T}\mathbf{N}_{1:T}\mathbf{N}^T_{1:T}$, $\quad \mathbf{N}_{1:T} = GRU^{SG}$ (*Conv2D*($\mathbf{\Sigma}^-_{1:T}$))

$(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$ = *Filtering Step* $(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T}, \mathbf{K}_{1:T}, \mathbf{w}_{1:T}, \hat{\mathbf{H}}_{1:T})$

$(\mathbf{x}_{1:T|T}, \mathbf{\Sigma}_{1:T|T})$ = *Smoothing Step* $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T}, \mathbf{J}_{1:T}, \hat{\mathbf{F}}_{1:T})$

$\mathbf{o}^+_{1:T}$ = *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 $\hat{\mathbf{F}}_{1:T}$ and $\hat{\mathbf{H}}_{1:T}$ by (3.17)

$(\mathbf{w}_{1:T}, \mathbf{r}_{1:T})$ = MLP $(\mathbf{y}_{1:T})$

$(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T})$ = *Prediction Step* $(\mathbf{x}^+_{0:T-1}, \mathbf{\Sigma}^+_{0:T-1}, \hat{\mathbf{F}}_{1:T})$

$\mathbf{K}_{1:T} = \mathbf{\Sigma}^-_{1:T}\hat{\mathbf{H}}_{1:T}\,\mathbf{M}_{1:T}\mathbf{M}^T_{1:T}, \quad \mathbf{M}_{1:T} = GRU^{KG}(\mathbf{\Sigma}^-_{1:T}, \mathbf{r}_{1:T})$

$\mathbf{J}_{1:T} = \mathbf{\Sigma}^+_{1:T}\hat{\mathbf{F}}^T_{1:T}\mathbf{N}_{1:T}\mathbf{N}^T_{1:T}, \quad \mathbf{N}_{1:T} = GRU^{SG}(\mathbf{\Sigma}^-_{1:T})$

$(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$ = *Filtering Step* $(\mathbf{x}^-_{1:T}, \mathbf{\Sigma}^-_{1:T}, \mathbf{K}_{1:T}, \mathbf{w}_{1:T}, \hat{\mathbf{H}}_{1:T})$

$(\mathbf{x}_{1:T|T}, \mathbf{\Sigma}_{1:T|T})$ = *Smoothing Step* $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T}, \mathbf{J}_{1:T}, \hat{\mathbf{F}}_{1:T})$

$\mathbf{o}^+_{1:T}$ = 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* $((\mathbf{x}^+_{0:T-1}, \mathbf{\Sigma}^+_{0:T-1}), \mathbf{Q}_{1:T}, \mathbf{F}_{1:T})$

$\mathbf{K}_{1:T} = \mathbf{\Sigma}^-_{1:T}\mathbf{H}_{1:T}\,\mathbf{M}_{1:T}\mathbf{M}^T_{1:T}, \quad \mathbf{M}_{1:T} = GRU^{KG}(\mathbf{\Sigma}^-_{1:T}, \mathbf{r}_{1:T})$

$\mathbf{J}_{1:T} = \mathbf{\Sigma}^+_{1:T}\mathbf{F}^T_{1:T}\mathbf{N}_{1:T}\mathbf{N}^T_{1:T}, \quad \mathbf{N}_{1:T} = GRU^{SG}(\mathbf{\Sigma}^-_{1:T})$

$(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T})$ = *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* $(\mathbf{x}^+_{1:T}, \mathbf{\Sigma}^+_{1:T}, \mathbf{J}_{1:T}, \mathbf{F}_{1:T})$

$\sigma_{1:T|T}$ = *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

Table 4.1: 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	$\text{SE}^{\text{Pos}}_{x_1}$	$\text{SE}^{\text{Pos}}_{x_3}$	$\text{SE}^{\text{Pos}}_{x_2}$	$\text{SE}^{\text{Pos}}_{x_4}$	Log Likelihood
LSTM (units=50)	0.163	0.171	0.148	0.167	3.901 $\pm$ 0.706
LSTM (units=100)	0.154	0.147	0.134	0.152	4.053 $\pm$ 0.565
GRU (units=50)	0.189	0.183	0.179	0.177	3.886 $\pm$ 0.369
GRU (units=100)	0.164	0.156	0.162	0.145	3.976 $\pm$ 0.231
KVAE (n=2m)	0.193	0.188	0.178	0.149	3.679 $\pm$ 0.101
KVAE (n=3m)	0.171	0.159	0.151	0.162	3.801 $\pm$ 0.116
RKN (n=2m)	0.134	0.129	0.139	0.118	4.176 $\pm$ 0.294
$\text{LGSSM}_{\text{filter}}$($n$=3$m$)	0.125	0.119	0.121	0.107	4.192 $\pm$ 0.127
$\text{LGSSM}_{\text{smooth}}$($n$=3$m$)	0.109	0.111	0.104	0.101	4.231 $\pm$ 0.154
$\text{GIN}_{\text{filter}}$($n$=2$m$)	0.115	0.109	0.119	0.109	4.224 $\pm$ 0.105
$\text{GIN}_{\text{filter}}$($n$=3$m$)	0.093	0.091	0.098	0.089	4.329 $\pm$ 0.151
$\text{GIN}_{\text{smooth}}$($n$=2$m$)	0.091	0.104	0.101	0.092	4.308 $\pm$ 0.123
$\text{GIN}_{\text{smooth}}$($n$=3$m$)	**0.079**	**0.083**	**0.085**	**0.077**	**4.477 $\pm$ 0.168**

Table 4.2: Pendulum state estimation. By consider $n = 3m$, intuitively the last part of the state is dedicated to the acceleration information causing a more liek-lihood.

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

4.1 High Dimensional Observation with Lack of Dynamics

For this case, we make a comparison between our method, LSTM and GRU baselines, RKN [17] and some variational-based approaches, e.g. [10] and [11]. The internal state is divided into two parts, the first one determines the mean with size of n, and the second indicates the covariance with size of n^2. We include three high dimensional experiments. The first two experiments are 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. The last experiment is visual odometry task.

Table 4.3: 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 $\pm$ 1.754
SIN	-101.268 $\pm$ 0.567
KVAE (informed smooth)	-14.217 $\pm$ 0.236
KVAE (unformed smooth)	-39.260 $\pm$ 5.399
EKVAE (informed smooth)	-12.897 $\pm$ 0.524
EKVAE (unformed smooth)	-29.246 $\pm$ 3.328
RKN (informed)	-12.782 $\pm$ 0.0160
RKN (uninformed)	-12.788 $\pm$ 0.0142
LGSSM(informed smooth)	-12.695 $\pm$ 0.048
GIN (informed smooth)	-12.215 $\pm$ 0.027
GIN (unformed smooth)	-12.246 $\pm$ 0.029

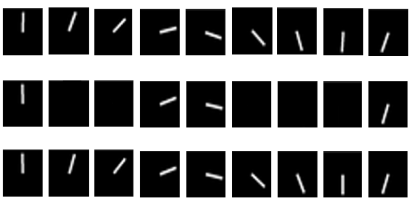


Figure 4.1: Pendulum image imputation. Each figure, beginning from up to down, indicates the ground truth, uninformed observation and the imputation results of the GIN(smoothed). Missingness is applied randomly for train and test.

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. 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

Table 4.4: 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

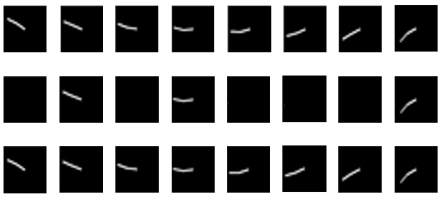


Figure 4.2: Double pendulum image imputation. Each figure, beginning from up to down, indicates the ground truth, uninformed observation and the imputation results of the GIN(smoothed).

log-likelihood and squared error (SE) of positions for single and double pendulum are given in table 4.2 and 4.1, respectively.

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 4.3 and 4.4.The GIN outperforms all the other models, although the variational inference models have more complex structures in KAVE and EKVAE. We include the results using the MSE as well, to illustrate that our approach is also competitive in prediction accuracy. The generated samples from trained smooth-filter distributions are shown here

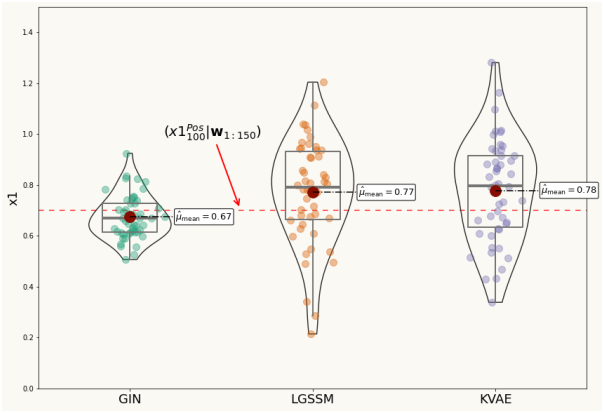


Figure 4.3: Inference for the single pendulum $x1$ position at 100-th time step. Generated samples from smoothened distribution, $f(x1_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1^{\text{Pos}}_{100}|\mathbf{w}_{1:150})$ is the ground truth state 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.

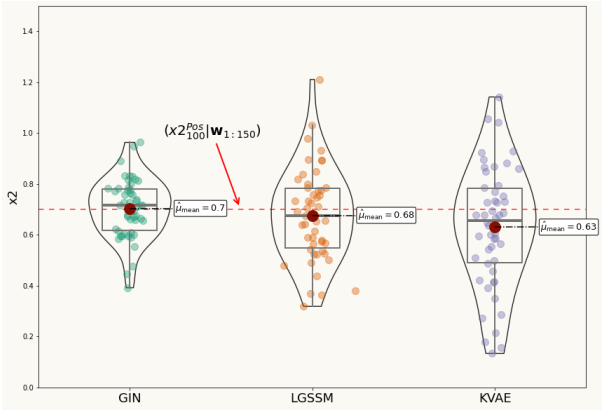
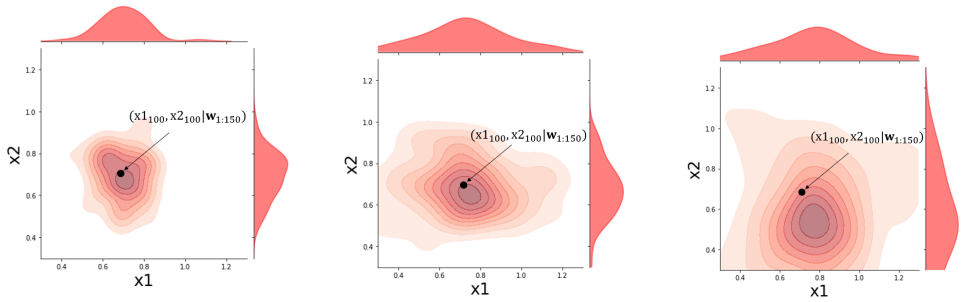


Figure 4.4: Inference for the single pendulum $x2$ position at 100-th time step. Generated samples from smoothened distribution, $f(x2_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively.



(a) GIN (b) LGSSM (c) KVAE

Figure 4.5: 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.

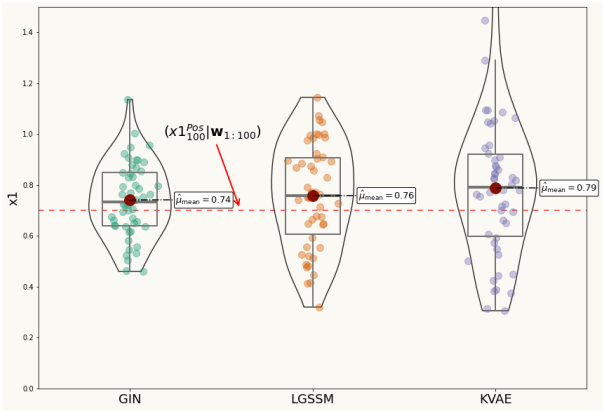


Figure 4.6: Inference for the single pendulum $x1$ position at 100-th time step. Generated samples from filter distribution, $f(x1_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1^{\text{Pos}}_{100}|\mathbf{w}_{1:100})$ is the ground truth state with distribution of $\delta(x1_{100} - 0.7)$.

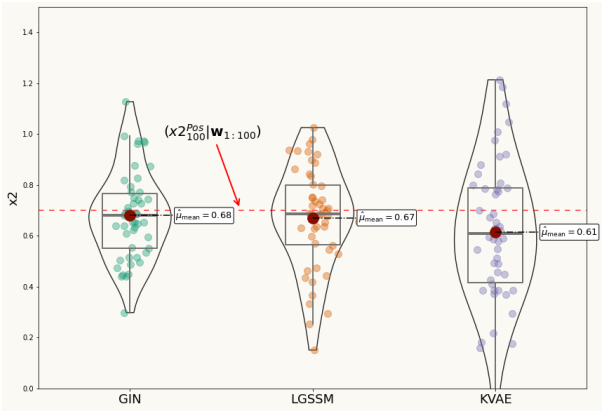
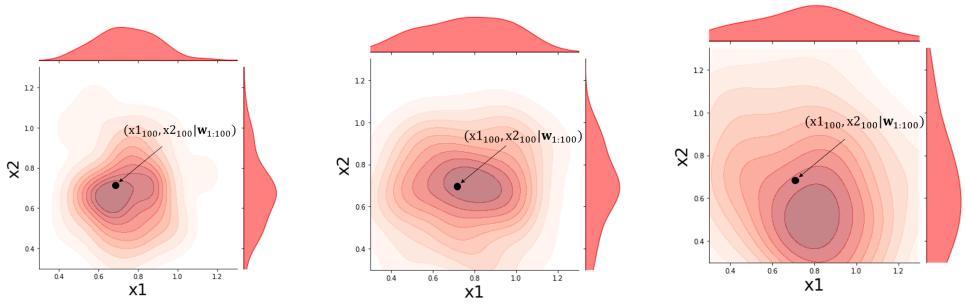


Figure 4.7: Inference for the single pendulum $x2$ position at 100-th time step. Generated samples from filter distribution, $f(x2_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively.



(a) GIN (b) LGSSM (c) KVAE

Figure 4.8: Generated samples from the trained filter 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.

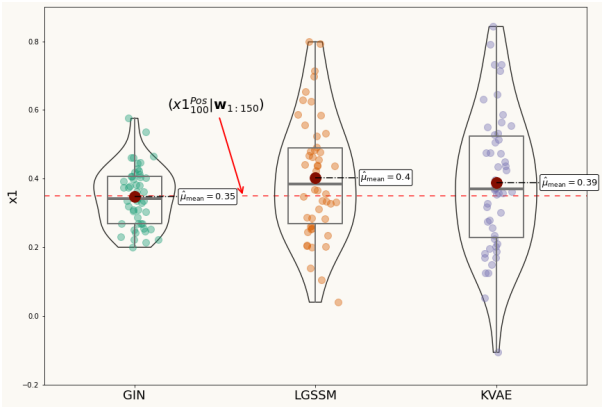


Figure 4.9: Inference for the double pendulum $x1$ position at 100-th time step. Generated samples from smoothened distribution, $f(x1_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1^{\text{Pos}}_{100}|\mathbf{w}_{1:150})$ is the ground truth state with distribution of $\delta(x1_{100} - 0.35)$.

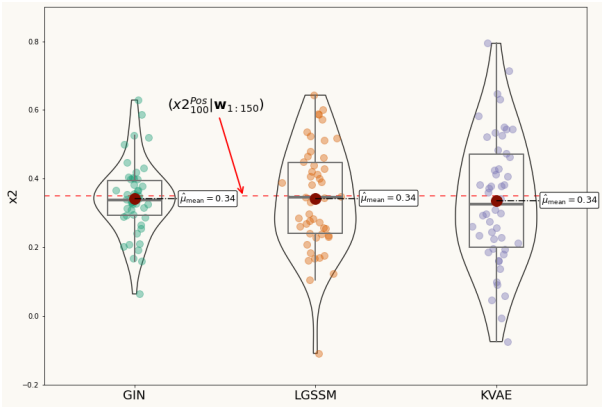
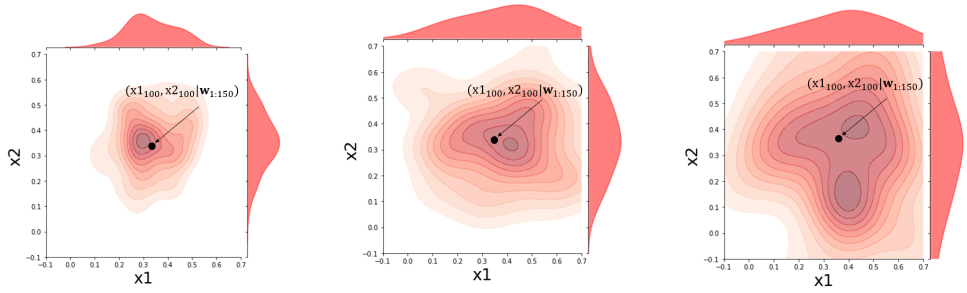


Figure 4.10: Inference for the double pendulum $x2$ position at 100-th time step. Generated samples from smoothened distribution, $f(x2_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x2^{\text{Pos}}_{100}|\mathbf{w}_{1:150})$ is the ground truth state with distribution of $\delta(x2_{100} - 0.35)$.



(a) GIN (b) LGSSM (c) KVAE

Figure 4.11: 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.

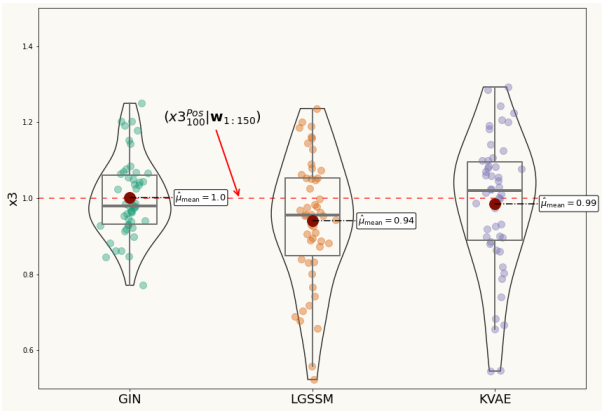


Figure 4.12: Inference for the double pendulum $x3$ position at 100-th time step. Generated samples from smoothened distribution, $f(x3_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x3^{\text{Pos}}_{100}|\mathbf{w}_{1:150})$ is the ground truth state with distribution of $\delta(x3_{100} - 1)$.

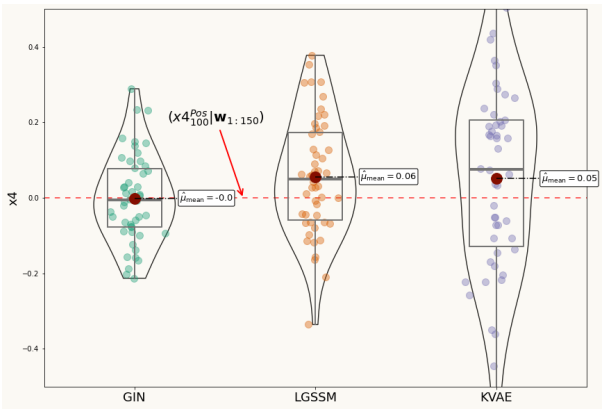
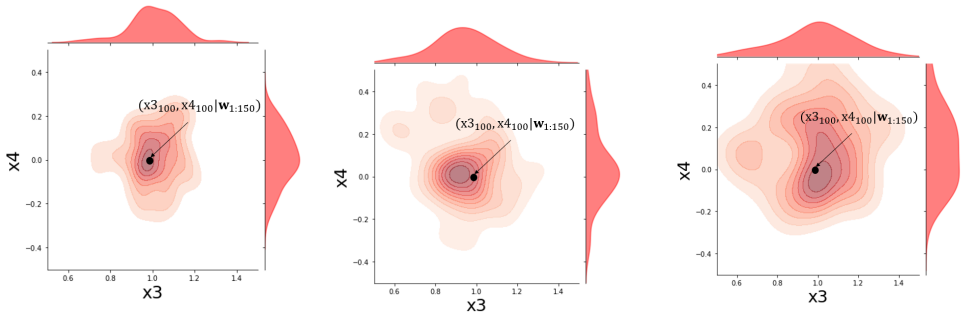


Figure 4.13: Inference for the double pendulum $x4$ position at 100-th time step. Generated samples from smoothened distribution, $f(x4_{100}|\mathbf{w}_{1:150})$, trained by the GIN, LGSSM and KVAE, respectively.



(a) GIN (b) LGSSM (c) KVAE

Figure 4.14: 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.

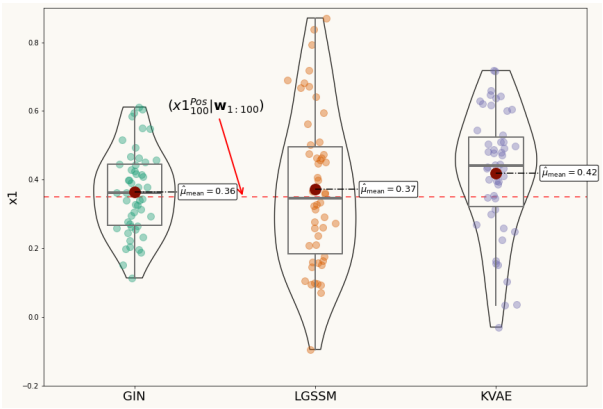


Figure 4.15: Inference for the double pendulum $x1$ position at 100-th time step. Generated samples from filter distribution, $f(x1_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1^{\text{Pos}}_{100}|\mathbf{w}_{1:100})$ is the ground truth state with distribution of $\delta(x1_{100} - 0.35)$.

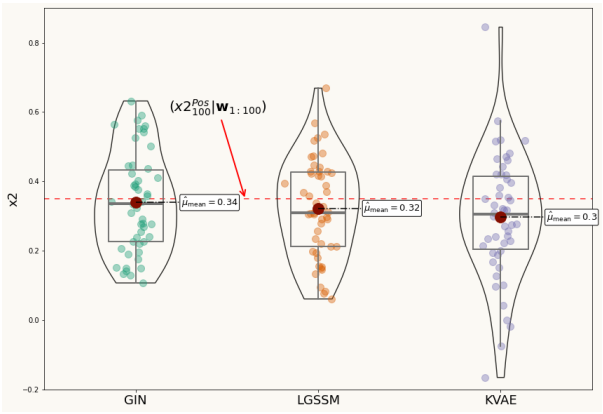
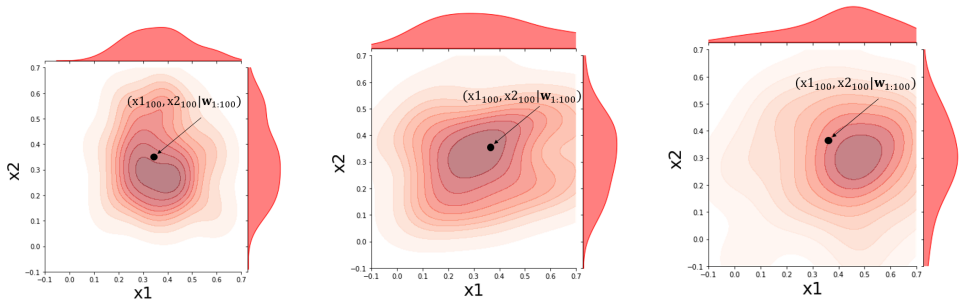


Figure 4.16: Inference for the double pendulum $x2$ position at 100-th time step. Generated samples from filter distribution, $f(x2_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x2^{\text{Pos}}_{100}|\mathbf{w}_{1:100})$ is the ground truth state with distribution of $\delta(x2_{100} - 0.35)$.



(a) GIN (b) LGSSM (c) KVAE

Figure 4.17: Generated samples from the trained filter 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.

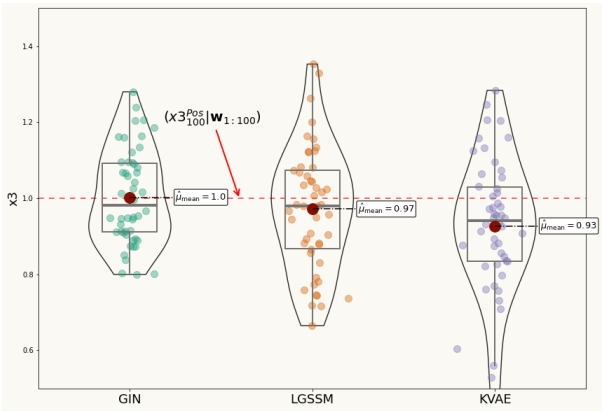


Figure 4.18: Inference for the double pendulum $x3$ position at 100-th time step. Generated samples from filter distribution, $f(x3_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x3^{\text{Pos}}_{100}|\mathbf{w}_{1:100})$ is the ground truth state with distribution of $\delta(x3_{100} - 1)$.

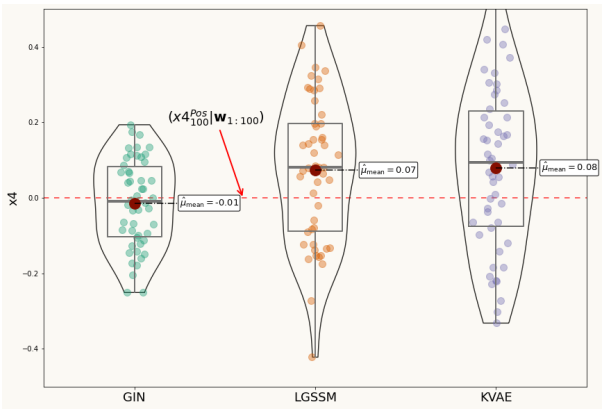
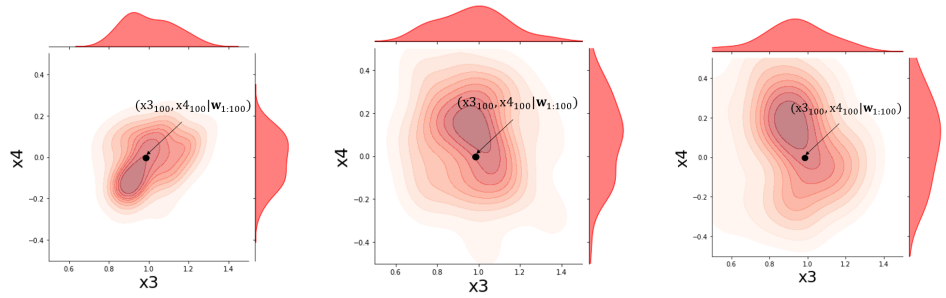


Figure 4.19: Inference for the double pendulum $x4$ position at 100-th time step. Generated samples from filter distribution, $f(x4_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively.



(a) GIN (b) LGSSM (c) KVAE

Figure 4.20: Generated samples from the trained filter 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.

The MSE results for the single and double pendulum experiments are in the table 4.5 and 4.7. In addition to (3.11), where $\mathbf{F}$ matrix includes the effects of the process noise, two other mentioned solutions introduced in section 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 4.5: 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

Table 4.6: MSE for single pendulum experiment.

Model	$\hat{\mathbf{F}}(\mathbf{Q})$	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 4.7: 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

Table 4.8: MSE for double pendulum experiment.

Model	$\hat{\mathbf{F}}(\mathbf{Q})$	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

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 [34]. 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 [35]. The obtained features are considered as the observations of the GIN, i.e. $(\mathbf{w}, \mathbf{r})$. Additionally, we compare the results with LSTM, GRU, DeepVO [36] and KVAE. The results are in table 4.9, where the common evaluation scheme for the KITTI dataset is exploited. The results of the KVAE degrades substantially as we have to reduce the size of the latent observation to prevent the complexity of matrix inversion in the smoothing-filtering, which causes an inevitable information loss.

Table 4.9: Comparison of model performance on KITTI dataset. The GIN is outperforming conventional memory cells, while its performance is comparable with DeepVO, a tailored technique for the visual odometry.

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

The generated samples from trained smooth-filter distributions are shown here

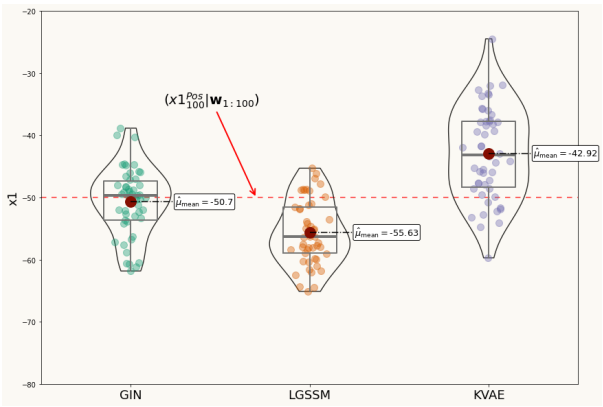


Figure 4.21: Inference for the visual odometry $x1$ position at 100-th time step. Generated samples from filter distribution, $f(x1_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x1^{\text{Pos}}_{100}|\mathbf{w}_{1:100})$ is the ground truth state with distribution of $\delta(x1_{100} + 50)$.

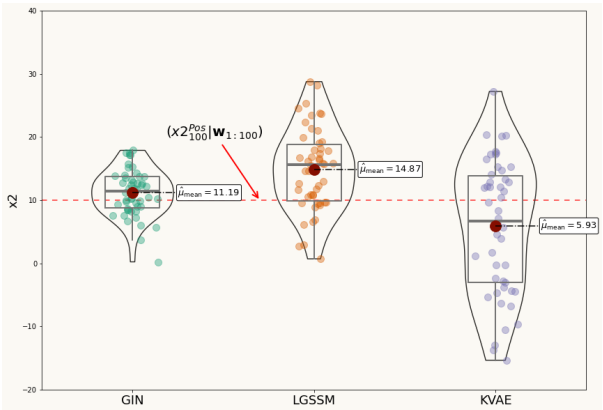
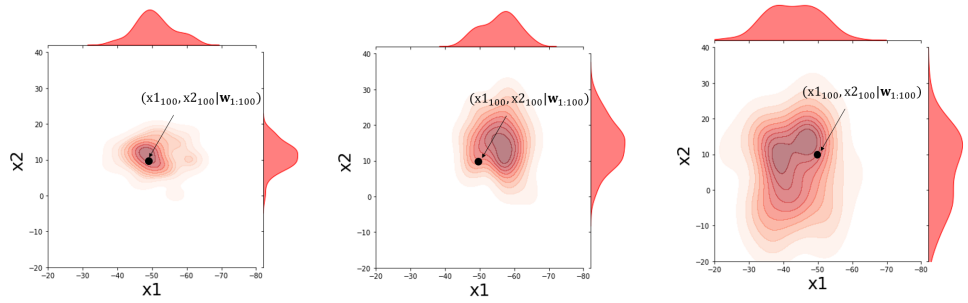


Figure 4.22: Inference for the visual odometry $x2$ position at 100-th time step. Generated samples from filter distribution, $f(x2_{100}|\mathbf{w}_{1:100})$, trained by the GIN, LGSSM and KVAE, respectively. The dashed red line $(x2^{\text{Pos}}_{100}|\mathbf{w}_{1:100})$ is the ground truth state with distribution of $\delta(x1_{100} - 10)$.



(a) GIN (b) LGSSM (c) KVAE

Figure 4.23: Generated samples from the trained filter 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.

4.2 Low Dimensional Observation with Presence of Dynamics

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, it can be a good evaluation for the GIN cell. The details of the dynamics of Lorenz attractor is explained here. There are three differential equations that model a Lorenz system, x the convection rate, y the horizontal temperature variation and z the vertical temper-

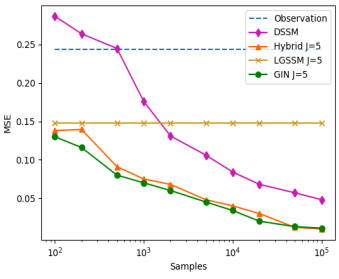


Figure 4.24: MSE of Lorenz attractor with respect to the training samples.

ature variation.

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z \tag{4.1}$$

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 (4.1) with $dt = 10^{-5}$, then we sample from it with the step time of $\Delta t = 0.01$.

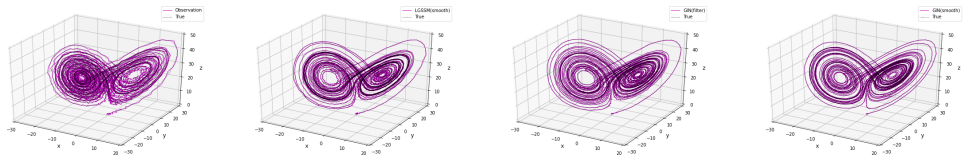
Based on the equations of the system (4.1), the state is $\mathbf{s_t} = [x_t, y_t, z_t]$ and we can write the dynamics of the system as $\mathbf{A_t}$ and obtain the transition matrix $\text{Exp}[\mathbf{A_t}] = \mathbf{F_t}$. To achieve this, we use the Taylor expansion of Exp function with 5 degrees.

$$\dot{\mathbf{s}}_t = \mathbf{A_t}\mathbf{s_t} = \begin{bmatrix} -10 & 10 & 0 \\ 28 - z & -1 & 0 \\ y & 0 & -\frac{8}{3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \text{and} \quad \mathbf{F_t} = \text{Exp}[\mathbf{A_t}] = \mathbf{I} + \sum_{j=1}^{J} \frac{(\mathbf{A_t}.\Delta t)^j}{j!} \tag{4.2}$$

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.

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 4.24. *Hybrid* is a graphical GNN based model [12] and DSSM [13] is a version of LGSSM using LSTM cells.

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. 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



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

Figure 4.25: Inferred 5k length trajectories for Lorenz attractor.

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. Inferred trajectories are in figure 4.25.

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 [37] 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.

Table 4.10: 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

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. For the dynamics of the system, we consider a uniform motion pattern with a constant velocity. The details of the dynamics of NCLT movement is explained here. We assume that the segway robot is moved with a constant velocity, that the equations for such dynamics are given by

$$\frac{\partial p_1}{\partial t} = v_1, \quad \frac{\partial p_2}{\partial t} = v_2, \quad \frac{\partial v_1}{\partial t} = 0, \quad \frac{\partial v_2}{\partial t} = 0, \quad \mathbf{x}_t = [p_1, v_1, p_2, v_2], \quad \mathbf{y}_t = [p_1, p_2]. \tag{4.3}$$

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

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \sigma^2 \begin{bmatrix} \Delta t & 0 & 0 & 0 \\ 0 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$
$$\mathbf{R} = \lambda^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.4}$$

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. 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. 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 4.10, 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.

Chapter 5

Conclusion

In this thesis, we have presented a number of contributions towards practical Bayesian learning of nonlinear time series models. Solutions have been offered to attempt to achieve a number of goals. In this thesis the proposed GIN, an approach for representation learning in both high and low dimensional SSMs, is introduced. The data flow is conducted by Bayesian filtering-smoothing, while, due to the usage of 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. 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. We can summarize the contributions and the goal of this thesis as follow

- **Bayesian learning:** The presented algorithm in this thesis has aimed at providing a posteriors over nonlinear dynamical systems. These posteriors provide an intuitive view about the uncertainty and give us some intuition about what the actual dynamical systems are given the particular time series. In most of scenarios and settings, quantifying predictive uncertainty is necessary to provide a belief about the accuracy of the decisions made with the model.

- **Nonparametric modelling:** We presented learning algorithms for models of time series based on Gaussian processes. The generated samples from smoothened-filtered distributions and their comparison with the ground truth indicates that the model has the capability of learning the underlying dynamics. Meaning that the proposed architecture is able to learn fully nonparametric models of time series which combine very large model capacity while avoiding the risk of overfitting.

- **Fast training:** In most of the practical problems, we are facing with sparse version of covariance matrices in Gaussian process systems. Models based on sparse Gaussian processes inherit most of the properties of their fully nonparametric counterparts whilst reducing the computational complexity. This means that mapping the high dimensional sparse information to low dimensional dense information is feasible without losing the general information. We achieve over this goal by introducing the convolutional operator in our structure, which learns to perform this map.

- **Ease of use:** The proposed smoothened-filtered distributions for nonlinear dynamics, provide robust and fast learning method that makes it well suited to application by non-experts. Its main advantage is that it avoids the expensive (and potentially difficult to tune) computational issues like matrix inversions that are key points in classic Gaussian process approaches.

Bibliography

[1] L. Ljung, “System identification: Theory for the users," prentice hall, new jersey,” 1999.

[2] R. H. Shumway, D. S. Stoffer, and D. S. Stoffer, *Time series analysis and its applications*. Springer, 2000, vol. 3.

[3] E. T. Jaynes, *Probability theory: The logic of science*. Cambridge university press, 2003.

[4] R. H. Shumway and D. S. Stoffer, “An approach to time series smoothing and forecasting using the em algorithm,” *Journal of time series analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[5] V. Peterka, “Bayesian approach to system identification,” in *Trends and Progress in System identification*. Elsevier, 1981, pp. 239–304.

[6] D. J. MacKay, D. J. Mac Kay *et al.*, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[7] R. Isermann and M. Münchhof, *Identification of dynamic systems: an introduction with applications*. Springer, 2011, vol. 85.

[8] P. Orbanz, “Lecture notes on bayesian nonparametrics,” *Journal of Mathematical Psychology*, vol. 56, pp. 1–12, 2012.

[9] Z. Ghahramani, "Bayesian non-parametrics and the probabilistic approach to modelling," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1984, p. 20110553, 2013.

[10] 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.

[11] 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.

[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] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[15] 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.

[16] 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.

[17] 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.

[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] 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.

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

[23] 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.

[24] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," *arXiv preprint arXiv:1502.02251*, 2015.

[25] 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.

[26] M. Karl, M. Soelch, J. Bayer, and P. Van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," *arXiv preprint arXiv:1605.06432*, 2016.

[27] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei, "Variational sequential monte carlo," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 968–977.

[28] Y. Li and S. Mandt, "Disentangled sequential autoencoder," *arXiv preprint arXiv:1803.02991*, 2018.

[29] 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.

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

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.