Deep Functional Factor Models: Forecasting High-Dimensional Functional Time Series via Bayesian Nonparametric Factorization

Yirui Liu [1 2] **Xinghao Qiao** [3] **Yulong Pei** [1] **Liying Wang** [4]

Abstract

This paper introduces the Deep Functional Factor Model (DF^{2}M), a Bayesian nonparametric model designed for analysis of high-dimensional functional time series. DF^{2}M is built upon the Indian Buffet Process and the multi-task Gaussian Process, incorporating a deep kernel function that captures non-Markovian and nonlinear temporal dynamics. Unlike many black-box deep learning models, DF^{2}M offers an explainable approach to utilizing neural networks by constructing a factor model and integrating deep neural networks within the kernel function. Additionally, we develop a computationally efficient variational inference algorithm to infer DF^{2}M. Empirical results from four real-world datasets demonstrate that DF^{2}M provides better explainability and superior predictive accuracy compared to conventional deep learning models for high-dimensional functional time series.

1. Introduction

Functional time series refers to a sequential collection of functional objects that exhibit temporal dependence, and this area of study has garnered increasing attention in recent years. With the advancements in data collection technology and computational power, high-dimensional datasets containing numerous functional time series have become more prevalent. Examples of such data include annual age-specific mortality rates for different countries, daily energy consumption curves from various households, and cumulative intraday return trajectories for hundreds of stocks. These datasets can be represented as p-dimensional functional time series $\boldsymbol{Y}_t(\cdot) = \big(Y_{t1}(\cdot), \ldots, Y_{tp}(\cdot)\big)^T$, where each $Y_{tj}(\cdot)$ is a random function defined on a compact interval $\mathcal{U}$. The number of functional variables p is comparable to, or even larger than, the number of temporally dependent observations n. Analyzing high-dimensional functional time series presents a challenging task, as it necessitates the use of dimension reduction techniques to address the high-dimensional problem, functional approaches to handle the infinite-dimensional nature of curve data, and time series modeling methods to capture the temporal dependence.

Several statistical methods have been proposed to address these challenges, such as those presented in Gao et al. (2019); Chen et al. (2022); Fang et al. (2022); Chang et al. (2023a); Guo & Qiao (2023); Zhou & Dette (2023). However, these approaches often assume the existence of linear and Markovian dynamics over time, which may fail to accurately capture the complex nonlinear or non-Markovian temporal dependence that often arises in real-world scenarios.

On the other hand, while deep learning has achieved impressive results in computer vision and natural language processing (NLP) (Guo et al., 2016; He et al., 2016; Vaswani et al., 2017; Torfi et al., 2020), applying deep neural networks directly to handle high-dimensional functional time series is challenging. One major issue when dealing with time series data is that deep learning is a general black-box method that lacks explainability, thus making it difficult to understand the cross-sectionally and serially correlated relationships. However, explainability is crucial in many applications. For instance, in finance, healthcare, and climate change, the accuracy and reliability of a model's predictions have significant impacts on business decisions, patient outcomes, or environmental safety, respectively. Additionally, the non-stationarity of data and the large number of parameters in deep neural networks pose extra challenges during training.

In this paper, we propose an explainable approach called the deep functional factor model (DF^{2}M), which has the ability to discover nonlinear and non-Markovian dynamics in high-dimensional functional time series. Developed as a Bayesian nonparametric model, DF^{2}M employs a functional version of a factor model for dimension reduction, incorporates an Indian buffet process prior in the infinite-dimensional loading matrix to encourage column sparsity (Guo et al.,

[1]J.P. Morgan [2]London School of Economics and Political Science [3]Faculty of Business and Economics, The University of Hong Kong [4]Management School, University of Liverpool. Correspondence to: Xinghao Qiao <xinghaoq@hku.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

2021), utilizes a functional version of a Gaussian process dynamical model to capture temporal dependence within latent functional factors, and employs deep neural networks to construct the temporal kernel.

DF^{2}M offers several advantages for the analysis of high-dimensional functional time series. (i) Firstly, it facilitates a more intuitive understanding of the underlying data structure by representing observed curve variables using a smaller set of latent functional factors. This enhances model explainability and provides a clear and interpretable mapping of relationships between variables, which is crucial for decision-making and subsequent analysis. (ii) Secondly, DF^{2}M is capable of discovering non-Markovian and nonlinear temporal dependence in the functional latent factor space. This enables more accurate predictions of future values. (iii) Lastly, DF^{2}M offers a flexible framework that combines modern sequential deep neural networks with a backbone Bayesian model. This allows for the utilization of sequential deep learning techniques such as gated recurrent unit (GRU) (Cho et al., 2014), long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), and attention mechanisms (Vaswani et al., 2017).

2. Preliminaries

2.1. Indian Buffet Process

The Indian buffet process (IBP) (Griffiths & Ghahramani, 2011) is a probability distribution over a sparse binary matrix with a finite number of rows and an infinite number of columns. The matrix $\boldsymbol{Z}$, generated from the IBP with parameter α, is denoted as $\boldsymbol{Z} \sim \text{IBP}(\alpha)$, where α controls the column sparsity of $\boldsymbol{Z}$.

IBP can be explained using a metaphor that customers sequentially visit a buffet and choose dishes. The first customer samples a number of dishes based on Poisson(α). Subsequent the i-th consumer, in turn, samples each previously selected dish with a probability proportional to its popularity (m_k/i for dish k), and also tries new dishes following Poisson(α/i).

It is worth noting that the distribution remains exchangeable with respect to the customers, meaning that the distribution is invariant to the permutations of the customers. The Indian buffet process admits a stick-breaking representation as $v_j \mid \alpha \sim \text{Beta}(\alpha, 1)$ independently for $j = 1, 2, \ldots,\ w_k = \prod_{j=1}^{k} v_j$ for $k = 1, 2, \ldots$, and $Z_{ik} \mid w_k \sim \text{Bernoulli}(w_k)$ independently for $i = 1, \ldots, n$, and the IBP is then defined as $\boldsymbol{Z} = (Z_{ik})_{1 \le i \le n, k \ge 1}$. The stick-breaking representation is frequently used in the inference for IBP.

2.2. Gaussian Process

A Gaussian process $X(\cdot)$, defined on a compact interval $\mathcal{U}$, is a continuous stochastic process characterized by the fact that every finite collection of its values, $X(u_1), \ldots, X(u_L)$ with $u_1, \ldots, u_L \in \mathcal{U}$, belongs to an L-dimensional multivariate Gaussian distribution (Williams & Rasmussen, 2006). This means that a Gaussian process is completely determined by its mean function $m(u) = \text{E}\big[X(u)\big]$ and its covariance function $\kappa(u, v) = \text{Cov}\big(X(u), X(v)\big) = \text{E}\Big[\big(X(u) - m(u)\big)\big(X(v) - m(v)\big)\Big]$ for any $u, v \in \mathcal{U}$.

The covariance function, also known as the kernel function in machine learning literature, specifies the correlation between values at distinct points. Examples include the squared exponential kernel $\kappa(u, v) = \exp\big(-|u - v|^2/\ell^2\big)$ and the Ornstein–Uhlenbeck kernel $\kappa(u, v) = \exp\big(-|u - v|/\ell\big)$, where ℓ is the length-scale parameter. Additionally, the kernel function can be made more complex using the kernel trick (Hofmann et al., 2008) by rewriting it as $\kappa(u, v) = \big\langle \phi(u), \phi(v) \big\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product, and the feature function $\phi(\cdot)$ maps x into a feature space. As $\phi(\cdot)$ can be an arbitrary function (linear or nonlinear), the Gaussian process offers considerable flexibility in modeling complex patterns in the data.

Furthermore, a multi-task Gaussian process (MTGP) (Bonilla et al., 2007) can be employed to model vector-valued random fields. It is defined as $\mathbf{X}(\cdot) = \big(X_1(\cdot), \ldots, X_M(\cdot)\big)^T$, where $X_1(\cdot), \ldots, X_M(\cdot)$ are M Gaussian processes defined on $\mathcal{U}$. The covariance function between the l-th and k-th task is given by $\text{Cov}\big(X_l(u), X_k(v)\big) = \Sigma_{lk}\kappa(u, v)$, where $\boldsymbol{\Sigma} = \{\Sigma_{lk}\}_{1 \le l, k \le M}$ is a positive semi-definite matrix encoding the similarities between pairs of tasks. The MTGP can effectively capture inter-task correlations and improve predictions (Moreno-Muñoz et al., 2018).

2.3. Sequential Deep Learning

Deep learning methods, widely used in computer vision, NLP, and reinforcement learning, have become increasingly popular for time series prediction as well (Lim & Zohren, 2021). In particular, recurrent neural networks (RNN) and attention mechanisms, commonly used for sequence prediction tasks in NLP, can be adapted for temporal forecasting tasks in time series data. A multivariate time series can be modeled recursively in RNN as $\mathbf{x}_t = g_{\text{dec}}(\mathbf{h}_t)$ and $\mathbf{h}_t = g_{\text{enc}}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1})$, where $\mathbf{h}_t$ is a latent variable and g_{dec} and g_{enc} are the decoder and encoder functions, respectively. Two renowned RNN models, LSTM and GRU, are designed to learn long-range dependencies in a sequence. For simplicity, we denote their encoder functions as $\mathbf{h}_t = \text{LSTM}(\mathbf{x}_{1:t})$ and $\mathbf{h}_t = \text{GRU}(\mathbf{x}_{1:t})$, respectively, where $\mathbf{x}_{1:t} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$. Moreover, attention mecha-

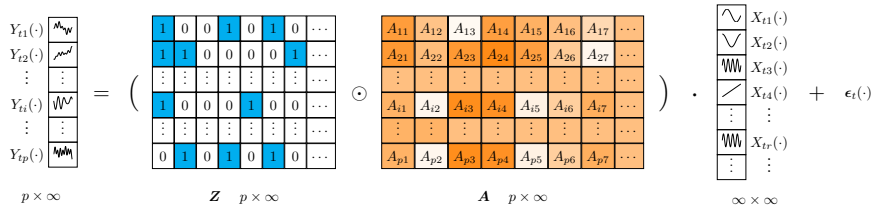


Figure 1. Sparse functional factor model. The blue (or white) cells in $\boldsymbol{Z}$ indicate 1 (or 0), while the darker (or lighter) shades of orange in $\boldsymbol{A}$ represent larger (or smaller) values.

nisms, which have achieved state-of-the-art performance in NLP tasks, can also be utilized to model time series data. Unlike RNNs, attention mechanisms directly aggregate information from multiple time steps in the past. Attention mechanisms can be expressed as $\mathbf{h}_t = \sum_{i=1}^{t-1} \omega(\mathbf{k}_t, \mathbf{q}_\tau)\mathbf{v}_{t-\tau}$, where key $\mathbf{k}_t$, query $\mathbf{q}_t$, and value $\mathbf{v}_t$ are intermediate representations generated by linear or nonlinear transformations of $\mathbf{x}_t$. We denote such attention mechanisms as $\mathbf{h}_t = \text{ATTN}(\mathbf{x}_{1:t})$. See detailed structures for both RNNs and attention mechanisms in Appendix A.

3. Deep Functional Factor model

3.1. Sparse Functional Factor Model

First, we propose a functional factor model from the Bayesian perspective,

$$\boldsymbol{Y}_t(\cdot) = (\boldsymbol{Z} \odot \boldsymbol{A})\boldsymbol{X}_t(\cdot) + \boldsymbol{\epsilon}_t(\cdot), \quad t = 1, \ldots, n. \tag{1}$$

The observed functional time series is denoted as $\boldsymbol{Y}_t(\cdot)$, and the binary matrix $\boldsymbol{Z}$ is sampled from the Indian buffet process, $\boldsymbol{Z} \sim \text{IBP}(\alpha)$. The Hadamard (elementwise) product is represented by $\odot$. The loading weight matrix $\boldsymbol{A}$ has elements $A_{tr} \sim \text{Normal}(0, \sigma_A^2)$ for any $r \in \mathbb{N}^+$ independently. The latent functional factor time series are denoted as $\boldsymbol{X}_t(\cdot) = \big(X_{t1}(\cdot), X_{t2}(\cdot), \ldots, X_{tr}(\cdot), \ldots\big)^T$, and the idiosyncratic component is denoted as $\boldsymbol{\epsilon}_t(\cdot)$, which follows a Gaussian distributed white noise process on a scale σ_ϵ.

In this framework, we do not specify the number of latent factors. Instead, $\boldsymbol{Y}_t(\cdot)$, $\boldsymbol{Z}$, and $\boldsymbol{A}$ can be regarded as $p \times \infty$ matrices, and $\boldsymbol{X}_t(\cdot)$ as an infinite-dimensional vector of functions, or heuristically, a $\infty \times \infty$ matrix. The dimension reduction framework in equation (1) is illustrated in Figure 1. This Bayesian nonparametric factor model allows for a potentially unlimited number of latent factors, eliminating the need to specify a fixed dimensionality of the factor space. The nonparametric approach introduces flexibility and provides a foundation for inferring the number of factors in the posterior distribution using nonparametric inference frameworks such as Gibbs sampling (Teh et al., 2006), merge-split algorithm (Hughes & Sudderth, 2013), and conditional and adaptively truncated variational inference (Liu et al., 2022, 2023).

Additionally, the Indian buffet process can provide column sparsity (Vu & Lei, 2013) to $\boldsymbol{Z}$ and, therefore, to the loading matrix $\boldsymbol{Z} \odot \boldsymbol{A}$. This implies that most elements in each row are zeros, as w_k in the stick-breaking representation of the IBP approaches zero as k increases. In the factor model, this column sparsity implies that each factor affects only a small fraction of functional variables (Guo et al., 2021), or equivalently, the factors are related to each other through a hierarchy (Griffiths & Ghahramani, 2011).

3.2. Functional Gaussian Process Dynamical Model

Secondly, by projecting high-dimensional observations $\boldsymbol{Y}_t(\cdot)$ onto low-dimensional latent functional factors $\boldsymbol{X}_t(\cdot)$, we can capture the sequential structure of the time series model through the factors. To model the temporal dependence of $\boldsymbol{X}_t(\cdot)$, we adopt a Gaussian process over time to encode historical information. In particular, we design the covariance across factors r and l as follows. Let $\mathcal{X}_t$ represent the historical information up to time t, and let $\mathcal{X}$ be the space containing $\{\mathcal{X}_t\}_{t\in\mathbb{N}}$. For any $u, v \in \mathcal{U}$,

$$\text{Cov}\big(X_{tr}(u), X_{sl}(v)\big) = \kappa_{\mathcal{X}}(\mathcal{X}_{t-1}, \mathcal{X}_{s-1})\kappa_{\mathcal{U}}(u, v)\mathbb{I}(r = l), \tag{2}$$

where t and s indicate two time stamps, and $\kappa_{\mathcal{X}}$ and $\kappa_{\mathcal{U}}$ are the kernels defined on $\mathcal{X}$ and $\mathcal{U}$, respectively. The indicator function $\mathbb{I}(r = l)$ equals 1 if $r = l$ and 0 otherwise. The kernel $\kappa_{\mathcal{X}}$ captures historical information from different periods and can be regarded as a temporal kernel. Similarly, the kernel $\kappa_{\mathcal{U}}$ can be seen as a spatial kernel.

Therefore, $\boldsymbol{X}_r(\cdot) = \big(X_{1r}(\cdot), \ldots, X_{tr}(\cdot), \ldots, X_{nr}(\cdot)\big)$ belongs to a multi-task Gaussian process (Bonilla et al., 2007) such that for any $u_1, \ldots, u_L \in \mathcal{U}$, $\text{vec}\big(\boldsymbol{X}_r(u_1, \ldots, u_L)\big) \sim \text{Normal}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^u)$, where $\otimes$ denotes the Kronecker product, or equivalently $\boldsymbol{X}_r(u_1, \ldots, u_L)$ follows a matrix normal distribution (Dawid, 1981) with mean $\mathbf{0}$, row covariance matrix $\boldsymbol{\Sigma}_{\mathcal{U}}^u$, and column covariance matrix $\boldsymbol{\Sigma}_{\mathcal{X}}$. Here, $\boldsymbol{X}_r(u_1, \ldots, u_L) = \big[X_{tr}(u_j)\big]_{1\le t\le n, 1\le j\le L}$, $\boldsymbol{\Sigma}_{\mathcal{X}} = \big[\kappa_{\mathcal{X}}(\mathcal{X}_t, \mathcal{X}_s)\big]_{0\le t,s\le n-1}$, and $\boldsymbol{\Sigma}_{\mathcal{U}}^u = \big[\kappa_{\mathcal{U}}(u_i, u_j)\big]_{1\le i,j\le L}$.

In Appendix B, we demonstrate the detailed relationship between the multi-task Gaussian process and the matrix normal distribution. In the literature, the n-task Gaussian

process is often used to refer to multiple outputs generated by the model, each corresponding to a specific timestamp in our time series setting. For ease of expression, we denote the presented multi-task Gaussian process as

$$\boldsymbol{X}_r(\cdot) \sim \text{MTGP}\big(\mathbf{0}, \kappa_{\mathcal{U}}(\cdot,\cdot), \kappa_{\mathcal{X}}(\cdot,\cdot)\big). \tag{3}$$

Importantly, the marginal distributions of latent factors, denoted as $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r, \ldots$, exhibit cross-sectional dependence due to their common temporal kernel $\kappa_{\mathcal{X}}$, which incorporates historical information up to period $t-1$. This shared kernel promotes similarity and dependence across different time periods in the multi-task Gaussian process. However, when conditioned on $\mathcal{X}_{t-1}$ and the corresponding kernel, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r, \ldots$, become conditionally independent and Gaussian distributed. This conditionality arises from our approach, where the predictive temporal kernel exclusively relies on past information $\mathcal{X}_{t-1}$ rather than current data $\mathcal{X}_t$. Consequently, this approach enables forward-looking predictions based solely on historical data.

The proposed model can be seen as a functional variant of the Gaussian process dynamical model (Wang et al., 2005). The connections between the two models can be found in Appendix A. Using the proposed model to capture the temporal dependence of functional time series has several advantages. First, the temporal kernel and spatial kernel can be separated, which allows for a closed form and computational convenience. Additionally, since the temporal kernel considers the entire historical information rather than just the latest state, the model can be non-Markovian. For example, by defining the temporal kernel as $\kappa(\mathcal{X}_{t-1}, \mathcal{X}_{s-1}) = \alpha_1 \int \mathbf{X}_{t-1}(u)^T \mathbf{X}_{s-1}(u) du + \alpha_2 \int \mathbf{X}_{t-2}(u)^T \mathbf{X}_{s-2}(u) du$, features from the last two periods can be incorporated. Furthermore, nonlinearity can be introduced using the kernel trick by setting a nonlinear kernel function. This opens the possibility of constructing deep kernels, which will be discussed in Section 3.3.

3.3. Deep Temporal Kernels

To capture the complex latent temporal structure, we use neural networks to construct the kernel function. However, compared to standard deep kernels (Wilson et al., 2016; Al-Shedivat et al., 2017; Xue et al., 2019; Li et al., 2019; Watson et al., 2021; Fortuin, 2022), two extra steps are needed when applying deep kernels to functional time series.

Firstly, since $\mathbf{X}_t(\cdot)$ is a continuous process on $\mathcal{U}$, a mapping function $F : \mathcal{F} \to \mathbb{R}^d$ is required to map the infinite-dimensional Gaussian processes to d-dimensional vectors. Here, $\mathcal{F}$ represents the space of continuous functions defined on $\mathcal{U}$. Various approaches can be used for this mapping function, including pre-specified basis expansion, data-dependent basis expansion (such as functional principal component analysis and its dynamic variants (Bathia et al., 2010; Hormann et al., 2015)), adaptive functional neural network (Yao et al., 2021), or even a simple specification such as $\boldsymbol{X}_t(u_0, \ldots, u_L)$ with $u_0, \ldots, u_L \in \mathcal{U}$.

Secondly, the d-dimensional vectors are used as inputs for deep neural networks, and the outputs generated by these networks are employed to construct kernel functions. Specifically, the input vector is transformed as

$$\boldsymbol{h}_t = H\big(F(\boldsymbol{X}_{t-1}), F(\boldsymbol{X}_{t-2}), \ldots\big), \tag{4}$$

where $\mathbf{X}_{t-1} = (X_{t-1,1}, \ldots, X_{t-1,r}, \ldots)^T$, F is the mapping function, and H represents a sequential deep learning framework. Various deep neural network architectures can be utilized for this purpose, such as LSTM, GRU, and attention mechanisms, which have shown their effectiveness in modeling complex patterns and dependencies. Since the inputs for the temporal kernels are ordered sequences from $\mathcal{X}_0$ to $\mathcal{X}_{n-1}$, unidirectional deep neural networks should be used instead of bidirectional networks. The transformed representations $\mathbf{h}_t$ and $\mathbf{h}_s$ are then used to construct a kernel

$$\kappa_{\mathcal{X}}(\mathcal{X}_{t-1}, \mathcal{X}_{s-1}) = \kappa(\boldsymbol{h}_t, \boldsymbol{h}_s), \tag{5}$$

where $\kappa(\cdot,\cdot)$ is a suitable kernel function, such as the squared exponential kernel or the Ornstein-Uhlenbeck kernel. It should be noted that the temporal kernel is related to the historical values of all the relevant factors and is shared across factors. This approach builds a time-dependent variant of a deep Gaussian process (Damianou & Lawrence, 2013), where the kernel incorporates historical information.

To address overfitting, spectrum normalization can be applied, which effectively enforces a Lipschitz condition on the neural networks as suggested by Miyato et al. (2018). Notably, this Lipschitz condition on the model's output concerning its input ensures that similar inputs within the Gaussian process exhibit comparable distances in the kernel space. This implies that the inputs for κ can reflect the distance between $\mathcal{X}_{t-1}$ and $\mathcal{X}_{s-1}$.

3.4. The Imperative of Element Integration

In summary, by combining the functional version of the sparse factor model, sequential deep learning kernel, and Gaussian process dynamical model, we define a probabilistic generative model for high-dimensional functional time series named the deep functional factor model (DF^{2}M).

Factorization is essential in this context. Directly feeding the original high-dimensional input into the kernel function is not viable. The issue arises because without the factorization of high-dimensional functional data, the kernel would have to handle extremely high-dimensional inputs using an excessive number of parameters, while in training time steps are limited. This situation is prone to overfitting, making it challenging to achieve accurate estimation. Additionally,

the interpretability of our proposed method relies on these factors.

Applying a deep kernel without utilizing an IBP with a finite number of latent factors poses difficulties. The challenge lies in determining the optimal number of latent factors. If there are too many factors or too few, it can affect the kernel distance. An excess of factors, often redundant and similar, can distort distance measurements, while an inadequate number of factors may miss critical distances, leading to inaccurate model representations.

The sequential deep learning kernel is essential for introducing non-Markovian and non-linear patterns. When combined with IBP and factorization, it effectively models temporal similarities among observations, enabling capture of high-dimensional functional data dynamics.

4. Bayesian Inference for DF^{2}M

4.1. Sparse Variational Inference

We adopt the variational inference framework to infer the proposed DF^{2}M. This algorithm approximates the posterior probability by maximizing the evidence lower bound (ELBO), which is equivalent to minimizing the Kullback–Leibler (KL) divergence between a variational distribution and true posterior distribution (Blei et al., 2017). For DF^{2}M, with mean-field factorization assuming independence among the variational distributions for latent variables, its ELBO can be expressed as

$$\begin{aligned}\text{ELBO} = \mathrm{E}_q \Big[\log p(\boldsymbol{Z} \mid \alpha) p(\boldsymbol{A} \mid \sigma_A) \\ \prod_{t=1}^{n} p\big(\boldsymbol{Y}_t(\cdot) \mid \boldsymbol{X}_t(\cdot), \boldsymbol{Z}, \boldsymbol{A}\big) \prod_{r\geq 1} p\big(\boldsymbol{X}_r(\cdot) \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big) \Big] \\ - \mathrm{E}_q \Big[\log q(\boldsymbol{Z}) q(\boldsymbol{A}) \prod_{r\geq 1} q\big(\boldsymbol{X}_r(\cdot)\big) \Big].\end{aligned} \tag{6}$$

Using the stick-breaking representation of the Indian buffet process as in Section 2.1, we factorize the variational distribution for $\boldsymbol{Z}$ as $q(v_j) = \text{Beta}(v_j; \tau_j^1, \tau_j^0)$ and $q(Z_{tj}) = \text{Bernoulli}(Z_{tj}; m_{tj})$. The corresponding variational distribution for $\boldsymbol{A}$ is factorized as $q(A_{tj}) = \text{Normal}(A_{tj}; \eta_{tj}, \sigma^2_{A,tj})$.

To avoid singular matrix inversions and improve computational efficiency, we propose a sparse variational inference approach for DF^{2}M based on Titsias (2009). Our method introduces a set of inducing variables representing the values of the latent function at a small subset of points in $\mathcal{U}$. Moreover, we adopt the approach of having common locations for the inducing variables across functional factors, as suggested by Hamelijnck et al. (2021). In other words, we utilize the same set of inducing points for all tasks, which can lead to further improvement in computational efficiency. Consequently, the variational distribution for multi-task Gaussian process with inducing variables is defined as,

$$\begin{aligned}q\big(\boldsymbol{X}_r(\cdot)\big) = p\Big(X_{1r}(\cdot), \ldots, X_{nr}(\cdot) \mid X_{1r}(\boldsymbol{v}), \ldots, \\ X_{nr}(\boldsymbol{v}), \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\Big) \prod_{t=1}^{n} q\big(X_{tr}(\boldsymbol{v})\big),\end{aligned} \tag{7}$$

where $\boldsymbol{v} = (v_1, \ldots, v_K)^T$ with $v_1, \ldots, v_K \in \mathcal{U}$ with K being the number of inducing points. The variational distribution for the inducing variables is constructed as $q(X_{tr}(\boldsymbol{v})) = \text{Normal}(\boldsymbol{\mu}_{tr}, \boldsymbol{S}_{tr})$. It is important to note that the conditional prior distribution for $X_r(\cdot)$, , which is the first term on the right-hand side of equation (7), cannot be factorized as $\prod_{t=1}^{n} p\big(X_{tr}(\cdot) \mid X_{tr}(\boldsymbol{v})\big)$ due to their temporal dependence. However, by exploiting the setting of equation (7), the conditional prior distribution appears in both the variational and prior distributions and therefore can be cancelled. In Appendix D.1, we derive that the ELBO in equation (6) can be simplified as

$$\begin{aligned}\text{ELBO} = \sum_{t=1}^{n} \mathrm{E}_q \Big[\log p\big(\boldsymbol{Y}_t(\cdot) \mid \boldsymbol{X}_t(\cdot), \boldsymbol{Z}, \boldsymbol{A}\big) \Big] \\ - \mathrm{KL}\big[q(\boldsymbol{Z}) \parallel p(\boldsymbol{Z} \mid \alpha)\big] - \mathrm{KL}\big[q(\boldsymbol{A}) \parallel p(\boldsymbol{A} \mid \sigma_A)\big] \\ - \sum_{r\geq 1} \mathrm{KL}\Big[q\big(\boldsymbol{X}_r(\boldsymbol{v})\big) \parallel p\big(\boldsymbol{X}_r(\boldsymbol{v}) \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big)\Big],\end{aligned} \tag{8}$$

where $\boldsymbol{X}_r(\boldsymbol{v}) = \big(X_{1r}(\boldsymbol{v}), \ldots, X_{nr}(\boldsymbol{v})\big)$ with $X_{tr}(\boldsymbol{v}) = \big(X_{tr}(v_1), \ldots, X_{tr}(v_K)\big)^T$ for $t = 1, \ldots, n$. Furthermore, using the formula of the KL divergence between two multivariate Gaussian distributions, we derive a closed form of the last term as

$$\begin{aligned}2\mathrm{KL}\Big[q\big(\boldsymbol{X}_r(\boldsymbol{v})\big) \parallel p\big(\boldsymbol{X}_r(\boldsymbol{v}) \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big)\Big] \\ = \text{trace}\Big((\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv-1})\big(\boldsymbol{S}_r + \text{vec}(\boldsymbol{\mu}_r)\text{vec}(\boldsymbol{\mu}_r)^T\big)\Big) \\ + K \log |\boldsymbol{\Sigma}_{\mathcal{X}}| + n \log |\boldsymbol{\Sigma}_{\mathcal{U}}^{vv}| - \sum_{t=1}^{n} \log |\boldsymbol{S}_{tr}| - nK,\end{aligned}$$

where $\boldsymbol{\mu}_r = (\boldsymbol{\mu}_{1r}, \ldots, \boldsymbol{\mu}_{nr})$, $\boldsymbol{S}_r = \text{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr})$, and $\boldsymbol{\Sigma}_{\mathcal{U}}^{vv} = \big[\kappa_{\mathcal{U}}(v_j, v_j)\big]_{1\leq i,j\leq K}$. See Appendix D.2 for the detailed derivation.

4.2. Sampling for Variational Distribution of Factors

To optimize the variational distributions, the automatic differentiation variational inference (ADVI) algorithm (Kucukelbir et al., 2017; Blei et al., 2017; Ranganath et al., 2014) is adopted to maximize the ELBO in equation (8).

To perform ADVI in our model, we need to sample $\boldsymbol{X}_r(\cdot)$ from its variational distribution as specified in equation (7).

However, directly sampling from a $nL \times nL$ matrix is computationally expensive even though this distribution is Gaussian conditional on $\boldsymbol{X}_r(\boldsymbol{v})$. To address this issue and accelerate the computation of the ELBO, we take advantage of the separability of the temporal and spatial kernels as described in Section 3.2, and propose the following method.

For any $\boldsymbol{u} = (u_1, \ldots, u_L)^T$ with $u_1, \ldots, u_L \in \mathcal{U}$ being the observation points in $\mathcal{U}$, we first partition the spatial covariance matrix for $\boldsymbol{X}(\boldsymbol{u}, \boldsymbol{v})$ into a blockwise matrix shown as $\begin{bmatrix} \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} & \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \\ \boldsymbol{\Sigma}_{\mathcal{U}}^{uvT} & \boldsymbol{\Sigma}_{\mathcal{U}}^{vv} \end{bmatrix}$, where $\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} = \left[\kappa_{\mathcal{U}}(u_i, u_j)\right]_{1 \le i,j \le L}$, and $\boldsymbol{\Sigma}_{\mathcal{U}}^{uv} = \left[\kappa_{\mathcal{U}}(u_i, v_j)\right]_{1 \le i \le L, 1 \le j \le K}$.

Theorem 1 (Posterior Mean) *The mean function of the posterior for $X_{tr}(\cdot)$ is solely dependent on the variational mean of $\boldsymbol{X}_{tr}(\boldsymbol{v})$, the inducing variables at time t. That is, for any $\boldsymbol{u}$*

$$\mathrm{E}\left(X_{tr}(\boldsymbol{u})\right) = \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1} \boldsymbol{\mu}_{tr}. \tag{9}$$

It means that for MTGP, the variational mean is independent of the inducing variables at timestamps other than the current one. See also an analogous theorem for Gaussian process regression in Bonilla et al. (2007).

Theorem 2 (Posterior Variance) *The variance function of the posterior for $\boldsymbol{X}_r(\cdot)$ contains two parts. For any $\boldsymbol{u}$,*

$$\begin{aligned} \mathrm{Var}_q\left[\mathrm{vec}\left(\boldsymbol{X}_r(\boldsymbol{u})\right)\right] = & (I \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1}) \,\mathrm{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr}) \\ & + \boldsymbol{\Sigma}_{\mathcal{X}} \otimes (\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1} \boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}). \end{aligned} \tag{10}$$

The first part is solely dependent on the variational variance of $\boldsymbol{X}_{tr}(\boldsymbol{v})$, while the second part is independent of the variational distributions of all inducing variables. In particular, the first part corresponds to a group of independent Gaussian processes such that $\tilde{\boldsymbol{X}}_{tr}^{(1)}(\boldsymbol{u}) \sim \mathrm{Normal}(\boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1} \boldsymbol{\mu}_{tr}, \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1} \boldsymbol{S}_{tr})$ for any $\boldsymbol{u}$. One the other hand, $\tilde{\boldsymbol{X}}_r^{(2)}(\cdot)$ is a zero-mean multi-task Gaussian process, with $\tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u}) \sim \mathrm{MatrixNormal}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1} \boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}, \boldsymbol{\Sigma}_{\mathcal{X}})$ for any $\boldsymbol{u}$. Therefore, based on Theorems 1 and 2, we can decompose $\boldsymbol{X}_r(\cdot) = \tilde{\boldsymbol{X}}_r^{(1)}(\cdot) + \tilde{\boldsymbol{X}}_r^{(2)}(\cdot)$ under the variational distribution. Notably, the sampling of $\tilde{\boldsymbol{X}}_r^{(1)}(\cdot)$ is more efficient as it only depends on inducting variables within the same period.

Theorem 3 (Irrelevance to ELBO) *Conditional on $\boldsymbol{\Sigma}_{\mathcal{X}}$ and $\boldsymbol{\Sigma}_{\mathcal{U}}$, sampling $\boldsymbol{X}_{tr}(\cdot)$ from the distribution of $\tilde{\boldsymbol{X}}_r^{(1)}(\cdot)$ does not change the variational mean. Moreover, the corresponding ELBO of DF^{2}M in equation (8) is only modified by a constant term given by*

$$\frac{1}{2\sigma_\epsilon^2} \|\boldsymbol{Z} \odot \boldsymbol{A}\|_F^2 \,\mathrm{trace}\left[\boldsymbol{\Sigma}_{\mathcal{X}}\right] \mathrm{trace}\left[\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1} \boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}\right], \tag{11}$$

where $\|\mathbf{M}\|_F = \left(\sum_{i,j} M_{ij}^2\right)^{\frac{1}{2}}$ denotes the Frobenius norm of any matrix $\mathbf{M}$.

See Appendices D.3, D.4 and D.5 for the derivations of Theorem 1, 2 and 3, respectively. With the help of these theorems, we can sample $\boldsymbol{X}_{tr}(\cdot)$ from the proxy variational distribution $\mathrm{Normal}(\boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1} \boldsymbol{\mu}_{tr}, \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1} \boldsymbol{S}_{tr})$, which relies solely on the variational distributions at time t. This approach provides a more efficient way of computing the ELBO compared to direct sampling, which requires the complete Cholesky decomposition of the $nL \times nL$ matrix.

4.3. Initialization, Training and Prediction

We use the technique of ADVI to train the variational parameters of the posteriors, by computing the gradient of the ELBO with respect to the parameters. The training process requires iterating through the following steps until the ELBO converges. The steps of Bayesian inference for DF^{2}M are summarized in Algorithm 1 in Appendix E.

First, conditional on $\boldsymbol{\Sigma}_{\mathcal{X}}$, we update the variational distribution parameters $\boldsymbol{\mu}_{tr}$ and $\boldsymbol{S}_{tr}$ for inducing variables $\boldsymbol{X}_{tr}(\boldsymbol{v})$ for all t and r, as well as other variational parameters including $\{\tau_j^1, \tau_j^2\}_{1 \le j \le M}$ and $\{m_{tj}\}_{1 \le t \le n, 1 \le j \le M}$ for India buffet process $\boldsymbol{Z}$, $\{\eta_{tj}, \sigma_{tj}^A\}_{1 \le t \le n, 1 \le j \le M}$ for loading weight matrix $\boldsymbol{A}$. We also update the idiosyncratic noise scale σ_ϵ and the parameters in the spatial kernel $\kappa_{\mathcal{U}}(\cdot, \cdot)$. In this step, the gradient of ELBO is accelerated by sampling $X_{tr}(\cdot)$ independently according to Theorem 3 and the analytical expression for the KL divergence in equation (9).

Second, conditional on a sample of $\boldsymbol{X}_r(\cdot)$, we update the trainable parameters in sequential deep learning framework H that constructs the temporal kernels $\kappa_{\mathcal{X}}(\cdot, \cdot)$, via the gradient of ELBO with respect to $\boldsymbol{\Sigma}_{\mathcal{X}}$. Although any mapping function F can be used in our model, it is natural to choose $F\left(\boldsymbol{X}_t(\cdot)\right) = \boldsymbol{X}_t(\boldsymbol{v})$, which eliminates the need to sample $\boldsymbol{X}_r(\cdot)$ when computing the gradient. This is inspired by the fact that the variational distribution of inducing variables can be regarded as sufficient statistics of the Gaussian processes (Titsias, 2009).

Once we have observed the data at time n, we use the trained model to generate a posterior distribution that captures our updated understanding of the underlying patterns in the data. Based on this distribution, we make a prediction for the value of the data at the next time step, $n+1$. We present the one-step ahead prediction as:

$$\begin{aligned} \bar{\boldsymbol{Y}}_{n+1}(\boldsymbol{u}) &= (\bar{\boldsymbol{Z}} \odot \bar{\boldsymbol{A}}) \bar{\boldsymbol{X}}_{n+1}(\boldsymbol{u}), \\ \bar{X}_{n+1,r}(\boldsymbol{u}) &= \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1} \boldsymbol{\mu}_r \boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \boldsymbol{\Sigma}_{\mathcal{X}}^{n+1,1:n\,T}, \end{aligned} \tag{12}$$

where $\bar{\boldsymbol{Y}}$ and $\bar{\boldsymbol{X}}$ represent the predictive means for the observations and factors, respectively. The terms $\bar{\boldsymbol{Z}}$ and $\bar{\boldsymbol{A}}$ are the posterior means of $\boldsymbol{Z}$ and $\boldsymbol{A}$, respectively. The component $\boldsymbol{\Sigma}_{\mathcal{X}}^{n+1,1:n}$ is a $1 \times n$ matrix given by $\boldsymbol{\Sigma}_{\mathcal{X}}^{n+1,1:n} = \big[\kappa_{\mathcal{X}}(\mathcal{X}_{n+1}, \mathcal{X}_0), \cdots, \kappa_{\mathcal{X}}(\mathcal{X}_{n+1}, \mathcal{X}_n)\big]$. See Appendix D.6 for the derivations. By repeating this process iteratively, we can generate a sequence of predictions for future time steps, thereby forecasting the behavior of the system over time.

5. Experiments

5.1. Datasets

We apply DF^{2}M to four real-world datasets consisting of high-dimensional functional time series. ***Japanese Mortality*** dataset contains age-specific mortality rates for 47 Japanese prefectures (p=47) from 1975 to 2017, with 43 observations per prefecture (n=43). ***Energy Consumption*** dataset includes half-hourly measured energy consumption curves for selected London households (p=40) between December 2012 and January 2013 (n=55). ***Global Mortality*** dataset provides a broader perspective on mortality rates by including age-specific mortality data across different countries (p=32) from 1960 to 2010 (n=50). ***Stock Intraday*** dataset comprises high-frequency price observations for the S&P 100 component stocks (we removed 2 stocks with missing values, so p=98) in 2017. The data includes 45 trading days (n=45), with ten-minute resolution prices and cumulative intraday return trajectories (Horváth et al., 2014). Each dataset is preprocessed and transformed into an appropriate format for analysis. See Appendix F for the details. We denote the data as $\big\{Y_{tj}(u_k)\big\}_{1\le t\le n, 1\le j\le p, 1\le k\le K}$, where K is the number of observations per curve. Examples of functional time series for a randomly selected j are plotted in Row (1) of Figure 2.

Our work is centered on functional time series, distinct from univariate or multivariate time series. It is important to highlight that functional time series analysis is inherently more complex due to the observations being infinite-dimensional functional objects, making it much more challenging than non-functional time series (Ramsay & Silverman, 2005). The choice of datasets with time steps n deliberately limited to less than 50 is a standard practice in functional time series literature, suitable for demonstrating the robustness of our method in more challenging scenarios with limited data, in line with Chang et al. (2023a) and Tang et al. (2022).

5.2. Experiment Setup and Metrics

To assess the predictive accuracy of the proposed model, we split the data into a training set with the first n_1 periods and a test set with the last n_2 periods. We use the training set to train the parameters in the model following the steps in Section 4. Then for an integer $h > 0$, we make the h-step-ahead prediction given the fitted model using the first n_1 periods. We then repeat this process by moving the training window by one period, refitting the model, and making the h-step-ahead prediction. We compute the mean absolute prediction error (MAPE) and mean squared prediction error (MSPE) using the following equations:

$$\text{MAPE}(h) = \frac{1}{M}\sum_{j=1}^{p}\sum_{k=1}^{K}\sum_{t=n_1+h}^{n}\big|\hat{Y}_{tj}(u_k) - Y_{tj}(u_k)\big|,$$

$$\text{MSPE}(h) = \frac{1}{M}\sum_{j=1}^{p}\sum_{k=1}^{K}\sum_{t=n_1+h}^{n}\big[\hat{Y}_{tj}(u_k) - Y_{tj}(u_k)\big]^2,$$

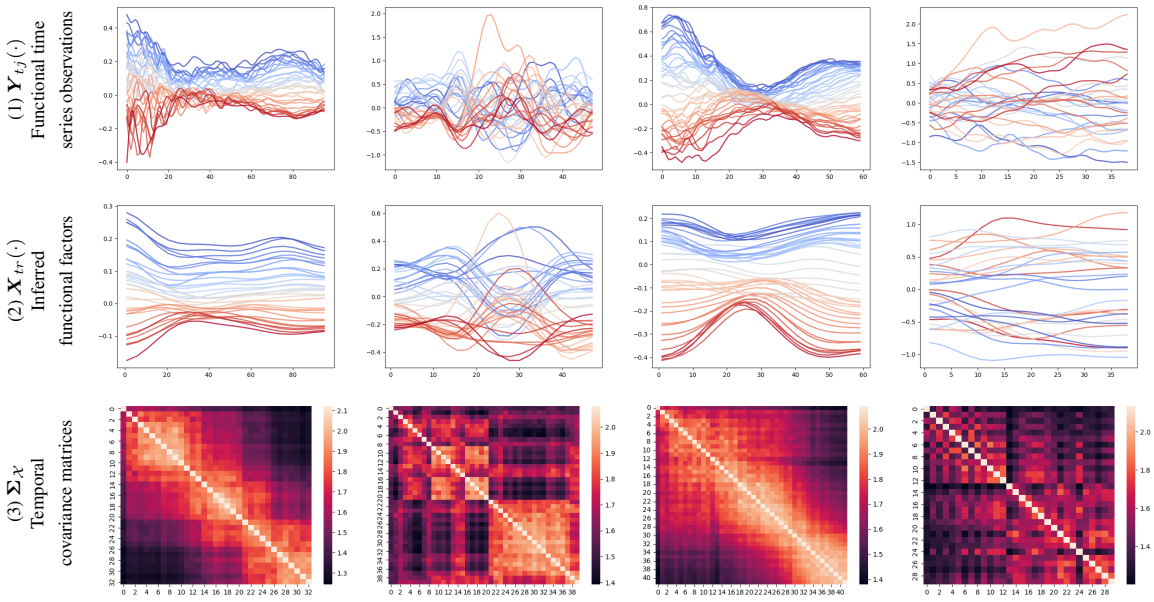
where $M = Kp(n_2 - h + 1)$. In our DF^{2}M implementation, we incorporate three cutting-edge deep learning modules: LSTM, GRU, and the self-attention mechanism. In the deep learning modules, we employ a feedforward neural network equipped with ReLU activation functions to map inputs into a designated hidden layer size. Subsequently, the transformed inputs are channeled through a time-invariant full connected neural network, LSTM, GRU, or self-attention mechanisms, denoted as DF^{2}M-LIN, DF^{2}M-LSTM, DF^{2}M-GRU and DF^{2}M-ATTN, respectively. For DF^{2}M, the outputs of the deep learning modules are passed to the kernel function, while in conventional deep learning, they are converted to outputs via a linear transformation. We evaluate their performance against conventional deep learning models under the same structural setting and regulations. The optimal hyperparameters, along with a detailed description of the deep learning architecture, can be found in Appendix G.

5.3. Empirical Results

Our primary objective is to improve the explainability of deep learning models like RNNs or transformers, while maintaining or enhancing prediction accuracy.

Explainability Firstly, Row (2) of Figure 2 shows the temporal dynamic of the largest factors in the fitted models. We can observe a decreasing trend over time for the first three datasets. This is particularly valuable as these factors exhibit a clear and smooth dynamic, which can be used to explain the underlying reasons for changes over time and also to make robust predictions. Secondly, the temporal covariance matrix ($\boldsymbol{\Sigma}_{\mathcal{X}}$) can be seen in Row (3) of Figure 2. It is evident that the first three datasets exhibit stronger autocorrelation than the *Stock Intraday* dataset, which aligns with the intuition that financial data is generally noisier and characterized by short-term dependencies.

Furthermore, both mortality datasets display a strong autoregressive pattern, as evidenced by the large covariance values close to the diagonal. They also show a blockwise pattern, which indicates the existence of change points in 1980s. Another interesting observation is the periodic pattern in the



(a) *Japanese mortality* (b) *Energy consumption* (c) *Global mortality* (d) *Stock Intraday*

Figure 2. A visualization of real datasets with analysis. Row (1): raw functional time series. Row (2): the largest functional factor. Row (3): temporal covariance matrix. Rows (1) and (2) use a blue-to-red gradient to denote time progression. Blue for older and red for recent data. Row (3) employs brightness variations to represent covariance, with brighter areas indicating higher covariance.

Energy Consumption dataset, which reveals distinct patterns for weekdays and weekends during the first 21 days. This data corresponds to the first 21 days in December. In contrast, the second half of the time steps do not exhibit this pattern. This could be attributed to the Christmas holidays in London, during which the differences between weekdays and weekends are smaller, as people are on holiday.

Predictive Accuracy Compared to standard deep learning models, the DF^{2}M framework consistently outperforms other models in terms of both MSPE and MAPE across all four datasets. The only exception to this is the *Stock Intraday* dataset, where DF^{2}M-ATTN and ATTN achieve similar levels of accuracy. Specifically, the DF^{2}M-LSTM model performs exceptionally well on the *Energy Consumption* and *Global Mortality* datasets, while the DF^{2}M-ATTN model exhibits the lowest prediction error for the *Japanese Mortality* dataset. These results demonstrate that the integration of an explainable structure with the nonlinearity of LSTM and attention mechanisms can significantly improve the overall performance of the model.

On the other hand, the DF^{2}M-LIN model outperforms both DF^{2}M-LSTM and DF^{2}M-GRU on the *Stock Intraday* dataset. This can be attributed to the fact that, in the context of financial data, long-term dependencies may not be present, rendering the Markovian model more suitable for capturing the underlying dynamics. Consequently, the DF^{2}M-LIN model emerges as a better choice for the *Stock Intraday* dataset. Compared to standard deep learning models with multiple layers, DF^{2}M achieves better or comparable results, as shown in Appendix G. However, in such cases, standard deep learning models sacrifice explainability due to their utilization of a large number of layers.

6. Related Works

In the literature concerning frequentist statistical methods for high-dimensional functional time series, various approaches have been employed. For instance, principal components based dimension reduction (Guo & Qiao, 2023; Chang et al., 2023a), factor model (Guo et al., 2021) and segmentation transformation (Chang et al., 2023b). However, all these methods use either vector autoregressive (VAR) or functional VAR to describe the temporal dynamics, implying linear and Markovian models. In contrast, our work is the first to propose a Bayesian model for high-dimensional functional time series that can handle nonlinear and non-Markovian dynamics.

Moreover, several studies (Wilson et al., 2016; Al-Shedivat et al., 2017; Xue et al., 2019; Li et al., 2019; Watson et al., 2021; Fortuin, 2022) have utilized deep kernels in Gaussian processes for classification or regression tasks. In contrast, our framework introduces the use of a deep kernel specifically designed for time series prediction.

Furthermore, prior work (Lawrence, 2003; Wang et al.,

Table 1. Comparision of DF^{2}M to Standard Deep Learning Models. For formatting reasons, MAPEs are multiplied by 10, and MSPEs are multiplied by 10^2, except for the *Energy Consumption* dataset.

(a) Comparison of DF^{2}M-LIN and LIN

		Japanese Mortality			Energy Consumption			Global Mortality			Stock Intraday		
h		1	2	3	1	2	3	1	2	3	1	2	3
DF^{2}M-LIN	MSPE	**4.707**	**4.567**	**5.623**	**10.29**	**17.58**	**17.64**	**10.78**	**9.300**	**9.706**	**99.58**	**101.2**	**89.82**
	MAPE	**1.539**	**1.446**	**1.635**	**2.334**	**3.060**	**3.100**	**2.319**	**2.041**	**2.106**	**6.424**	**6.505**	**6.269**
LIN	MSPE	7.808	8.774	9.228	16.16	18.95	20.27	16.84	18.05	19.93	137.5	127.8	139.1
	MAPE	2.092	2.227	2.313	2.939	3.214	3.342	2.783	2.949	3.174	7.896	7.491	7.924

(b) Comparison of DF^{2}M-LSTM and LSTM

		Japanese Mortality			Energy Consumption			Global Mortality			Stock Intraday		
h		1	2	3	1	2	3	1	2	3	1	2	3
DF^{2}M-LSTM	MSPE	**3.753**	**4.164**	**4.513**	**8.928**	**11.60**	**17.26**	**7.672**	**8.088**	**8.954**	**107.5**	**118.8**	**113.6**
	MAPE	**1.205**	**1.322**	**1.427**	**2.176**	**2.478**	**3.063**	**1.726**	**1.823**	**1.978**	**6.741**	**7.141**	**7.294**
LSTM	MSPE	4.989	5.597	6.501	13.51	19.71	24.61	13.28	16.29	17.08	193.3	176.0	213.8
	MAPE	1.447	1.523	1.684	2.635	3.278	3.759	2.332	2.572	2.680	9.281	9.283	10.20

(c) Comparison of DF^{2}M-GRU and GRU

		Japanese Mortality			Energy Consumption			Global Mortality			Stock Intraday		
h		1	2	3	1	2	3	1	2	3	1	2	3
DF^{2}M-GRU	MSPE	**4.092**	**4.395**	**4.898**	**9.132**	**8.714**	**9.730**	**8.741**	**8.714**	**9.730**	**102.5**	**117.3**	**95.49**
	MAPE	**1.318**	**1.402**	**1.537**	**2.204**	**1.951**	**2.110**	**1.967**	**1.951**	**2.110**	**6.675**	**7.339**	**6.649**
GRU	MSPE	8.800	8.552	10.41	15.55	24.02	17.53	14.12	15.33	17.53	414.0	445.9	427.2
	MAPE	1.691	1.809	1.865	2.872	3.518	2.597	2.211	2.403	2.597	14.12	14.66	14.07

(d) Comparison of DF^{2}M-ATTN and ATTN

		Japanese Mortality			Energy Consumption			Global Mortality			Stock Intraday		
h		1	2	3	1	2	3	1	2	3	1	2	3
DF^{2}M-ATTN	MSPE	**3.608**	**3.839**	**3.985**	**14.22**	18.70	19.03	**14.22**	**18.70**	**19.03**	104.2	103.4	93.93
	MAPE	**1.119**	**1.203**	**1.264**	**2.741**	**3.141**	**3.163**	**2.741**	**3.141**	**3.163**	6.695	6.646	6.427
ATTN	MSPE	13.44	14.85	16.17	17.03	**17.79**	**18.24**	39.52	41.83	43.95	**103.4**	**98.39**	**91.21**
	MAPE	3.166	3.363	3.546	3.130	3.216	3.268	5.332	5.506	5.643	**6.579**	**6.392**	**6.257**

2005; Titsias & Lawrence, 2010) has employed MTGPs to model cross-sectional correlations among static data. In contrast, we propose the use of a factor model to describe cross-sectional relationships, where the temporal kernel is constructed based on the features of the historical functional factors. This unique structure represents a novel contribution to the current literature.

7. Conclusion

In this paper, we present DF^{2}M, a novel deep Bayesian nonparametric approach for discovering non-Markovian and nonlinear dynamics in high-dimensional functional time series. DF^{2}M combines the strengths of the Indian buffet process, factor model, Gaussian process, and deep neural networks to offer a flexible and powerful framework. Our model effectively captures non-Markovian and nonlinear dynamics while using deep learning in a structured and explainable manner. It bridges modern deep learning and statistical time series. We also propose a computationally efficient inference algorithm.

Empirical results show the superior predictive performance of DF^{2}M compared to corresponding standard deep learning models. However, a potential limitation of our study is its reliance on simple spatial kernels, thereby neglecting to account for the intricate relationships within the observation space. We leave this as an area for future research.

Acknowledgements

This paper was prepared for informational purposes by the CDAO group of JPMorgan Chase & Co and its affiliates ("J.P. Morgan") and is not a product of the Research Department of J.P. Morgan. J.P. Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, finan-

cial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There is no societal consequence of our work.

References

Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z., and Xing, E. P. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research*, 18 (1):2850–2886, 2017.

Bathia, N., Yao, Q., and Ziegelmann, F. Identifying the finite dimensionality of curve time series. *The Annals of Statistics*, 38:3352–3386, 2010.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: a review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Bonilla, E. V., Chai, K., and Williams, C. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

Chang, J., Chen, C., Qiao, X., and Yao, Q. An autocovariance-based learning framework for high-dimensional functional time series. *Journal of Econometrics*, 2023a.

Chang, J., Fang, Q., Qiao, X., and Yao, Q. On the modelling and prediction of high-dimensional functional time series. *Working Paper*, 2023b.

Chen, C., Guo, S., and Qiao, X. Functional linear regression: dependence and error contamination. *Journal of Business and Economic Statistics*, 40:444–457, 2022.

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

Damianou, A. and Lawrence, N. D. Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 207–215, 2013.

Dawid, A. P. Some matrix-variate distribution theory: notational considerations and a Bayesian application. *Biometrika*, 68(1):265–274, 1981.

Fang, Q., Guo, S., and Qiao, X. Finite sample theory for high-dimensional functional/scalar time series with applications. *Electronic Journal of Statistics*, 16:527–591, 2022.

Fortuin, V. Priors in Bayesian deep learning: a review. *International Statistical Review*, pp. 12502, 2022.

Gao, Y., Shang, H. L., and Yang, Y. High-dimensional functional time series forecasting: An application to age-specific mortality rates. *Journal of Multivariate Analysis*, 170:232–243, 2019.

Griffiths, T. L. and Ghahramani, Z. The Indian buffet process: an introduction and review. *Journal of Machine Learning Research*, 12(32):1185–1224, 2011.

Guo, S. and Qiao, X. On consistency and sparsity for high-dimensional functional time series with application to autoregressions. *Bernoulli*, 29(1):451–472, 2023.

Guo, S., Qiao, X., and Wang, Q. Factor modelling for high-dimensional functional time series. *arXiv:2112.13651*, 2021.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M. S. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

Hamelijnck, O., Wilkinson, W., Loppi, N., Solin, A., and Damoulas, T. Spatio-temporal variational Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 34, pp. 23621–23633, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, June 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. Publisher: MIT press.

Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *The annals of statistics*, 36(3): 1171–1220, 2008.

Hormann, S., Kidzinski, L., and Hallin, M. Dynamic functional principal components. *Journal of the Royal Statistical Society: Series B*, 77:319–348, 2015.

Horváth, L., Kokoszka, P., and Rice, G. Testing stationarity of functional time series. *Journal of Econometrics*, 179 (1):66–82, 2014.

Hughes, M. C. and Sudderth, E. Memoized online variational inference for Dirichlet process mixture models. In *Advances in Neural Information Processing Systems 26*, pp. 1133–1141, 2013.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. Automatic differentiation variational inference. *Journal of machine learning research*, 2017.

Lawrence, N. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in Neural Information Processing Systems*, 16, 2003.

Li, W., Sutherland, D. J., Strathmann, H., and Gretton, A. Learning deep kernels for exponential family densities. In *International Conference on Machine Learning*, pp. 6737–6746, 2019.

Lim, B. and Zohren, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021. Publisher: The Royal Society Publishing.

Liu, Y., Qiao, X., and Lam, J. CATVI: Conditional and adaptively truncated variational inference for hierarchical bayesian nonparametric models. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pp. 3647–3662. PMLR, 2022.

Liu, Y., Qiao, X., Wang, L., and Lam, J. EEGNN: Edge enhanced graph neural network with a Bayesian nonparametric graph model. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pp. 2132–2146. PMLR, 2023.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Moreno-Muñoz, P., Artés, A., and Álvarez, M. Heterogeneous multi-output gaussian process prediction. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Ramsay, J. O. and Silverman, B. W. *Functional data analysis*. Springer, New York, 2005.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Tang, C., Shang, H. L., and Yang, Y. Clustering and forecasting multiple functional time series. *The Annals of Applied Statistics*, 16(4):2523–2553, December 2022.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 567–574, April 2009. ISSN: 1938-7228.

Titsias, M. and Lawrence, N. D. Bayesian Gaussian process latent variable model. In *Proceedings of the 13th international conference on artificial intelligence and statistics*, pp. 844–851, 2010.

Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N., and Fox, E. A. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Vu, V. Q. and Lei, J. Minimax sparse principal subspace estimation in high dimensions. *The Annals of Statistics*, 41(6):2905–2947, 2013.

Wang, J., Hertzmann, A., and Fleet, D. J. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems*, volume 18, 2005.

Watson, J., Lin, J. A., Klink, P., Pajarinen, J., and Peters, J. Latent derivative Bayesian last layer networks. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pp. 1198–1206, 2021.

Williams, C. K. and Rasmussen, C. E. *Gaussian Processes for Machine Learning*. MIT Press Cambridge, 2006.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 370–378, 2016.

Xue, H., Wu, Z.-F., and Sun, W.-X. Deep spectral kernel learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4019–4025, 2019.

Yao, J., Mueller, J., and Wang, J.-L. Deep learning for functional data analysis with adaptive basis layers. In *International Conference on Machine Learning*, pp. 11898–11908. PMLR, 2021.

Zhou, Z. and Dette, H. Statistical inference for high-dimensional panel functional time series. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 85(2):523–549, 2023.

Supplementary Material to "Deep Functional Factor Models: Forecasting High-Dimensional Functional Time Series via Bayesian Nonparametric Factorization"

This supplementary material contains a short review of sequential deep learning modules in Appendix A, a description of multi-task Gaussian process and its connection to matrix normal distribution in Appendix B, the functional version of Gaussian process dynamical model in Appendix C, technical derivations and proofs in Appendix D, the algorithm of inference in Appendix E, datasets and their preprocessing in Appendix F, deep learning structures and hyperparameters in training in Appendix G, finally the standard deviation of the results in Appendix H.

A. An Introduction for Sequential Deep Learning Modules

A.1. Recurrent Neural Networks

LSTM and GRU are both types of Recurrent Neural Networks (RNNs). They are designed to address the problem of vanishing gradients of vanilla RNNs and to preserve long-term dependencies in the sequential data.

LSTM, proposed by Hochreiter & Schmidhuber (1997), is composed of memory cells and gates that control the flow of information into and out of the memory cells. The standard structure of LSTM is composed of three types of gates: input gate, output gate and forget gate. The input gate controls the flow of new information into the memory cell, the output gate controls the flow of information out of the memory cell, and the forget gate controls the information that is removed from the memory cell. The standard structure of LSTM is defined as follows,

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f) \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i) \\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}_C[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_c) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\tilde{\boldsymbol{c}}_t),
\end{aligned}
$$

where $[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t]$ is the stack of hidden state vector $\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$. $\boldsymbol{f}_t$, $\boldsymbol{i}_t$, and $\boldsymbol{o}_t$ are the activation vectors for forget gate, update gate, and output gate, respectively. $\tilde{\boldsymbol{c}}_t$ is cell input activation vector, and c_t is cell state vector, σ denotes sigmoid function. $\boldsymbol{W}$s and $\boldsymbol{b}$s refer to weight matrices and bias vectors to be estimated.

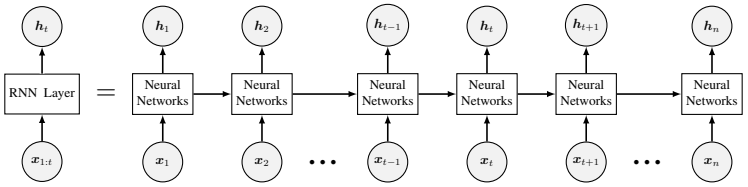
GRU is a simplified version of LSTM. It has two gates: update gate and reset gate. The update gate controls the flow of new information into the memory cell, while the reset gate controls the flow of information out of the memory cell. The structure of GRU is defined as follows,

$$
\begin{aligned}
\boldsymbol{z}_t &= \sigma(\boldsymbol{W}_z[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_z) \\
\boldsymbol{r}_t &= \sigma(\boldsymbol{W}_r[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r) \\
\tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{W}_h[\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_h) \\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t,
\end{aligned}
$$

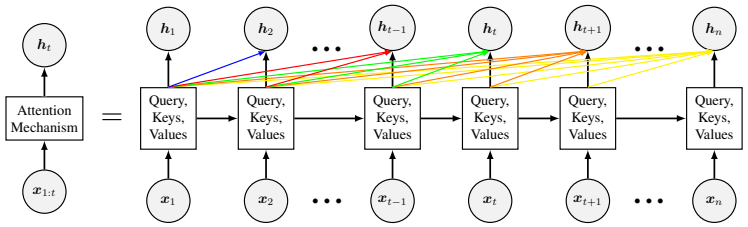
where $\boldsymbol{z}_t$ and $\boldsymbol{r}_t$ are the activation vectors for update gate and reset gate, respectively, and $\tilde{\boldsymbol{h}}_t$ is cell input activation vector.

A.2. Attention Mechanism

Attention mechanism is a deep learning model that is especially effective for sequential data prediction. It allows the model to assign different weights to different parts of the input, rather than treating them all equally. This can improve the model's ability to make predictions by allowing it to focus on the most relevant parts of the input. The commonly used self-attention



(a) RNN Layer



(b) Attention Mechanism

Figure A.1. The structures for sequential deep learning modules. In the attention mechanism, the colored links demonstrate that the current state relies exclusively on past states, ensuring that the model considers historical information without incorporating future data.

mechanism computes a weight for each element of the input, and the final output is a weighted sum of the input elements, where the weights are computed based on a query, a set of key-value pairs and a similarity function such as dot-product or MLP. The structure of standard self-attention mechanism is shown as follows,

$$\boldsymbol{q}_t = \boldsymbol{x}_t \boldsymbol{W}_Q, \quad \boldsymbol{k}_t = \boldsymbol{x}_t \boldsymbol{W}_K, \quad \boldsymbol{v}_t = \boldsymbol{x}_t \boldsymbol{W}_V$$
$$a_{t,s} = \frac{\exp(\boldsymbol{q}_t \cdot \boldsymbol{k}_s^\top / \sqrt{d_k})}{\sum_{i=1}^{T} \exp(\boldsymbol{q}_t \cdot \boldsymbol{k}_i^\top / \sqrt{d_k})}, \quad \boldsymbol{h}_t = \sum_{s=1}^{T} a_{t,s} \boldsymbol{v}_s,$$

where $\boldsymbol{q}_t$, $\boldsymbol{k}_t$, and $\boldsymbol{v}_t$ are query, key, and value vectors at time step t, respectively, $a_{t,s}$ is the attention value between time steps t and s, d_k is the dimension of the key vector, and T is the total number of time steps in the sequence.

In particular, for time series modeling, the attention value should only depend on historical information rather than future information. Therefore, the attention value should be revised as

$$a_{t,s} = \frac{\exp(\boldsymbol{q}_t \cdot \boldsymbol{k}_s^\top / \sqrt{d_k}) \mathbb{1}_{s \le t}}{\sum_{i=1}^{T} \exp(\boldsymbol{q}_t \cdot \boldsymbol{k}_i^\top / \sqrt{d_k}) \mathbb{1}_{s \le t}},$$

We illustrate the differences between RNN and attention mechanisms in Figure A.1. In RNNs, the current state depends on the most recent state, implying a sequential dependence on past states. By contrast, attention mechanisms allow the current state to depend directly on all past states, providing a more flexible and potentially more expressive way to capture the relationships between past and current states in the time series.

B. Multi-task Gaussian Process and Matrix Normal Distribution

We first provide a brief introduction of matrix normal distribution. A random matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ is said to have a matrix normal distribution, denoted as $\boldsymbol{M} \sim \text{MatrixNormal}_{m \times n}(\boldsymbol{M}_0, \boldsymbol{U}, \boldsymbol{V})$, if its probability density function is given by

$$p(\boldsymbol{M}) = \frac{\exp\left(-\frac{1}{2} \operatorname{trace}\left[\boldsymbol{V}^{-1}(\boldsymbol{M} - \boldsymbol{M}_0)^\top \boldsymbol{U}^{-1}(\boldsymbol{M} - \boldsymbol{M}_0)\right]\right)}{(2\pi)^{\frac{mn}{2}} |\boldsymbol{U}|^{\frac{n}{2}} |\boldsymbol{V}|^{\frac{m}{2}}},$$

where $\boldsymbol{M}_0 \in \mathbb{R}^{m \times n}$ is the mean matrix, $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ is a positive definite row covariance matrix, and $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ is a positive definite column covariance matrix. Moreover, the distribution of $\text{vec}(\boldsymbol{M})$ is given by

$$\text{vec}(\boldsymbol{M}) \sim \mathcal{N}_{mn}\left(\text{vec}(\boldsymbol{M}_0), \boldsymbol{V} \otimes \boldsymbol{U}\right),$$

where $\mathcal{N}_{mn}(\cdot, \cdot)$ represents a multivariate normal distribution with dimension mn. Here, $\text{vec}(\boldsymbol{M}_0)$ is the mean vector, and the covariance matrix is formed by the Kronecker product of the row covariance matrix $\boldsymbol{V}$ and the column covariance matrix $\boldsymbol{U}$.

For any $u_1, \ldots, u_L \in \mathcal{U}$, given equation (2), we have $\operatorname{vec}\big(\boldsymbol{X}_r(u_1, \ldots, u_L)\big) \sim \text{Normal}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}^u_{\mathcal{U}})$,

where

$$\boldsymbol{X}_r(u_1, \ldots, u_L) = \begin{bmatrix} X_{1r}(u_1) & \cdots & X_{nr}(u_1) \\ \cdots & \cdots & \cdots\cdots \\ X_{1r}(u_L) & \cdots & X_{nr}(u_L) \end{bmatrix},$$

$$\boldsymbol{\Sigma}_{\mathcal{X}} = \begin{bmatrix} \kappa_{\mathcal{X}}(\mathcal{X}_0, \mathcal{X}_0) & \cdots & \kappa_{\mathcal{X}}(\mathcal{X}_0, \mathcal{X}_{n-1}) \\ \cdots & \cdots & \cdots\cdots \\ \kappa_{\mathcal{X}}(\mathcal{X}_{n-1}, \mathcal{X}_0) & \cdots & \kappa_{\mathcal{X}}(\mathcal{X}_{n-1}, \mathcal{X}_{n-1}) \end{bmatrix}, \quad \text{and} \quad \boldsymbol{\Sigma}^u_{\mathcal{U}} = \begin{bmatrix} \kappa_{\mathcal{U}}(u_1, u_1) & \cdots & \kappa_{\mathcal{U}}(u_1, u_L) \\ \cdots & \cdots & \cdots\cdots \\ \kappa_{\mathcal{U}}(u_L, u_1) & \cdots & \kappa_{\mathcal{U}}(u_L, u_L) \end{bmatrix}.$$

Therefore, $\boldsymbol{X}_r(u_1, \ldots, u_L) \sim \text{MatrixNormal}(\mathbf{0}, \boldsymbol{\Sigma}^u_{\mathcal{U}}, \boldsymbol{\Sigma}_{\mathcal{X}})$.

C. Functional Version of Gaussian Process Dynamical Model

Following Wang et al. (2005), we consider a nonlinear function g with respect to historical information, achieved by a linear combination of nonlinear kernel function ϕ_is,

$$\boldsymbol{X}_t(\cdot) = g(\mathcal{X}_{t-1}) = \sum_i \phi_i(\mathcal{X}_{t-1}) \boldsymbol{a}_i(\cdot), \tag{C.1}$$

where $\mathcal{X}_{t-1} = \{\boldsymbol{X}_{t-1}, \boldsymbol{X}_{t-2}, \ldots\}$ is the set of all historical factors till period $t-1$, ϕ_i is a nonlinear basis function with respect to $\mathcal{X}_{t-1}$, and $\boldsymbol{a}_i(\cdot)$ is a function defined on $\mathcal{U}$. Equivalently, the equation above can be presented as

$$\begin{bmatrix} X_{t1}(\cdot) \\ \cdots \\ X_{tr}(\cdot) \\ \cdots \end{bmatrix} = \sum_i \phi_i(\mathcal{X}_{t-1}) \begin{bmatrix} a_{1i}(\cdot) \\ \cdots \\ a_{ri}(\cdot) \\ \cdots \end{bmatrix},$$

where $\boldsymbol{a}_i(\cdot) = \big\{a_{1i}(\cdot), a_{2i}(\cdot), \ldots, a_{ri}(\cdot), \ldots, \big\}^T$. This functional version of dynamical system corresponds to equation (3) in Wang et al. (2005). In analogy, the specific form of $g(\cdot)$ in equation (C.1), including the numbers of kernel functions, is incidental, and therefore can be marginalized out from a Bayesian perspective. Assigning each $a_{ri}(\cdot)$ an independent Gaussian process prior with kernel $\kappa_{\mathcal{U}}$, marginalizing over g leads to equation (2), where $\kappa_{\mathcal{X}}(\mathcal{X}_{t-1}, \mathcal{X}_{s-1}) = \sum_i \big\langle \phi_i(\mathcal{X}_{t-1}), \phi_i(\mathcal{X}_{s-1}) \big\rangle$.

D. Technical Derivations and Proofs

D.1. Derivations for Equation (8)

Using the variational seting in equation (7), the ELBO in equation (6) can be written as

$$\begin{aligned}
\text{ELBO} = & \operatorname{E}_q \Big[\log p(\boldsymbol{Z} \mid \alpha) p(\boldsymbol{A} \mid \boldsymbol{\Sigma}_A) \prod_{t=1}^{n} p\big(\boldsymbol{Y}_t(\cdot) \mid \boldsymbol{X}_t(\cdot), \boldsymbol{Z}, \boldsymbol{A}\big) \prod_{r \geq 1} p\big(\boldsymbol{X}_r(\cdot) \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big) \Big] \\
& - \operatorname{E}_q \Big[\log q(\boldsymbol{Z}) q(\boldsymbol{A}) \prod_{r \geq 1} q\big(\boldsymbol{X}_r(\cdot)\big) \Big] \\
= & \operatorname{E}_q \big[\log p(\boldsymbol{Z} \mid \alpha) \big] - \operatorname{E}_q \big[\log q(\boldsymbol{Z}) \big] + \operatorname{E}_q \big[\log p(\boldsymbol{A} \mid \boldsymbol{\Sigma}_A) \big] - \operatorname{E}_q \big[\log q(\boldsymbol{A}) \big] \\
& + \sum_{t=1}^{n} \operatorname{E}_q \Big[\log p\big(\boldsymbol{Y}_t(\cdot) \mid \boldsymbol{X}_t(\cdot), \boldsymbol{Z}, \boldsymbol{A}\big) \Big] \\
& + \sum_{r \geq 1} \operatorname{E}_q \Big[\log p\big(X_{1r}(\cdot), \ldots, X_{nr}(\cdot) \mid X_{1r}(\boldsymbol{v}), \ldots, X_{nr}(\boldsymbol{v}), \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big) \\
& \qquad p\big(X_{1r}(\boldsymbol{v}), \ldots, X_{nr}(\boldsymbol{v}) \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big) \Big] \\
& - \sum_{r \geq 1} \operatorname{E}_q \Big[\log p\big(X_{1r}(\cdot), \ldots, X_{nr}(\cdot) \mid X_{1r}(\boldsymbol{v}), \ldots, X_{nr}(\boldsymbol{v}), \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big) \\
& \qquad q\big(X_{1r}(\boldsymbol{v}), \ldots, X_{nr}(\boldsymbol{v})\big) \Big].
\end{aligned}$$

Next, we cancel the same items from the equation above to get:

$$\begin{aligned}\text{ELBO} = &\,\mathrm{E}_q\left[\log p(\boldsymbol{Z} \mid \alpha)\right] - \mathrm{E}_q\left[\log q(\boldsymbol{Z})\right] + \mathrm{E}_q\left[\log p(\boldsymbol{A} \mid \boldsymbol{\Sigma}_A)\right] - \mathrm{E}_q\left[\log q(\boldsymbol{A})\right] \\ &+ \sum_{t=1}^{n} \mathrm{E}_q\left[\log p\big(\boldsymbol{Y}_t(\cdot) \mid \boldsymbol{X}_t(\cdot), \boldsymbol{Z}, \boldsymbol{A}\big)\right] \\ &+ \sum_{r\geq 1} \mathrm{E}_q\left[\log p\big(X_{1r}(\boldsymbol{v}), \ldots, X_{nr}(\boldsymbol{v}) \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}}\big) - \log q\big(X_{1r}(\boldsymbol{v}), \ldots, X_{nr}(\boldsymbol{v})\big)\right].\end{aligned}$$

Finally, equation (8) is obtained using the definition of KL divergence.

D.2. Derivations for Equation (9)

The Kullbuck–Leibler divergence between two k-dimensional multivariate Gaussian distribution $\mathcal{N}_0 = \text{Normal}(\boldsymbol{m}_0, \boldsymbol{\Sigma}_0)$ and $\mathcal{N}_1 = \text{Normal}(\boldsymbol{m}_1, \boldsymbol{\Sigma}_1)$ is defined as,

$$\text{KL}\left(\mathcal{N}_0 \parallel \mathcal{N}_1\right) = \frac{1}{2}\left(\text{trace}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right) - k + \left(\boldsymbol{m}_1 - \boldsymbol{m}_0\right)^{\mathsf{T}} \boldsymbol{\Sigma}_1^{-1}\left(\boldsymbol{m}_1 - \boldsymbol{m}_0\right) + \log\left(\frac{\det \boldsymbol{\Sigma}_1}{\det \boldsymbol{\Sigma}_0}\right)\right).$$

In our settings, $\boldsymbol{v} = (v_1, \ldots, v_K)^T$, the prior and variational distributions for $\boldsymbol{X}_r(\boldsymbol{v})$ are

$$p\Big(\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{v})\big)\Big) = \text{Normal}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv})$$

and

$$q\Big(\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{v})\big)\Big) = \text{Normal}\Big(\begin{bmatrix}\boldsymbol{\mu}_{1r} \\ \cdots \\ \boldsymbol{\mu}_{nr}\end{bmatrix}, \text{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr})\Big),$$

respectively, where

$$\boldsymbol{\Sigma}_{\mathcal{X}} = \begin{bmatrix}\kappa_{\mathcal{X}}(\mathcal{X}_0, \mathcal{X}_0) & \cdots & \kappa_{\mathcal{X}}(\mathcal{X}_0, \mathcal{X}_{n-1}) \\ \cdots & \cdots & \cdots\cdots \\ \kappa_{\mathcal{X}}(\mathcal{X}_{n-1}, \mathcal{X}_0) & \cdots & \kappa_{\mathcal{X}}(\mathcal{X}_{n-1}, \mathcal{X}_{n-1})\end{bmatrix}, \quad \boldsymbol{\Sigma}_{\mathcal{U}}^{vv} = \begin{bmatrix}\kappa_{\mathcal{U}}(v_1, v_1) & \cdots & \kappa_{\mathcal{U}}(v_1, v_K) \\ \cdots & \cdots & \cdots\cdots \\ \kappa_{\mathcal{U}}(v_K, v_1) & \cdots & \kappa_{\mathcal{U}}(v_K, v_K)\end{bmatrix}.$$

Let $\boldsymbol{m}_0 = \begin{bmatrix}\boldsymbol{\mu}_{1r} \\ \cdots \\ \boldsymbol{\mu}_{nr}\end{bmatrix}$, $\boldsymbol{m}_1 = \boldsymbol{0}$, $\boldsymbol{\Sigma}_0 = \text{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr})$ and $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv}$, we have

$$\text{trace}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) = \text{trace}\big((\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\text{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr})\big),$$

$$\det(\boldsymbol{\Sigma}_1) = |\boldsymbol{\Sigma}_{\mathcal{X}}|^M |\boldsymbol{\Sigma}_{\mathcal{U}}^{vv}|^n, \quad \det(\boldsymbol{\Sigma}_0) = \prod_{t=1}^{n} |\boldsymbol{S}_{tr}|,$$

and

$$\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right)^{\mathsf{T}} \boldsymbol{\Sigma}_1^{-1}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right) = \text{trace}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_0\boldsymbol{\mu}_0^T) = \text{trace}\big((\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\text{vec}(\boldsymbol{\mu}_r)\text{vec}(\boldsymbol{\mu}_r)^T\big).$$

Therefore,

$$\begin{aligned}2\text{KL}\big(q(\boldsymbol{v}_r) \parallel p(\boldsymbol{v}_r \mid \kappa_{\mathcal{X}}, \kappa_{\mathcal{U}})\big) = &\,\text{trace}\Big((\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\big(\boldsymbol{S}_r + \text{vec}(\boldsymbol{\mu}_r)\text{vec}(\boldsymbol{\mu}_r)^T\big)\Big) \\ &+ K\log|\boldsymbol{\Sigma}_{\mathcal{X}}| + n\log|\boldsymbol{\Sigma}_{\mathcal{U}}^{vv}| - \sum_{t=1}^{n}\log|\boldsymbol{S}_{tr}| - nK,\end{aligned}$$

where $\boldsymbol{\mu}_r = (\boldsymbol{\mu}_{1r}, \ldots, \boldsymbol{\mu}_{nr})$ and $\boldsymbol{S}_r = \text{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr})$. Moreover, to get avoid of large matrix computation, we can further simplify

$$\text{trace}\big((\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\boldsymbol{S}_r\big) = \sum_{t=1}^{n} {\Sigma_{\mathcal{X}}^{-1}}_{t,t}\text{trace}(\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{S}_{tr}),$$

where ${\Sigma_{\mathcal{X}}^{-1}}_{t,t}$ denotes the (t,t)-th entry of $\boldsymbol{\Sigma}_{\mathcal{X}}^{-1}$ and

$$\text{trace}\big((\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\text{vec}(\boldsymbol{\mu}_r)\text{vec}(\boldsymbol{\mu}_r)^T\big) = \text{vec}(\boldsymbol{\mu}_r)^T\text{vec}(\boldsymbol{\Sigma}_{\mathcal{X}}^{-1}\boldsymbol{\mu}_r\boldsymbol{\Sigma}_{\mathcal{U}}^{-1}) = \text{trace}(\boldsymbol{\mu}_r^T\boldsymbol{\Sigma}_{\mathcal{U}}^{-1}\boldsymbol{\mu}_r\boldsymbol{\Sigma}_{\mathcal{X}}^{-1}).$$

D.3. Proof for Theorem 1

For any $\boldsymbol{u} = (u_1, \ldots, u_L)^T$ with $u_1, \ldots, u_L \in \mathcal{U}$, in the prior distribution, $\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u}, \boldsymbol{v})\big)$ is also normally distributed. We first partition the spatial covariance matrix as

$$\begin{bmatrix} \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} & \boldsymbol{\Sigma}_{\mathcal{U}}^{uv} \\ \boldsymbol{\Sigma}_{\mathcal{U}}^{vu} & \boldsymbol{\Sigma}_{\mathcal{U}}^{vv} \end{bmatrix},$$

where $\boldsymbol{\Sigma}_{\mathcal{U}}^{uu}$ and $\boldsymbol{\Sigma}_{\mathcal{U}}^{vv}$ correspond to the block covariance matrix of $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively, and $\boldsymbol{\Sigma}_{\mathcal{U}}^{uv}$ is the cross term. Based on this partition, using the formula of conditional multivariate Gaussian distribution, we then have

$$\begin{aligned}
\text{E}_q\Big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big)\Big] &= \text{E}_q\Big[\text{E}_q\big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big)|\boldsymbol{X}_r(\boldsymbol{v})\big]\Big] \\
&= \text{E}_q\Big[\text{E}_p\big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big)|\boldsymbol{X}_r(\boldsymbol{v})\big]\Big] \\
&= (\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv})(\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv})^{-1}\,\text{E}_q\Big[\text{vec}(\boldsymbol{X}_r(\boldsymbol{v})\Big] \\
&= (\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv})(\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\text{vec}(\boldsymbol{\mu}_r) \\
&= (I \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})\text{vec}(\boldsymbol{\mu}_r).
\end{aligned}$$

Therefore, $\text{E}_q\Big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big)\Big] = \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\text{vec}(\boldsymbol{\mu}_r)$, which means that conditional on $\boldsymbol{\Sigma}_{\mathcal{X}}$ and $\boldsymbol{\Sigma}_{\mathcal{U}}$, the mean of variational distribution are mutually independent over factors.

D.4. Proof for Theorem 2

We first derive the variance for the variational distribution of $\boldsymbol{X}_r(\boldsymbol{u})$. Note that

$$\text{Var}_q\Big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big)\Big] = \text{Var}_q\Big[\text{E}_q\big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big) \mid \boldsymbol{X}_r(\boldsymbol{v})\big]\Big] + \text{E}_q\Big[\text{Var}_q\big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u}) \mid \boldsymbol{X}_r(\boldsymbol{v})\big)\big]\Big].$$

The first term is obviously

$$(I \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1})\,\text{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr}).$$

In an analogy of proof for Theorem 1, the second term equals to

$$\begin{aligned}
&\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - (\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv})(\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv})^{-1}(\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv})^T \\
&= \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - (\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv})(\boldsymbol{\Sigma}_{\mathcal{X}}^{-1} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{vv} - 1)(\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv})^T \\
&= \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - (I \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1})(\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv})^T \\
&= \boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - (\boldsymbol{\Sigma}_{\mathcal{X}} \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}]) \\
&= \boldsymbol{\Sigma}_{\mathcal{X}} \otimes (\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}).
\end{aligned}$$

Therefore, the variance for $\boldsymbol{X}_r(u)$ with variational distribution is,

$$\text{Var}_q\Big[\text{vec}\big(\boldsymbol{X}_r(\boldsymbol{u})\big)\Big] = (I \otimes \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{X}}^{vv\,-1})\text{diag}(\boldsymbol{S}_{1r}, \ldots, \boldsymbol{S}_{nr}) + \boldsymbol{\Sigma}_{\mathcal{X}} \otimes (\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}).$$

D.5. Proof for Theorem 3

Though the model is infinite-dimensional, the inference is conducted on a finite grid of observations. Suppose $\{\boldsymbol{Y}_t(\cdot)\}_{1\le t\le n}$ have observations at points $\boldsymbol{u}$. Conditional on $\boldsymbol{\Sigma}_{\mathcal{X}}$ and $\boldsymbol{\Sigma}_{\mathcal{U}}$, in equation (8) we have

$$\begin{aligned}
&\sum_{t=1}^{n}\text{E}_q\Big[\log p\big(\boldsymbol{Y}_t(\cdot) \mid \boldsymbol{X}_t(\cdot), \boldsymbol{Z}, \boldsymbol{A}\big)\Big] \\
&= \frac{1}{2\sigma_\epsilon^2}\,\text{E}_q\sum_{i=1}^{p}\text{trace}\Big[\big(\boldsymbol{Y}_i(\boldsymbol{u}) - \sum_r \beta_{ir}\boldsymbol{X}_r(\boldsymbol{u})\big)\big(\boldsymbol{Y}_i(\boldsymbol{u}) - \sum_r \beta_{ir}\boldsymbol{X}_r(\boldsymbol{u})\big)^T\Big] + \text{constant} \\
&= \frac{1}{2\sigma_\epsilon^2}\,\text{E}_q\sum_{i=1}^{p}\sum_{r,j}\text{trace}\Big[\beta_{ir}\beta_{il}\boldsymbol{X}_r(\boldsymbol{u})\boldsymbol{X}_l(\boldsymbol{u})^T\Big] - \frac{1}{\sigma_\epsilon^2}\,\text{E}_q\sum_{i=1}^{p}\sum_{r}\text{trace}\Big[\beta_{ir}\boldsymbol{X}_r(\boldsymbol{u})\boldsymbol{Y}_i(\boldsymbol{u})^T\Big] + \text{constant},
\end{aligned}$$

where $\beta_{ir} = (\boldsymbol{Z} \odot \boldsymbol{A})_{ir}$, $\boldsymbol{Y}_i(\boldsymbol{u}) = \begin{bmatrix} Y_{1i}(u_1) & \cdots & Y_{ni}(u_1) \\ \cdots & \cdots & \cdots\cdots \\ Y_{1i}(u_L) & \cdots & Y_{ni}(u_L) \end{bmatrix}$, and $\boldsymbol{X}_r(\boldsymbol{u}) = \begin{bmatrix} X_{1r}(u_1) & \cdots & X_{nr}(u_1) \\ \cdots & \cdots & \cdots\cdots \\ X_{1r}(u_L) & \cdots & X_{nr}(u_L) \end{bmatrix}$. Using the above construction for $\boldsymbol{X}_r(\cdot)$, we also have

$$\mathrm{E}_q\, \boldsymbol{X}_r(\boldsymbol{u})\boldsymbol{X}_l(\boldsymbol{u})^T = \begin{cases} \mathrm{E}_q\, \tilde{\boldsymbol{X}}_r^{(1)}(\boldsymbol{u})\tilde{\boldsymbol{X}}_r^{(1)}(\boldsymbol{u})^T + \mathrm{E}_q\, \tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u})\tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u})^T & r = l, \\ \mathrm{E}_q\, \tilde{\boldsymbol{X}}_r^{(1)}(\boldsymbol{u})\tilde{\boldsymbol{X}}_l^{(1)}(\boldsymbol{u})^T & \text{otherwise.} \end{cases}$$

and

$$\mathrm{E}_q\, \boldsymbol{X}_r(\boldsymbol{u})\boldsymbol{Y}_i(\boldsymbol{u})^T = \mathrm{E}_q\, \tilde{\boldsymbol{X}}_r^{(1)}(\boldsymbol{u})\boldsymbol{Y}_i(\boldsymbol{u})^T,$$

because $\mathrm{E}_q\, \tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u}) = \boldsymbol{0}$, where $\tilde{\boldsymbol{X}}_r^{(1)}(\boldsymbol{u}) = \begin{bmatrix} \tilde{X}_{1r}^{(1)}(u_1) & \cdots & \tilde{X}_{nr}^{(1)}(u_1) \\ \cdots & \cdots & \cdots\cdots \\ \tilde{X}_{1r}^{(1)}(u_L) & \cdots & \tilde{X}_{nr}^{(1)}(u_L) \end{bmatrix}$ and $\tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u}) = \begin{bmatrix} \tilde{X}_{1r}^{(2)}(u_1) & \cdots & \tilde{X}_{nr}^{(2)}(u_1) \\ \cdots & \cdots & \cdots\cdots \\ \tilde{X}_{1r}^{(2)}(u_L) & \cdots & \tilde{X}_{nr}^{(2)}(u_L) \end{bmatrix}$.

Furthermore,

$$\mathrm{E}_q \operatorname{trace}\big[\tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u})\tilde{\boldsymbol{X}}_r^{(2)}(\boldsymbol{u})^T\big] = \operatorname{trace}[\Sigma_{\mathcal{X}}]\operatorname{trace}[\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}].$$

Given the above results, we obtain that

$$\begin{aligned} &\sum_{t=1}^{n} \mathrm{E}_q \Big[\log p\big(\boldsymbol{Y}_t(\cdot) \mid \boldsymbol{X}_t(\cdot), \boldsymbol{Z}, \boldsymbol{A}\big) \Big] \\ &= \frac{1}{2\sigma_\epsilon^2} \mathrm{E}_q \sum_{i=1}^{p} \sum_{r,j} \operatorname{trace}\Big[\beta_{ir}\beta_{il}\boldsymbol{X}_r^{(1)}(\boldsymbol{u})\boldsymbol{X}_l^{(1)}(\boldsymbol{u})^T\Big] - \frac{1}{\sigma_\epsilon^2} \mathrm{E}_q \sum_{i=1}^{p} \sum_{r} \operatorname{trace}\Big[\beta_{ir}\boldsymbol{X}_r^{(1)}(\boldsymbol{u})\boldsymbol{Y}_i(\boldsymbol{u})^T\Big] \\ &\quad + \frac{1}{2\sigma_\epsilon^2} \|\boldsymbol{Z} \odot \boldsymbol{A}\|_F^2 \operatorname{trace}[\boldsymbol{\Sigma}_{\mathcal{X}}]\operatorname{trace}[\boldsymbol{\Sigma}_{\mathcal{U}}^{uu} - \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\Sigma}_{\mathcal{U}}^{uv\,T}] + \text{constant}. \end{aligned}$$

Therefore, conditional on $\boldsymbol{\Sigma}_{\mathcal{X}}$ and $\boldsymbol{\Sigma}_{\mathcal{U}}$, ELBO is irrelevant to the inter-task component $\boldsymbol{X}_r^{(2)}(\boldsymbol{u})$.

D.6. Derivations for Equation (12)

$\bar{\boldsymbol{Y}}_{t+1}(\boldsymbol{u}) = (\bar{\boldsymbol{Z}} \odot \bar{\boldsymbol{A}})\bar{\boldsymbol{X}}_{t+1}(\boldsymbol{u})$ is obvious as the variational variables are assumed to be independent.

We first compute the predictive mean for the inducing variables at time $n+1$, $\bar{X}_{n+1,r}(\boldsymbol{v})$. In an analogy to Theorem 1, as the spatial kernel and temporal kernel are separable, we have

$$\bar{X}_{n+1,r}(\boldsymbol{v})^T = \boldsymbol{\Sigma}_{\mathcal{X}}^{n+1,1:n}\boldsymbol{\Sigma}_{\mathcal{X}}^{-1}\boldsymbol{\mu}_r^T,$$

where $\boldsymbol{\Sigma}_{\mathcal{X}}^{n+1,1:n} = \big[\kappa_{\mathcal{X}}(\mathcal{X}_{n+1}, \mathcal{X}_0), \kappa_{\mathcal{X}}(\mathcal{X}_{n+1}, \mathcal{X}_1), \cdots, \kappa_{\mathcal{X}}(\mathcal{X}_{n+1}, \mathcal{X}_{n-1}), \kappa_{\mathcal{X}}(\mathcal{X}_{n+1}, \mathcal{X}_n)\big] \in \mathbb{R}^{1\times n}$. Moreover, we can predict $\bar{X}_{n+1,r}(\boldsymbol{u})$ by

$$\bar{X}_{n+1,r}(\boldsymbol{u}) = \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\bar{X}_{n+1,r}(\boldsymbol{v}) = \boldsymbol{\Sigma}_{\mathcal{U}}^{uv}\boldsymbol{\Sigma}_{\mathcal{U}}^{vv\,-1}\boldsymbol{\mu}_r\boldsymbol{\Sigma}_{\mathcal{X}}^{-1}\boldsymbol{\Sigma}_{\mathcal{X}}^{n+1,1:n\,T}$$

E. Algorithm of Inference

The steps of Bayesian inference for DF^{2}M are summarized in Algorithm 1 below.

F. Dataset and Preprocessing

Japanese Mortality dataset is available at https://www.ipss.go.jp/p-toukei/JMD/index-en.html. We use log transformation and only keep the data with ages less than 96 years

Algorithm 1 Bayesian Inference for DF^{2}M

Set up initialization of trainable parameters in deep learning models.
repeat
1. Update variational distribution parameters $\boldsymbol{\mu}_{tr}$ and $\boldsymbol{S}_{tr}$ for inducing variables $\boldsymbol{X}_{tr}(\boldsymbol{v})$, along with other variational parameters,
2. Update trainable parameters in sequential deep learning framework H using the gradient of ELBO with respect to $\boldsymbol{\Sigma}_{\mathcal{X}}$,

until the convergence of the ELBO in equation (8).

old. *Energy Consumption* dataset is available at https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households. After removing samples with too many missing values, we randomly split the data into 40 groups and take the average to alleviate the impact of randomness. *Global Mortality* dataset downloaded from http://www.mortality.org/ contains mortality data from 32 countries, we use log transformation as well and keep the data with age less than 60 years old. *Stock Intraday* dataset is obtained from the Wharton Research Data Services (WRDS) database. Link to the codes for the experiments: https://github.com/yiruiliu110/df2m.

G. Deep Learning Structures and Hyperparameters

Our deep learning model structure begins with a layer normalization process, designed to standardize the features within each individual sample in a given batch. Following this, the data is fed into a custom linear layer that implements a fully-connected layer alongside a ReLU activation function. The architecture then varies based on the specific model used, with the possibilities including a fully-connected neural network with Relu activation, LSTM, GRU, or an attention mechanism. The final component of the model is a linear layer that translates the output from the LSTM, GRU, or attention mechanism into the final predictions with the desired output size. To ensure an unbiased comparison between DF^{2}M and conventional deep learning models, we configure both to have a `hidden_size` of 15 and restrict them to a single layer. For the ATTN model, we also set it to use one head. We also run experiments using multiple layers and heads with Bayesian hyperparameter optimization and compare the results in Table F.1. Compared to standard deep learning models with multiple layers, DF^{2}M achieves better or comparable results.

Table F.1. The comparison of DF^{2}M to standard deep learning models with multiple layers. For formatting reasons, the standard deviations for MAPEs are multiplied by 10, and the standard deviations for MSPEs are multiplied by 10^2, except for *Energy Consumption* dataset.

		DF^{2}M			Standard Deep learning		
		h=1	h=2	h=3	h=1	h=2	h=3
Japanese Mortality	MSPE	3.608	3.839	3.958	3.786	4.159	4.341
	MAPE	1.119	1.203	1.264	1.180	1.288	1.367
Energy Consm.	MSPE	8.928	11.60	17.26	9.380	11.19	12.79
	MAPE	2.176	2.478	3.063	2.230	2.440	2.651
Global Mortality	MSPE	7.672	8.088	8.954	8.196	8.755	9.322
	MAPE	1.726	1.823	1.978	1.639	1.753	1.857
Stock Intraday	MSPE	99.58	101.2	89.82	100.0	95.68	88.52
	MAPE	6.424	6.505	6.269	6.450	6.283	6.162

We employ Bayesian hyperparameter optimization to tune the key hyperparameters of our model. The tuned hyperparameters are listed below. The best outcomes for *Japanese Mortality* are reached through a 3-layer LSTM model, which utilizes a dropout rate of 0.07, a learning rate of 0.0008, a weight decay coefficient of 0.0002, and a hidden size of 64. Similarly, for *Energy Consumption*, a 3-layer GRU model providing the best results employs a dropout rate of 0.08, a learning rate of 0.0004, a weight decay coefficient of 0.00009, and a hidden layer size of 64. In the case of *Global Mortality*, the best performance is achieved with a 2-layer GRU model that operates with a dropout rate of 0.33, a learning rate of 0.001, a weight decay coefficient of 0.0002, and a hidden layer size of 48. Lastly, for *Stock Intraday*, the best results are seen with a 5-layer model featuring a 3-head attention mechanism, with a dropout rate of 0.10, a learning rate of 0.0007, a weight decay coefficient of 0.0010, and a hidden layer size of 2.

H. Standard Deviation of the Results

In parallel to the computation of MAPE and MSPE, we calculate their associated standard deviations by

$$\text{MAPE-STD}(h)=\Big(\frac{1}{n_2-h}\sum_{t=n_1+h}^{n}\Big\{\sum_{j=1}^{p}\sum_{k=1}^{K}\frac{1}{Kp}\big|\hat{Y}_{tj}(u_k)-Y_{tj}(u_k)\big|-\text{MAPE}(h)\Big\}^2\Big)^{\frac{1}{2}},$$

$$\text{MSPE-STD}(h)=\Big(\frac{1}{n_2-h}\sum_{t=n_1+h}^{n}\Big\{\sum_{j=1}^{p}\sum_{k=1}^{K}\frac{1}{Kp}\big[\hat{Y}_{tj}(u_k)-Y_{tj}(u_k)\big]^2-\text{MSPE}(h)\Big\}^2\Big)^{\frac{1}{2}}.$$

The findings are presented in Table H.1 below. The results indicate that the DF^{2}M-based methods exhibit a smaller or comparable standard deviation compared to other competitors.

Table H.1. Standard deviation of DF^{2}M and Standard Deep Learning Models. For formatting reasons, the standard deviations for MAPEs are multiplied by 10, and the standard deviations for MSPEs are multiplied by 10^2, except for *Energy Consumption* dataset.

	h	DF^{2}M-LIN		LIN		DF^{2}M-LSTM		LSTM		DF^{2}M-GRU		GRU		DF^{2}M-ATTN		ATTN	
		MSPE -STD	MAPE -STD	MSPE -STD	MAPE -STD	MSPE -STD	MAPE -STD	MSPE -STD	MAPE -STD	MSPE -STD	MAPE -STD	MSPE -STD	MAPE -STD	MSPE -STD	MAPE -STD	MSPE -STD	MAPE -STD
Japanese Mortality	1	1.794	0.179	3.909	0.757	1.687	0.168	2.180	0.197	1.988	0.198	6.578	0.608	1.780	0.178	1.017	0.107
	2	1.737	0.173	4.788	0.864	1.717	0.171	2.833	0.200	2.066	0.206	5.746	0.457	1.449	0.144	1.043	0.116
	3	3.841	0.384	5.150	0.915	1.728	0.172	3.040	0.316	2.577	0.257	7.320	0.568	1.735	0.173	0.763	0.077
Energy Consm.	1	0.841	0.841	14.91	1.354	0.724	0.724	11.39	1.039	0.679	0.679	9.683	0.833	1.029	1.029	12.47	1.104
	2	1.134	1.134	15.32	1.297	0.846	0.846	11.98	0.979	1.121	1.121	18.41	1.407	1.203	1.203	12.67	1.082
	3	1.080	1.080	16.35	1.262	1.229	1.229	12.89	1.049	0.985	0.985	13.05	0.949	1.308	1.308	12.72	1.060
Global Mortality	1	3.519	0.351	14.03	1.514	0.686	0.068	3.484	0.546	1.088	0.108	4.108	0.238	1.379	0.137	1.483	0.103
	2	2.191	0.219	13.61	1.503	1.469	0.146	4.883	0.623	1.461	0.146	4.318	0.276	1.386	0.138	1.664	0.139
	3	2.580	0.258	14.03	1.466	2.676	0.267	4.803	0.640	2.365	0.236	5.747	0.269	1.386	0.138	2.602	0.217
Stock Intraday	1	20.73	2.073	99.88	2.720	21.75	2.175	117.6	2.858	18.87	1.887	291.3	4.987	18.93	1.893	77.01	2.058
	2	22.27	2.227	86.99	2.361	27.17	2.717	93.25	1.823	19.63	1.963	329.2	4.917	21.25	2.125	78.82	2.124
	3	18.80	1.880	109.2	2.989	26.59	2.659	115.7	2.614	18.85	1.885	305.7	5.071	20.78	2.078	79.17	2.184