Sequential Inference for Deep Gaussian Process

Yali Wang
Chinese Academy of Sciences
yl.wang@siat.ac.cn

Marcus Brubaker
University of Toronto
mbrubake@cs.toronto.edu

Brahim Chaib-draa
Laval University
chaib@ift.ulaval.ca

Raquel Urtasun
University of Toronto
urtasun@cs.toronto.edu

Abstract

A deep Gaussian process (DGP) is a deep network in which each layer is modelled with a Gaussian process (GP). It is a flexible model that can capture highly-nonlinear functions for complex data sets. However, the network structure of DGP often makes inference computationally expensive. In this paper, we propose an efficient sequential inference framework for DGP, where the data is processed sequentially. We also propose two DGP extensions to handle heteroscedasticity and multi-task learning. Our experimental evaluation shows the effectiveness of our sequential inference framework on a number of important learning tasks.

1 Introduction

Gaussian processes (GP) [1] are a popular Bayesian nonparametric model due to the simplicity of learning and inference. However, traditional GPs are often limited when the underlying function exhibits complex non-stationarity [1, 2], or dependencies between the output dimensions. Many GP variants have been proposed to address non-stationarity, e.g., by designing non-stationary covariance functions [1, 2, 3], or warping GPs with different nonlinear functions [4, 5, 6]. Multi-output GP approaches have also been investigated [7, 8, 9] to better capture correlations between outputs. However, in multi-output GP approaches, the correlations between outputs remain independent of the input space. Hence their performance is often limited when data reflects input-dependent non-stationarity [10, 11] or heteroscedastic noise.

Neal [12] showed that the limit of a single-layer network as the number of hidden units goes to infinity yields a Gaussian process. Inspired by this and the success of multi-layer neural networks, Damianou et al. [11] proposed deep GPs (DGPs), where each layer of a deep network structure is modelled as a GP. DGPs can address both input-dependent non-stationarity and multi-output modeling via its flexible deep structure. More importantly, it allows us to learn multi-level representations of complex data [11, 13]. However, its network structure makes inference computationally expensive [11].

In this paper, we propose an efficient sequential inference framework for DGP models. By performing state estimation and model updates recursively, we process the input-output data pairs sequentially, allowing for efficient learning. Furthermore, we extend DGP to handle heteroscedastic outputs and multi-task learning with partially observed data. We demonstrate the effectiveness of our novel inference algorithm on a range of problems and datasets, showing both improved performance and reduced computational cost.

2 Background

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [1]. Let $\mathcal{D} = \{(\mathbf{x}^n, y^n)\}_{n=1}^N$ be our training set composed of input ($\mathbf{x} \in \mathbb{R}^{D_x}$) and output ($y \in \mathbb{R}$) pairs. We assume that the output is generated from

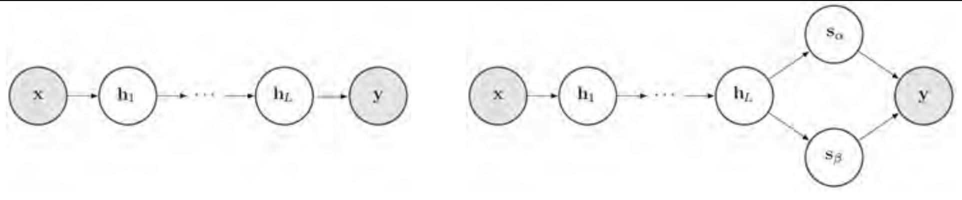
$$y = f(\mathbf{x}) + \epsilon, \tag{1}$$

with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, i.i.d. Gaussian noise with variance σ^2. Furthermore, we assume a Gaussian process prior over functions, i.e., $f(\mathbf{x}) \sim \mathcal{GP}(0, k_\theta(\mathbf{x}, \mathbf{x}'))$. For simplicity, we denote the covariance function $k_\theta(\mathbf{x}, \mathbf{x}')$ where θ is a vector of hyperparameters. We collect θ and σ^2 into $\Theta = \{\theta, \sigma^2\}$.

Given a new input $\mathbf{x}^\star$, the predictive distribution over the output $y^\star$ and the hyperparameter Θ is

$$p(y^\star, \Theta|\mathbf{x}^\star, \mathcal{D}) = p(y^\star|\mathbf{x}^\star, \mathcal{D}, \Theta)p(\Theta|\mathcal{D}), \tag{2}$$

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.



(a) Deep GP (b) Heteroscedastic Deep GP

Figure 1: The graphical model of DGP and its heteroscedastic extension.

where $p(y^\star|\mathbf{x}^\star, \mathcal{D}, \Theta)$ is Gaussian with mean $\mu_{gp\star}$ and variance $\sigma^2_{gp\star}$ [1]

$$\begin{aligned}\mu_{gp\star} &= \mathbf{k}^\star_\theta[K_\theta + \sigma^2 I]^{-1}\mathbf{y},\\ \sigma^2_{gp\star} &= k^{\star,\star}_\theta - \mathbf{k}^\star_\theta[K_\theta + \sigma^2 I]^{-1}(\mathbf{k}^\star_\theta)^T + \sigma^2.\end{aligned} \tag{3}$$

The observation vector is $\mathbf{y} = [y^1, ..., y^N]^T$. The covariance matrix K_θ is computed with $(K_\theta)_{i,j} = k_\theta(\mathbf{x}^i, \mathbf{x}^j)$, $i, j = 1, ...N$. Similarly, $k^{\star,\star}_\theta$ is computed with $k_\theta(\mathbf{x}^\star, \mathbf{x}^\star)$ and the vector $\mathbf{k}^\star_\theta$ is computed with $(\mathbf{k}^\star_\theta)_i = k_\theta(\mathbf{x}^\star, \mathbf{x}^i)$, $i = 1, ...N$.

As $p(\Theta|\mathcal{D}) \propto p(\mathbf{y}|\mathbf{x}^{1:N}, \Theta)$, a popular approach to infer Θ is to minimize the negative log likelihood

$$\begin{aligned}&-\log p(\mathbf{y}|\mathbf{x}^{1:N}, \Theta) \qquad (4)\\ &= \frac{1}{2}\mathbf{y}^T[K_\theta + \sigma^2 I]^{-1}\mathbf{y} + \frac{1}{2}\log|K_\theta + \sigma^2 I| + \frac{1}{2}n\log 2\pi.\end{aligned}$$

The matrix inversion involved in Eq. (3-4) leads to $O(N^3)$ complexity. Many approximations have been proposed to improve computational efficiency. A detailed review can be found in [14, 15].

2.1 Deep Gaussian Process

GPs are limited in their ability to learn non-stationary functions (e.g., a function with sudden jumps) [5, 6]. Furthermore, the correlations between different output variables are typically ignored if GPs are applied independently for each output variable [8, 9]. Damianou and Lawrence proposed a deep GP (DGP) where the input-output mappings in a multi-layer deep network are modelled by GPs [11]. Besides addressing the above-mentioned difficulties of GPs, DGPs allow for the flexible discovery of multiple-level representations of complex data [11, 13].

In DGPs, the output is assumed to be multi dimensional, i.e., $\mathbf{y} \in \mathbb{R}^{D_y}$ and to be generated as follows

$$\mathbf{h}_0 = \mathbf{x}, \quad \mathbf{h}_i = \mathbf{f}_i(\mathbf{h}_{i-1}) + \mathbf{v}_i, \quad (i = 1, ..., L) \tag{5}$$

$$\mathbf{y} = \mathbf{f}_y(\mathbf{h}_L) + \mathbf{v}_y, \tag{6}$$

where the latent state vector in the i-th layer is $\mathbf{h}_i \in \mathbb{R}^{D_i}$. All functions are assumed to have GP priors, i.e., $\mathbf{f}_i \sim \mathcal{GP}(0, k_{\theta_i})$ and $\mathbf{f}_y \sim \mathcal{GP}(0, k_{\theta_y})$ and noise is Gaussian, i.e., $\mathbf{v}_i \sim \mathcal{N}(0, \sigma^2_i I)$ and $\mathbf{v}_y \sim \mathcal{N}(0, \sigma^2_y I)$. For simplicity, we combine all hyper-parameters (θ_i, σ^2_i, θ_y, σ^2_y) into a vector Θ_{dgp}. The graphical model of a DGP is shown in Fig.1(a).

Algorithm 1 Sequential inference for DGP

```
1: Input:
2: The $n$-th data pair: $(\mathbf{x}^n, \mathbf{y}^n)$
3: The DGP model parameters at the $(n-1)$-th step: $M^{n-1} = \{M_1^{n-1}, \dots, M_L^{n-1}, M_y^{n-1}\}$
4: Output:
5: Estimates of the latent state at the $n$-th step: $\hat{\mathbf{S}}^n = \{\hat{\mathbf{h}}^n_{1:L}\}$
6: DGP model parameters at the $n$-th step: $M^n = \{M_1^n, \dots, M_L^n, M_y^n\}$
7: State Estimation:
8: Draw $N_p$ samples of $\mathbf{S}^n$ from (7) & (8)
9: Weight samples by (9) & (11)
10: Estimate $\hat{\mathbf{S}}^n$ by (10)
11: Model Update:
12: Update $M^{n-1}$ to $M^n$ by using GP$_{so}$ with $(\mathbf{x}^n, \hat{\mathbf{h}}^n_1)$, $\dots$, $(\hat{\mathbf{h}}^n_{L-1}, \hat{\mathbf{h}}^n_L)$, $(\hat{\mathbf{h}}^n_L, \mathbf{y}^n)$.
```

3 Sequential Inference for Deep GP

In this work, we propose a sequential inference algorithm which is able to perform learning for the DGP model effectively and efficiently. The algorithm (summarized in Alg. 1) assumes an online setting where data is presented for learning in sequence. Specifically, for each training pair, $(\mathbf{x}^n, \mathbf{y}^n)$, the algorithm consists of a state estimation phase and a model update phase.

During state estimation, the latent states $\mathbf{S}^n = \{\mathbf{h}^n_{1:L}\}$ are estimated using a sampling mechanism inspired by sequential Monte Carlo [16, 17]. These estimated latent states are then used to update all the layers of the DGP model. However, if a standard GP is used for this update, the computation required would have a complexity of $O(n^3)$ due to the inversion of the kernel matrix. To avoid the cubic complexity and to prevent the complexity from growing as more data is presented, we propose to use sparse online GP (GP$_{so}$) [18, 19] for model update. We describe these phases next.

3.1 State Estimation

In the state estimation phase, we seek to infer the latent states $\mathbf{S}^n$ given the current DGP model and the current training data $(\mathbf{x}^n, \mathbf{y}^n)$. To achieve this, we estimate $\mathbf{S}^n$, based on its posterior expectation

$$E[\mathbf{S}^n] \quad = \quad \int \mathbf{S}^n p(\mathbf{S}^n|\mathbf{x}^n, \mathbf{y}^n, M^{n-1}, \Theta_{dgp}) d\mathbf{S}^n,$$

where $M^{n-1} = \{M_1^{n-1}, \dots, M_L^{n-1}, M_y^{n-1}\}$ denotes the DGP model parameters given observations up to $n-1$, and Θ_{dgp} denotes the hyper-parameters. Unfortunately, this integral is not tractable due to the deep architecture. Instead, we approximate it using a sampling approach inspired by sequential Monte Carlo, where we draw samples of $\mathbf{S}^n$ by sampling each layer of our DGP in turn and weighting the resulting samples by the likelihood in the observation layer.

Specifically, the kth sample (also called particle) of $\mathbf{S}^n$ is drawn by sampling from the current DGP model which is built upon the history of training pairs

$$\mathbf{h}_1^n(k) \sim p(\mathbf{h}_1^n|\mathbf{x}^n, M_1^{n-1}, \Theta_{dgp}), \tag{7}$$

$$\mathbf{h}_i^n(k) \sim p(\mathbf{h}_i^n|\mathbf{h}_{i-1}^n(k), M_i^{n-1}, \Theta_{dgp}) \ (i = 2.., L). \tag{8}$$

Note that, as each layer has a GP prior, Eq. (7) and (8) are Gaussian when conditioned on its input. Hence, sampling from these distributions is straightforward. Next, the (unnormalized) weight of the kth sample is computed by using the current observation model, M_y^{n-1}, which is based on all previous observations. The value of the weight is given by

$$w^n(k) = p(\mathbf{y}^n|\mathbf{h}_L^n(k), M_y^{n-1}, \Theta_{dgp}). \tag{9}$$

The above sampling procedure is repeated to produce a set of N_p weighted samples. These samples are then used to estimate the expected latent states as

$$\hat{\mathbf{h}}_i^n = \sum_{k=1}^{N_p} \hat{w}^n(k) \mathbf{h}_i^n(k) \quad (i = 1, \dots, L), \tag{10}$$

where the normalized weight of the kth sample is

$$\hat{w}^n(k) = \frac{w^n(k)}{\sum_{k=1}^{N_p} w^n(k)}. \tag{11}$$

This estimate of the latent states, $\hat{\mathbf{S}}^n = \{\hat{\mathbf{h}}_{1:L}^n\}$, is then used in the model update phase.

3.2 Model Update

Using the observations $(\mathbf{x}^n, \mathbf{y}^n)$ and the expected latent state $\hat{\mathbf{S}}^n$, we can update the model parameters M^{n-1} to M^n. As mentioned above, if this update was performed using a full GP, the computational complexity would be cubic in n, growing quickly as the number of data points increases. Instead, we use the sparse online GP (GP$_{so}$) [18, 19] to update each layer of the DGP.

The GP$_{so}$ uses the estimated states at the n-th observation as input-output pairs $(\mathbf{x}^n, \hat{\mathbf{h}}_1^n)$, $\cdots$, $(\hat{\mathbf{h}}_{L-1}^n, \hat{\mathbf{h}}_L^n)$, $(\hat{\mathbf{h}}_L^n, \mathbf{y}^n)$ to update M^n. This consists of updating the posterior mean and covariance of the active sets in all the layers. Note that, the active set in one layer is a fixed-size subset of the estimated state set in that layer. The active set is recursively updated at each step, based on the squared prediction error. Since the technical details of GP$_{so}$ [18, 19] are standard in our model update, they are omitted here for brevity but are reviewed in our Supplementary Material.

By taking advantage of GP$_{so}$, we can maintain the complexity of both sampling from Eq.(7)-(8) and updating the DGP model as $O(N_{AC}^2)$ for each step, where N_{AC} is a fixed constant which specifies the size of the active set. In this case, the overall cost is $O(N_{AC}^2 N)$ that is linear in the number of training points N. Hence, compared to $O(N^3)$ in the full GP, GP$_{so}$ enhances the effectiveness and efficiency of our sequential inference, even for large datasets.

3.3 Prediction & Hyperparameter Learning

Given a trained DGP model $M = \{M_1, \dots, M_L, M_y\}$, one can use it to perform prediction for a new input $\mathbf{x}^\star$. This is done efficiently through sampling

$$\mathbf{h}_1^\star(k) \sim p(\mathbf{h}_1^\star|\mathbf{x}^\star, M_1, \Theta_{dgp}) \tag{12}$$

$$\mathbf{h}_i^\star(k) \sim p(\mathbf{h}_i^\star|\mathbf{h}_{i-1}^\star(k), M_i, \Theta_{dgp}) \ (i = 2, \dots, L) \tag{13}$$

where each distribution is Gaussian as before. Consequently, the predictive distribution for the output $\mathbf{y}^\star$ is the Gaussian mixture

$$\frac{1}{N_p} \sum_{k=1}^{N_p} p(\mathbf{y}^\star|\mathbf{h}_L^\star(k), M_y, \Theta_{dgp}). \tag{14}$$

Additionally, the hyperparameter vector Θ_{dgp} can be learned. After all the training pairs are processed as described, each layer of the DGP model has a fixed-size active set. Based on these small active sets, Θ_{dgp} can be efficiently refined by minimizing the negative log likelihood with a standard gradient optimization [1]. In our experiments, we iteratively perform our sequential inference and hyper-parameter learning by following the strategy in [18].

4 DGP Extensions

The proposed inference framework can be easily extended to handle more complex models and scenarios. We describe two such instances below.

4.1 Heteroscedastic Noise

Traditionally, the observation layer of a (deep) Gaussian process is homoscedastic, i.e., σ_y^2 is a constant which is independent of the input. However, many real-world data sets exhibit varying levels of uncertainty depending on the location within the input space. To account for this, we extend DGP with a heteroscedastic observation layer

$$\mathbf{s}_\alpha = \mathbf{f}_\alpha(\mathbf{h}_L) + \mathbf{v}_\alpha, \quad \mathbf{s}_\beta = \mathbf{f}_\beta(\mathbf{h}_L) + \mathbf{v}_\beta, \tag{15}$$

$$p(\mathbf{y}|\mathbf{s}_\alpha, \mathbf{s}_\beta) = \mathcal{N}(\mathbf{s}_\alpha, \exp(\mathbf{s}_\beta)), \tag{16}$$

where the input to this layer is the output, $\mathbf{h}_L$, of a DGP (see Eq. 5); the means $\mathbf{s}_\alpha \in \mathbb{R}^{D_y}$ and variances $\mathbf{s}_\beta \in \mathbb{R}^{D_y}$ of the observations are both modelled using a GP prior, i.e., $\mathbf{f}_\alpha \sim \mathcal{GP}(0, k_{\theta_\alpha})$, $\mathbf{f}_\beta \sim \mathcal{GP}(0, k_{\theta_\beta})$ and $\mathbf{v}_\alpha \sim \mathcal{N}(0, \sigma_\alpha^2 I)$, $\mathbf{v}_\beta \sim \mathcal{N}(0, \sigma_\beta^2 I)$.

We denote this extended model as the heteroscedastic DGP (HDGP) and denote the hyperparameter vector as Θ_{hdgp}=$(\theta_i, \sigma_i^2, \theta_\alpha, \sigma_\alpha^2, \theta_\beta, \sigma_\beta^2)$. The graphical model of the HDGP is shown in Figure 1(b). This model is a generalization of both heteroscedastic GPs and deep GPs. When $\mathbf{s}_\alpha = \mathbf{y}$, the HDGP reduces to an L-layer deep GP [11]. When $L = 0$, i.e., $\mathbf{x} = \mathbf{h}_L$, the HDGP reduces to a standard heteroscedastic GP [20].

The sequential inference framework can be directly applied to the HDGP model. During the state estimation phase, one draws the k-th sample of $\mathbf{s}_\alpha^n$ and $\mathbf{s}_\beta^n$ by using $\mathbf{h}_L^n(k)$ (see Eq.7-8) in the heteroscedastic layer

$$\mathbf{s}_j^n(k) \sim p(\mathbf{s}_j^n|\mathbf{h}_L^n(k), M_j^{n-1}, \Theta_{hdgp}) \quad (j = \alpha, \beta). \tag{17}$$

The unnormalized weight of the k-th sample is then computed as $w^n(k) = p(\mathbf{y}^n|\mathbf{s}_\alpha^n(k), \mathbf{s}_\beta^n(k)) = \mathcal{N}(\mathbf{s}_\alpha^n(k), \exp(\mathbf{s}_\beta^n(k)))$, and the state estimate is computed as above. In the model update phase, besides of updating $M_{1:L}^n$ of DGP, one can use GP$_{so}$ with $(\hat{\mathbf{h}}_L^n, \hat{\mathbf{s}}_\alpha^n)$, and $(\hat{\mathbf{h}}_L^n, \hat{\mathbf{s}}_\beta^n)$ to update the model parameters M_α^n and M_β^n of this heteroscedastic layer.

4.2 Multi-Task Learning

In practice, correlations between multiple outputs are often important to make good predictions [8, 9]. However, the standard GP learns each individual outputs independently without accounting for the output correlations. As a result, its predictive performance is limited, especially when some training data is only partially observed (i.e., some output dimensions are missing). In contrast, the deep network structure of a DGP allows correlations between outputs to be represented by sharing the latent layers [10, 21]. Dealing with partial observations is straightforward with our sequential inference scheme. During state estimation, the unnormalized weight (Eq. 9) is computed by using only observed dimensions of the output $\mathbf{y}_{obs}^n$

$$w^n(k) = p(\mathbf{y}_{obs}^n|\mathbf{h}_L^n(k), M_y^{n-1}, \Theta_{dgp}) \tag{18}$$

and the missing output dimensions are estimated using the same procedure as is used for estimating the latent states. Then, in the model update phase, the observed dimensions and the estimated missing dimensions form the output vector to update the model parameters M_y^n of the output layer.

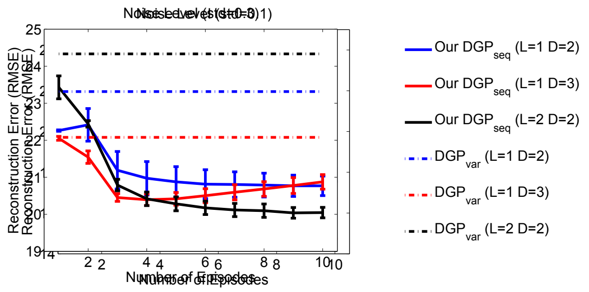


Figure 4: Dimensionality Reduction for Image Reconstruction (the *Frey* dataset): the reconstruction error as a function of the number of episodes. Note that there is no sequential inference & hyperparameter learning episode in DGP$_{var}$. The results of DGP$_{var}$ are lines. The error bar is mean±standard deviation.

5 Experiments

Here we experimentally validate the proposed approach. We mainly compare our proposed method (our DGP$_{seq}$) to variational inference for Deep GP (DGP$_{var}$) [11] and sparse online GP (GP$_{so}$) [18]. For all datasets, we normalize the output y to $[0, 1]$ and then subtract the mean. We use five randomized train/test partitions for all data sets. Based on these five partitions, we report average root mean squared error (RMSE) and mean negative log probability (MNLP) as our metrics, and average training time (seconds) as a measure of computational complexity. We use PCA to initialize the latent layers of DGP, and choose an Automatic Relevance Determination (ARD) squared exponential kernel for all approaches, i.e., $k(\mathbf{x}, \mathbf{x}') = \sigma_{ker}^2 \exp[-0.5 \sum_{i=1}^{d} c_i(\mathbf{x}_i - \mathbf{x}_i')^2]$, with the amplitude σ_{ker}^2 and the ARD weights $c_1, \cdots, c_d$.

In the following we highlight our evaluation on a number of learning tasks and datasets. The Supplementary Material contains expanded results.

5.1 Unsupervised Learning

We start our experimental evaluation with two unsupervised learning applications: learning a deep dynam-

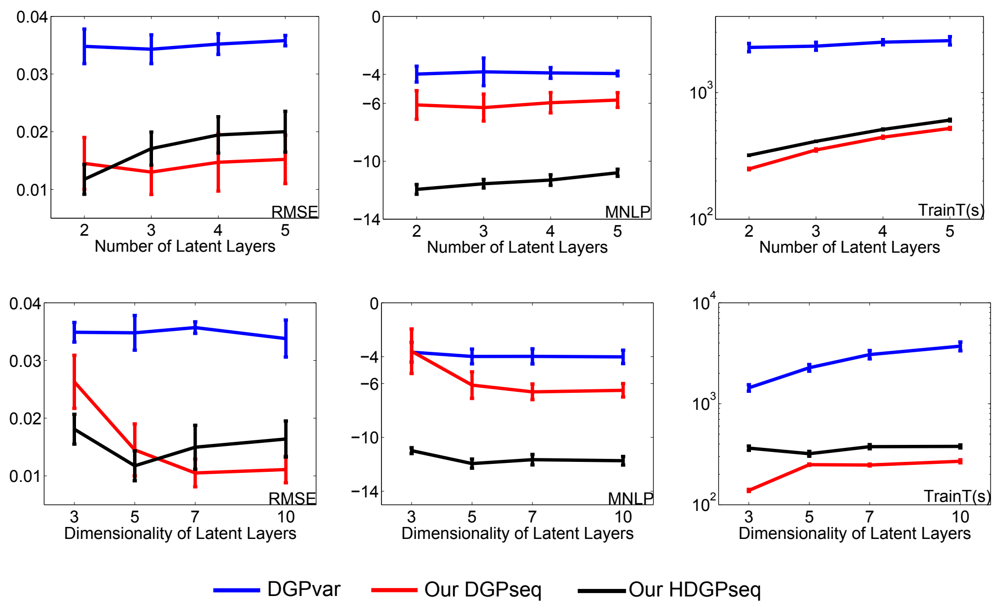


Figure 2: DGP as deep dynamical prior (the *motion* dataset): the RMSE/MNLP/training time (Column 1/2/3) when changing the number of latent layers L (Row 1) and the dimensionality of latent layers D (Row 2). The error bar is mean±standard deviation.

Data	Methods	RMSE(%)	MNLP	Train_T(s)
Motion	GP$_{so}$	1.45	-4.81	20
	our DGP$_{seq}$	1.0±0.1	-4.95±0.75	202±2
	our HDGP$_{seq}$	1.1±0.3	-5.15±0.20	236±8
Flu	GP$_{so}$	5.3	-2.42	8
	our DGP$_{seq}$	4.0±0.6	-3.09±0.32	50±2
	our HDGP$_{seq}$	3.0±0.7	-3.16±0.11	108±6
Stock	GP$_{so}$	6.4	-1.95	28
	our DGP$_{seq}$	6.4±0.1	-2.18±0.07	305±2
	our HDGP$_{seq}$	6.6±0.3	-2.41±0.04	413±4

Table 1: Missing Data in Multi-Task Learning (Motion/Flu/Stock).

ical prior for time series data and dimensionality reduction for image reconstruction.

5.1.1 Deep Dynamical Prior

DGP can be used as a flexible dynamical prior, where the input is the time step and the output is the multi-dimensional observation [22, 11]. We use three different datasets to illustrate the robustness of DGP: the *motion*, *flu* and *stock* datasets. The *motion* dataset[1] consists of 2465 time/pose pairs of five physical exercise activities (jumping jacks, two types of side twists, squats and jogging) from CMU Mocap database. Each pose is parameterized with a 62 dimensional vector containing the 3D rotations of all joints. The *flu* dataset[2] consists of 543 time/flu-activity-rate pairs from 2003-11-09 to 2014-03-30. Each flu-activity-rate is a 9 dimensional vector containing the flu rates of AB / BC / MB / NB / NL / NS / ON / SK / QC in Canada. The *stock* dataset[3] consists of 1500 time/Nasdaq index pairs collected from 1997-01-02 to 2014-11-18 (sampled every three working days). Each index is a 5 dimensional vector containing Nasdaq log-returns of Biotechnology, Composite, Industrial, Nasdaq100 and Telecommunications.

We randomly create five partitions of the data with training set sizes 1500/300/1000 for *motion*/*flu*/*stock* datasets respectively. We use $L = 2$, $D = 5$, $N_p = 100$ and $N_{AC} = 100/100/200$ as the basic settings for our DGP$_{seq}$. We run our sequential inference and hyperparameter learning twice (called two episodes) to get the hyperparameters into a reasonable region. Additionally, we run our sequential inference for both heteroscedastic GP and DGP, and denote them as our HGP$_{seq}$ and our HDGP$_{seq}$. We employ the same parameters of our DGP$_{seq}$ (whenever applicable) for our

[1]http://mocap.cs.cmu.edu/

[2]http://www.google.org/flutrends/ca/

[3]http://finance.yahoo.com/

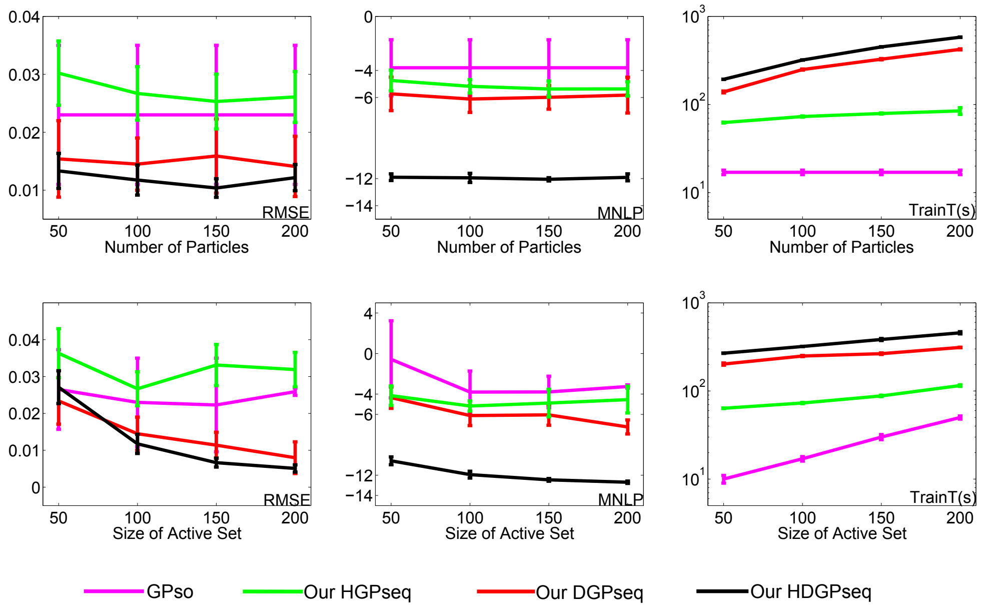


Figure 3: DGP as deep dynamical prior (the *motion* dataset): the RMSE/MNLP/training time (Column 1/2/3) when changing the number of particles N_p (Row 1) and the size of the active set N_{AC} (Row 2). The error bar is mean±standard deviation.

HGP$_{seq}$, our HDGP$_{seq}$, GP$_{so}$ [18] and DGP$_{var}$ [11]. Note that here we mainly show our evaluation on the *motion* dataset, but similar results on the other two datasets are available in the Supplemental Material.

Complexity of the Deep Model: To evaluate the robustness of DGP$_{var}$ [11] and our DGP$_{seq}$, we varied the number of latent layers L and dimensionality of each layer D. As shown in Fig. 2, for all different L and D, our DGP$_{seq}$ outperforms DGP$_{var}$ [11] in terms of both prediction error while being more computationally efficient to train.

Properties of Our Inference Framework: To study the influence of the inference algorithm parameters, we varied the number of particles, N_p, and the size of active set, N_{AC}. As shown in Fig. 3, the performance of all our approaches tend to improve (as expected) when N_p and/or N_{AC} increases. Furthermore, our DGP$_{seq}$ outperforms the baseline GP$_{so}$ [18]. It illustrates that the predictive performance of deep GP models is generally better than shallow GP models.

Heteroscedastic Learning: We now evaluate the sequential inference procedure for the heteroscedastic DGP extension. As shown in Figs. 2 and 3, our HDGP$_{seq}$ generally outperforms our DGP$_{seq}$ with a competitive RMSE but a much lower MNLP. This is due to the fact that the heteroscedastic observation layer in our HDGP$_{seq}$ is able to achieve a lower prediction uncertainty in regions of the space which have less noise. Hence, compared to our DGP$_{seq}$, our HDGP$_{seq}$ can further improve the prediction performance without significantly increasing computation.

Missing Data in Multi-Task Learning: A DGP can be used as a multi-output GP model, and we assess the performance of our framework to handle missing data. For *motion*, the observations of dimensions 27-33/49-55 (right arm / left leg) are missing during the time steps 700-720/1200-1220. For *flu*, the observations of provinces 1-5/6-9 are missing during the time steps 80-100/30-50. For *stock*, the observations of Biotechnology / Composite / Industrial / Nasdaq100 / Telecommunications are missing during the time steps 150-200/350-400/550-600/750-800/950-1000. Table 1 shows that our DGP$_{seq}$ and HDGP$_{seq}$ outperform GP$_{so}$[18] with a lower reconstruction error of missing dimensions. This demonstrates that deep GPs are able to capture correlations between different outputs due to the shared latent layers.

Comparison to the state-of-the-art: Heteroscedastic GPs and multi-task GPs have been well studied in the GP community [23, 24, 21, 25, 10]. We use the *motorcycle* dataset [26] and the multi-output *Jura* dataset [10] in order to compare with state-of-the-art approaches in these two domains. Here we show the results of *Jura* (for multi-task learning), and report the results of *motorcycle* (for heteroscedasticity) in the Supplementary Material.

In the *Jura* dataset, the input is a 2D location and the output is the measurement of cadmium, nickel and zinc concentrations. The total number of data pairs is 359, where 100 measurements of cadmium are missing. We follow the experimental settings of [10] and use mean absolute error (MAE) for evaluation, choose

Methods	MAE	Train_T(s)
GP	0.5739±0.0003	74
CMOGP	0.4552±0.0013	784
SLFM	0.4578±0.0025	792
GPRN	0.4040±0.0006	1040
GPRN-NPV1	0.4147±0.0001	130
our DGP$_{seq}$	0.4150±0.0061	21

Table 2: Comparison with State-of-the-art Multi-task GPs (the *Jura* dataset). The results of GP, CMOGP, SLFM, GPRN are from [10].

the number of latent layers $L = 1$, the dimensionality of latent layers $D = 2$. Both the number of particles and the size of active set in our DGP$_{seq}$ are 200. We run our DGP$_{seq}$ five times for one training episode and compare the results with the standard GP, convolved multiple outputs GP (CMOGP)[25], semiparametric latent factor model (SLFM)[21], GP regression networks (GPRN)[10], and GPRN with nonparametric variational inference (mode 1, GPRN-NPV1) [27]. As shown in Table 2, our DGP$_{seq}$ achieves a competitive MAE but with much less training time indicating that our sequential inference is efficient and able to capture the correlations between multiple outputs.

5.1.2 Dimensionality Reduction

Low-dimensional latent representations provide a means to avoid the so-called "curse of dimensionality" and the low-dimensional latent layers of a DGP can be understood as a form of dimensionality reduction. Inspired by [28], we use DGP for dimensionality reduction to reconstruct images from noisy observations. Towards this goal we use the *Frey* face dataset[4] which is composed of 1900 images of size $20 \times 28 = 560$. We add Gaussian noise (std dev 0.1) to the images, and use these noisy images as both the input and output of the DGP model.

Figure 4 shows the RMSE between the noiseless image and the reconstruction as a function of the number of training episodes in our learning framework. Hyperparameter learning was performed after the first episode and the hyperparameters were fixed for the remaining episodes. We chose the number of particles N_p and the size of active set N_{AC} to be 100 in our DGP$_{seq}$. To reduce the randomness of sampling, we run our DGP$_{seq}$ five times for each episode and report the average reconstruction RMSE. As shown in Fig. 4, our DGP$_{seq}$ outperforms DGP$_{var}$. Furthermore, as expected, our DGP$_{seq}$ performs better when the number of training episodes increases, or the width and the depth of the deep structure also increases. Additional qualitative results can be found in the Supplementary Material.

[4]http://www.cs.nyu.edu/∼roweis/data.html

Methods	RMSE	MNLP	Train_T(s)
GP$_{so}$	9.33 ±0.15	7.45 ±0.02	37±2
DGP$_{var}$	9.26±0.41	8.51±1.35	33486±1370
our DGP$_{seq}$	**8.86±0.23**	**7.23±0.08**	546±18

Table 3: DGP as Regressor (the *Parkinsons* dataset).

5.2 Supervised Learning

In this section, we focus our evaluation on the supervised setting for large training sets and/or high-dimensional data sets.

5.2.1 Regression

We employ the *Parkinsons* telemonitoring dataset[5], which is a six-month biomedical voice recording from 42 people with early-stage Parkinson's disease for a total of 5875 input/output pairs. The input is a 16-dimensional biomedical voice feature vector and the output is a 2-dimensional score vector (motor UPDRS score and total UPDRS score). We randomly partition the data five times and choose each time training/test sets to be of size 5000/875. The basic setting of our DGP$_{seq}$ is $L = 2$, $D = 2$, $N_p = 500$, $N_{AC} = 100$. We run our DGP$_{seq}$ for one training episode with sequential inference and hyperparameter learning. We employ the same parameters of our DGP$_{seq}$ to GP$_{so}$ and DGP$_{var}$ whenever applicable. As shown in Table 3, our DGP$_{seq}$ outperforms GP$_{so}$ and DGP$_{var}$. Furthermore, compared to DGP$_{var}$, our DGP$_{seq}$ significantly reduces the computation.

5.2.2 Classification

We evaluate our approach for classification on the *Tumor* gene expression dataset[6] and the *MNIST* Digits dataset[7]. The tumor data consists of a predefined training and test sets of sizes 144 and 46 respectively. The input contains 16,063 tumor genes (16,063 dimensional input vector) and the output contains 14 cancer classes (14 dimensional binary output vector). For the tumor data, the basic settings of our DGP$_{seq}$ are $L = 3$, $D = 12$, $N_p = 200$. N_{AC} is the size of the data as we did not employ sparsification. We use a linear kernel to reduce computation as the data is very high-dimensional. We run our DGP$_{seq}$ for five training episodes with sequential inference and hyperparameter learning. We employ the same parameters for our DGP$_{seq}$, GP$_{so}$ [18] and DGP$_{var}$ [11] whenever applicable. As shown in Table 4, our DGP$_{seq}$ achieves the best classification accuracy and also significantly outperforms DGP$_{var}$ [11] in terms of computation.

[5]http://archive.ics.uci.edu/ml/index.html

[6]http://www.genome.wi.mit.edu/

[7]http://yann.lecun.com/exdb/mnist/

Data	Methods	Acc	Train_T(s)
Tumor	GP_{so}	0.65	1
	DGP_{var}	0.50	2256
	our DGP_{seq}	**0.73**	53
MNIST	GP_{so}	0.9500	60mins
	DGP_{var}	-	-
	DVI-GPLVM	0.9405	20mins*
	our DGP_{seq}	0.9424	180mins

Table 4: Classification. For MNIST, the results of DVI-GPLVM are from [30]. (*) Distributed implementation using an unreported number of cores.

For the MNIST data, there are 60,000 training images and 10,000 test images. The input is a 28×28 image (784 dimensional input vector) and the output contains 0-9 digits (10 dimensional binary output vector). Classification accuracy on the test set (i.e., the percentage of correctly classified examples) is evaluation metric. The basic settings of our DGP_{seq} are $L = 1$, $D = 400$, $N_p = 100$ and $N_{AC} = 1000$. The covariance function of GP is chosen as the 1st-order arc-cosine kernel [29]. Additionally, instead of using the latent state h as the output of each layer, we use the logistic activation function of h, i.e., $(1 + e^{-h})^{-1}$ as the output of each layer, which mimics neural networks for classification. Note that this modification is simple to accommodate in our inference algorithm by simply propagating the particles through the activation function when sampling. We run our DGP_{seq} for one training episode. We employ the same parameters for GP_{so} [18] whenever applicable. As this dataset is large-size, DGP_{var} [11] is infeasible. Instead, we compare our DGP_{seq} to the state-of-the-art scalable GPLVM with distributed variational Inference (DVI-GPLVM) [30]. As shown in Table 4, our DGP_{seq} outperforms DVI-GPLVM in terms of accuracy with a comparable amount of training time (considering that their reported timing was parallelized while our implementation is currently serial). GP_{so} outperforms both DVI-GPLVM and our DGP_{seq} on this dataset, however as noted in [30], the primary purpose here is to demonstrate the scalability of the approach. These results demonstrate the ability of our approach to scale to both large and high dimensional datasets.

6 Related Work

Many GP variants have been proposed to address non-stationarity. The ideas are primarily based on designing non-stationary covariance functions [1, 2, 3], or warping GP with different nonlinear functions [4, 5, 6]. However, these GP approaches are designed for single-output modelling. Hence, their performance is often limited for multi-output modelling, because the correlations between outputs are ignored. Multi-task GP approaches have been investigated [7, 8, 9] with the motivation that dependencies between outputs can improve the prediction accuracy. However, the correlations between outputs remain independent of the input space and the performance of these methods is often limited when data reflect non-stationarity [10].

Several GP based latent variable models [22, 31, 32, 33, 34] and GP based network models [10, 21] can be used to represent non-stationarity and capture multi-output correlations, by learning a shared nonlinear latent space for multiple outputs. The DGP model [11] can be seen as a multi-layer generalization of these latent variable models. Due to the intractability of the posterior in DGP, variational approximations [11, 24, 32, 33] are commonly used for inference. However, these approximate inference frameworks are often computationally expensive, especially when the size of the datasets increases. Some extensions have been proposed by using the distributed variational computation [30] and the nested variational compression [28].

Here we proposed a novel and efficient inference framework which processes training pairs in sequence. The framework is flexible, potentially allowing us to exploit other deep learning techniques [35, 36] such as the ReLU activation functions and dropout. Furthermore, our sequential inference framework opens the possibility of using GPs inside other network structures from the neural network community (e.g., CNNs, RNNs, LSTMs) [36]. Finally, our HDGP can be seen as a generalization of the DGP model [11] and the heteroscedastic GP model (HGP) [20, 23, 24].

7 Conclusion

In this paper, we develop an efficient sequential inference framework for the DGP model. By recursively performing state estimation and model update, our sequential inference framework allows us to process the training pairs in a Bayesian forward-backward fashion to maintain computation efficiency. Furthermore, we show the generality of the inference framework by applying it to heteroscedastic noise and multi-task learning. In the future, it would be interesting to perform our sequential inference for input-connected architectures of deep GP [13] and other deep networks [36].

Acknowledgements

Yali Wang is supported by National Natural Science Foundation of China (61502470).

References

[1] C. E. Rasmussen, C. K. I. Williams, Gaussian Process for Machine learning, MIT Press, 2006.

[2] C. J. Paciorek, M. J. Schervish, Nonstationary Covariance Functions for Gaussian Process Regression, in: NIPS, 2004.

[3] A. M. Schmidt, A. O'Hagan, Bayesian Inference for Non-stationary Spatial Covariance Structure via Spatial Deformations, Journal of the Royal Statistical Society: Series B 65 (3) (2003) 743 758.

[4] E. Snelson, C. E. Rasmussen, Z. Ghahramani, Warped Gaussian Processes, in: NIPS, 2004.

[5] R. P. Adams, O. Stegle, Gaussian Process Product Models for Nonparametric Nonstationarity, in: ICML, 2008.

[6] M. Lázaro-Gredilla, Bayesian Warped Gaussian Processes, in: NIPS, 2012.

[7] P. Boyle, M. Frean, Dependent gaussian processes, in: NIPS, 2004.

[8] E. V. Bonilla, K. M. A. Chai, C. K. I. Williams, Multi-task Gaussian Process Prediction, in: NIPS, 2008.

[9] M. Alvarez, N. D. Lawrence, Sparse Convolved Gaussian Processes for Multi-output Regression, in: NIPS, 2008.

[10] A. G. Wilson, D. A. Knowles, Z. Ghahramani, Gaussian Process Regression Networks, in: ICML, 2012.

[11] A. C. Damianou, N. D. Lawrence, Deep Gaussian Processes, in: AISTATS, 2013.

[12] R. M. Neal, Bayesian Learning for Neural Networks, Springer-Verlag, 1996.

[13] D. Duvenaud, O. Rippel, R. P. Adams, Z. Ghahramani, Avoiding Pathologies in Very Deep Networks, in: AISTATS, 2014.

[14] J. Quinonero-Candela, C. E. Rasmussen, A Unifying View of Sparse Approximate Gaussian Process Regression, Journal of Machine Learning Research 6 (2005) 1939 1959.

[15] K. Chalupka, C. K. I. Williams, I. Murray, A Framework for Evaluating Approximation Methods for Gaussian Process Regression, JMLR.

[16] A. Doucet, N. de Freitas, N. Gordon, Sequential Monte Carlo Methods in Practice, Springer, 2001.

[17] O. Cappé, S. J. Godsill, E. Moulines, An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo, Proceedings of the IEEE 95 (5) (2007) 899–924.

[18] L. Csató, M. Opper, Sparse Online Gaussian Processes, Neural Computation 14 (3) (2002) 641–668.

[19] S. V. Vaerenbergh, M. Lázaro-Gredilla, I. Santamaría, Kernel Recursive Least-Squares Tracker for Time-Varying Regression, IEEE Transactions on Neural Networks and Learning Systems 23 (8) (2012) 1313 – 1326.

[20] P. W. Goldberg, C. K. I. Williams, C. M. Bishop, Regression with Input-Dependent Noise: A Gaussian Process Treatment, in: NIPS, 1998.

[21] Y. W. Teh, M. Seeger, M. I. Jordan, Semiparametric Latent Factor Models, in: AISTATS, 2005.

[22] N. D. Lawrence, A. J. Moore, Hierarchical Gaussian Process Latent Variable Models, in: ICML, 2007.

[23] K. Kersting, C. Plagemann, P. Pfaff, W. Burgard, Most Likely Heteroscedastic Gaussian Process Regression, in: ICML, 2007.

[24] M. Lázaro-Gredilla, M. K. Titsias, Variational Heteroscedastic Gaussian Process Regression, in: ICML, 2011.

[25] M. A. Alvarez, N. Lawrence, Computationally efficient convolved multiple output gaussian processes, JMLR 12 (2011) 1459–1500.

[26] B. W. Silverman, Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting, Journal of the Royal Statistical Society. Series B (Methodological) 47 (1) (1985) 1–52.

[27] T. V. Nguyen, E. V. Bonilla, Efficient Variational Inference for Gaussian Process Regression Networks, in: AISTATS, 2013.

[28] J. Hensman, N. Lawrence, Nested Varitaional Compression in Deep Gaussian Processes, in: http://arxiv.org/pdf/1412.1370.pdf, 2014.

[29] Y. Cho, L. K. Saul, Kernel Methods for Deep Learning, in: NIPS, 2009.

[30] Y. Gal, M. van der Wilk, C. E. Rasmussen, Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models, in: NIPS, 2014.

[31] N. Lawrence, Probabilistic non-linear principal component analysis with gaussian process latent variable models, JMLR 6 (2005) 1783–1816.

[32] M. K. Titsias, N. Lawrence, Bayesian Gaussian Process Latent Variable Model, in: AISTATS, 2010.

[33] A. C. Damianou, M. K. Titsias, N. Lawrence, Variational Gaussian Process Dynamical Model, in: NIPS, 2011.

[34] Y. Wang, M. Brubaker, B. Chaib-draa, R. Urtasun, Bayesian Filtering with Online Gaussian Process Latent Variable Models, in: UAI, 2014.

[35] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: NIPS, 2012.

[36] Y. Bengio, I. J. Goodfellow, A. Courville, Deep Learning (2015).

Supplementary Material: Sequential Inference for Deep Gaussian Process

Yali Wang
Chinese Academy of Sciences
yl.wang@siat.ac.cn

Marcus Brubaker
University of Toronto
mbrubake@cs.toronto.edu

Brahim Chaib-draa
Laval University
chaib@ift.ulaval.ca

Raquel Urtasun
University of Toronto
urtasun@cs.toronto.edu

1 Sparse Online Gaussian Process

In this section we briefly review sparse online GPs (GP$_{so}$) [1, 2]. The key idea is to learn GPs recursively by updating the posterior mean and covariance of the training set $\{(\mathbf{x}^n, y^n)\}_{n=1}^N$ in a sequential fashion. This online procedure is coupled with a sparsification mechanism in which a fixed-size subset of the training set (called the active set) is iteratively selected to avoid the unbounded computation growth of updating.

Specifically, the model parameters at the $(n-1)$-th step of GP$_{so}$ are [1, 2]

$$M^{n-1} = \{\tilde{\mathcal{X}}^{n-1}, \mu^{n-1}, \Sigma^{n-1}, Q^{n-1}\},$$

where $\tilde{\mathcal{X}}^{n-1}$ is the active set which is a training subset selected from the first $(n-1)$ training pairs, $\mathcal{N}(\mu^{n-1}, \Sigma^{n-1})$ is the posterior over $\tilde{\mathcal{X}}^{n-1}$, and Q^{n-1} is the inverse covariance matrix of $\tilde{\mathcal{X}}^{n-1}$.

Once the n-th training pair $(\mathbf{x}^n, y^n)$ is available, the model parameters is updated from M^{n-1} to M^n, based on the following strategy.

1.1 Update at the n-th step

Firstly, the following update is performed to take the information of $(\mathbf{x}^n, y^n)$ into account [2],

$$\gamma_n^2 = k(\mathbf{x}^n, \mathbf{x}^n) - \mathbf{k}_{n-1}^T(\mathbf{x}^n)\mathbf{q}_n, \tag{1}$$

$$\mu^n = \begin{bmatrix} \mu^{n-1} \\ \mathbf{q}_n^T \mu^{n-1} \end{bmatrix} + \frac{y^n - \mathbf{q}_n^T \mu^{n-1}}{\eta_n^2} \psi_n, \tag{2}$$

$$\Sigma^n = \begin{bmatrix} \Sigma^{n-1} & \Sigma^{n-1}\mathbf{q}_n \\ \mathbf{q}_n^T \Sigma^{n-1} & \gamma_n^2 + \mathbf{q}_n^T \Sigma^{n-1} \mathbf{q}_n \end{bmatrix} - \frac{\psi_n \psi_n^T}{\eta_n^2}, \tag{3}$$

where the relevant computation quantities are

$$\mathbf{q}_n = Q_{n-1}\mathbf{k}_{n-1}(\mathbf{x}^n),$$
$$\psi_n = \begin{bmatrix} \Sigma^{n-1}\mathbf{q}_n \\ \gamma_n^2 + \mathbf{q}_n^T \Sigma^{n-1}\mathbf{q}_n \end{bmatrix},$$

and the i-th entry of the vector $\mathbf{k}_{n-1}(\mathbf{x}^n)$ is computed by $k(\mathbf{x}^n, \mathbf{x}^i)$ and $\mathbf{x}^i$ is the i-th input in $\tilde{\mathcal{X}}^{n-1}$.

Next, depending on the value of γ_n^2 in Eq.(1), we decide if the n-th training point is added to the active set and how to further refine the model parameters M^n.

• When $\gamma_n^2 < \delta$ ($\delta = 10^{-6}$ in this work), the training pair $(\mathbf{x}^n, y^n)$ is not added to the active set. As a result, μ^n and Σ^n in Eq.(2-3) are reduced to

$$\mu^n \leftarrow [\mu^n]_{-n}, \quad \Sigma^n \leftarrow [\Sigma^n]_{-n,-n}, \tag{4}$$

where $[\cdot]_{-n}$ deletes the n-th entry of a vector; $[\cdot]_{-n,-n}$ deletes the n-th row and column of a matrix. Additionally, since the active set does not change $\tilde{\mathcal{X}}^n = \tilde{\mathcal{X}}^{n-1}$, the inverse covariance matrix is $Q^n = Q^{n-1}$.

• When $\gamma_n^2 \geq \delta$, the training pair $(\mathbf{x}^n, y^n)$ is added to the active set, i.e., $\tilde{\mathcal{X}}^n = \tilde{\mathcal{X}}^{n-1} \cup (\mathbf{x}^n, y^n)$. The μ^n and Σ^n are computed using Eq.(2-3). The inverse covariance matrix is then updated as follows

$$Q^n = \begin{bmatrix} Q^{n-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + \frac{1}{\gamma_n^2} \begin{bmatrix} \mathbf{q}_n \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{q}_n \\ -1 \end{bmatrix}^T. \tag{5}$$

In this case, we have to make sure that the size of the active set does not exceed our budget. If the size of $\tilde{\mathcal{X}}^n$ exceeds N_{AC}, we remove the data pair that has the lowest squared prediction error

$$k = \arg\min_k \left(\frac{[Q^n \mu^n]_k}{[Q^n]_{k,k}}\right)^2,$$

where $[\cdot]_k$ is the k-th entry of a vector, $[\cdot]_{k,k}$ is the k-th diagonal entry of a matrix. Consequentially, the active set $\tilde{\mathcal{X}}^n$ is reduced to $\tilde{\mathcal{X}}^n \leftarrow \tilde{\mathcal{X}}^n \setminus (\mathbf{x}^k, y^k)$ and μ^n, Σ^n, Q^n in Eq.(2),(3),(5) are updated to be

$$\mu^n \leftarrow [\mu^n]_{-k},$$
$$\Sigma^n \leftarrow [\Sigma^n]_{-k,-k},$$
$$Q^n \leftarrow [Q^n]_{-k,-k} - \frac{[Q^n]_{-k,k}[Q^n]_{-k,k}^T}{[Q^n]_{k,k}},$$

where $[\cdot]_{-k,k}$ is the k-th column of a matrix without the entry in the k-th row.

1.2 Prediction at the n-th step

Based on the model parameters M^n at the n-th step, the predictive distribution at a given test input $\mathbf{x}^\star$ is Gaussian [1, 2]

$$p(y^\star|\mathbf{x}^\star, M^n) = \mathcal{N}(\mu_{sogp\star}, \sigma^2_{sogp\star}) \tag{6}$$

with mean

$$\mu_{sogp\star} = \mathbf{q}_\star^T \mu^n, \quad \mathbf{q}_\star = Q^n \mathbf{k}_n(\mathbf{x}^\star)$$

and variance

$$\begin{aligned}\sigma^2_{sogp\star} =& \sigma^2 + k(\mathbf{x}^\star, \mathbf{x}^\star) + \mathbf{q}_\star^T \Sigma^n \mathbf{q}_\star \\ &- \mathbf{k}_n^T(\mathbf{x}^\star) Q^n \mathbf{k}_n(\mathbf{x}^\star)\end{aligned}$$

where σ^2 is the noise variance, the i-th entry of the vector $\mathbf{k}_n(\mathbf{x}^\star)$ is computed by $k(\mathbf{x}^\star, \mathbf{x}^i)$ and $\mathbf{x}^i$ is the i-th input in $\tilde{\mathcal{X}}^n$.

2 Experiments

In this section, we describe the full set of experiments we did to validate our approach. We mainly perform our sequential inference for Deep GP (our DGP$_{seq}$) and compare it to variational inference for Deep GP (DGP$_{var}$) [3] and sparse online GP (GP$_{so}$) [1]. Unless otherwise stated we keep the experimental settings (which we now describe) consistent. We normalize the output y of all the data sets to $[0, 1]$ and then subtract the mean. We use five random train/test partitions for all the data sets. Based on these five partitions, we report average root mean squared error (RMSE) and mean negative log probability (MNLP) as a measure of predictive error, and report average training time (second) as a measure of computational complexity. We use PCA to initialize the latent layers of DGP, and choose an Automatic Relevance Determination (ARD) squared exponential kernel for all relevant approaches,

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2_{ker} \exp[-0.5 \sum\nolimits_{i=1}^{d} c_i (\mathbf{x}_i - \mathbf{x}'_i)^2]$$

where σ^2_{ker} is the amplitude and $c_1, \cdots, c_d$ are the ARD weights.

In the following we demonstrate the effectiveness of our approach for a number of important learning tasks in the GP community.

2.1 DGP for Unsupervised Learning

We start our experimental evaluation with two unsupervised learning applications: learning a deep dynamical prior for time series data and dimensionality reduction for image reconstruction.

2.1.1 DGP as Deep Dynamical Prior

DGP can be used as a flexible deep dynamical prior, where the input is the time step and the output is the multi-dimensional observation [4, 3]. We use three different datasets to illustrate the robustness of DGP: the *motion*, *flu* and *stock* datasets. The *motion* dataset[1] consists of 2465 time/pose pairs of five physical exercise activities (jumping jacks, two types of side twists, squats and jogging) from CMU Mocap database. Each pose is parameterized with a 62 dimensional vector containing the 3D rotations of all joints. The *flu* dataset[2] consists of 543 time/flu-activity-rate pairs from 2003-11-09 to 2014-03-30. Each flu-activity-rate is a 9 dimensional vector containing the flu rates of AB / BC / MB / NB / NL / NS / ON / SK / QC in Canada. The *stock* dataset[3] consists of 1500 time/Nasdaq index pairs collected from 1997-01-02 to 2014-11-18 (sampled every three working days). Each index is a 5 dimensional vector containing Nasdaq log-returns of Biotechnology, Composite, Industrial, Nasdaq100 and Telecommunications.

We randomly create five partitions of the data with training set sizes 1500/300/1000 for *motion*/*flu*/*stock* datasets respectively. We use $L = 2$, $D = 5$, $N_p = 100$ and $N_{AC} = 100/100/200$ as the basic settings for our DGP$_{seq}$. We run our sequential inference and hyperparameter learning twice (called two episodes) to get the hyperparameters into a reasonable region. Additionally, we run our sequential inference for both heteroscedastic GP and DGP, and denote them as our HGP$_{seq}$ and our HDGP$_{seq}$. We employ the same parameters of our DGP$_{seq}$ (whenever applicable) to our HGP$_{seq}$, our HDGP$_{seq}$, GP$_{so}$ [1] and DGP$_{var}$ [3].

Complexity of the Deep Model: We evaluate DGP$_{var}$ [3] and our DGP$_{seq}$ when varying the number of latent layers L and dimensionality of each layer D. The results are shown in Figure 1 and 2. For the motion and flu datasets, our DGP$_{seq}$ outperforms the basic DGP$_{var}$[3] in terms of both prediction error and training efficiency. This illustrates that our sequential inference is more effective than the variational inference of [3]. For the stock dataset, our DGP$_{seq}$ (when changing L) and DGP$_{var}$[3] (when changing D) tend to achieve a poor performance because of the strong heteroscedasticity in this data. Additionally, in this data, the RMSE is similar for all approaches and settings. This might be because the stock observations are log-returns in which the fluctuation of the underlying function is flat. All the deep GP models can capture this fluctuation, and thus produce similar RMSEs.

[1]http://mocap.cs.cmu.edu/
[2]http://www.google.org/flutrends/ca/
[3]http://finance.yahoo.com/

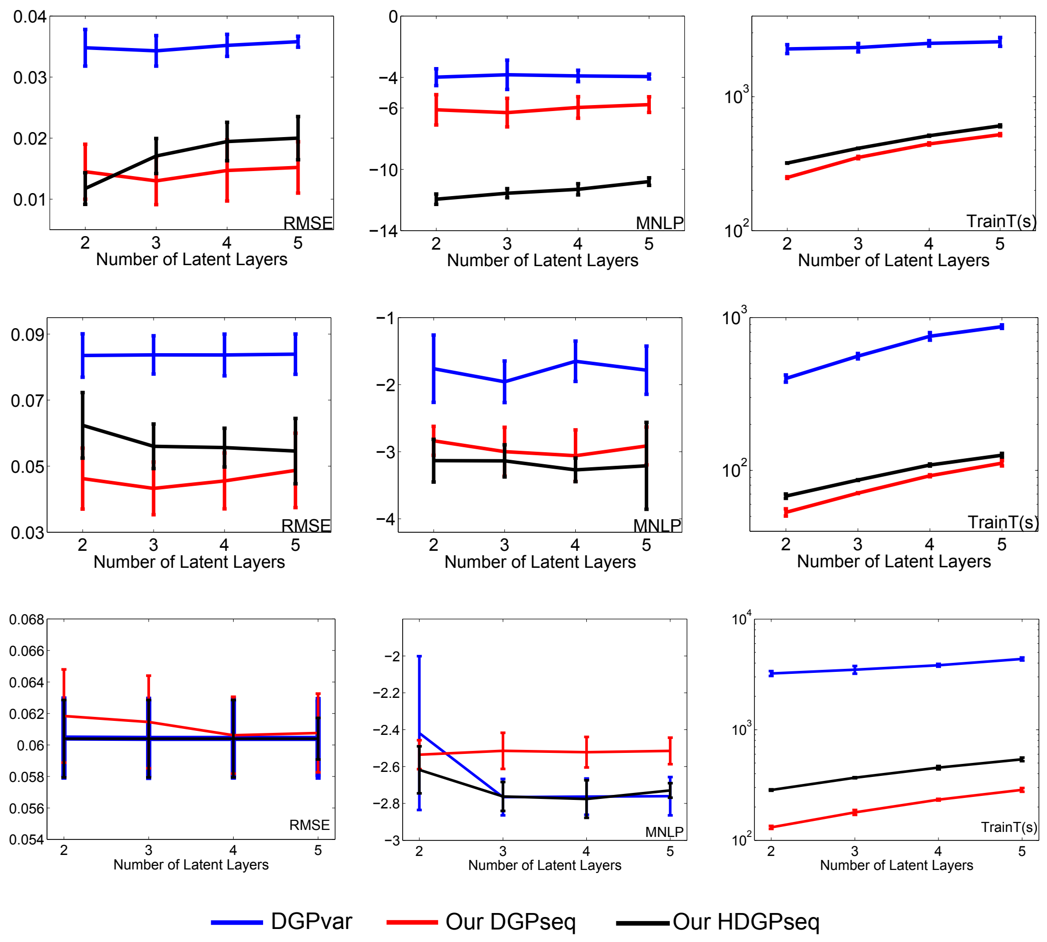


Figure 1: DGP as Deep Dynamical Prior (Robustness to Complexity of the Deep Model): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the number of latent layers L. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation.

Properties of Our Inference Framework: We next evaluate the influence of the number of particles, N_p, and the size of the active set, N_{AC} in our sequential inference framework. The results are shown in Figure 3 and 4. As expected, the performance of all our approaches tend to improve when N_p and/or N_{AC} increases. Additionally, our DGP$_{seq}$ outperforms GP$_{so}$ [1]. This illustrates that the predictive performance of deep GP models is generally better than shallow GP models.

Heteroscedastic Learning: We now evaluate the sequential inference procedure for the heteroscedastic DGP extension. As shown in Figure 1, 2, 3 and 4, our HDGP$_{seq}$ generally outperforms our DGP$_{seq}$ with a competitive RMSE but a lower MNLP. This is due to the fact that the heteroscedastic observation layer in our HDGP$_{seq}$ is able to achieve a lower prediction uncertainty in regions of the space which have less noise. Hence, compared to our DGP$_{seq}$, our HDGP$_{seq}$ can further improve the prediction performance without significantly increasing computation.

Missing Data in Multi-task Learning: Since DGP can be used as a multi-output GP model, we assess the performance of our approaches for missing data cases. For *motion*, the observations of dimensions 27-33/49-55 (right arm / left leg) are missing during the time steps 700-720/1200-1220. For *flu*, the observations of provinces 1-5/6-9 are missing during the time steps 80-100/30-50. For *stock*, the observations of Nasdaq Biotechnology / Nasdaq Composite / Nasdaq Industrial / Nasdaq100 / Nasdaq Telecommunications are missing during the time steps 150-200/350-400/550-600/750-800/950-1000. We compare our DGP$_{seq}$ and HDGP$_{seq}$ to the baseline GP$_{so}$ [1]. Table 1 shows that our DGP$_{seq}$ and HDGP$_{seq}$ outperform GP$_{so}$ with a lower reconstruction error of missing dimensions. This is primarily due to the fact that deep structures can capture correlations between different outputs via

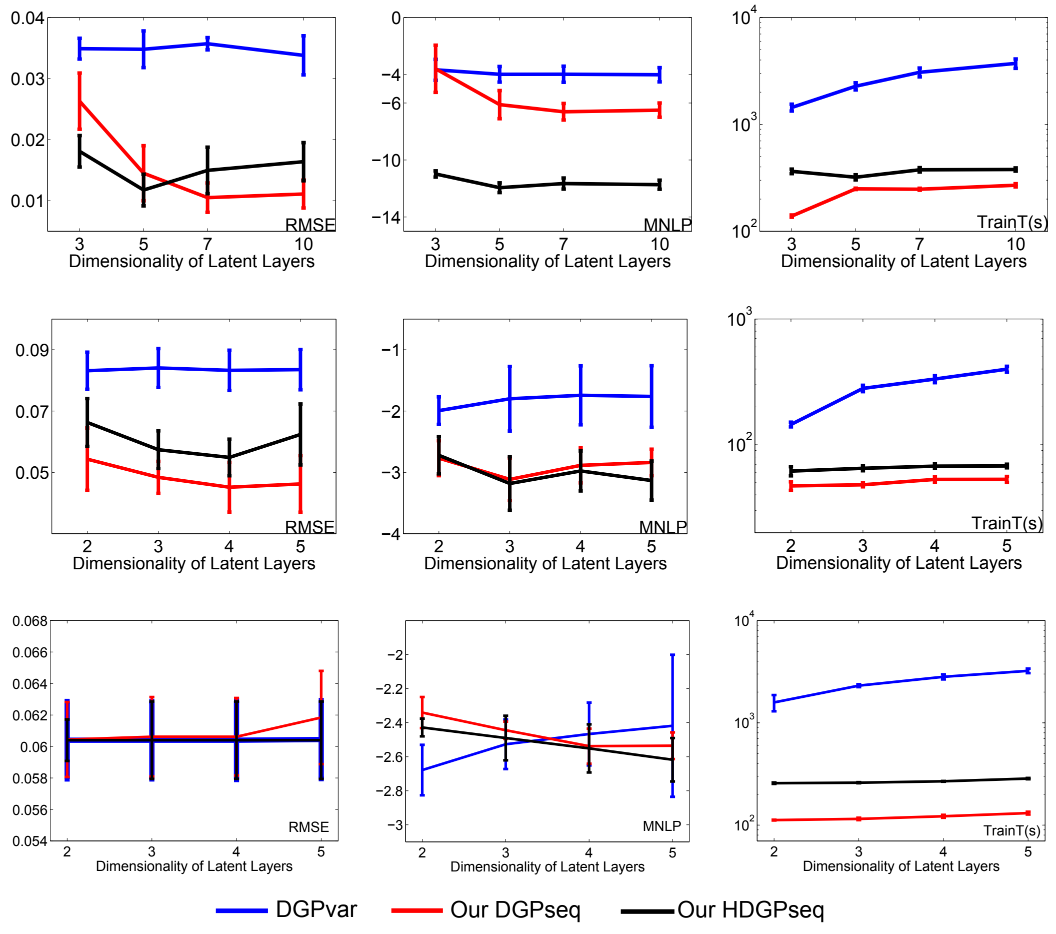


Figure 2: DGP as Deep Dynamical Prior (Robustness to Complexity of the Deep Model): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the dimensionality of latent layers D. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation.

sharing a number of latent layers.

Qualitative Results: **(i)** For the *motion* dataset, we show that our particles are effective and efficient to estimate $\mathbf{y}$ in Fig. 5. The settings of our DGP$_{seq}$ in this figure are the number of latent layers $L = 2$, the dimensionality of latent layers $D = 10$, the number of particles $N_p = 100$, the size of active set $N_{AC} = 100$. **(ii)** For the *flu* dataset, we estimate its posterior using our HDGP$_{seq}$ with the number of latent layers $L = 2$, the dimensionality of latent layers $D = 5$, the number of particles $N_p = 100$, the size of active set $N_{AC} = 100$. As shown in Fig. 6, our HDGP$_{seq}$ successfully captures non-stationarity and heteroscedasticity of flu-activity-rate observations. **(iii)** For the *stock* dataset, we estimate its posterior using our HDGP$_{seq}$ with the number of latent layers $L = 2$, the dimensionality of latent layers $D = 10$, the number of particles $N_p = 100$, the size of active set $N_{AC} = 200$. As shown in Fig. 7, our HDGP$_{seq}$ successfully captures the strong heteroscedasticity of this data.

2.1.2 Dimensionality Reduction for Image Reconstruction

Low-dimensional latent representations provide a means to avoid the so-called "curse of dimensionality" and the low-dimensional latent layers of a DGP can be understood as a form of dimensionality reduction. Inspired by [5], we use DGP for dimensionality reduction to reconstruct images from noisy observations. Towards this goal we use the *Frey* face dataset[4] which is composed of 1900 images of size $20 \times 28 = 560$. We add Gaussian noise (std dev 0.1) to the images, and use these noisy images as both the input and output of the DGP model.

Figure 8 shows the RMSE between the noiseless image and the reconstruction as a function of the number of training episodes in our learning framework. Hyperparameter learning was performed after the first episode and the hyperparameters were fixed for the remaining

[4]http://www.cs.nyu.edu/∼roweis/data.html

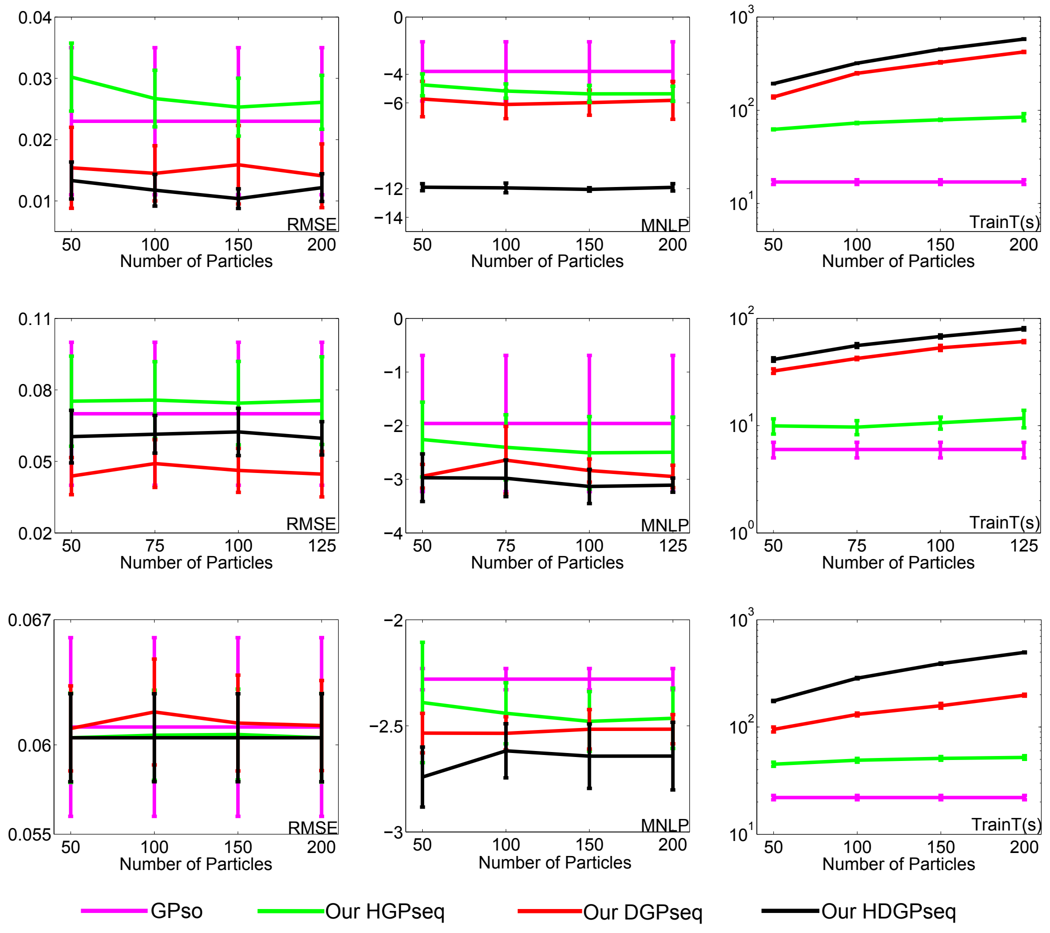


Figure 3: DGP as Deep Dynamical Prior (Properties of Our Inference Framework): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the number of particles N_p. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation. Note that there is no N_p in GP$_{so}$, hence GP$_{so}$ is a straight line for the plot of N_p.

episodes. We chose the number of particles N_p and the size of active set N_{AC} to be 100 in our DGP$_{seq}$. To reduce the randomness of sampling, we run our DGP$_{seq}$ five times for each episode and report the average reconstruction RMSE. As shown in Fig. 8, our DGP$_{seq}$ outperforms DGP$_{var}$. Furthermore, as expected, our DGP$_{seq}$ performs better when the number of training episodes increases, or the width and the depth of the deep structure also increases. Finally, Fig. 9 shows qualitatively that our DGP$_{seq}$ achieves better reconstruction than DGP$_{var}$.

2.2 DGP for Supervised Learning

In this section, we focus our evaluation in the supervised setting for large training sets and/or high-dimensional data sets.

2.2.1 Regression

We employ the *Parkinsons* telemonitoring dataset[5], which is a six-month biomedical voice recording from 42 people with early-stage Parkinson's disease for a total of 5875 input/output pairs. The input is a 16-dimensional biomedical voice feature vector and the output is a 2-dimensional score vector (motor UPDRS score and total UPDRS score). We randomly partition the data five times and choose each time training/test sets to be of size 5000/875. The basic setting of our DGP$_{seq}$ is $L = 2$, $D = 2$, $N_p = 500$, $N_{AC} = 100$. We run our DGP$_{seq}$ for one training episode with sequential inference and hyperparameter learning. We employ the same parameters of our DGP$_{seq}$ to GP$_{so}$ and DGP$_{var}$ whenever applicable. As shown in Table 2, our DGP$_{seq}$ outperforms GP$_{so}$ and DGP$_{var}$. Furthermore, compared to DGP$_{var}$, our DGP$_{seq}$ significantly reduces the computation.

[5]http://archive.ics.uci.edu/ml/index.html

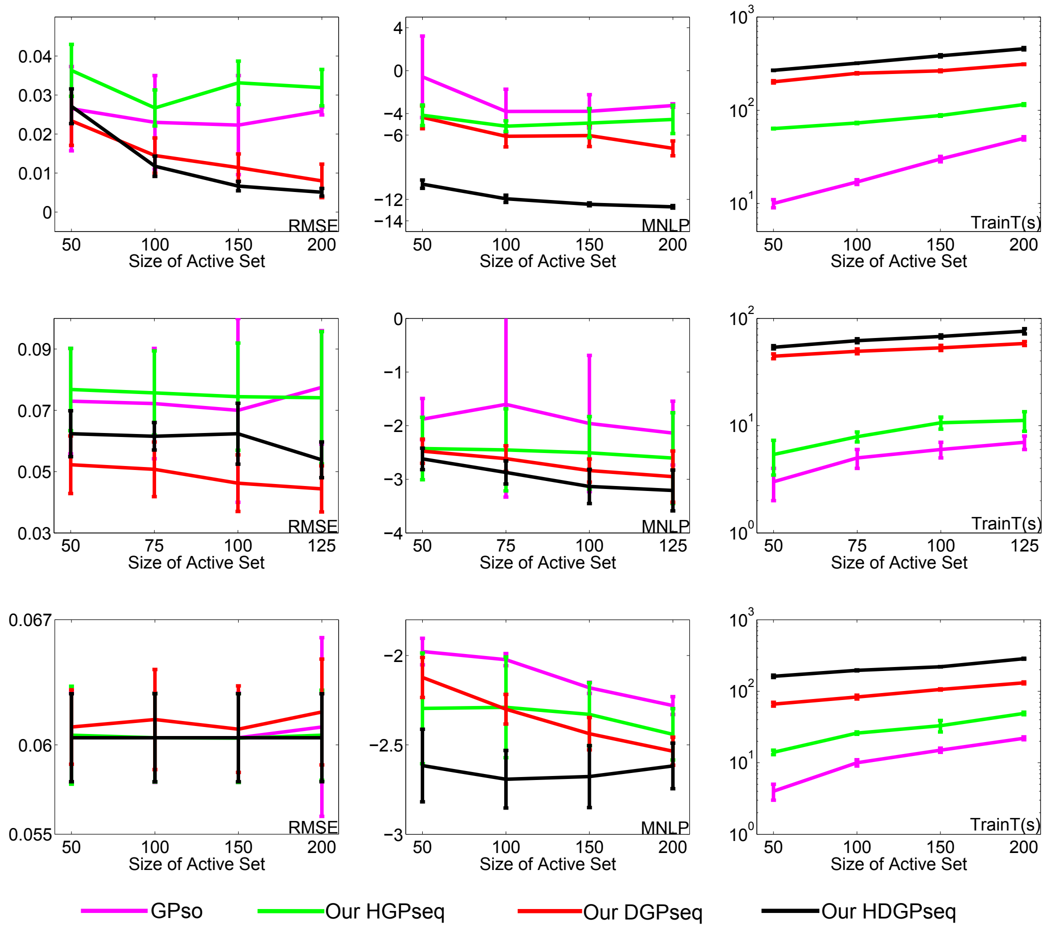


Figure 4: DGP as Deep Dynamical Prior (Properties of Our Inference Framework): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the size of active set N_{AC}. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation.

2.2.2 Classification

We evaluate our approach for classification on the *Tumor* gene expression dataset[6] and the *MNIST* Digits dataset[7]. The tumor data consists of a predefined training and test sets of sizes 144 and 46 respectively. The input contains 16,063 tumor genes (16,063 dimensional input vector) and the output contains 14 cancer classes (14 dimensional binary output vector). For the tumor data, the basic settings of our DGP$_{seq}$ are $L = 3$, $D = 12$, $N_p = 200$. N_{AC} is the size of the data as we did not employ sparsification. We use a linear kernel to reduce computation as the data is very high-dimensional. We run our DGP$_{seq}$ for five training episodes with sequential inference and hyperparameter learning. We employ the same parameters for our DGP$_{seq}$, GP$_{so}$ [1] and DGP$_{var}$ [3] whenever applicable. As shown in Table 3, our DGP$_{seq}$ achieves the best classification accuracy and also significantly outperforms DGP$_{var}$ [3] in terms of computation.

For the MNIST data, there are 60,000 training images and 10,000 test images. The input is a 28×28 image (784 dimensional input vector) and the output contains 0-9 digits (10 dimensional binary output vector). Classification accuracy on the test set (i.e., the percentage of correctly classified examples) is evaluation metric. The basic settings of our DGP$_{seq}$ are $L = 1$, $D = 400$, $N_p = 100$ and $N_{AC} = 1000$. The covariance function of GP is chosen as the 1st-order arc-cosine kernel [6]. Additionally, instead of using the latent state h as the output of each layer, we use the logistic activation function of h, i.e., $(1 + e^{-h})^{-1}$ as the output of each layer, which mimics neural networks for classification. Note that this modification is simple to accommodate in our inference algorithm by simply propagating the particles through the activation function when sampling. We run our DGP$_{seq}$ for one training episode. We employ the same parameters for GP$_{so}$ [1] whenever applicable. As this data

[6]http://www.genome.wi.mit.edu/
[7]http://yann.lecun.com/exdb/mnist/

Data	Methods	RMSE(%)	MNLP	Train_T(s)
Motion	GP$_{so}$	1.45	-4.81	20
	our DGP$_{seq}$	1.0±0.1	-4.95±0.75	202±2
	our HDGP$_{seq}$	1.1±0.3	-5.15±0.20	236±8
Flu	GP$_{so}$	5.3	-2.42	8
	our DGP$_{seq}$	4.0±0.6	-3.09±0.32	50±2
	our HDGP$_{seq}$	3.0±0.7	-3.16±0.11	108±6
Stock	GP$_{so}$	6.4	-1.95	28
	our DGP$_{seq}$	6.4±0.1	-2.18±0.07	305±2
	our HDGP$_{seq}$	6.6±0.3	-2.41±0.04	413±4

Table 1: Missing Data in Multi-Task Learning (Motion/Flu/Stock).

Data	Methods	Acc	Train_T(s)
Tumor	GP$_{so}$	0.65	1
	DGP$_{var}$	0.50	2256
	our DGP$_{seq}$	**0.73**	53
MNIST	GP$_{so}$	0.9500	60mins
	DGP$_{var}$	-	-
	DVI-GPLVM	0.9405	20mins*
	our DGP$_{seq}$	0.9424	180mins

Table 3: Classification. For MNIST, the results of DVI-GPLVM are from [7]. (*) Distributed implementation using an unreported number of cores.

set is large-size, DGP$_{var}$ [3] is infeasible. Instead, we compare our DGP$_{seq}$ to the state-of-the-art scalable GPLVM with distributed variational Inference (DVI-GPLVM) [7]. As shown in Table 3, our DGP$_{seq}$ outperforms DVI-GPLVM in terms of accuracy with a comparable amount of training time (considering that their reported timing was parallelized while our implementation is currently serial). GP$_{so}$ outperforms both DVI-GPLVM and our DGP$_{seq}$ on this dataset, however as noted in [7], the primary purpose here is to demonstrate the scalability of the approach. These results demonstrate the ability of our approach to scale to both large and high dimensional datasets.

2.3 Further Comparison for Extensions

The heteroscedastic GPs and multi-task GPs have been well studied in the GP community [8, 9, 10, 11, 12]. Hence, we use the benchmark heteroscedastic *motorcycle* dataset [13] and the multi-output *Jura* dataset [12] in order to compare with state-of-the-art approaches in these two domains.

2.3.1 Comparison with State-of-the-art Heteroscedastic GP Approaches

We use a benchmark motorcycle dataset [13] to evaluate heteroscedasticity. This data consists of 94 time/acceleration pairs, where there are three noise regions including a flat low noise region, a curved region and a flat high noise region [14].

Methods	RMSE	MNLP	Train_T(s)
GP	22.81±3.96	9.22±0.34	0.24±0.07
NGP	22.80±3.95	9.22±0.33	0.45±0.09
WGP	22.88±4.18	8.73±0.90	1.79±0.19
MLHGP	22.89±3.60	8.44±1.07	1.03±0.22
VHGP	22.88±3.76	8.45±0.41	1.95±0.28
our HGP$_{seq}$	22.79±3.98	**8.25±0.52**	0.93±0.05

Table 4: Comparison with State-of-the-art Heteroscedastic GP Approaches (Motorcycle Data).

We compare our HGP$_{seq}$ with a number of state-of-the-art heteroscedastic GP approaches. Namely, the standard GP with ARD kernel (GP); the standard GP with the non-stationary kernel (NGP) compounded by ARD kernel, neural network kernel and linear kernel [15]; warped GP (WGP) [16]; most likely heteroscedastic GP (MLHGP) [8]; variational heteroscedastic GP (VHGP) [9].

For each of five date partitions, 60/34 data pairs are used for training/test. For our HGP$_{seq}$, $N_p = 100$ and N_{AC} is the size of training set (as we did not use sparsification). The number of training episodes is two. The average results in Table 4 show that our HGP$_{seq}$ outperforms other GP approaches with a better MNLP and a competitive NMSE & training time. Additionally, we evaluate the posterior of the motorcycle data by using our HGP$_{seq}$ with the same setting before. As shown in Fig. 10, our HGP$_{seq}$ outperforms the standard GP due to the heteroscedastic noise modelling.

2.3.2 Comparison with State-of-the-art Multi-Task GP Approaches

We choose a benchmark Jura data[8] to show our comparison with state-of-the-art multi-task GP approaches. In the *Jura* dataset, the input is a 2D location and the output is the measurement of cadmium,

[8]http://home.comcast.net/ pgoovaerts/book.html

Data	Methods	RMSE	MNLP	Train_T(s)
	GP_{so}	9.33 ±0.15	7.45 ±0.02	37±2
Parkinsons	DGP_{var}	9.26±0.41	8.51±1.35	33486±1370
	our DGP_{seq}	**8.86±0.23**	**7.23±0.08**	546±18

Table 2: DGP as Regressor (Parkinsons Data): Prediction Error & Training Efficiency.

Methods	MAE	Train_T(s)
GP	0.5739±0.0003	74
CMOGP	0.4552±0.0013	784
SLFM	0.4578±0.0025	792
GPRN	0.4040±0.0006	1040
GPRN-NPV1	0.4147±0.0001	130
our DGP_{seq}	0.4150±0.0061	21

Table 5: Comparison with State-of-the-art Multi-task GPs (the Jura dataset). The results of GP, CMOGP, SLFM, GPRN are from [12].

nickel and zinc concentrations. The total number of data pairs is 359, where 100 measurements of cadmium are missing. We follow the experimental settings of [12] and use mean absolute error (MAE) for evaluation, choose the number of latent layers $L = 1$, the dimensionality of latent layers $D = 2$. Both the number of particles and the size of active set in our DGP_{seq} are 200. We run our DGP_{seq} five times for one training episode and compare the results with the standard GP, convolved multiple outputs GP (CMOGP)[11], semiparametric latent factor model (SLFM)[10], GP regression networks (GPRN)[12], and GPRN with nonparametric variational inference (mode 1, GPRN-NPV1) [17]. As shown in Table 5, our DGP_{seq} achieves a competitive MAE but with much less training time indicating that our sequential inference is efficient and able to capture the correlations between multiple outputs.

References

[1] L. Csató, M. Opper, Sparse Online Gaussian Processes, Neural Computation 14 (3) (2002) 641–668.

[2] S. V. Vaerenbergh, M. Lázaro-Gredilla, I. Santamaría, Kernel Recursive Least-Squares Tracker for Time-Varying Regression, IEEE Transactions on Neural Networks and Learning Systems 23 (8) (2012) 1313 – 1326.

[3] A. C. Damianou, N. D. Lawrence, Deep Gaussian Processes, in: AISTATS, 2013.

[4] N. D. Lawrence, A. J. Moore, Hierarchical Gaussian Process Latent Variable Models, in: ICML, 2007.

[5] J. Hensman, N. Lawrence, Nested Varitaional Compression in Deep Gaussian Processes, in: http://arxiv.org/pdf/1412.1370.pdf, 2014.

[6] Y. Cho, L. K. Saul, Kernel Methods for Deep Learning, in: NIPS, 2009.

[7] Y. Gal, M. van der Wilk, C. E. Rasmussen, Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models, in: NIPS, 2014.

[8] K. Kersting, C. Plagemann, P. Pfaff, W. Burgard, Most Likely Heteroscedastic Gaussian Process Regression, in: ICML, 2007.

[9] M. Lázaro-Gredilla, M. K. Titsias, Variational Heteroscedastic Gaussian Process Regression, in: ICML, 2011.

[10] Y. W. Teh, M. Seeger, M. I. Jordan, Semiparametric Latent Factor Models, in: AISTATS, 2005.

[11] M. A. Alvarez, N. Lawrence, Computationally efficient convolved multiple output gaussian processes, JMLR 12 (2011) 1459–1500.

[12] A. G. Wilson, D. A. Knowles, Z. Ghahramani, Gaussian Process Regression Networks, in: ICML, 2012.

[13] B. W. Silverman, Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting, Journal of the Royal Statistical Society. Series B (Methodological) 47 (1) (1985) 1–52.

[14] C. E. Rasmussen, Z. Ghahramani, Infinite Mixture of Gaussian Process Experts, in: NIPS, 2002.

[15] C. E. Rasmussen, C. K. I. Williams, Gaussian Process for Machine learning, MIT Press, 2006.

[16] E. Snelson, C. E. Rasmussen, Z. Ghahramani, Warped Gaussian Processes, in: NIPS, 2004.

[17] T. V. Nguyen, E. V. Bonilla, Efficient Variational Inference for Gaussian Process Regression Networks, in: AISTATS, 2013.

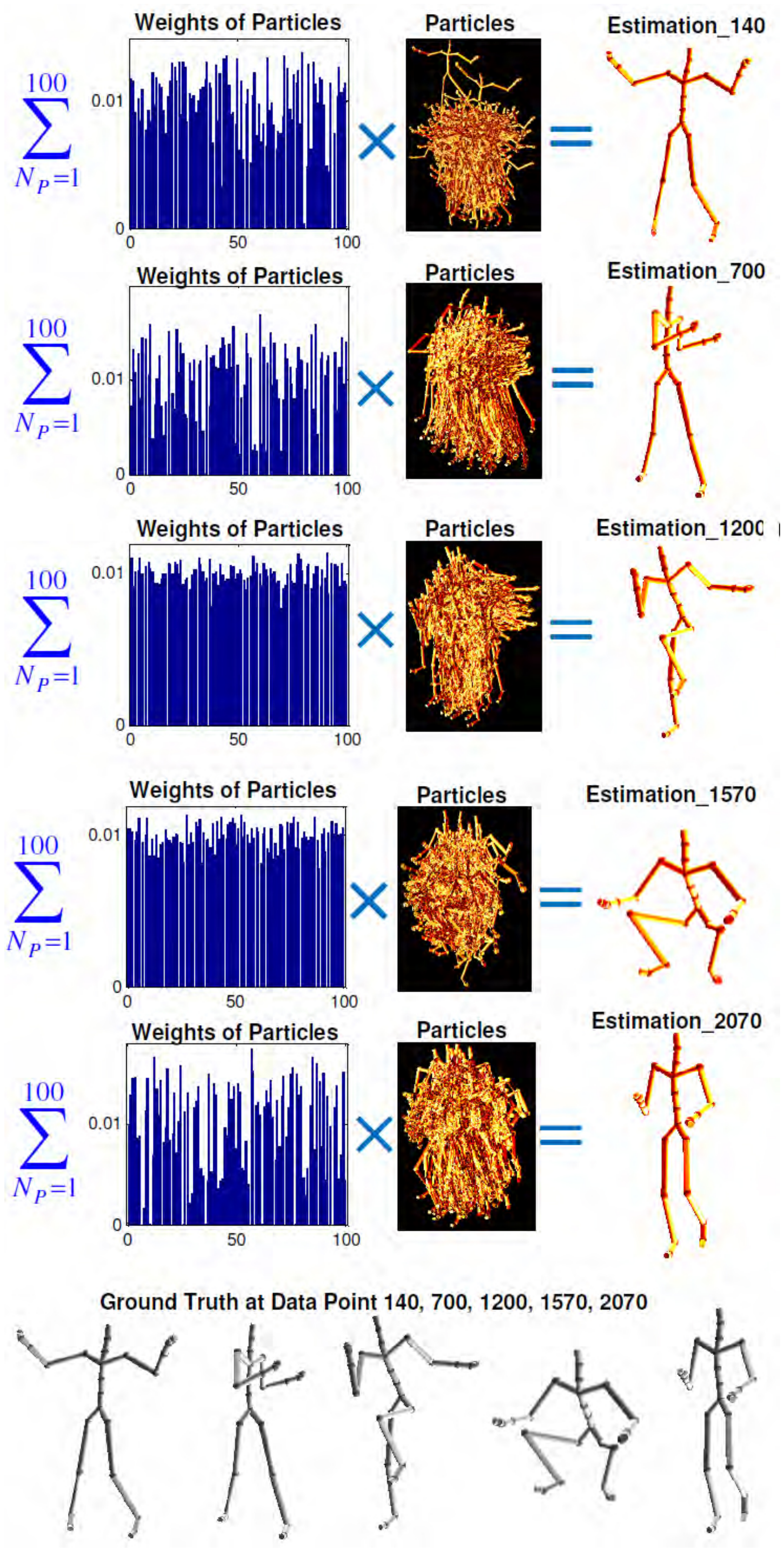


Figure 5: DGP as Deep Dynamical Prior (the motion dataset). Particle Estimation of $\mathbf{y}$ at the time steps 140, 700, 1200, 1570, 2070 where the number of particles N_p is 100.

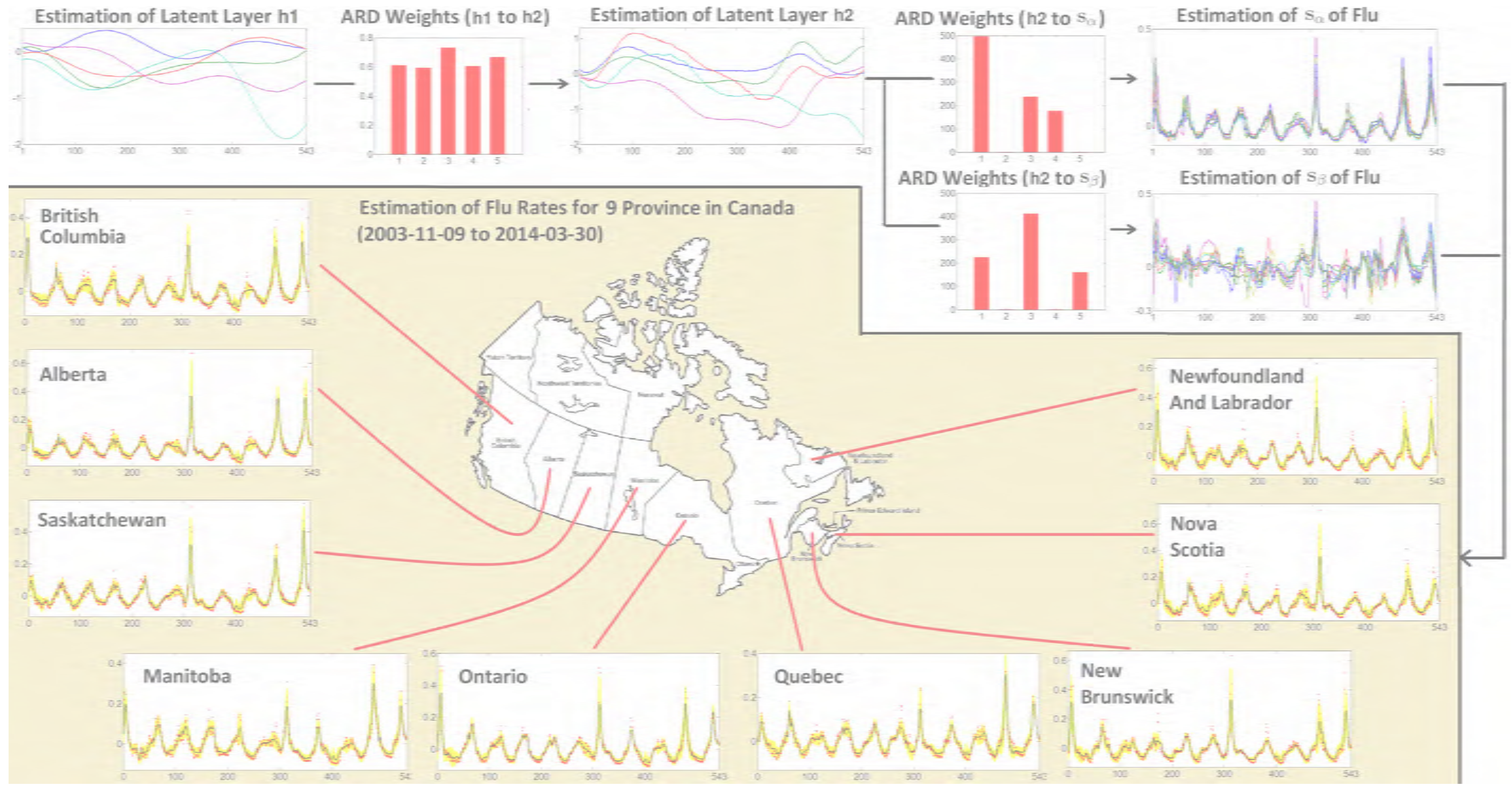


Figure 6: DGP as Deep Dynamical Prior (the flu dataset). Our HDGP$_{seq}$ has 2 latent layers, the dimensionality of each layer is 5. In the plots for the estimation of flu rates (9 provinces in Canada), the red dots are the observations. The black lines are the estimated means. The yellow regions are the 95% confidence interval.

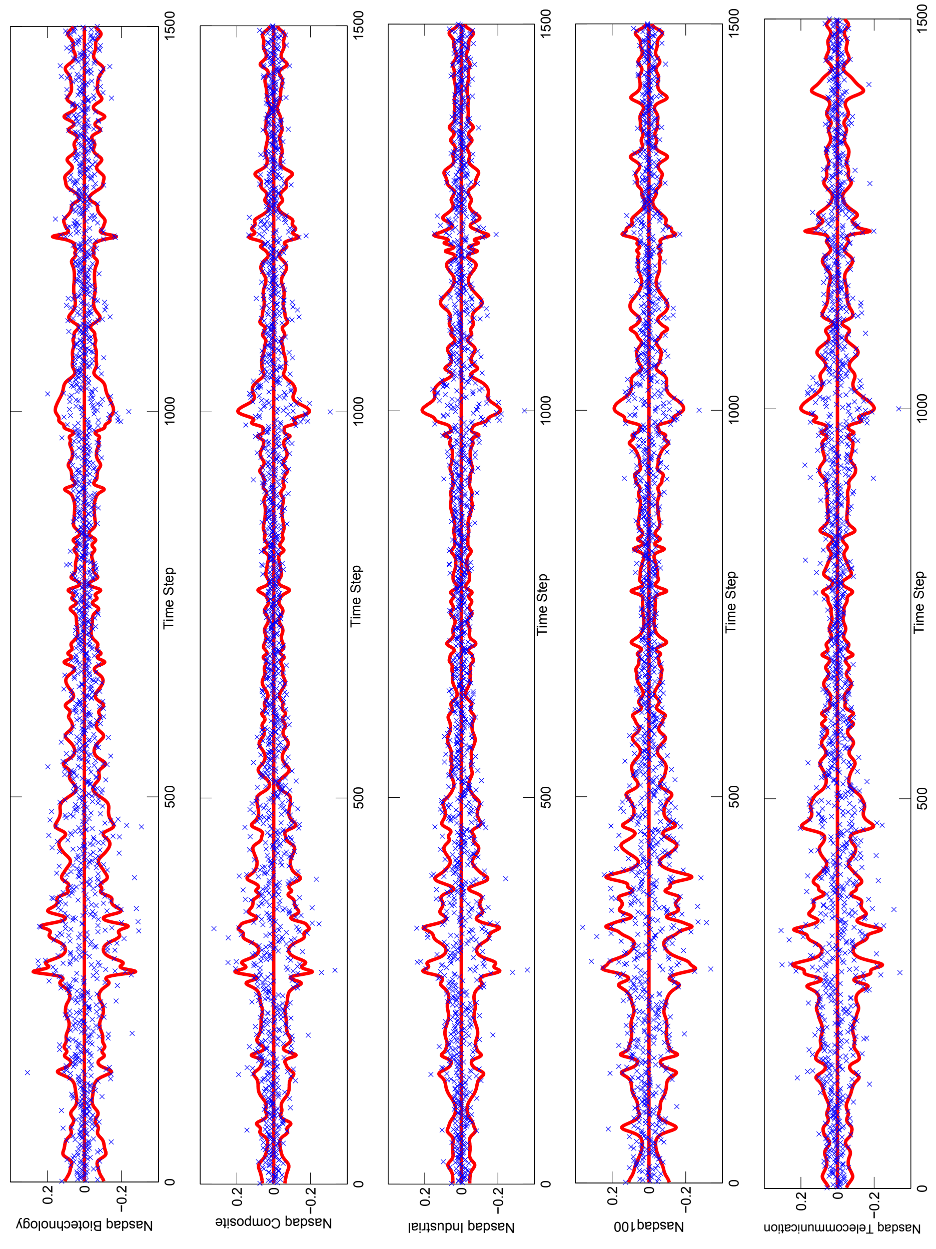


Figure 7: DGP as Deep Dynamical Prior (the stock dataset). The observations are blue crosses, the posterior mean with 95% confidence interval of our HDGP$_{seq}$ are red lines.

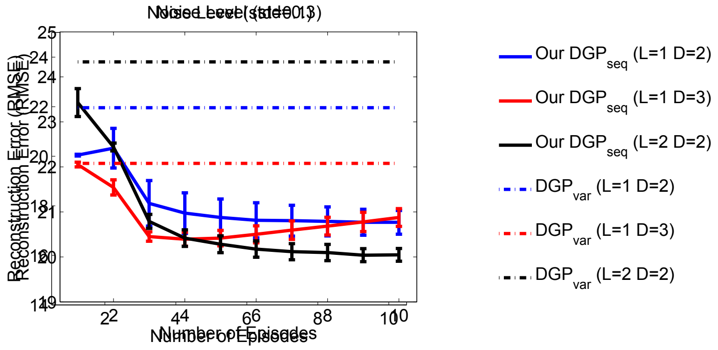


Figure 8: Dimensionality Reduction for Image Reconstruction (Face Data): the reconstruction error as a function of the number of episodes. Note that there is no sequential inference & hyperparameter learning episode in DGP$_{var}$. The results of DGP$_{var}$ are lines. The error bar is mean$\pm$standard deviation.

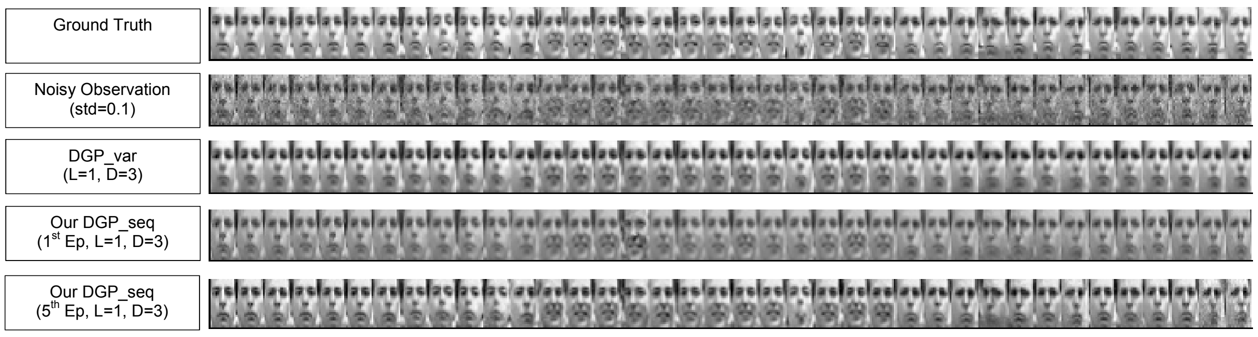

Ground Truth
Noisy Observation (std=0.1)
DGP_var (L=1, D=3)
Our DGP_seq (1st Ep, L=1, D=3)
Our DGP_seq (5th Ep, L=1, D=3)

Figure 9: Dimensionality Reduction for Image Reconstruction (Face Data): the reconstruction image comparison at the time step 50:50:1900.

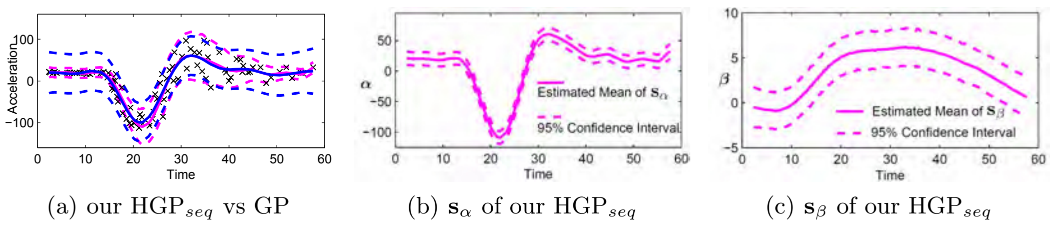


Figure 10: Heteroscedastic GP Modeling (Posterior of Motorcycle Data). In Plot10(a), the observations are black crosses, the mean of our HGP$_{seq}$ / GP are pink / blue lines, 95% confidence interval of our HGP$_{seq}$ / GP are pink / blue dashed lines.